MySQL 5.6 リファレンスマニュアル

MySQL Cluster NDB 7.3-7.4 リファレンスガイド付属

概要

これは MySQL™ リファレンスマニュアルです。ここでは、MySQL 5.6 (5.6.23 まで) と、バージョン 7.3 と 7.4 の NDB に基づいた MySQL Cluster リリース (それぞれ 5.6.22-ndb-7.3.9 と 5.6.22-ndb-7.4.4 まで) について説明します。

MySQL 5.6 の機能　このマニュアルでは、MySQL 5.6 のエディションによっては含まれていない機能について説明します。このような機能は、ご自身にライセンス付与されている MySQL 5.6 のエディションに含まれていない場合があります。MySQL 5.6 の使用している工ディションに含まれる機能に関する質問がある場合は、MySQL 5.6 ライセンス契約をご覧いただくか、Oracle セールス担当者に連絡してください。

各リリースの変更内容の詳細は、MySQL 5.6 リリースノートを参照してください。

法的情報については、法的通知を参照してください。

ドキュメント生成日: 2020-11-05 (revision: 202)

目次

序文と法的通知 xxi
1 一般情報 1
1.1 このマニュアルについて 2
1.2 表記規則および構文規則 2
1.3 MySQL データベース管理システムの概要 4
1.3.1 MySQL とは 4
1.3.2 MySQL の主な機能 5
1.3.3 MySQL の歴史 7
1.4 MySQL 5.6 の新機能 8
1.5 MySQL の情報源 18
1.5.1 MySQL メーリングリスト 18
1.5.2 MySQL フォーラムにおける MySQL コミュニティーサポート 20
1.5.3 IRC (インターネットリレーチャット) の MySQL コミュニティーサポート 20
1.5.4 MySQL Enterprise 21
1.6 質問またはバグをレポートする方法 21
1.7 MySQL の標準への準拠 24
1.7.1 標準 SQL に対する MySQL 拡張機能 26
1.7.2 MySQL と標準 SQL との違い 28
1.7.3 MySQL における制約の処理 32
1.8 クレジット 35
1.8.1 MySQL への貢献者 35
1.8.2 ドキュメント作成者および翻訳者 39
1.8.3 MySQL をサポートするパッケージ 40
1.8.4 MySQL の作成に使用されたツール 40
1.8.5 MySQL のサポータ 41
2 MySQL のインストールと更新 43
2.1 一般的なインストールガイド 45
2.1.1 MySQL Community Server がサポートしているオペレーティングシステム 45
2.1.2 インストールする MySQL のバージョンと配布の選択 45
2.1.3 MySQL の取得方法 46
2.1.4 MD5 チェックサムまたは GnuPG を用いたパッケージの完全性の確認 47
2.1.5 インストールのレイアウト 56
2.1.6 コンパイラ固有のビルドの特徴 56
2.2 一般的なバイナリを使用した MySQL の Unix/Linux へのインストール 56
2.3 Microsoft Windows に MySQL をインストールする 59
2.3.1 Microsoft Windows 上での MySQL のインストールレイアウト 61
2.3.2 インストール用パッケージの選択 61
2.3.3 MySQL Installer を使用した MySQL の Microsoft Windows へのインストール 62
2.3.4 MySQL Notifier 88
2.3.5 非インストール Zip アーカイブを使用して Microsoft Windows に MySQL をインストールする 99
2.3.6 Microsoft Windows MySQL Server インストールのトラブルシューティング 106
2.3.7 Windows 上の MySQL をアップグレードする 107
2.3.8 Windows でのインストール後の手順 108
2.4 OS X に MySQL をインストールする 109
2.4.1 OS X への MySQL のインストールに関する一般的な注記 110
2.4.2 ネイティブパッケージを使用して OS X に MySQL をインストールする 111
2.4.3 MySQL 起動デーモンのインストール 117
2.4.4 MySQL 起動アイテムのインストール 118
2.4.5 MySQL Preference Pane のインストールと使用 119
2.5 Linux に MySQL をインストールする 123
2.5.1 MySQL Yum リポジトリを使用して MySQL を Linux にインストールする 123
2.5.2 サードパーティーの MySQL 配布を MySQL Yum リポジトリを使用して置換する 127
2.5.3 MySQL APT リポジトリを使用して MySQL を Linux にインストールする 129
2.5.4 MySQL SLES リポジトリを使用して MySQL を Linux にインストールする 129
2.5.5 RPM パッケージを使用して MySQL を Linux にインストールする 129
2.5.6 オラクルからの Debian パッケージを使用して MySQL を Linux にインストールする 134
2.5.7 ネイティブソフトウェアリポジトリから MySQL を Linux にインストールする 136
2.6 Unbreakable Linux Network (ULN) を使用した MySQL のインストール 139
2.7 Solaris および OpenSolaris に MySQL をインストールする 139
2.7.1 Solaris PKG を使用して Solaris に MySQL をインストールする 140

2.7.2 IPS を使用して MySQL を OpenSolaris にインストールする 141
2.8 FreeBSD に MySQL をインストールする 142
2.9 ソースから MySQL をインストールする 143
2.9.1 ソースインストールの MySQL のレイアウト 144
2.9.2 標準ソース配布を使用して MySQL をインストールする 144
2.9.3 開発ソースツリーを使用して MySQL をインストールする 148
2.9.4 MySQL ソース構成オプション 150
2.9.5 MySQL のコンパイルに関する問題 162
2.9.6 MySQL の構成とサードパーティーツール 163
2.10 インストール後のセットアップとテスト 163
2.10.1 Unix 類似システムでのインストール後の手順 164
2.10.2 最初の MySQL アカウントのセキュリティー設定 173
2.11 MySQL のアップグレードとダウングレード 176
2.11.1 MySQL のアップグレード 177
2.11.2 MySQL のダウングレード 186
2.11.3 テーブルまたはインデックスの再構築が必要かどうかのチェック 187
2.11.4 テーブルまたはインデックスの再作成または修復 189
2.11.5 MySQL データベースのほかのマシンへのコピー 190
2.12 環境変数 191
2.13 Perl のインストールに関する注釈 193
2.13.1 Unix に Perl をインストールする 193
2.13.2 Windows に ActiveState Perl をインストールする 194
2.13.3 Perl DBI/DBD インタフェース使用の問題 194
3 チュートリアル 197
3.1 サーバーへの接続とサーバーからの切断 197
3.2 クエリーの入力 198
3.3 データベースの作成と使用 200
3.3.1 データベースの作成と選択 201
3.3.2 テーブルの作成 202
3.3.3 テーブルへのデータのロード 203
3.3.4 テーブルからの情報の取り出し 204
3.4 データベースとテーブルに関する情報の取得 215
3.5 バッチモードでの MySQL の使用 216
3.6 一般的なクエリーの例 217
3.6.1 カラムの最大値 217
3.6.2 特定のカラムの最大値が格納されている行 218
3.6.3 グループごとのカラムの最大値 218
3.6.4 特定のカラムのグループごとの最大値が格納されている行 218
3.6.5 ユーザー定義の変数の使用 219
3.6.6 外部キーの使用 219
3.6.7 2 つのキーを使用した検索 221
3.6.8 日ごとの訪問数の計算 221
3.6.9 AUTO_INCREMENT の使用 221
3.7 Apache での MySQL の使用 223
4 MySQL プログラム 225
4.1 MySQL プログラムの概要 226
4.2 MySQL プログラムの使用 230
4.2.1 MySQL プログラムの起動 230
4.2.2 MySQL サーバーへの接続 230
4.2.3 プログラムオプションの指定 233
4.2.4 コマンド行でのオプションの使用 234
4.2.5 プログラムオプション修飾子 235
4.2.6 オプションファイルの使用 236
4.2.7 オプションファイルの処理に影響するコマンド行オプション 240
4.2.8 プログラム変数の設定へのオプションの使用 241
4.2.9 オプションのデフォルト、値を想定するオプション、および = 記号 242
4.2.10 環境変数の設定 244
4.3 MySQL サーバーとサーバー起動プログラム 245
4.3.1 mysqld — MySQL サーバー 245
4.3.2 mysqld_safe — MySQL サーバー起動スクリプト 246
4.3.3 mysql.server — MySQL サーバー起動スクリプト 250
4.3.4 mysqld_multi — 複数の MySQL サーバーの管理 250
4.4 MySQL インストール関連のプログラム 254
4.4.1 comp_err — MySQL エラーメッセージファイルのコンパイル 254

4.4.2 mysqlbug — バグレポートの生成 255
4.4.3 mysql_install_db — MySQL データディレクトリの初期化 255
4.4.4 mysql_plugin — MySQL サーバープラグインの構成 258
4.4.5 mysql_secure_installation — MySQL インストールのセキュリティー改善 260
4.4.6 mysql_tzinfo_to_sql — タイムゾーンテーブルのロード 260
4.4.7 mysql_upgrade — MySQL テーブルのチェックとアップグレード 261
4.5 MySQL クライアントプログラム 266
4.5.1 mysql — MySQL コマンド行ツール 266
4.5.2 mysqladmin — MySQL サーバーの管理を行うクライアント 286
4.5.3 mysqlcheck — テーブル保守プログラム 293
4.5.4 mysqldump — データベースバックアッププログラム 300
4.5.5 mysqlimport — データインポートプログラム 316
4.5.6 mysqlshow — データベース、テーブル、およびカラム情報の表示 321
4.5.7 mysqlslap — 負荷エミュレーションクライアント 325
4.6 MySQL 管理プログラムおよびユーティリティープログラム 332
4.6.1 innochecksum — オフライン InnoDB ファイルチェックサムユーティリティー 332
4.6.2 myisam_ftdump — 全文インデックス情報の表示 333
4.6.3 myisamchk — MyISAM テーブルメンテナンスユーティリティー 334
4.6.4 myisamlog — MyISAM ログファイルの内容の表示 349
4.6.5 myisampack — 圧縮された読み取り専用の MyISAM テーブルの生成 350
4.6.6 mysql_config_editor — MySQL 構成ユーティリティー 355
4.6.7 mysqlaccess — アクセス権限をチェックするクライアント 360
4.6.8 mysqlbinlog — バイナリログファイルを処理するためのユーティリティー 362
4.6.9 mysqldumpslow — スロークエリーログファイルの要約 379
4.6.10 mysqlhotcopy — データベースバックアッププログラム 381
4.6.11 mysql_convert_table_format — 指定されたストレージエンジンを使用するためのテーブルの変換 384
4.6.12 mysql_find_rows — ファイルからの SQL ステートメントの抽出 384
4.6.13 mysql_fix_extensions — テーブルファイル名の拡張子の正規化 385
4.6.14 mysql_setpermission — 付与テーブルに許可をインタラクティブに設定 385
4.6.15 mysql_waitpid — プロセスを強制終了して終了を待機 386
4.6.16 mysql_zap — パターンに一致するプロセスを強制終了 387
4.7 MySQL プログラム開発ユーティリティー 387
4.7.1 msql2mysql — mSQL プログラムを MySQL で使用するために変換 387
4.7.2 mysql_config — クライアントのコンパイル用オプションの表示 388
4.7.3 my_print_defaults — オプションファイルからのオプションの表示 389
4.7.4 resolve_stack_dump — 数値スタックトレースダンプをシンボルに解決 390
4.8 その他のプログラム 390
4.8.1 perror — エラーコードの説明 390
4.8.2 replace — 文字列置換ユーティリティー 391
4.8.3 resolveip — ホスト名と IP アドレスの解決 392
5 MySQL サーバーの管理 393
5.1 MySQL Server 393
5.1.1 サーバーオプションおよび変数リファレンス 394
5.1.2 サーバー構成のデフォルト値 422
5.1.3 サーバーコマンドオプション 424
5.1.4 サーバーシステム変数 452
5.1.5 システム変数の使用 548
5.1.6 サーバーステータス変数 560
5.1.7 サーバー SQL モード 585
5.1.8 サーバープラグイン 593
5.1.9 IPv6 サポート 596
5.1.10 サーバー側のヘルプ 600
5.1.11 シグナルへのサーバー応答 600
5.1.12 シャットダウンプロセス 601
5.2 MySQL Server ログ 602
5.2.1 一般クエリーログおよびスロークエリーログの出力先の選択 603
5.2.2 エラーログ 605
5.2.3 一般クエリーログ 606
5.2.4 バイナリログ 607
5.2.5 スロークエリーログ 617
5.2.6 DDL ログ 618
5.2.7 サーバーログの保守 619
5.3 1 つのマシン上での複数の MySQL インスタンスの実行 620

5.3.1 複数のデータディレクトリのセットアップ 621
5.3.2 Windows 上での複数の MySQL インスタンスの実行 622
5.3.3 Unix 上での複数の MySQL インスタンスの実行 624
5.3.4 複数サーバー環境でのクライアントプログラムの使用 625
5.4 DTrace を使用した mysqld のトレース 626
5.4.1 mysqld DTrace プローブリファレンス 627
6 セキュリティー 643
6.1 一般的なセキュリティーの問題 643
6.1.1 セキュリティーガイドライン 644
6.1.2 パスワードをセキュアな状態にする 645
6.1.3 攻撃者に対する MySQL のセキュアな状態の維持 656
6.1.4 セキュリティー関連の mysqld オプションおよび変数 658
6.1.5 MySQL を通常ユーザーとして実行する方法 658
6.1.6 LOAD DATA LOCAL のセキュリティーの問題 659
6.1.7 クライアントプログラミングのセキュリティーガイドライン 660
6.2 MySQL アクセス権限システム 661
6.2.1 MySQL で提供される権限 662
6.2.2 権限システム付与テーブル 666
6.2.3 アカウント名の指定 670
6.2.4 アクセス制御、ステージ 1: 接続の検証 672
6.2.5 アクセス制御、ステージ 2: リクエストの確認 674
6.2.6 権限変更が有効化される時期 676
6.2.7 アクセス拒否エラーの原因 676
6.3 MySQL ユーザーアカウントの管理 680
6.3.1 ユーザー名とパスワード 681
6.3.2 ユーザーアカウントの追加 682
6.3.3 ユーザーアカウントの削除 685
6.3.4 アカウントリソース制限の設定 685
6.3.5 アカウントパスワードの割り当て 687
6.3.6 パスワードの期限切れとサンドボックスモード 688
6.3.7 プラガブル認証 690
6.3.8 MySQL で使用可能な認証プラグイン 693
6.3.9 プロキシユーザー 712
6.3.10 セキュアな接続のための SSL の使用 715
6.3.11 SSH を使用した Windows から MySQL へのリモート接続 725
6.3.12 MySQL Enterprise Audit ログプラグイン 726
6.3.13 SQL ベースの MySQL アカウントアクティビティーの監査 747
7 バックアップとリカバリ 751
7.1 バックアップとリカバリの種類 752
7.2 データベースバックアップ方法 754
7.3 バックアップおよびリカバリ戦略の例 756
7.3.1 バックアップポリシーの確立 757
7.3.2 リカバリへのバックアップの使用 759
7.3.3 バックアップ戦略サマリー 759
7.4 バックアップへの mysqldump の使用 759
7.4.1 mysqldump による SQL フォーマットでのデータのダンプ 760
7.4.2 SQL フォーマットバックアップのリロード 760
7.4.3 mysqldump による区切りテキストフォーマットでのデータのダンプ 761
7.4.4 区切りテキストフォーマットバックアップのリロード 762
7.4.5 mysqldump のヒント 762
7.5 バイナリログを使用したポイントインタイム (増分) リカバリ 764
7.5.1 イベント時間を使用したポイントインタイムリカバリ 765
7.5.2 イベントの位置を使用したポイントインタイムリカバリ 766
7.6 MyISAM テーブルの保守とクラッシュリカバリ 766
7.6.1 クラッシュリカバリへの myisamchk の使用 767
7.6.2 MyISAM テーブルのエラーのチェック方法 767
7.6.3 MyISAM テーブルの修復方法 768
7.6.4 MyISAM テーブルの最適化 770
7.6.5 MyISAM テーブル保守スケジュールのセットアップ 770
8 最適化 773
8.1 最適化の概要 774
8.2 SQL ステートメントの最適化 775
8.2.1 SELECT ステートメントの最適化 775
8.2.2 DML ステートメントの最適化 816

8.2.3 データベース権限の最適化 ... 817
8.2.4 INFORMATION_SCHEMA クエリーの最適化 ... 817
8.2.5 その他の最適化のヒント ... 822
8.3 最適化とインデックス ... 824
8.3.1 MySQL のインデックスの使用の仕組み ... 824
8.3.2 主キーの使用 ... 825
8.3.3 外部キーの使用 ... 825
8.3.4 カラムインデックス ... 826
8.3.5 マルチカラムインデックス ... 827
8.3.6 インデックスの使用の確認 ... 828
8.3.7 InnoDB および MyISAM インデックス統計コレクション ... 828
8.3.8 B ツリーインデックスとハッシュインデックスの比較 ... 829
8.4 データベース構造の最適化 ... 830
8.4.1 データサイズの最適化 ... 830
8.4.2 MySQL データ型の最適化 ... 832
8.4.3 多数のテーブルの最適化 ... 833
8.4.4 MySQL が内部一時テーブルを使用する仕組み ... 835
8.5 InnoDB テーブルの最適化 ... 836
8.5.1 InnoDB テーブルのストレージレイアウトの最適化 ... 836
8.5.2 InnoDB トランザクション管理の最適化 ... 836
8.5.3 InnoDB ロギングの最適化 ... 837
8.5.4 InnoDB テーブルの一括データロード ... 838
8.5.5 InnoDB クエリーの最適化 ... 839
8.5.6 InnoDB DDL 操作の最適化 ... 839
8.5.7 InnoDB ディスク I/O の最適化 ... 839
8.5.8 InnoDB 構成変数の最適化 ... 840
8.5.9 多くのテーブルのあるシステムに対する InnoDB の最適化 ... 842
8.6 MyISAM テーブルの最適化 ... 842
8.6.1 MyISAM クエリーの最適化 ... 842
8.6.2 MyISAM テーブルの一括データロード ... 843
8.6.3 REPAIR TABLE ステートメントの速度 ... 844
8.7 MEMORY テーブルの最適化 ... 845
8.8 クエリー実行プランの理解 ... 846
8.8.1 EXPLAIN によるクエリーの最適化 ... 846
8.8.2 EXPLAIN 出力フォーマット ... 846
8.8.3 EXPLAIN EXTENDED 出力フォーマット ... 857
8.8.4 クエリーパフォーマンスの推定 ... 858
8.8.5 クエリーオプティマイザの制御 ... 859
8.9 バッファリングとキャッシュ ... 861
8.9.1 InnoDB バッファープール ... 861
8.9.2 MyISAM キーキャッシュ ... 864
8.9.3 MySQL クエリーキャッシュ ... 868
8.9.4 プリペアドステートメントおよびストアドプログラムのキャッシュ ... 873
8.10 ロック操作の最適化 ... 874
8.10.1 内部ロック方法 ... 874
8.10.2 テーブルロックの問題 ... 876
8.10.3 同時挿入 ... 877
8.10.4 メタデータのロック ... 878
8.10.5 外部ロック ... 879
8.11 MySQL サーバーの最適化 ... 880
8.11.1 システム要素およびスタートアップパラメータのチューニング ... 880
8.11.2 サーバーパラメータのチューニング ... 880
8.11.3 ディスク I/O の最適化 ... 887
8.11.4 メモリーの使用の最適化 ... 891
8.11.5 ネットワークの使用の最適化 ... 894
8.11.6 スレッドプールプラグイン ... 896
8.12 パフォーマンスの測定 (ベンチマーク) ... 901
8.12.1 式と関数の速度の測定 ... 901
8.12.2 MySQL ベンチマークスイート ... 901
8.12.3 独自のベンチマークの使用 ... 902
8.12.4 performance_schema によるパフォーマンスの測定 ... 903
8.12.5 スレッド情報の検査 ... 903
9 言語構造 ... 917
9.1 リテラル値 ... 917

9.1.1 文字列リテラル 917
9.1.2 数値リテラル 919
9.1.3 日付リテラルと時間リテラル 920
9.1.4 16 進数リテラル 921
9.1.5 boolean リテラル 922
9.1.6 ビットフィールドリテラル 922
9.1.7 NULL 値 922
9.2 スキーマオブジェクト名 923
9.2.1 識別子の修飾子 924
9.2.2 識別子の大文字と小文字の区別 925
9.2.3 識別子とファイル名のマッピング 927
9.2.4 関数名の構文解析と解決 928
9.3 予約語 931
9.4 ユーザー定義変数 939
9.5 式の構文 942
9.6 コメントの構文 943
10 グローバリゼーション 945
10.1 文字セットのサポート 945
10.1.1 一般の文字セットおよび照合順序 946
10.1.2 MySQL での文字セットと照合順序 946
10.1.3 文字セットと照合順序の指定 948
10.1.4 接続文字セットおよび照合順序 954
10.1.5 アプリケーションの文字セットおよび照合順序の構成 956
10.1.6 エラーメッセージの文字セット 957
10.1.7 照合順序の問題 958
10.1.8 文字列のレパートリー 965
10.1.9 文字セットのサポートによる影響を受ける演算 967
10.1.10 Unicode のサポート 969
10.1.11 以前の Unicode サポートから現在の Unicode サポートへのアップグレード 973
10.1.12 メタデータ用の UTF-8 975
10.1.13 カラム文字セットの変換 976
10.1.14 MySQL でサポートされる文字セットと照合順序 977
10.2 エラーメッセージ言語の設定 988
10.3 文字セットの追加 989
10.3.1 文字定義配列 990
10.3.2 複雑な文字セットの文字列照合のサポート 991
10.3.3 複雑な文字セットのマルチバイト文字のサポート 991
10.4 文字セットへの照合順序の追加 992
10.4.1 照合順序の実装タイプ 993
10.4.2 照合順序 ID の選択 995
10.4.3 8 ビットの文字セットへの単純な照合順序の追加 996
10.4.4 Unicode 文字セットへの UCA 照合順序の追加 997
10.5 文字セットの構成 1002
10.6 MySQL Server でのタイムゾーンのサポート 1003
10.6.1 タイムゾーンの変更による現在の時間の維持 1005
10.6.2 タイムゾーンのうるう秒のサポート 1006
10.7 MySQL Server のロケールサポート 1007
11 データ型 1011
11.1 データ型の概要 1011
11.1.1 数値型の概要 1011
11.1.2 日付と時間型の概要 1014
11.1.3 文字列型の概要 1016
11.2 数値型 1019
11.2.1 整数型 (真数値) - INTEGER、INT、SMALLINT、TINYINT、MEDIUMINT、BIGINT 1019
11.2.2 固定小数点型 (真数値) - DECIMAL、NUMERIC 1020
11.2.3 浮動小数点型 (概数値) - FLOAT、DOUBLE 1020
11.2.4 ビット値型 - BIT 1020
11.2.5 数値型の属性 1021
11.2.6 範囲外およびオーバーフローの処理 1021
11.3 日付と時間型 1022
11.3.1 DATE、DATETIME、および TIMESTAMP 型 1023
11.3.2 TIME 型 1025
11.3.3 YEAR 型 1025
11.3.4 YEAR(2) の制限と YEAR(4) への移行 1026

11.3.5 TIMESTAMP および DATETIME の自動初期化および更新機能 1028
11.3.6 時間値での小数秒 1031
11.3.7 日付と時間型間での変換 1032
11.3.8 日付での 2 桁の年 1033
11.4 文字列型 1033
11.4.1 CHAR および VARCHAR 型 1033
11.4.2 BINARY および VARBINARY 型 1035
11.4.3 BLOB 型と TEXT 型 1036
11.4.4 ENUM 型 1037
11.4.5 SET 型 1040
11.5 空間データの拡張 1041
11.5.1 空間データ型 1043
11.5.2 OpenGIS 幾何モデル 1043
11.5.3 空間データの使用 1048
11.6 データ型デフォルト値 1054
11.7 データ型のストレージ要件 1055
11.8 カラムに適した型の選択 1058
11.9 その他のデータベースエンジンのデータ型の使用 1058
12 関数と演算子 1061
12.1 関数と演算子のリファレンス 1062
12.2 式評価での型変換 1071
12.3 演算子 1073
12.3.1 演算子の優先順位 1074
12.3.2 比較関数と演算子 1074
12.3.3 論理演算子 1079
12.3.4 割り当て演算子 1080
12.4 制御フロー関数 1081
12.5 文字列関数 1083
12.5.1 文字列比較関数 1096
12.5.2 正規表現 1099
12.6 数値関数と演算子 1103
12.6.1 算術演算子 1104
12.6.2 数学関数 1106
12.7 日付および時間関数 1113
12.8 MySQL で使用されるカレンダー 1131
12.9 全文検索関数 1132
12.9.1 自然言語全文検索 1133
12.9.2 ブール全文検索 1135
12.9.3 クエリー拡張を使用した全文検索 1140
12.9.4 全文ストップワード 1140
12.9.5 全文制限 1145
12.9.6 MySQL の全文検索の微調整 1146
12.9.7 全文インデックス作成用の照合順序の追加 1148
12.10 キャスト関数と演算子 1149
12.11 XML 関数 1152
12.12 ビット関数 1161
12.13 暗号化関数と圧縮関数 1163
12.14 情報関数 1170
12.15 空間分析関数 1179
12.15.1 空間関数のリファレンス 1179
12.15.2 空間関数による引数処理 1181
12.15.3 WKT 値から幾何値を作成する関数 1181
12.15.4 WKB 値から幾何値を作成する関数 1182
12.15.5 幾何値を作成する MySQL 固有の関数 1183
12.15.6 幾何形式変換関数 1183
12.15.7 幾何プロパティー関数 1184
12.15.8 空間演算子関数 1189
12.15.9 幾何オブジェクト間の空間関係をテストする関数 1190
12.16 グローバルトランザクション ID とともに使用される関数 1193
12.17 MySQL Enterprise Encryption の関数 1195
12.17.1 Enterprise Encryption のインストール 1195
12.17.2 Enterprise Encryption の使用法と例 1195
12.17.3 Enterprise Encryption 関数のリファレンス 1197
12.17.4 Enterprise Encryption 関数の説明 1197

12.18 その他の関数 1200
12.19 GROUP BY 句で使用される関数と修飾子 1207
12.19.1 GROUP BY (集約) 関数 1207
12.19.2 GROUP BY 修飾子 1210
12.19.3 MySQL での GROUP BY の処理 1213
12.20 高精度計算 1214
12.20.1 数値の型 1214
12.20.2 DECIMAL データ型の特性 1215
12.20.3 式の処理 1216
12.20.4 丸め動作 1217
12.20.5 高精度計算の例 1217
13 SQL ステートメントの構文 1221
13.1 データ定義ステートメント 1222
13.1.1 ALTER DATABASE 構文 1222
13.1.2 ALTER EVENT 構文 1223
13.1.3 ALTER LOGFILE GROUP 構文 1224
13.1.4 ALTER FUNCTION 構文 1225
13.1.5 ALTER PROCEDURE 構文 1226
13.1.6 ALTER SERVER 構文 1226
13.1.7 ALTER TABLE 構文 1226
13.1.8 ALTER TABLESPACE 構文 1244
13.1.9 ALTER VIEW 構文 1245
13.1.10 CREATE DATABASE 構文 1245
13.1.11 CREATE EVENT 構文 1246
13.1.12 CREATE FUNCTION 構文 1250
13.1.13 CREATE INDEX 構文 1250
13.1.14 CREATE LOGFILE GROUP 構文 1253
13.1.15 CREATE PROCEDURE および CREATE FUNCTION 構文 1254
13.1.16 CREATE SERVER 構文 1259
13.1.17 CREATE TABLE 構文 1260
13.1.18 CREATE TABLESPACE 構文 1285
13.1.19 CREATE TRIGGER 構文 1287
13.1.20 CREATE VIEW 構文 1289
13.1.21 DROP DATABASE 構文 1292
13.1.22 DROP EVENT 構文 1293
13.1.23 DROP FUNCTION 構文 1293
13.1.24 DROP INDEX 構文 1294
13.1.25 DROP LOGFILE GROUP 構文 1294
13.1.26 DROP PROCEDURE および DROP FUNCTION 構文 1295
13.1.27 DROP SERVER 構文 1295
13.1.28 DROP TABLE 構文 1295
13.1.29 DROP TABLESPACE 構文 1296
13.1.30 DROP TRIGGER 構文 1296
13.1.31 DROP VIEW 構文 1296
13.1.32 RENAME TABLE 構文 1296
13.1.33 TRUNCATE TABLE 構文 1297
13.2 データ操作ステートメント 1298
13.2.1 CALL 構文 1298
13.2.2 DELETE 構文 1300
13.2.3 DO 構文 1303
13.2.4 HANDLER 構文 1304
13.2.5 INSERT 構文 1305
13.2.6 LOAD DATA INFILE 構文 1313
13.2.7 LOAD XML 構文 1321
13.2.8 REPLACE 構文 1326
13.2.9 SELECT 構文 1328
13.2.10 サブクエリー構文 1344
13.2.11 UPDATE 構文 1353
13.3 MySQL トランザクションおよびロックステートメント 1355
13.3.1 START TRANSACTION、COMMIT、および ROLLBACK 構文 1355
13.3.2 ロールバックできないステートメント 1358
13.3.3 暗黙的なコミットを発生させるステートメント 1358
13.3.4 SAVEPOINT、ROLLBACK TO SAVEPOINT、および RELEASE SAVEPOINT 構文 1359
13.3.5 LOCK TABLES および UNLOCK TABLES 構文 1360

13.3.6 SET TRANSACTION 構文 1365
13.3.7 XA トランザクション 1367
13.4 レプリケーションステートメント 1371
13.4.1 マスターサーバーを制御するための SQL ステートメント 1371
13.4.2 スレーブサーバーを制御するための SQL ステートメント 1373
13.5 準備済みステートメントのための SQL 構文 1382
13.5.1 PREPARE 構文 1386
13.5.2 EXECUTE 構文 1386
13.5.3 DEALLOCATE PREPARE 構文 1386
13.6 MySQL 複合ステートメント構文 1387
13.6.1 BEGIN ... END 複合ステートメント構文 1387
13.6.2 ステートメントラベルの構文 1387
13.6.3 DECLARE 構文 1388
13.6.4 ストアドプログラム内の変数 1388
13.6.5 フロー制御ステートメント 1389
13.6.6 カーソル 1393
13.6.7 条件の処理 1394
13.7 データベース管理ステートメント 1414
13.7.1 アカウント管理ステートメント 1414
13.7.2 テーブル保守ステートメント 1430
13.7.3 プラグインおよびユーザー定義関数ステートメント 1438
13.7.4 SET 構文 1440
13.7.5 SHOW 構文 1443
13.7.6 その他の管理ステートメント 1480
13.8 MySQL ユーティリティーステートメント 1488
13.8.1 DESCRIBE 構文 1488
13.8.2 EXPLAIN 構文 1488
13.8.3 HELP 構文 1489
13.8.4 USE 構文 1491
14 InnoDB ストレージエンジン 1493
14.1 InnoDB 入門 1495
14.1.1 デフォルトの MySQL ストレージエンジンとしての InnoDB 1496
14.1.2 InnoDB の可用性チェック 1499
14.1.3 InnoDB の無効化 1499
14.2 InnoDB の概念とアーキテクチャー 1500
14.2.1 MySQL および ACID モデル 1500
14.2.2 InnoDB のトランザクションモデルおよびロック 1502
14.2.3 InnoDB のロックモード 1502
14.2.4 一貫性非ロック読み取り 1504
14.2.5 ロック読み取り (SELECT ... FOR UPDATE および SELECT ... LOCK IN SHARE MODE) 1506
14.2.6 InnoDB のレコード、ギャップ、およびネクストキーロック 1507
14.2.7 ネクストキーロックによるファントム問題の回避 1508
14.2.8 InnoDB のさまざまな SQL ステートメントで設定されたロック 1509
14.2.9 暗黙的なトランザクションコミットとロールバック 1512
14.2.10 デッドロックの検出とロールバック 1512
14.2.11 デッドロックの対処方法 1512
14.2.12 InnoDB マルチバージョン 1513
14.2.13 InnoDB テーブルおよびインデックスの構造 1514
14.3 InnoDB の構成 1522
14.3.1 読み取り専用操作用の InnoDB の構成 1526
14.4 InnoDB の管理 1527
14.5 InnoDB テーブルスペース管理 1527
14.5.1 InnoDB テーブルスペースの作成 1527
14.5.2 InnoDB File-Per-Table モード 1528
14.5.3 File-Per-Table モードの有効化および無効化 1530
14.5.4 テーブルスペースの位置の指定 1531
14.5.5 テーブルスペースの別のサーバーへのコピー (トランスポータブルテーブルスペース) 1532
14.5.6 個別のテーブルスペースへの InnoDB Undo ログの格納 1535
14.5.7 InnoDB ログファイルの数またはサイズの変更、および InnoDB テーブルスペースのサイズの変更 1536
14.5.8 共有テーブルスペースでの RAW ディスクパーティションの使用 1538
14.6 InnoDB テーブルの管理 1539
14.6.1 InnoDB テーブルの作成 1539
14.6.2 別のマシンへの InnoDB テーブルの移動またはコピー 1540

14.6.3 トランザクションを使用した DML 操作のグループ化 1541
14.6.4 MyISAM から InnoDB へのテーブルの変換 1542
14.6.5 InnoDB での AUTO_INCREMENT 処理 1546
14.6.6 InnoDB と FOREIGN KEY 制約 1551
14.6.7 InnoDB テーブル上の制限 1552
14.7 InnoDB 圧縮テーブル 1555
14.7.1 テーブル圧縮の概要 1555
14.7.2 テーブル圧縮の有効化 1555
14.7.3 InnoDB テーブルの圧縮の調整 1556
14.7.4 実行時の圧縮のモニタリング 1560
14.7.5 InnoDB テーブルでの圧縮の動作 1560
14.7.6 OLTP ワークロードの圧縮 1563
14.7.7 SQL 圧縮構文の警告とエラー 1564
14.8 InnoDB のファイル形式管理 1566
14.8.1 ファイル形式の有効化 1566
14.8.2 ファイル形式の互換性の確認 1566
14.8.3 使用されているファイル形式の識別 1570
14.8.4 ファイル形式のダウングレード 1571
14.8.5 将来の InnoDB ファイル形式 1571
14.9 InnoDB の行ストレージと行フォーマット 1571
14.9.1 InnoDB 行ストレージの概要 1571
14.9.2 テーブルの行フォーマットの指定 1571
14.9.3 DYNAMIC および COMPRESSED 行フォーマット 1572
14.9.4 COMPACT および REDUNDANT 行フォーマット 1572
14.10 InnoDB のディスク I/O とファイル領域管理 1572
14.10.1 InnoDB ディスク I/O 1573
14.10.2 ファイル領域管理 1573
14.10.3 InnoDB チェックポイント 1574
14.10.4 テーブルのデフラグ 1575
14.10.5 TRUNCATE TABLE によるディスク領域の再利用 1575
14.11 InnoDB とオンライン DDL 1575
14.11.1 オンライン DDL の概要 1576
14.11.2 オンライン DDL でのパフォーマンスと並列性に関する考慮事項 1581
14.11.3 オンライン DDL の SQL 構文 1583
14.11.4 DDL ステートメントの結合または分離 1583
14.11.5 オンライン DDL の例 1584
14.11.6 オンライン DDL の実装の詳細 1602
14.11.7 オンライン DDL でのクラッシュリカバリの動作のしくみ 1604
14.11.8 パーティション化された InnoDB テーブルに対するオンライン DDL 1604
14.11.9 オンライン DDL の制限 1605
14.12 InnoDB の起動オプションおよびシステム変数 1605
14.13 InnoDB のパフォーマンス 1663
14.13.1 InnoDB バッファープールの構成 1663
14.13.2 InnoDB 相互排他ロックおよび読み取り/書き込みロックの実装 1669
14.13.3 InnoDB のためのメモリーアロケータの構成 1669
14.13.4 InnoDB 変更バッファリングの構成 1670
14.13.5 InnoDB のスレッド並列性の構成 1671
14.13.6 InnoDB バックグラウンド I/O スレッドの数の構成 1672
14.13.7 グループコミット 1672
14.13.8 InnoDB マススレッドの I/O レートの構成 1672
14.13.9 InnoDB スピンループでの PAUSE 命令の使用 1673
14.13.10 スピンロックのポーリングの構成 1673
14.13.11 InnoDB の MySQL パフォーマンススキーマとの統合 1674
14.13.12 複数のロールバックセグメントによるスケーラビリティーの向上 1674
14.13.13 InnoDB のパージスケジューリングの構成 1675
14.13.14 InnoDB の読み取り専用トランザクションの最適化 1675
14.13.15 チェックサムの高速化のための CRC32 チェックサムアルゴリズムの使用 1676
14.13.16 オプティマイザ統計 1676
14.13.17 InnoDB テーブルに対する ANALYZE TABLE の複雑さの推定 1683
14.14 InnoDB INFORMATION_SCHEMA テーブル 1684
14.14.1 圧縮に関する InnoDB INFORMATION_SCHEMA テーブル 1685
14.14.2 InnoDB INFORMATION_SCHEMA トランザクションおよびロックテーブル 1686
14.14.3 InnoDB INFORMATION_SCHEMA システムテーブル 1691
14.14.4 InnoDB INFORMATION_SCHEMA FULLTEXT インデックステーブル 1696

14.14.5 InnoDB INFORMATION_SCHEMA バッファープールテーブル 1698
14.14.6 InnoDB INFORMATION_SCHEMA メトリックテーブル 1702
14.15 InnoDB モニター 1709
14.15.1 InnoDB モニターのタイプ 1710
14.15.2 InnoDB モニターの有効化 1710
14.15.3 InnoDB 標準モニターおよびロックモニターの出力 1712
14.15.4 InnoDB テーブルスペースモニターの出力 1716
14.15.5 InnoDB テーブルモニターの出力 1718
14.16 InnoDB のバックアップとリカバリ 1720
14.16.1 InnoDB のリカバリプロセス 1722
14.17 InnoDB と MySQL レプリケーション 1723
14.18 InnoDB と memcached の統合 1724
14.18.1 InnoDB と memcached の組み合わせの利点 1725
14.18.2 InnoDB および memcached の統合のアーキテクチャー 1726
14.18.3 InnoDB Memcached プラグインの概要 1729
14.18.4 InnoDB memcached プラグインのセキュリティーに関する考慮事項 1731
14.18.5 InnoDB memcached インタフェース用のアプリケーションの作成 1733
14.18.6 レプリケーションでの InnoDB memcached プラグインの使用 1742
14.18.7 InnoDB memcached プラグインの内部構造 1745
14.18.8 InnoDB memcached プラグインのトラブルシューティング 1749
14.19 InnoDB のトラブルシューティング 1751
14.19.1 InnoDB の I/O に関する問題のトラブルシューティング 1751
14.19.2 InnoDB のリカバリの強制的な実行 1752
14.19.3 InnoDB データディクショナリの操作のトラブルシューティング 1753
14.19.4 InnoDB のエラー処理 1755
14.19.5 InnoDB のエラーコード 1755
14.19.6 オペレーティングシステムのエラーコード 1756
15 代替ストレージエンジン 1759
15.1 ストレージエンジンの設定 1763
15.2 MyISAM ストレージエンジン 1763
15.2.1 MyISAM 起動オプション 1766
15.2.2 キーに必要な容量 1767
15.2.3 MyISAM テーブルのストレージフォーマット 1767
15.2.4 MyISAM テーブルの問題点 1769
15.3 MEMORY ストレージエンジン 1771
15.4 CSV ストレージエンジン 1774
15.4.1 CSV テーブルの修復と確認 1775
15.4.2 CSV の制限 1776
15.5 ARCHIVE ストレージエンジン 1776
15.6 BLACKHOLE ストレージエンジン 1778
15.7 MERGE ストレージエンジン 1780
15.7.1 MERGE テーブルの長所と短所 1782
15.7.2 MERGE テーブルの問題点 1783
15.8 FEDERATED ストレージエンジン 1784
15.8.1 FEDERATED ストレージエンジンの概要 1784
15.8.2 FEDERATED テーブルの作成方法 1785
15.8.3 FEDERATED ストレージエンジンの注記とヒント 1787
15.8.4 FEDERATED ストレージエンジンのリソース 1789
15.9 EXAMPLE ストレージエンジン 1789
15.10 ほかのストレージエンジン 1789
15.11 MySQL ストレージエンジンアーキテクチャーの概要 1789
15.11.1 プラガブルストレージエンジンのアーキテクチャー 1790
15.11.2 共通データベースサーバーレイヤー 1790
16 高可用性と拡張性 1793
16.1 Oracle VM Template for MySQL Enterprise 1795
16.2 DRBD/Pacemaker/Corosync/Oracle Linux を使用した MySQL の概要 1796
16.3 Windows フェイルオーバークラスタリングを使用した MySQL の概要 1798
16.4 Amazon EC2 インスタンスでの MySQL の使用 1800
16.4.1 EC2 AMI での MySQL のセットアップ 1800
16.4.2 EC2 インスタンスの制限 1801
16.4.3 EC2 を使用した MySQL データベースの配備 1802
16.5 ZFS レプリケーションの使用 1804
16.5.1 ZFS を使用したファイルシステムレプリケーション 1805
16.5.2 ZFS レプリケーションのための MySQL の構成 1806

16.5.3 ZFS での MySQL リカバリーの扱い ... 1806
16.6 MySQL と memcached の併用 ... 1807
16.6.1 memcached のインストール ... 1808
16.6.2 memcached の使用 ... 1809
16.6.3 memcached アプリケーションの開発 ... 1825
16.6.4 memcached の統計の取得 ... 1846
16.6.5 memcached の FAQ ... 1853
17 レプリケーション ... 1857
17.1 レプリケーション構成 ... 1858
17.1.1 レプリケーションのセットアップ方法 ... 1859
17.1.2 レプリケーション形式 ... 1867
17.1.3 グローバルトランザクション識別子を使用したレプリケーション ... 1873
17.1.4 レプリケーションおよびバイナリロギングのオプションと変数 ... 1879
17.1.5 一般的なレプリケーション管理タスク ... 1949
17.2 レプリケーションの実装 ... 1951
17.2.1 レプリケーション実装の詳細 ... 1952
17.2.2 レプリケーションリレーおよびステータスログ ... 1953
17.2.3 サーバーがレプリケーションフィルタリングルールをどのように評価するか ... 1958
17.3 レプリケーションソリューション ... 1964
17.3.1 バックアップ用にレプリケーションを使用する ... 1964
17.3.2 異なるマスターおよびスレーブストレージエンジンでレプリケーションを使用する ... 1968
17.3.3 スケールアウトのためにレプリケーションを使用する ... 1969
17.3.4 異なるデータベースを異なるスレーブに複製する ... 1970
17.3.5 レプリケーションパフォーマンスを改善する ... 1971
17.3.6 フェイルオーバー中にマスターを切り替える ... 1971
17.3.7 SSL を使用してレプリケーションをセットアップする ... 1973
17.3.8 準同期レプリケーション ... 1975
17.3.9 遅延レプリケーション ... 1978
17.4 レプリケーションの注釈とヒント ... 1979
17.4.1 レプリケーションの機能と問題 ... 1979
17.4.2 MySQL バージョン間のレプリケーション互換性 ... 2001
17.4.3 レプリケーションセットアップをアップグレードする ... 2002
17.4.4 レプリケーションのトラブルシューティング ... 2003
17.4.5 レプリケーションバグまたは問題を報告する方法 ... 2004
18 MySQL Cluster NDB 7.3 および MySQL Cluster NDB 7.4 ... 2007
18.1 MySQL Cluster の概要 ... 2010
18.1.1 MySQL Cluster の主な概念 ... 2011
18.1.2 MySQL Cluster のノード、ノードグループ、レプリカ、およびパーティション ... 2014
18.1.3 MySQL Cluster のハードウェア、ソフトウェア、およびネットワーク要件 ... 2016
18.1.4 MySQL Cluster の開発履歴 ... 2018
18.1.5 InnoDB を使用した MySQL Server と MySQL Cluster の比較 ... 2020
18.1.6 MySQL Cluster の既知の制限 ... 2023
18.2 MySQL Cluster のインストール ... 2031
18.2.1 MySQL Cluster Auto-Installer ... 2033
18.2.2 Linux での MySQL Cluster のインストール ... 2047
18.2.3 Windows での MySQL Cluster のインストール ... 2052
18.2.4 MySQL Cluster の初期構成 ... 2059
18.2.5 MySQL Cluster の初期起動 ... 2061
18.2.6 テーブルとデータを含む MySQL Cluster の例 ... 2062
18.2.7 MySQL Cluster の安全なシャットダウンと再起動 ... 2064
18.2.8 MySQL Cluster NDB 7.3 のアップグレードとダウングレード ... 2065
18.3 MySQL Cluster の構成 ... 2066
18.3.1 MySQL Cluster の簡易テストセットアップ ... 2067
18.3.2 MySQL Cluster の構成ファイル ... 2069
18.3.3 MySQL Cluster 構成パラメータの概要 ... 2141
18.3.4 MySQL Cluster 用の MySQL Server オプションおよび変数 ... 2162
18.3.5 MySQL Cluster での高速インターコネクトの使用 ... 2220
18.4 MySQL Cluster プログラム ... 2222
18.4.1 ndbd — MySQL Cluster データノードデーモン ... 2222
18.4.2 ndbinfo_select_all — ndbinfo テーブルからの選択 ... 2228
18.4.3 ndbmtd — MySQL Cluster データノードデーモン (マルチスレッド) ... 2229
18.4.4 ndb_mgmd — MySQL Cluster 管理サーバーデーモン ... 2230
18.4.5 ndb_mgm — MySQL Cluster 管理クライアント ... 2237

18.4.6 ndb_blob_tool — MySQL Cluster テーブルの BLOB および TEXT カラムのチェックおよび修復 2238
18.4.7 ndb_config — MySQL Cluster 構成情報の抽出 2240
18.4.8 ndb_cpcd — NDB 開発のためのテストの自動化 2247
18.4.9 ndb_delete_all — NDB テーブルからのすべての行の削除 2247
18.4.10 ndb_desc — NDB テーブルの表示 2248
18.4.11 ndb_drop_index — NDB テーブルからのインデックスの削除 2251
18.4.12 ndb_drop_table — NDB テーブルの削除 2252
18.4.13 ndb_error_reporter — NDB エラーレポートユーティリティー 2253
18.4.14 ndb_index_stat — NDB インデックス統計ユーティリティー 2254
18.4.15 ndb_print_backup_file — NDB バックアップファイルの内容の出力 2258
18.4.16 ndb_print_file — NDB ディスクデータファイル内容の出力 2259
18.4.17 ndb_print_schema_file — NDB スキーマファイル内容の出力 2259
18.4.18 ndb_print_sys_file — NDB システムファイル内容の出力 2260
18.4.19 ndbd_redo_log_reader — クラスタ Redo ログ内容のチェックおよび出力 2260
18.4.20 ndb_restore — MySQL Cluster バックアップのリストア 2261
18.4.21 ndb_select_all — NDB テーブルの行の出力 2272
18.4.22 ndb_select_count — NDB テーブルの行数の出力 2275
18.4.23 ndb_setup.py — MySQL Cluster のブラウザベース自動インストーラの開始 2276
18.4.24 ndb_show_tables — NDB テーブルのリストの表示 2278
18.4.25 ndb_size.pl — NDBCLUSTER サイズ要件エスティメータ 2279
18.4.26 ndb_waiter — MySQL Cluster が指定したステータスになるまで待機する 2282
18.4.27 MySQL Cluster プログラムに共通するオプション — MySQL Cluster プログラムに共通するオプション 2284
18.5 MySQL Cluster の管理 2287
18.5.1 MySQL Cluster の起動フェーズのサマリー 2288
18.5.2 MySQL Cluster 管理クライアントのコマンド 2289
18.5.3 MySQL Cluster のオンラインバックアップ 2292
18.5.4 MySQL Cluster での MySQL サーバーの使用法 2296
18.5.5 MySQL Cluster のローリング再起動の実行 2297
18.5.6 MySQL Cluster で生成されたイベントレポート 2299
18.5.7 MySQL Cluster ログメッセージ 2308
18.5.8 MySQL Cluster のシングルユーザーモード 2317
18.5.9 クイックリファレンス: MySQL Cluster の SQL ステートメント 2317
18.5.10 ndbinfo MySQL Cluster 情報データベース 2319
18.5.11 MySQL Cluster のセキュリティー上の問題 2341
18.5.12 MySQL Cluster ディスクデータテーブル 2347
18.5.13 MySQL Cluster データノードのオンライン追加 2353
18.5.14 MySQL Cluster の配布された MySQL 権限 2362
18.5.15 NDB API 統計のカウンタと変数 2365
18.6 MySQL Cluster レプリケーション 2375
18.6.1 MySQL Cluster レプリケーション: 略語と記号 2376
18.6.2 MySQL Cluster レプリケーションの一般要件 2376
18.6.3 MySQL Cluster レプリケーションの既知の問題 2377
18.6.4 MySQL Cluster レプリケーションスキーマとテーブル 2382
18.6.5 レプリケーションのための MySQL Cluster の準備 2385
18.6.6 MySQL Cluster レプリケーションの起動 (レプリケーションチャネルが 1 つ) 2386
18.6.7 2 つのレプリケーションチャネルを使用する MySQL Cluster レプリケーション 2388
18.6.8 MySQL Cluster レプリケーションを使用したフェイルオーバーの実装 2389
18.6.9 MySQL Cluster レプリケーションを使用した MySQL Cluster バックアップ 2390
18.6.10 MySQL Cluster レプリケーション: マルチマスターと循環レプリケーション 2395
18.6.11 MySQL Cluster レプリケーションの競合解決 2399
18.7 MySQL Cluster リリースノート 2410
19 パーティション化 2411
19.1 MySQL のパーティショニングの概要 2413
19.2 パーティショニングタイプ 2415
19.2.1 RANGE パーティショニング 2416
19.2.2 LIST パーティショニング 2420
19.2.3 COLUMNS パーティショニング 2422
19.2.4 HASH パーティショニング 2428
19.2.5 KEY パーティショニング 2431
19.2.6 サブパーティショニング 2432
19.2.7 MySQL パーティショニングによる NULL の扱い 2435
19.3 パーティション管理 2439

19.3.1 RANGE および LIST パーティションの管理 ... 2439
19.3.2 HASH および KEY パーティションの管理 ... 2444
19.3.3 パーティションとサブパーティションをテーブルと交換する ... 2445
19.3.4 パーティションの保守 ... 2450
19.3.5 パーティションに関する情報を取得する ... 2451
19.4 パーティションプルーニング ... 2453
19.5 パーティション選択 ... 2455
19.6 パーティショニングの制約と制限 ... 2460
19.6.1 パーティショニングキー、主キー、および一意キー ... 2465
19.6.2 ストレージエンジンに関連するパーティショニング制限 ... 2468
19.6.3 関数に関連するパーティショニング制限 ... 2469
19.6.4 パーティショニングとロック ... 2470
20 ストアドプログラムおよびビュー ... 2473
20.1 ストアドプログラムの定義 ... 2474
20.2 ストアドルーチン (プロシージャーと関数) の使用 ... 2474
20.2.1 ストアドルーチンの構文 ... 2475
20.2.2 ストアドルーチンと MySQL 権限 ... 2476
20.2.3 ストアドルーチンのメタデータ ... 2476
20.2.4 ストアドプロシージャー、関数、トリガー、および LAST_INSERT_ID() ... 2476
20.3 トリガーの使用 ... 2477
20.3.1 トリガーの構文と例 ... 2477
20.3.2 トリガーのメタデータ ... 2480
20.4 イベントスケジューラの使用 ... 2481
20.4.1 イベントスケジューラの概要 ... 2481
20.4.2 イベントスケジューラの構成 ... 2482
20.4.3 イベント構文 ... 2484
20.4.4 イベントメタデータ ... 2484
20.4.5 イベントスケジューラのステータス ... 2485
20.4.6 イベントスケジューラと MySQL 権限 ... 2485
20.5 ビューの使用 ... 2488
20.5.1 ビューの構文 ... 2488
20.5.2 ビュー処理アルゴリズム ... 2488
20.5.3 更新可能および挿入可能なビュー ... 2489
20.5.4 ビューのメタデータ ... 2491
20.6 ストアドプログラムおよびビューのアクセスコントロール ... 2491
20.7 ストアドプログラムのバイナリロギング ... 2493
21 INFORMATION_SCHEMA テーブル ... 2499
21.1 INFORMATION_SCHEMA CHARACTER_SETS テーブル ... 2502
21.2 INFORMATION_SCHEMA COLLATIONS テーブル ... 2502
21.3 INFORMATION_SCHEMA COLLATION_CHARACTER_SET_APPLICABILITY テーブル ... 2503
21.4 INFORMATION_SCHEMA COLUMNS テーブル ... 2503
21.5 INFORMATION_SCHEMA COLUMN_PRIVILEGES テーブル ... 2504
21.6 INFORMATION_SCHEMA ENGINES テーブル ... 2504
21.7 INFORMATION_SCHEMA EVENTS テーブル ... 2504
21.8 INFORMATION_SCHEMA GLOBAL_STATUS および SESSION_STATUS テーブル ... 2507
21.9 INFORMATION_SCHEMA GLOBAL_VARIABLES および SESSION_VARIABLES テーブル ... 2508
21.10 INFORMATION_SCHEMA KEY_COLUMN_USAGE テーブル ... 2508
21.11 INFORMATION_SCHEMA OPTIMIZER_TRACE テーブル ... 2509
21.12 INFORMATION_SCHEMA PARAMETERS テーブル ... 2509
21.13 INFORMATION_SCHEMA PARTITIONS テーブル ... 2510
21.14 INFORMATION_SCHEMA PLUGINS テーブル ... 2512
21.15 INFORMATION_SCHEMA PROCESSLIST テーブル ... 2513
21.16 INFORMATION_SCHEMA PROFILING テーブル ... 2514
21.17 INFORMATION_SCHEMA REFERENTIAL_CONSTRAINTS テーブル ... 2515
21.18 INFORMATION_SCHEMA ROUTINES テーブル ... 2515
21.19 INFORMATION_SCHEMA SCHEMATA テーブル ... 2517
21.20 INFORMATION_SCHEMA SCHEMA_PRIVILEGES テーブル ... 2517
21.21 INFORMATION_SCHEMA STATISTICS テーブル ... 2517
21.22 INFORMATION_SCHEMA TABLES テーブル ... 2518
21.23 INFORMATION_SCHEMA TABLESPACES テーブル ... 2519
21.24 INFORMATION_SCHEMA TABLE_CONSTRAINTS テーブル ... 2520
21.25 INFORMATION_SCHEMA TABLE_PRIVILEGES テーブル ... 2520
21.26 INFORMATION_SCHEMA TRIGGERS テーブル ... 2520
21.27 INFORMATION_SCHEMA USER_PRIVILEGES テーブル ... 2522

21.28 INFORMATION_SCHEMA VIEWS テーブル 2522
21.29 InnoDB の INFORMATION_SCHEMA テーブル 2523
21.29.1 INFORMATION_SCHEMA INNODB_CMP および INNODB_CMP_RESET テーブル 2524
21.29.2 INFORMATION_SCHEMA INNODB_CMP_PER_INDEX および INNODB_CMP_PER_INDEX_RESET テーブル 2525
21.29.3 INFORMATION_SCHEMA INNODB_CMPMEM および INNODB_CMPMEM_RESET テーブル 2526
21.29.4 INFORMATION_SCHEMA INNODB_TRX テーブル 2527
21.29.5 INFORMATION_SCHEMA INNODB_LOCKS テーブル 2529
21.29.6 INFORMATION_SCHEMA INNODB_LOCK_WAITS テーブル 2530
21.29.7 INFORMATION_SCHEMA INNODB_SYS_TABLES テーブル 2531
21.29.8 INFORMATION_SCHEMA INNODB_SYS_INDEXES テーブル 2532
21.29.9 INFORMATION_SCHEMA INNODB_SYS_COLUMNS テーブル 2533
21.29.10 INFORMATION_SCHEMA INNODB_SYS_FIELDS テーブル 2534
21.29.11 INFORMATION_SCHEMA INNODB_SYS_FOREIGN テーブル 2535
21.29.12 INFORMATION_SCHEMA INNODB_SYS_FOREIGN_COLS テーブル 2535
21.29.13 INFORMATION_SCHEMA INNODB_SYS_TABLESTATS ビュー 2536
21.29.14 INFORMATION_SCHEMA INNODB_SYS_DATAFILES テーブル 2537
21.29.15 INFORMATION_SCHEMA INNODB_SYS_TABLESPACES テーブル 2537
21.29.16 INFORMATION_SCHEMA INNODB_BUFFER_PAGE テーブル 2540
21.29.17 INFORMATION_SCHEMA INNODB_BUFFER_PAGE_LRU テーブル 2541
21.29.18 INFORMATION_SCHEMA INNODB_BUFFER_POOL_STATS テーブル 2543
21.29.19 INFORMATION_SCHEMA INNODB_METRICS テーブル 2545
21.29.20 INFORMATION_SCHEMA INNODB_FT_CONFIG テーブル 2546
21.29.21 INFORMATION_SCHEMA INNODB_FT_DEFAULT_STOPWORD テーブル 2547
21.29.22 INFORMATION_SCHEMA INNODB_FT_INDEX_TABLE テーブル 2548
21.29.23 INFORMATION_SCHEMA INNODB_FT_INDEX_CACHE テーブル 2549
21.29.24 INFORMATION_SCHEMA INNODB_FT_DELETED テーブル 2550
21.29.25 INFORMATION_SCHEMA INNODB_FT_BEING_DELETED テーブル 2551
21.30 MySQL Cluster の INFORMATION_SCHEMA テーブル 2552
21.30.1 INFORMATION_SCHEMA FILES テーブル 2552
21.30.2 INFORMATION_SCHEMA ndb_transid_mysql_connection_map テーブル 2556
21.31 スレッドプールの INFORMATION_SCHEMA テーブル 2557
21.31.1 INFORMATION_SCHEMA TP_THREAD_STATE テーブル 2558
21.31.2 INFORMATION_SCHEMA TP_THREAD_GROUP_STATE テーブル 2558
21.31.3 INFORMATION_SCHEMA TP_THREAD_GROUP_STATS テーブル 2560
21.32 SHOW ステートメントの拡張 2561
22 MySQL パフォーマンススキーマ 2565
22.1 パフォーマンススキーマクイックスタート 2566
22.2 パフォーマンススキーマ構成 2571
22.2.1 パフォーマンススキーマビルド構成 2571
22.2.2 パフォーマンススキーマ起動構成 2572
22.2.3 パフォーマンススキーマ実行時構成 2574
22.3 パフォーマンススキーマクエリー 2589
22.4 パフォーマンススキーマインストゥルメント命名規則 2589
22.5 パフォーマンススキーマステータスモニタリング 2591
22.6 パフォーマンススキーマの原子的および分子的イベント 2594
22.7 パフォーマンススキーマステートメントダイジェスト 2595
22.8 パフォーマンススキーマの一般的なテーブル特性 2596
22.9 パフォーマンススキーマテーブルの説明 2596
22.9.1 パフォーマンススキーマテーブルインデックス 2597
22.9.2 パフォーマンススキーマセットアップテーブル 2598
22.9.3 パフォーマンススキーマインスタンステーブル 2602
22.9.4 パフォーマンススキーマ待機イベントテーブル 2606
22.9.5 パフォーマンススキーマステージイベントテーブル 2609
22.9.6 パフォーマンススキーマステートメントイベントテーブル 2612
22.9.7 パフォーマンススキーマ接続テーブル 2617
22.9.8 パフォーマンススキーマ接続属性テーブル 2620
22.9.9 パフォーマンススキーマサマリーテーブル 2620
22.9.10 パフォーマンススキーマのその他のテーブル 2631
22.10 パフォーマンススキーマオプションおよび変数リファレンス 2637
22.11 パフォーマンススキーマコマンドオプション 2639
22.12 パフォーマンススキーマシステム変数 2640
22.13 パフォーマンススキーマステータス変数 2648

22.14 パフォーマンススキーマとプラグイン 2650
22.15 問題を診断するためのパフォーマンススキーマの使用 2650
23 Connector および API 2653
23.1 MySQL Connector/ODBC 2656
23.2 MySQL Connector/Net 2656
23.3 MySQL Connector/J 2656
23.4 MySQL Connector/C++ 2657
23.5 MySQL Connector/Python 2657
23.6 組み込み MySQL サーバーライブラリ libmysqld 2657
23.6.1 libmysqld によるプログラムのコンパイル 2657
23.6.2 組み込み MySQL サーバーを使用する場合の制限 2658
23.6.3 組み込みサーバーとオプション 2658
23.6.4 組み込みサーバーの例 2658
23.7 MySQL C API 2661
23.7.1 MySQL C API の実装 2662
23.7.2 MySQL サーバーと MySQL Connector/C の同時インストール 2663
23.7.3 C API クライアントプログラムの例 2664
23.7.4 C API クライアントプログラムの構築と実行 2664
23.7.5 C API データ構造 2667
23.7.6 C API 関数の概要 2671
23.7.7 C API 関数の説明 2675
23.7.8 C API プリペアドステートメント 2722
23.7.9 C API プリペアドステートメントデータ構造 2722
23.7.10 C API プリペアドステートメント関数の概要 2727
23.7.11 C API プリペアドステートメント関数の説明 2730
23.7.12 C API スレッド関数の説明 2750
23.7.13 C API 組み込みサーバー関数の説明 2751
23.7.14 C API クライアントプラグイン関数 2752
23.7.15 C API を使用する場合の一般的な質問と問題 2754
23.7.16 自動再接続動作の制御 2756
23.7.17 複数ステートメント実行の C API サポート 2757
23.7.18 C API プリペアドステートメントの問題 2758
23.7.19 C API プリペアドステートメントの日時値の処理 2759
23.7.20 C API のプリペアド CALL ステートメントのサポート 2760
23.8 MySQL PHP API 2763
23.9 MySQL Perl API 2763
23.10 MySQL Python API 2764
23.11 MySQL Ruby API 2764
23.11.1 MySQL/Ruby API 2764
23.11.2 Ruby/MySQL API 2765
23.12 MySQL Tcl API 2765
23.13 MySQL Eiffel ラッパー 2765
24 MySQL の拡張 2767
24.1 MySQL の内部仕様 2767
24.1.1 MySQL のスレッド 2767
24.1.2 MySQL テストスイート 2768
24.2 MySQL プラグイン API 2768
24.2.1 プラグイン API の特徴 2769
24.2.2 プラグイン API のコンポーネント 2770
24.2.3 プラグインのタイプ 2771
24.2.4 プラグインの作成 2773
24.2.5 プラグインのための MySQL サービス 2810
24.3 MySQL への新しい関数の追加 2812
24.3.1 ユーザー定義関数インタフェースの機能 2813
24.3.2 新しいユーザー定義関数の追加 2813
24.3.3 新しいネイティブ関数の追加 2821
24.4 MySQL のデバッグおよび移植 2822
24.4.1 MySQL サーバーのデバッグ 2823
24.4.2 MySQL クライアントのデバッグ 2828
24.4.3 DBUG パッケージ 2829
25 MySQL Enterprise Edition 2833
25.1 MySQL Enterprise Monitor 2833
25.2 MySQL Enterprise Backup 2834
25.3 MySQL Enterprise Security 2834

25.4 MySQL Enterprise Encryption 2835
25.5 MySQL Enterprise Audit 2835
25.6 MySQL Enterprise Thread Pool 2835
26 MySQL Workbench 2837
A MySQL 5.6 のよくある質問 2839
A.1 MySQL 5.6 FAQ: 全般 2839
A.2 MySQL 5.6 FAQ: ストレージエンジン 2840
A.3 MySQL 5.6 FAQ: サーバー SQL モード 2841
A.4 MySQL 5.6 FAQ: ストアドプロシージャーおよびストアドファンクション 2842
A.5 MySQL 5.6 FAQ: トリガー 2845
A.6 MySQL 5.6 FAQ: ビュー 2847
A.7 MySQL 5.6 FAQ: INFORMATION_SCHEMA 2848
A.8 MySQL 5.6 FAQ: 移行 2848
A.9 MySQL 5.6 FAQ: セキュリティー 2849
A.10 MySQL 5.6 FAQ: MySQL Cluster 2850
A.11 MySQL 5.6 FAQ: MySQL の中国語、日本語、および韓国語の文字セット 2861
A.12 MySQL 5.6 FAQ: コネクタおよび API 2871
A.13 MySQL 5.6 FAQ: レプリケーション 2871
A.14 MySQL 5.6 FAQ: MySQL エンタープライズスケーラビリティースレッドプール 2874
B エラー、エラーコード、および一般的な問題 2877
B.1 エラー情報のソース 2877
B.2 エラー値のタイプ 2877
B.3 サーバーのエラーコードおよびメッセージ 2878
B.4 クライアントのエラーコードおよびメッセージ 2937
B.5 問題および一般的なエラー 2941
B.5.1 問題の原因を判別する方法 2941
B.5.2 MySQL プログラム使用時の一般的なエラー 2942
B.5.3 インストール関連の問題 2953
B.5.4 管理関連の問題 2954
B.5.5 クエリー関連の問題 2960
B.5.6 オプティマイザ関連の問題 2966
B.5.7 テーブル定義関連の問題 2966
B.5.8 MySQL の既知の問題 2967
C MySQL リリースノート 2971
D 制約と制限 2973
D.1 ストアドプログラムの制約 2973
D.2 条件処理の制約 2976
D.3 サーバー側のカーソルの制約 2976
D.4 サブクエリーの制約 2977
D.5 ビューの制約 2978
D.6 XA トランザクションの制約 2979
D.7 文字セットの制約 2980
D.8 パフォーマンススキーマの制約 2980
D.9 プラガブルな認証の制約 2981
D.10 MySQL での制限 2982
D.10.1 結合の制限 2982
D.10.2 データベースおよびテーブルの数に対する制限 2982
D.10.3 テーブルサイズの制限 2983
D.10.4 テーブルカラム数と行サイズの制限 2984
D.10.5 .frm ファイル構造により課せられる制限 2985
D.10.6 Windows プラットフォームの制限 2986
MySQL 用語集 2989
E サードパーティーコンポーネントライセンス 3039
E.1 ANTLR 3 ライセンス 3040
E.2 Artistic ライセンス (Perl) 1.0 3040
E.3 Boost ライブラリライセンス 3042
E.4 dtoa.c ライセンス 3042
E.5 Editline ライセンス (libedit) ライセンス 3042
E.6 Editline ライセンス (libedit) ライセンス 3044
E.7 Expect.pm ライセンス 3047
E.8 Facebook Fast Checksum Patch ライセンス 3053
E.9 Facebook Patches ライセンス 3053
E.10 FindGTest.cmake ライセンス 3054
E.11 Fred Fish's Dbug Library ライセンス 3054

E.12 getarg ライセンス 3055
E.13 Gmock ライセンス 3056
E.14 GNU General Public License バージョン 2.0、1991 年 6 月 3056
E.15 GNU General Public License バージョン 3.0、2007 年 6 月 29 日および GCC Runtime Library Exception バージョン 3.1、2009 年 3 月 31 日 3061
E.16 GNU Lesser General Public License バージョン 2.1、1999 年 2 月 3070
E.17 GNU Libtool ライセンス 3077
E.18 GNU Readline ライセンス 3078
E.19 GNU Standard C++ ライブラリ (libstdc++) ライセンス 3078
E.20 Google Controlling Master Thread I/O Rate Patch ライセンス 3079
E.21 Google Perftools (TCMalloc ユーティリティー) ライセンス 3080
E.22 Google SMP Patch ライセンス 3080
E.23 Janson ライセンス 3081
E.24 lib_sql.cc ライセンス 3081
E.25 libevent ライセンス 3081
E.26 Linux-PAM ライセンス 3083
E.27 md5 (メッセージダイジェストアルゴリズム 5) ライセンス 3084
E.28 memcached ライセンス 3084
E.29 Memcached.pm ライセンス 3084
E.30 mkpasswd.pl ライセンス 3085
E.31 Node.js ライセンス 3088
E.32 nt_servc (Windows NT Service クラスライブラリ) ライセンス 3093
E.33 OpenPAM ライセンス 3093
E.34 OpenSSL v1.0 ライセンス 3093
E.35 Paramiko ライセンス 3095
E.36 Percona Multiple I/O スレッドパッチライセンス 3095
E.37 Pion ライセンス 3095
E.38 Python ライセンス 3096
E.39 Red Hat RPM Spec ファイルライセンス 3105
E.40 RegEX-Spencer ライブラリライセンス 3105
E.41 Richard A.O'Keefe 文字列ライブラリライセンス 3105
E.42 sajson ライセンス 3106
E.43 SHA-1 in C ライセンス 3106
E.44 Unicode データファイル 3106
E.45 V8 ライセンス 3107
E.46 zlib ライセンス 3110
全般的な索引 3111
C 関数の索引 3187
コマンドの索引 3195
関数の索引 3221
INFORMATION_SCHEMA の索引 3239
結合型の索引 3245
演算子の索引 3247
オプションの索引 3251
権限の索引 3305
SQL モードの索引 3311
ステートメント/構文の索引 3313
ステータス変数の索引 3357
システム変数の索引 3367
トランザクション分離レベルのインデックス 3397

序文と法的通知

これは、MySQL Database System バージョン 5.6 (リリース 5.6.23 まで) のリファレンスマニュアルです。MySQL 5.6 のマイナーバージョン間での相違点は、リリース番号 (5.6.x) に関連した現在のテキストに記されます。ライセンス情報については、法的通知を参照してください。この製品には、サードパーティーのコードが含まれる場合があります。サードパーティーコードのライセンス情報については、付録E「サードパーティーコンポーネントライセンス」を参照してください。

MySQL 5.6 と以前のバージョンとの間に機能などの多くの相違点があるため、このマニュアルは MySQL ソフトウェアの古いバージョンで用することは想定されていません。MySQL ソフトウェアの以前のリリースを使用している場合は、該当するマニュアルを参照してください。たとえば、MySQL 5.5 リファレンスマニュアルには、5.5 シリーズの MySQL ソフトウェアリリースが取り上げられています。

法律上の注意点

第 1 章 一般情報

目次

1.1 このマニュアルについて 2
1.2 表記規則および構文規則 2
1.3 MySQL データベース管理システムの概要 4
1.3.1 MySQL とは 4
1.3.2 MySQL の主な機能 5
1.3.3 MySQL の歴史 7
1.4 MySQL 5.6 の新機能 8
1.5 MySQL の情報源 18
1.5.1 MySQL メーリングリスト 18
1.5.2 MySQL フォーラムにおける MySQL コミュニティーサポート 20
1.5.3 IRC (インターネットリレーチャット) の MySQL コミュニティーサポート 20
1.5.4 MySQL Enterprise 21
1.6 質問またはバグをレポートする方法 21
1.7 MySQL の標準への準拠 24
1.7.1 標準 SQL に対する MySQL 拡張機能 26
1.7.2 MySQL と標準 SQL との違い 28
1.7.3 MySQL における制約の処理 32
1.8 クレジット 35
1.8.1 MySQL への貢献者 35
1.8.2 ドキュメント作成者および翻訳者 39
1.8.3 MySQL をサポートするパッケージ 40
1.8.4 MySQL の作成に使用されたツール 40
1.8.5 MySQL のサポータ 41

このマニュアルで特に興味深いセクションは次のとおりです。

- MySQL Database Server の機能の説明については、セクション1.3.2「MySQL の主な機能」を参照してください。
- MySQL の新しい機能の概要については、セクション1.4「MySQL 5.6 の新機能」を参照してください。各バージョンの変更内容の詳細は、リリースノートを参照してください。
- インストールの手順については、第2章「MySQL のインストールと更新」を参照してください。MySQL のアップグレードの詳細は、セクション2.11.1「MySQL のアップグレード」を参照してください。
- MySQL Database Server のチュートリアルについては、第3章「チュートリアル」を参照してください。
- MySQL Server の構成と管理の詳細は、第5章「MySQL サーバーの管理」を参照してください。
- MySQL でのセキュリティーの詳細は、第6章「セキュリティー」を参照してください。
- レプリケーションサーバーの設定の詳細は、第17章「レプリケーション」を参照してください。
- 高度な機能および管理ツールを備えた商用 MySQL リリースである MySQL Enterprise の詳細は、第25章「MySQL Enterprise Edition」を参照してください。
- MySQL Database Server とその機能についてよくある質問に対する答えは、付録A「MySQL 5.6 のよくある質問」を参照してください。

- 新しい機能とバグ修正の履歴については、リリースノートを参照してください。

> **重要**
>
> 問題またはバグをレポートするには、セクション1.6「質問またはバグをレポートする方法」で説明する手順を使用してください。MySQL Server で慎重に扱うべきセキュリティー上のバグを発見した場合は、ただちに <secalert_us@oracle.com> に電子メールメッセージを送信してお知らせください。例外: サポートのお客様は、セキュリティーのバグを含むすべての問題を Oracle Support までレポートしてください。

1.1 このマニュアルについて

これは、MySQL Database System バージョン 5.6 (リリース 5.6.23 まで) のリファレンスマニュアルです。MySQL 5.6 のマイナーバージョン間での相違点は、リリース番号 (5.6.x) に関連した現在のテキストに記されます。ライセンス情報については、法的通知を参照してください。この製品には、サードパーティーのコードが含まれる場合があります。サードパーティーコードのライセンス情報については、付録E「サードパーティーコンポーネントライセンス」を参照してください。

MySQL 5.6 と以前のバージョンとの間に機能などの多くの相違点があるため、このマニュアルは MySQL ソフトウェアの古いバージョンで用することは想定されていません。MySQL ソフトウェアの以前のリリースを使用している場合は、該当するマニュアルを参照してください。たとえば、MySQL 5.5 リファレンスマニュアルには、5.5 シリーズの MySQL ソフトウェアリリースが取り上げられています。

これはリファレンスマニュアルであるため、SQL やリレーショナルデータベースの概念に関する一般的な説明は記載していません。また、使用しているオペレーティングシステムやコマンド行インタープリタの使用法も記載していません。

MySQL データベースソフトウェアは継続して開発が行われているため、リファレンスマニュアルも頻繁に更新されます。マニュアルの最新版は、オンラインで https://dev.mysql.com/doc/ の検索可能なフォームから入手できます。ここでは、HTML、PDF、EPUB バージョンなどのその他の形式も入手できます。

リファレンスマニュアルのソースファイルは、DocBook XML 形式で記述されています。HTML バージョンとその他の形式は、主に DocBook XSL スタイルシートを使用して自動的に作成されます。DocBook の詳細は、http://docbook.org/を参照してください。

MySQL を利用するにあたって質問がある場合は、メーリングリストやフォーラムを使用してお尋ねください。セクション1.5.1「MySQL メーリングリスト」およびセクション1.5.2「MySQL フォーラムにおける MySQL コミュニティーサポート」を参照してください。マニュアル自体に対する追加または修正に関する提案がある場合は、http://www.mysql.com/company/contact/ に送信してください。

このマニュアルは当初、David Axmark と Michael 「Monty」 Widenius によって執筆されました。Paul DuBois、Edward Gilmore、Stefan Hinz、David Moss、Philip Olson、Daniel Price、Daniel So、および Jon Stephens から構成される MySQL ドキュメントチームによって維持されています。

1.2 表記規則および構文規則

このマニュアルは、次の表記規則に従って記載されています。

- このスタイルのテキストは、SQL ステートメント、データベース、テーブル、カラム名、プログラムリスト、ソースコード、環境変数に使用されます。例: 「付与テーブルをリロードするには、FLUSH PRIVILEGES ステートメントを使用します。」
- このスタイルのテキストは、例の中で入力する文字の例を示します。
- このスタイルのテキストは、mysql (MySQL コマンド行のクライアントプログラム) および mysqld (MySQL Server 実行可能ファイル) である実行可能プログラムおよびスクリプトの名前を示します。
- このスタイルのテキストは、独自に選択する値に置き換える変数入力に使用されます。
- このスタイルのテキストは、強調に使用されます。
- このスタイルのテキストは、表の見出しや特に重要なことを示すときに使用されます。
- このスタイルのテキストは、プログラムの実行方法に影響を与えるか、またはプログラムが特定の方法で機能するために必要な情報を提供するプログラムオプションを示すために使用されます。例: 「--host オプション (短縮形 -h) は、mysql クライアントプログラムに対して接続すべき MySQL Server のホスト名または IP アドレスを指示します」。

- ファイル名とディレクトリ名は次のように表示されます。「グローバル my.cnf ファイルは、/etc ディレクトリにあります。」
- 文字シーケンスは次のように表示されます。「ワイルドカードを指定するには、『%』 文字を使用します。」

特定のプログラム内から実行されるコマンドが示される場合、そのコマンドの前に記されるプロンプトは、どのコマンドが使用されるかを示しています。たとえば、shell> はログインシェルから実行するコマンドを示し、root-shell> は同様ですが root として実行する必要があり、mysql> は mysql クライアントプログラムから実行するステートメントを示します。

```
shell> type a shell command here
root-shell> type a shell command as root here
mysql> type a mysql statement here
```

領域によっては、コマンドを 2 つの異なる環境で実行すべきであることを示すために、異なるシステムが区別される場合があります。たとえば、レプリケーションを操作する場合、コマンドの前に master および slave が記される場合があります。

```
master> type a mysql command on the replication master here
slave> type a mysql command on the replication slave here
```

「shell」 は、コマンドインタープリタです。UNIX の場合、これは通常、sh、csh、bash などのプログラムです。Windows の場合、同等のプログラムは command.com または cmd.exe で、通常コンソールウィンドウで実行します。

例に示されるコマンドまたはステートメントを入力する場合、その例に示されるプロンプトを入力する必要はありません。

データベース名、テーブル名、およびカラム名は多くの場合、ステートメントに代入する必要があります。このような代入が必要であることを示す場合、このマニュアルでは db_name、tbl_name、および col_name を使用します。たとえば、次のようなステートメントがあるとします。

```
mysql> SELECT col_name FROM db_name.tbl_name;
```

これは、同様のステートメントを入力する場合、データベース名、テーブル名、およびカラム名を自分で指定することを意味します。たとえば、次のようになります。

```
mysql> SELECT author_name FROM biblio_db.author_list;
```

SQL キーワードは大文字と小文字を区別しないので、大文字で記述されることも小文字で記述されることもあります。このマニュアルでは大文字を使用します。

構文の説明では、角かっこ (「[」 および 「]」) はオプションの語または句を示します。たとえば、次のステートメントでは、IF EXISTS はオプションです。

```
DROP TABLE [IF EXISTS] tbl_name
```

構文要素が複数の選択肢で構成される場合、選択肢は縦棒 (「|」) で区切られます。一連の選択肢から 1 つのメンバーを選択できる場合、選択肢は角かっこ (「[」 および 「]」) 内にリストされます。

```
TRIM([[BOTH | LEADING | TRAILING] [remstr] FROM] str)
```

一連の選択肢から 1 つのメンバーを選択しなければならない場合、選択肢は中かっこ (「{」 および 「}」) 内にリストされます。

```
{DESCRIBE | DESC} tbl_name [col_name | wild]
```

省略記号 (...) は、ステートメントの一部が省略されていることを示し、通常、複雑な構文を簡単に記しています。たとえば、SELECT ... INTO OUTFILE は、ステートメントのほかの部分に INTO OUTFILE 句が続く SELECT ステートメントを省略したものです。

省略記号は、ステートメントで前にある構文要素を繰り返すことができる場合にも使用されます。次の例では、reset_option 値を、最初のもの以外はそれぞれカンマを前に付けて、複数回繰り返すことができます。

```
RESET reset_option [,reset_option] ...
```

シェル変数を設定するコマンドは、Bourne シェル構文を使用して示されます。たとえば、CC 環境変数を設定し、configure コマンドを実行するシーケンスは、Bourne シェル構文では次のように示されます。

```
shell> CC=gcc ./configure
```

csh または tcsh を使用する場合、多少異なるコマンドを入力する必要があります。

```
shell> setenv CC gcc
shell> ./configure
```

1.3 MySQL データベース管理システムの概要

1.3.1 MySQL とは

MySQL は、もっとも普及しているオープンソース SQL データベース管理システムで、オラクル社により開発、流通、およびサポートが行われています。

MySQL ウェブサイト (http://www-jp.mysql.com/) では、MySQL ソフトウェアに関する最新情報を提供しています。

- MySQL はデータベース管理システムです。

  データベースとは、構造化されたデータの集合です。簡単なショッピングリストから絵画のギャラリー、または社内ネットワークの膨大な量の情報など、さまざまなものがあります。コンピュータデータベースに格納されているデータに対して追加、アクセス、および処理などを行うには、MySQL Server のようなデータベース管理システムが必要となります。コンピュータは大量のデータの処理に非常に優れているので、データベース管理システムは、スタンドアロンのユーティリティーまたはほかのアプリケーションの一部として、データ処理の中心的な役割を果たします。

- MySQL データベースはリレーショナルデータベースです。

  リレーショナルデータベースは、すべてのデータを 1 つの大きな保管場所に置くのではなく、独立したテーブルに保存します。データベースの構造は、速度を最適化した物理ファイルにわかれています。データベース、テーブル、ビュー、行、およびカラムなどのオブジェクトをもつ論理モデルは、柔軟なプログラミング環境を提供します。さまざまなデータフィールドの間の関係を統治する、1 対 1、1 対多、一意、必須またはオプションなどのルールと、さまざまなテーブル間の「ポインタ」を設定します。適切に設計されたデータベースを使用すれば、アプリケーションにはデータの矛盾、重複、孤立、期限切れ、または欠損が決して現れないように、データベースはこれらのルールを実施します。

  「MySQL」 の、SQL の部分は 「Structured Query Language」 (構造化クエリー言語) を表しています。SQL は、データベースにアクセスするために使用されるもっとも一般的な標準化言語です。プログラミング環境によって、SQL を直接入力 (たとえばレポートの作成)、別の言語で作成されたコードに SQL ステートメントを埋め込み、または SQL 構文を隠す言語固有の API を使用する場合があります。

  SQL は ANSI/ISO SQL 標準で定義されます。SQL 標準は 1986 年以降開発が繰り返され、複数のバージョンがあります。本マニュアルでは、「SQL-92」 は 1992 年にリリースされた標準に、「SQL:1999」 は 1999 年にリリースされた標準に、そして 「SQL:2003」 は標準の現バージョンに対応しています。「SQL 標準」 という用語は、任意の時点における現行バージョンの SQL 標準を表す場合に使用します。

- MySQL ソフトウェアはオープンソースです。

  オープンソースとは、そのソフトウェアをだれでも使用および修正できることを意味します。MySQL ソフトウェアは、だれもが無料でインターネットからダウンロードし、使用することができます。必要に応じて、ソースコードを調べ、ニーズに合わせて変更することができます。MySQL ソフトウェアは GPL (GNU General Public License)、http://www.fsf.org/licenses/、を使用して、さまざまな状況で実行するものとしないものを定義します。GPL では不都合な場合や商用アプリケーションに MySQL コードを組み込む必要がある場合は、商用ライセンス版を購入することができます。詳細は、MySQL ライセンス情報 (http://www-jp.mysql.com/company/legal/licensing/) を参照してください。

- MySQL Database Server は非常に高速で信頼性が高く、スケーラブルで使いやすいです。

  それを求めていたのでしたら、ぜひお試しください。MySQL Server は、デスクトップまたはラップトップ上で、ほかのアプリケーションや Web サーバーなどと併用して、ほとんどまたはまったく処理を要することなく快適に実行できます。マシン全体を MySQL 専用にする場合、使用可能なすべてのメモリー、CPU パワー、および I/O 能力を利用するように設定を調整できます。MySQL は、ネットワークで接続されたマシンのクラスターにスケールアップすることもできます。

  ベンチマークページで、ほかのデータベース管理システムと MySQL Server のパフォーマンスを比較することができます。セクション8.12.2「MySQL ベンチマークスイート」を参照してください。

  MySQL Server は当初、既存のソリューションよりもはるかに速く大規模なデータベースを処理することを目的として開発され、多くを要求される運用環境で数年間にわたって使用されています。引き続き開発が行われていますが、MySQL Server は現在、便利で多彩な関数を備えています。その接続性、速度、安全性によって、MySQL Server はインターネット上のデータベースへのアクセスに非常に適しています。

- MySQL Server は、クライアント/サーバー組み込みシステムで機能します。

  MySQL Database Software は、さまざまなバックエンドをサポートするマルチスレッド形式の SQL Server、複数の異なるクライアントプログラムおよびライブラリ、管理ツール、幅広いアプリケーションプログラミングインタフェース(API)から成るクライアント/サーバーシステムです。

  また、MySQL Server はアプリケーションへのリンクが可能なマルチスレッドライブラリとして提供されており、さらに小規模で高速な、管理しやすいスタンドアロン製品を作り出すことができます。

- 大量の MySQL ソフトウェアが提供されており、使用可能です。

  MySQL Server には、ユーザーと緊密に協力して開発された、実際的な一連の機能があります。必要なアプリケーションや言語ですでに MySQL Database Server がサポートされていると思われます。

「MySQL」 の正式な読み方は、「マイエスキューエル」(「マイシークエル」ではありません) ですが、「マイシークエル」やその他のローカライズされた方法で発音してもかまいません。

1.3.2 MySQL の主な機能

このセクションでは MySQL Database Software の重要な特徴の一部を説明します。ほとんどの場合、ロードマップはすべてのバージョンの MySQL に適用されます。シリーズごとに新しく導入される MySQL の機能については、対応するマニュアルの「新機能」セクションを参照してください。

- MySQL 5.6: MySQL 5.6 の新機能
- MySQL 5.5: MySQL 5.5 の新機能
- MySQL 5.1: MySQL 5.1 の新機能
- MySQL 5.0: MySQL 5.0 の新機能

内部および移植性

- C および C++ で記述されています。
- さまざまなコンパイラでテストされています。
- さまざまなプラットフォームで動作します。https://www.mysql.com/support/supportedplatforms/database.html を参照してください。
- 移植性のために、MySQL 5.5 以降では CMake を使用しています。以前のシリーズでは GNU Automake、Autoconf、および Libtool を使用しています。
- Purify (商用メモリーリーク検出システム) と GPL ツールの Valgrind (http://developer.kde.org/~sewardj/) でテストされています。
- 独立モジュールを備えた多層サーバー設計を使用しています。
- カーネルスレッドを使用した完全なマルチスレッドとなるよう設計され、使用可能な場合、複数の CPU を簡単に使用することができます。
- トランザクションストレージエンジンと非トランザクションストレージエンジンを備えています。
- インデックス圧縮を備えた非常に高速な B-tree ディスクテーブル(MyISAM) を使用しています。
- 別のストレージエンジンの追加が比較的容易になるよう設計されています。これは、社内データベースへの SQL インタフェースを追加する場合に便利です。
- スレッドベースの非常に高速なメモリー割り当てシステムを使用しています。
- 最適化されたネストループ結合を使用して非常に高速な結合を実行します。
- インメモリーハッシュテーブルを実装し、一時テーブルとして使用します。
- 高度に最適化されたクラスライブラリを使用して SQL 関数が実装されるため、最大限の速度が確保されます。通常は、クエリーの初期化後にメモリー割り当てが行われることはありません。
- クライアント/サーバーネットワーク環境で使用するために、サーバーを独立したプログラムとして提供しています。単独のアプリケーションに組み込み (リンク) できるライブラリとしても提供されています。このようなアプリケーションは単一で、あるいはネットワーク環境の整っていない場所でも使用することができます。

データ型

- 多数のデータタイプ: 1、2、3、4、および 8 バイト長の符号付き/符号なし整数、FLOAT、DOUBLE、CHAR、VARCHAR、BINARY、VARBINARY、TEXT、BLOB、DATE、TIME、DATETIME、TIMESTAMP、YEAR、SET、ENUM、および OpenGIS 空間型。第11章「データ型」 を参照してください。
- 固定長および可変長の文字列型。

ステートメントと関数

- クエリーの SELECT 句および WHERE 句での演算子と関数の完全なサポート。例:

```
mysql> SELECT CONCAT(first_name, ' ', last_name)
    -> FROM citizen
    -> WHERE income/dependents > 10000 AND age > 30;
```

- SQL の GROUP BY 句および ORDER BY 句の完全なサポート。グループ関数 (COUNT()、AVG()、STD()、SUM()、MAX()、MIN()、および GROUP_CONCAT()) のサポート。
- 標準の SQL 構文および ODBC 構文での LEFT OUTER JOIN および RIGHT OUTER JOIN のサポート。
- 標準 SQL で必要な、テーブルおよびカラムにおけるエイリアスのサポート。
- 変更された (影響を受けた) 行の数を返す DELETE、INSERT、REPLACE、および UPDATE のサポート。サーバーに接続する際にフラグを設定することで、代わりに一致したレコードの数を返すことも可能です。
- データベース、ストレージエンジン、テーブル、およびインデックスに関する情報を取得する、MySQL 固有の SHOW ステートメントのサポート。MySQL 5.0 では、INFORMATION_SCHEMA データベースのサポートも、標準 SQL に基づき追加されています。
- オプティマイザによるクエリーの解決方法を表示する EXPLAIN ステートメント。
- 関数名の、テーブル名やカラム名との独立性。たとえば、ABS は有効なカラム名です。唯一の制限事項は、関数呼び出しで、関数名とその後に続く 「(」 との間にスペースを使用できないことです。セクション9.3「予約語」を参照してください。
- 同じステートメント内で、さまざまなデータベースのテーブルを参照することができます。

セキュリティー

- 非常に柔軟でセキュアな権限およびパスワードシステム。ホストベースの検証が可能です。
- サーバーに接続する際にすべてのパスワードトラフィックが暗号化されるので、パスワードは安全です。

拡張性と制限

- 大規模なデータベースのサポート。当社は、MySQL Server を使用して 50,000,000 レコードが格納されたデータベースを処理しています。また、MySQL Server を使用して 200,000 テーブル、約 5,000,000,000 行を処理しているユーザーもいます。
- 各テーブルで最高 64 個のインデックスをサポートします。(MySQL 4.1.2 では 32 個)。各インデックスは、1 から 16 個のカラムまたはカラムの一部で構成されます。インデックスの最大幅は InnoDB テーブルでは 767 バイト、MyISAM では 1000 バイトです。MySQL 4.1.2 では 500 が限度でした。インデックスでは、CHAR、VARCHAR、BLOB、あるいは TEXT 型のカラムのプリフィクスを使用することができます。

接続性

- クライアントは複数のプロトコルを使用して MySQL Server に接続できます。
  - クライアントは、あらゆるプラットフォームで TCP/IP ソケットを使用して接続することができます。
  - Windows NT ファミリ (NT、2000、XP、2003、または Vista) の Windows システムでは、サーバーが --enable-named-pipe オプションで起動された場合、クライアントは名前付きパイプを使用して接続できます。MySQL 4.1 以降では、--shared-memory オプションで起動されていれば Windows のサーバーは共有メモリー接続もサポートします。クライアントは --protocol=memory オプションを使用して共有メモリーで接続できます。

  - Unix システムでは、クライアントは Unix ドメインソケットファイルを使用して接続することができます。
- MySQL クライアントプログラムはさまざまな言語で記述できます。C 言語で記述されたクライアントライブラリは C、C++、あるいは C バインディングを提供する任意の言語で記述されたクライアントでも使用可能です。
- C、C++、Eiffel、Java、Perl、PHP、Python、Ruby、および Tcl 用の API が提供されており、MySQL クライアントを多くの言語で記述できます。第23章「Connector および API」 を参照してください。
- Connector/ODBC (MyODBC) インタフェースによって、ODBC (Open DataBase Connectivity) 接続を使用するクライアントプログラムに MySQL サポートが提供されます。たとえば、MS Access を使用して MySQL Server に接続することができます。クライアントは、Windows と Unix のどちらで実行されていてもかまいません。Connector/ODBC ソースが使用可能です。ほかの多くの機能と同様に、ODBC 2.5 のすべての機能がサポートされます。「MySQL Connector/ODBC Developer Guide」を参照してください。
- Connector/J インタフェースは JDBC 接続を使用する Java クライアントプログラムの MySQL サポートを提供しています。クライアントは、Windows と Unix のどちらで実行されていてもかまいません。Connector/J ソースが使用可能です。「MySQL Connector/J 5.1 Developer Guide」を参照してください。
- MySQL Connector/Net により、開発者は MySQL 上でセキュアな高性能データ接続性を要する .NET アプリケーションの作成を容易に行えます。必要な ADO.NET インタフェースを実装し、ADO.NET 対応のツールに統合します。開発者は好みの .NET 言語でアプリケーションを構築できます。MySQL Connector/Net は 100% C# で記述され、完全に管理される ADO.NET ドライバです。「MySQL Connector/NET Developer Guide」を参照してください。

ローカライズ

- サーバーは、クライアントに多数の言語でエラーメッセージを送信することができます。セクション10.2「エラーメッセージ言語の設定」を参照してください。
- latin1 (cp1252)、german、big5、ujis、などのさまざまな文字セットを完全にサポートします。たとえば、スカンジナビア語の文字 「å」、「ä」、および 「ö」 をテーブル名やカラム名で使用できます。ユニコードは MySQL 4.1 以降でサポートされます。
- すべてのデータが、選択した文字セットで保存されます。
- ソートと比較は、選択した文字セットと照合順序に基づいて行われます (デフォルトは latin1 とスウェーデン語の照合順序)。これは、MySQL Server の起動時に変更することができます。非常に高度なソートの例については、チェコ語のソートコードを参照してください。MySQL Server ではさまざまな文字セットがサポートされており、コンパイル時および実行時に指定することができます。
- MySQL 4.1 では、サーバーのタイムゾーンは動的に変更可能で、各クライアントは独自のタイムゾーンを指定できます セクション10.6「MySQL Server でのタイムゾーンのサポート」。

クライアントとツール

- MySQL には複数のクライアントとユーティリティープログラムが含まれます。これには、mysqldump および mysqladmin といったコマンド行プログラム、そして MySQL Workbench などのグラフィックプログラムも含まれます。
- MySQL Server には、テーブルのチェック、最適化、および修復を行う SQL ステートメントのサポートが組み込まれています。これらのステートメントは、mysqlcheck クライアントを介してコマンド行から使用可能です。また、MySQL には、MyISAM テーブルでこれらの操作を実行するための myisamchk という非常に高速なコマンド行ユーティリティーが組み込まれています。第4章「MySQL プログラム」 を参照してください。
- MySQL プログラムを --help または -? オプションを指定して呼び出すと、オンラインヘルプを参照できます。

1.3.3 MySQL の歴史

当初は、高速で低レベルな独自の (ISAM) ルーチンを使用してテーブルに接続するために mSQL データベースシステムを使用するつもりでした。しかし、いくつかのテストを行なった結果、mSQL はそれほど高速でも柔軟でもないため、ニーズに合わないという結論に至りました。これが要因となって、私たちのデータベースへの新しい SQL インタフェースを開発することになりました。ただし、mSQL とほとんど同じ API インタフェースを使用することにしました。この API は、mSQL で使用するために記述されたサードパーティーのコードを MySQL で使用するために簡単に移植できるように設計されています。

MySQL は、共同創設者 Monty Widenius の娘の My にちなんで名づけられました。

MySQL のドルフィン (当社の口ゴ) の名前は 「Sakila,」 です。「ドルフィンのネーミング」コンテストで、ユーザーによって提案された膨大な数の名前の中から選ばれました。採用された名前を提案したのは、アフリカのスワジランド出身の Ambrose Twebaze というオープンソースソフトウェアの開発者でした。Ambrose によると、Sakila という女性の名前は、スワジランドの現地語であるシスワティ語にルーツがあるということです。また、Sakila は、Ambrose の出生国であるウガンダに近い、タンザニアのアルーシャにある町の名前でもあります。

1.4 MySQL 5.6 の新機能

このセクションでは、MySQL 5.6 で追加された機能、非推奨になった機能、および削除された機能について要約しています。

- 追加された機能
- 非推奨になった機能
- 削除された機能

追加された機能

次の機能が、MySQL 5.6 に追加されました。

- セキュリティーの改善　次のセキュリティーの改善が行われました。
  - MySQL に、.mylogin.cnf という名前のオプションファイルに認証証明書を暗号化して格納する方法が用意されました。このファイルを作成するには、mysql_config_editor ユーティリティーを使用します。MySQL クライアントプログラムがあとからこのファイルを読み取って、MySQL Server に接続するための認証証明書を取得することができます。mysql_config_editor は、暗号化を使用して .mylogin.cnf ファイルを作成するため、証明書は平文として格納されず、その内容はクライアントプログラムによって復号化されると、メモリー内でのみ使用されます。このように、パスワードは平文でない形式でファイルに格納でき、コマンド行または環境変数で表示する必要もなくあとから使用できます。詳細は、セクション 4.6.6「mysql_config_editor — MySQL 構成ユーティリティー」を参照してください。
  - MySQL は、SHA-256 パスワードハッシングを実装する sha256_password という名前の認証プラグインを通じて利用できる、ユーザーアカウントパスワードのより強力な暗号化をサポートするようになりました。このプラグインは組み込まれているため、常に使用でき、明示的にロードする必要はありません。SHA-256 パスワードを使用するアカウントを作成する手順などの詳細は、セクション6.3.8.4「SHA-256 認証プラグイン」を参照してください。
  - mysql.user テーブルに password_expired カラムが追加されました。そのデフォルト値は 'N' ですが、新しい ALTER USER ステートメントでは 'Y' に設定できます。アカウントのパスワードの有効期限が切れたあと、そのアカウントを使用したサーバーへの以降の接続で操作を実行すると、ユーザーが SET PASSWORD ステートメントを発行して、新しいアカウントパスワードを確立するまで、すべての場合でエラーになります。詳細は、セクション13.7.1.1「ALTER USER 構文」およびセクション6.3.6「パスワードの期限切れとサンドボックスモード」を参照してください。
  - MySQL にはパスワードセキュリティーのチェック対策が施されました。
    - 平文の値として指定されるパスワードを割り当てるステートメントで、値は現在のパスワードポリシーと照合して検査され、弱い場合は拒否されます (ステートメントは ER_NOT_VALID_PASSWORD エラーを返します)。これは、CREATE USER、GRANT、および SET PASSWORD ステートメントに影響します。PASSWORD() および OLD_PASSWORD() 関数への引数として指定されるパスワードも検査されます。
    - パスワード候補の強度は、新しい VALIDATE_PASSWORD_STRENGTH() SQL 関数を使用して評価できます。これは、パスワード引数を取得して、0 (弱い) から 100 (強い) の整数を返します。

    どちらの機能も validate_password プラグインで実装されます。詳細は、セクション6.1.2.6「パスワード検証プラグイン」を参照してください。
  - mysql_upgrade は、4.1 以前の古いハッシング方法でハッシングされたパスワードを持つユーザーアカウントを見つけた場合、警告を発するようになりました。このようなアカウントは、よりセキュアなパスワードハッシングを使用するように更新する必要があります。セクション6.1.2.4「MySQL でのパスワードハッシュ」を参照してください。
  - Unix プラットフォームでは、mysql_install_db は、よりセキュアな MySQL インストールを実現する新しいオプションである --random-passwords をサポートします。--random-passwords を付けて mysql_install_db

を呼び出すと、ランダムパスワードが MySQL root アカウントに割り当てられ、これらのアカウントに対して「有効期限切れパスワード」フラグが設定され、匿名ユーザー MySQL アカウントが削除されます。追加情報についてはセクション4.4.3「mysql_install_db — MySQL データディレクトリの初期化」を参照してください。

- パスワードが平文で一般クエリーログ、スロークエリーログ、およびバイナリログに書き込まれたステートメントに表示されないように、ロギングが変更されました。セクション6.1.2.3「パスワードおよびロギング」を参照してください。

  mysql クライアントは、パスワードを参照するその履歴ファイルステートメントに記録しないようになりました。セクション4.5.1.3「mysql のロギング」を参照してください。

- START SLAVE 構文は、マスターへの接続に接続パラメータを指定できるように変更されました。これは、master.info ファイルにパスワードを格納する代替方法になります。セクション13.4.2.5「START SLAVE 構文」を参照してください。

- MySQL Enterprise　監査ログプラグインで生成されたファイルの形式は、Oracle Audit Vault との互換性が高くなるように変更されました。セクション6.3.12「MySQL Enterprise Audit ログプラグイン」およびセクション6.3.12.3「監査ログファイル」を参照してください。

  MySQL Enterprise Edition には、SQL レベルでの OpenSSL 機能を公開している OpenSSL ライブラリに基づいて、一連の暗号化関数が含まれるようになりました。これらの関数を使用することによって、エンタープライズアプリケーションが次の操作を実行できるようになります。

  - 公開鍵非対称暗号方式を使用した、追加のデータ保護の実装
  - 公開鍵、秘密鍵、およびデジタル署名の作成
  - 非対称暗号化および非対称復号化の実行
  - デジタル署名およびデータの検証や妥当性検査に対する暗号化ハッシュの使用

  詳細は、セクション12.17「MySQL Enterprise Encryption の関数」を参照してください。

  MySQL Enterprise Edition に含まれる監査ログプラグインには、ユーザーアカウントおよびイベントステータスに基づいて監査イベントをフィルタリングする機能が用意されました。複数の新しいシステム変数は、DBA にフィルタリング制御をもたらします。さらに、複数のステータス変数の追加によって、監査ログプラグインレポート機能が改善されました。詳細は、セクション6.3.12.4「監査ログプラグインのロギング制御」およびセクション6.3.12.7「監査ログプラグインのステータス変数」を参照してください。

- サーバーデフォルトに対する変更　MySQL 5.6.6 以降では、MySQL Server のいくつかのデフォルトパラメータが、前のリリースのデフォルト値と異なっています。これらの変更の目的は、初期設定のままで優れたパフォーマンスを提供し、データベース管理者が設定を手動で変更することの必要性を低下させることです。詳細は、セクション5.1.2.1「サーバーのデフォルト値への変更」を参照してください。

- InnoDB の拡張機能　次の InnoDB の拡張機能が追加されました。

  - InnoDB テーブル上に FULLTEXT インデックスを作成し、MATCH() ... AGAINST 構文を使用してそれらのクエリーを実行できます。この機能には、新しい近似検索演算子 (@) と、複数の新しい構成オプションおよび INFORMATION_SCHEMA テーブルが含まれます。詳細は、セクション14.2.13.3「FULLTEXT インデックス」を参照してください。

  - テーブルをコピーせず、テーブルへの挿入、更新、および削除をブロックせず、またはその両方を行わずに、複数の ALTER TABLE 操作を実行できます。これらの拡張機能はまとめて、オンライン DDL と呼ばれています。詳細は、セクション14.11「InnoDB とオンライン DDL」を参照してください。

  - InnoDB は、MySQL データディレクトリ外の場所で InnoDB file-per-table テーブルスペース (.ibd ファイル) を作成できるようにする、CREATE TABLE ステートメントの DATA DIRECTORY='directory' 句をサポートするようになりました。この拡張機能は、サーバー環境により適切に適合する場所に file-per-table テーブルスペースを作成できる柔軟性をもたらします。たとえば、ビジーテーブルを SSD デバイス上に、または大きなテーブルを大容量 HDD デバイス上に配置できます。

    詳細は、セクション14.5.4「テーブルスペースの位置の指定」を参照してください。

  - InnoDB は、「トランスポータブルテーブルスペース」の概念をサポートするようになり、バッファーされたデータ、進行中のトランザクション、およびスペース ID や LSN などの内部管理詳細によって不整合や不一致が起きることなく、実行中の MySQL インスタンスから file-per-table テーブルスペース (.ibd ファイル) をエクスポートし、別の実行中のインスタンスにインポートできます。

FLUSH TABLE コマンドの FOR EXPORT 句は、未保存のあらゆる変更内容を InnoDB メモリーバッファーから .ibd ファイルに書き込みます。.ibd ファイルと個別のメタデータファイルを別のサーバーにコピーしたあと、ALTER TABLE ステートメントの DISCARD TABLESPACE 句と IMPORT TABLESPACE 句が、別の MySQL インスタンスにテーブルデータをもたらすために使用されます。

この拡張機能は、サーバー環境により適切に適合できるように、file-per-table テーブルスペースを自由に移動できる柔軟性をもたらします。たとえば、ビジーテーブルを SSD デバイス上に、または大きなテーブルを大容量 HDD デバイス上に移動できます。詳細は、セクション14.5.5「テーブルスペースの別のサーバーへのコピー (トランスポータブルテーブルスペース)」を参照してください。

- 非圧縮テーブルの InnoDB ページサイズを、デフォルトの 16K バイトに代わるサイズとして、8K バイトまたは 4K バイトに設定できるようになりました。この設定は、innodb_page_size 構成オプションで制御されます。このサイズは MySQL インスタンスの作成時に指定します。インスタンス内のすべての InnoDB テーブルスペースは同じページサイズを共有します。ページサイズが小さくなると、ワークロードとストレージデバイスとの特定の組み合わせ、特にブロックサイズの小さな SSD デバイスとの組み合わせについて、冗長または非効率的な I/O を回避できるようになります。

- 適応型フラッシュのアルゴリズムに対する改善により、さまざまなワークロード下での I/O 操作の効率性と整合性が向上します。新しいアルゴリズムとデフォルトの構成値は、ほとんどのユーザーにとって、パフォーマンスと並列性を改善すると考えられています。上級ユーザーは、複数の構成オプションを通じて、I/O 応答性を微調整できます。詳細は、セクション14.13.1.6「InnoDB バッファープールのフラッシュのチューニング」を参照してください。

- NoSQL スタイルの API を通じて、InnoDB テーブルにアクセスする MySQL アプリケーションをコード化できます。この機能は、普及している memcached デーモンを使用して、鍵と値のペアに ADD、SET、GET などのリクエストをリレーします。データを格納および取得するこれらの単純な操作により、クエリー実行計画の解析や構築などの SQL オーバーヘッドが回避されます。NoSQL API および SQL を通じて同じデータにアクセスできます。たとえば、高速な更新および検索に NoSQL API を、複雑なクエリーおよび既存のアプリケーションとの互換性に SQL を使用できます。詳細は、セクション14.18「InnoDB と memcached の統合」を参照してください。

- InnoDB テーブルのオプティマイザ統計は、より予測可能な間隔で収集され、サーバーの再起動にまたがって維持でき、計画安定性が向上します。オプティマイザ統計の精度を高め、クエリー実行計画を改善するように、InnoDB インデックスに対して行われるサンプル収集の量を制御することもできます。詳細は、セクション14.13.16.1「永続的オプティマイザ統計のパラメータの構成」を参照してください。

- 新しい最適化は、読み取り専用のトランザクションに適用され、アドホッククエリーとレポート生成アプリケーションのパフォーマンスと並列性を改善します。可能な場合にはこれらの最適化は自動的に適用されます。または、START TRANSACTION READ ONLY を指定してトランザクションが読み取り専用になるようにすることができます。詳細は、セクション14.13.14「InnoDB の読み取り専用トランザクションの最適化」を参照してください。

- InnoDB Undo ログを、システムテーブルスペースから 1 つ以上の個別のテーブルスペースに移動できます。Undo ログの I/O パターンは、ハードディスクストレージにシステムテーブルスペースを保持しながら、これらの新しいテーブルスペースを、SSD ストレージに移動する適切な候補にします。詳細は、セクション14.5.6「個別のテーブルスペースへの InnoDB Undo ログの格納」を参照してください。

- より高速なチェックサムアルゴリズムを有効にする構成オプション innodb_checksum_algorithm=crc32 を指定することによって、InnoDB チェックサム機能の効率性を改善できます。このオプションは innodb_checksums オプションを置き換えます。古いチェックサムアルゴリズム (オプション値 innodb) を使用して書き込まれたデータは完全に上方互換性があります。新しいチェックサムアルゴリズム (オプション値 crc32) を使用して変更されたテーブルスペースは、innodb_checksum_algorithm オプションをサポートしない以前のバージョンの MySQL にダウングレードできません。

- InnoDB Redo ログファイルの最大合計サイズは、4G バイトから 512G バイトに増大しました。innodb_log_file_size オプションで、さらに大きな値を指定できます。既存の Redo ログファイルのサイズが innodb_log_file_size および innodb_log_files_in_group で指定されたサイズと一致しない状況を、起動動作で自動処理するようになりました。

- --innodb-read-only オプションを使用すると、MySQL Server を読み取り専用モードで実行できます。DVD や CD などの読み取り専用メディア上の InnoDB テーブルにアクセスすることも、すべて同じデータディレクトリを共有する複数のインスタンスを含むデータウェアハウスを設定できます。使用法の詳細は、セクション14.3.1「読み取り専用操作用の InnoDB の構成」を参照してください。

- 新しい構成オプション innodb_compression_level では、zlib で使用する 0-9 の範囲から、InnoDB 圧縮テーブルの圧縮レベルを選択できます。更新操作によってページが再度圧縮されるときに、バッファープール内の圧縮ページが Redo ログに格納されるかどうかを制御することもできます。この動作は、innodb_log_compressed_pages 構成オプションで制御されます。

- InnoDB 圧縮テーブル内のデータブロックには、一定量の空スペース (パディング) が含まれ、DML 操作は、新しい値を再圧縮することなく、行データを変更できます。パディングが多すぎると、大きな変更後にデータを再圧縮する必要があるときに、圧縮が失敗する可能性が増え、ページの分割が必要になる可能性があります。パディングの量を動的に調整できるようになったため、DBA は、新しいパラメータでテーブル全体を再作成したり、ページサイズの異なるインスタンス全体を再作成したりすることなく、圧縮の失敗の割合を低減できるようになりました。関連した新しい構成オプションは、innodb_compression_failure_threshold_pct、innodb_compression_pad_pct_max です。

- 複数の新しい InnoDB 関連の INFORMATION_SCHEMA テーブルは、InnoDB バッファープールに関する情報、テーブル、インデックス、および InnoDB データディクショナリからの外部キーに関するメタデータ、パフォーマンススキーマテーブルからの情報を補完するパフォーマンスメトリックに関する低レベル情報を提供します。

- 膨大な数のテーブルを含むシステムでのメモリーロードを簡単にするために、InnoDB は、もっとも長い間アクセスされずに経過したテーブルを選択する LRU アルゴリズムを使用して、開いているテーブルに関連付けられたメモリーを解放するようになりました。開いた InnoDB テーブルのメタデータを保持するためにより多くのメモリーを予約するには、table_definition_cache 構成オプションの値を増やしてください。InnoDB は、InnoDB データディクショナリキャッシュ内の開いているテーブルインスタンスの数に関する「ソフト制限」として、この値を扱います。追加情報については、table_definition_cache のドキュメントを参照してください。

- InnoDB には、カーネルミューテックスの分割による競合の軽減、メインスレッドから個別のスレッドへのフラッシュ操作の移動、複数のパージスレッドの有効化、大容量メモリーシステムでのバッファープールの競合の軽減などの、複数の内部パフォーマンス拡張機能があります。

- InnoDB は、高速な新しいアルゴリズムを使用して、deadlocks を検出します。すべての InnoDB デッドロックに関する情報は、MySQL Server エラーログに書き込むことができ、アプリケーション問題の診断に役立ちます。

- サーバー再起動後の長時間のウォームアップを回避するために、特に大きな InnoDB バッファープールを伴うインスタンスの場合は、再起動直後にバッファープールにページをリロードできます。MySQL は、シャットダウン時にコンパクトなデータファイルをダンプし、続いてそのデータファイルを参照して、次回の再起動時にリロードするページを見つけることができます。ベンチマーキング中や複雑なレポート生成クエリー後など、いつでもバッファープールを手動でダンプまたはリロードすることもできます。詳細は、セクション14.13.1.5「再起動を高速化するための InnoDB バッファープールのプリロード」を参照してください。

- MySQL 5.6.16 以降、innochecksum は 2G バイト以上のサイズのファイルのサポートに対応しています。以前は、innochecksum は 2G バイトまでのサイズのファイルだけをサポートしていました。

- MySQL 5.6.16 以降では、新しいグローバル構成パラメータ innodb_status_output および innodb_status_output_locks を使用すると、標準の InnoDB Monitor および InnoDB Lock Monitor を動的に有効および無効にすることができます。特別な名前が付けられたテーブルを作成および削除することによって、定期的な出力のモニターを有効および無効にする方法は非推奨であり、今後のリリースで削除される可能性があります。追加情報については、セクション14.15「InnoDB モニター」を参照してください。

- MySQL 5.6.17 以降、MySQL は、次の操作にオンライン DDL (ALGORITHM=INPLACE) を使用した通常のパーティション化された InnoDB テーブルの再構築をサポートします。

  - OPTIMIZE TABLE
  - ALTER TABLE ... FORCE
  - ALTER TABLE ... ENGINE=INNODB (InnoDB テーブルで実行した場合)

  オンライン DDL のサポートにより、テーブル再構築時間が短縮し、ユーザーアプリケーションの停止時間の短縮に役立つ並列 DML が可能になります。詳細は、セクション14.11.1「オンライン DDL の概要」を参照してください。

- パーティション化　次のテーブルパーティション化拡張機能が追加されました。
  - パーティションの最大数は 8192 まで増加しています。これは、テーブルのすべてのパーティションとすべてのサブパーティションを含めた数字です。
  - ALTER TABLE ... EXCHANGE PARTITION ステートメントを使用して、パーティション化されたテーブルのパーティションまたはサブパーティション化されたテーブルのサブパーティションを、パーティション化されていないことを除けば同じ構造のテーブルと交換できるようになりました。これはたとえば、パーティションのインポートおよびエクスポートに使用できます。詳細および例については、セクション19.3.3「パーティションとサブパーティションをテーブルと交換する」を参照してください。
  - パーティション化されたテーブルで機能するクエリーや多数のデータ変更ステートメントで、1 つ以上のパーティションまたはサブパーティションの明示的な選択がサポートされるようになりました。たとえば、整数カラム c を含むテーブル t に、p0、p1、p2、および p3 という名前の 4 つのパーティションがあるとします。この場合、クエリー SELECT * FROM t PARTITION (p0, p1) WHERE c < 5 は、パーティション p0 および p1 から、c が 5 未満である行だけを返します。

    次のステートメントは、明示的なパーティション選択をサポートします。
    - SELECT
    - DELETE
    - INSERT
    - REPLACE
    - UPDATE
    - LOAD DATA。
    - LOAD XML。

    構文については、個々のステートメントの説明を参照してください。追加情報および例については、セクション19.5「パーティション選択」を参照してください。
  - パーティションロックプルーニングは、多くのパーティションを含むテーブル上で機能する多数の DML および DDL ステートメントのパフォーマンスを大幅に改善しますが、これは、これらのステートメントの影響を受けないパーティション上のロックを除去できるようにすることによって行います。このようなステートメントには、SELECT、SELECT ... PARTITION、UPDATE、REPLACE、INSERT などの多数のステートメントが含まれます。この結果パフォーマンスが改善されたステートメントの全リストなどの詳細は、セクション19.6.4「パーティショニングとロック」を参照してください。
- パフォーマンススキーマ　パフォーマンススキーマには、次のような複数の新機能が含まれています。
  - テーブル入力および出力のインストゥルメンテーション。インストゥルメントされた操作には、永続的ベーステーブルまたは一時テーブルへの行レベルのアクセスが含まれます。行に影響する操作は、フェッチ、挿入、更新、および削除です。
  - スキーマ名またはテーブル名に基づいた、テーブルによるイベントフィルタリング。
  - スレッドによるイベントフィルタリング。スレッドに関する詳細が収集されます。
  - テーブルおよびインデックス I/O 用とテーブルロック用のサマリーテーブル。
  - ステートメントとステートメント内のステージのインストゥルメンテーション。
  - サーバーの起動時のインストゥルメントとコンシューマの構成。以前は実行時にのみ可能でした。
- MySQL Cluster　MySQL Cluster は個別の製品としてリリースされます。最新の GA リリースは MySQL 5.6 に基づき、バージョン 7.3 の NDB ストレージエンジンを使用しています。クラスタリングのサポートは、主要な MySQL Server 5.6 リリースでは利用できません。MySQL Cluster NDB 7.3 の詳細は、第18章「MySQL Cluster NDB 7.3 および MySQL Cluster NDB 7.4」を参照してください。最新の開発バージョンは、バージョン 7.4 の NDB ストレージエンジンと MySQL Server 5.6 に基づいた、MySQL Cluster NDB 7.4 です。現

在、MySQL Cluster NDB 7.4 はテストおよび評価のために使用できます。最新の MySQL Cluster NDB 7.4 リリースは https://dev.mysql.com/downloads/cluster/ から入手できます。

詳細および MySQL Cluster NDB 7.4 で行われた改善の概要については、セクション18.1.4.2「MySQL Cluster NDB 7.4 での MySQL Cluster の開発」を参照してください。

- レプリケーションとロギング　次のレプリケーション拡張機能が追加されました。

  - MySQL は、グローバルトランザクション識別子 (「GTID」とも呼ばれます) を使用したトランザクションベースのレプリケーションをサポートするようになりました。これにより、発信サーバー上でコミットされ、いずれかのスレーブによって適用されたときに、各トランザクションの識別と追跡が可能になります。

    レプリケーションの設定で GTID を有効にする場合、主に --gtid-mode および --enforce-gtid-consistency の新しいサーバーオプションを使用して行います。GTID のサポートで導入された追加オプションおよび変数の詳細は、セクション17.1.4.5「グローバルトランザクション ID のオプションと変数」を参照してください。

    GTID を使用する場合、新しいスレーブを開始したり、新しいマスターにフェイルオーバーするときに、ログファイルやこれらのファイル内での位置を参照したりする必要はありません。このため、これらのタスクが非常に簡略化されます。バイナリログファイルを参照して、または参照せずに、GTID レプリケーションに対してサーバーをプロビジョニングする場合の詳細は、セクション17.1.3.3「フェイルオーバーおよびスケールアウトでの GTID の使用」を参照してください。

    GTID ベースのレプリケーションは完全にトランザクションベースであるため、マスターとスレーブの一貫性のチェックが簡単になります。所定のマスター上でコミットされたすべてのトランザクションが、所定のスレーブ上でもコミットされている場合、2 台のサーバー間の一貫性は保証されます。

    MySQL Replication での GTID の実装および使用についてさらに詳しい情報が必要な場合は、セクション17.1.3「グローバルトランザクション識別子を使用したレプリケーション」を参照してください。

  - MySQL の行ベースのレプリケーションは、行イメージコントロールをサポートするようになりました。各行の変更に対し、すべてのカラムではなく、それぞれの行での変更を一意に識別および実行するために必要なカラムだけを記録することによって、ディスク容量、ネットワークリソース、およびメモリー使用量を節約できます。binlog_row_image サーバーシステム変数を、minimal (必要なカラムだけを記録)、full (すべてのカラムを記録)、noblob (不要な BLOB または TEXT カラムを除いたすべてのカラムを記録) のいずれかの値に設定することによって、すべての行を記録するか、最低限の行を記録するかを指定できます。詳細は、バイナリロギングで使用されるシステム変数を参照してください。

  - MySQL Server で書き込まれ、読み取られるバイナリログは、完全なイベント (トランザクション) だけが記録されたり読み戻されたりするため、クラッシュセーフになりました。デフォルトで、サーバーはイベント自体のほかにイベントの長さを記録し、この情報を使用して、イベントが正しく書き込まれたことを検証します。binlog_checksum システム変数を設定することによって、サーバーで、CRC32 チェックサムを使用してイベントのチェックサムを書き込ませることもできます。サーバーにバイナリログからチェックサムを読み取らせるには、master_verify_checksum システム変数を使用します。--slave-sql-verify-checksum システム変数は、スレーブ SQL スレッドに、リレーログからチェックサムを読み取らせます。

  - MySQL は、テーブルおよびファイルへのマスター接続情報のロギングと、スレーブリレーログ情報のロギングをサポートするようになりました。これらのテーブルの使用は、--master-info-repository と --relay-log-info-repository のサーバーオプションによって別々に制御できます。--master-info-repository を TABLE に設定すると、接続情報が slave_master_info テーブルに記録されます。--relay-log-info-repository を TABLE に設定すると、リレーログ情報が slave_relay_log_info テーブルに記録されます。これらのテーブルはどちらも、mysql システムデータベース内に自動的に作成されます。

    レプリケーションをクラッシュセーフにするには、slave_master_info テーブルと slave_relay_log_info テーブルがそれぞれ、トランザクションストレージエンジンを使用する必要があります。この理由のため、MySQL 5.6.6 以降では、これらのテーブルは InnoDB を使用して作成されます。(Bug #13538891) これらの両方のテーブルで MyISAM を使用している以前の MySQL 5.6 リリースを使用している場合は、レプリケーションがクラッシュセーフであることを望むならば、レプリケーションを開始する前に、両方をトランザクションストレージエンジン (InnoDB など) に変換する必要があることになります。このような場合は、適切な ALTER TABLE ... ENGINE=... ステートメントを使用して、これを行えます。レプリケーションが実際に実行している間、これらのテーブルのどちらかが使用するストレージエンジンを変更しようとしないでください。

    詳細は、クラッシュセーフレプリケーションを参照してください。

  - mysqlbinlog には、そのオリジナルのバイナリ形式でバイナリログをバックアップする機能が用意されました。mysqlbinlog は、--read-from-remote-server オプションと --raw オプションを付けて呼び出された場合、

サーバーに接続し、ログファイルをリクエストし、元のファイルと同じ形式で出力ファイルを書き込みます。セクション4.6.8.3「バイナリログファイルのバックアップのための mysqlbinlog の使用」を参照してください。

- MySQL は、スレーブサーバーが少なくとも指定した時間だけ意図的にマスターに遅れるような遅延レプリケーションをサポートするようになりました。デフォルトの遅延は 0 秒です。CHANGE MASTER TO の新しい MASTER_DELAY オプションを使用して、遅延を設定します。

  遅延レプリケーションは、マスターでのユーザーの誤りから保護したり (DBA は遅延スレーブを災害の直前の時間までロールバックできます)、遅れがあるときのシステムの動作をテストしたりするなどの目的に使用できます。セクション17.3.9「遅延レプリケーション」を参照してください。

- CHANGE MASTER TO ステートメントの発行時に MASTER_BIND オプションを使用することによって、複数のネットワークインタフェースがあるレプリケーションスレーブに、これらのうち 1 つだけを使用させられるようになりました。

- log_bin_basename システム変数が追加されました。この変数には、完全なファイル名と、バイナリログファイルへのパスが含まれます。log_bin システム変数はバイナリロギングが有効かどうかだけを示しますが、log_bin_basename は --log-bin サーバーオプションで設定された名前を反映します。

  同様に、relay_log_basename システム変数は、ファイル名と、リレーログファイルへの完全パスを示します。

- MySQL Replication は、スレーブ上でのマルチスレッドを使用したトランザクションの並列実行をサポートするようになりました。並列実行が有効な場合、スレーブ SQL スレッドは、slave_parallel_workers サーバーシステム変数の値によって決定される、多数のスレーブワーカースレッドに対する調整役として機能します。スレーブでのマルチスレッドの現在の実装は、データと更新がデータベースごとにパーティション化されていることと、所定のデータベース内の更新が、マスター上の場合と同じ相対的順序で行われることを前提としています。ただし、異なるデータベース間でトランザクションを調整する必要はありません。この場合、トランザクションもデータベースごとに配分できます。つまり、スレーブ上のワーカースレッドは、別のデータベースへの更新が完了するまで待機せずに、所定のデータベースで連続してトランザクションを処理できます。

  別々のデータベースに対するトランザクションは、スレーブとマスターとでは実行順序が異なることがあるため、単に直近で実行したトランザクションを確認するだけでは、マスター上の以前のすべてのトランザクションがスレーブ上で実行されたという保証にはなりません。これは、マルチスレッド化したスレーブを使用する場合にロギングとリカバリに影響します。スレーブ上でマルチスレッドを使用したときにバイナリロギング情報をどのように解釈すればよいかについては、セクション13.7.5.35「SHOW SLAVE STATUS 構文」を参照してください。

- オプティマイザの拡張機能　クエリーオプティマイザで次の改善が実装されました。

  - オプティマイザは、次の形式のクエリー (およびサブクエリー) をより効率よく処理できるようになりました。

    ```
    SELECT ... FROM single_table ... ORDER BY non_index_column [DESC] LIMIT [M,]N;
    ```

    この種のクエリーは、大きな結果セットの数行だけを表示する Web アプリケーションで一般的なものです。例:

    ```
    SELECT col1, ... FROM t1 ... ORDER BY name LIMIT 10;
    SELECT col1, ... FROM t1 ... ORDER BY RAND() LIMIT 15;
    ```

    ソートバッファーには、sort_buffer_size のサイズが入ります。N 行のソート要素が、ソートバッファー (M が指定された場合は M+N 行) に収まる程度に小さい場合は、サーバーはマージファイルの使用を避けて、メモリー内全体でソートを実行できます。詳細は、セクション8.2.1.19「LIMIT クエリーの最適化」を参照してください。

  - オプティマイザは、Disk-Sweep Multi-Range Read を実装します。セカンダリインデックスでの範囲スキャンを使用して行を読み取ると、テーブルが大きく、ストレージエンジンのキャッシュに格納されていない場合、ベーステーブルへのランダムディスクアクセスが多発する結果になることがあります。Disk-Sweep Multi-Range Read (MRR) 最適化を使用すると、MySQL は、最初にインデックスだけをスキャンし、該当する行のキーを収集することによって、範囲スキャンのランダムディスクアクセスの回数を軽減しようとします。続いてキーがソートされ、最後に主キーの順序を使用してベーステーブルから行が取得されます。Disk-Sweep MRR の目的は、ランダムディスクアクセスの回数を減らし、その代わりに、ベーステーブルデータの順次スキャンを増やすことです。詳細は、セクション8.2.1.13「Multi-Range Read の最適化」を参照してください。

- オプティマイザは、MySQL がインデックスを使用してテーブルから行を取得する場合の最適化として、Index Condition Pushdown (ICP) を実装します。ICP を使用しない場合、ストレージエンジンはインデックスをトラバースして、ベーステーブル内で行を検索し、MySQL Server に返し、MySQL Server が行に対して WHERE 条件を評価します。ICP を有効にすると、インデックスからのフィールドだけを使用して WHERE 条件の部分を評価できる場合は、MySQL Server はこの WHERE 条件の部分をストレージエンジンにプッシュダウンします。ストレージエンジンは続いて、インデックスエントリを使用することによって、プッシュされたインデックス条件を評価し、これが満たされた場合にのみ、ベース行が読み取られます。ICP は、ストレージエンジンがベーステーブルに対して行う必要のあるアクセスの回数と、MySQL Server がストレージエンジンに対して行う必要のあるアクセスの回数を軽減できます。詳細は、セクション8.2.1.6「インデックスコンディションプッシュダウンの最適化」を参照してください。

- EXPLAIN ステートメントは、DELETE、INSERT、REPLACE、および UPDATE のステートメントの実行プラン情報を提供するようになりました。以前には、EXPLAIN は SELECT ステートメントの情報だけを提供していました。さらに、EXPLAIN ステートメントは JSON 形式で出力を生成できるようになりました。セクション13.8.2「EXPLAIN 構文」を参照してください。

- FROM 句内でのサブクエリー (つまり派生テーブル) のオプティマイザによる処理の効率性が向上しました。FROM 句内のサブクエリーのマテリアライズはクエリー実行中にその内容が必要になるまで延期されるため、パフォーマンスが向上します。さらに、クエリー実行中に、オプティマイザは派生テーブルにインデックスを追加して、そこからの行の取得速度を上げることができます。詳細は、FROM 句内のサブクエリー (派生テーブル) の最適化を参照してください。

- オプティマイザは、準結合とマテリアライズ戦略を使用して、サブクエリーの実行を最適化します。準結合変換によるサブクエリーの最適化およびサブクエリー実体化によるサブクエリーの最適化を参照してください。

- 結合したテーブルと結合バッファーの両方にインデックスアクセスを使用する、Batched Key Access (BKA) 結合アルゴリズムが利用できるようになりました。BKA アルゴリズムは、ネスト外部結合とネスト準結合を含め、内部結合、外部結合、および準結合操作をサポートします。BKA には、テーブルスキャンの効率性の向上による結合パフォーマンスの改善というメリットもあります。詳細は、セクション8.2.1.14「Block Nested Loop 結合と Batched Key Access 結合」を参照してください。

- オプティマイザに、主に開発者が使用するためのトレース機能が装備されました。一連の optimizer_trace_xxx システム変数と INFORMATION_SCHEMA.OPTIMIZER_TRACE テーブルによってインタフェースが提供されます。詳細については、「MySQL Internals: Tracing the Optimizer」を参照してください。

- 条件処理　MySQL は GET DIAGNOSTICS ステートメントをサポートするようになりました。GET DIAGNOSTICS は、以前の SQL ステートメントが例外を生成したかどうかや、それが何だったかなどの情報を診断領域から取得するための標準化された方法をアプリケーションにもたらします。詳細は、セクション13.6.7.3「GET DIAGNOSTICS 構文」を参照してください。

  さらに、MySQL の動作が標準 SQL にさらに類似するように、条件ハンドラ処理ルールでの複数の欠陥が修正されました。

  - 選択するハンドラを決定するときにブロックスコープが使用されます。以前には、ストアドプログラムが、ハンドラ選択の単一のスコープを持つものとして扱われていました。

  - 条件の優先順位は、より正確に解決されるようになりました。

  - 診断領域のクリアーが変更されました。Bug #55843 のために、ハンドラを有効にする前に、処理される条件が診断領域からクリアーされていました。これにより、条件の情報がハンドラ内で利用できなくなっていました。現在、条件情報はハンドラで利用できるようになり、ハンドラは GET DIAGNOSTICS ステートメントでこの情報を調べることができます。ハンドラの実行中にまだクリアーされていない条件情報は、ハンドラが終了するときにクリアーされます。

  - 以前には、ハンドラは、条件が発生するとすぐに有効になっていました。現在、条件が発生したステートメントの実行が終了するまで有効にならず、終了した時点でもっとも適切なハンドラが選択されるようになりました。これは、ステートメントの実行中にあとから発生した条件がその前に発生した条件より高い優先順位を持ち、どちらの条件のハンドラも同じスコープにある場合、複数の条件を発生させるステートメントに影響することがあります。以前には、最初に発生した条件のハンドラが、ほかのハンドラより優先順位が低い場合でも、選択されていました。現在、優先順位がもっとも高い条件のハンドラが、ステートメントで最初に発生した条件でない場合でも、選択されるようになりました。

  詳細は、セクション13.6.7.6「ハンドラのスコープに関するルール」を参照してください。

- データ型　次のデータ型の変更が実装されました。
  - MySQL では、マイクロ秒 (6 桁) までの精度を持つ TIME、DATETIME、および TIMESTAMP 値の小数秒に対応できるようになりました。セクション11.3.6「時間値での小数秒」を参照してください。
  - 以前には、テーブルごとに最大 1 つの TIMESTAMP カラムを自動的に初期化するか、現在の日時に更新することが可能でした。この制限は撤廃されました。どの TIMESTAMP カラム定義にも、DEFAULT CURRENT_TIMESTAMP 句と ON UPDATE CURRENT_TIMESTAMP 句の任意の組み合わせを指定できます。さらに、これらの句は、DATETIME カラム定義で使用できるようになりました。詳細は、セクション11.3.5「TIMESTAMP および DATETIME の自動初期化および更新機能」を参照してください。
  - MySQL では、TIMESTAMP データ型は、自動的な初期化および更新属性のデフォルト値と割り当ての点で、ほかのデータ型とは非標準的な方法で異なります。これらの動作はデフォルトのままですが、現在、非推奨になっており、サーバーの起動時に explicit_defaults_for_timestamp システム変数を有効にすることによってオフにすることができます。セクション11.3.5「TIMESTAMP および DATETIME の自動初期化および更新機能」およびセクション5.1.4「サーバーシステム変数」を参照してください。
- ホストキャッシュ　MySQL では、クライアントがサーバーに接続するときに生じたエラーの原因に関する詳細が提供されるようになるとともに、クライアント IP アドレスとホスト名情報を含み DNS ルックアップを回避するために使用されるホストキャッシュへのアクセスが改善されました。次の変更が実装されました。
  - 新しい Connection_errors_xxx ステータス変数は、特定のクライアント IP アドレスに適用されない接続エラーに関する情報を提供します。
  - 特定の IP アドレスに適用されるエラーを追跡するために、ホストキャッシュにカウンタが追加され、新しい host_cache パフォーマンススキーマテーブルが、SELECT ステートメントを使用して調べられるようにホストキャッシュの内容を公開します。ホストキャッシュの内容にアクセスすると、どれだけの数のホストがキャッシュされているか、どのホストにどの種類の接続エラーが発生しているか、またはホストエラーカウントが max_connect_errors システム変数の上限にどれだけ近づいているかなどの質問に回答することができます。
  - ホストキャッシュサイズは、host_cache_size システム変数を使用して構成できるようになりました。

  詳細は、セクション8.11.5.2「DNS ルックアップの最適化とホストキャッシュ」およびセクション22.9.10.1「host_cache テーブル」を参照してください。
- OpenGIS　OpenGIS 仕様は、2 つの幾何値の関係をテストする関数を定義します。MySQL は当初、オブジェクトバウンディング矩形を使用し、対応する MBR ベースの関数と同じ結果を返すように、これらの関数を実装しました。正確なオブジェクトの形状を使用する、対応するバージョンが利用できるようになりました。これらのバージョンの名前には ST_ プリフィクスが付けられています。たとえば、Contains() はオブジェクトバインディング矩形を使用しますが、ST_Contains() はオブジェクト形状を使用します。詳細は、セクション12.15.9「幾何オブジェクト間の空間関係をテストする関数」を参照してください。

非推奨になった機能

次の機能は、MySQL 5.6 で非推奨になり、今後のリリースで削除される可能性または予定があります。ほかのオプションがあれば、それを利用するためにアプリケーションを更新する必要があります。

- ERROR_FOR_DIVISION_BY_ZERO、NO_ZERO_DATE、および NO_ZERO_IN_DATE の SQL モードは非推奨になり、これらのいずれかを含むように sql_mode 値を設定すると警告が生成されます。MySQL 5.7 では、これらのモードは何も行いません。代わりに、その効果は、厳密 SQL モード (STRICT_ALL_TABLES または STRICT_TRANS_TABLES) の効果に含まれます。MySQL 5.7 での変更の目的は、効果が厳密モードに依存した SQL モードの数を減らし、これらを厳密モード自体の一部にすることです。

  MySQL 5.7 へのアップグレードの高度な準備を行うには、SQL Mode Changes in MySQL 5.7を参照してください。その説明では、アプリケーションが MySQL 5.7 での SQL モードの変更によって影響されるかどうかを評価するためのガイドラインが示されます。
- MySQL 5.6 における暗黙の GROUP BY ソートへの依存は、非推奨になっています。グループ化された結果の特定のソート順序を実現するには、明示的な ORDER BY 句を使用することをお勧めします。GROUP BY ソートは、たとえば、オプティマイザがもっとも効率的であると考えるどのような方法でも、グループ化を指示できるようにしたり、ソートオーバーヘッドを回避したりするためなどに、今後のリリースで変更される可能性のある MySQL 拡張機能です。
- 4.1 以前のパスワードと mysql_old_password 認証プラグイン。MySQL 4.1 より前で使用される古いハッシュ形式で格納されているパスワードは、ネイティブのパスワードハッシュ方式を使用するパスワードよりもセキュアでないため、使用しないようにしてください。4.1 以前のパスワードと mysql_old_password 認証プラグイン

が非推奨になりました。4.1 より前のパスワードハッシュを持つアカウントを使用して接続することを防ぐために、secure_auth システム変数はデフォルトで有効になりました。(このようなパスワードハッシュを持つアカウントに対して接続を許可するには、--secure_auth=0 を使用してサーバーを起動してください。ただし、4.1 以前のパスワードは非推奨であるため、secure_auth を無効にすることも非推奨です。)

データベース管理者は、mysql_old_password 認証プラグインを使用するアカウントを、代わりに mysql_native_password を使用するように変換することが推奨されます。アカウントのアップグレード手順については、セクション6.3.8.3「4.1 よりも前のパスワードハッシュ方式と mysql_old_password プラグインからの移行」を参照してください。

OLD_PASSWORD() 関数は 4.1 以前のパスワードハッシュを生成しますが、これは、old_passwords システム変数が 1 に設定されている場合に PASSWORD() が行う処理と同じです。OLD_PASSWORD() と old_passwords=1 は非推奨になっています。

- --skip-innodb オプションとそのシノニム (--innodb=OFF、--disable-innodb など)。
- date_format、datetime_format、および time_format システム変数 (これらは使用されていません)。
- have_profiling、profiling、および profiling_history_size システム変数。
- innodb_use_sys_malloc および innodb_additional_mem_pool_size システム変数。
- timed_mutexes システム変数。これは何も行わず、何の効果もありません。
- --language オプション。代わりに --lc-messages-dir オプションと --lc-messages オプションを使用してください。
- ALTER TABLE の IGNORE 句。ALTER IGNORE TABLE は、レプリケーションの問題を引き起こし、一意のインデックスの作成用のオンライン ALTER TABLE を妨げ、外部キーの問題 (親テーブルで削除された行) を引き起こします。
- msql2mysql、mysql_convert_table_format、mysql_find_rows、mysql_fix_extensions、mysql_setpermission、mysql_waitp および mysqlbug ユーティリティー。
- mysqlhotcopy ユーティリティー。代替方法には、mysqldump と MySQL Enterprise Backup があります。

削除された機能

次の項目は廃止され、MySQL 5.6 で削除されました。ほかのオプションがあれば、それを利用するためにアプリケーションを更新する必要があります。

- --log サーバーオプションと log システム変数。代わりに、一般クエリーログを有効にするには --general_log オプションを使用し、一般クエリーログファイル名を設定するには --general_log_file=file_name オプションを使用してください。
- --log-slow-queries サーバーオプションと log_slow_queries システム変数。代わりに、スロークエリーログを有効にするには --slow_query_log オプションを使用し、スロークエリーログファイル名を設定するには --slow_query_log_file=file_name オプションを使用してください。
- --one-thread サーバーオプション。代わりに --thread_handling=no-threads を使用してください。
- --safe-mode サーバーオプション。
- --skip-thread-priority サーバーオプション。
- --table-cache サーバーオプション。代わりに table_open_cache システム変数を使用してください。
- --init-rpl-role オプションと --rpl-recovery-rank オプション、rpl_recovery_rank システム変数、および Rpl_status ステータス変数。
- engine_condition_pushdown システム変数。optimizer_switch 変数の engine_condition_pushdown フラグを代わりに使用します。
- have_csv、have_innodb、have_ndbcluster、および have_partitioning システム変数。代わりに SHOW PLUGINS を使用するか、INFORMATION_SCHEMA データベースの PLUGINS テーブルをクエリーします。
- sql_big_tables システム変数。代わりに big_tables を使用してください。
- sql_low_priority_updates システム変数。代わりに low_priority_updates を使用してください。

- sql_max_join_size システム変数。代わりに max_join_size を使用してください。
- max_long_data_size システム変数。代わりに max_allowed_packet を使用してください。
- FLUSH MASTER および FLUSH SLAVE ステートメント。代わりに RESET MASTER および RESET SLAVE ステートメントを使用してください。
- SLAVE START および SLAVE STOP ステートメント。START SLAVE および STOP SLAVE ステートメントを使用してください。
- SHOW AUTHORS および SHOW CONTRIBUTORS ステートメント。
- SET ステートメントの OPTION および ONE_SHOT 修飾子。
- 値 DEFAULT をストアドプロシージャーまたはストアドファンクションのパラメータや、ストアドプログラムのローカル変数 (SET var_name = DEFAULT ステートメント) に割り当てることは、明示的に禁じられています。DEFAULT をシステム変数に割り当てることは、以前と同様に許可されています。
- ほとんどの SHOW ENGINE INNODB MUTEX 出力は 5.6.14 で削除されます。SHOW ENGINE INNODB MUTEX 出力は、MySQL 5.7.2 で完全に削除されます。パフォーマンススキーマテーブル上にビューを作成することによって、比較可能な情報を生成できます。

1.5 MySQL の情報源

このセクションでは、MySQL メーリングリスト、ユーザーフォーラム、IRC (インターネットリレーチャット) など、役に立つと思われる情報を提供します。

1.5.1 MySQL メーリングリスト

このセクションでは MySQL メーリングリストの概要を説明するとともに、メーリングリストの使用ガイドラインを提供します。メーリングリストを購読すると、メーリングリストに投稿されたすべてのメッセージを電子メールとして受け取ることができます。自分の質問や回答をリストに送信することもできます。

このセクションに記載されているメーリングリストの購読を開始または購読を中止するには、http://lists.mysql.com/ にアクセスしてください。ほとんどは、個別のメッセージを得られる通常版、または毎日 1 つの大きなメッセージを受け取れるダイジェスト版から選択できます。

購読または購読中止に関するメッセージはいずれのメーリングリストにも送信しないでください。送信した場合、そのメッセージを購読する多くのユーザーへ自動的に配信されてしまいます。

あなたのローカルサイトには MySQL メーリングリストに登録した多くの購読者がいるかもしれません。その場合、ローカルなメーリングリストがあって、lists.mysql.com からサイトに送信されたメッセージはそのローカルリストに送信されるようになっている場合があります。その際は、ローカルの MySQL リストへの追加または削除については、あなたのシステム管理者にお問い合わせください。

メールプログラム内の個別のメールボックスにメーリングリストが送信されるようにするには、メッセージヘッダーから判別するようにフィルタを設定してください。List-ID: または Delivered-To: ヘッダーを使用して、リストのメッセージを識別することができます。

MySQL メーリングリストは次のとおりです。

- announce

  新しいバージョンの MySQL および関連するプログラムの通知に使用されます。これは量の少ないリストで、すべての MySQL ユーザーが購読するべきです。

- mysql

  MySQL の一般的な議論に使用される主要なリストです。議題によっては、より細分化されたメーリングリストで議論を行なった方がよい場合があります。適切でないリストに投稿すると、回答が得られないことがあります。

- bugs

  MySQL の最新リリース以降に報告された問題を常に把握しておきたいユーザーや、バグの検出と修正の過程に積極的に参加したいユーザーのためのリストです。セクション1.6「質問またはバグをレポートする方法」を参照してください。

- internals

MySQL コードに携わる人を対象とするリストです。MySQL の開発に関する議論およびパッチ投稿を行うフォーラムでもあります。

- mysqldoc

  MySQL のドキュメント化に携わる人を対象とするリストです。

- benchmarks

  パフォーマンスに関する問題に関心がある人を対象としています。データベースのパフォーマンス (MySQL に限らない) に関する議論が中心ですが、カーネル、ファイルシステム、ディスクシステムのパフォーマンスなど、より広範なカテゴリも含まれます。

- packagers

  MySQL のパッケージングと配布に関する議論に関するリストです。これは、MySQL のパッケージングや、サポートするすべてのプラットフォームとオペレーティングシステム上でできるかぎり同じような MySQL の外観と操作性を確保することに関するアイデアを交換するため、配布管理者によって使用されるフォーラムです。

- java

  MySQL Server と Java に関する議論に使用されます。ほとんどが、MySQL Connector/J などの JDBC ドライバに関する議論に使用されています。

- win32

  Microsoft のオペレーティングシステム (Windows 9x、Me、NT、2000、XP、2003 など) で実行される MySQL ソフトウェアに関するすべてのトピックに使用されます。

- myodbc

  ODBC を使用した MySQL Server への接続に関するすべてのトピックに使用されます。

- gui-tools

  MySQL のグラフィカルユーザーインタフェースツール、たとえば MySQL Workbench に関するすべてのトピックに使用されます。

- cluster

  MySQL Cluster の議論に使用されます。

- dotnet

  MySQL Server および .NET プラットフォームに関する議論に使用されます。ほとんどが MySQL Connector/Net に関連するものです。

- plusplus

  MySQL への C++ API を使用したプログラミングに関するすべてのトピックに使用されます。

- perl

  MySQL の DBD::mysql を使用した Perl サポートに関するすべてのトピックに使用されます。

質問に対する回答が MySQL のメーリングリストまたはフォーラムから得られなかった場合、Oracle から有料サポートを受ける方法があります。これにより、MySQL の開発者と連絡を直接とることができます。

次の MySQL メーリングリストは英語以外のものです。これらのリストは Oracle による運営ではありません。

- <mysql-france-subscribe@yahoogroups.com>

  フランス語のメーリングリスト。

- <list@tinc.net>

  韓国語のメーリングリスト。購読するには、このメーリングリストに subscribe mysql your@email.address と書いた電子メールを送信してください。

- <mysql-de-request@lists.4t2.com>

  ドイツ語のメーリングリスト。購読するには、このメーリングリストに subscribe mysql-de your@email.address と書いた電子メールを送信してください。このメーリングリストに関する詳細な情報は、http://www.4t2.com/mysql/ を参照してください。

- <mysql-br-request@listas.linkway.com.br>

  ポルトガル語のメーリングリスト。購読するには、このメーリングリストに subscribe mysql-br your@email.address と書いた電子メールを送信してください。

- <mysql-alta@elistas.net>

  スペイン語のメーリングリスト。購読するには、このメーリングリストに subscribe mysql your@email.address と書いた電子メールを送信してください。

1.5.1.1 メーリングリストを使用する際のガイドライン

HTML モードがオンになっているブラウザからメールメッセージを投稿しないでください。多くのユーザーはブラウザでメールを読みません。

メーリングリストの質問に回答するとき、多数のユーザーにとって興味深いと思われる回答は、質問した個人に直接返信する代わりにメーリングリストに投稿してもかまいません。その場合、元の投稿者以外のユーザーにも役立つように、包括的な回答をするようにしてください。メーリングリストに投稿する際には、回答が前の回答と重複していないことを確認してください。

回答で、質問の重要な部分を要約するように努めてください。元のメッセージ全体を引用する必要はありません。

回答がメーリングリストではなくあなた個人に送られた場合、問題を解決するのに役立った回答の恩恵をほかの人も受けられるように、回答を要約してメーリングリストに載せることは、よい工チケットとみなされます。

1.5.2 MySQL フォーラムにおける MySQL コミュニティーサポート

http://forums.mysql.com のフォーラムは重要なコミュニティーリソースです。多くのフォーラムに参加可能です。次のカテゴリがあります。

- Migration (移行)
- MySQL Usage (MySQL の使い方)
- MySQL Connector
- Programming Languages (プログラム言語)
- Tools (ツール)
- 3rd-Party Applications (サードパーティーによるアプリケーション)
- Storage Engines (ストレージエンジン)
- MySQL Technology (MySQL テクノロジ)
- SQL Standards (SQL の標準化)
- Business (ビジネス)

1.5.3 IRC (インターネットリレーチャット) の MySQL コミュニティーサポート

さまざまな MySQL メーリングリストやフォーラムに加え、IRC (インターネットリレーチャット) にも経験豊富なコミュニティーがあります。次は、当社が現在把握しているもっとも優れたネットワーク/チャネルです。

freenode (サーバーは http://www.freenode.net/ を参照してください。)

- #mysql は MySQL に関する質問が中心ですが、ほかのデータベースや SQL に関する質問も受け付けています。MySQL と PHP、Perl、C 言語の組み合わせに関する質問も行われています。
- #workbench は MySQL Workbench 関係の質問や考えが中心で、MySQL Workbench の開発者に出会うのに適した場所でもあります。

IRC ネットワークに接続するための IRC クライアントソフトウェアを探している場合は、xChat (http://www.xchat.org/) を参照してください。X-Chat (GPL ライセンス) Unix および Windows プラットフォームで使用できます (無料の Windows 版 X-Chat は http://www.silverex.org/download/ から入手できます)。

1.5.4 MySQL Enterprise

Oracle は、MySQL Enterprise のフォーラムで技術サポートを提供しています。ビジネスに不可欠な本番アプリケーションに MySQL DBMS を使用している組織の場合は、MySQL Enterprise は次を含む商用サブスクリプション製品です。

- MySQL Enterprise Server
- MySQL Enterprise Monitor
- Monthly Rapid Update および Quarterly Service Pack
- MySQL Knowledge Base
- 24 時間 365 日の技術およびコンサルタントサポート

MySQL Enterprise は複数の層で利用可能で、ニーズにもっとも適したサービスのレベルを柔軟に選択できます。詳細については、MySQL Enterprise を参照してください。

1.6 質問またはバグをレポートする方法

問題についてバグレポートを投稿する前に、それが実際にバグであることと、バグがまだレポートされていないことを確認してください。

- まず、https://dev.mysql.com/doc/ で MySQL オンラインマニュアルを検索してください。マニュアルは、常に最新の状態にしておくために、新しく見つかった問題に対する解決策によって頻繁に更新されています。また、問題に関する解決方法が新しいバージョンにすでに組み込まれている可能性が高いので、マニュアルに付随するリリースノートは特に有用です。リリースノートはマニュアル用の場所から入手可能です。
- SQL ステートメントに対するパースエラーが生じた場合は、構文を念入りにチェックしてください。そこで問題が見当たらなければ、現在使用している MySQL Server のバージョンが、使用されている構文をサポートしていない可能性が高くなります。最新のバージョンを使用しており、マニュアルが使用された構文をカバーしていない場合、MySQL Server はそのステートメントをサポートしません。

  マニュアルがその構文をカバーしているが、MySQL Server が旧バージョンの場合、MySQL の変更履歴を調べ、いつその構文が実装されたのかを確認してください。この場合、新しいバージョンの MySQL Server へアップグレードするという選択肢もあります。
- 一般的な問題の解決法については次を参照してください。セクションB.5「問題および一般的なエラー」。
- http://bugs.mysql.com/ でバグデータベースを検索して、バグがすでにレポートおよび解決されているかどうかを確認します。
- http://lists.mysql.com/ で MySQL メーリングリストのアーカイブを検索します。セクション1.5.1「MySQL メーリングリスト」 を参照してください。
- また、http://www.mysql.com/search/ を使用して、MySQL Web サイトにある Web ページすべて (マニュアルを含む) を検索することもできます。

マニュアル、バグデータベース、およびメーリングリストのアーカイブで回答を見つけることができなかった場合、お近くの MySQL の専門家にお問い合わせください。それでも質問に対する回答を見つけることができなかった場合は、次のガイドラインに従ってバグをレポートしてください。

通常バグをレポートする場合は、http://bugs.mysql.com/ にアクセスしてください。これはバグデータベースのアドレスです。このバグデータベースは一般に公開されているので、だれでも参照および検索することができます。システムにログインすると、新しいレポートを入力できます。

http://bugs.mysql.com/ のバグデータベースに投稿され、所定のリリースで修正されたバグは、リリースノートに記載されます。

MySQL Server で慎重に扱うべきセキュリティー上のバグを発見した場合は、ただちに <secalert_us@oracle.com> に電子メールメッセージを送信してお知らせください。例外: サポートのお客様は、セキュリティーのバグを含むすべての問題を http://support.oracle.com/ の Oracle Support までレポートしてください。

ほかのユーザーと問題について話し合う場合、MySQL メーリングリストのいずれかを使用することもできます。セクション1.5.1「MySQL メーリングリスト」。

よいバグレポートを書くのは時間がかかるものですが、最初に正しく行うことで報告者と弊社の双方の時間を節約できます。そのバグの完全なテストケースを含むよいバグレポートを提供していただければ、次回のリリースではその問題を修正できる可能性が非常に高くなります。このセクションでは、あまり役に立たないことをして時間を無駄にすることがないよう、レポートの正しい書き方を紹介します。このセクションを注意深く読み、ここに記載されているすべての情報がレポートに含まれているか、確認してください。

できれば、MySQL Server の最新の製品版または開発版を使用して問題をテストしてから投稿してください。テストケースに対して mysql test < script_file を使用するか、バグレポートに含まれているシェルまたは Perl のスクリプトを実行するだけで、だれでもバグを再現できるはずです。弊社で再現が可能なバグであれば、次回の MySQL リリースで修正される可能性が高くなります。

問題に関する適切な説明がバグレポートに記載されていると、もっとも効果的です。そのため、問題につながったすべての操作の適切な例を挙げ、問題自体を詳細に記述してください。もっとも効果的なレポートは、バグや問題を再現する方法を示す詳細な例が記載されたものです。セクション24.4「MySQL のデバッグおよび移植」を参照してください。

情報が多すぎるレポートに対応することはできますが、少なすぎるレポートに対応することはできません。多くの場合、問題の原因がわかっていると思い、細部を重要でないと考えるため、事実を省略してしまいます。記載するかどうかを迷ったときは、記載することをお勧めします。最初のレポートに十分な情報を記載していなかったために、再度質問して回答を待たなければならなくなるよりも、レポートに数行を追加する方が、はるかに時間が節約される上に、煩わしくありません。

バグレポートでもっともよくある誤りは、(a) 使用している MySQL ディストリビューションのバージョン番号を記載していない、(b) MySQL Server がインストールされているプラットフォームの説明 (プラットフォームの種類およびバージョン番号を含む) が十分でないというものです。これは非常に重要な情報なので、ほとんどの場合、この情報が記載されていないバグレポートは役に立ちません。「なぜうまく動作しないのか ?」 という質問が頻繁に寄せられます。その場合、要求した機能がその MySQL バージョンに実装されていなかったり、レポートに記載されているバグが新しい MySQL バージョンですでに修正されていたりすることがあります。エラーがプラットフォーム依存である場合もよくあります。そのような場合、オペレーティングシステムやプラットフォームのバージョン番号を知らないで問題を修正することはほとんど不可能です。

また、MySQL をソースからコンパイルした場合は、コンパイラが問題に関連している場合は、コンパイラに関する情報も記載してください。ユーザーがコンパイラのバグを見つけて、それが MySQL 関連の問題であると考えることがよくあります。ほとんどのコンパイラは常に開発中なので、バージョンごとに改良されています。問題がコンパイラに関連するものであるかどうかを判断するには、使用しているコンパイラを知る必要があります。コンパイルに関するすべての問題はバグとみなし、適宜レポートしてください。

プログラムでエラーメッセージが生成された場合、そのメッセージをレポートに記載することが非常に重要です。アーカイブから情報を検索しようとする場合、レポートされたエラーメッセージがプログラムで生成されたものと正確に一致している方が効果的です。(大文字小文字の違いにも注意してください。)エラーメッセージ全体をレポートにコピー&ペーストしてください。記憶に頼ってエラーメッセージを思い出そうとしないでください。

Connector/ODBC (MyODBC) に関する問題がある場合、トレースファイルを生成し、レポートともに送信してください。How to Report Connector/ODBC Problems or Bugs を参照してください。

レポートに、mysql コマンド行ツールを使用して実行したテストケースからの長いクエリー出力が含まれる場合、--vertical オプション (または \G ステートメントターミネータ) を使用すると、読みやすくなります。このセクションで後述される EXPLAIN SELECT の例では、\G の使用方法を示します。

レポートには次の情報を記載してください。

- 使用している MySQL ディストリビューションのバージョン番号 (MySQL 5.0.19 など)。実行しているバージョンを確認するには、mysqladmin version を実行します。mysqladmin プログラムは、MySQL インストールディレクトリの下の bin ディレクトリにあります。
- 問題が発生したマシンの製造元とモデル。
- オペレーティングシステムの名前とバージョン。Windows を使用している場合、名前とバージョン番号を取得するには、通常「マイコンピュータ」アイコンをダブルクリックし、「ヘルプ/バージョン情報」メニューをプルダウンします。ほとんどの Unix 系のオペレーティングシステムでは、この情報を取得するには、コマンド uname -a を実行します。
- メモリー (実メモリーと仮想メモリー) の量が関連することもあります。不確かな場合は、これらの値を記載します。

- MySQL ソフトウェアのソースディストリビューションを使用している場合、使用したコンパイラの名前とバージョン番号を記載します。バイナリディストリビューションを使用している場合は、ディストリビューション名を記載します。
- コンパイル時に問題が発生した場合、正確なエラーメッセージ、およびエラーが発生したファイル内の問題のあるコード付近の数行のコンテキストを記載します。
- mysqld が停止した場合、mysqld のクラッシュの原因となったステートメントもレポートする必要があります。通常、クエリーのロギングを有効にして mysqld を実行して、mysqld がクラッシュしたあとでログを調べることでこの情報を取得できます。セクション24.4「MySQL のデバッグおよび移植」 を参照してください。
- データベーステーブルが問題に関連している場合、SHOW CREATE TABLE db_name.tbl_name ステートメントからの出力をバグレポートに記載します。これは、データベース内のテーブルの定義を取得する非常に簡単な方法です。この情報は、発生した状況と同じ状況を再現する際に役立ちます。
- 問題発生時の SQL モードも有効な情報になる場合があるので、sql_mode システム変数の値をレポートしてください。ストアドプロシージャー、ストアドファンクション、およびトリガーオブジェクトでは、関連する sql_mode の値は、そのオブジェクトが作成された際に有効だったものです。ストアドプロシージャーまたはストアドファンクションでは、SHOW CREATE PROCEDURE または SHOW CREATE FUNCTION ステートメントは関連する SQL モードを示しています。また、INFORMATION_SCHEMA で情報を問い合わせできます。

  ```
  SELECT ROUTINE_SCHEMA, ROUTINE_NAME, SQL_MODE
  FROM INFORMATION_SCHEMA.ROUTINES;
  ```

  トリガーに対しては次のステートメントが利用できます。

  ```
  SELECT EVENT_OBJECT_SCHEMA, EVENT_OBJECT_TABLE, TRIGGER_NAME, SQL_MODE
  FROM INFORMATION_SCHEMA.TRIGGERS;
  ```

- 性能に関連するバグや SELECT ステートメントに関する問題については、EXPLAIN SELECT ... の出力、および少なくとも SELECT ステートメントが生成する行の数も含めてください。関連する各テーブルについて、SHOW CREATE TABLE tbl_name からの出力も含めてください。状況について情報を提供できればできるほど、有効な手助けが可能となります。

  下記は非常によいバグレポートの例です。mysql コマンド行ツールを使ってステートメントが実行されています。読みにくい長い出力行に対して、\G ステートメントターミネータが使用されていることに注意してください。

  ```
  mysql> SHOW VARIABLES;
  mysql> SHOW COLUMNS FROM ...\G
      <output from SHOW COLUMNS>
  mysql> EXPLAIN SELECT ...\G
      <output from EXPLAIN>
  mysql> FLUSH STATUS;
  mysql> SELECT ...;
      <A short version of the output from SELECT,
      including the time taken to run the query>
  mysql> SHOW STATUS;
      <output from SHOW STATUS>
  ```

- mysqld の実行中にバグまたは問題が発生した場合、その異常を再現する入力スクリプトを提供します。このスクリプトには、必要なソースファイルを含める必要があります。スクリプトによって再現される状況が実際に発生した状況に近いほど、効果的です。再現可能なテストケースを作成できる場合は、バグレポートに添付してアップロードしてください。

  スクリプトを提供できない場合は、少なくとも mysqladmin variables extended-status processlist からの出力をレポートに記載して、システムの動作に関する情報を提供してください。
- 数行ではテストケースを再現できない場合や、テストテーブルが大きすぎてバグレポートに含めることができない場合 (10 行を超える場合)、mysqldump を使用してテーブルをダンプし、問題を説明する README ファイルを作成してください。tar と gzip または zip を使用してファイルの圧縮アーカイブを作成します。http://bugs.mysql.com/ のバグデータベースのバグレポートを開始してから、バグレポートの「ファイル」タブをクリックして、アーカイブをバグデータベースにアップロードする指示に従ってください。
- MySQL Server でステートメントから正しくない結果が生成されたと思う場合、結果だけでなく、正しいと考える結果と、その考えの根拠を示す説明も記載します。
- 問題のサンプルを提供する際には、テーブル名や変数名などは、新しい名前を使用するよりも実際の状況で存在するものを使用するのが適切です。問題が、テーブルや変数の名前に関連している可能性があります。このようなケースはほとんどないと思われますが、後悔するよりも安全策をとるべきです。結局、実際の状況に基づいたサンプルを提供する方がユーザーにとって簡単であると同時に、当社にとっても効率的です。バグレ

ポート内に、ほかのユーザーに公開したくないデータがある場合は、前述のように「ファイル」タブを使用してアップロードできます。データが実際に最高機密で、当社にも公開したくない場合は、ほかの名前を使用したサンプルを提供することもできますが、これは最後の選択肢です。

- 可能な場合、関連するプログラムのすべてのオプションを記載します。たとえば、mysqld サーバーを開始する際に使用したオプションや MySQL クライアントプログラムの実行に使用したオプションを記載します。mysqld や mysql のようなプログラムや、configure スクリプトのオプションは多くの場合、問題解決の鍵となり、非常に重要です。これらを記載することは、決して無駄ではありません。問題が、Perl や PHP などの言語で記述されたプログラムに関係する場合は、言語プロセッサーのバージョン番号だけでなく、プログラムが使用するモジュールのバージョンも記載します。たとえば、DBI モジュールや DBD::mysql モジュールを使用する Perl スクリプトがある場合、Perl、DBI、および DBD::mysql のバージョン番号を記載します。

- 質問が権限システムに関連する場合、mysqlaccess の出力、mysqladmin reload の出力、および接続しようとした際に表示されたすべてのエラーメッセージを記載します。権限をテストする際には、まず mysqlaccess を実行します。その後、mysqladmin reload version を実行し、問題が発生したプログラムに接続します。mysqlaccess は、MySQL インストールディレクトリの下の bin ディレクトリにあります。

- バグのパッチがある場合、それを含めます。ただし、当社はパッチだけを必要としているわけではありません。パッチによって修正されるバグを示すテストケースなどの必要な情報が記載されていない場合、当社はそのパッチを使用できません。当社がパッチに関連する問題を見つける場合もあれば、パッチをまったく理解できない場合もあります。そのような場合は、パッチを使用できません。

  パッチの目的を正確に確認することができない場合、当社はそのパッチを使用しません。この場合、テストケースが役立つことになります。発生する可能性があるすべての状況がパッチによって処理されることを示してください。パッチが機能しないボーダーラインケースが見つかった場合 (それがまれなケースでも) 、そのパッチは役に立たない可能性があります。

- 何がバグであるか、そのバグがなぜ発生するのか、そのバグが何に関連するのかについての推測は、間違っていることが多いです。MySQL チームでさえも、最初にデバッガを使用しなければ、そのようなことを推測してバグの実際の原因を特定することはできません。

- ユーザー自身で問題を解決しようとしたことがほかのユーザーにわかるように、リファレンスマニュアルやメールアーカイブを確認したことをバグレポートに記載します。

- データが壊れていると思われる場合や、特定のテーブルにアクセスした際にエラーが発生した場合は、まず CHECK TABLE でテーブルをチェックしてください。そのステートメントでエラーがレポートされた場合、

  - InnoDB クラッシュリカバリメカニズムは、サーバーが強制終了したあとに再起動した場合にクリーンアップを処理します。そのため、通常の操作ではテーブルを「修復」する必要はありません。InnoDB テーブルでエラーが生じる場合は、サーバーを再起動して問題が継続するかどうか、またはエラーがメモリー内のキャッシュデータのみに影響するのかを確認します。ディスク上のデータが壊れている場合、影響を受けたテーブルをダンプできるように、innodb_force_recovery オプションを有効にして再起動してみてください。

  - トランザクショナルでないテーブルについては、REPAIR TABLE または myisamchk を使用して修復を試みてください。第5章「MySQL サーバーの管理」 を参照してください。

  Windows を実行している場合は、lower_case_table_names の値を SHOW VARIABLES LIKE 'lower_case_table_names' ステートメントを使用して確認します。この変数は、データベースおよびテーブル名の大文字/小文字をサーバーがどのように扱うかに対して影響を与えます。ある値に対しての効果は セクション9.2.2「識別子の大文字と小文字の区別」 で説明されているとおりです。

- テーブルが頻繁に壊れる場合、その問題が発生する状況と理由を特定する必要があります。この場合、MySQL データディレクトリのエラーログに、発生した問題に関する情報が格納されていることがあります。(そのファイルは、名前に .err のサフィクスが付いたファイルです。)セクション5.2.2「エラーログ」 を参照してください。このファイル内の関連する情報をバグレポートに記載します。通常、更新の途中で mysqld が強制終了されないかぎり、mysqld によってテーブルが破損することはありません。mysqld が停止した原因が特定されれば、当社はその問題の解決策をはるかに簡単に提供できます。セクションB.5.1「問題の原因を判別する方法」 を参照してください。

- 可能な場合、最新バージョンの MySQL Server をダウンロードしてインストールし、これによって問題が解決されるかどうかを確認します。MySQL ソフトウェアのすべてのバージョンが詳細にテストされているため、問題なく動作するはずです。すべてにおいて最大限の下位互換性が確保されているため、MySQL のバージョンを問題なく切り替えることができると当社は確信しています。セクション2.1.2「インストールする MySQL のバージョンと配布の選択」 を参照してください。

1.7 MySQL の標準への準拠

このセクションでは、MySQL と ANSI/ISO SQL 標準との関連について説明します。MySQL Server には SQL 標準に対する多数の拡張機能があり、ここでは、それらの拡張機能と使用方法について説明します。また、MySQL Server に不足している機能と、この不足分に対処する方法に関する情報も示します。

SQL 標準は 1986 年以降開発が繰り返され、複数のバージョンがあります。このマニュアルでは、「SQL-92」は 1992 年にリリースされた標準に、「SQL:1999」は 1999 年にリリースされた標準に、「SQL:2003」は 2003 年にリリースされた標準に、そして「SQL:2008」は 2008 年にリリースされた最新バージョンの標準に対応しています。「SQL 標準」や「標準 SQL」という語句は、常に SQL 標準の現バージョンを意味します。

製品に関する当社の主な目標の 1 つは、速度や信頼性を犠牲にすることなく、SQL 標準への準拠に向けた取り組みを続けることです。大部分のユーザーにとって MySQL Server が大幅に使いやすくなるのであれば、当社は SQL に対する拡張機能や SQL 以外の機能のサポートを積極的に追加します。HANDLER インタフェースがこの方針の一例です。セクション13.2.4「HANDLER 構文」を参照してください。

24 時間 365 日のミッションクリティカルな使用にも、負荷のかかる Web または口ギングの使用にも対応できるように、トランザクションデータベースと非トランザクションデータベースのサポートを継続しています。

MySQL Server は当初、小規模なコンピュータシステムで中規模のデータベース (1,000 万から 1 億行、または 1 テーブルあたり約 100M バイト) を操作できるように設計されました。現在の MySQL Server は、テラバイト規模のデータベースを処理しますが、ハンドヘルドデバイスや組み込みデバイスにより適した縮小バージョンでコードをコンパイルすることもできます。MySQL Server のコンパクトな設計によって、ソースツリーで競合することなく、これらのどちらの方向の開発も実現することができます。

現時点では、リアルタイムのサポートは予定されていませんが、MySQL のレプリケーション能力は重要な機能を実現します。

MySQL は ODBC レベル 0 から 3.51 をサポートします。

MySQL は、NDBCLUSTER ストレージエンジンを使用した高可用性データベースクラスタリングをサポートします。第18章「MySQL Cluster NDB 7.3 および MySQL Cluster NDB 7.4」を参照してください。

ほとんどの W3C XPath 標準をサポートする XML 機能を実装しています。セクション12.11「XML 関数」を参照してください。

SQL モードの選択

MySQL Server は異なる SQL モードで動作でき、sql_mode システム変数の値に応じて異なるクライアントにこれらの異なるモードを適用できます。DBA はサイトサーバーの動作要件に一致するグローバル SQL モードを設定でき、各アプリケーションはアプリケーションのセッション SQL モードをアプリケーション独自の要件に設定できます。

モードは MySQL がサポートする SQL 構文と、MySQL が実行するデータ検証に影響します。これにより、MySQL をさまざまな環境で使用したり、MySQL をほかのデータベースサーバーと一緒に使用したりすることが、さらに容易になります。

SQL モードの設定の詳細は、セクション5.1.7「サーバー SQL モード」を参照してください。

ANSI モードでの MySQL の実行

ANSI モードで MySQL Server を実行するには、--ansi オプションを付けて mysqld を起動します。ANSI モードでのサーバーの実行は、次のオプションを指定してサーバーを起動する場合と同じです。

```
--transaction-isolation=SERIALIZABLE --sql-mode=ANSI
```

実行時に同じ効果を得るには、次の 2 つのステートメントを実行します。

```
SET GLOBAL TRANSACTION ISOLATION LEVEL SERIALIZABLE;
SET GLOBAL sql_mode = 'ANSI';
```

次のように、sql_mode システム変数値を 'ANSI' に設定すると、ANSI モードに関連するすべての SQL モードオプションが有効になります。

```
mysql> SET GLOBAL sql_mode='ANSI';
mysql> SELECT @@GLOBAL.sql_mode;
       -> 'REAL_AS_FLOAT,PIPES_AS_CONCAT,ANSI_QUOTES,IGNORE_SPACE,ANSI'
```

--ansi を使用して ANSI モードでサーバーを実行した場合は、--ansi オプションがトランザクション分離レベルも設定するため、SQL モードを 'ANSI' に設定した場合とは少し異なります。

セクション5.1.3「サーバーコマンドオプション」を参照してください。

1.7.1 標準 SQL に対する MySQL 拡張機能

MySQL Server は、ほかの SQL DBMS にはない拡張機能をサポートします。それらを使用した場合、ほかの SQL サーバーにコードを移植できなくなるため注意してください。場合によっては、MySQL 拡張機能を含むコードを記述しても、次の形式のコメントを使用することで移植することができます。

```
/*! MySQL-specific code */
```

この場合、MySQL Server は、ほかの SQL ステートメントのようにコメント内のコードを構文解析して実行しますが、ほかの SQL サーバーはその拡張機能を認識しません。たとえば、MySQL Server は次のステートメント内の STRAIGHT_JOIN キーワードを認識しますが、ほかのサーバーは認識しません。

```
SELECT /*! STRAIGHT_JOIN */ col1 FROM table1,table2 WHERE ...
```

「!」文字のあとにバージョン番号を追加すると、コメント内の構文は、MySQL のバージョンが指定されたバージョン番号以上の場合にだけ実行されます。次のコメント内の TEMPORARY キーワードは MySQL 3.23.02 以降のサーバーでのみ実行されます。

```
CREATE /*!32302 TEMPORARY */ TABLE t (a INT);
```

次の説明は、MySQL 拡張機能をカテゴリでまとめた一覧です。

- ディスク上のデータの構成

  MySQL Server は、各データベースを MySQL データディレクトリ下のディレクトリにマップし、データベース内のテーブルをデータベースディレクトリ内のファイル名にマップします。これには、次のような意味があります。

  - ファイル名で大文字と小文字を区別するオペレーティングシステム (多くの Unix システムなど) 上の MySQL Server では、データベース名とテーブル名で大文字と小文字が区別されます。セクション9.2.2「識別子の大文字と小文字の区別」を参照してください。
  - 標準的なシステムコマンドを使用して、MyISAM ストレージエンジンで管理されるテーブルのバックアップ、名前変更、移動、削除、およびコピーを行うことができます。たとえば、MyISAM テーブルの名前変更を行うには、テーブルに対応する .MYD、.MYI、および .frm ファイルの名前を変更します。(ただし、RENAME TABLE や ALTER TABLE ... RENAME を使用して、サーバーにファイル名を変更させることをお勧めします。)

- 一般言語構文

  - デフォルトでは、「'」のほかに「"」でも文字列を囲むことができます。ANSI_QUOTES SQL モードが有効な場合、文字列を囲むことができるのは「'」だけであり、サーバーは「"」で囲まれた文字列を識別子と解釈します。
  - 「\」は文字列内のエスケープ文字です。
  - SQL ステートメントでは、db_name.tbl_name 構文を使用して、さまざまなデータベースのテーブルにアクセスできます。同様の機能を備えた SQL サーバーもありますが、これは User space と呼ばれます。MySQL Server は、CREATE TABLE ralph.my_table ... IN my_tablespace など、ステートメントで使用されるようなテーブルスペースをサポートしません。

- SQL ステートメント構文

  - ANALYZE TABLE、CHECK TABLE、OPTIMIZE TABLE、および REPAIR TABLE ステートメント。
  - CREATE DATABASE、DROP DATABASE、および ALTER DATABASE ステートメント。セクション13.1.10「CREATE DATABASE 構文」、セクション13.1.21「DROP DATABASE 構文」、およびセクション13.1.1「ALTER DATABASE 構文」を参照してください。
  - DO ステートメント。
  - クエリーオプティマイザによるテーブルの処理方法に関する説明を取得する EXPLAIN SELECT。
  - FLUSH および RESET ステートメント。
  - SET ステートメント。セクション13.7.4「SET 構文」を参照してください。

  - SHOW ステートメント。セクション13.7.5「SHOW 構文」を参照してください。MySQL 固有の SHOW ステートメントの多くによって生成される情報は、SELECT を使用して INFORMATION_SCHEMA をクエリーすることによって、より標準的な方法で取得できます。第21章「INFORMATION_SCHEMA テーブル」を参照してください。
  - LOAD DATA INFILE の使用。多くの場合、この構文は Oracle の LOAD DATA INFILE と互換性があります。セクション13.2.6「LOAD DATA INFILE 構文」を参照してください。
  - RENAME TABLE の使用。セクション13.1.32「RENAME TABLE 構文」を参照してください。
  - DELETE + INSERT の代わりとしての REPLACE の使用。セクション13.2.8「REPLACE 構文」を参照してください。
  - ALTER TABLE ステートメントにおける CHANGE col_name、DROP col_name、あるいは DROP INDEX、IGNORE または RENAME の使用。ALTER TABLE ステートメントにおける、複数の ADD、ALTER、DROP、または CHANGE 句の使用。セクション13.1.7「ALTER TABLE 構文」を参照してください。
  - インデックス名の使用、カラムのプリフィクス上のインデックス、および CREATE TABLE ステートメントでの INDEX または KEY の使用。セクション13.1.17「CREATE TABLE 構文」を参照してください。
  - CREATE TABLE を用いた TEMPORARY または IF NOT EXISTS の使用。
  - DROP TABLE および DROP DATABASE を用いた IF EXISTS の使用。
  - 1 つの DROP TABLE ステートメントで複数のテーブルを破棄できる機能。
  - UPDATE および DELETE ステートメントの ORDER BY および LIMIT 句。
  - INSERT INTO tbl_name SET col_name = ... 構文。
  - INSERT および REPLACE ステートメントの DELAYED 句。
  - INSERT、REPLACE、DELETE、および UPDATE ステートメントの LOW_PRIORITY 句。
  - SELECT ステートメントにおける INTO OUTFILE または INTO DUMPFILE の使用。セクション13.2.9「SELECT 構文」を参照してください。
  - SELECT ステートメントにおける STRAIGHT_JOIN や SQL_SMALL_RESULT などのオプション。
  - GROUP BY 句で、選択したすべてのカラムの名前を列挙する必要はありません。これにより、ごく一部ではありますが、きわめて一般的なクエリーのパフォーマンスが向上します。セクション12.19「GROUP BY 句で使用される関数と修飾子」を参照してください。
  - ORDER BY を用いるだけでなく GROUP BY を用いても、ASC および DESC を指定できます。
  - := 割り当て演算子で、ステートメント内の変数を設定する機能。セクション9.4「ユーザー定義変数」を参照してください。
- データ型
  - MEDIUMINT、SET、および ENUM データ型と、さまざまな BLOB および TEXT データ型。
  - AUTO_INCREMENT、BINARY、NULL、UNSIGNED、および ZEROFILL データ型の属性。
- 関数と演算子
  - ほかの SQL 環境を使用していたユーザーにわかりやすいように、MySQL Server では多数の関数のエイリアスがサポートされています。たとえば、すべての文字列関数で、標準の SQL 構文と ODBC 構文の両方がサポートされています。
  - MySQL Server は、|| および && 演算子が、C プログラミング言語の場合と同様に論理 OR および AND を意味することを理解しています。MySQL Server では、|| と OR はシノニムであり、&& と AND も同様です。この優れた構文のために、MySQL Server では、文字列の連結に標準 SQL の || 演算子を使用することができません。その代わりに、CONCAT() を使用します。CONCAT() は任意の数の引数を取るため、|| 演算子の使用を MySQL Server に変換することは簡単です。
  - value_list に複数の要素がある場合の、COUNT(DISTINCT value_list) の使用。

- 文字列比較はデフォルトで、大文字と小文字が区別され、ソート順序は現在の文字セットの照合順序によって決定されます。デフォルトの文字セットは latin1 (cp1252 西ヨーロッパ) です。これ以外を希望する場合は、BINARY 属性で自身のカラムを宣言するか、BINARY キャストを使用する必要があります。これにより、辞書順ではなくベースとなる文字コード値を使用して比較が行われます。
- % 演算子は MOD() のシノニムです。つまり、N % M は MOD(N,M) と同等です。% は、C プログラマと、PostgreSQL との互換性のためにサポートされています。
- SELECT ステートメントの出力カラムリスト (FROM の左側) の式で、=、<>、<=、<、>=、>、<<、>>、<=>、AND、OR、または LIKE 演算子を使用できます。例:

```
mysql> SELECT col1=1 AND col2=2 FROM my_table;
```

- LAST_INSERT_ID() 関数は、最新の AUTO_INCREMENT 値を返します。セクション12.14「情報関数」を参照してください。
- LIKE は、数値に対して使用できます。
- REGEXP および NOT REGEXP 拡張正規表現演算子。
- 1 つまたは複数の引数を使用する CONCAT() または CHAR()。(MySQL Server では、これらの関数は引数をいくつでも使用することができます。)
- BIT_COUNT()、CASE、ELT()、FROM_DAYS()、FORMAT()、IF()、PASSWORD()、ENCRYPT()、MD5()、ENCODE()、D および WEEKDAY() 関数。
- 部分文字列を削除する TRIM() の使用。標準 SQL では、1 つの文字しか削除できません。
- GROUP BY 関数 STD()、BIT_OR()、BIT_AND()、BIT_XOR()、および GROUP_CONCAT()。セクション12.19「GROUP BY 句で使用される関数と修飾子」を参照してください。

1.7.2 MySQL と標準 SQL との違い

MySQL Server を ANSI SQL 標準および ODBC SQL 標準に準拠したものにすることを目標としていますが、次のように、MySQL Server が異なる動作を示すことがあります。

- MySQL と標準 SQL の権限システムには、いくつかの違いがあります。たとえば MySQL では、テーブルを削除しても、テーブルの権限が自動的に取り消されません。テーブルの権限を取り消すには、明示的に REVOKE ステートメントを発行する必要があります。詳細は、セクション13.7.1.6「REVOKE 構文」を参照してください。
- CAST() 関数は、REAL または BIGINT に対するキャストをサポートしていません。セクション12.10「キャスト関数と演算子」を参照してください。

1.7.2.1 SELECT INTO TABLE の違い

MySQL Server では、Sybase SQL 拡張機能 SELECT ... INTO TABLE はまだサポートされていません。MySQL Server では代わりに、基本的に同じ処理を行う INSERT INTO ... SELECT 標準 SQL 構文がサポートされています。セクション13.2.5.1「INSERT ... SELECT 構文」を参照してください。例:

```
INSERT INTO tbl_temp2 (fld_id)
   SELECT tbl_temp1.fld_order_id
   FROM tbl_temp1 WHERE tbl_temp1.fld_order_id > 100;
```

あるいは、SELECT ... INTO OUTFILE または CREATE TABLE ... SELECT を使用することもできます。

ユーザー定義変数で SELECT ... INTO を使用できます。カーソルとローカル変数を用いてストアドルーチン内でも同じ構文を使用できます。セクション13.2.9.1「SELECT ... INTO 構文」を参照してください。

1.7.2.2 UPDATE の違い

式で更新されるテーブルのカラムにアクセスする場合、UPDATE はそのカラムの現在の値を使用します。次のステートメントの 2 番目の割り当ては、col2 を元の col1 値ではなく、現在の (更新された) col1 値に設定します。この結果、col1 と col2 の値が同じになります。この動作は標準 SQL とは異なります。

```
UPDATE t1 SET col1 = col1 + 1, col2 = col1;
```

1.7.2.3 トランザクションおよびアトミック操作の違い

MySQL Server (バージョン 3.23-max およびすべてのバージョン 4.0 以降) では、InnoDB トランザクションストレージエンジンでトランザクションがサポートされています。MySQL 5.5 以上では、セクション14.1.1「デフォルトの MySQL ストレージエンジンとしての InnoDB」で説明しているように、新しく作成したテーブルではデフォルトで InnoDB が使用されます。デフォルトで、InnoDB は完全な ACID コンプライアンスを提供します。ACID コンプライアンスと raw パフォーマンスとのバランスを取るように設定を調整する方法については、セクション14.2.1「MySQL および ACID モデル」を参照してください。トランザクションエラーの取り扱いにおける、標準 SQL と InnoDB との違いについては、セクション14.19.4「InnoDB のエラー処理」を参照してください。

MySQL Server での非トランザクションストレージエンジン (MyISAM など) は、「アトミック操作」と呼ばれるデータ整合性に関する別のパラダイムに従います。MyISAM テーブルは実質上常に、autocommit = 1 モードで操作します。変更されたデータは、一度に 1 つのステートメントでディスクに書き込まれるため、一連の関連する DML 操作は途中で妨害される可能性があり、この操作の一貫性を保証することが非常に困難になります。したがって、このモードは読み取りが大半のワークロードに適しています。トランザクションの条件では、それぞれの特定の更新が実行している間、ほかのユーザーがこれを妨げることができず、自動的なロールバックが存在できず、ダーティー読み取りが存在しません。ただし、これらの特性は単一の操作に適用され、一つの単位として成功または失敗する関連した更新には適用されません。LOCK TABLES ステートメントなどの回避策は、非トランザクションテーブルへの並列書き込みアクセスを制限します。

同じアプリケーション内の別々のテーブルの場合でも、どちらのパラダイムを使用するかを選択できます。信頼性と高いパフォーマンスを両立させる場合にはトランザクション機能を、(たとえばレプリケーションスレーブサーバーでの) 重要でない読み取りが大半のデータにはアトミック操作を選択できます。

InnoDB などのトランザクションストレージエンジンでは、負荷のかかる読み取り/書き込みワークロードに対する高い信頼性をサポートする多くの重要な機能が用意されています。結果として、トランザクションテーブルのメモリーおよびディスク容量の要件は高くなり、CPU オーバーヘッドが大きくなります。MySQL Server のモジュラー設計によって、さまざまなストレージエンジンの並列使用が可能になり、さまざまな要件に適合し、あらゆる状況で最適なパフォーマンスを実現できます。

非トランザクションテーブルでの信頼性に対する回避策

しかし、非トランザクションの MyISAM テーブルとの完全性も維持するには、MySQL Server の機能をどのように使用すればよいでしょうか。また、このような機能はトランザクションストレージエンジンにどのように匹敵するでしょうか。

- 重大な状況で COMMIT ではなく ROLLBACK の呼び出しに依存するようにアプリケーションが作成されている場合、トランザクションの方が便利です。トランザクションでは、完了していない更新や失敗したアクティビティーがデータベースにコミットされていないことも確認します。サーバーには、自動ロールバックを行う機会が与えられ、データベースは保存されます。

  非トランザクションテーブルを使用する場合は、更新の前にチェックを含めることや、不整合についてデータベースをチェックし、このような不整合が発生した場合には自動的に修復または警告するスクリプトを実行することによって、アプリケーションレベルで潜在的な問題を解決する必要があります。MySQL ログを使用するか、さらに 1 つのログを追加することになっても、データの完全性を損なわずに、通常どおりテーブルを修正できます。

- 場合によっては、重要なトランザクション更新は、アトミックになるように書き換えることができます。LOCK TABLES またはアトミック更新で複数の DML 操作を行うことができ、並列書き込みアクセスを制限することによってデッドロックが起こらないようにします。テーブルの末尾での並列挿入を可能にする、テーブルに対する READ LOCAL ロック (書き込みロックの反対) を取得した場合、読み取りは許可され、ほかのクライアントによる挿入も許可されます。新しく挿入したレコードは、読み取りがロックされているクライアントには、読み取りロックが解除されるまで表示されません。INSERT DELAYED を使用すると、ロックが解除されるまで挿入をローカルキューに入れることができ、クライアントは挿入が完了するまで待機する必要はありません。セクション8.10.3「同時挿入」およびセクション13.2.5.2「INSERT DELAYED 構文」を参照してください。

- MySQL Server で安全性を確保するには、使用するテーブルの種類に関係なく、定期的にバックアップを作成してバイナリロギングをオンにします。使用するデータベースシステムに関係なく、どのような場合でもバックアップを作成することは賢明です。

次に、非トランザクションテーブルを操作するための手法を示します。

- LOCK TABLES を使用して、通常はトランザクションを必要とするループをコード化することができ、実行中にレコードを更新するカーソルが不要です。

- ROLLBACK を使用しないようにするため、次の方法を使用することができます。

1. LOCK TABLES を使用して、アクセスするすべてのテーブルをロックします。
2. 更新を実行する前に、true になっている必要がある条件をテストします。
3. 条件が満たされていれば、更新します。
4. UNLOCK TABLES を使用して、ロックを解除します。

> **注記**
>
> この解決方法は、だれかが更新の途中でスレッドを強制終了したときの状況に対処しません。その場合、すべてのロックが解除されますが、一部の更新が実行されていない可能性があります。

- 次の手法を使用して、関数を用いて単一の操作でレコードを更新することもできます。
  - 現在の値に関連してカラムを変更します。これにより、別のクライアントがその間にカラム値を変更した場合でも、更新は正しくなります。
  - 実際に変更されたカラムのみを更新します。これが一般的にデータベースにとって適切な方法になります。
- 一意の識別子を管理するときに、AUTO_INCREMENT カラムと、LAST_INSERT_ID() SQL 関数または mysql_insert_id() C API 関数のどちらかを使用することによって、LOCK TABLES や ROLLBACK などのステートメントを回避できます。セクション12.14「情報関数」およびセクション23.7.7.37「mysql_insert_id()」を参照してください。

  行レベルロックが必要な状況では、InnoDB テーブルを使用します。それ以外の場合、MyISAM テーブルでは、テーブルでフラグカラムを使用して、次のような操作を実行することができます。

```
UPDATE tbl_name SET row_flag=1 WHERE id=ID;
```

  レコードが見つかり、元の行で row_flag がすでに 1 でなくなっていた場合、MySQL は、影響を受けた行の数として 1 を返します。MySQL Server が前述のステートメントを次のように変更したと考えることができます。

```
UPDATE tbl_name SET row_flag=1 WHERE id=ID AND row_flag <> 1;
```

1.7.2.4 外部キーの違い

InnoDB ストレージエンジンは、CASCADE、ON DELETE、ON UPDATE などの外部キー制約のチェックをサポートします。セクション14.6.6「InnoDB と FOREIGN KEY 制約」を参照してください。

InnoDB および NDB 以外のストレージエンジンの場合、MySQL Server は CREATE TABLE ステートメントの FOREIGN KEY 構文を解析しますが、これを使用したり格納したりしません。この情報は、mysqldump にも存在し、Connector/ODBC を使用して取得できます。INFORMATION_SCHEMA 情報データベース内の INFORMATION_SCHEMA.TABLE_CONSTRAINTS テーブルをチェックすることによって、どのテーブルに外部キー制約があるかを確認できます。INFORMATION_SCHEMA.REFERENTIAL_CONSTRAINTS テーブルから、外部キーに関する詳細を取得できます。さらに、InnoDB には、InnoDB テーブル上の外部キーに関する情報を含む多数の INFORMATION_SCHEMA テーブルが用意されています。セクション21.29「InnoDB の INFORMATION_SCHEMA テーブル」を参照してください。

データベース開発者にとって、次のような外部キーを使用するメリットがあります。

- 関係が適切に設計されている場合、外部キー制約によって、プログラマがデータベースで不整合を引き起こすことが少なくなります。
- データベースサーバーによって集中的に制約チェックが行われるため、アプリケーション側でのこれらのチェックが不要になります。このことで、さまざまなアプリケーションで制約が同じようにチェックされない場合があるという可能性がなくなります。
- 連鎖更新および削除を使用すると、アプリケーションコードを単純化することができます。
- 適切に設計された外部キールールは、テーブル間の関係の記述に役立ちます。

SQL の外部キーは、テーブルの結合ではなく、参照整合性のチェックと実施に使用されます。SELECT ステートメントで複数のテーブルからの結果を取得する場合は、次のようにそれらの結合を実行してください。

```
SELECT * FROM t1 INNER JOIN t2 ON t1.id = t2.id;
```

セクション13.2.9.2「JOIN 構文」およびセクション3.6.6「外部キーの使用」を参照してください。

ON DELETE ... を使用しない FOREIGN KEY 構文は、自動的に WHERE 句を作成するために、ODBC アプリケーションで使用されることがよくあります。

SQL 標準からの逸脱

MySQL での外部キーの実装は、次の重要な点で SQL 標準と異なります。

- 同じ参照キー値を持つ複数の行が親テーブルにある場合、InnoDB は、同じキー値を持つほかの親の行が存在しないかのように、外部キーチェックで動作します。たとえば、RESTRICT 型の制約が定義されていて、複数の親の行を含む子の行が存在する場合は、これらの親の行のいずれかを削除することが InnoDB で許可されません。

  InnoDB では、外部キー制約に対応するインデックス内のレコードに基づいて、深さ優先アルゴリズムを使用したカスケード操作が実行されます。

- UNIQUE でないキーを参照する FOREIGN KEY 制約は、標準 SQL ではなく InnoDB の拡張機能です。

- ON UPDATE CASCADE または ON UPDATE SET NULL が同じカスケード中に以前に更新していた同じテーブルを更新するように再帰する場合、RESTRICT のように機能します。つまり、自己参照型 ON UPDATE CASCADE または ON UPDATE SET NULL 操作は使用できません。この目的は、カスケード更新で発生する無限ループを回避することです。反対に、自己参照型 ON DELETE SET NULL は、自己参照型 ON DELETE CASCADE と同様に動作できます。カスケード操作は、15 レベルよりも深くネストされる可能性がありません。

- 多くの行を挿入、削除、または更新する SQL ステートメントでは、外部キー制約 (一意の制約など) が行ごとにチェックされます。外部キーチェックを行なっているとき、InnoDB は、調べる必要のある子レコードまたは親レコード上に共有行レベルロックを設定します。MySQL は外部キー制約の確認を即座に行います。その確認がトランザクションコミットまで遅延されることはありません。SQL 標準によると、デフォルトの動作は遅延チェックにするべきです。つまり、SQL ステートメント全体が処理されたあとにはじめて、制約がチェックされます。これは、外部キーを使用して自己参照する行を削除できないことを意味します。

InnoDB 外部キーが SQL 標準とどのように異なるかについては、セクション14.6.6「InnoDB と FOREIGN KEY 制約」を参照してください。

1.7.2.5 コメントの先頭としての「--」

標準 SQL では、コメントに C 構文 /* this is a comment */ が使用され、MySQL Server でも同様にこの構文がサポートされています。セクション9.6「コメントの構文」で説明するように、MySQL は、MySQL 固有の SQL をコメントに埋め込めるようにする、この構文の拡張機能もサポートします。

標準 SQL では、「--」 をコメント開始連続文字として使用します。MySQL Server では、「#」 をコメント開始文字として使用します。MySQL Server 3.23.3 以上では、「--」 コメントスタイルのバリアントもサポートします。つまり、「--」 コメント開始連続文字には空白 (または改行などの制御文字) を続ける必要があります。この空白は、payment の値を自動的に挿入する次のような構造構文を使用した自動生成の SQL クエリーでの問題を防止するために必要です。

```
UPDATE account SET credit=credit-payment
```

payment の値が負数 (-1など) の場合、どうなるかを考えてみてください。

```
UPDATE account SET credit=credit--1
```

credit--1 は SQL では有効な式ですが、「--」 はコメントの先頭として解釈され、式の一部は破棄されます。結果として、意図したものとはまったく異なる意味を持つステートメントとなってしまいます。

```
UPDATE account SET credit=credit
```

このステートメントでは、値が変更されることは一切ありません。これは、コメントを 「--」 で開始できるようにすると、深刻な結果を招く可能性があること示します。

実装を使用するには、「--」 が MySQL Server 3.23.3 以降でのコメント開始連続文字として認識されるように、これに空白を続けることが必要になります。したがって、credit--1 は安全に使用できます。

別の安全な機能は、mysql コマンド行クライアントで 「--」 で始まる行を無視するというものです。

次の情報は、3.23.3 より前のバージョンの MySQL を実行している場合にのみ関連します。

テキストファイルに「--」コメントを含む SQL スクリプトがある場合は、スクリプトを実行する前に、次のように replace ユーティリティーを使用して「#」文字を使用するようにコメントを変換してください。

```
shell> replace " --" " #" < text-file-with-funny-comments.sql \
       | mysql db_name
```

これは通常の方法でスクリプトを実行するよりも安全です。

```
shell> mysql db_name < text-file-with-funny-comments.sql
```

「所定の場所」のスクリプトファイルを編集して、「--」コメントを「#」コメントに変更することもできます。

```
shell> replace " --" " #" -- text-file-with-funny-comments.sql
```

次のコマンドで元に戻してください。

```
shell> replace " #" " --" -- text-file-with-funny-comments.sql
```

セクション4.8.2「replace — 文字列置換ユーティリティー」を参照してください。

1.7.3 MySQL における制約の処理

MySQL では、ロールバックを許可するトランザクションテーブルと、許可しない非トランザクションテーブルの両方を操作できます。このため、MySQL とほかの DBMS とでは制約の処理が多少異なります。エラーが起きたときに変更をロールバックできない非トランザクションテーブルに、多数の行を挿入または更新した場合を扱う必要があります。

基本的な考え方は、MySQL Server が、実行するステートメントを構文解析している間に検出できるすべてのものに対してエラーを生成しようとし、ステートメントを実行する間に発生するエラーからリカバリしようとするというものです。ほとんどの場合でこれを行いますが、すべての場合にはまだです。

エラーが発生したときに MySQL で可能な選択肢は、途中でステートメントを中止するか、問題からリカバリするためにできるかぎりのことを行なって処理を続行するかです。デフォルトでは、サーバーは後者の方法に従います。これは、たとえば、サーバーは無効な値をもっとも近い有効な値に強制的に修正するということです。

不良データ値の処理や、エラー時にステートメント実行を継続するか中止するかをより適切に制御できるようにする複数の SQL モードオプションを利用できます。これらのオプションを使用して、不適切な入力を拒絶するほかの DBMS と同じように、より従来に近い方法で機能するように MySQL Server を構成できます。サーバー起動時に、SQL モードをグローバルに設定して、すべてのクライアントに影響を与えることができます。実行時に個々のクライアントで SQL モードを設定できるため、各クライアントは要件にもっとも適した動作を選択できます。セクション5.1.7「サーバー SQL モード」を参照してください。

次のセクションでは、さまざまな種類の制約を MySQL Server で処理する方法について説明します。

1.7.3.1 PRIMARY KEY および UNIQUE インデックス制約

通常、データ変更ステートメント (INSERT や UPDATE など) がプライマリキー、一意キー、または外部キーの制約に違反すると、エラーが発生します。InnoDB などのトランザクションストレージエンジンを使用している場合、MySQL ではステートメントが自動的にロールバックされます。非トランザクションストレージエンジンを使用している場合、MySQL は、エラーが発生した行でステートメントの処理を停止し、残りの行を未処理のままにしておきます。

MySQL では、INSERT や UPDATE などに対して IGNORE キーワードをサポートします。これを使用する場合は、MySQL は、プライマリキーまたは一意キーの違反を無視し、次の行を処理し続けます。使用しているステートメントに関するセクション (セクション13.2.5「INSERT 構文」やセクション13.2.11「UPDATE 構文」など) を参照してください。

mysql_info() C API 関数で、実際に挿入または更新される行数についての情報を取得できます。SHOW WARNINGS ステートメントを使用することもできます。セクション23.7.7.35「mysql_info()」およびセクション13.7.5.41「SHOW WARNINGS 構文」を参照してください。

現時点では、外部キーがサポートされているのは InnoDB テーブルのみです。セクション14.6.6「InnoDB と FOREIGN KEY 制約」を参照してください。

1.7.3.2 FOREIGN KEY の制約

外部キーを使用すると、複数のテーブルにわたる関連データをクロス参照することができ、外部キー制約は、この分散したデータの整合性の維持に役立ちます。

MySQL は、CREATE TABLE および ALTER TABLE ステートメントにおける ON UPDATE および ON DELETE 外部キー参照をサポートします。使用可能な参照アクションは、RESTRICT (デフォルト)、CASCADE、SET NULL、および NO ACTION です。

> 注記
>
> NDB は、参照カラムが親テーブルの主キーである ON UPDATE CASCADE アクションをサポートしません。

SET DEFAULT も MySQL Server でサポートされますが、現在、InnoDB および NDB によって無効として拒否されます。MySQL は遅延した制約のチェックをサポートしないので、NO ACTION は RESTRICT として扱われます。外部キーについて MySQL によってサポートされる正確な構文については、セクション13.1.17.2「外部キー制約の使用」を参照してください。

MATCH FULL、MATCH PARTIAL、および MATCH SIMPLE は許可されますが、同じステートメントで使用されるすべて ON DELETE 句と ON UPDATE 句を MySQL Server に無視させるため、これらを使用しないでください。MySQL では MATCH オプションはほかの効果がなく、事実上常に MATCH SIMPLE セマンティクスが強制されます。

MySQL では、外部キーカラムにインデックスを付ける必要があります。外部キー制約はあるが所定のカラムのインデックスがないテーブルを作成する場合、インデックスが作成されます。例外: MySQL Cluster には、外部キーカラムで明示的な一意キー (または主キー) が必要です。

INFORMATION_SCHEMA.KEY_COLUMN_USAGE テーブルから、外部キーに関する情報を取得できます。このテーブルに対するクエリーの例を次に示します。

```
mysql> SELECT TABLE_SCHEMA, TABLE_NAME, COLUMN_NAME, CONSTRAINT_NAME
    > FROM INFORMATION_SCHEMA.KEY_COLUMN_USAGE
    > WHERE REFERENCED_TABLE_SCHEMA IS NOT NULL;
+-------------+---------------+-------------+-----------------+
| TABLE_SCHEMA | TABLE_NAME    | COLUMN_NAME | CONSTRAINT_NAME |
+-------------+---------------+-------------+-----------------+
| fk1         | myuser        | myuser_id   | f               |
| fk1         | product_order | customer_id | f2              |
| fk1         | product_order | product_id  | f1              |
+-------------+---------------+-------------+-----------------+
3 rows in set (0.01 sec)
```

InnoDB テーブル上での外部キーの情報は、INFORMATION_SCHEMA データベースにおける INNODB_SYS_FOREIGN および INNODB_SYS_FOREIGN_COLS テーブルにもあります。

1.7.3.3 無効データの制約

デフォルトでは、MySQL は無効または不適切なデータ値を許容しており、これらをデータエントリに対する有効な値に強制的に変更します。ただし、厳密 SQL モードを有効にして、サーバーが不良値を拒否し、不良値が発生するステートメントを中止するという従来の方法に近い不良値の取り扱いを選択できます。セクション5.1.7「サーバー SQL モード」を参照してください。

このセクションでは、MySQL のデフォルトの (許可されている) 動作のほか、厳密 SQL モードとこれがどのように異なるかについて説明します。

厳密モードを使用していない場合、NOT NULL カラムに NULL を挿入したり、数値カラムに大きすぎる数値を挿入したりするなど、「正しくない」値をカラムに挿入すると、MySQL は、エラーを生成するのではなく、カラムを「最適可能値」に設定します。この動作ルールについて、次に詳しく説明します。

- 数値カラムに範囲外の値を格納しようとすると、MySQL Server は代わりに、0、指定可能な最小値、または指定可能な最大値 (いずれも無効値に近い値) を格納します。
- 文字列については、MySQL は空白文字列、またはカラム内に格納可能な最長文字列を格納します。
- 数字で始まらない文字列を数値カラムに格納しようとすると、MySQL Server は 0 を格納します。
- セクション1.7.3.4「ENUM および SET の制約」で説明するように、ENUM カラムと SET カラムの無効な値は処理されます。
- MySQL では、特定の不正なデータ値を、DATE カラムと DATETIME カラムに格納できます ('2000-02-31' や '2000-02-00' など)。この場合、アプリケーションが厳密 SQL モードを有効にしていなければ、これらを格納する前に日付を検証するのはアプリケーションに任されます。MySQL が日付値を格納し、ちょうど同じ値を検索

できる場合、MySQL は与えられたとおりに格納します。日付が完全に不正な場合 (サーバーの格納能力を超えている場合) は、代わりに特殊な「ゼロ」日付値である '0000-00-00' がカラムに格納されます。

- NULL 値を使用することができないカラムに NULL を格納しようとすると、単一行の INSERT ステートメントに対するエラーが生じます。複数行の INSERT ステートメントや INSERT INTO ... SELECT ステートメントに対して、MySQL Server は、カラムデータ型に暗黙のデフォルト値を格納します。一般に、これは数値型には 0、文字列型には空の文字列 (''')、および日時型には「ゼロ」値になります。暗黙のデフォルト値については、セクション11.6「データ型デフォルト値」で説明されています。

- INSERT ステートメントがカラムに対して値を指定しない場合、MySQL は、カラム定義に明示的な DEFAULT 句が含まれていると、そのデフォルト値を挿入します。定義に DEFAULT のような句が含まれていなければ、MySQL はカラムデータ型に暗黙のデフォルト値を挿入します。

非厳密モードで前述のルールを使用する理由は、ステートメントが実行を開始するまで、これらの条件をチェックできないからです。ストレージエンジンでロールバックがサポートされていない可能性があるため、複数の行を更新したあとで問題が発生した場合は、単にロールバックすればよいというわけにはいきません。ステートメントを終了するというオプションは、ここまでよくはありません。この場合、更新が「未完了」のため、最悪のシナリオになる可能性があります。この場合、「できるかぎり最善を尽くし」、何も起こらなかったように続けることがお勧めです。

MySQL 5.0.2 以降では、STRICT_TRANS_TABLES または STRICT_ALL_TABLES SQL モードを使用して、より厳密な入力値処理を選択できます。

```
SET sql_mode = 'STRICT_TRANS_TABLES';
SET sql_mode = 'STRICT_ALL_TABLES';
```

STRICT_TRANS_TABLES は、トランザクションストレージエンジンに対して厳密モードを有効にし、非トランザクションエンジンに対してもある程度有効にします。これは、次のように動作します。

- トランザクションストレージエンジンの場合、ステートメント内で生じる不良データ値は、ステートメントの実行中止やロールバックを引き起こします。

- 非トランザクションストレージエンジンの場合、ステートメントは、挿入または更新される最初の行でエラーが発生した場合に中止します。(トランザクションテーブルの場合と同じように、最初の行でエラーが生じた場合、ステートメントを中止して、テーブルを変更しないままにしておくことができます。)2 行以降でエラーが生じた場合、最初の行によってテーブルがすでに変更されているため、ステートメントは中止しません。代わりに、不良データ値が調整され、この結果、エラーではなく警告が発せられます。言い換えると、STRICT_TRANS_TABLES を使用すると、テーブルを変更せずに行える場合は、間違った値によって、MySQL はこれまでに行われたすべての更新をロールバックします。ただし、いったんテーブルが変更されると、それ以降に生じるエラーは調整や警告になります。

より厳密なチェックの場合にも、STRICT_ALL_TABLES を有効にします。これは、STRICT_TRANS_TABLES と同じですが、非トランザクションストレージエンジンの場合に、2 行目以降の行の不良データであっても、エラーによりステートメントが中止する点が異なります。これは、非トランザクションテーブルで複数行の挿入または更新の途中でエラーが発生した場合、部分更新が行われることを意味します。これより前の行は挿入または更新されますが、エラーの時点以降の行は挿入も更新もされません。非トランザクションテーブルでこれを回避するには、単一行ステートメントを使用するか、エラーではなく変換警告が許容できる場合は STRICT_TRANS_TABLES を使用してください。初めから問題を避けるには、MySQL でカラム内容をチェックさせないようにしてください。データベースに有効な値だけを渡していることをアプリケーションに確認させることがもっとも安全です (多くの場合で高速になります)。

厳密モードオプションのいずれかで、IGNORE のない INSERT または UPDATE ではなく、INSERT IGNORE または UPDATE IGNORE を使用することによって、エラーを警告として処理することができます。

1.7.3.4 ENUM および SET の制約

ENUM および SET カラムによって、特定の値セットのみを含むカラムを効率的に定義することができます。セクション11.4.4「ENUM 型」およびセクション11.4.5「SET 型」を参照してください。ただし、MySQL 5.0.2 以前では、ENUM および SET カラムは無効な値のエントリに対する実際の制約はありません。

- ENUM カラムには常にデフォルト値があります。デフォルトでない値を指定した場合、NULL を指定できるカラムではこの値は NULL になり、それ以外のカラムではカラム定義の最初の列挙値になります。

- ENUM カラムに不正な値を挿入した場合、または IGNORE を付けた ENUM カラムに強制的にある値を指定した場合は、予約された列挙値である 0に設定され、文字列コンテキストでは空の文字列として表示されます。

- SET カラムに不正な値を挿入した場合、その値は無視されます。たとえば、カラムが 'a'、'b'、および 'c' の値を含む場合、'a,x,b,y' を割り当てようとすると 'a,b' の値になってしまいます。

MySQL 5.0.2 以降では、厳密 SQL モードを使用するようにサーバーを構成できます。セクション5.1.7「サーバー SQL モード」を参照してください。厳密モードを有効にすると、ENUM や SET カラムの定義は、カラムに入力した値に対して制約として機能します。これらの条件を満たさない値には、エラーが発生します。

- ENUM 値は、カラム定義に一覧表示されている値のいずれかか、それと内部数値的に同等のものである必要があります。その値はエラー値 (つまり、0 または空の文字列) にはなりえません。ENUM('a','b','c') として定義されたカラムでは、''、'd'、'ax' などの値は無効であり拒否されます。

- SET 値は空の文字列にするか、カンマで区切られたカラム定義に一覧表示されている値だけから構成された値にする必要があります。SET('a','b','c') として定義されたカラムの場合、'd' や 'a,b,c,d' などの値は無効であり拒否されます。

無効な値に対するエラーは、INSERT IGNORE または UPDATE IGNORE を使用している場合、厳密モードで抑制できます。この場合、エラーではなく警告が発せられます。ENUM に対しては、その値はエラーメンバー (0) として挿入されます。SET に対しては、無効な部分文字列が削除されるという点を除いて、その値は与えられたとおりに挿入されます。たとえば、'a,x,b,y' は 'a,b' の値となります。

1.8 クレジット

次のセクションでは、MySQL を今日の形にするのを支援した開発者、貢献者、および支援者をリストします。

1.8.1 MySQL への貢献者

MySQL Server と MySQL マニュアルの全著作権はオラクル社およびその関連会社が保有しますが、MySQL ディストリビューションに何らかの形で貢献していただいた方々に感謝申し上げます。貢献者は、次のとおりです。順不同です。

- Gianmassimo Vigazzola <qwerg@mbox.vol.it> または <qwerg@tin.it>

  Win32/NT への最初の移植。

- Per Eric Olsson

  動的レコード形式に対する建設的な批評と実際のテスト。

- Irena Pancirov <irena@mail.yacc.it>

  Borland コンパイラを使用した Win32 の移植。mysqlshutdown.exe および mysqlwatch.exe。

- David J.Hughes

  シェアウェア SQL データベースの作成に尽力。MySQL AB の前身である TcX で、mSQL を始めましたが、それでは目的を達成できないことがわかったため、アプリケーションビルダー Unireg に対する SQL インタフェースの作成に転換しました。mysqladmin および mysql クライアントは、mSQL の対応するプログラムに大きな影響を受けています。MySQL 構文が mSQL のスーパーセットになるよう非常に努力しました。無料の mSQL プログラムを MySQL API に移植しやすくするため、API のアイデアの多くは mSQL に基づいたものです。MySQL ソフトウェアには、mSQL のコードは含まれません。ディストリビューションの 2 つのファイル (client/insert_test.c および client/select_test.c) は、mSQL ディストリビューションの対応する (著作権のない) ファイルに基づいていますが、例として、コードを mSQL から MySQL Server に変換するために必要な変更を示すために変更されています。(mSQL の著作権は David J.Hughes にあります。)

- Patrick Lynch

  http://www.mysql.com/ の取得を支援。

- Fred Lindberg

  MySQL メーリングリストを処理する qmail を設定し、MySQL メーリングリストの管理に関して多大な支援をいただいてます。

- Igor Romanenko <igor@frog.kiev.ua>

  mysqldump (以前の msqldump ですが、Monty により移植および拡張されています)。

- Yuri Dario

  MySQL OS/2 の移植の継続と拡張。

- Tim Bunce

  mysqlhotcopy の作成者。

- Zarko Mocnik <zarko.mocnik@dem.si>

  スロベニア語のソート。

- "TAMITO" <tommy@valley.ne.jp>

  _MB 文字セットのマクロと、ujis および sjis 文字セット。

- Joshua Chamas <joshua@chamas.com>

  同時挿入の基礎、拡張日付構文、NT でのデバッグ、および MySQL メーリングリストでの回答。

- Yves Carlier <Yves.Carlier@rug.ac.be>

  ユーザーのアクセス権を表示するプログラム mysqlaccess。

- Rhys Jones <rhys@wales.com> (および GWE Technologies Limited)

  初期 JDBC ドライバの 1 つ。

- Dr Xiaokun Kelvin ZHU <X.Zhu@brad.ac.uk>

  初期 JDBC ドライバの 1 つと、ほかの MySQL 関連 Java ツールのさらなる開発。

- James Cooper <pixel@organic.com>

  自分自身のサイトで検索可能なメーリングリストアーカイブのセットアップ。

- Rick Mehalick <Rick_Mehalick@i-o.com>

  MySQL Server のグラフィカルな X クライアントである xmysql。

- Doug Sisk <sisk@wix.com>

  Red Hat Linux 対応 MySQL の RPM パッケージの提供。

- Diemand Alexander V. <axeld@vial.ethz.ch>

  Red Hat Linux-Alpha 対応 MySQL の RPM パッケージの提供。

- Antoni Pamies Olive <toni@readysoft.es>

  Intel および SPARC 対応の多くの MySQL クライアントの RPM バージョンの提供。

- Jay Bloodworth <jay@pathways.sde.state.sc.us>

  MySQL バージョン 3.21 対応 RPM バージョンの提供。

- David Sacerdote <davids@secnet.com>

  DNS ホスト名のセキュアなチェックの方針。

- Wei-Jou Chen <jou@nematic.ieo.nctu.edu.tw>

  中国語 (BIG5) キャラクタのサポート。

- Wei He <hewei@mail.ied.ac.cn>

  中国語 (GBK) 文字セット対応の多くの機能。

- Jan Pazdziora <adelton@fi.muni.cz>

  チェコ語のソート順序。

- Zeev Suraski <bourbon@netvision.net.il>

  FROM_UNIXTIME() の時間書式、ENCRYPT() 関数、および bison のアドバイザ。メーリングリストのアクティブなメンバー。

- Luuk de Boer <luuk@wxs.nl>

  ベンチマークスイートを DBI/DBD に移植 (および拡張)。crash-me とベンチマークの実行の大きな助けとなりました。一部の新規日付関数。mysql_setpermission スクリプト。

- Alexis Mikhailov <root@medinf.chuvashia.su>

  ユーザー定義関数 (UDF)、CREATE FUNCTION および DROP FUNCTION。

- Andreas F.Bobak <bobak@relog.ch>

  ユーザー定義関数の AGGREGATE 拡張。

- Ross Wakelin <R.Wakelin@march.co.uk>

  MySQL-Win32 用 InstallShield の設定の支援。

- Jethro Wright III <jetman@li.net>

  libmysql.dll ライブラリ。

- James Pereria <jpereira@iafrica.com>

  MySQL Server を管理する Win32 GUI ツール Mysqlmanager。

- Curt Sampson <cjs@portal.ca>

  MIT-pthreads の NetBSD/Alpha および NetBSD 1.3/i386 への移植。

- Martin Ramsch <m.ramsch@computer.org>

  MySQL チュートリアル内のサンプル。

- Steve Harvey

  mysqlaccess の安全性の改善。

- Persistent Systems Private Limited の Konark IA-64 Centre

  MySQL Server の Win64 への移植の支援。

- Albert Chin-A-Young.

  Tru64 用アップデートの構成、大きなファイルのサポート、優れた TCP ラッパーのサポート。

- John Birrell

  OS/2 用 pthread_mutex() のエミュレーション。

- Benjamin Pflugmann

  INSERTS を処理する拡張 MERGE テーブル。MySQL メーリングリストのアクティブなメンバー。

- Jocelyn Fournier

  (特に MySQL 4.1 サブクエリーコードの) 無数のバグを発見しレポートしました。

- Marc Liyanage

  OS X パッケージの保守および OS X パッケージの作成方法に関する非常に有益なフィードバックを提供。

- Robert Rutherford

  QNX 移植に関する非常に有益な情報とフィードバックを提供。

- NDB Cluster の以前の開発者

  夏期の学生、修士論文生、従業員など、多くの方がさまざまな方法で参加しました。合計で 100 名を超え、ここで全員の名前を挙げることができません。注目に値するのは、1999 年までコードベースの約 1/3 に貢献してきた Ataullah Dabaghi です。また、NDB Cluster のアーキテクチャーの基礎にブロック、信号、およびクラッシュ追跡機能を提供した AXE システムの開発者にも深く感謝いたします。1992 年から現在まで、開発に予算を割り当てるのに十分なアイデアであることを確信した方々も称賛に値します。

- Google 社

  MySQL ディストリビューション、Mark Callaghan の SMP Performance パッチその他のパッチへの貢献で、Google 社に感謝いたします。

その他の貢献者、バグ発見者、およびテスター: James H.Thompson、Maurizio Menghini、Wojciech Tryc、Luca Berra、Zarko Mocnik、Wim Bonis、Elmar Haneke、<jehamby@lightside>、<psmith@BayNetworks.com>、<duane@connect.com.au>、Ted Deppner <ted@psyber.com>、Mike Simons、Jaakko Hyvatti。

メーリングリストのメンバーから提供された多くのバグレポートおよびパッチ。

MySQL メーリングリストの質問に回答いただいた方々に感謝いたします。

- Daniel Koch <dkoch@amcity.com>

  Irix セットアップ。

- Luuk de Boer <luuk@wxs.nl>

  ベンチマークの質問。

- Tim Sailer <tps@users.buoy.com>

  DBD::mysql の質問。

- Boyd Lynn Gerber <gerberb@zenez.com>

  SCO 関連の質問。

- Richard Mehalick <RM186061@shellus.com>

  xmysql 関連の質問と基本的なインストールの質問。

- Zeev Suraski <bourbon@netvision.net.il>

  Apache モジュール構成の質問 (ログおよび権限)、PHP 関連の質問、SQL 構文関係の質問およびその他の一般的な質問。

- Francesc Guasch <frankie@citel.upc.es>

  一般的な質問。

- Jonathan J Smith <jsmith@wtp.net>

  Linux の OS 固有事項、SQL 構文、および何らかの作業を要するその他の事項に関する質問。

- David Sklar <sklar@student.net>

  PHP および Perl からの MySQL の使用。

- Alistair MacDonald <A.MacDonald@uel.ac.uk>

  柔軟性があり、Linux およびおそらく HP-UX も操作できます。

- John Lyon <jlyon@imag.net>

  .rpm ファイルまたはソースからのコンパイルのいずれかによる、MySQL の Linux システムへのインストールに関する質問。

- Lorvid Ltd. <lorvid@WOLFENET.com>

  簡単な請求/ライセンス/サポート/著作権の問題。

- Patrick Sherrill <patrick@coconet.com>

  ODBC および VisualC++ インタフェースに関する質問。

- Randy Harmon <rjharmon@uptimecomputers.com>

  DBD、Linux、一部の SQL 構文に関する質問。

1.8.2 ドキュメント作成者および翻訳者

MySQL のドキュメント作成、およびドキュメントまたはエラーメッセージ翻訳にご協力いただいた方々を次に記載します。

- Paul DuBois

  このマニュアルを正確でわかりやすいものにするために、継続してご協力いただいています。よく知られているように、Monty と David の英語ドキュメントのリライトも含まれます。

- Kim Aldale

  Monty と David の初期の英語ドキュメントのリライト支援。

- Michael J.Miller Jr. <mke@terrapin.turbolift.com>

  初回の MySQL マニュアルの作成。FAQ (ずいぶん前に MySQL マニュアルになりました) の多くのスペリングまたは言語の修正。

- Yan Cailin

  2000 年初めに、Big5 および HK コード化バージョンが準拠している簡体字中国語に MySQL リファレンスマニュアルをはじめて翻訳。

- Jay Flaherty <fty@mediapulse.com>

  マニュアル内の Perl DBI/DBD セクションの大部分に携わりました。

- Paul Southworth <pauls@etext.org>, Ray Loyzaga <yar@cs.su.oz.au>

  リファレンスマニュアルの校正。

- Therrien Gilbert <gilbert@ican.net>、Jean-Marc Pouyot <jmp@scalaire.fr>

  フランス語のエラーメッセージ。

- Petr Snajdr, <snajdr@pvt.net>

  チェコ語のエラーメッセージ。

- Jaroslaw Lewandowski <jotel@itnet.com.pl>

  ポーランド語のエラーメッセージ。

- Miguel Angel Fernandez Roiz

  スペイン語のエラーメッセージ。

- Roy-Magne Mo <rmo@www.hivolda.no>

  ノルウェー語のエラーメッセージと MySQL 3.21.xx のテスト。

- Timur I.Bakeyev <root@timur.tatarstan.ru>

  ロシア語のエラーメッセージ。

- <brenno@dewinter.com> & Filippo Grassilli <phil@hyppo.com>

  イタリア語のエラーメッセージ。

- Dirk Munzinger <dirk@trinity.saar.de>

  ドイツ語のエラーメッセージ。

- Billik Stefan <billik@sun.uniag.sk>

  スロバキア語のエラーメッセージ。

- Stefan Saroiu <tzoompy@cs.washington.edu>

  ルーマニア語のエラーメッセージ。

- Peter Feher

  ハンガリー語のエラーメッセージ。

- Roberto M.Serqueira

  ポルトガル語のエラーメッセージ。

- Carsten H.Pedersen

  デンマーク語のエラーメッセージ。

- Arjen Lentz

  オランダ語のエラーメッセージの以前の部分翻訳の完成 (一貫性とスペリングについてもチェック)。

1.8.3 MySQL をサポートするパッケージ

MySQL で多くのユーザーが使用するもっとも重要な API/パッケージ/アプリケーションの作成者/保守者のリストを次に記載します。

すべてのパッケージを記載すると保守が非常に困難になるため、すべてを記載することはできません。ほかのパッケージについては、http://solutions.mysql.com/software/ のソフトウェアポータルを参照してください。

- Tim Bunce、Alligator Descartes

  DBD (Perl) インタフェース。

- Andreas Koenig <a.koenig@mind.de>

  MySQL Server 用 Perl インタフェース。

- Jochen Wiedmann <wiedmann@neckar-alb.de>

  Perl DBD::mysql モジュールの保守。

- Eugene Chan <eugene@acenet.com.sg>

  MySQL Server 用 PHP の移植。

- Georg Richter

  MySQL 4.1 のテストとバグ検出。MySQL 4.1 で使用するための新規 PHP 5.0 mysqli 拡張 (API)。

- Giovanni Maruzzelli <maruzz@matrice.it>

  iODBC (Unix ODBC) の移植用。

- Xavier Leroy <Xavier.Leroy@inria.fr>

  LinuxThreads (Linux 上で MySQL Server が使用) の作成者。

1.8.4 MySQL の作成に使用されたツール

MySQL を作成するために使用したツールの一覧を次に示します。ここで、これらのツールを作成した方々に感謝申し上げます。これらのツールがなければ、MySQL は今日の形になりませんでした。

- フリーソフトウェア財団 (Free Software Foundation)

  優れたコンパイラ (gcc)、優れたデバッガ (gdb)、および libc ライブラリを提供 (このライブラリの strto.c を使用して、Linux で動作するいくつかのコードを作成しました)。

- フリーソフトウェア財団と XEmacs 開発チーム

  非常に優れたエディタ/環境を提供。

- Julian Seward

  MySQL のバグ検出に役立った非常に優れたメモリーチェッカーツール valgrind の作成者。これがなければバグ検出は困難だったでしょう。

- Dorothea Lütkehaus および Andreas Zeller

  gdb に対する優れたグラフィカルフロントエンドである DDD (データ表示デバッガ) を提供。

1.8.5 MySQL のサポータ

MySQL Server と MySQL マニュアル の全著作権はオラクル社およびその関連会社が保有しますが、新機能の開発に対する融資、MySQL Server の開発用のハードウェアの提供など、MySQL Server の開発を助成していただいた次の会社に感謝申し上げます。

- VA Linux / Andover.net

  レプリケーションに対する資金提供。

- NuSphere

  MySQL マニュアルの編集。

- Stork Design studio

  1998 年 - 2000 年の MySQL Web サイト。

- Intel

  Windows および Linux プラットフォーム上での開発に貢献。

- Compaq

  Linux/Alpha 上での開発に貢献。

- SWSoft

  埋め込み mysqld バージョンでの開発。

- FutureQuest

  --skip-show-database オプション。

第 2 章 MySQL のインストールと更新

目次

2.1 一般的なインストールガイド 45
2.1.1 MySQL Community Server がサポートしているオペレーティングシステム 45
2.1.2 インストールする MySQL のバージョンと配布の選択 45
2.1.3 MySQL の取得方法 46
2.1.4 MD5 チェックサムまたは GnuPG を用いたパッケージの完全性の確認 47
2.1.5 インストールのレイアウト 56
2.1.6 コンパイラ固有のビルドの特徴 56
2.2 一般的なバイナリを使用した MySQL の Unix/Linux へのインストール 56
2.3 Microsoft Windows に MySQL をインストールする 59
2.3.1 Microsoft Windows 上での MySQL のインストールレイアウト 61
2.3.2 インストール用パッケージの選択 61
2.3.3 MySQL Installer を使用した MySQL の Microsoft Windows へのインストール 62
2.3.4 MySQL Notifier 88
2.3.5 非インストール Zip アーカイブを使用して Microsoft Windows に MySQL をインストールする 99
2.3.6 Microsoft Windows MySQL Server インストールのトラブルシューティング 106
2.3.7 Windows 上の MySQL をアップグレードする 107
2.3.8 Windows でのインストール後の手順 108
2.4 OS X に MySQL をインストールする 109
2.4.1 OS X への MySQL のインストールに関する一般的な注記 110
2.4.2 ネイティブパッケージを使用して OS X に MySQL をインストールする 111
2.4.3 MySQL 起動デーモンのインストール 117
2.4.4 MySQL 起動アイテムのインストール 118
2.4.5 MySQL Preference Pane のインストールと使用 119
2.5 Linux に MySQL をインストールする 123
2.5.1 MySQL Yum リポジトリを使用して MySQL を Linux にインストールする 123
2.5.2 サードパーティーの MySQL 配布を MySQL Yum リポジトリを使用して置換する 127
2.5.3 MySQL APT リポジトリを使用して MySQL を Linux にインストールする 129
2.5.4 MySQL SLES リポジトリを使用して MySQL を Linux にインストールする 129
2.5.5 RPM パッケージを使用して MySQL を Linux にインストールする 129
2.5.6 オラクルからの Debian パッケージを使用して MySQL を Linux にインストールする 134
2.5.7 ネイティブソフトウェアリポジトリから MySQL を Linux にインストールする 136
2.6 Unbreakable Linux Network (ULN) を使用した MySQL のインストール 139
2.7 Solaris および OpenSolaris に MySQL をインストールする 139
2.7.1 Solaris PKG を使用して Solaris に MySQL をインストールする 140
2.7.2 IPS を使用して MySQL を OpenSolaris にインストールする 141
2.8 FreeBSD に MySQL をインストールする 142
2.9 ソースから MySQL をインストールする 143
2.9.1 ソースインストールの MySQL のレイアウト 144
2.9.2 標準ソース配布を使用して MySQL をインストールする 144
2.9.3 開発ソースツリーを使用して MySQL をインストールする 148
2.9.4 MySQL ソース構成オプション 150
2.9.5 MySQL のコンパイルに関する問題 162
2.9.6 MySQL の構成とサードパーティーツール 163
2.10 インストール後のセットアップとテスト 163
2.10.1 Unix 類似システムでのインストール後の手順 164
2.10.2 最初の MySQL アカウントのセキュリティー設定 173
2.11 MySQL のアップグレードとダウングレード 176
2.11.1 MySQL のアップグレード 177
2.11.2 MySQL のダウングレード 186
2.11.3 テーブルまたはインデックスの再構築が必要かどうかのチェック 187
2.11.4 テーブルまたはインデックスの再作成または修復 189
2.11.5 MySQL データベースのほかのマシンへのコピー 190
2.12 環境変数 191
2.13 Perl のインストールに関する注釈 193
2.13.1 Unix に Perl をインストールする 193
2.13.2 Windows に ActiveState Perl をインストールする 194
2.13.3 Perl DBI/DBD インタフェース使用の問題 194

この章では MySQL の取得とインストールの方法を説明します。最初に手順の概要を、後半のセクションで詳細を説明します。MySQL をはじめてインストールするのではなく、現行のバージョンの MySQL を新しいバージョンにアップグレードする場合は、セクション2.11.1「MySQL のアップグレード」 を参照してください。アップグレードの手順およびアップグレード前に考慮すべき問題点に関する情報を記載しています。

別のデータベースシステムから MySQL に移行する場合には セクションA.8「MySQL 5.6 FAQ: 移行」 を参照してください。この資料に中には、移行の問題に関する一般的な質問に対する解答が含まれています。

MySQL のインストールは、一般に次の手順に従います。

1. 使用しているプラットフォームで MySQL が動作し、サポートされているかどうかを判断します。

   どのプラットフォームも同じように MySQL の実行に適しているわけではありません。また、MySQL が動作するといわれているプラットフォームのすべてをオラクル社が公式にサポートしているわけでもありません。

2. インストールする配布を選択します。

   MySQL には使用可能なバージョンがいくつかあり、ほとんどが複数の配布形式で使用可能です。バイナリ (コンパイル済み) プログラムあるいはソースコードを含むパッケージ済み配布を選択できます。不確かな場合にはバイナリ配布をご使用ください。また、弊社の最新の開発に興味があり、新しいコードのテストにご協力頂ける方に最新のソースツリーを公開しています。どのバージョンのどの配布を選択したらよいかは セクション2.1.2「インストールする MySQL のバージョンと配布の選択」 を参照してください。

3. インストールする配布をダウンロードします。

   その手順は、セクション2.1.3「MySQL の取得方法」 を参照してください。配布の完全性を確認するには、セクション2.1.4「MD5 チェックサムまたは GnuPG を用いたパッケージの完全性の確認」 の指示に従ってください。

4. 配布をインストールします。

   MySQL をバイナリの配布からインストールするには、セクション2.2「一般的なバイナリを使用した MySQL の Unix/Linux へのインストール」 を参照してください。

   MySQL をソースの配布からインストールする、あるいは現在の開発ソースツリーからインストールする場合は、セクション2.9「ソースから MySQL をインストールする」 を参照してください。

5. インストール後のセットアップが必要な場合は実行します。

   MySQL のインストール後に、MySQL サーバーが正しく動作していることを確認するには、セクション 2.10「インストール後のセットアップとテスト」 を参照してください。セクション2.10.2「最初の MySQL アカウントのセキュリティー設定」 の情報も参照してください。このセクションでは、MySQL 初期アカウントの安全性確保の方法を説明します。ユーザーがパスワードを割り当てるまで、このアカウントにはパスワードがありません。このセクションはバイナリ配布およびソース配布の MySQL のインストールの両方に適用されます。

6. MySQL のベンチマークスクリプトを実行する場合、Perl の MySQL サポートが必要になります。セクション 2.13「Perl のインストールに関する注釈」 を参照してください。

さまざまなプラットフォームおよび環境での MySQL のインストール手順は、プラットフォーム別に提供されます。

- Unix、Linux、FreeBSD

  一般的なバイナリ (たとえば、.tar.gz パッケージ) を使用して MySQL をほとんどの Linux および Unix プラットフォームにインストールする手順は、セクション2.2「一般的なバイナリを使用した MySQL の Unix/Linux へのインストール」 を参照してください。

  MySQL をすべてソースコード配布またはソースコードリポジトリからビルドする場合の詳細は、セクション 2.9「ソースから MySQL をインストールする」を参照してください。

  インストール、構成、およびソースからのビルドに関するプラットフォーム固有のヘルプは、対応するプラットフォームのセクションを参照してください。

  - Linux (配布固有の方法を含む) は、セクション2.5「Linux に MySQL をインストールする」 を参照してください。
  - Solaris および OpenSolaris (PKG および IPS 形式を含む) は、セクション2.7「Solaris および OpenSolaris に MySQL をインストールする」 を参照してください。

  - IBM AIX は、セクション2.7「Solaris および OpenSolaris に MySQL をインストールする」 を参照してください。
  - FreeBSD は、セクション2.8「FreeBSD に MySQL をインストールする」 を参照してください。
- Microsoft Windows

  バイナリ zip または MSI パッケージを使用して、MySQL を Microsoft Windows にインストールする方法については、 セクション2.3「Microsoft Windows に MySQL をインストールする」 を参照してください。

  Microsoft Visual Studio を使用して MySQL をソースコードからビルドする方法の詳細は、セクション2.9「ソースから MySQL をインストールする」 を参照してください。
- OS X

  OS X のインストール (バイナリパッケージおよびネイティブ PKG 形式の両方の使用を含む) の詳細は、セクション2.4「OS X に MySQL をインストールする」 を参照してください。

  MySQL 起動アイテムを使用して MySQL を自動的に起動および停止する方法の詳細は、セクション 2.4.4「MySQL 起動アイテムのインストール」 を参照してください。

  MySQL Preference Pane については、セクション2.4.5「MySQL Preference Pane のインストールと使用」 を参照してください。

2.1 一般的なインストールガイド

次の数セクションには、配布の選択、ダウンロード、および確認に必要な情報が含まれています。この章の次のセクション以降で、選択した配布のインストールの方法を説明します。バイナリ配布については、セクション 2.2「一般的なバイナリを使用した MySQL の Unix/Linux へのインストール」 の指示、または使用しているプラットフォームに対応するセクションがあればそちらを参照してください。MySQL をソースからビルドするには、セクション2.9「ソースから MySQL をインストールする」 を参照してください。

2.1.1 MySQL Community Server がサポートしているオペレーティングシステム

MySQL は、多数のオペレーティングシステムおよびプラットフォームで使用可能です。公式にサポートされているプラットフォームの詳細は、MySQL Web サイトの https://www.mysql.com/support/supportedplatforms/database.html を参照してください。

2.1.2 インストールする MySQL のバージョンと配布の選択

MySQL のインストールを準備する際、使用するバージョン、およびインストールに使用する配布の形式 (バイナリまたはソース) を決定してください。

まず、開発リリースまたは GA リリースのどちらをインストールするかを決定します。開発リリースには最新の機能がありますが、本番での使用は推奨されません。GA (一般提供) リリースは本番または安定リリースとも呼ばれ、本番での使用を意図しています。最新の GA リリースを使用することを推奨します。

MySQL 5.6 の命名スキームでは、mysql-5.6.1-m1 のように 3 つの数字とサフィクスで構成されるリリース名を使用します。リリース名の番号は次のように解釈されます。

- 最初の (5) はメジャーバージョンおよびファイル形式を表します。MySQL 5 のすべてのリリースのファイル形式は同じです。
- 2 番目の番号 (6) はリリースレベルです。メジャーバージョン番号とリリースレベルは、合わせてリリースのシリーズ番号を表します。
- 3 番目の番号 (1) はリリースシリーズ内でのバージョン番号です。これは新しいリリース毎に数が増えます。一般的には、選択したシリーズの最新のバージョンを使用します。

マイナーな更新が行われるたびに、バージョン文字列の最後の数字が大きくなります。メジャーな機能の追加あるいは旧バージョンとのマイナーな非互換性が加えられた場合には、バージョン文字列の 2 番目の番号が大きくなります。ファイル形式が変更になった場合、最初の番号の値が増えます。

リリース名には、リリースの安定性を示すサフィクスが含まれる場合もあります。シリーズ内のリリースは、サフィクスが順に進展していくことで、安定性レベルの改善を示します。サフィクスには次のようなものがあります。

- サフィクスがない場合、そのリリースは一般提供 (GA) すなわち本番リリースを意味します。GA リリースは安定しており、それまでのリリース段階をすべて合格したもので、信頼性があり重大なバグのない、本番システムの使用に適したものだと認識されています。そのリリースには、重大なバグ修正のみが適用されます。

- mN (m1、m2、m3、など) は、マイルストーン番号を表します。MySQL の開発では、マイルストーンモデルを使用し、各マイルストーンは、徹底的にテストされた機能の小規模なサブセットにしっかりと焦点を当てた、少数のバージョンを進展していきます。あるマイルストーンのリリース後、開発は次の小規模な機能セット (これも徹底的にテストされます) に焦点を当てて、別の少数のリリースに進みます。マイルストーンリリース内の機能は、試作レベルの品質であるとみなされます。

- rc はリリース候補を意味します。リリース候補は安定しているとみなされ、MySQL の社内のすべてのテストに合格したもので、すべての既知の致命的なランタイムのバグは修正されています。しかしながら、これらのリリースはすべてのバグが特定されたと確信するに足る広範な使用事例を経たものではありません。マイナーな修正のみが追加されています。

インストールする MySQL のバージョンが決まったら、オペレーティングシステムにインストールする配布を決定しなければなりません。ほとんどのユースケースでは、バイナリ配布を選択するのが適切です。バイナリ配布は、Linux の RPM パッケージや OS X の DMG パッケージなど、多くのプラットフォームでネイティブ形式が利用可能です。配布は、Zip アーカイブや圧縮 tar ファイルなど、より一般的な形式でも利用可能です。Windows では、MySQL Installer を使用してバイナリ配布をインストールできます。

環境によっては、MySQL をソースの配布でインストールしたほうがよい場合もあります。

- MySQL を明示的な場所にインストールしたい場合。標準のバイナリの配布はインストールの場所にかかわらずすぐに動作しますが、MySQL コンポーネントを希望する場所に配置することで柔軟性をさらに向上させる必要に駆られる場合があります。

- mysqld を、標準のバイナリの配布には含まれていない機能が確実に使用できるように構成したい場合。機能が確実に使用できるようにするためのもっとも一般的な予備オプションの一覧を次に示します。

  - TCP ラッパーサポートのための -DWITH_LIBWRAP=1。

  - 圧縮に依存する機能のための -DWITH_ZLIB={system|bundled}

  - デバッグサポートのための -DWITH_DEBUG=1

  詳細は セクション2.9.4「MySQL ソース構成オプション」 を参照してください。

- mysqld を、標準のバイナリの配布に含まれる一部の機能を使用しないように構成する場合。たとえば、配布は通常すべての文字セットをサポートするようにコンパイルされています。小規模な MySQL Server を希望される場合、必要な文字セットのみのサポートで再コンパイルできます。

- いずれかの Git リポジトリから最新のソースを使用して、現在のすべてのバグ修正にアクセスできるようにする場合。たとえば、バグが見つかりそれを MySQL の開発チームにレポートすると、バグ修正がソースのリポジトリにコミットされ、そこにアクセスできます。バグの修正は、リリースが実際に公開されるまでそのリリースには表示されません。

- MySQL を構成する C および C++ コードを読んだり修正したりする場合。この目的のためには、ソースの配布を取得するとよいでしょう。

- ソースの配布には、バイナリの配布より多くのテストおよび例が含まれています。

2.1.3 MySQL の取得方法

MySQL の最新バージョンおよびダウンロードの説明書については、ダウンロードページ https://dev.mysql.com/downloads/ を参照してください。MySQL のダウンロードのミラーサイトの最新の完全なリストは、https://dev.mysql.com/downloads/mirrors.html を参照してください。そこには、MySQL ミラーサイトになる方法および不良ミラーおよび旧式ミラーのレポート方法に関する情報もあります。

パッケージ管理システムとして Yum を使用する RPM ベースの Linux プラットフォームでは、MySQL Yum Repository を使用して MySQL をインストールできます。詳細は、セクション2.5.1「MySQL Yum リポジトリを使用して MySQL を Linux にインストールする」 を参照してください。

多くの Debian ベースの Linux プラットフォームでは、MySQL APT Repository を使用して MySQL をインストールできます。詳細は、セクション2.5.3「MySQL APT リポジトリを使用して MySQL を Linux にインストールする」 を参照してください。

SUSE Linux Enterprise Server (SLES) プラットフォームでは、MySQL SLES Repository を使用して MySQL をインストールできます。詳細は、セクション2.5.4「MySQL SLES リポジトリを使用して MySQL を Linux にインストールする」 を参照してください。

最新の開発ソースの取得については、セクション2.9.3「開発ソースツリーを使用して MySQL をインストールする」 を参照してください。

2.1.4 MD5 チェックサムまたは GnuPG を用いたパッケージの完全性の確認

お客様の要件に適した MySQL のパッケージをダウンロードしてインストールする前に、そのパッケージが元のままで改ざんされていないことを確認する必要があります。完全性の確認には 3 つの方法があります。

- MD5 チェックサム
- GnuPG (GNU Privacy Guard) を使用した暗号署名
- RPM パッケージの場合、内蔵 RPM 完全性確認メカニズム

次のセクションではこれらの方法の使用方法について説明します。

MD5 チェックサムまたは GPG 署名が一致しない場合には、別のミラーサイトから対応するパッケージを再度ダウンロードします。

2.1.4.1 MD5 チェックサムの確認

MySQL パッケージをダウンロードしたら、MD5 チェックサムが MySQL のダウンロードページのチェックサムと一致することを確認してください。各パッケージには固有のチェックサムがあり、ダウンロードしたパッケージに対して確認できます。各 MySQL 製品に対して、正しい MD5 チェックサムがダウンロードページに記載されているので、ダウンロードしたファイル (製品) の MD5 チェックサムと比較します。

各オペレーティングシステムおよびセットアップで、MD5 チェックサムをチェックするための独自のツールが提供されています。通常、このコマンドは md5sum という名前か、または md5 という名前で、一部のオペレーティングシステムではまったく提供されていません。Linux の場合は GNU テキストユーティリティーパッケージの一部で、さまざまなプラットフォームに利用できます。ソースコードを http://www.gnu.org/software/textutils/ からダウンロードすることもできます。OpenSSL がインストールされている場合は、代わりにコマンド openssl md5 package_name を使用できます。md5 コマンド行ユーティリティーの Windows 実装は、http://www.fourmilab.ch/md5/ から入手できます。winMd5Sum はグラフィカルな MD5 チェックツールで、http://www.nullriver.com/index/products/winmd5sum から入手できます。Microsoft Windows の例では、名前は md5.exe であるものとします。

Linux および Microsoft Windows の例

```
shell> md5sum mysql-standard-5.6.23-linux-i686.tar.gz
aaab65abbec64d5e907dcd41b8699945  mysql-standard-5.6.23-linux-i686.tar.gz
```

```
shell> md5.exe mysql-installer-community-5.6.23.msi
aaab65abbec64d5e907dcd41b8699945  mysql-installer-community-5.6.23.msi
```

結果のチェックサム (16 進数の文字列) が、ダウンロードページの対応するパッケージの真下のチェックサムと一致していることを確認してください。

> **注記**
>
> アーカイブファイルの内部に含まれるファイルではなく、アーカイブファイル (たとえば .zip、.tar.gz、または .msi ファイル) のチェックサムを必ず確認してください。つまり、内容を抽出する前のファイルを確認します。

2.1.4.2 GnuPG を使用した署名確認

パッケージの完全性および信憑性を確認する別の方法は、暗号署名を使用するものです。これは、MD5 チェックサムを使用するより信頼性が高いですが、手間がかかります。

MySQL のダウンロード可能なパッケージは、GnuPG (GNU Privacy Guard) を使用して署名されます。GnuPG は、Phil Zimmerman 氏の周知の Pretty Good Privacy (PGP) のオープンソース版です。GnuPG の詳細とその取得およびシステムへのインストール方法については、http://www.gnupg.org/ を参照してください。ほとんどの Linux の配布にはデフォルトで GnuPG がインストールされています。GnuPG の詳細は、http://www.openpgp.org/ を参照してください。

特定のパッケージの署名を確認するには、まず公開 GPG ビルド鍵を取得する必要があります。これは http://pgp.mit.edu/ からダウンロードできます。取得する鍵は mysql-build@oss.oracle.com という名前です。また、次のテキストから直接鍵をカット&ペーストできます。

```
-----BEGIN PGP PUBLIC KEY BLOCK-----
Version: GnuPG v1.4.9 (SunOS)

mQGiBD4+owwRBAC14GIfUfCyEDSIePvEW3SAFUdJBtoQHH/nJKZyQT7h9bPlUWC3
RODjQReyCITRrdwyrKUGku2FmeVGwn2u2WmDMNABLnpprWPkBdCk96+OmSLN9brZ
fw2vOUgCmYv2hW0hyDHuvYlQA/BThQoADgj8AW6/0Lo7V1W9/8VuHP0gQwCgvzV3
BqOxRznNCRCRxAuAuVztHRcEAJooQK1+iSiunZMYD1WufeXfshc57S/+yeJkegNW
hxwR9pRWVArNYJdDRT+rf2RUe3vpquKNQU/hnEIUHJRQqYHo8gTxvxXNQc7fJYLV
K2HtkrPbP72vwsEKMYhhr0eKCbtLGfls9krjJ6sBgACyP/Vb7hiPwxh6rDZ7lTnE
kYpXBACmWpP8NJTkamEnPCia2ZoOHODANwpUkP43I7jsDmgtobZX9qnrAXw+uNDI
QJEXM6FSbi0LLtZciNlYsafwAPEOMDKpMqAK6IyisNtPvaLd8lH0bPAnWqcyefep
rv0sxxqUEMcM3o7wwgfN83POkDasDbs3pjwPhxvhz6//62zQJ7Q2TXlTUUwgUmVs
ZWFzZSBFbmdpbmVlcmluZyA8bXlzcWwtYnVpbGRAb3NzLm9yYWNsZS5jb20+iGkE
ExECACkCGyMGCwkIBwMCBBUCCAMEFgIDAQIeAQIXgAIZAQUCUwHUZgUJGmbLywAK
CRCMcY07UHLh9V+DAKCjS1gGwgVI/eut+5L+l2v3ybl+ZgCcD7ZoA341HtoroV3U
6xRD09fUgeq0O015U1FMIFBhY2thZ2Ugc2lnbmluZyBrZXkgKHd3dy5teXNxbC5j
b20pIDxidWlsZEBteXNxbC5jb20+iG8EMBECAC8FAk53Pa0oHSBidWlsZEBteXNx
bC5jb20gd2lsbCBzdG9wIHdvcmtpbmcgc29vbgAKCRCMcY07UHLh9bU9AJ9xDK0o
xJFL9vTl9OSZC4IX0K9AzwCcCrS9cnJyz79eaRjL0s2r/CcljdyIZQQTEQIAHQUC
R6yUtAUJDTBYqAULBwoDBAMVAwIDFgIBAheAABIJEIxxjTtQcuH1B2VHUEcAAQGu
kgCffz4GUEjzXkOi71VcwgCxASTgbe0An34LPr1j9fCbrXWXO14msIADfb5piEwE
ExECAAwFAj4+o9EFgwImALsACgkQSVDhKrJykfIk4QCfWbEeKN+3TRspe+5xKj+k
QJSammIAnjUz0xFWPlVx0f8o38qNG1bq0cU9iEwEExECAAwFAj5CggMFgwIiIokA
CgkQtvXNTca6JD+WkQCgiGmnoGjMojynp5ppvMXkyUkfnykAoK79E6h8rwkSDZou
iz7nMRisH8uyiEYEEBECAAYFAj+s468ACgkQr8UjSHiDdA/2lgCg21IhIMMABTYd
p/IBiUsP/JQLiEoAnRzMywEtujQz/E9ono7H1DkebDa4iEYEEBECAAYFAj+0Q3cA
CgkQhZavqzBzTmbGwwCdFqD1frViC7WRt8GKoOS7hzNN32kAnirlbwpnT7a6NOsQ
83nk11a2dePhiEYEEBECAAYFAkNbs+oACgkQi9gubzC5S1x/dACdELKoXQKkwJN0
gZztsM7kjsIgyFMAnRRMbHQ7V39XC90OIpaPjk3a01tgiEYEExECAAYFAkTxMyYA
CgkQ9knE9GCTUwwKcQCgibak/SwhxWH1ijRhgYCo5GtM4vcAnAhtzL57wcw1Kg1X
m7nVGetUqJ7fiEwEEBECAAwFAkGBywEFgwYi2YsACgkQGFnQH2d7oexCjQCcD8sJ
NDc/mS8m8OGDUOx9VMWvcnGkAnj1YWOD+Qhxo3ml/Ul9oEAhNkjcfiEwEEBECAAwF
AkGByzQFgwYi2VgACgkQgcL36+ITtpIiIwCdFVNVUB8xe8mFXoPm4d9Z54PTjpMA
niSPA/ZsfJ3oOMLKar4F0QPPrdrGiEwEEBECAAwFAkGBy2IFgwYi2SoACgkQa3Ds
2V3D9HMJqgCbBYzr5GPXOXgP88jKzmdbjweqXeEAnRss4G2G/3qD7uhTL1SPT1SH
jWUXiEwEEBECAAwFAkHQkyQFgwXUEWgACgkQfSXKCsEpp8JiVQCghvWvkPqowsw8
w7WSseTcw1tflvkAni+vLHl/DqIly0LkZYn5jzK1dpvfiEwEEBECAAwFAkIrW7oF
gwV5SNIACgkQ5hukiRXruavzEwCgkzL5QkLSypcw9LGHcFSx1ya0VL4An35nXkum
g6cCJ1NP8r2l4NcZWIrqiEwEEhECAAwFAkAqWToFgwd6S1IACgkQPKEfNJT6+GEm
XACcD+A53A5OGM7w750W11ukq4iZ9ckAnRMvndAqn3YTOxxlLPj2UPZiSgSqiEwE
EhECAAwFAkA9+roFgwdmqdIACgkQ8tdcY+OcZZyy3wCgtDcwlaq20w0cNuXFLLNe
EUaFFTwAni6RHN80moSVAdDTRkzZacJU3M5QiEwEEhECAAwFAkEOCoQFgwaWmggA
CgkQOcor9D1qil/83QCeITZ9wlo7XAMjC6y4ZWUL4m+edZsAoMOhRIRi42fmrNFu
vNZbnMGej81viEwEEhECAAwFAkApTQFgwUj/1gACgkQBA3AhXyDn6jjJACcD1A4
UtXk84J13JQyoH9+dy24714Aniwlsso/9ndICJOkqs2j5dlHFq6oiEwEExECAAwF
Aj5NTYQFgwlXVwgACgkQLbt2v63UyTMFDACgIT5G5NVKf5Mj65bFSlPzb92zk2QA
n1uc2h19/lwwrsbIyK/9POJ+JMP7iEwEExECAAwFAkHXgHYFgwXNJBYACgkQZu/b
yM2C/T4/vACfXe67xiSHB80wkmFZ2krb+oz/gBAAnjR2ucpbaonkQQgnC3GnBqmC
vNaJiEwEExECAAwFAkIYgQ4FgwWMI34ACgkQdsEDHKIxbqGg7gCfQi2HcrHn+yLF
uNlH1oSOh48ZM0oAn3hKV0uIRJphonHaUYiUP1ttWgdBiGUEExECAB0FCwcKAwQD
FQMCAxYCAQIXgAUCS3AvygUJEPPzpwASB2VHUEcAAQEJEIxxjTtQcuH1sNsAniYp
YBGqy/HhMnw3WE8kXahOOR5KAJ4xUmWPGYP4I3hKxyNK9OAUbpDVYIh7BDARAgA7
BQJCdzX1NB0AT29wcy4uLiBzaG91bGQgaGF2ZSBiZWVuIGxvY2FsISBJJ20gKnNv
KiBzdHVwaWQuLi4ACgkQOcor9D1qil/vRwCdFo08f66oKLiuEAqzlf9iDlPozEEA
n2EgvCYLCCHjfGosrkrU3WK5NFVgiI8EMBECAE8FAkVvAL9IHQBTaG91bGQgaGF2
ZSBiZWVuIGEgbG9jYWwgc2lnbmF0dXJlLCBvciBzb21ldGhpbmcgLSBXVEYgd2Fz
IEkgdGhpbmtpbmc/AAoJEDnKK/Q9aopfoPsAn3BVqKOalJeF0xPSvLR90PsRlnmG
AJ44oisY7Tl3NJbPgZal8W32fbqgbIkCIgQQAQIADAUCQYHLhQWDBiLZBwAKCRCq
4+bOZqFEaKgvEACCErnaHGyUYa0wETjj6DLEXsqeOiXad4i9aBQxnD35GUgcFofC
/nCY4XcnCMMEnmdQ9ofUuU3OBJ6BNJIbEusAabgLooebP/3KEaiCliyhHYU5jarp
ZAh+Zopgs3Oc11mQ1tIaS69iJxrGTLodkAsAJAeEUwTPq9fHFFzC1eGBysoyFWg4
bIjz/zCII+qyTbFA5g6tRoiXTo8ko7QhY2AA5UGEg+83Hdb6akC04Z2QRErxKAqr
phHzj8XpjVOsQAdAi/qVKQeNKROlJ+iq6+YesmcWGfzeb87dGNweVFDJIGA0qY27
pTb2lExYjsRFN4Cb13NfodAbMTOxcAWZ7jAPCxAPlHUG++mHMrhQXETоZnBFE4nb
nC7vOBNgWdjUgXcpkUCkop4b17BFpR+k8ZtYLSS8p2LLz4uAeCcSm2/msJxT7rC/
FvoH8428oHincqs2ICo9zO/Ud4HmmO0O+SsZdVKIIjinGyOVWb4OOzkAlnnhEZ3o
6hAHcREIsBgPwEYVTj/9ZdC0AO44Nj9cU7awaqgtrnwwfr/o4V2gl8bLSkltZU27
/29HeuOeFGjlFe0YrDd/aRNsxbyb2O28H4sG1CVZmC5uK1iQBDiSyA7Q0bbdofCW
oQzm5twlpKWnY8Oe0ub9XP5p/sVfck4FceWFHwv+/PC9RzSl33lQ6vM2wIkCIgQT
AQIADAUCQp8KHAWDBQWacAAKCRDYwgoJWiRXzyE+D/9uc7z6fIsalfOYoLN60ajA
bQbl/uRKBFugyZ5RoaItusn9Z2rAtn61WrFhu4uCSJtFN1ny2RERg40f56pTghKr
D+YEt+Nze6+FKQ5AbGIdFsR/2bUk+ZZRSt83e14Lcb6ii/fJfzkoIox9ltkifQxq
Y7Tvk4noKu4oLSc8O1Wsfc/y0B9sYUUCmUfcnq58DEmGie9ovUsImyt5NPnveXxp
5UeaRc5Rqt9tK2B4A+7/cqENrdZJbAMSunt2+2fkYiRunAFPKPBdJBsY1sxeL/A9
aKe0viKEXQdAWqdNZKNCi8rd/oOP99/9lMbFudAbX6nL2DSb1OG2Z7NWEqgIAzjm
pwYYPCKeVz5Q8R+if9/fe5+STY/55OaI33fJ2H3v+U435VjYqbrerWe36xJItcJe
```

```
qUzW71fQtXi1CTEl3w2ch7VF5oj/QyjabLnAlHgSlkSi6p7By5C2MnbCHlCfPnIi
nPhFoRcRGPjJe9nFwGs+QblvS/Chzc2WX3s/2SWm4gEUKRX4zsAJ5ocyfa/vkxCk
SxK/erWlCPf/J1T70+i5waXDN/E3enSet/WL7h94pQKpjz8OdGL4JSBHuAVGA+a+
dknqnPF0KMKLhjrgV+L7O84FhbmAP7PXm3xmiMPriXf+el5fZZequQoIagf8rdRH
HhRJxQgI0HNknkaOqs8dtrkCDQQ+PqMdEAgA7+GJfxbMdY4wslPnjH9rF4N2qfWs
EN/lxaZoJYc3a6M02WCnHl6ahT2/tBK2w1QI4YFteR47gCvtgb6O1JHffOo2HfLm
RDRiRjd1DTCHqeyX7CHhcghj/dNRlW2Z0l5QFEcmV9U0Vhp3aFfWC4Ujfs3LU+hk
AWzE7zaD5cH9J7yv/6xuZVw411x0h4UqsTcWMu0iM1BzELqX1DY7LwoPEb/O9Rkb
f4fmLe11EzIaCa4PqARXQZc4dhSinMt6K3X4BrRsKTfozBu74F47D8Ilbf5vSYHb
uE5p/1oIDznkg/p8kW+3FxuWrycciqFTcNz215yyX39LXFnlLzKUb/F5GwADBQf+
Lwqqa8CGrRfsOAJxim63CHfty5mUc5rUSnTslGYEIOCR1BeQauyPZbPDsDD9MZ1Z
aSafanFvwFG6Llx9xkU7tzq+vKLoWkm4u5xf3vn55VjnSd1aQ9eQnUcXiL4cnBGo
TbOWI39EcyzgslzBdC++MPjcQTcA7p6JUVsP6oAB3FQWg54tuUo0Ec8bsM8b3Ev4
2LmuQT5NdKHGwHsXTPtl0klk4bQk4OajHsiy1BMahpT27jWjJlMiJc+IWJ0mghkK
Ht926s/ymfdf5HkdQ1cyvsz5tryVI3Fx78XeSYfQvuuwqp2H139pXGEkg0n6KdUO
etdZWhe70YGNPw1yjWJT1IhUBBgRAgAMBQJOdz3tBQkT+wG4ABIHZUdQRwABAQkQ
jHGNO1By4fUUmwCbBYr2+bBEn/L2BOcnw9Z/QFWuhRMAoKVgCFm5fadQ3Afi+UQl
AcOphrnJ
=443I
-----END PGP PUBLIC KEY BLOCK-----
```

ビルド鍵を個人の公開 GPG 鍵リングにインポートするには、gpg --import を使用します。たとえば、鍵を mysql_pubkey.asc ファイルに保存した場合、インポートコマンドは次のようになります。

```
shell> gpg --import mysql_pubkey.asc
gpg: key 5072E1F5: public key "MySQL Release Engineering
<mysql-build@oss.oracle.com>" imported
gpg: Total number processed: 1
gpg:               imported: 1
gpg: no ultimately trusted keys found
```

公開鍵 ID の 5072E1F5 を使用して、公開鍵サーバーから鍵をダウンロードすることもできます。

```
shell> gpg --recv-keys 5072E1F5
gpg: requesting key 5072E1F5 from hkp server keys.gnupg.net
gpg: key 5072E1F5: "MySQL Release Engineering <mysql-build@oss.oracle.com>"
1 new user ID
gpg: key 5072E1F5: "MySQL Release Engineering <mysql-build@oss.oracle.com>"
53 new signatures
gpg: no ultimately trusted keys found
gpg: Total number processed: 1
gpg:          new user IDs: 1
gpg:        new signatures: 53
```

鍵を RPM 構成にインポートして RPM インストールパッケージを検証する場合、鍵を直接インポートできるはずです。

```
shell> rpm --import mysql_pubkey.asc
```

問題が生じた場合、または RPM 固有の情報が必要な場合は、セクション2.1.4.4「RPM を使用した署名確認」 を参照してください。

公開ビルド鍵をダウンロードしてインポートしたあと、所望の MySQL パッケージとそれに対応する署名をダウンロードします。これもダウンロードページで入手できます。次の表の例に示すように、署名ファイルの名前は配布ファイルと同じで、拡張子 .asc が付いています。

表 2.1 ソースファイルの MySQL パッケージと署名ファイル

ファイルタイプ	ファイル名
配布ファイル	mysql-standard-5.6.23-linux-i686.tar.gz
署名ファイル	mysql-standard-5.6.23-linux-i686.tar.gz.asc

両方のファイルが同じディレクトリにあることを確認してから、次のコマンドを実行して配布ファイルの署名を確認します。

```
shell> gpg --verify package_name.asc
```

ダウンロードしたパッケージが有効である場合は、次のように「Good signature」 (良好な署名) であることが示されます。

```
shell> gpg --verify mysql-standard-5.6.23-linux-i686.tar.gz.asc
gpg: Signature made Tue 01 Feb 2011 02:38:30 AM CST using DSA key ID 5072E1F5
gpg: Good signature from "MySQL Release Engineering <mysql-build@oss.oracle.com>"
```

Good signature メッセージは、ファイルの署名が弊社のサイトに記載されている署名と比較して有効であることを示します。しかし、次のような警告が表示される場合もあります。

```
shell> gpg --verify mysql-standard-5.6.23-linux-i686.tar.gz.asc
gpg: Signature made Wed 23 Jan 2013 02:25:45 AM PST using DSA key ID 5072E1F5
gpg: checking the trustdb
gpg: no ultimately trusted keys found
gpg: Good signature from "MySQL Release Engineering <mysql-build@oss.oracle.com>"
gpg: WARNING: This key is not certified with a trusted signature!
gpg:          There is no indication that the signature belongs to the owner.
Primary key fingerprint: A4A9 4068 76FC BD3C 4567  70C8 8C71 8D3B 5072 E1F5
```

セットアップおよび構成に依存するので、これは正常です。これらの警告の説明は次のとおりです。

- gpg: no ultimately trusted keys found: これは、特定の鍵がユーザーまたはユーザーの信頼する Web によって「最終的に信頼されて」いないことを意味します。これは、ファイルの署名を確認する目的に関しては問題ありません。
- WARNING: This key is not certified with a trusted signature!There is no indication that the signature belongs to the owner.: これは、ユーザーの所有している鍵が弊社の本当の公開鍵であることをどの程度信用しているかに言及しています。これは個人的な判断です。理想的なのは MySQL 開発者が鍵を直接手渡すことですが、一般的には鍵はダウンロードされます。ダウンロードは改ざんされていますか?おそらくそうではないでしょう。しかし、その判断はユーザー次第です。信頼網をセットアップするのが、信頼するための 1 つの方法です。

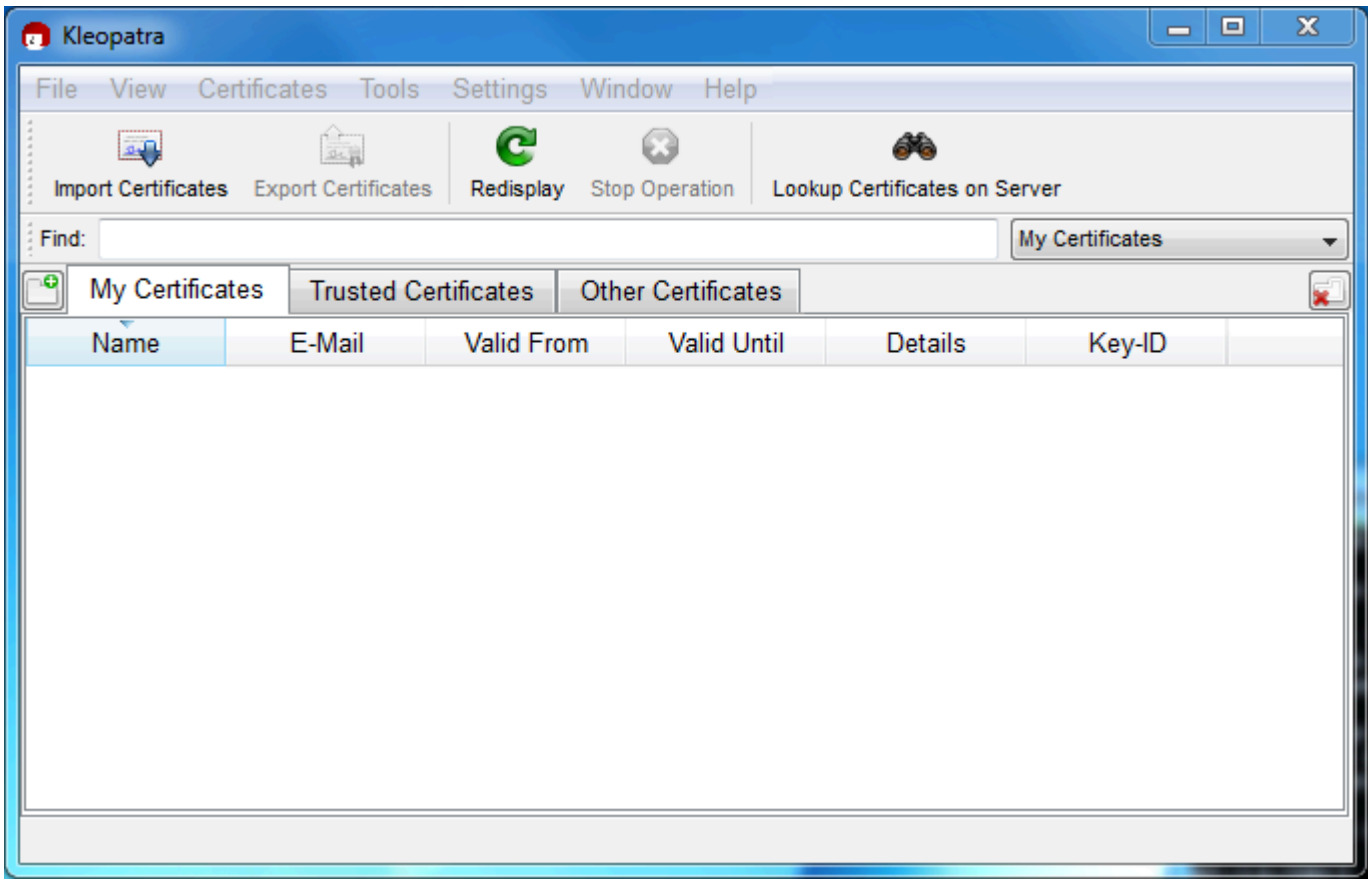
公開鍵を処理する方法の詳細は GPG のドキュメントを参照してください。

2.1.4.3 Gpg4win for Windows を使用した署名確認

セクション2.1.4.2「GnuPG を使用した署名確認」 のセクションでは、GPG を使用して MySQL ダウンロードを確認する方法を説明しています。このガイドは Microsoft Windows にも適用されますが、もう 1 つの選択肢は Gpg4win などの GUI ツールを使用することです。ほかのツールを使用してもかまいませんが、ここでの例は Gpg4win に基づいており、バンドルの Kleopatra GUI を利用しています。

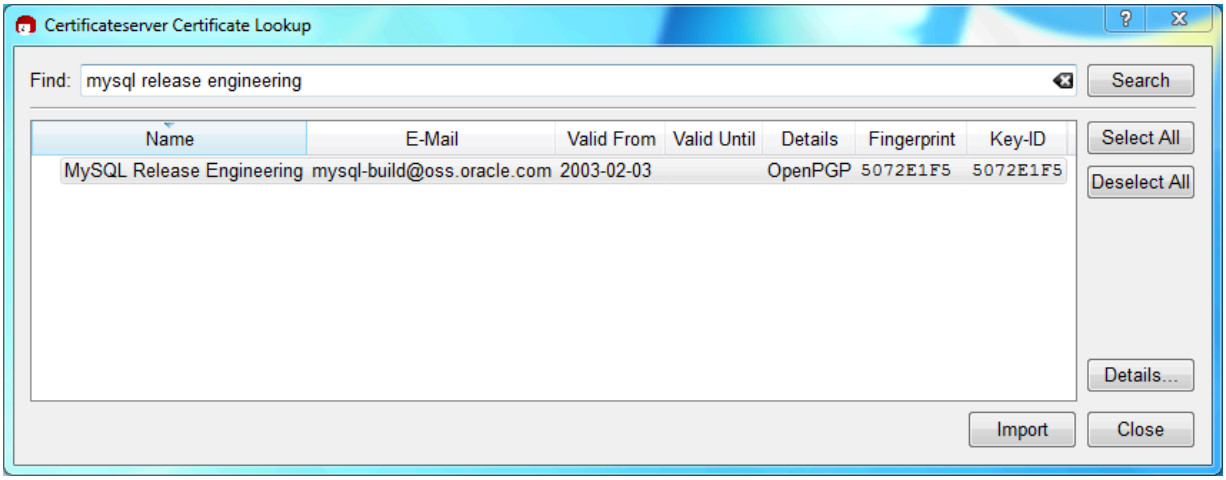
Gpg4win をダウンロードしてインストールしてから Kleopatra をロードします。次のようなダイアログが表示されるはずです。

図 2.1 Kleopatra ロード後の初期画面



次に、MySQL Release Engineering 証明書を追加します。そのためには、「File」、「Lookup Certificates on Server」をクリックします。検索ボックスに「Mysql Release Engineering」と入力して「Search」をクリックします。

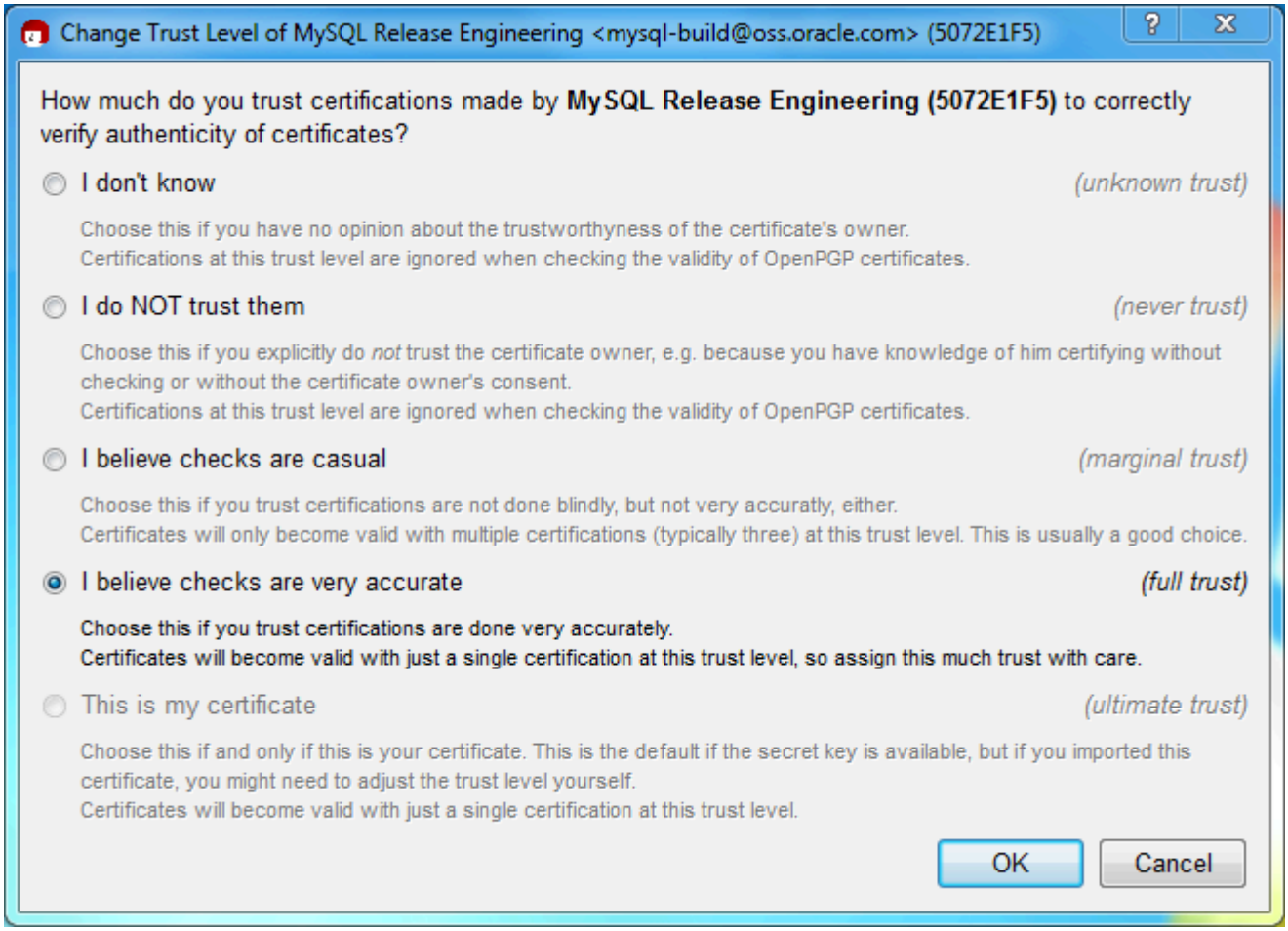
図 2.2 MySQL Release Engineering 証明書の検索



「MySQL Release Engineering」証明書を選択します。Fingerprint および Key-ID は「5072E1F5」でなければなりません。そうでない場合は、「Details...」を選択して証明書が有効であることを確認します。次に、「Import」をクリックしてインポートします。インポートダイアログが表示されます。「Okay」を選択すると、この証明書が「Imported Certificates」タブにリストされます。

次に、弊社の証明書の信頼レベルを構成します。弊社の証明書を選択して、メインメニューから「Certificates」、「Change Owner Trust...」を選択します。弊社の証明書に対して、「I believe checks are very accurate」を選択することを推奨します。そうしないと弊社の署名を確認することができません。「I believe checks are very accurate」を選択してから、「OK」を押します。

図 2.3 信頼レベルの変更



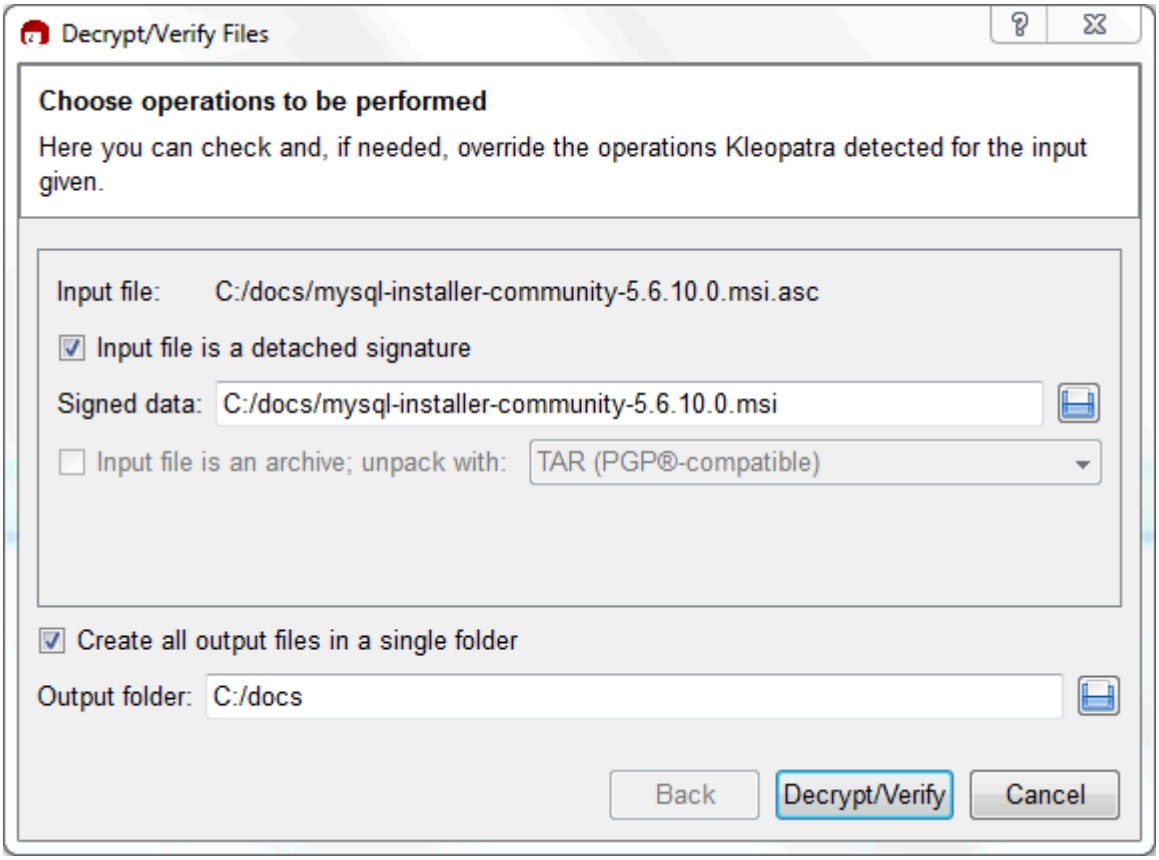
次に、ダウンロードした MySQL パッケージファイルを確認します。これには、パッケージファイルと署名の両方のファイルが必要です。次の表の例に示すように、署名ファイルの名前はパッケージファイルと同じで、拡張子 .asc が付いていなければなりません。署名は各 MySQL 製品のダウンロードページにリンクされています。この署名で .asc ファイルを作成しなければなりません。

表 2.2 MySQL Installer for Microsoft Windows の MySQL パッケージと署名ファイル

ファイルタイプ	ファイル名
配布ファイル	mysql-installer-community-5.6.23.msi
署名ファイル	mysql-installer-community-5.6.23.msi.asc

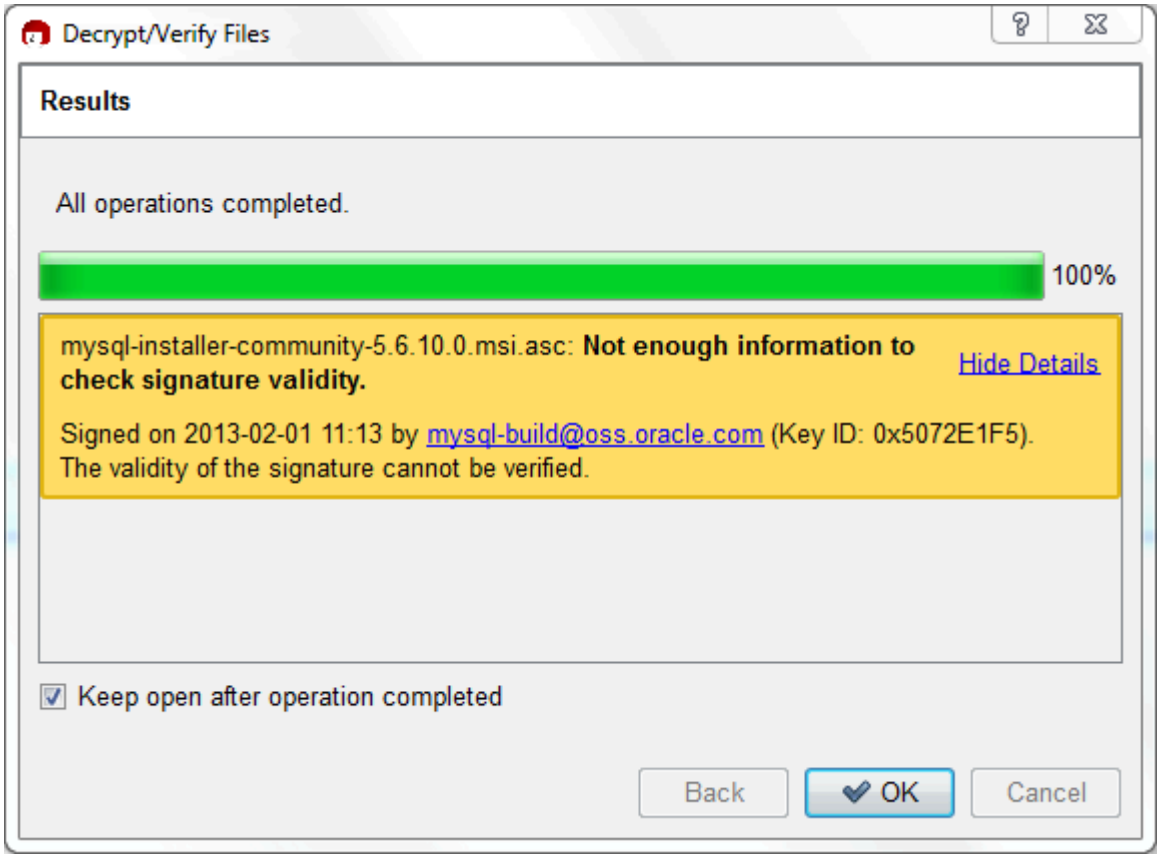
両方のファイルが同じディレクトリにあることを確認してから、次のコマンドを実行して配布ファイルの署名を確認します。署名 (.asc) ファイルを Kleopatra にドラッグ&ドロップするか、「File」、「Decrypt/Verify Files...」からダイアログをロードしてから .msi ファイルまたは .asc ファイルのいずれかを選択します。

図 2.4 「Decrypt/Verify Files」ダイアログ



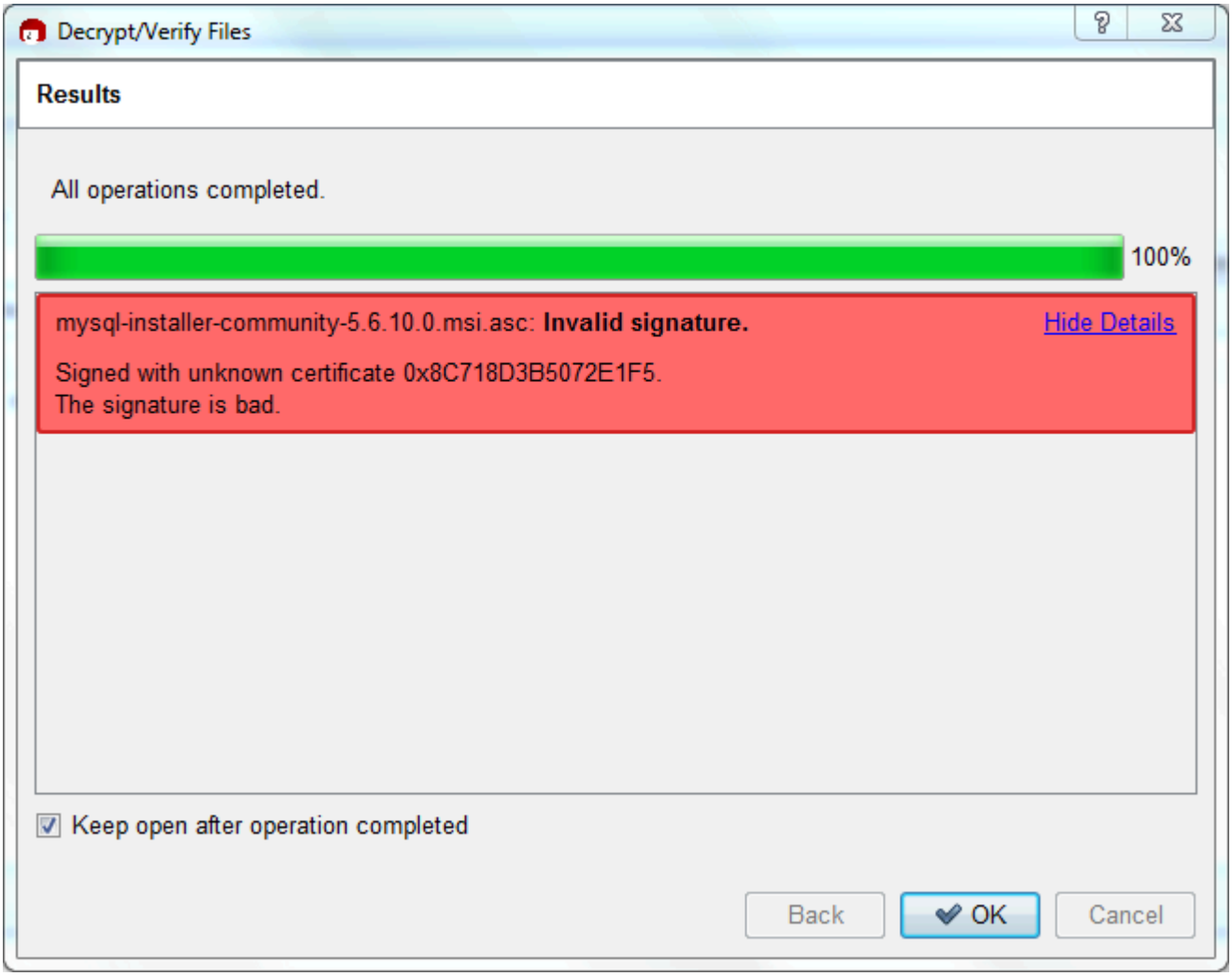
「Decrypt/Verify」をクリックしてファイルをチェックします。もっとも一般的な結果 2 つは次のように表示されます。黄色の警告は問題があるように見えますが、ファイルチェックが成功したことを意味します。このインストーラを実行できます。

図 2.5 Decrypt/Verify の結果: OK



「The signature is bad」というエラーが赤で表示された場合は、ファイルが無効であることを意味します。このエラーが表示された場合は、MSI ファイルを実行しないでください。

図 2.6 Decrypt/Verify の結果: NG



セクション2.1.4.2「GnuPG を使用した署名確認」 セクションでは、緑の「Good signature」という結果がおそらく表示されないであろう理由を説明します。

2.1.4.4 RPM を使用した署名確認

RPM パッケージには、個別の署名はありません。RPM パッケージには内蔵の GPG 署名および MD5 チェックサムがあります。次のコマンドを実行してパッケージを確認できます。

```
shell> rpm --checksig package_name.rpm
```

例:

```
shell> rpm --checksig MySQL-server-5.6.23-0.linux_glibc2.5.i386.rpm
MySQL-server-5.6.23-0.linux_glibc2.5.i386.rpm: md5 gpg OK
```

> **注記**
>
> RPM 4.1 を使用していて、「(GPG) NOT OK (MISSING KEYS:GPG#5072e1f5)」と表示された場合、MySQL 公開ビルド鍵を自分の GPG 鍵リングにインポートした場合でも、この鍵をまず RPM 鍵リングにインポートする必要があります。RPM 4.1 は、お客様個人の GPG 鍵リング (あるいは GPG そのもの) を使用しなくなりました。RPM はシステム全体にわたるアプリケーションであり、ユーザーの GPG 公開鍵リングはユーザー固有のファイルであるため、RPM はむしろ独立した鍵リングを保持します。MySQL 公開鍵を個人の RPM 鍵リングにインポートするには、まず鍵を取得してから、rpm --import を使用して鍵をインポートします。例:

```
shell> gpg --export -a 5072e1f5 > 5072e1f5.asc
```

```
shell> rpm --import 5072e1f5.asc
```

あるいは、rpm は鍵を直接 URL からロードすることもサポートしていますので、このマニュアルページを使用できます。

```
shell> rpm --import https://dev.mysql.com/doc/refman/5.6/en/checking-gpg-signature.html
```

MySQL 公開鍵を取得する必要がある場合には、セクション2.1.4.2「GnuPG を使用した署名確認」 を参照してください。

2.1.5 インストールのレイアウト

インストールのレイアウトは、インストールの種類 (ネイティブパッケージ、バイナリ tarball、ソース tarball など) によって異なります。そのため、異なるシステムを管理する場合や異なるインストールソースを使用する場合に混乱しがちです。個々のレイアウトは、次に記載するように、対応するインストールの種類またはプラットフォームの章にあります。オラクル社以外のベンダーからのインストールのレイアウトは、これらのレイアウトと異なる場合がありますのでご注意ください。

- セクション2.3.1「Microsoft Windows 上での MySQL のインストールレイアウト」
- セクション2.9.1「ソースインストールの MySQL のレイアウト」
- 表2.3「一般的な Unix/Linux バイナリパッケージの MySQL インストールのレイアウト」
- 表2.6「MySQL Developer Zone からの Linux RPM パッケージの MySQL インストールレイアウト」
- 表2.5「OS X 上での MySQL のインストールレイアウト」

2.1.6 コンパイラ固有のビルドの特徴

場合によっては、MySQL のビルドに使用するコンパイラによって、使用できる機能が影響を受けることがあります。このセクションのノートは、オラクル社が提供するバイナリ配布またはご自身でソースからコンパイルしたものに適用されます。

icc (Intel C++ コンパイラ) ビルド

icc でビルドされたサーバーは次の特徴があります。

- SSL のサポートは含まれません。

2.2 一般的なバイナリを使用した MySQL の Unix/Linux へのインストール

オラクル社は、MySQL の一連のバイナリ配布を提供しています。これには、多数のプラットフォーム向けの圧縮 tar ファイル (.tar.gz 拡張子を持つファイル) の形式のバイナリ配布、および選ばれたプラットフォーム向けのプラットフォーム固有のパッケージ形式のバイナリが含まれます。

このセクションでは、圧縮 tar ファイルのバイナリ配布からの MySQL のインストールについて説明します。その他のプラットフォーム固有のパッケージについては、その他のプラットフォーム固有のセクションを参照してください。たとえば、Windows の配布は セクション2.3「Microsoft Windows に MySQL をインストールする」 を参照してください。

MySQL を取得するには、セクション2.1.3「MySQL の取得方法」 を参照してください。

MySQL の圧縮 tar ファイルのバイナリ配布は、mysql-VERSION-OS.tar.gz という形式の名前を持ちます。ここで、VERSION は数字 (5.6.23 など)、OS は配布が対象とするオペレーティングシステムのタイプ (pc-linux-i686 または winx64 など) です。

MySQL を圧縮 tar ファイルのバイナリ配布からインストールするには、配布を展開するための GNU gunzip、およびそれをアンパックするための妥当な tar がシステムになければなりません。使用している tar プログラムが z オプションをサポートする場合は、ファイルの展開とアンパックの両方を実行できます。

GNU tar が機能することが知られています。一部のオペレーティングシステムで提供される標準の tar は、MySQL 配布内の長いファイル名をアンパックできません。GNU tar をダウンロードしてインストー

ルするか、プリインストールバージョンの GNU tar が利用可能であればそれを使用します。通常、これは gnutar、gtar、または tar という名前です (/usr/sfw/bin または /usr/local/bin などの GNU または Free Software ディレクトリ内)。GNU tar は、http://www.gnu.org/software/tar/ から入手可能です。

> **警告**
>
> yum または apt-get などの、使用しているオペレーティングシステムのネイティブパッケージ管理システムを使用して MySQL を以前にインストールしたことがある場合は、ネイティブバイナリを使用してインストールすると問題が生じる場合があります。(パッケージ管理システムを使用して) 以前の MySQL のインストールが完全に削除され、データファイルの古いバージョンなどの追加ファイルもすべて削除されていることを確認してください。/etc/my.cnf などの構成ファイルの存在や、/etc/mysql ディレクトリが削除されていることもチェックしてください。

> **警告**
>
> MySQL には、libaio ライブラリへの依存関係があります。mysql_install_db および後続の mysqld_safe の手順は、このライブラリがローカルにインストールされていない場合は失敗します。必要に応じて、適切なパッケージマネージャーを使用してインストールします。たとえば、Yum ベースのシステムでは次のようにします。
>
> ```
> shell> yum search libaio  # search for info
> shell> yum install libaio # install library
> ```
>
> または、apt-get ベースのシステムでは次のようにします。
>
> ```
> shell> apt-cache search libaio # search for info
> shell> apt-get install libaio1 # install library
> ```

問題が発生してバグをレポートする必要が生じた場合には、セクション1.6「質問またはバグをレポートする方法」 の手順に従ってください。

Unix では、圧縮 tar ファイルのバイナリ配布をインストールするには、選択したインストールの場所 (通常 /usr/local/mysql) でアンパックします。これにより、次の表に示すディレクトリが作成されます。

表 2.3 一般的な Unix/Linux バイナリパッケージの MySQL インストールのレイアウト

ディレクトリ	ディレクトリの内容
bin	クライアントプログラムおよび mysqld サーバー
data	ログファイル、データベース
docs	Info 形式のドキュメント
man	Unix マニュアルページ
include	インクルード (ヘッダー) ファイル
lib	ライブラリ
scripts	mysql_install_db
share	エラーメッセージ、サンプル構成ファイル、データベースインストールのための SQL を含む種々のサポートファイル
sql-bench	ベンチマーク

mysqld バイナリのデバッグバージョンは、mysqld-debug として利用可能です。ソース配布から独自のデバッグバージョンの MySQL をコンパイルするには、適切な構成オプションを使用してデバッグのサポートを有効にします。ソースからのコンパイルの詳細は、セクション2.9「ソースから MySQL をインストールする」 を参照してください。

MySQL バイナリ配布をインストールして使用するには、基本的なコマンドシーケンスは次のようになります。

```
shell> groupadd mysql
shell> useradd -r -g mysql mysql
shell> cd /usr/local
shell> tar zxvf /path/to/mysql-VERSION-OS.tar.gz
shell> ln -s full-path-to-mysql-VERSION-OS mysql
shell> cd mysql
shell> chown -R mysql .
```

```
shell> chgrp -R mysql .
shell> scripts/mysql_install_db --user=mysql
shell> chown -R root .
shell> chown -R mysql data
shell> bin/mysqld_safe --user=mysql &
# Next command is optional
shell> cp support-files/mysql.server /etc/init.d/mysql.server
```

mysql_install_db は、my.cnf という名前のデフォルトオプションファイルを基本インストールディレクトリに作成します。このファイルは my-default.cnf という名前の配布パッケージに含まれるテンプレートから作成されます。詳細は、セクション5.1.2.2「サンプルのデフォルトサーバー構成ファイルの使用」 を参照してください。

バイナリ配布のインストールに関する前述の説明を、次に詳細に説明します。

> **注記**
>
> この手順では、システムへの root (管理者) アクセスがあるものとします。あるいは、sudo (Linux) または pfexec (OpenSolaris) コマンドを使用して、各コマンドにプリフィクスを付けることができます。
>
> 手順では、MySQL アカウントにパスワードはセットアップしません。その手順のあとは、セクション2.10.2「最初の MySQL アカウントのセキュリティー設定」 に進みます。

mysql ユーザーおよびグループの作成

使用しているシステムに、実行するための mysqld のユーザーおよびグループがまだない場合は、作成する必要があるでしょう。次のコマンドは、mysql グループおよび mysql ユーザーを作成します。ユーザーやグループを mysql のではなく別の名前に変更したい場合があります。その場合は、以降の説明では適切な名前に置き換えてください。useradd および groupadd の構文は、Unix のバージョンによって多少異なる場合があり、また adduser および addgroup などの別な名前を使用している場合もあります。

```
shell> groupadd mysql
shell> useradd -r -g mysql mysql
```

> **注記**
>
> このユーザーは、ログイン目的ではなく所有の目的で必要なだけであるため、useradd コマンドでは -r オプションを使用して、サーバーホストへのログイン権限を持たないユーザーを作成します。ユーザーにログインを許可する場合 (または useradd がこのオプションをサポートしない場合) は、このオプションを省略します。

配布の取得とアンパック

配布をアンパックするディレクトリを選択してそこに移動します。この例では、配布を /usr/local の下にアンパックします。したがって、この説明では /usr/local にファイルおよびディレクトリを作成する許可があるものとします。そのディレクトリが保護されている場合は、インストールを root として実行する必要があります。

```
shell> cd /usr/local
```

配布ファイルを セクション2.1.3「MySQL の取得方法」 の説明に従って取得します。所定のリリースでは、すべてのプラットフォームのバイナリの配布は同じ MySQL のソース配布でビルドされています。

配布をアンパックすると、インストールディレクトリが作成されます。次に、そのディレクトリへのシンボリックリンクを作成します。tar が z オプションをサポートする場合、配布の展開とアンパックを実行できます。

```
shell> tar zxvf /path/to/mysql-VERSION-OS.tar.gz
shell> ln -s full-path-to-mysql-VERSION-OS mysql
```

tar コマンドが mysql-VERSION-OS という名前のディレクトリを作成します。ln コマンドがそのディレクトリへのシンボリックリンクを作成します。これにより、インストールディレクトリを /usr/local/mysql として、より簡単に参照できるようになります。

tar が z オプションをサポートしない場合は、gunzip を使用して配布をパックし、tar を使用してアンパックします。前述の tar コマンドを次の代替コマンドに置き換えて、配布を展開して抽出します。

```
shell> gunzip < /path/to/mysql-VERSION-OS.tar.gz | tar xvf -
```

インストール後のセットアップの実行

インストールプロセスの残りの部分は、構成ファイルのセットアップ、コアデータベースの作成、および MySQL Server の開始などです。次の手順については、セクション2.10「インストール後のセットアップとテスト」 を参照してください。

> 注記
>
> MySQL の付与テーブルにリストされているアカウントには、最初はパスワードがありません。サーバーの起動後に セクション2.10.2「最初の MySQL アカウントのセキュリティー設定」 の説明に従ってパスワードをアカウントに設定する必要があります。

2.3 Microsoft Windows に MySQL をインストールする

Microsoft Windows に MySQL をインストールするには、いくつかの方法があります。

単純なインストール方法

もっとも単純で推奨される方法は、MySQL Installer (for Windows) をダウンロードし、それを使用してすべての MySQL 製品をシステムにインストールして構成することです。次に方法を説明します。

- MySQL Installer を https://dev.mysql.com/downloads/installer/ からダウンロードして実行します。

  > 注記
  >
  > 標準の MySQL Installer とは異なり、より小さい「Web コミュニティー」バージョンには MySQL アプリケーションは一切バンドルされていませんが、インストールすることを選択した MySQL 製品がダウンロードされます。

- システムに適した「Setup Type」を選択します。通常、「Developer Default」を選択して MySQL Server および MySQL の開発に関係する、MySQL Workbench などの有用なその他の MySQL ツールをインストールします。あるいは、「Custom」セットアップタイプを選択して、必要な MySQL 製品を手動で選択します。

  > 注記
  >
  > 単一のシステムに複数のバージョンの MySQL Server が存在できます。1 つまたは複数のバージョンを選択できます。

- MySQL インストールウィザードの指示に従って、インストールプロセスを完了します。これにより、いくつかの MySQL 製品がインストールされ、MySQL Server が起動されます。

- MySQL がインストールされました。おそらく、システムを再起動するたびに MySQL Server を自動的に起動するサービスとして、MySQL を構成したでしょう。

  > 注記
  >
  > また、MySQL Workbench および MySQL Notifier などのその他の有用な MySQL 製品もシステムにインストールしたでしょう。新しい MySQL Server 接続のチェック用に 第26章「MySQL Workbench」 を、接続のステータスの表示用に セクション2.3.4「MySQL Notifier」 をロードすることを考慮してください。デフォルトでは、これら 2 つのプログラムは MySQL のインストール後自動的に起動します。

このプロセスでは、MySQL Installer アプリケーションもシステムにインストールされ、あとで MySQL Installer を使用して MySQL 製品のアップグレードや再構成を実行できます。

インストールの追加情報

MySQL は、32 ビットバージョンおよび 64 ビットバージョンの Microsoft Windows に対応しています。サポートされる Windows プラットフォームの情報については、https://www.mysql.com/support/supportedplatforms/database.html を参照してください。

MySQL は、標準アプリケーションまたは Windows サービスとして実行できます。サービスを使用することで、標準 Windows サービス管理ツールを介してサーバーの動作をモニターおよび制御できます。詳細については、セクション2.3.5.7「Windows のサービスとして MySQL を起動する」を参照してください。

一般的には、Windows に MySQL をインストールするには管理者権限のアカウントを使用する必要があります。そうでない場合、PATH 環境変数の編集あるいは サービス管理マネージャーにアクセスするなどの操作の場合に、問題が発生することがあります。インストール後の MySQL は、管理者権限を持つユーザーを使用して実行する必要はありません。

Windows プラットフォームで MySQL を使用する場合の制限のリストは、セクションD.10.6「Windows プラットフォームの制限」 を参照してください。

MySQL Server パッケージのほかに、アプリケーションまたは開発環境で MySQL を使用するために、追加のコンポーネントが必要またはあると便利な場合があります。次のものが含まれますがこれに限りません。

- ODBC を使用して MySQL Server に接続するには、Connector/ODBC ドライバが必要です。インストールおよび構成の説明を含む詳細情報は、MySQL Connector/ODBC Developer Guide を参照してください。

  > **注記**
  >
  > MySQL Installer が、Connector/ODBC のインストールおよび構成を行います。

- MySQL Server を .NET アプリケーションとともに使用するには、Connector/Net ドライバが必要です。インストールおよび構成の説明を含む詳細情報は、MySQL Connector/NET Developer Guide を参照してください。

  > **注記**
  >
  > MySQL Installer が、Connector/NET のインストールおよび構成を行います。

MySQL の Windows 版配布は、https://dev.mysql.com/downloads/ からダウンロードできます。セクション 2.1.3「MySQL の取得方法」を参照してください。

MySQL for Windows は複数の配布形式で利用可能です。次に詳細を説明します。一般的には、MySQL Installer を使用するとよいでしょう。以前の MSI よりも多くの機能と MySQL 製品を含み、ZIP ファイルよりも単純で使いやすく、MySQL を起動して実行するために追加のツールは不要です。MySQL Installer は、自動的に MySQL Server および追加の MySQL 製品をインストールし、オプションファイルを作成してサーバーを起動し、デフォルトのユーザーアカウントを作成できるようにします。パッケージ選択の詳細は、セクション2.3.2「インストール用パッケージの選択」 を参照してください。

- MySQL Installer の配布には MySQL Server、および MySQL Workbench、MySQL Notifier、MySQL for Excel などの追加の MySQL 製品が含まれます。MySQL Installer は、将来これらの製品をアップグレードするときにも使用できます。

  MySQL Installer を使用して MySQL をインストールする方法の説明は、セクション2.3.3「MySQL Installer を使用した MySQL の Microsoft Windows へのインストール」 を参照してください。

- 標準のバイナリ配布 (Zip ファイルとしてパッケージ化) にはすべての必要なファイルが含まれます。任意の場所にアンパックしてください。このパッケージには、完全な Windows MSI Installer パッケージ内のすべてのファイルが含まれますが、インストールプログラムは含まれません。

  Zip ファイルを使用して MySQL をインストールする方法の説明は、セクション2.3.5「非インストール Zip アーカイブを使用して Microsoft Windows に MySQL をインストールする」 を参照してください。

- ソース配布形式には、Visual Studio コンパイラシステムを使用して実行可能ファイルをビルドするためのすべてのコードとサポートファイルが含まれます。

  Windows 上で MySQL をソースからビルドする方法の詳細は、セクション2.9「ソースから MySQL をインストールする」 を参照してください。

Windows 上の MySQL の考慮事項

- 大規模テーブルのサポート

  4G バイト以上のテーブルが必要な場合、NTFS 以降のファイルシステムに MySQL をインストールします。テーブルの作成時には、必ず MAX_ROWS と AVG_ROW_LENGTH を使用します。セクション 13.1.17「CREATE TABLE 構文」を参照してください。

- MySQL およびウィルスチェックソフトウェア

  MySQL のデータおよび一時テーブル含むディレクトリに、Norton/Symantec Anti-Virus などのウィルススキャンソフトウェアがあると、MySQL のパフォーマンス、およびウィルススキャンソフトウェアがファイルの内容をスパムを含むものと誤認するという問題が生じる可能性があります。これは、ウィルススキャンソフトウェ

アのフィンガープリント解析メカニズムと、MySQL がさまざまなファイルを高速で更新する方法が潜在的なセキュリティーリスクと認識される可能性があるということが原因です。

MySQL Server のインストール後、MySQL のテーブルデータを格納するために使用するメインディレクトリ (datadir) 上でのウィルススキャンを無効にすることが推奨されます。通常、ウィルススキャンソフトウェアには特定のディレクトリを無視するシステムが組み込まれており、有効にできます。

また、MySQL はデフォルトで、標準の Windows 一時ディレクトリに一時ファイルを作成します。一時ファイルがスキャンされることも避けるために、MySQL の一時ファイル用に独立した一時ディレクトリを構成し、このディレクトリをウィルススキャンの除外リストに追加します。これを行うには、tmpdir パラメータの構成オプションを my.ini 構成ファイルに追加します。詳細については、セクション2.3.5.2「オプションファイルの作成」 を参照してください。

2.3.1 Microsoft Windows 上での MySQL のインストールレイアウト

Windows 上の MySQL 5.6 では、デフォルトのインストールディレクトリは C:\Program Files\MySQL\MySQL Server 5.6 です。Windows ユーザーの中には、以前デフォルトで使用されていたディレクトリである C:\mysql にインストールすることを好むユーザーもいます。しかし、サブディレクトリのレイアウトは同じです。

すべてのファイルは、次の表に示される構造を使用してこの親ディレクトリ内に配置されています。

表 2.4 Microsoft Windows の MySQL のデフォルトインストールレイアウト

ディレクトリ	ディレクトリの内容	メモ
bin	クライアントプログラムおよび mysqld サーバー	
%ALLUSERSPROFILE%\MySQL\MySQL Server 5.6\	ログファイル、データベース (Windows XP、Windows Server 2003)	Windows システム変数 %ALLUSERSPROFILE% のデフォルトは C:\Documents and Settings\All Users\Application Data です。
%PROGRAMDATA%\MySQL\MySQL Server 5.6\	ログファイル、データベース (Vista、Windows 7、Windows Server 2008 以降)	Windows システム変数 %PROGRAMDATA% のデフォルトは C:\ProgramData です。
examples	プログラムおよびスクリプト例	
include	インクルード (ヘッダー) ファイル	
lib	ライブラリ	
scripts	ユーティリティースクリプト	
share	エラーメッセージ、文字セットファイル、サンプル構成ファイル、データベースインストールのための SQL を含む種々のサポートファイル	

MySQL Installer を使用して MySQL をインストールした場合、このパッケージはインストールされたサーバーが使用するデータディレクトリを作成してセットアップします。また、インストールディレクトリの下に data という名前の新しい「テンプレート」データディレクトリも作成します。このパッケージを使用してインストールを実行したあと、テンプレートデータディレクトリをコピーして追加の MySQL インスタンスをセットアップできます。セクション5.3「1 つのマシン上での複数の MySQL インスタンスの実行」 を参照してください。

2.3.2 インストール用パッケージの選択

MySQL 5.6 では、Windows 上に MySQL をインストールする際にインストールパッケージの形式を選択できます。

- MySQL Installer: このパッケージは mysql-installer-community-5.6.23.0.msi または mysql-installer-commercial-5.6.23.0.msi のようなファイル名を持ち、MSI を利用して MySQL Server およびその他の製品を自動的にインストールします。自分自身およびインストールした各製品のアップデートをダウンロードして適用します。また、追加の非サーバー製品の構成も行います。

  インストールされた製品は構成可能で、これにはサンプルと例を含むドキュメント、コネクタ (C、C++、J、NET、および ODBC など)、MySQL Workbench、MySQL Notifier、MySQL for Excel、および MySQL Server とそのコンポーネントが含まれます。

MySQL Installer は、MySQL がサポートするすべての Windows プラットフォームで実行できます (https://www.mysql.com/support/supportedplatforms/database.html を参照してください)。

> **注記**
>
> MySQL Installer は Microsoft Windows のネイティブコンポーネントではなく .NET に依存するため、Windows Server 2008 の「Server Core」バージョンなどの最小インストールオプションでは動作しません。

MySQL Installer を使用して MySQL をインストールする方法の説明は、セクション2.3.3「MySQL Installer を使用した MySQL の Microsoft Windows へのインストール」 を参照してください。

- 非インストールアーカイブ: このパッケージは mysql-5.6.23-win32.zip または mysql-5.6.23-winx64.zip のようなファイル名を持ち、完全なインストールパッケージ内にある、GUI を除くすべてのファイルを含みます。このパッケージには Automated Installer が含まれていないので、手動でインストールして構成する必要があります。

ほとんどのユーザーには MySQL Installer を推奨します。

インストールのパッケージの選択によってインストールのプロセスが変わります。MySQL Installer を使用することを選択した場合は、セクション2.3.3「MySQL Installer を使用した MySQL の Microsoft Windows へのインストール」 を参照してください。非インストールアーカイブを使用することを選択した場合は、セクション2.3.5「非インストール Zip アーカイブを使用して Microsoft Windows に MySQL をインストールする」 を参照してください。

2.3.3 MySQL Installer を使用した MySQL の Microsoft Windows へのインストール

MySQL Installer を使用すると、MySQL 製品の Microsoft Windows へのインストールと更新が簡潔になります。この中心的なアプリケーションから、使用しているシステム上の MySQL 製品の表示、削除、更新、および再構成を実行できます。MySQL Installer では、プラグイン、ドキュメント、チュートリアル、およびサンプルデータベースのインストールも行えます。MySQL Installer が使用可能なのは Microsoft Windows のみで、GUI とコマンド行インタフェースの両方を含みます。

サポートされる製品は次のとおりです。

- MySQL Server (1 つまたは複数のバージョン)
- MySQL Workbench
- MySQL Connectors (.Net / Python / ODBC / Java / C / C++)
- MySQL Notifier
- MySQL for Excel
- MySQL for Visual Studio
- MySQL Utilities および MySQL Fabric
- MySQL のサンプルと例
- MySQL ドキュメント
- MySQL Installer もインストールされ、独立したアプリケーションとしてシステム上に残ります。

Installer パッケージのタイプ

- Full: すべての MySQL 製品 (MySQL Server を含む) がバンドルされています。ファイルのサイズは 200MB を超え、名前は mysql-installer-community-VERSION.N.msi の形式です。ここで、VERSION は 5.6 などの MySQL Server のバージョン番号、N は 0 から始まるパッケージ番号です。
- Web: Installer と構成ファイルのみを含み、インストールすることを選択した MySQL 製品のみをダウンロードします。このファイルのサイズは約 2MB です。ファイル名は mysql-installer-community-web-VERSION.N.msi の形式で、VERSION は 5.6 などの MySQL Server のバージョン番号、N は 0 から始まるパッケージ番号です。

Installer のエディション

- コミュニティーエディション: https://dev.mysql.com/downloads/installer/ からダウンロードできます。これは、すべての MySQL 製品のコミュニティーエディションをインストールします。
- 商用エディション: My Oracle Support (MOS) または https://edelivery.oracle.com/ からダウンロードできます。これは、Workbench SE/EE を含むすべての MySQL 製品の商用バージョンをインストールします。MOS アカウントとも統合します。

> **注記**
>
> バンドルの MySQL 製品をインストールする際には MOS の資格情報の入力はオプションですが、MySQL Installer がダウンロードする必要のある、バンドルされていない MySQL 製品を選択する場合は視覚情報が必要です。

MySQL Installer の各リリースの変更内容の詳細は、MySQL Installer リリースノート を参照してください。

MySQL Installer は、前から存在するインストールと互換性があり、インストール済みコンポーネントのリストにそれらを追加します。標準の MySQL Installer は MySQL Server の特定のバージョンにバンドルされていますが、単独の MySQL Installer インスタンスで MySQL Server の複数のバージョンのインストールと管理が可能です。たとえば、単独の MySQL Installer インスタンスで 5.5、5.6、および 5.7 のバージョンをホストにインストール (および更新) 可能です。

> **注記**
>
> 単独のホストに MySQL Server のコミュニティーエディションと商用エディションの両方をインストールすることはできません。たとえば、MySQL Server 5.5 と 5.6 の両方を単独のホストにインストールする場合、両方が同じエディションでなければなりません。

MySQL Installer はアプリケーションの初期構成およびセットアップを処理します。例:

1. MySQL Workbench に MySQL Server の初期接続を作成します。
2. MySQL Server の構成に使用される構成ファイル (my.ini) を作成します。このファイルに書き込まれる値は、インストールのプロセスで選択した内容に影響されます。
3. オプションでサンプルデータベースをインポートできます。
4. オプションで、DB Administrator、DB Designer、および Backup Admin などの一般的なロールに基づいた構成可能な権限を持つ MySQL Server ユーザーアカウントを作成できます。オプションで、MysqlSys という名前の制限付き権限を持つ Windows ユーザーを作成し、このユーザーが MySQL Server を実行します。

   MySQL Workbench でユーザーアカウントを追加して構成することもできます。
5. 「Advanced Configuration」オプションをチェックした場合は、「Logging Options」も構成されます。これには、エラーログ、(クエリー実行に必要な秒数の構成を含む) スロークエリーログ、およびバイナリログのファイルパスの定義が含まれます。

MySQL Installer では、オプションで更新されたコンポーネントをチェックしてダウンロードすることも可能です。

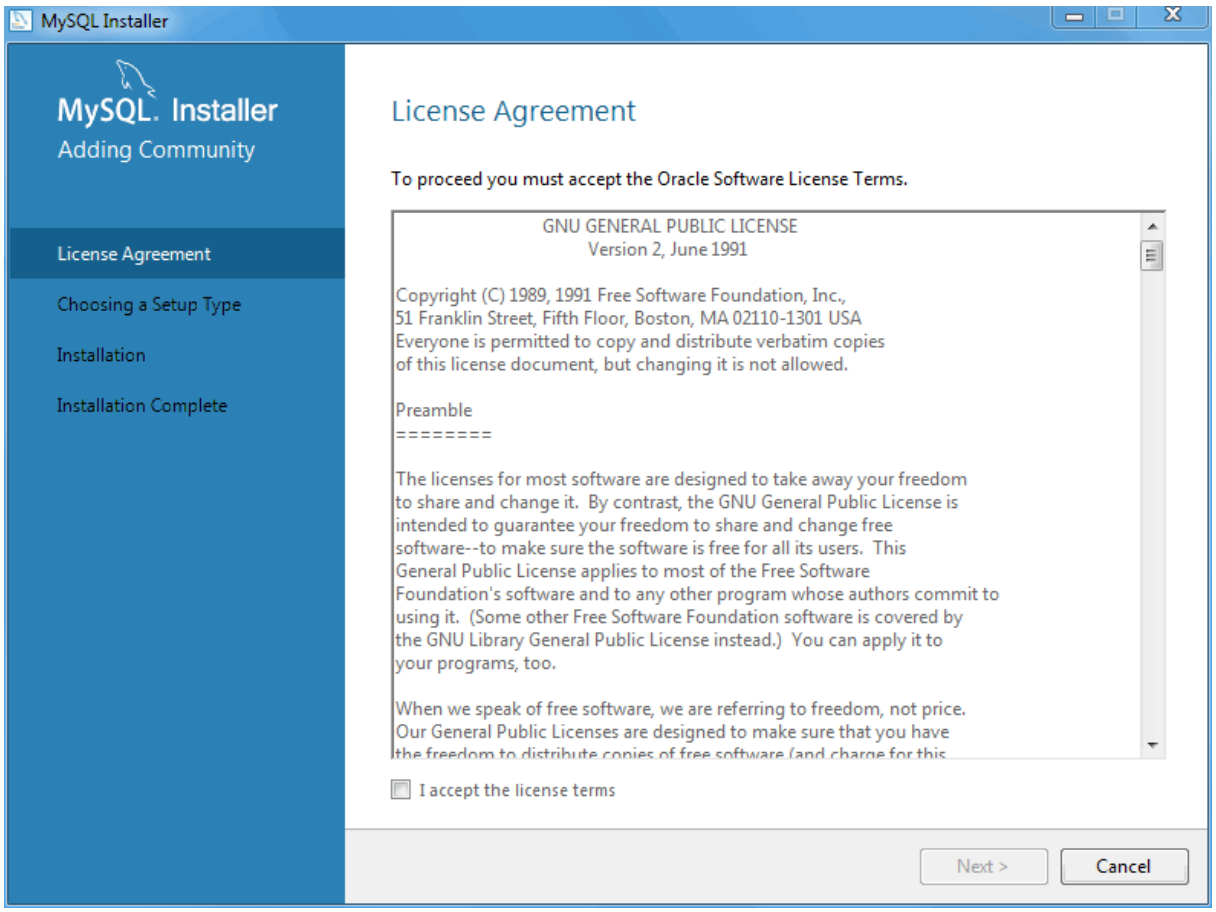
2.3.3.1 MySQL Installer GUI

MySQL Installer をインストールすると、「スタート」メニューに「MySQL」グループへのリンクが追加されます。「スタート」、「すべてのプログラム」、「MySQL」、「MySQL Installer」をクリックして MySQL Installer GUI をリロードします。

> **注記**
>
> MySQL Installer によって生成されたファイルは、my.ini を含めて MySQL Installer を実行するユーザーにフルアクセス権を付与します。これは、SYSTEM が所有する %ProgramData% にある MySQL Server データディレクトリなど、固有の製品のファイルとディレクトリには該当しません。

MySQL Installer を最初に実行するためには、MySQL 製品をインストールする前にライセンス契約を受け入れる必要があります。

図 2.7 MySQL Installer - ライセンス契約



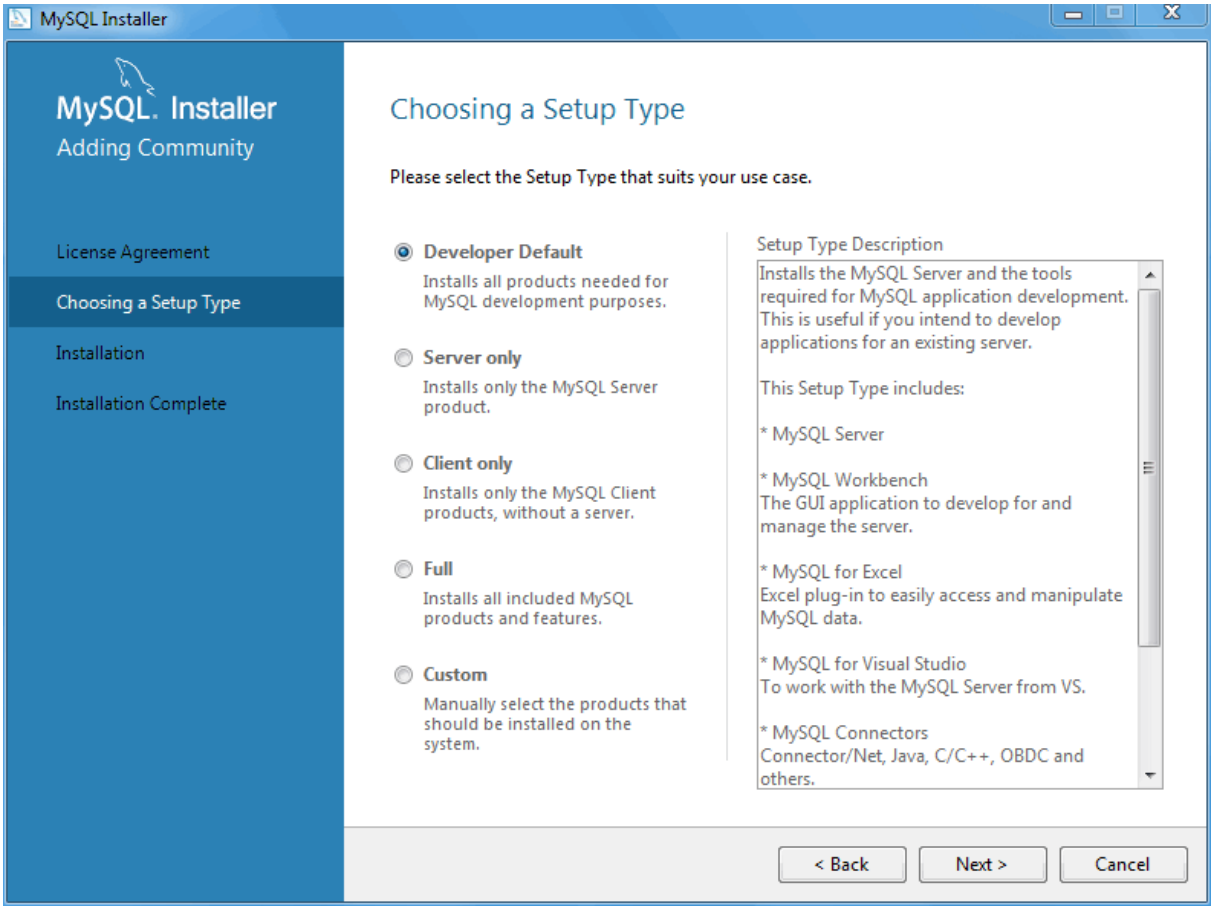
新規パッケージのインストール

システムに適した「Setup Type」を選択します。選択したタイプによって、どの MySQL 製品がシステムにインストールされるかが決まります。あるいは、「Custom」を選択して個々の製品を手動で選択します。

- Developer: MySQL でアプリケーションを開発するために必要なすべての製品をインストールします。これはデフォルトのオプションです。
- Server only: MySQL Server のみをインストールします。
- Client only: MySQL クライアント製品のみをインストールします。MySQL Server は含まれません。
- Full: すべての MySQL 製品をインストールします。
- Custom: インストールする MySQL 製品を手動で選択します。

> **注記**
>
> 初期インストールのあと、MySQL Installer を使用して、インストールまたは削除する MySQL 製品を手動で選択できます。つまり、MySQL Installer は MySQL 製品管理システムになります。

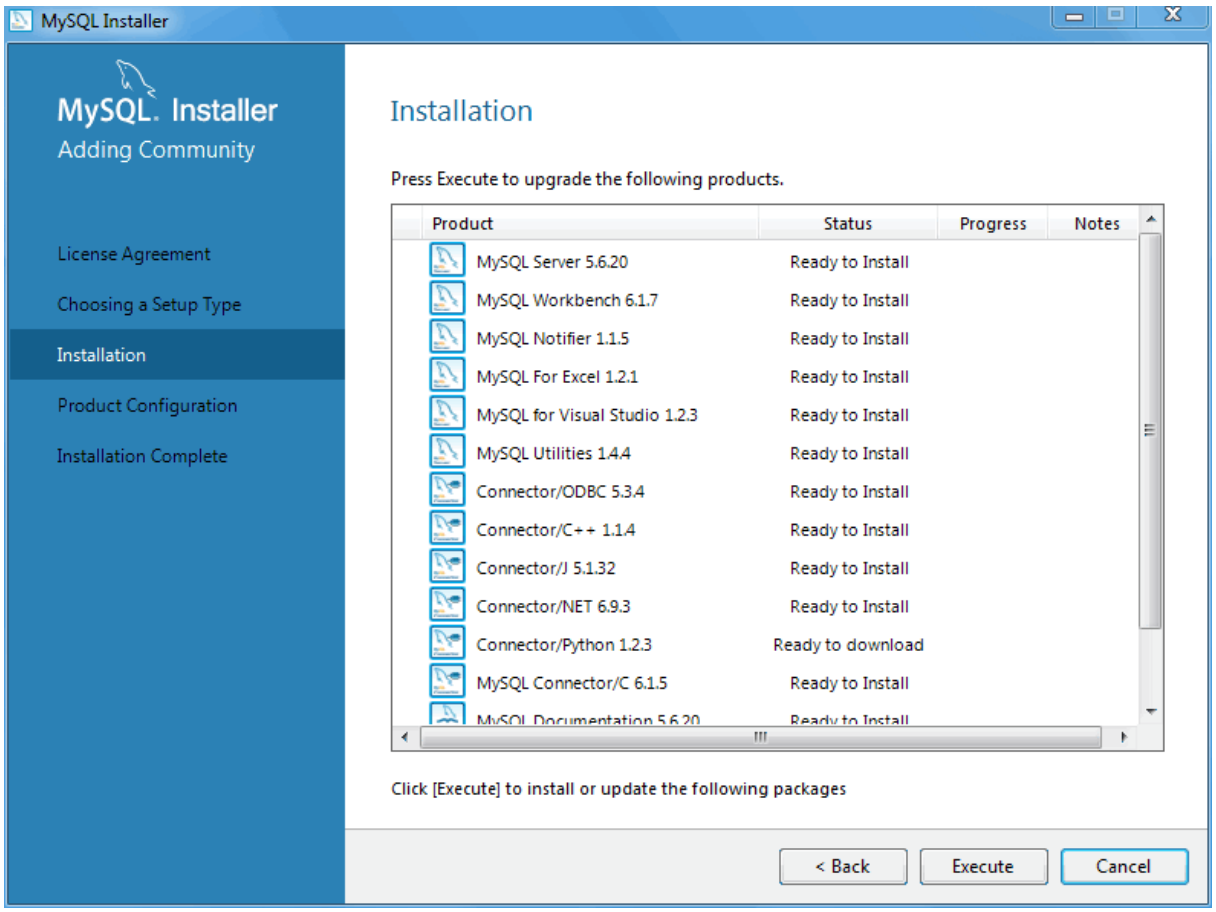
図 2.8 MySQL Installer - セットアップタイプの選択



セットアップタイプを選択すると、MySQL Installer は、選択された各 MySQL 製品の必要な外部要件に関して、システムをチェックします。MySQL Installer は、足りないコンポーネントをダウンロードしてシステムにインストールするか、あるいはダウンロードの場所を指示して「Status」を「Manual」に設定します。

次のウィンドウは、インストールがスケジュールされている MySQL 製品をリストします。

図 2.9 MySQL Installer - インストールの進捗



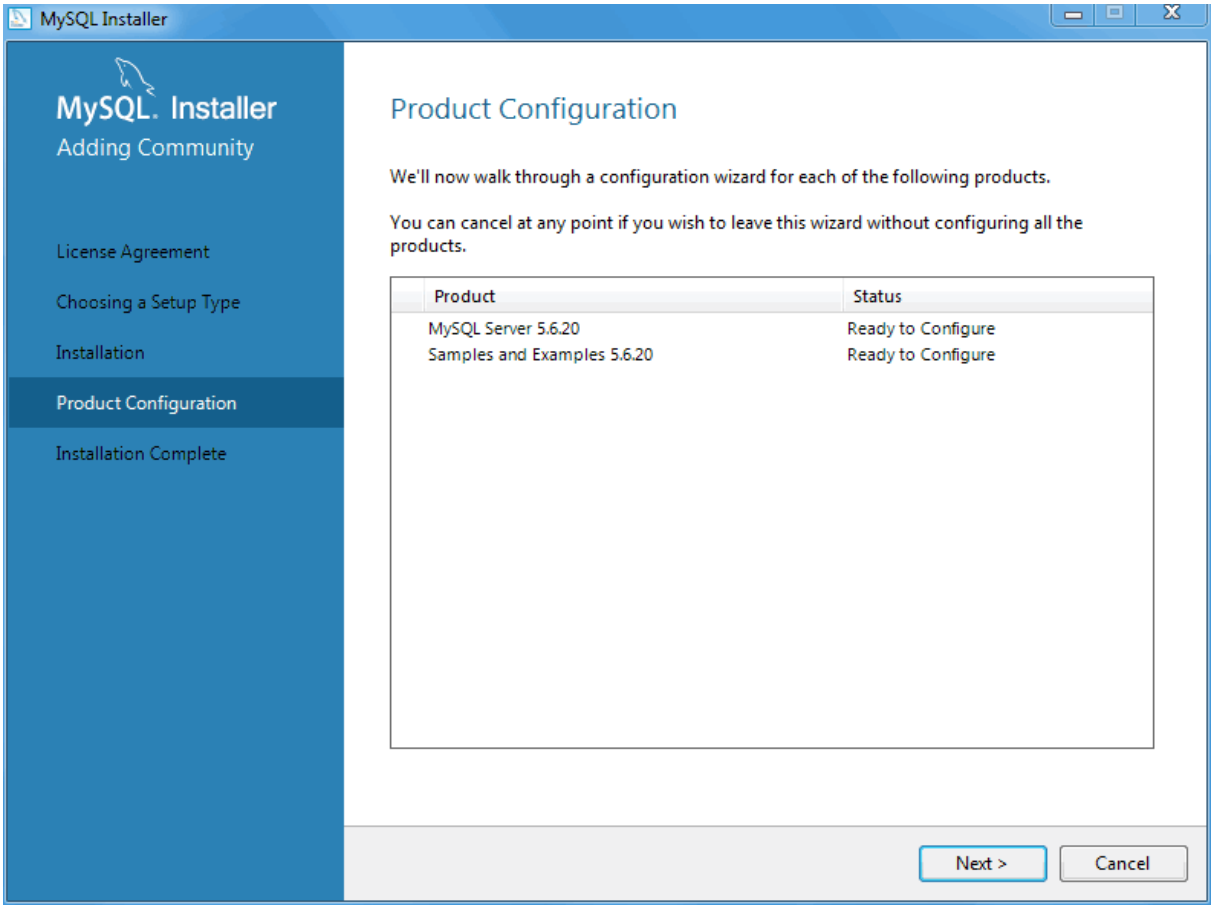
コンポーネントがインストールされると、その「Status」が進捗のパーセントから「Complete」に変わります。

すべてのコンポーネントがインストールされると、次のステップでは最近インストールされた MySQL 製品のいくつかが構成されます。「Configuration Overview」ウィンドウは、進捗を表示してから必要に応じて構成ウィンドウをロードします。この例では、MySQL Server 5.6.x を構成します。

MySQL Server の構成

MySQL Server の構成では、まずいくつかの「Type and Networking」オプションを定義します。

図 2.10 MySQL Installer - 構成概要



Server の構成タイプ

セットアップを記述する MySQL Server の構成タイプを選択します。この設定により、MySQL Server インスタンスに割り当てられるシステムリソースの量が決まります。

- Developer: ほかに多数のアプリケーションをホストするマシンで、通常ユーザーの個人的なワークステーションです。このオプションでは、MySQL が最小のメモリーを使用するように構成されます。
- Server: このマシンでは、Web サーバーなど、ほかにいくつかのアプリケーションが実行されています。このオプションでは、MySQL が中程度のメモリーを使用するように構成されます。
- Dedicated: MySQL Server 実行専用のマシンです。サーバーでは、ほかに Web サーバーのような主要なアプリケーションが実行されていないため、このオプションでは利用可能なすべてのメモリーを MySQL が使用するように構成されます。

接続性

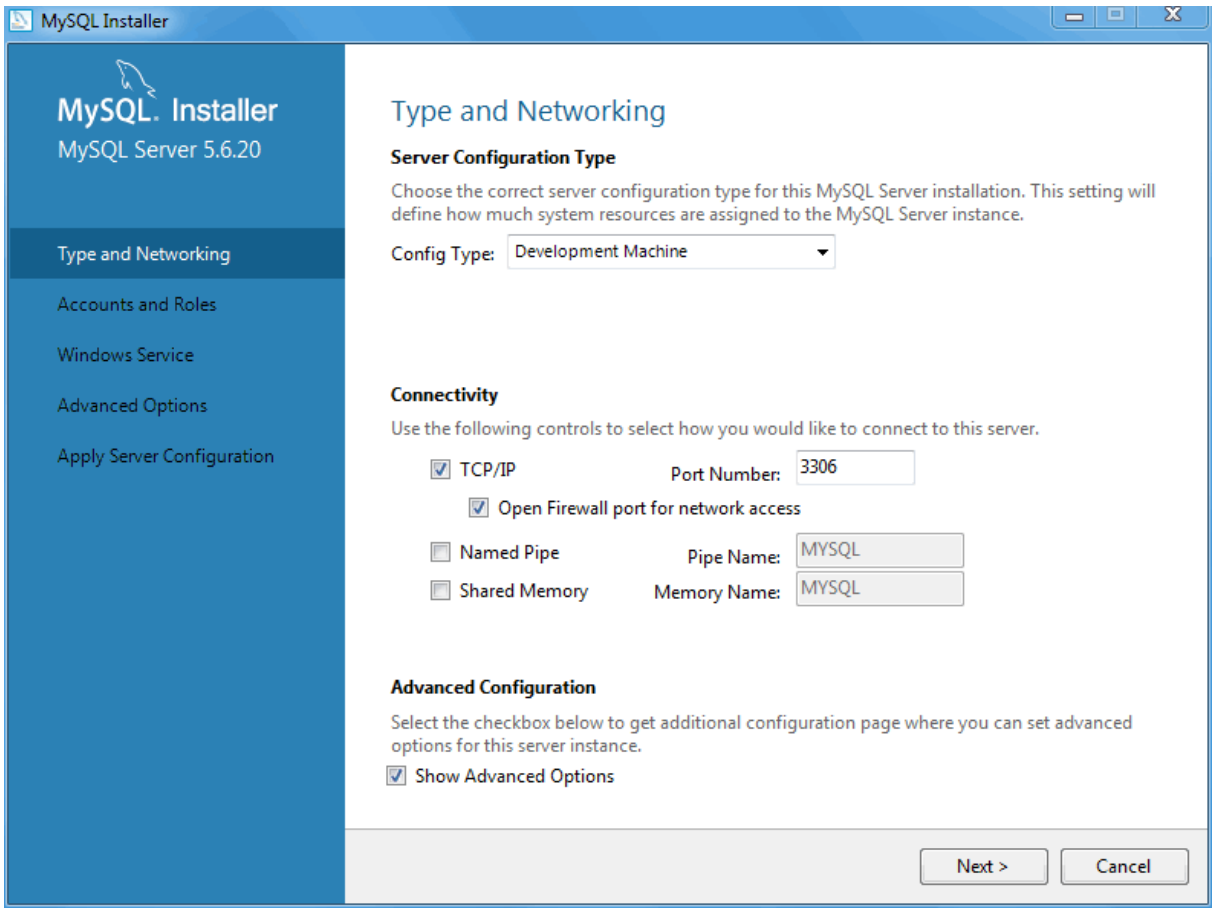
接続性オプションは、MySQL への接続方法を制御します。次のようなオプションがあります。

- TCP/IP: TCP/IP ネットワークを有効にできます。そううしないとローカルホスト接続のみが許可されます。「Port Number」およびネットワークアクセス用にファイアウォールポートをオープンするかどうかも定義します。
- Named Pipe: パイプを有効にしてパイプ名を定義します。--enable-named-pipe オプションを使用するのと同様です。
- Shared Memory: メモリー名を有効にしてから定義します。--shared-memory オプションを使用するのと同様です。

高度な構成

「Advanced Configuration」オプションをチェックした場合は、構成する追加の「Logging Options」が提供されます。これには、エラーログ、(クエリー実行に必要な秒数の構成を含む) スロークエリーログ、およびバイナリログのファイルパスの定義が含まれます。

図 2.11 MySQL Installer - MySQL Server の構成: タイプとネットワーク



アカウントとロール

次に、MySQL のアカウント情報を定義します。ルートパスワードの割り当ては必須です。

オプションで、事前定義されたユーザーロールを持つ MySQL ユーザーアカウントを追加できます。「DB Admin」などの事前定義ロールは、それぞれ独自の権限セットで構成されています。たとえば、「DB Admin」ロールは「DB Designer」ロールより多くの権限を持っています。「Role」ドロップダウンをクリックするとロールの説明のリストが表示されます。

> 注記
>
> MySQL Server がすでにインストールされている場合は、現在の Root のパスワードも入力しなければなりません。

図 2.12 MySQL Installer - MySQL Server 構成: ユーザーアカウントとロール

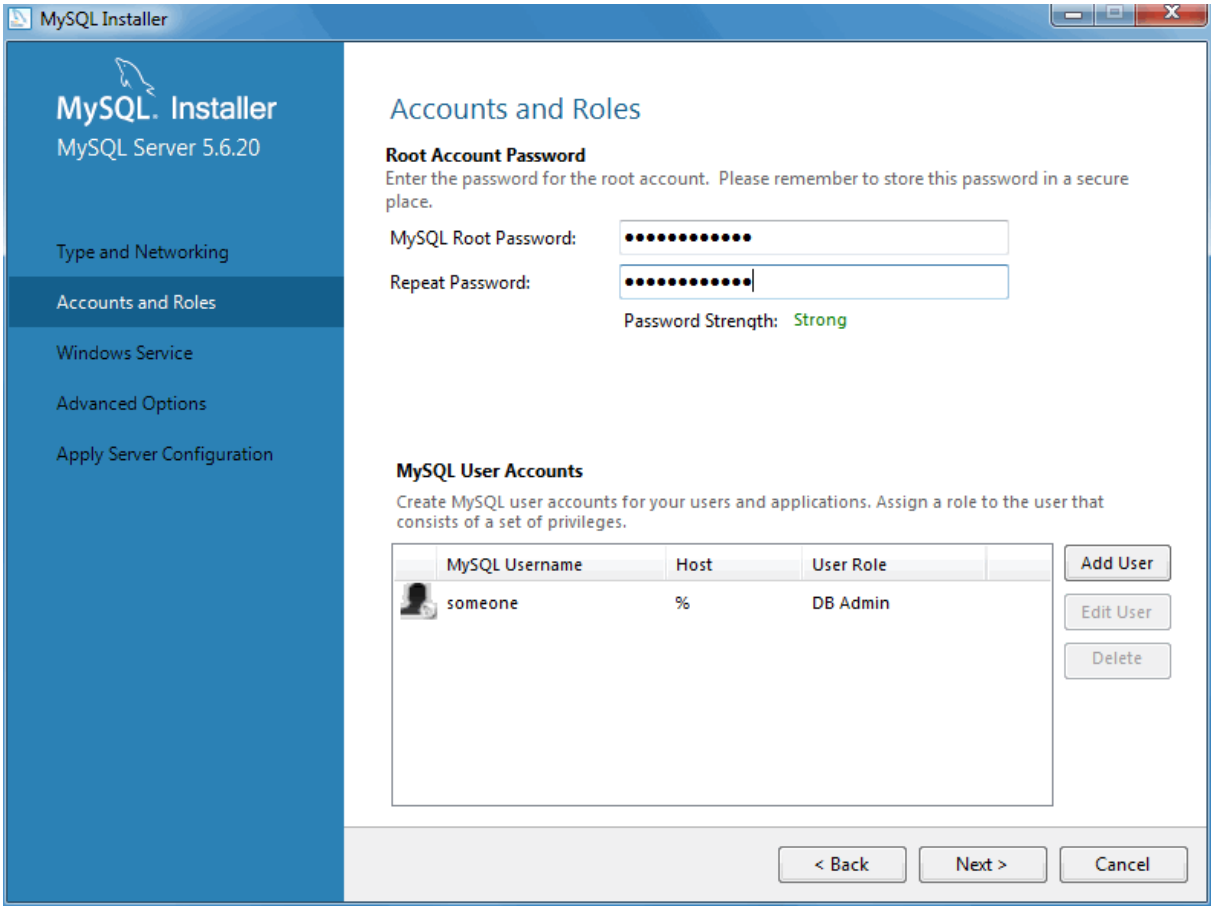
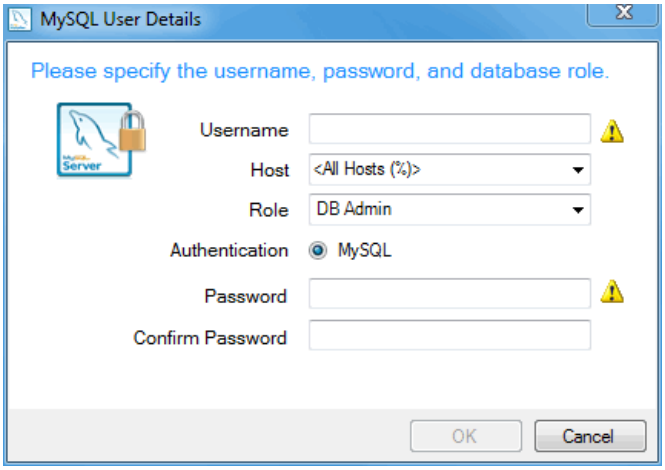


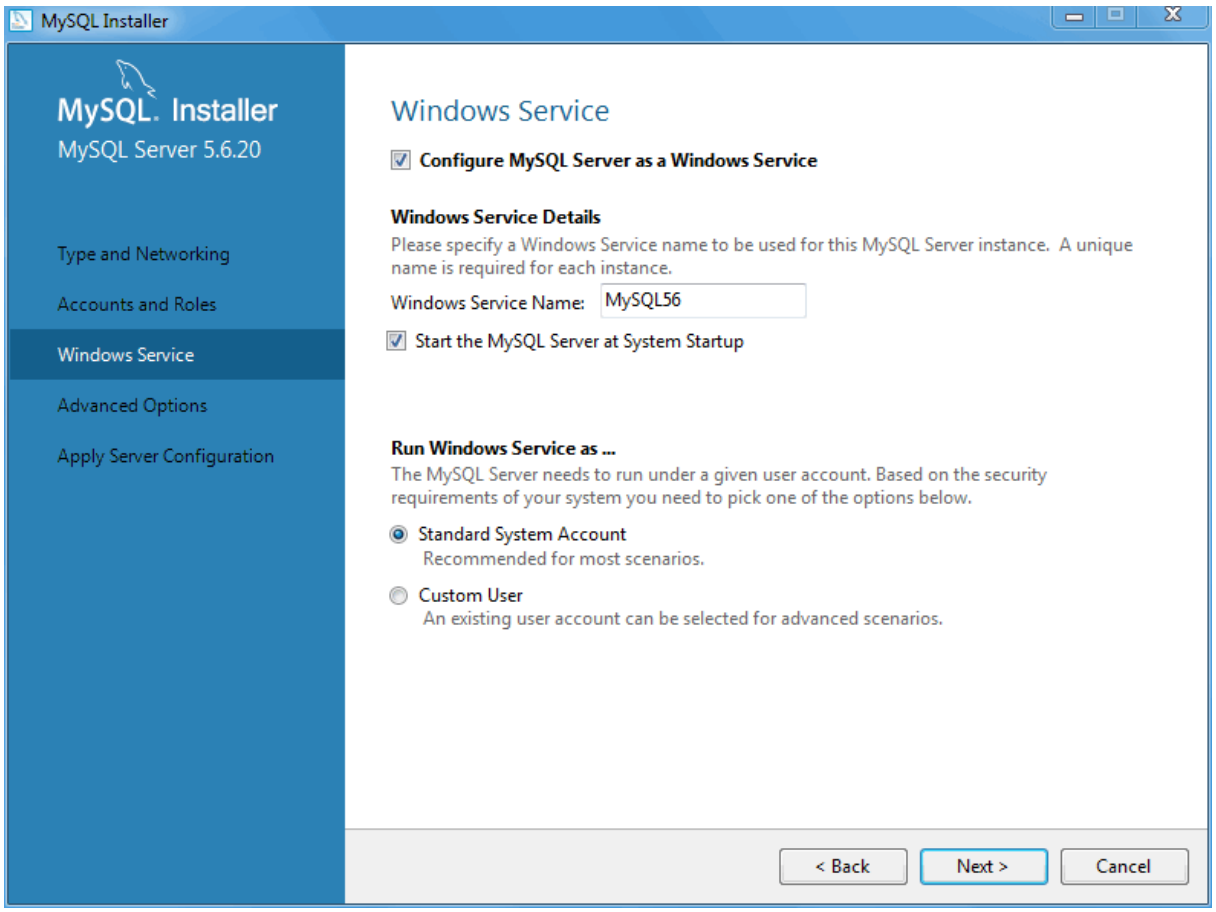
図 2.13 MySQL Installer - MySQL Server 構成: ユーザーアカウントとロール: ユーザーの追加



Windows サービス

次に、「Windows Service」詳細を構成します。これには、サービス名、起動時に MySQL Server をロードするかどうか、および MySQL Server の Windows サービスの実行方法が含まれます。

図 2.14 MySQL Installer - MySQL Server 構成: Windows サービス

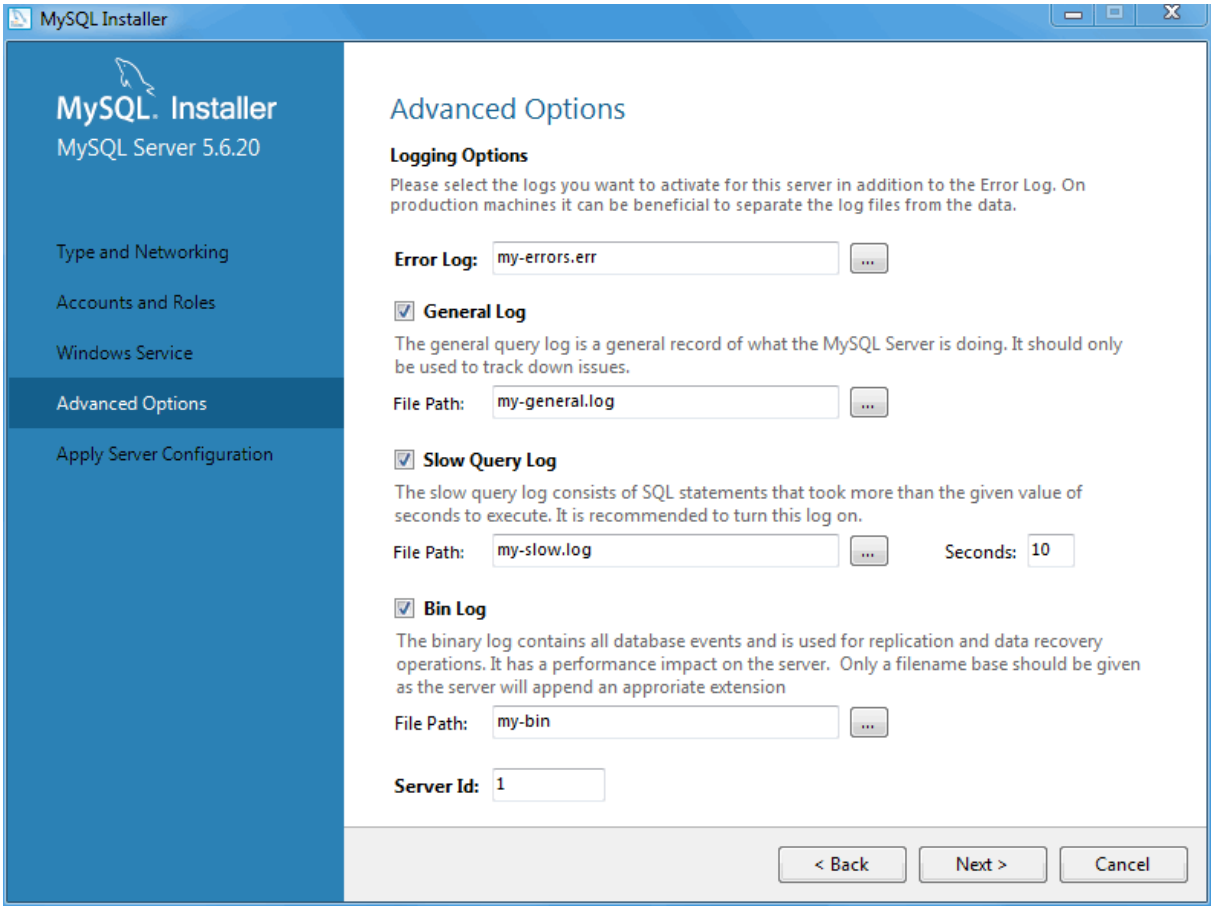


> **注記**
>
> 「Run Windows Services as ...」を「Custom User」を使用して構成する場合は、Microsoft Windows にサービスとしてログインする権限をカスタムユーザーが持っていなければなりません。また、このユーザーがこれらのユーザー権限で構成されるまで、「Next」ボタンは無効です。
>
> Microsoft Windows 7 では、これは,「スタート」メニュー、「コントロール パネル」、「管理ツール」、「ローカル セキュリティ ポリシー」、「ローカル ポリシー」、「ユーザー権利の割り当て」、「サービスとしてログオン」をロードすることで構成されます。ここで「ユーザーまたはグループの追加」を選択してカスタムユーザーを追加し、次に「OK」、「OK」を選択して保存します。

高度なオプション

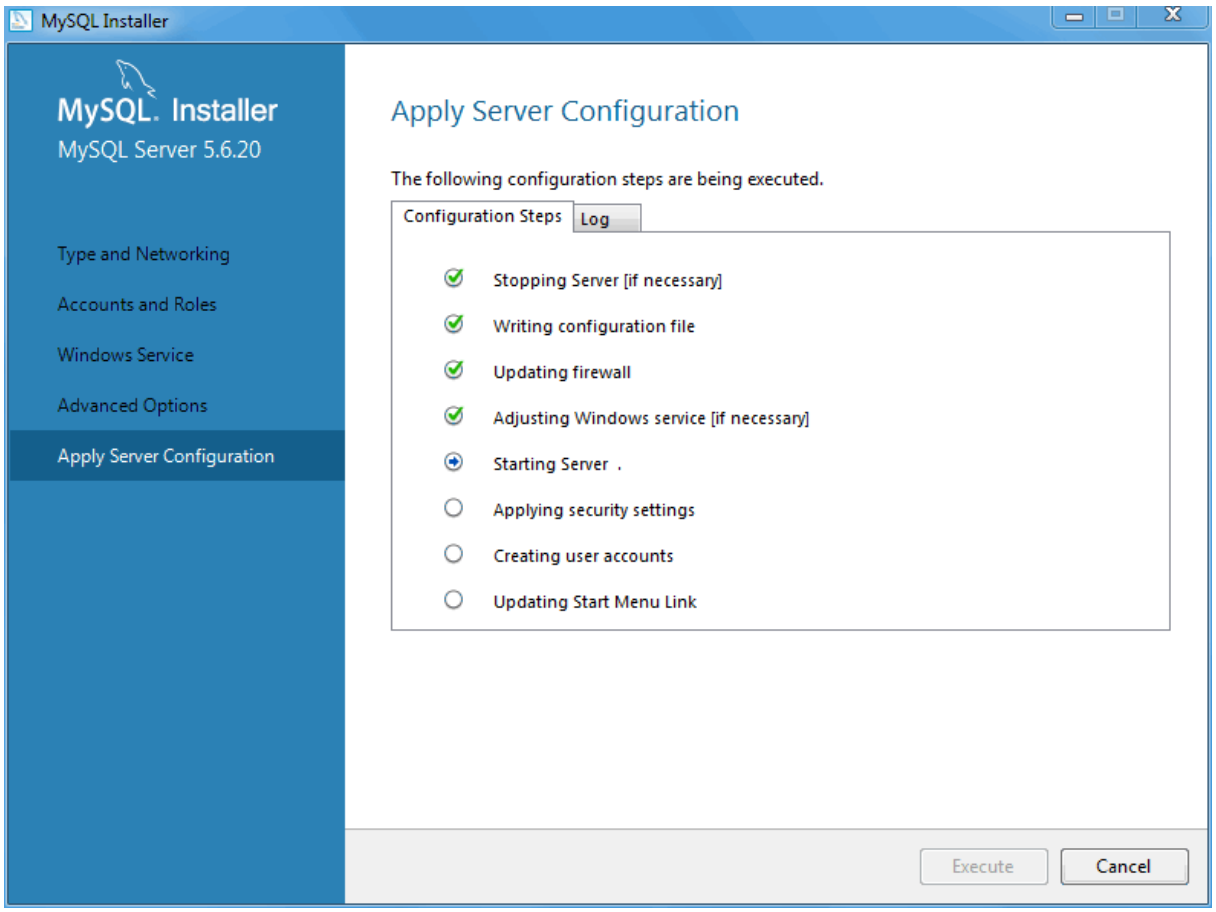
次の構成手順は、「Advanced Configuration」オプションをチェックした場合に使用可能になります。このセクションでは、MySQL のログファイルに関係するオプションを説明します。

図 2.15 MySQL Installer - MySQL Server 構成: ロギングオプション



すべてのリクエストした変更が適用される前に、「Next」をクリックして、最後のページに進みます。この「Apply Server Configuration」ページでは、実行される構成手順の詳細が示されます。

図 2.16 MySQL Installer - MySQL Server 構成: サーバー構成の適用

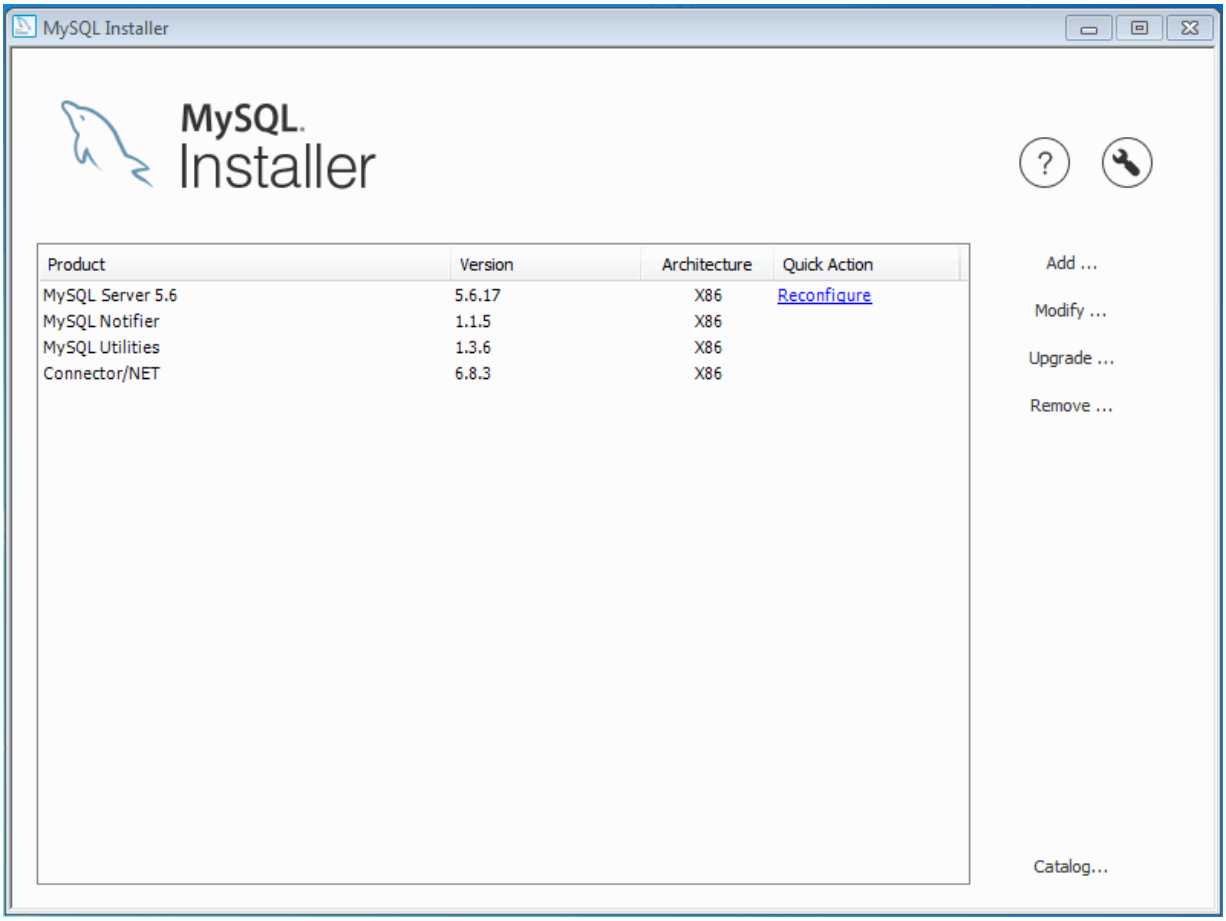


「Execute」をクリックして構成手順を実行します。各手順が成功するとアイコンが白から緑に変わり、失敗するとプロセスは停止します。「Log」タブをクリックしてログを表示します。

MySQL Installer 構成プロセスが終了すると、MySQL Installer は開始ページをリロードし、そこでインストールおよび構成に関連するその他のアクションを実行できます。

MySQL Installer が Microsoft Windows の「スタート」メニューの「MySQL」グループに追加されます。MySQL Installer をオープンするとダッシュボードがロードされ、インストール済みの MySQL 製品がリストされます。また、その他の MySQL Installer のアクションが使用可能です。

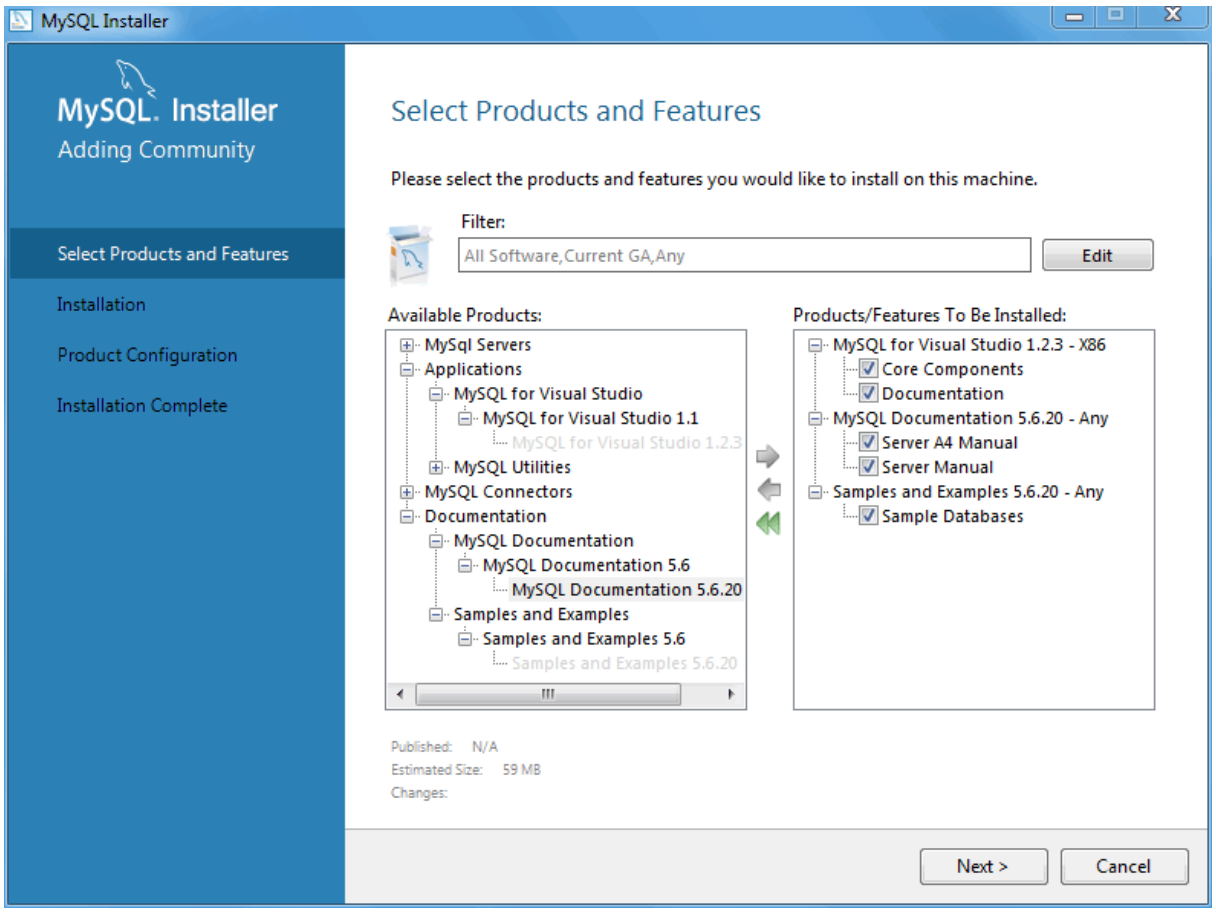
図 2.17 MySQL Installer - メインダッシュボード



MySQL 製品の追加

「Add」をクリックして新しい製品を追加します。これにより、「Select Products and Features」ページがロードされます。

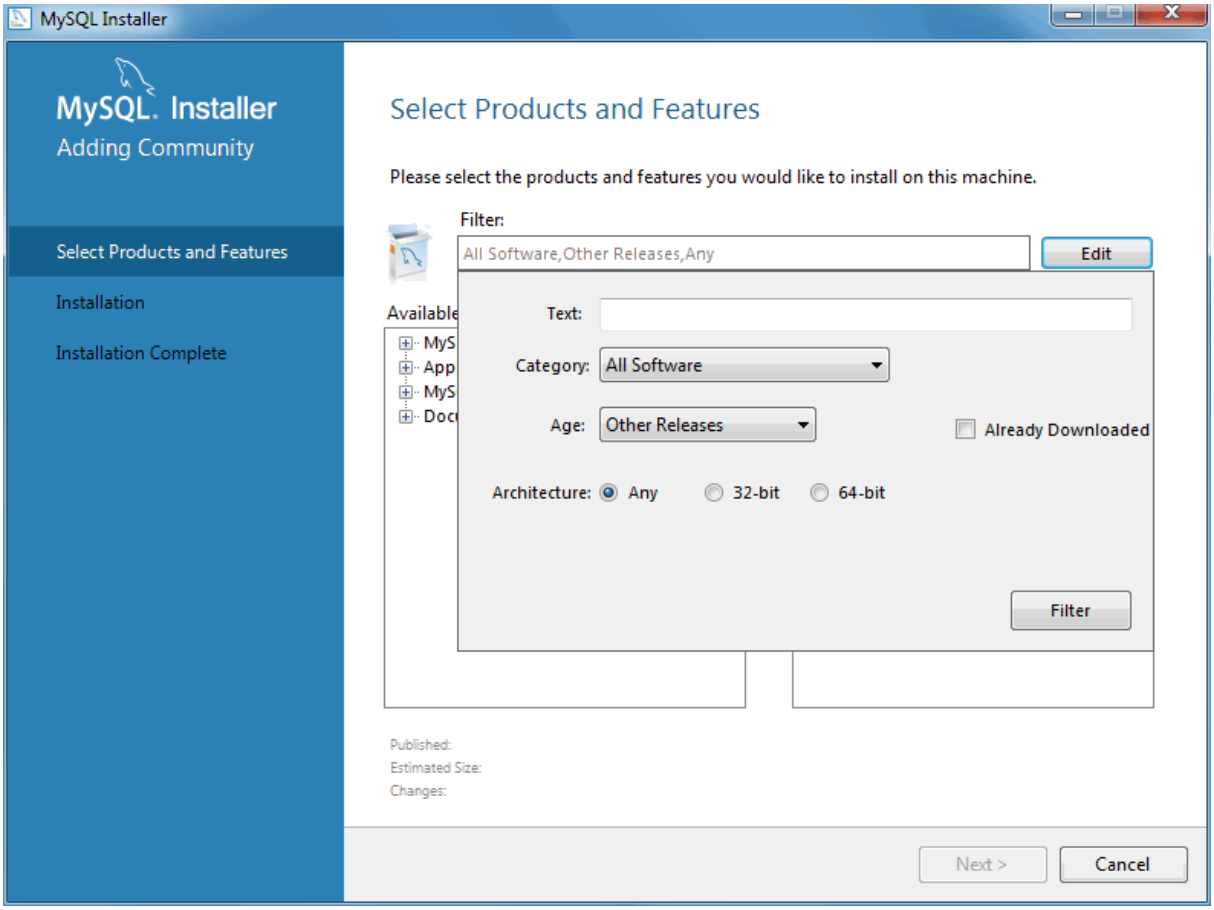
図 2.18 MySQL Installer - 製品と機能の選択



ここで、インストールする MySQL 製品を左の「Available Products」ペインから選択してから、緑色の右矢印をクリックしてインストール用のキューに入れます。

オプションで、「Edit」をクリックして製品と機能の検索フィルタをオープンします。

図 2.19 MySQL Installer - 製品と機能の選択フィルタ



たとえば、まだ GA ステータスに到達していないベータ製品などの、リリース前の製品を選択に含めることがあります。

> **注記**
>
> MySQL 製品のリリース前のバージョンをインストールする機能は、MySQL Installer 1.4.0 で加えられました。

インストールするすべての MySQL 製品を選択してから「Next」をクリックして続行し、次に「Execute」をクリックしてインストールプロセスを実行して、選択したすべての製品をインストールします。

MySQL 製品カタログ

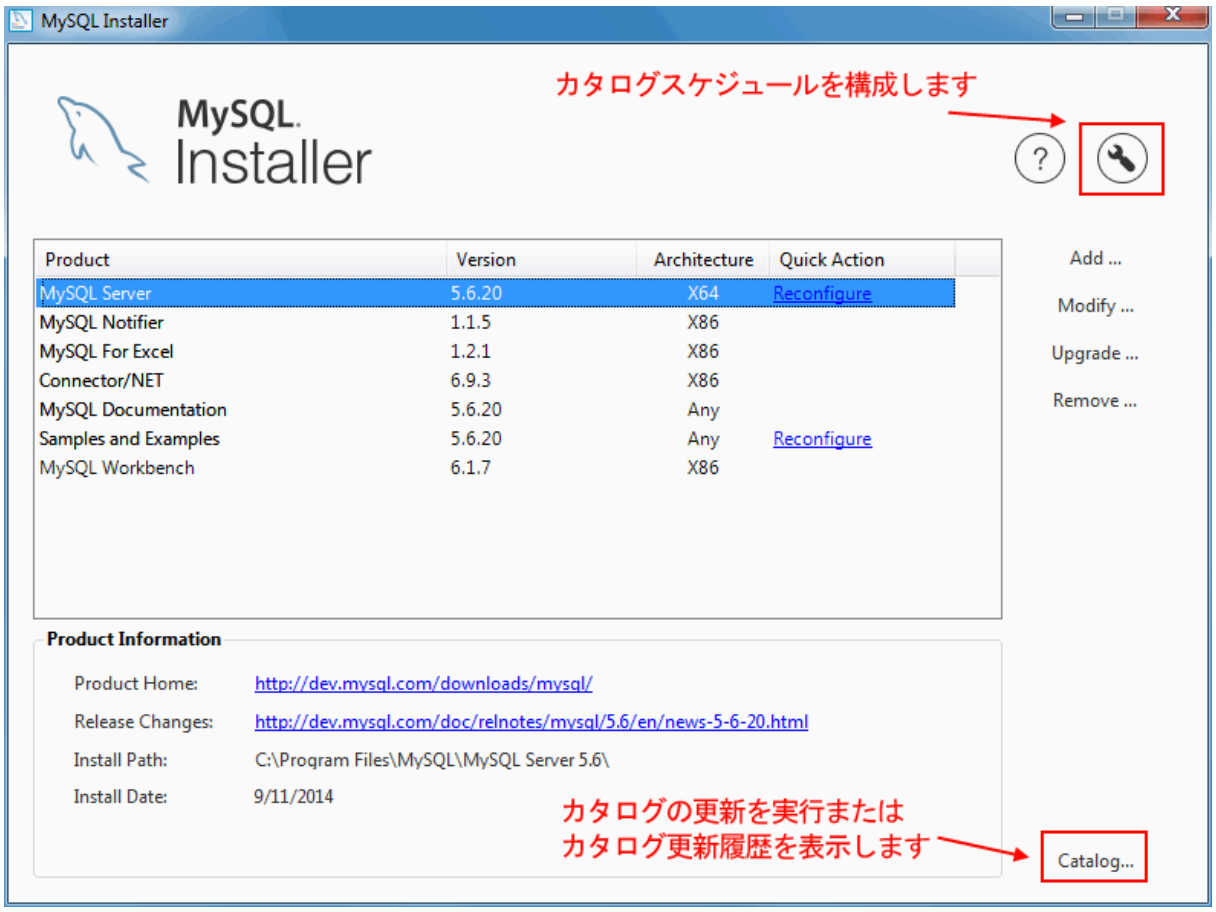
MySQL Installer は MySQL 製品カタログを保管しています。このカタログは手動または自動で更新でき、カタログの変更履歴も利用できます。

> **注記**
>
> MySQL 製品カタログは MySQL Installer 1.4.0 で加えられました。

手動更新

MySQL 製品カタログは、Installer のダッシュボードで「Catalog」をクリックすることでいつでも更新できます。

図 2.20 MySQL Installer - MySQL 製品カタログのオープン



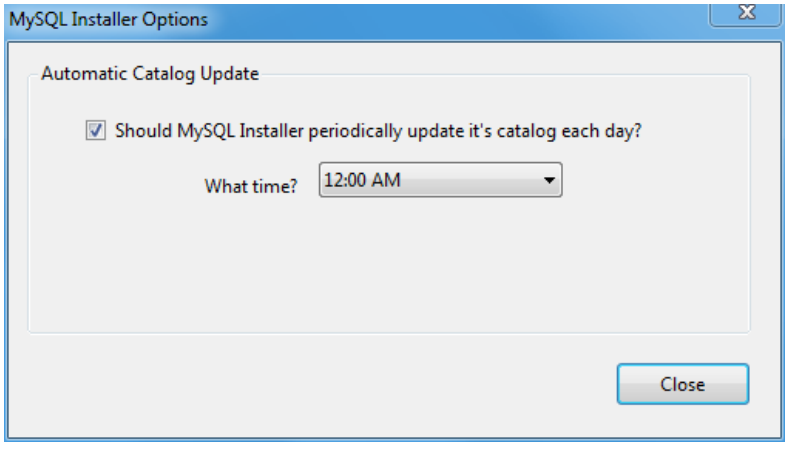
そこから、「Execute」をクリックして製品カタログを更新します。

自動更新

MySQL 製品カタログを 1 日 1 度自動的に更新するように MySQL Installer を構成できます。この機能を有効にして更新時間を設定するには、Installer ダッシュボードのレンチアイコンをクリックします。

次のウィンドウで、「Automatic Catalog Update」を構成します。この機能を有効または無効にして時間も設定します。

図 2.21 MySQL Installer - カタログスケジューラの構成

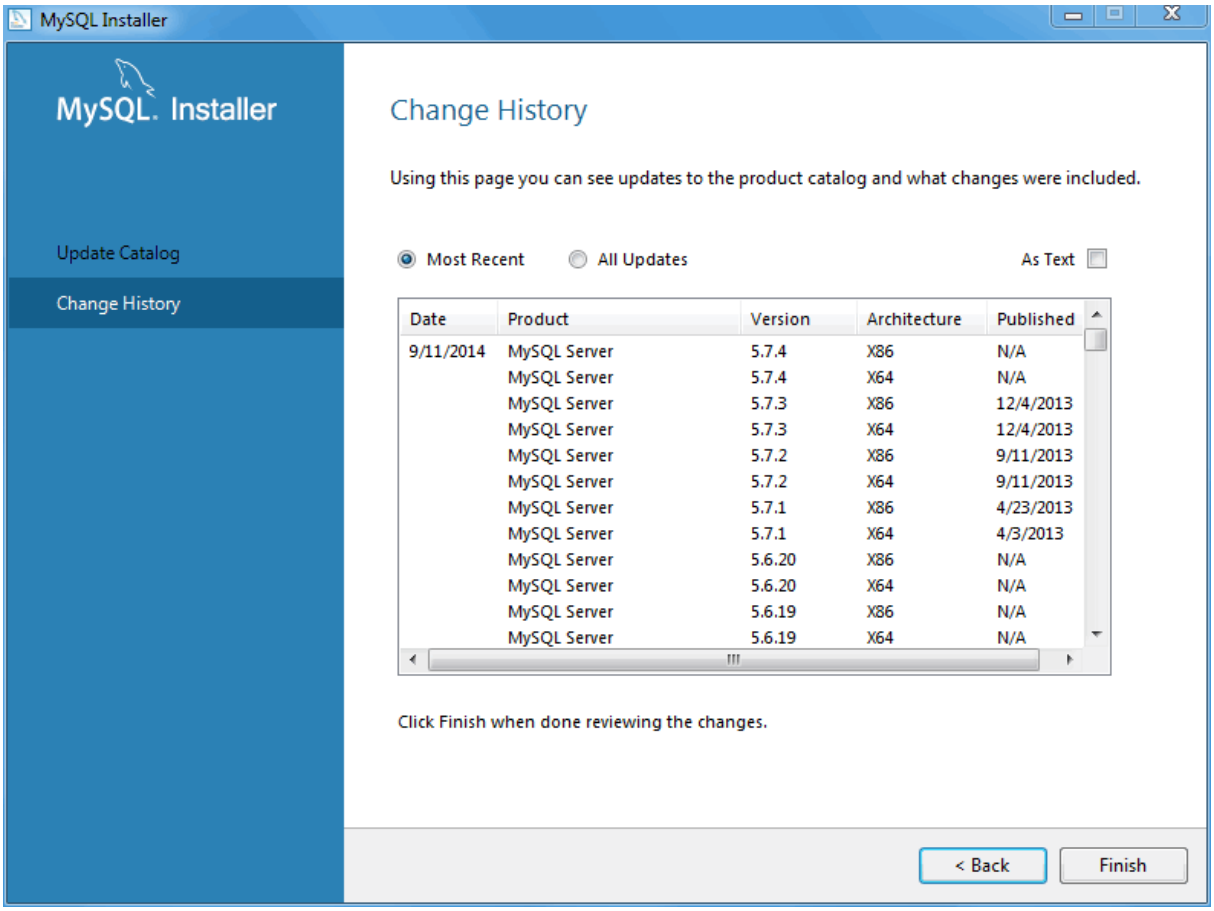


このオプションは Windows のタスクスケジューラを使用して「ManifestUpdate」という名前のタスクをスケジュールします。

変更履歴

MySQL Installer はすべての MySQL 製品の変更履歴を追跡しています。ダッシュボードから「Catalog」をクリックし、オプションでカタログを更新し (または「Do not update at this time」チェックボックスを切り替え)、「Next」/「Execute」をクリックしてから変更履歴を表示します。

図 2.22 MySQL Installer - カタログ変更履歴



MySQL 製品の削除

MySQL Installer では、MySQL 製品をシステムから削除することもできます。MySQL 製品を削除するには、Installer のダッシュボードから「Remove」をクリックします。これにより、ウィンドウが開いてインストール済みの MySQL 製品が表示されます。削除する (アンインストールする) MySQL 製品を選択してから、「Execute」をクリックすると削除プロセスが開始します。

> **注記**
>
> すべての MySQL 製品を選択するには、「Product」ラベルの左の [ ] チェックボックスをクリックします。

図 2.23 MySQL Installer - 製品の削除: 選択

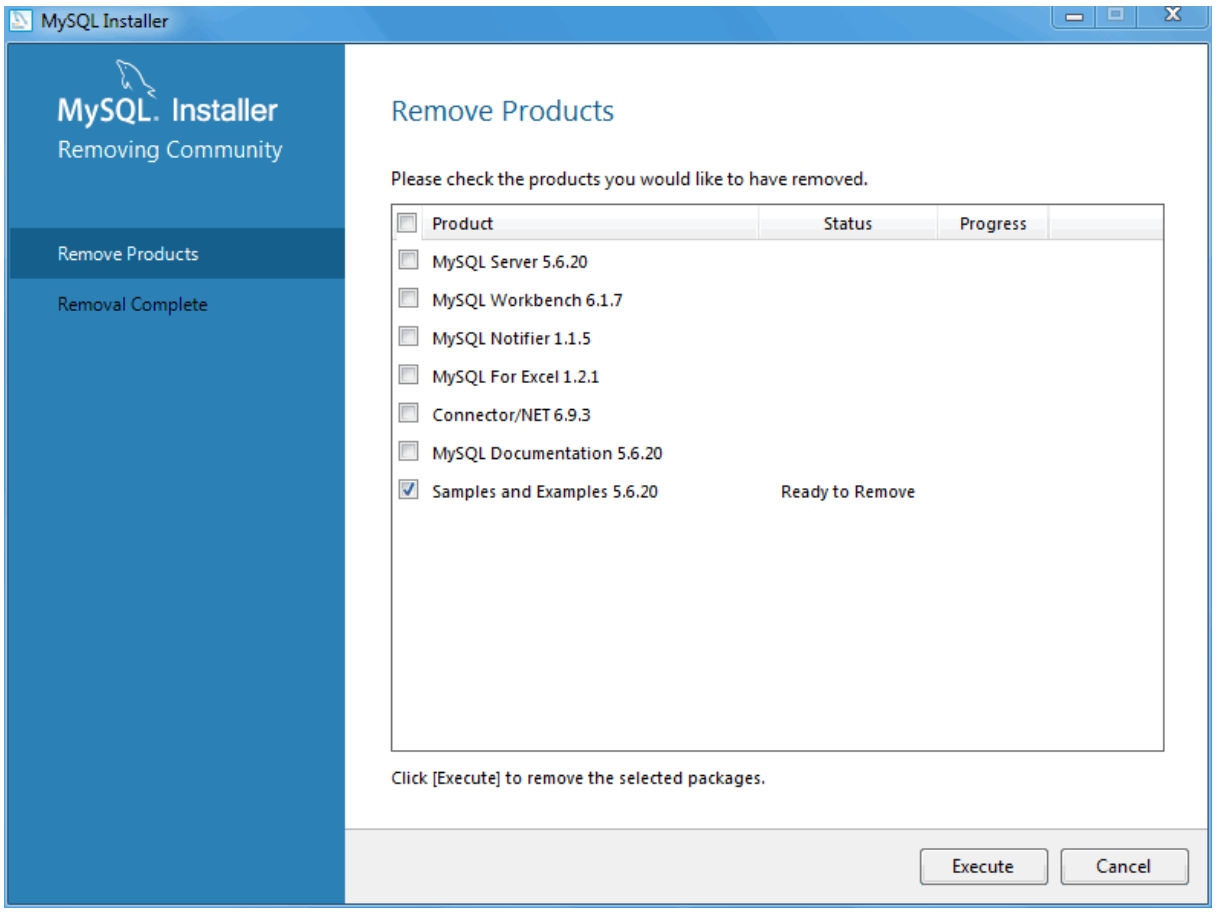
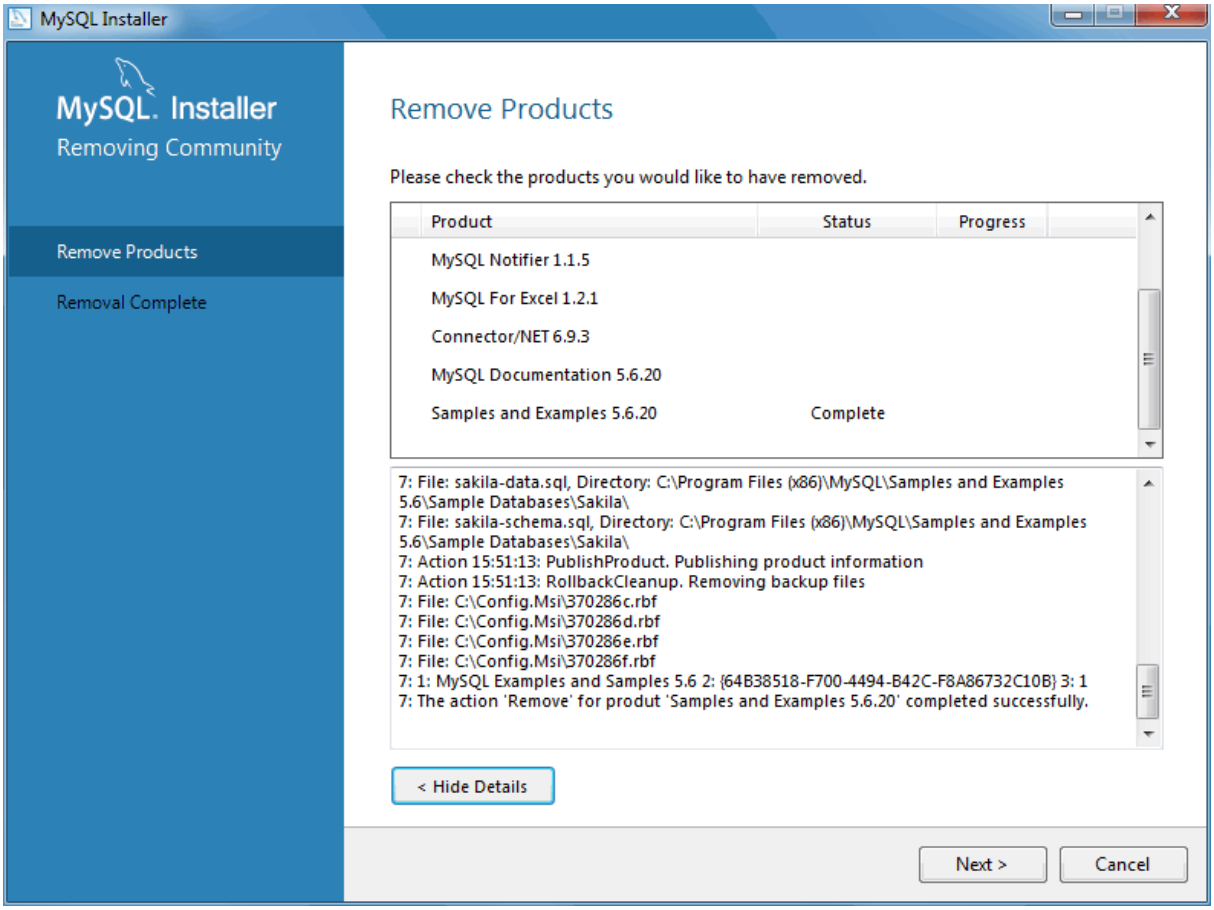


図 2.24 MySQL Installer - 製品の削除: 実行



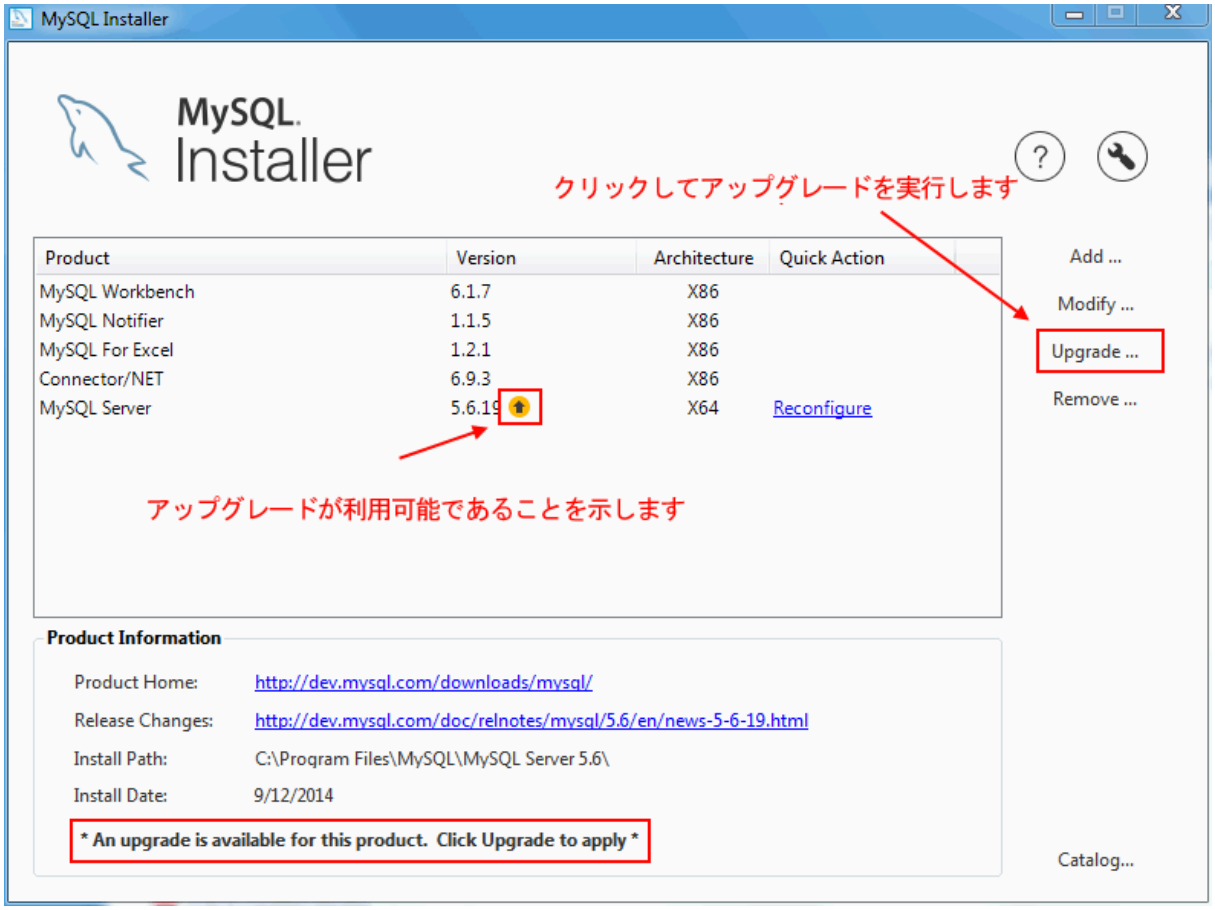
MySQL 製品の変更

MySQL Installer では、MySQL 製品のインストールを変更するいくつかのオプションがあります。

アップグレード

アップグレードが利用可能な MySQL 製品は、メインダッシュボード上で強調表示されます。アップグレードが利用可能な製品には、バージョン番号の隣にアップグレードアイコンがあります。

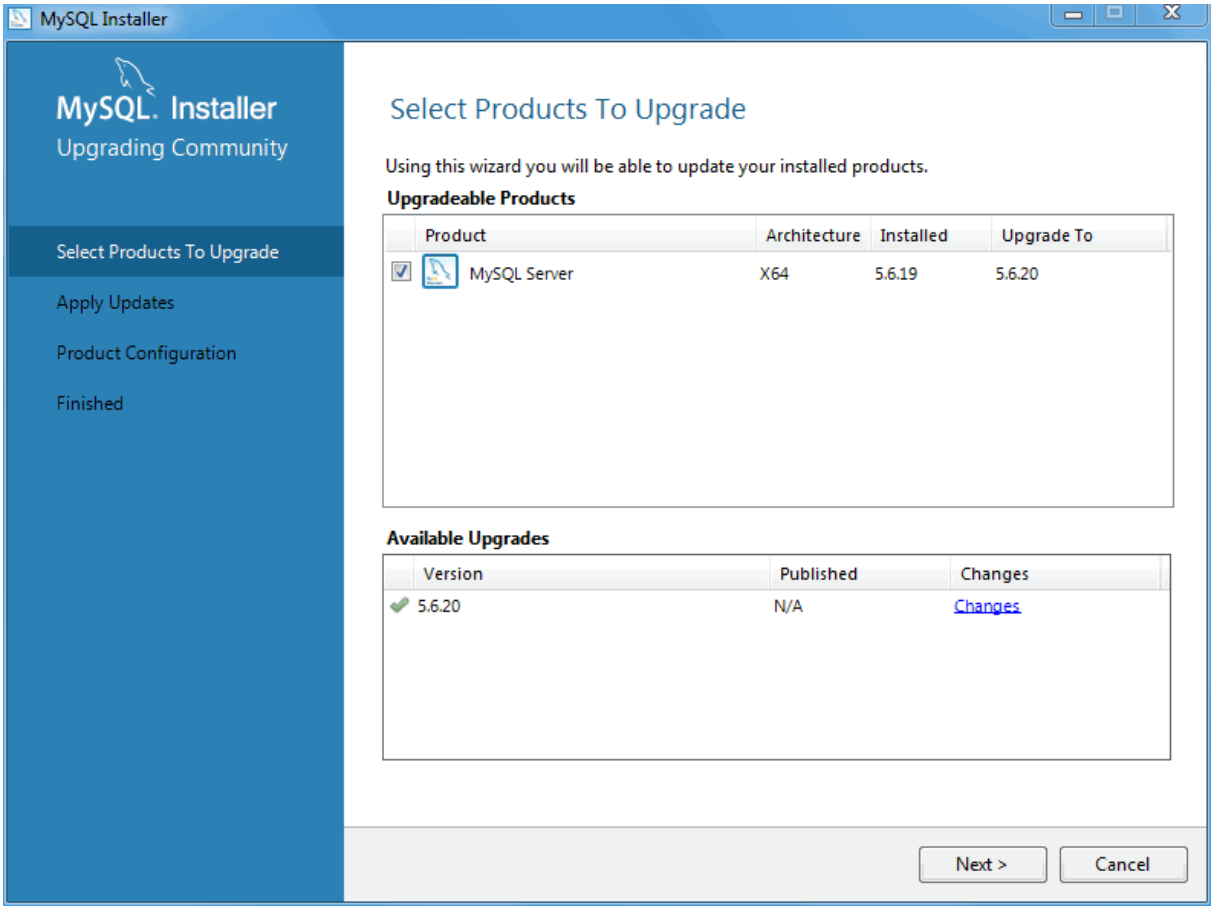
図 2.25 MySQL Installer - MySQL 製品のアップグレード



> **注記**
>
> 利用可能なアップグレードは、最新のカタログがあることで判断されます。MySQL 製品カタログを最新の状態に維持する方法については、MySQL 製品カタログを参照してください。

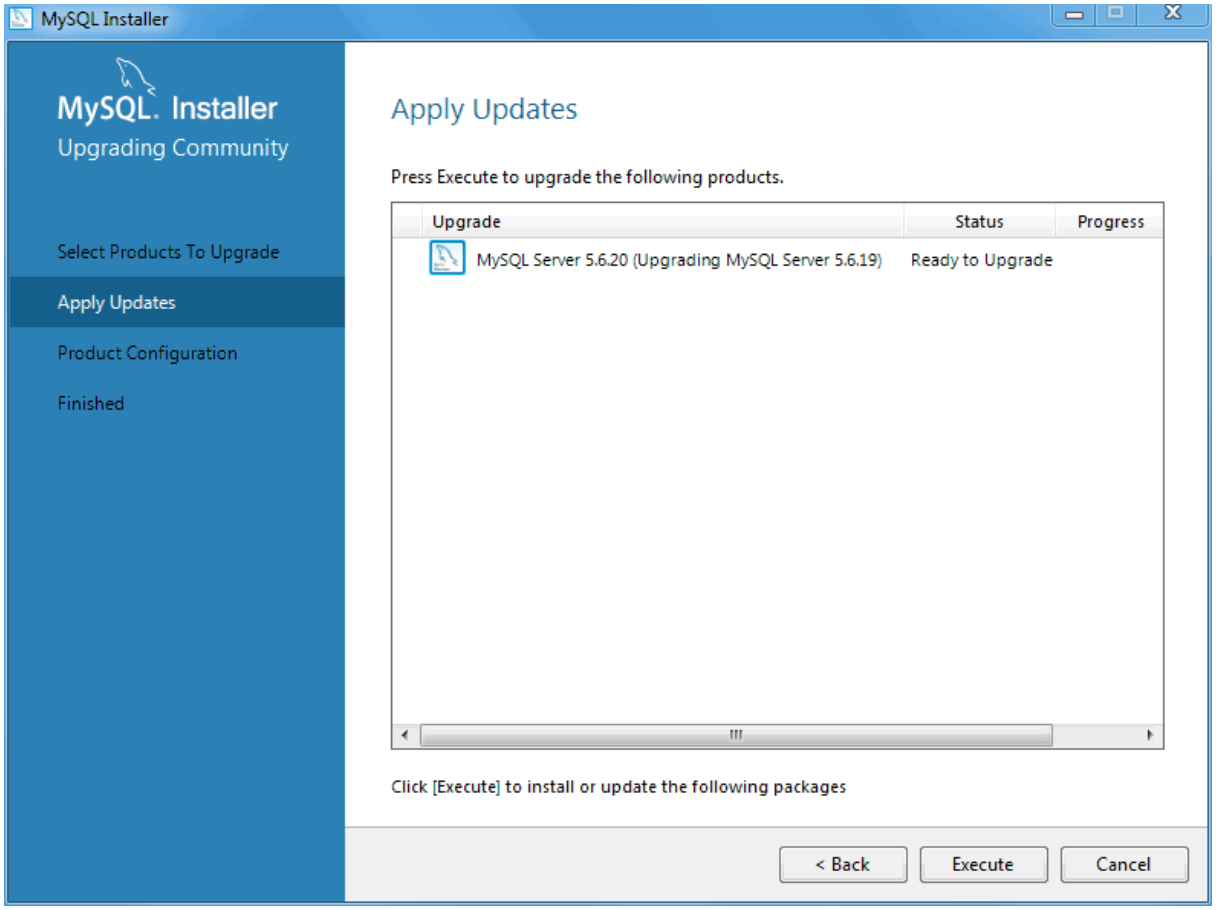
「Upgrade」をクリックしてアップグレード可能な製品のリストを表示します。この例では、MySQL Server 5.6.19 をバージョン 5.6.20 にアップグレードできることが示されています。

図 2.26 MySQL Installer - アップグレードする製品の選択



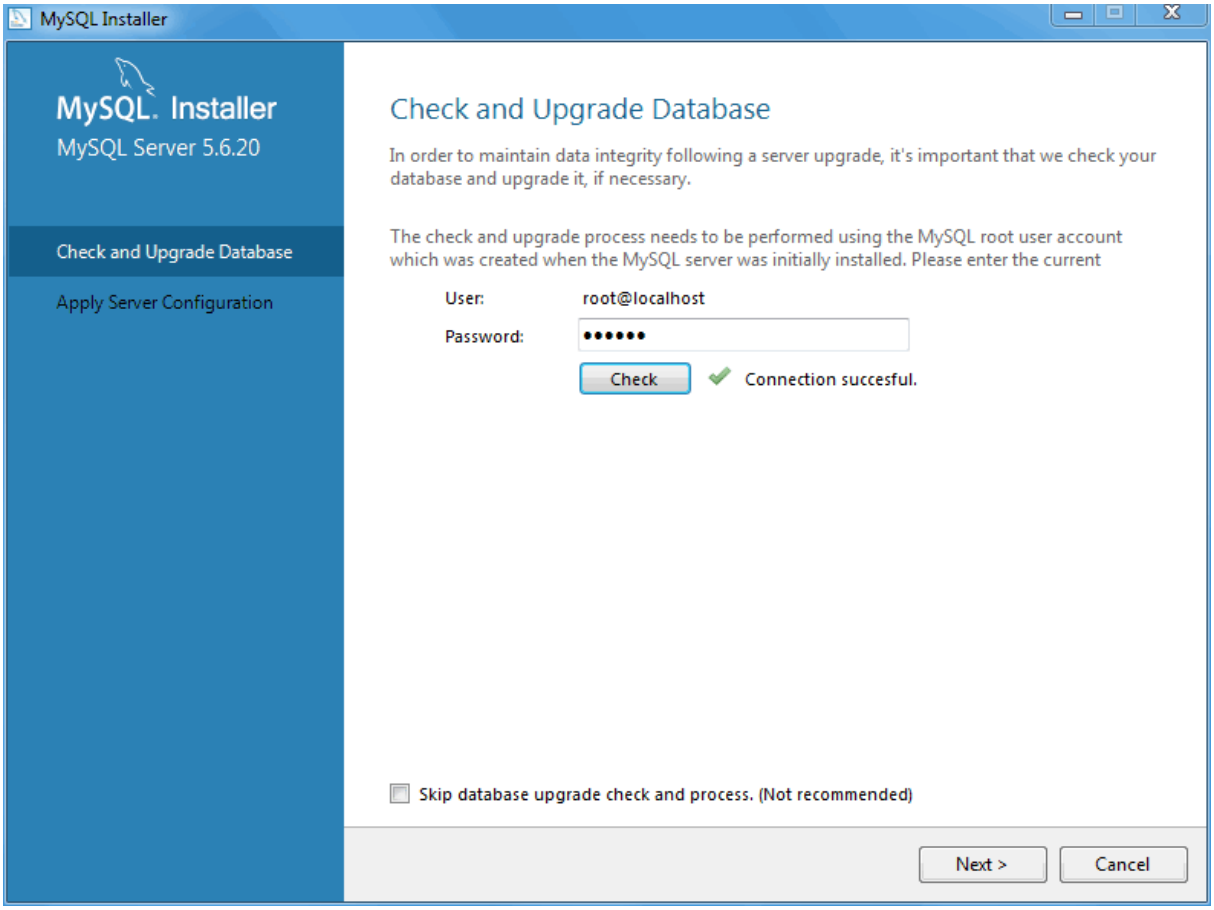
アップグレードする製品を選択 (チェック) し、オプションで「changes」リンクをクリックして製品のリリースノートをブラウザに表示します。「Next」をクリックしてアップグレードプロセスを開始します。

図 2.27 MySQL Installer - 更新の適用



MySQL Server アップグレードでは、サーバーのデータベースもチェックしてアップグレードします。この手順はオプションですがお勧めしています。

図 2.28 MySQL Installer - データベースのチェックとアップグレード



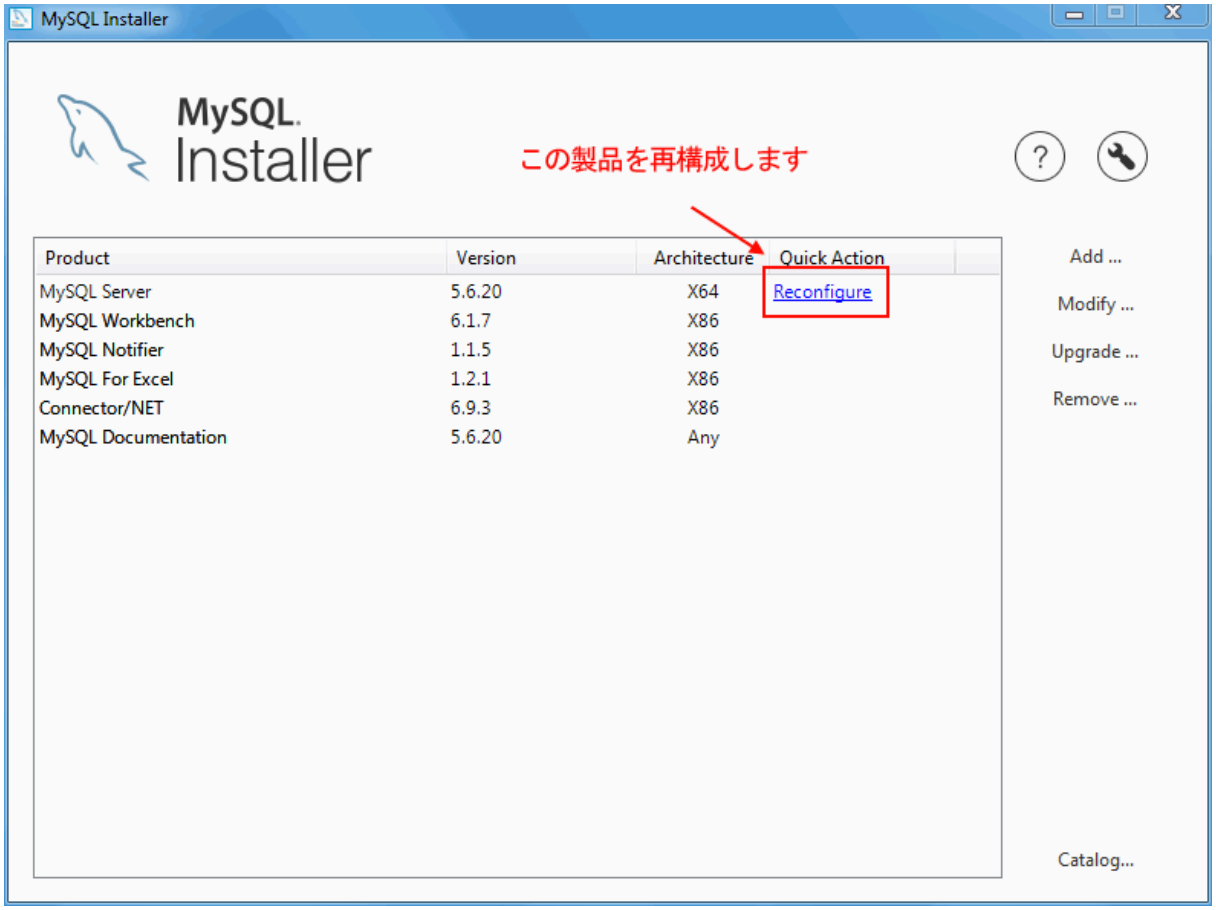
完了すると、製品がアップグレードされ使用できるようになります。MySQL Server アップグレードが、MySQL Server の再起動も行います。

再構成

MySQL Server などの一部の MySQL 製品には、「Reconfigure」オプションがあります。その MySQL 製品がインストールされたときに設定した構成オプションと同じものが開き、現在の値が事前に移入されます。

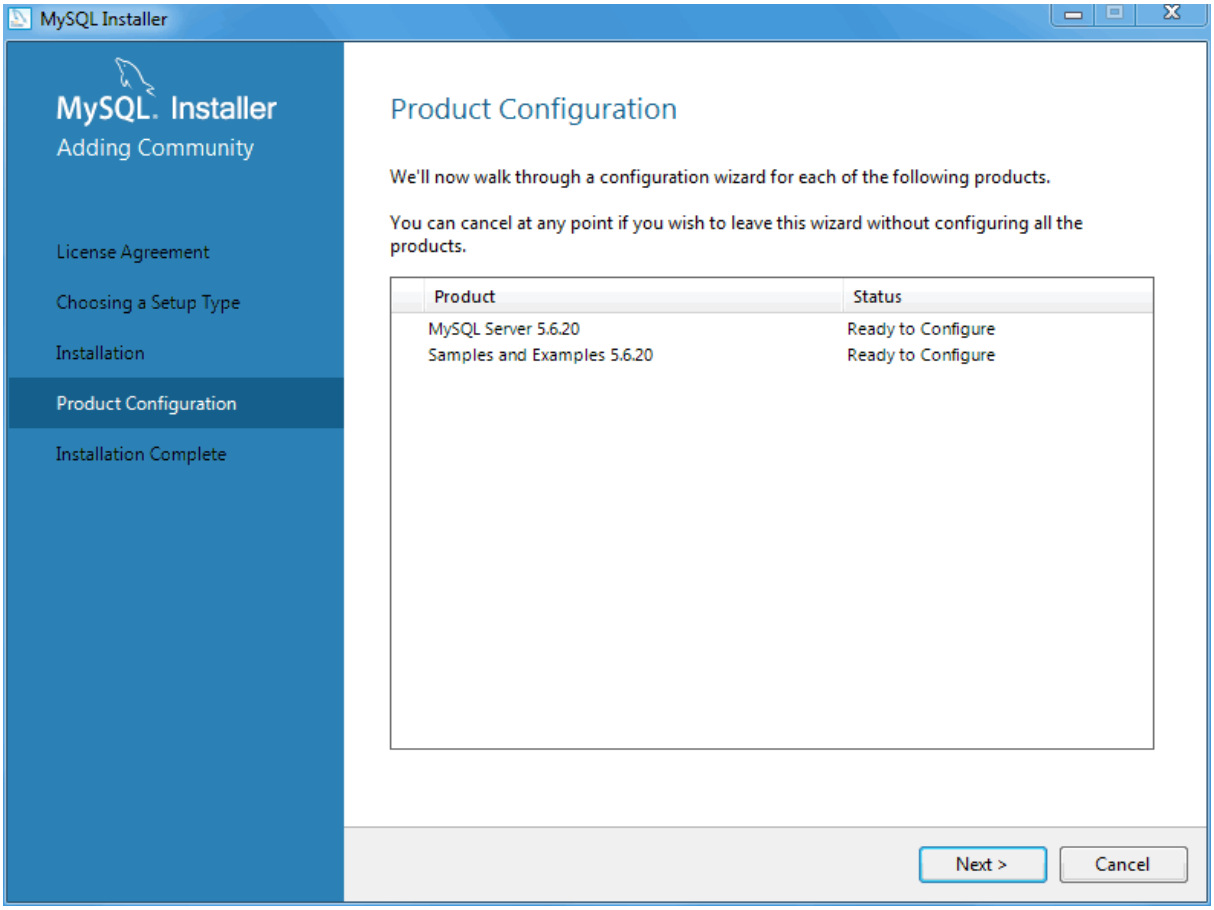
実行するには、メインダッシュボードで、再構成する MySQL 製品の「Quick Action」カラムの「Reconfigure」リンクをクリックします。

図 2.29 MySQL Installer - MySQL 製品の再構成



MySQL Server の場合は、これにより見慣れた構成ウィザードが開きます。

図 2.30 MySQL Installer - 再構成ウィザード

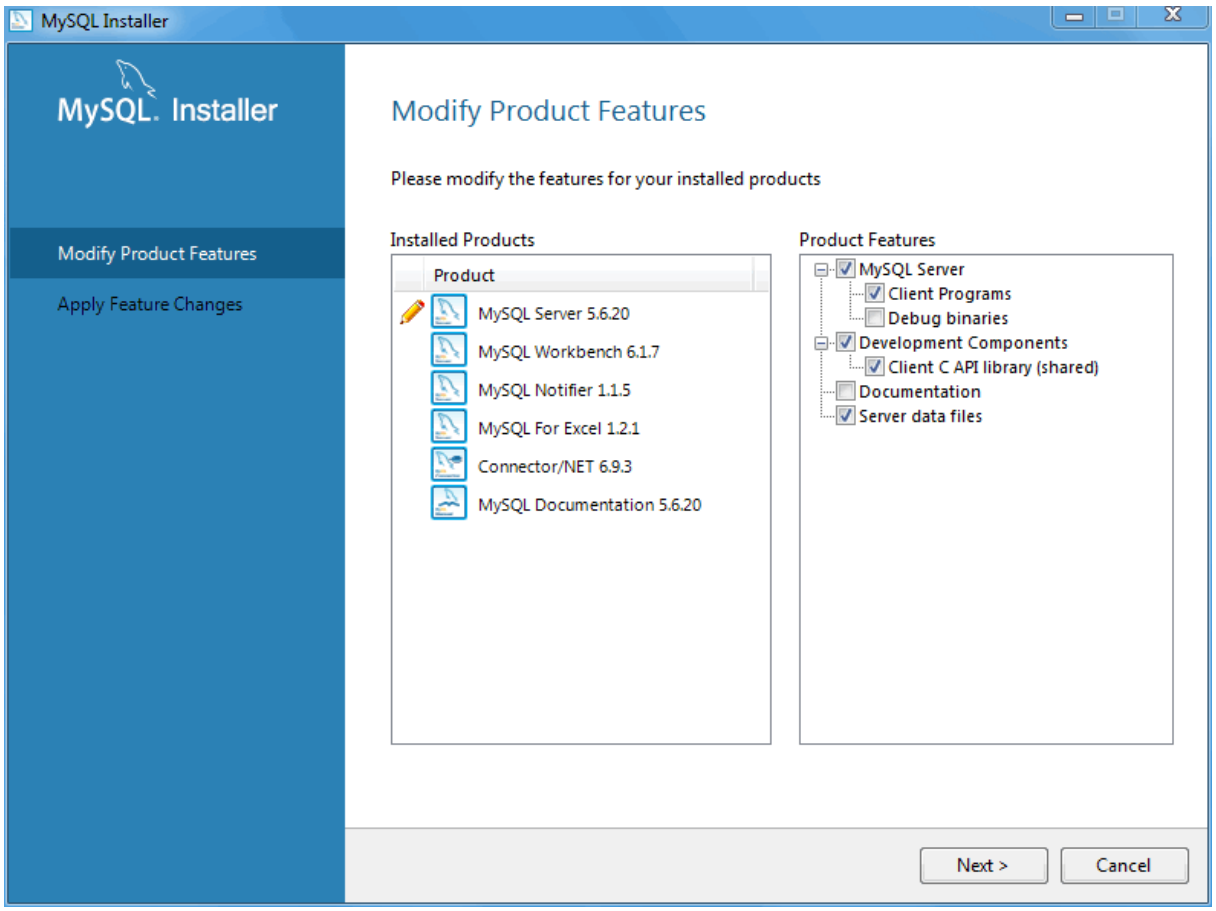


変更

多くの MySQL 製品には、追加または削除が可能な機能コンポーネントが含まれます。たとえば、Debug binaries および Client Programs は MySQL Server のサブコンポーネントです。

製品の機能を変更するには、メインダッシュボードで「Modify」をクリックします。

図 2.31 MySQL Installer - 製品の機能の変更



「Execute」をクリックして変更リクエストを実行します。

2.3.3.2 MySQL Installer Console

MySQLInstallerConsole は、GUI バージョンの MySQL Installer と同様の機能を提供しますが、コマンド行から実行します。これは、MySQL Installer をはじめて実行した際にインストールされ、MySQL Installer ディレクトリ内で利用可能になります。通常これは C:\Program Files (x86)\MySQL\MySQL Installer\ で、Console は管理者権限で実行しなければなりません。

使用するには、「スタート」、「アクセサリ」を選択してから「コマンド プロンプト」を右クリックし、「管理者として実行」を選択して、管理者権限でコマンドプロンプトを起動します。そして、オプションでコマンド行からディレクトリを MySQLInstallerConsole のある場所に変更します。

```
C:\> cd "C:\Program Files (x86)\MySQL\MySQL Installer"
C:\> MySQLInstallerConsole.exe help

C:\Program Files (x86)\MySQL\MySQL Installer for Windows>MySQLInstallerConsole.exe help

The following commands are available:

Configure - Configures one or more of your installed programs.
Help      - Provides list of available commands.
Install   - Install and configure one or more available MySQL programs.
List      - Provides an interactive way to list all products available.
Modify    - Modifies the features of installed products.
Remove    - Removes one or more products from your system.
Status    - Shows the status of all installed products.
Update    - Update the current product catalog.
Upgrade   - Upgrades one or more of your installed programs.
```

MySQLInstallerConsole は次のオプションをサポートします。これらはコマンド行で指定します。

- configure [product1]:[setting]=[value]; [product2]:[setting]=[value]; [...]

  システム上の 1 つまたは複数の MySQL 製品を構成します。

次のようなスイッチがあります。

- -showsettings: 製品名を -showsettings の後ろに渡すことによって、選択された製品の使用可能なオプションを表示します。
- -silent: 確認プロンプトを無効にします。

```
C:\> MySQLInstallerConsole configure -showsettings server
C:\> MySQLInstallerConsole configure server:port=3307
```

- help [command]

ヘルプメッセージを使用例とともに表示して終了します。コマンドに固有のヘルプを受信するには、そのコマンドを渡します。

```
C:\> MySQLInstallerConsole help
C:\> MySQLInstallerConsole help install
```

- install [product]:[features]:[config block]:[config block]:[config block]; [...]

システムに 1 つまたは複数の MySQL 製品をインストールします。

次のようなスイッチと構文オプションがあります。

- -type=[SetupType]: 事前定義されたソフトウェアのセットをインストールします。「SetupType」は次のいずれかです。

  > **注記**
  >
  > Custom 以外のセットアップタイプは、ほかに MySQL 製品がインストールされていない場合のみ選択できます。

  - Developer: 完全な開発環境をインストールします。
  - Server: 単一の MySQL Server をインストールします。
  - Client: クライアントプログラムとライブラリをインストールします。
  - Full: すべてをインストールします。
  - Custom: ユーザーが選択した製品をインストールします。これはデフォルトのオプションです。
- -showsettings: 製品名を -showsettings の後ろに渡すことによって、選択された製品の使用可能なオプションを表示します。
- -silent: 確認プロンプトを無効にします。
- [config block]: 1 つまたは複数の構成ブロックを指定できます。各構成ブロックは、セミコロンで区切られたキーと値のペアのリストです。ブロックには、「config」または「user」タイプのいずれかのキーを含むことができ、指定されない場合は「config」がデフォルトタイプです。

  「config」タイプのブロックは製品あたり 1 つのみ定義できます。「user」ブロックは、製品のインストール中に作成された各ユーザーに対して定義してください。

  > **注記**
  >
  > 製品の再構成中のユーザーの追加はサポートされていません。
- [feature]: 機能ブロックは、セミコロンで区切られた機能のリストまたは「*」 (全機能を選択) です。

```
C:\> MySQLInstallerConsole install server;5.6.22:*:port=3307;serverid=2:type=user;username=foo;password=bar;role=DBManager
C:\> MySQLInstallerConsole install server;5.6.22;x64 -silent
```

- list

利用可能なすべての MySQL 製品を検索できるインタラクティブなコンソールをリストします。MySQLInstallerConsole list を実行してコンソールを起動し、検索する部分文字列を入力します。

```
C:\> MySQLInstallerConsole list
```

- modify [product1:-removelist|+addlist] [product2:-removelist|+addlist] [...]

  以前にインストールされた MySQL 製品の機能を変更または表示します。

  - -silent: 確認プロンプトを無効にします。

```
C:\> MySQLInstallerConsole modify server
C:\> MySQLInstallerConsole modify server:+documentation
C:\> MySQLInstallerConsole modify server:-debug
```

- remove [product1] [product2] [...]

  システム上から 1 つまたは複数の製品を削除します。

  - *: すべての MySQL 製品を削除するには * を渡します。
  - -continue: エラーが発生しても操作を継続します。
  - -silent: 確認プロンプトを無効にします。

```
C:\> MySQLInstallerConsole remove *
C:\> MySQLInstallerConsole remove server
```

- status

  システムにインストールされている MySQL 製品の簡単な概要を提供します。情報には、製品名とバージョン、アーキテクチャー、インストール日、およびインストールの場所が含まれます。

```
C:\> MySQLInstallerConsole status
```

- upgrade [product1:version] [product2:version], [...]

  システム上の 1 つまたは複数の製品をアップグレードします。次のような構文オプションがあります。

  - * : すべての製品を最新のバージョンにアップグレードするには * を渡します。あるいは具体的な製品を渡します。
  - ! : MySQL 製品を最新バージョンにアップグレードするには、バージョン番号として ! を渡します。
  - -silent: 確認プロンプトを無効にします。

```
C:\> MySQLInstallerConsole upgrade *
C:\> MySQLInstallerConsole upgrade workbench:6.2.2
C:\> MySQLInstallerConsole upgrade workbench:!
C:\> MySQLInstallerConsole upgrade workbench:6.2.2 excel:1.3.2
```

- update

  最新の MySQL 製品カタログをシステムにダウンロードします。成功すると、次に MySQLInstaller または MySQLInstallerConsole が実行されたときに、ダウンロードカタログが適用されます。

```
C:\> MySQLInstallerConsole update
```

> **注記**
>
> 「Automatic Catalog Update」 GUI オプションはこのコマンドを Windows のタスクスケジューラから実行します。

2.3.4 MySQL Notifier

MySQL Notifier は、システムトレイにあるインジケータを介して、ローカルおよびリモートの MySQL Server インスタンスのステータスをモニターおよび調整できるツールです。MySQL Notifier では、コンテキストメニューからいくつかの MySQL GUI ツール (MySQL Workbench など) にすばやくアクセスすることもできます。

MySQL Notifier は MySQL Installer によってインストールされ、(デフォルトでは) Microsoft Windows の起動時に起動します。

> **注記**
>
> インストールするには、MySQL Installer をダウンロードして実行し、MySQL Notifier 製品が選択されていることを確認してからインストールに進みます。詳細は、MySQL Installer マニュアル を参照してください。
>
> MySQL Notifier の各リリースの変更内容の詳細は、MySQL Notifier リリースノート を参照してください。
>
> MySQL Notifier の追加のヘルプとサポートは、MySQL Notifier フォーラム にアクセスしてください。

次のような機能があります。

- MySQL Server インスタンスの起動、停止、および再起動を行います。
- 新しい MySQL Server サービスを自動的に検出 (して追加) します。これらは「Manage Monitored Items」の下にリストされ、構成することもできます。
- ステータスに応じてトレイアイコンが変化します。モニターされているすべての MySQL Server インスタンスが実行中であれば緑、少なくとも 1 つのサービスが停止している場合は赤です。この動作は、「Update MySQL Notifier tray icon based on service status」オプションで決まり、デフォルトでは各サービスに対して有効です。
- MySQL Workbench、MySQL Installer、および MySQL Utilities などのその他のアプリケーションにリンクします。たとえば、「Configure Instance」を選択すると、そのインスタンスの MySQL Workbench の「Server Administration」ウィンドウがロードされます。
- MySQL Workbench もインストールされている場合は、「Configure Instance」 および 「SQL Editor」オプションがローカルの MySQL インスタンスに対して利用可能です (リモートのインスタンスには利用できません)。
- ローカルおよびリモートの MySQL インスタンスのモニタリング。

> **注記**
>
> リモートモニタリングは MySQL Notifier 1.1.0 以降利用可能です。

MySQL Notifier はシステムトレイにあり、MySQL Server インスタンスのステータスの視覚的情報を提供します。現在の MySQL Server が実行中の場合は緑のアイコンが、サービスが停止している場合は赤のアイコンが、トレイアイコンの左上に表示されます。

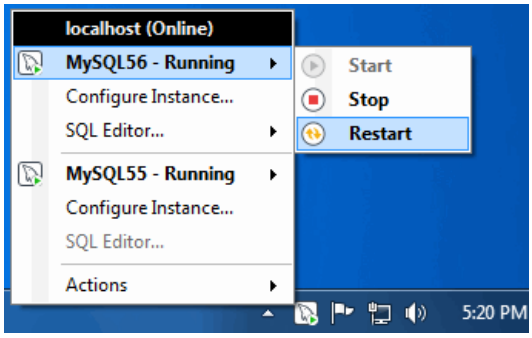
MySQL Notifier は、ローカルマシンで検出された MySQL サービスを自動的に追加し、各サービスは保存されて構成可能です。デフォルトでは、「Automatically add new services whose name contains」オプションは有効で mysql に設定されています。関連する「Notifications Options」には、新しいサービスが発見された場合またはステータスが変化した場合に通知を受ける、などがあり、これらもデフォルトで有効です。また、サービスをアンインストールすると、そのサービスは MySQL Notifier からも削除されます。

> **注記**
>
> 「Automatically add new services whose name contains」オプションのデフォルトは、Notifier 1.1.0 では「.*mysqld.*」から「mysql」に変更されました。

システムトレイアイコンをクリックすると、次のスクリーンショットに示すようにいくつかのオプションが表示されます。

「Service Instance」メニューは MySQL Notifier のメインウィンドウで、MySQL Server の停止、起動、および再起動が可能です。

図 2.32 MySQL Notifier Service Instance メニュー

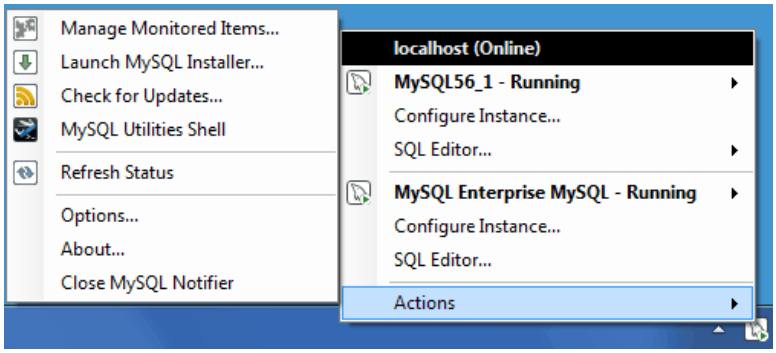


「Actions」メニューには、外部アプリケーションへのいくつかのリンク (インストールされている場合) と、モニターされている (ローカルおよびリモートのコンピュータの) すべてのサービスと MySQL のステータスを手動でリフレッシュする「Refresh Status」オプションが含まれます。

> **注記**
>
> MySQL Notifier によってモニターされているサービスがない場合は、メインメニューには「Actions」メニューが表示されます。

> **注記**
>
> 「Refresh Status」機能は MySQL Notifier 1.1.0 以降で利用可能です。

図 2.33 MySQL Notifier Actions メニュー

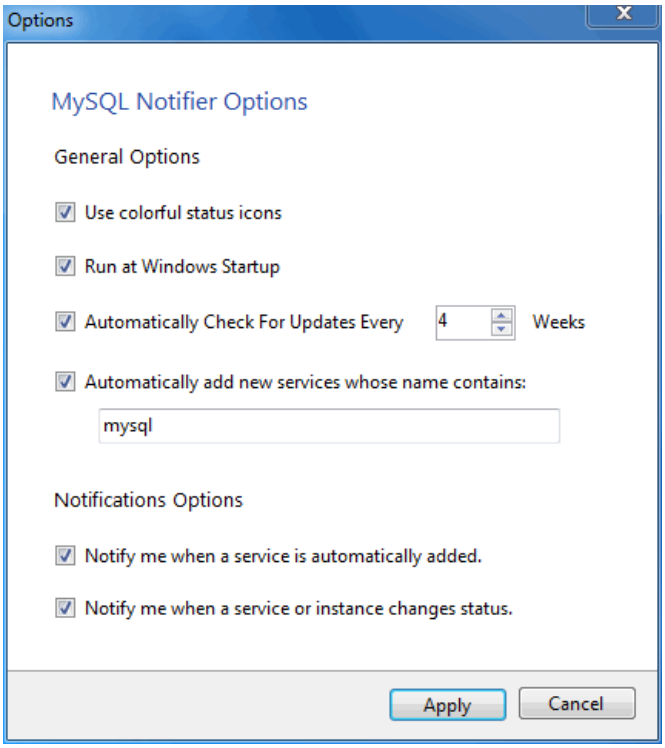


「Actions」、「Options」メニューは MySQL Notifier を構成し、次のようなオプションがあります。

- Use colorful status icons: MySQL Notifier のトレイのカラフルなスタイルを有効にします。
- Run at Windows Startup: Microsoft Windows の起動時にアプリケーションがロードできるようにします。
- Automatically Check For Updates Every # Weeks: MySQL Notifier の新しいバージョンをチェックします。このチェックは指定した週数ごとに実行します。
- Automatically add new services whose name contains: サービスをフィルタするのに使用され、MySQL Notifier を実行しているモニター対象のローカルコンピュータのリスト、およびすでに Windows サービスをモニタリングしているリモートコンピュータに自動的にサービスを追加するためのテキスト。モニターされるサービス、および「Add New Service」ダイアログの Microsoft Windows サービスのリストもフィルタします。

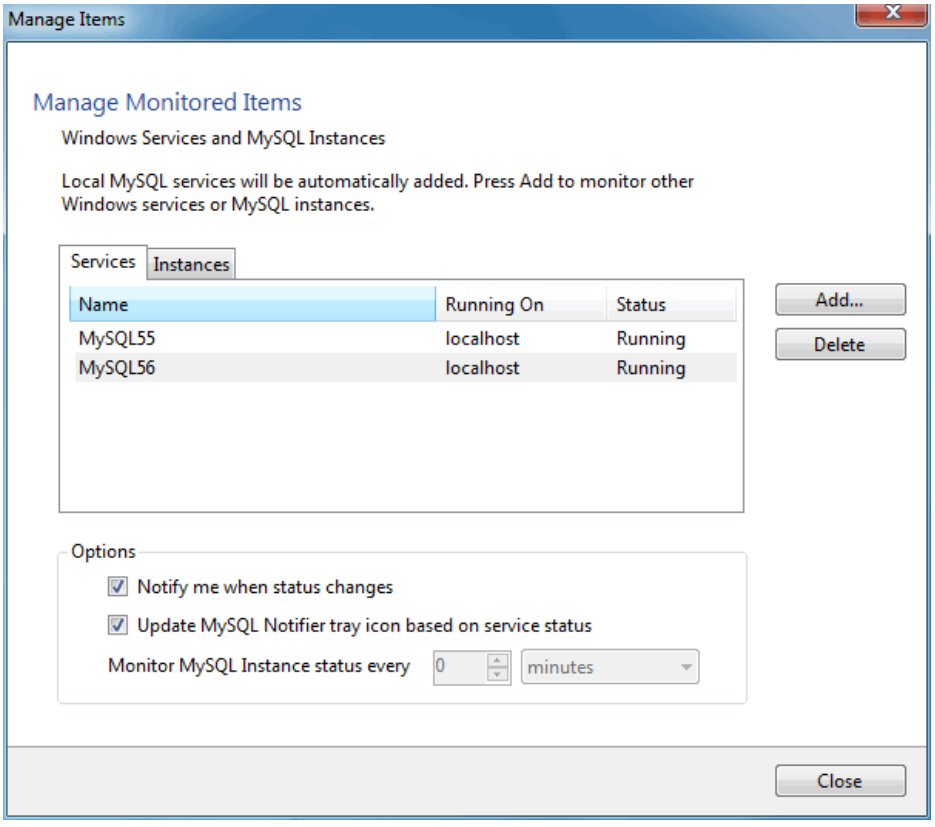
  バージョン 1.1.0 より前では、このオプションは「Automatically add new services that match this pattern」という名前でした。
- Notify me when a service is automatically added: 新しく発見されたサービスがモニター対象のサービスのリストに追加されると、タスクバーから吹き出し通知を表示します。
- Notify me when a service changes status: モニター対象のサービスのステータスが変更されると、タスクバーから吹き出し通知を表示します。

図 2.34 MySQL Notifier Options メニュー



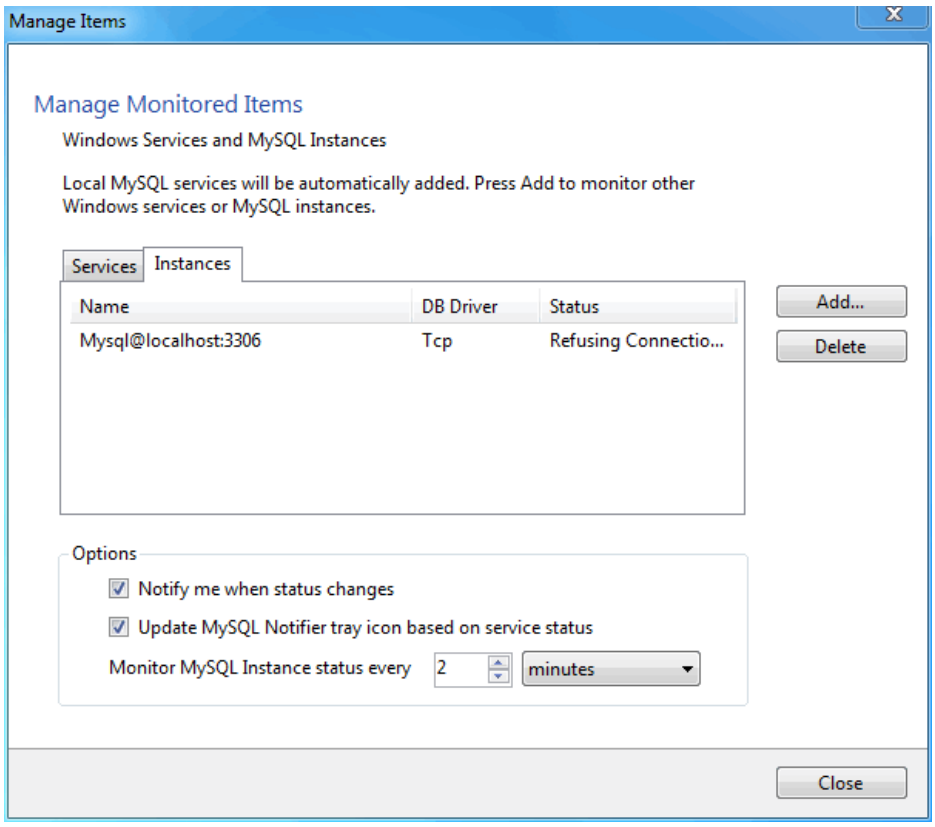
「Actions」、「Manage Monitored Items」メニューにより、モニター対象のサービスおよび MySQL インスタンスを構成できます。まず、「Services」タブで次を開きます。

図 2.35 MySQL Notifier Manage Services メニュー



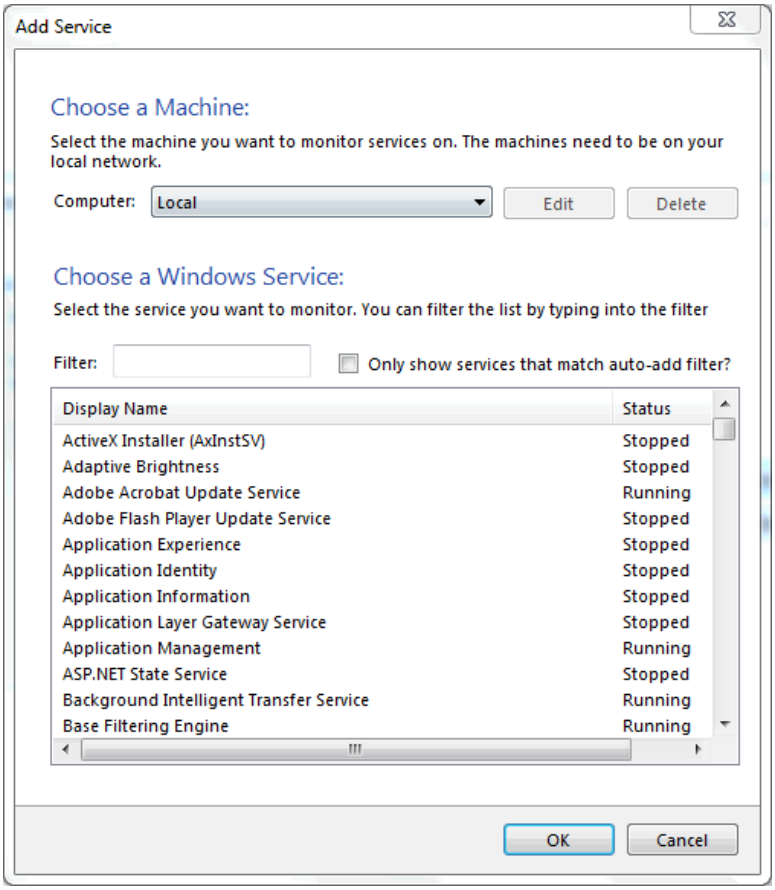
「Instances」タブは同様です。

図 2.36 MySQL Notifier Manage Instances メニュー



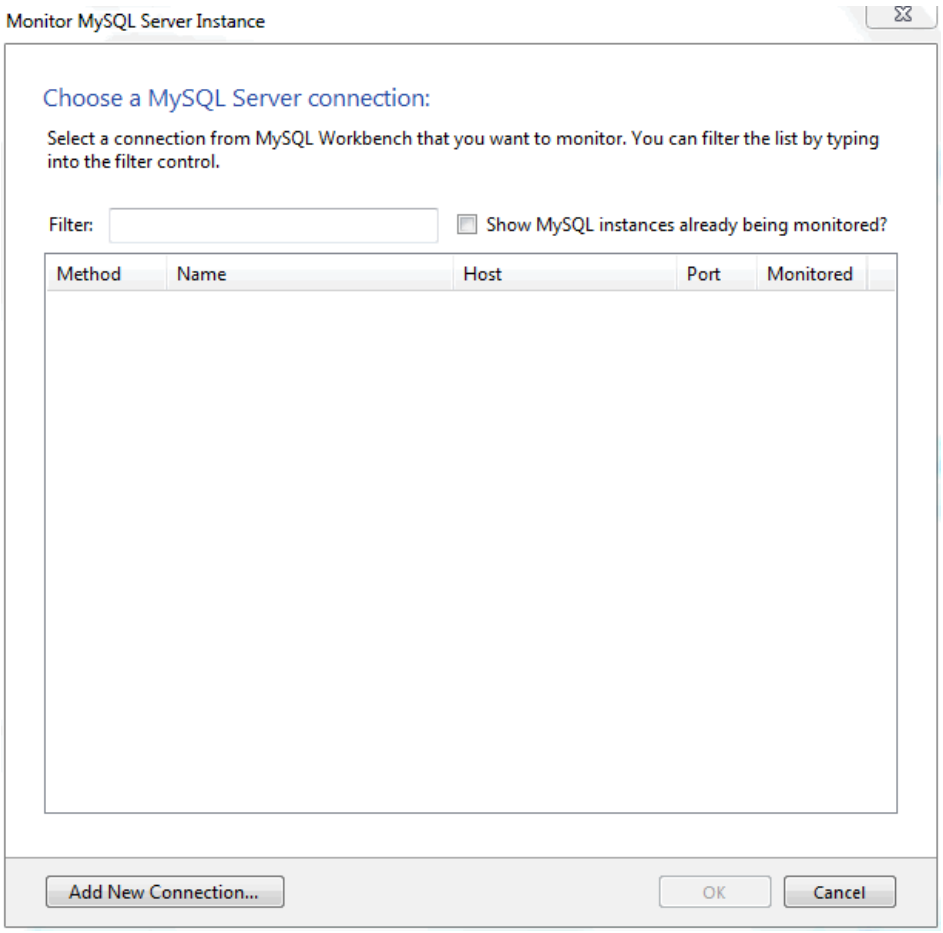
サービスまたはインスタンスを追加 (「Manage Monitored Items」ウィンドウで「Add」をクリック) すると、実行中の Microsoft Windows サービスまたはインスタンス接続を選択して、MySQL Notifier がそれをモニターするように構成できます。リストからサービス名をクリックして新しいサービスまたはインスタンスを追加してから、「OK」をクリックして受け入れます。複数のサービスおよびインスタンスを選択できます。

図 2.37 MySQL Notifier 新しいサービスの追加



インスタンスの追加

図 2.38 MySQL Notifier 新しいインスタンスの追加



> 注記
>
> 「Instances」タブは MySQL Notifier 1.1.0 以降利用可能です。

2.3.4.1 リモートモニタリングのセットアップとインストール手順

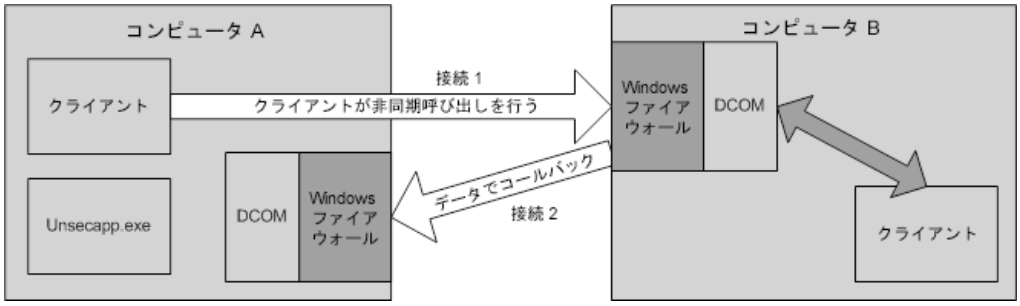
MySQL Notifier は、Windows Management Instrumentation (WMI) を使用して、Windows XP 以降を実行しているリモートコンピュータのサービスの管理とモニターを行います。このガイドでは、その仕組み、およびシステムをセットアップしてリモート MySQL インスタンスをモニターする方法を説明します。

> 注記
>
> リモートモニタリングは MySQL Notifier 1.1.0 以降利用可能です。

WMI を構成するためには、ベースとなる Distributed Component Object Model (DCOM) アーキテクチャーが WMI の作業を行なっていることを理解することが重要です。具体的には、MySQL Notifier は、リモート Microsoft Windows ホストの非同期通知クエリーを .NET イベントとして使用しています。これらのイベントは、MySQL Notifier を実行しているコンピュータに非同期コールバックを送信するため、MySQL Notifier はリモートコンピュータ上のサービスのステータスがいつ変更されたかを認識できます。非同期通知は、タイマーを使用する準同期通知または同期通知と比較して、パフォーマンスが優れています。

非同期通知では、リモートコンピュータがコールバックをクライアントコンピュータに送信する (したがって逆接続をオープンする) ことが必要なため、通信が正しく機能するためには、Windows ファイアウォールおよび DCOM が正しく構成されていなければなりません。

図 2.39 MySQL Notifier Distributed Component Object Model (DCOM)



非同期 WMI 通知によってスローされる一般的なエラーのほとんどは、Windows ファイアウォールが通信をブロックすること、または DCOM / WMI 設定が正しくセットアップされていないことに関係しています。一般的なエラーとその解決方法のリストは、Common Errorsを参照してください。

次の手順は、WMI を機能させるために必須です。これらの手順は 2 つのマシン間にわかれてます。MySQL Notifier を実行する単独のホストコンピュータ (コンピュータ A) と、モニターされる複数のリモートマシン (コンピュータ B) です。

MySQL Notifier を実行しているコンピュータ (コンピュータ A)

1. 「グループ ポリシー エディター」を編集するか、または NETSH を使用して、リモート管理を有効にします。

   「グループ ポリシー エディター」 を使用する場合

   a. 「スタート」をクリックしてから「実行」をクリックし、「GPEDIT.MSC」と入力してから「OK」をクリックします。

   b. 「ローカル コンピュータ ポリシー」見出しの「コンピューターの構成」をダブルクリックします。

   c. 「管理用テンプレート」、「ネットワーク」、「ネットワーク接続」、「Windows ファイアウォール」の順にダブルクリックします。

   d. コンピュータがドメイン内にある場合は、「ドメイン プロファイル」をダブルクリックします。そうでない場合は「標準プロファイル」をダブルクリックします。

   e. 「Windows ファイアウォール: 着信リモート管理の例外を許可する」をクリックします。

   f. 「アクション」メニューで、「編集」を選択するか、前の手順で選択したものをダブルクリックします。

   g. 「有効」ラジオボタンをチェックして「OK」をクリックします。

   NETSH コマンドを使用する場合

   a. 管理者権限でコマンドプロンプトウィンドウをオープンします (「コマンド プロンプト」アイコンを右クリックして「管理者として実行」をクリックします)。

   b. 次のコマンドを実行します。

   ```
   NETSH firewall set service RemoteAdmin enable
   ```

2. DCOM ポート TCP 135 のオープン

   a. 管理者権限でコマンドプロンプトウィンドウをオープンします (「コマンド プロンプト」アイコンを右クリックして「管理者として実行」をクリックします)。

   b. 次のコマンドを実行します。

   ```
   NETSH firewall add portopening protocol=tcp port=135 name=DCOM_TCP135
   ```

3. コールバックのシンクを含むクライアントアプリケーションを (MySqlNotifier.exe) Windows ファイアウォールの例外リストに追加します (Windows ファイアウォール構成または NETSH を使用します)。

   Windows ファイアウォール構成を使用する場合

   a. 「コントロール パネル」で、「Windows ファイアウォール」をダブルクリックします。

b. 「Windows ファイアウォール」ウィンドウの左のパネルで、「Windows ファイアウォールを介したプログラムまたは機能を許可する」をクリックします。

c. 「許可されたプログラム」ウィンドウで「設定の変更」をクリックします。

d. MySqlNotifier.exe が許可されたプログラムと機能のリスト内にある場合は、コンピュータが接続するネットワークのタイプ (プライベート、パブリック、または両方) にチェックされていることを確認します。

e. MySqlNotifier.exe がリストにない場合は、「別のプログラムの許可...」をクリックします。

f. 「プログラムの追加」ウィンドウで、MySqlNotifier.exe が「プログラム」リストにある場合は選択し、そうでない場合は「参照...」をクリックして MySqlNotifier.exe がインストールされたディレクトリに移動して選択してから、「追加」をクリックします。

g. MySqlNotifier.exe が、コンピュータが接続するネットワークのタイプ (プライベート、パブリック、または両方) にチェックされていることを確認します。

NETSH コマンドを使用する場合

a. 管理者権限でコマンドプロンプトウィンドウをオープンします (「コマンド プロンプト」アイコンを右クリックして 「管理者として実行」をクリックします)。

b. 次のコマンドを実行します。このとき、「[YOUR_INSTALL_DIRECTORY]」は変更します。

```
NETSH firewall add allowedprogram program=[YOUR_INSTALL_DIRECTORY]\MySqlNotifier.exe name=MySqlNotifier
```

4. コンピュータ B が WORKGROUP のメンバーであるか、またはコンピュータ A によって信頼されていない別のドメインにある場合は、コールバック接続 (接続 2) が匿名接続として作成されます。匿名接続に DCOM リモートアクセス権限を付与するには:

a. 「スタート」、「実行」をクリックし、「DCOMCNFG」と入力してから「OK」をクリックします。

b. 「コンポーネント サービス」ダイアログボックスで、「コンポーネント サービス」、「コンピューターの順に展開してから「マイ コンピューター」を右クリックし、「プロパティ」をクリックします。

c. 「My Computer Properties」ダイアログボックスで「COM セキュリティ」タブをクリックします。

d. 「アクセス許可」で、「制限の編集」をクリックします。

e. 「アクセス許可」ダイアログボックスで、「グループ」または「ユーザー名」ボックスから「ANONYMOUS LOGON 名」を選択します。「ユーザー」の「アクセス許可」の「許可」カラムで、「リモート アクセス」を選択してから「OK」をクリックします。

モニター対象のリモートコンピュータ (コンピュータ B)

MySQL Notifier を実行しているコンピュータ (コンピュータ A) にログインしたユーザーアカウントがリモートコンピュータ (コンピュータ B) のローカル管理者である場合、すなわち同じアカウントがコンピュータ B では管理者である場合、「リモート管理の許可」手順までスキップできます。

管理者以外のユーザーがリモートでコンピュータにアクセスできるようにするための DCOM セキュリティーの設定

1. ユーザーまたはグループに「DCOM のリモート起動」と有効化の権限を付与します。

a. 「スタート」、「実行」をクリックし、「DCOMCNFG」と入力してから「OK」をクリックします。

b. 「コンポーネント サービス」ダイアログボックスで、「コンポーネント サービス」、「コンピューターの順に展開してから「マイ コンピューター」を右クリックし、「プロパティ」をクリックします。

c. 「My Computer Properties」ダイアログボックスで「COM セキュリティ」タブをクリックします。

d. 「アクセス許可」で、「制限の編集」をクリックします。

e. 「グループ」または「ユーザー名」のリストにご自分の名前またはグループが表示されない場合は、「Launch Permission」ダイアログボックスで次の手順に従ってください。

i. 「起動アクセス許可」ダイアログボックスで、「追加」をクリックします。

ii. 「Select Users, Computers, or Groups」ダイアログボックスで、「選択するオブジェクト名を入力してください」ボックスに自分の名前とグループを追加してから、「OK」をクリックします。

f. 「Launch Permission」ダイアログボックスで、「グループ」または「ユーザー名」ボックスから自分のユーザーおよびグループを選択します。「ユーザー」の「アクセス許可」の「許可」カラムで、「リモートからの起動」を選択し、「リモートからのアクティブ化」を選択してから「OK」をクリックします。

DCOM リモートアクセス許可の付与

a. 「スタート」、「実行」をクリックし、「DCOMCNFG」と入力してから「OK」をクリックします。

b. 「コンポーネント サービス」ダイアログボックスで、「コンポーネント サービス」、「コンピューターの順に展開してから「マイ コンピューター」を右クリックし、「プロパティ」をクリックします。

c. 「My Computer Properties」ダイアログボックスで「COM セキュリティ」タブをクリックします。

d. 「アクセス許可」で、「制限の編集」をクリックします。

e. 「アクセス許可」ダイアログボックスで、「グループ」または「ユーザー名」ボックスから「ANONYMOUS LOGON 名」を選択します。「ユーザー」の「アクセス許可」の「許可」カラムで、「リモート アクセス」を選択してから「OK」をクリックします。

2. 管理者以外のユーザーへの特定の WMI 名前空間へのアクセス許可

a. 「コントロール パネル」で、「管理ツール」をダブルクリックします。

b. 「管理ツール」ウィンドウで、「コンピューターの管理」をダブルクリックします。

c. 「コンピューターの管理」ウィンドウで、「サービスとアプリケーション」ツリーを展開して「WMI コントロール」をダブルクリックします。

d. 「WMI コントロール」アイコンを右クリックして、「プロパティ」を選択します。

e. 「WMI Control Properties」ウィンドウで、「セキュリティ」タブをクリックします。

f. 「セキュリティ」タブで、名前空間を選択して「セキュリティ」をクリックします。

g. 適切なアカウントを探して「Remote Enable in the Permissions list」にチェックを付けます。

3. 「グループ ポリシー エディター」を編集するか、または NETSH を使用して、リモート管理を有効にします。

   「グループ ポリシー エディター」 を使用する場合

   a. 「スタート」をクリックしてから「実行」をクリックし、「GPEDIT.MSC」と入力してから「OK」をクリックします。

   b. 「ローカル コンピュータ ポリシー」見出しの「コンピューターの構成」をダブルクリックします。

   c. 「管理用テンプレート」、「ネットワーク」、「ネットワーク接続」、「Windows ファイアウォール」の順にダブルクリックします。

   d. コンピュータがドメイン内にある場合は、「ドメイン プロファイル」をダブルクリックします。そうでない場合は「標準プロファイル」をダブルクリックします。

   e. 「Windows ファイアウォール: 着信リモート管理の例外を許可する」をクリックします。

   f. 「アクション」メニューで、「編集」を選択するか、前の手順で選択したものをダブルクリックします。

   g. 「有効」ラジオボタンをチェックして「OK」をクリックします。

   NETSH コマンドを使用する場合

   a. 管理者権限でコマンドプロンプトウィンドウをオープンします (「コマンド プロンプト」アイコンを右クリックして「管理者として実行」をクリックします)。

   b. 次のコマンドを実行します。

   ```
   NETSH firewall set service RemoteAdmin enable
   ```

4. ここで、Full Name 値ではなく Name 値を使用してログインしていることを確認します。

   a. 「コントロール パネル」で、「管理ツール」をダブルクリックします。

   b. 「管理ツール」ウィンドウで、「コンピューターの管理」をダブルクリックします。

   c. 「コンピューターの管理」ウィンドウで、「システム ツール」、「ローカル ユーザーとグループ」の順に展開します。

   d. 「ユーザー」ノードをクリックし、右側のパネルで自分のユーザーを探し、接続に Full Name 値ではなく Name 値を使用していることを確認します。

5. リモートコンピュータが Windows XP Professional 上で実行されている場合は、リモートログインが強制的にゲストアカウントユーザーに変更されていないことを確認します (ForceGuest とも呼ばれます)。これは、ドメインに接続されていないコンピュータではデフォルトで有効です。

   a. 「スタート」をクリックしてから「実行」をクリックし、「SECPOL.MSC」と入力してから「OK」をクリックします。

   b. 「ローカル ポリシー」ノードで、「セキュリティ オプション」をダブルクリックします。

   c. 「ネットワーク アクセス: ローカル アカウントの共有とセキュリティ モデル」を選択して保存します。

一般的なエラー

- 0x80070005
  - DCOM セキュリティーが正しく構成されませんでした (コンピュータ B、DCOM セキュリティーの設定... の手順を参照してください)。
  - リモートコンピュータ (コンピュータ B) が、WORKGROUP のメンバーであるか、またはクライアント (コンピュータ A) によって信頼されていないドメインにあります (コンピュータ A、匿名接続に DCOM リモートアクセス権限を付与するには の手順を参照してください)。
- 0x8007000E

- リモートコンピュータ (コンピュータ B) が、WORKGROUP のメンバーであるか、またはクライアント (コンピュータ A) によって信頼されていないドメインにあります (コンピュータ A、匿名接続に DCOM リモートアクセス権限を付与するにはの手順を参照してください)。

- 0x80041003

  - リモート WMI 名前空間へのアクセスが正しく構成されませんでした (コンピュータ B、管理者以外のユーザーへの特定の WMI 名前空間へのアクセス許可の手順を参照してください)。

- 0x800706BA

  - クライアントコンピュータ (コンピュータ A) のファイアウォールで DCOM ポートがオープンしていません。コンピュータ A の、DCOM ポート TCP 135 のオープンの手順を参照してください。

  - リモートコンピュータ (コンピュータ B) のネットワークの場所がパブリックに設定されているため、アクセスできません。Windows エクスプローラーを介してアクセスできることを確認してください。

2.3.5 非インストール Zip アーカイブを使用して Microsoft Windows に MySQL をインストールする

非インストールパッケージからインストールするユーザーは、このセクションの説明に従って手動で MySQL をインストールできます。MySQL を Zip アーカイブからインストールする手順は次のとおりです。

1. アーカイブを任意のインストールディレクトリに抽出する
2. オプションファイルを作成する
3. MySQL のサーバータイプを選択する
4. MySQL Server を起動する
5. デフォルトのユーザーアカウントをセキュアにする

この手順はこのあとのセクションで説明します。

2.3.5.1 インストールアーカイブを抽出する

MySQL を手動でインストールするには、次の手順に従います。

1. 以前のバージョンからアップグレードする場合にはアップグレードの手順を始める前に セクション 2.3.7「Windows 上の MySQL をアップグレードする」 を参照してください。
2. 管理者権限を持つユーザーとしてログインしていることを確認します。
3. インストールの場所を選択します。通常、MySQL Server は C:\mysql にインストールされます。MySQL Installation Wizard が MySQL を C:\Program Files\MySQL にインストールします。MySQL を C:\mysql にインストールしない場合、起動時あるいはオプションファイルでインストールディレクトリへのパスを指定する必要があります。セクション2.3.5.2「オプションファイルの作成」 を参照してください。

   > **注記**
   >
   > MySQL Installer が MySQL を C:\Program Files\MySQL にインストールします。

4. 任意の Zip アーカイブツールを使用して、インストールするアーカイブを選択したインストール場所に抽出します。ツールによっては、選択したインストール場所のフォルダにアーカイブを抽出します。その場合、そのサブフォルダの内容を、選択したインストール場所に移動します。

2.3.5.2 オプションファイルの作成

サーバーの起動時にスタートアップオプションを指定する必要がある場合には、それらをコマンド行で指示するかオプションファイルに配置します。サーバーの起動時に常に使用するオプションは、オプションファイルを使用して MySQL の構成を指定すると非常に便利です。これは次の状況に特に当てはまります。

- インストールまたはデータディレクトリの場所が、デフォルトの場所 (C:\Program Files\MySQL\MySQL Server 5.6 および C:\Program Files\MySQL\MySQL Server 5.6\data) とは異なり場合。
- メモリー、キャッシュ、InnoDB 構成情報などのサーバー設定を調整する必要がある場合。

MySQL Server が Windows 上で起動する際、Windows ディレクトリ C:\、および MySQL インストールディレクトリなど、いくつかの場所にあるオプションファイルを検索します (場所の完全なリストについては、セクション4.2.6「オプションファイルの使用」 を参照してください)。通常、Windows ディレクトリの名前は、C:\WINDOWS のようになります。次のコマンドを使用して WINDIR 環境変数の値から正確な場所を割り出すことができます。

```
C:\> echo %WINDIR%
```

MySQL は、各場所の my.ini ファイル内のオプションを最初に検索し、次に my.cnf ファイルを調べます。ただし、混乱を避けるためにはファイルを 1 つだけ使用するのが最善です。お客様の PC が、C: がブートドライブではないブートローダーを使用している場合、my.ini ファイルしか使用できません。どちらのオプションファイルを使用するにしろ、プレーンテキストファイルでなければなりません。

> **注記**
>
> MySQL Installer を使用して MySQL Server をインストールする場合、my.ini はデフォルトの場所に作成されます。また、MySQL Server 5.5.27 では、MySQL Installer を実行するユーザーにはこの新しい my.ini への完全な許可が与えられます。
>
> つまり、必ず MySQL Server ユーザーが my.ini ファイルを読み取る権限があるようにしてください。

MySQL 配布に含まれている参考例のオプションファイルを使用することもできます。詳細は セクション5.1.2「サーバー構成のデフォルト値」 を参照してください。

オプションファイルは、ノートパッドなどのテキストエディタで作成および変更ができます。たとえば、MySQL を E:\mysql にインストールし、データディレクトリが E:\mydata\data にある場合は、[mysqld] セクションを含むオプションファイルを作成して、basedir および datadir オプションの値を指定できます。

```
[mysqld]
# set basedir to your installation path
basedir=E:/mysql
# set datadir to the location of your data directory
datadir=E:/mydata/data
```

Microsoft Windows のパス名は、オプションファイル内でバックスラッシュではなく (フォワード) スラッシュを使用して指定されます。バックスラッシュを使用する場合は、2 つ使用します。

```
[mysqld]
# set basedir to your installation path
basedir=E:\\mysql
# set datadir to the location of your data directory
datadir=E:\\mydata\\data
```

オプションファイル値でのバックスラッシュの使用に関するルールは セクション4.2.6「オプションファイルの使用」 にあります。

データディレクトリは、MySQL を実行しているユーザーの AppData ディレクトリにあります。

データディレクトリを異なる場所で使用したい場合、data ディレクトリの内容全体を新しい場所にコピーしてください。たとえば、代わりに E:\mydata をデータディレクトリとして使用する場合は、次の 2 つのことを行う必要があります。

1. data ディレクトリおよびその内容すべてを、デフォルトの場所 (C:\Program Files\MySQL\MySQL Server 5.6\data など) から E:\mydata に移動します。
2. サーバーの起動時に常に新しいデータディレクトリの場所を指定するには --datadir オプションを使用します。

2.3.5.3 MySQL サーバータイプの選択

次の表は、Windows で使用できる MySQL 5.6 のサーバーを示しています。

バイナリ	説明
mysqld	名前付きパイプのサポートがある最適化バイナリ
mysqld-debug	mysqld と同様ですが、完全なデバッグと自動メモリー割り当ての確認を行なってコンパイルされています

以前のバイナリはすべて最新の Intel プロセッサに最適化されていますが、Intel i386 クラスあるいはそれ以上のプロセッサで動作するはずです。

配布内の各サーバーは、同じストレージエンジンセットをサポートします。SHOW ENGINES ステートメントは、指定されたサーバーがサポートするエンジンを表示します。

すべての Windows MySQL 5.6 Server は、データベースディレクトリのシンボリックリンクをサポートします。

MySQL はすべての Windows プラットフォームで TCP/IP をサポートしています。Windows 上の MySQL は、サーバーを --enable-named-pipe オプション付きで起動した場合は名前付きパイプもサポートします。一部のユーザーで、名前付きパイプを使用した際に MySQL サーバーのシャットダウンに関する問題が生じたため、このオプションは明示的に使用する必要があります。多くの Windows 構成において、名前付きパイプは TCP/IP より遅いため、デフォルトではプラットフォームにかかわらず TCP/IP が使用されます。

2.3.5.4 サーバーをはじめて起動する

このセクションでは MySQL サーバーの起動に関する一般的な概要を説明します。次の数セクションでは、MySQL サーバーのコマンド行あるいは Windows のサービスとしての起動に関して、より具体的な情報を提供します。

ここに記載する情報は、MySQL を 非インストール バージョンを使用してインストールした場合、あるいは MySQL を GUI ツールを使用しないで手動で構成してテストする場合に適用されます。

> **注記**
>
> MySQL Server は、MySQL Installer を使用したあと自動的に起動し、いつでも MySQL Notifier GUI を使用して起動/停止/再起動できます。

これらのセクションの例では、MySQL がデフォルトの場所の C:\Program Files\MySQL\MySQL Server 5.6 にインストールされていることを前提とします。異なる場所に MySQL をインストールしている場合には例に示したパス名を調整してください。

クライアントには 2 つのオプションがあります。TCP/IP が使用でき、サーバーが名前付きパイプの接続をサポートしている場合には名前付きパイプも使用できます。

サーバーを --shared-memory オプションで起動した場合、Windows 上の MySQL は共有メモリーの接続もサポートします。クライアントは --protocol=MEMORY オプションを使用して共有メモリーで接続できます。

どのサーバーバイナリを実行するかについては、セクション2.3.5.3「MySQL サーバータイプの選択」 を参照してください。

テストはコンソールウィンドウ (あるいは 「DOS ウィンドウ」) のコマンドプロンプトで行います。これにより、ウィンドウにサーバーのステータスに関するメッセージが表示されますので、容易に確認できます。構成に何か問題があった場合には、これらのメッセージにより問題の特定と修正が容易になります。

サーバーを起動するには、次のコマンドを入力します。

```
C:\> "C:\Program Files\MySQL\MySQL Server 5.6\bin\mysqld" --console
```

InnoDB サポートを含むサーバーでは、次のようなメッセージがサーバーの起動時に表示されます (パス名とサイズは異なる場合があります)。

```
InnoDB: The first specified datafile c:\ibdata\ibdata1 did not exist:
InnoDB: a new database to be created!
InnoDB: Setting file c:\ibdata\ibdata1 size to 209715200
InnoDB: Database physically writes the file full: wait...
InnoDB: Log file c:\iblogs\ib_logfile0 did not exist: new to be created
InnoDB: Setting log file c:\iblogs\ib_logfile0 size to 31457280
InnoDB: Log file c:\iblogs\ib_logfile1 did not exist: new to be created
InnoDB: Setting log file c:\iblogs\ib_logfile1 size to 31457280
InnoDB: Log file c:\iblogs\ib_logfile2 did not exist: new to be created
InnoDB: Setting log file c:\iblogs\ib_logfile2 size to 31457280
InnoDB: Doublewrite buffer not found: creating new
InnoDB: Doublewrite buffer created
InnoDB: creating foreign key constraint system tables
InnoDB: foreign key constraint system tables created
011024 10:58:25  InnoDB: Started
```

サーバーが起動シーケンスを終了すると、次のようなメッセージが表示されます。このメッセージが表示されると、サーバーのクライアント接続の用意が整ったことを意味します。

```
mysqld: ready for connections
```

```
Version: '5.6.23'  socket: ''  port: 3306
```

サーバーは、診断を生成して出力をコンソールに書き込み続けます。クライアントプログラムを実行する新しいコンソールウィンドウを開くことができます。

--console オプションを省略すると、サーバーは診断の出力をデータディレクトリ (デフォルトでは C:\Program Files\MySQL\MySQL Server 5.6\data) のエラーログに出力します。エラーログは、.err 拡張子の付いたファイルで、--log-error オプションを使用して設定できます。

> **注記**
>
> MySQL の付与テーブルにリストされているアカウントには、最初はパスワードがありません。サーバーの起動後に セクション2.10.2「最初の MySQL アカウントのセキュリティー設定」 の説明に従ってパスワードをアカウントに設定する必要があります。

2.3.5.5 Windows のコマンド行からの MySQL の起動

MySQL Server はコマンド行から手動で起動できます。これは Windows のどのバージョンでもできます。

> **注記**
>
> MySQL Notifier GUI を使用して、MySQL Server を起動/停止/再起動することも可能です。

コマンド行から mysqld サーバーを起動するには、コンソールウィンドウ (あるいは 「DOS ウィンドウ」) を開け、次のコマンドを入力します。

```
C:\> "C:\Program Files\MySQL\MySQL Server 5.6\bin\mysqld"
```

mysqld へのパスはお客様のシステムの MySQL のインストール場所によって異なる場合があります。

MySQL Server を停止するには次のコマンドを実行します。

```
C:\> "C:\Program Files\MySQL\MySQL Server 5.6\bin\mysqladmin" -u root shutdown
```

> **注記**
>
> MySQL の root ユーザーアカウントにパスワードが設定されている場合は、mysqladmin を -p オプションを使って起動し、要求されたらパスワードを入力する必要があります。

このコマンドは MySQL 管理ユーティリティー mysqladmin を起動してサーバーに接続し、サーバーにシャットダウンを命令します。そのコマンドは MySQL root ユーザーとして接続します。これは MySQL の許可システムのデフォルトの管理アカウントです。

> **注記**
>
> MySQL 許可システム内のユーザーは、Microsoft Windows のログインユーザーとは完全に独立しています。

mysqld が起動しない場合、エラーログを確認してサーバーがその問題の原因を示すメッセージを書き込んでいないか確認します。デフォルトでは、エラーログは C:\Program Files\MySQL\MySQL Server 5.6\data ディレクトリにあります。サフィクス .err の付いたファイルです。または、--log-error オプションで渡して指定できます。あるいは、--console オプションを使用してサーバーを起動することもできます。この場合、サーバーは問題の解決に役立つ有用な情報を画面に表示することがあります。

最後の選択肢は、mysqld を --standalone および --debug オプションで起動することです。この場合、mysqld がログファイル C:\mysqld.trace に書き出し、その中に mysqld が起動しない理由が含まれているはずです。セクション24.4.3「DBUG パッケージ」を参照してください。

mysqld --verbose --help を使用して、mysqld がサポートするすべてのオプションを表示します。

2.3.5.6 MySQL ツールの PATH をカスタマイズする

MySQL プログラムの起動を簡単にするために、MySQL の bin ディレクトリのパス名を Windows システムの PATH 環境変数に追加できます。

- Windows デスクトップで、「マイコンピュータ」 アイコンを右クリックして「プロパティ」を選択します。
- 次に、表示された「システムのプロパティ」メニューから「詳細設定」タブを選択し、「環境変数」ボタンをクリックします。

- 「システム環境変数」で、「Path」を選択し、次に「編集」ボタンをクリックします。「システム変数の編集」のダイアログが表示されます。
- 「変数値」と示されたスペースに表示されたテキストの最後にカーソルを持って行きます。(「End」キーを使用して、カーソルが確実にそのスペースのテキストのいちばん後ろにあるようにします)。次に、MySQL bin ディレクトリの完全なパス名を入力します (C:\Program Files\MySQL\MySQL Server 5.6\bin など)。

  > 注記
  >
  > このフィールドにあるほかの値とこのパスを区切るために、セミコロンが必要です。

  開いているダイアログがすべて閉じるまで、「OK」をクリックしてこのダイアログとその他のダイアログを順番に閉じます。この段階で、システムのどのディレクトリからでも、DOS プロンプトにパスを指定せずに MySQL の実行プログラム名を入力して、すべての MySQL 実行プログラムを呼び出すことができます。これにはサーバー、mysql クライアント、および mysqladmin と mysqldump などのすべての MySQL コマンド行ユーティリティーが含まれています。

  同じマシンで複数の MySQL サーバーを動作させている場合には、MySQL bin ディレクトリを Windows PATH に追加しないでください。

  > 警告
  >
  > システムの PATH を手動で編集する際には最大限の注意が必要です。既存の PATH の値の一部でも間違って削除したり変更したりすると、誤動作を引き起こしたりシステムが使用できなくなったりする場合があります。

2.3.5.7 Windows のサービスとして MySQL を起動する

Windows で MySQL を実行する場合、Windows の起動および停止時に MySQL が自動的に起動および停止するように、Windows サービスとしてインストールすることが推奨されています。MySQL Server をサービスとしてインストールすると、NET コマンドを使用してコマンド行から、あるいはグラフィカル Services ユーティリティーで管理することもできます。一般に、MySQL を Windows サービスとしてインストールするときは、管理者権限のあるアカウントを使ってログインするようにしてください。

> 注記
>
> MySQL Notifier GUI を使用して、MySQL サービスのステータスをモニターすることも可能です。

Services ユーティリティー (Windows サービスコントロールマネージャー) は、Windows のコントロールパネル (Windows 2000、XP、Vista、および Server 2003 では 管理ツールの下) にあります。競合を避けるために、サーバーのインストールまたはコマンド行からの削除操作の際には、Services ユーティリティーを閉じることを推奨します。

サービスのインストール

現在サーバーを実行中の場合は、MySQL を Windows のサービスとしてインストールする前に次のコマンドを使用して停止してください。

```
C:\> "C:\Program Files\MySQL\MySQL Server 5.6\bin\mysqladmin"
        -u root shutdown
```

> 注記
>
> MySQL の root ユーザーアカウントにパスワードが設定されている場合は、mysqladmin を -p オプションを使って起動し、要求されたらパスワードを入力する必要があります。

このコマンドは MySQL 管理ユーティリティー mysqladmin を起動してサーバーに接続し、サーバーにシャットダウンを命令します。そのコマンドは MySQL root ユーザーとして接続します。これは MySQL の許可システムのデフォルトの管理アカウントです。

> 注記
>
> MySQL 許可システム内のユーザーは、Windows のログインユーザーとは完全に独立しています。

サーバーをこのコマンドを使用してサービスとしてインストールします。

```
C:\> "C:\Program Files\MySQL\MySQL Server 5.6\bin\mysqld" --install
```

サービスのインストールコマンドはサーバーは起動しません。それについてはこのセクションで後述します。

MySQL プログラムの起動を簡単にするために、MySQL の bin ディレクトリのパス名を Windows システムの PATH 環境変数に追加できます。

- Windows デスクトップで、「マイコンピュータ」 アイコンを右クリックして「プロパティ」を選択します。
- 次に、表示された「システムのプロパティ」メニューから「詳細設定」タブを選択し、「環境変数」ボタンをクリックします。
- 「システム環境変数」で、「Path」を選択し、次に「編集」ボタンをクリックします。「システム変数の編集」のダイアログが表示されます。
- 「変数値」と示されたスペースに表示されたテキストの最後にカーソルを持って行きます。(「End」キーを使用して、カーソルが確実にそのスペースのテキストのいちばん後ろにあるようにします)。次に、MySQL bin ディレクトリの完全なパス名を入力します (C:\Program Files\MySQL\MySQL Server 5.6\bin など)。このフィールドにあるほかの値とこのパスを区切るために、セミコロンが必要です。開いているダイアログがすべて閉じるまで、「OK」をクリックしてこのダイアログとその他のダイアログを順番に閉じます。この段階で、システムのどのディレクトリからでも、DOS プロンプトにパスを指定せずに MySQL の実行プログラム名を入力して、すべての MySQL 実行プログラムを呼び出すことができます。これにはサーバー、mysql クライアント、および mysqladmin と mysqldump などのすべての MySQL コマンド行ユーティリティーが含まれています。

  同じマシンで複数の MySQL サーバーを動作させている場合には、MySQL bin ディレクトリを Windows PATH に追加しないでください。

  > **警告**
  >
  > システムの PATH を手動で編集する際には最大限の注意が必要です。既存の PATH の値の一部でも間違って削除したり変更したりすると、誤動作を引き起こしたりシステムが使用できなくなったりする場合があります。

サービスをインストールする際に、次の追加引数を使用できます。

- --install オプションの直後にサービス名を指定できます。デフォルトのサービス名は MySQL です。
- サービス名を指定すると、そのあとに 1 つのオプションを指定できます。規則では、--defaults-file=file_name でオプションファイル名を指定し、サーバーが起動時にそこからオプションを読み取ります。

  --defaults-file 以外に 1 つのオプションを使用することは可能ですが、お勧めできません。--defaults-file を使用すると、指名したオプションファイルに複数のスタートアップオプションを指定できるので、より柔軟です。
- サービス名のあとに --local-service オプションも指定できます。これにより、システム権限の制限付き LocalService Windows アカウントを使用してサーバーを起動できます。このアカウントは Windows XP またはそれ以降でのみ利用できます。--defaults-file および --local-service の両方がサービス名のあとに指定される場合、どのような順序でもかまいません。

MySQL Server を Windows のサービスとしてインストールした場合、サーバーが使用するサービス名およびオプションファイルは次のルールで決められます。

- サービスインストールのコマンドで、--install オプションのあとにサービス名を指名しないかデフォルトのサービス名 (MySQL) を指定した場合、サーバーは MySQL のサービス名を使用し、標準のオプションファイルの [mysqld] グループのオプションを読み取ります。
- サービスインストールのコマンドで MySQL 以外のサービス名を --install オプションのあとに指定した場合、サーバーはそのサービス名を使用します。標準のオプションファイルの [mysqld] グループと、サービスと名前が同じグループからオプションを読み取ります。これにより、[mysqld] グループをすべての MySQL サービスで使用すべきオプション用に、サービス名の付いたオプショングループをそのサービス名でインストールされたサーバー用に使用できます。
- サービスインストールのコマンドでサービス名のあとに --defaults-file オプションを指定した場合、サーバーは前の項目で説明したのと同じ方法でオプションを読み取りますが、名前付きのファイルからのみオプションを読み取り、標準のオプションファイルは無視します。

さらに複雑な例として、次のコマンドがあります。

```
C:\> "C:\Program Files\MySQL\MySQL Server 5.6\bin\mysqld"
      --install MySQL --defaults-file=C:\my-opts.cnf
```

ここでは、--install オプションのあとにデフォルトのサービス名 (MySQL) が指定されています。--defaults-file オプションが指定されていなければ、このコマンドによりサーバーは標準のオプションファイルから [mysqld] グループを読み取ります。しかし、--defaults-file オプションが指定されているので、サーバーは [mysqld] オプショングループのオプションを、指名されたファイルのみから読み取ります。

> **注記**
>
> Windows では、サーバーが --defaults-file および --install オプションで起動される場合、--install が最初でなければなりません。そうでないと、mysqld.exe は MySQL Server を起動しようとします。

MySQL サービスを実行する前に、Windows の Services ユーティリティーで Start パラメータとしてオプションを指定することもできます。

サービスの開始

MySQL サーバーをサービスとしてインストールすると、Windows が起動されるたびにサービスが自動的に起動されます。サービスは、Services ユーティリティーから、あるいは NET START MySQL コマンドを使用してすぐに実行することもできます。NET コマンドは大文字と小文字を区別しません。

サービスとして起動する場合、mysqld はコンソールウィンドウにアクセスできないので、メッセージは表示されません。mysqld が起動しない場合、サーバーがエラーログにその問題の原因を知らせるメッセージを書き込んでいないか確認します。エラーログは MySQL データディレクトリ (C:\Program Files\MySQL\MySQL Server 5.6\data など) にあります。.err のサフィクスが付いたファイルです。

MySQL Server をサービスとしてインストールした場合、サービスを実行しているときに Windows がシャットダウンすると Windows が自動的にサービスを停止します。サーバーは、Services ユーティリティー、NET STOP MySQL コマンド、あるいは mysqladmin shutdown コマンドを使用して手動で停止することもできます。

サービスがブートプロセスで自動的に実行されないように、サーバーをマニュアルサービスとしてインストールすることもできます。これには --install オプションではなく --install-manual オプションを使用します。

```
C:\> "C:\Program Files\MySQL\MySQL Server 5.6\bin\mysqld" --install-manual
```

サービスの削除

サービスとしてインストールしたサーバーを削除するには、起動中の場合にはまず NET STOP MySQL を実行して停止します。次に --remove オプションを使用して削除します。

```
C:\> "C:\Program Files\MySQL\MySQL Server 5.6\bin\mysqld" --remove
```

mysqld をサービスとして実行していない場合は、コマンド行で起動できます。その手順は、セクション 2.3.5.5「Windows のコマンド行からの MySQL の起動」 を参照してください。

インストール中に問題が生じた場合は、セクション2.3.6「Microsoft Windows MySQL Server インストールのトラブルシューティング」 を参照してください。

2.3.5.8 MySQL インストールのテスト

次のコマンドのいずれかを実行して、MySQL Server が動作しているかテストできます。

```
C:\> "C:\Program Files\MySQL\MySQL Server 5.6\bin\mysqlshow"
C:\> "C:\Program Files\MySQL\MySQL Server 5.6\bin\mysqlshow" -u root mysql
C:\> "C:\Program Files\MySQL\MySQL Server 5.6\bin\mysqladmin" version status proc
C:\> "C:\Program Files\MySQL\MySQL Server 5.6\bin\mysql" test
```

mysqld のクライアントプログラムから TCP/IP 接続への反応が鈍い場合、おそらくの DNS に問題があります。この場合、mysqld を --skip-name-resolve オプションを使用して起動し、MySQL の付与テーブルの Host カラムの localhost および IP アドレスのみを使用します。

--pipe あるいは --protocol=PIPE オプションを指定するか、あるいは .(ピリオド) をホスト名として指定すると、MySQL クライアントに TCP/IP ではなく名前付けパイプ接続を強制的に使用させることができます。デフォルトのパイプ名を使用しない場合には --socket オプションを使用してパイプ名を指定します。

root アカウントのパスワードを設定した場合、匿名アカウントを削除した場合、または新規ユーザーアカウントを作成した場合に MySQL Server に接続するには、前述のコマンドに適切な -u および -p オプションを使用しなければなりません。セクション4.2.2「MySQL サーバーへの接続」 を参照してください。

mysqlshow に関する詳細は、セクション4.5.6「mysqlshow — データベース、テーブル、およびカラム情報の表示」 を参照してください。

2.3.6 Microsoft Windows MySQL Server インストールのトラブルシューティング

MySQL をインストールして最初に起動する際に、エラーが発生して MySQL サーバーが起動できない場合があります。このセクションは、エラーの診断と修正に役立ちます。

サーバーの問題のトラブルシューティングを行う際の、最初のリソースは エラーログです。MySQL Server は、サーバーが起動しない原因となるエラーに関する情報を記録するのにエラーログを使用しています。エラーログは、my.ini ファイルで指定された データディレクトリにあります。データディレクトリのデフォルトの場所は、Windows 7 および Windows Server 2008 では C:\Program Files\MySQL\MySQL Server 5.6\data、または C:\ProgramData\Mysql です。C:\ProgramData ディレクトリはデフォルトでは非表示です。ディレクトリと内容を表示するには、フォルダオプションを変更する必要があります。エラーログとその内容の理解については、セクション5.2.2「エラーログ」 を参照してください。

起こりうるエラーについては、MySQL サービスの起動中に表示されるコンソールメッセージも調べてください。mysqld をサービスとしてインストールしたあとに、コマンド行から NET START MySQL コマンドを使用して、MySQL Server のサービスとしての起動に関するエラーメッセージを表示します。セクション2.3.5.7「Windows のサービスとして MySQL を起動する」を参照してください。

MySQL をインストールしてサーバーを最初に起動する際によく生じる、その他の一般的なエラーメッセージを次の例に示します。

- MySQL Server が mysql 権限データベースまたはその他の重要なファイルを見つけられない場合、次のメッセージを表示します。

```
System error 1067 has occurred.
Fatal error: Can't open and lock privilege tables:
Table 'mysql.user' doesn't exist
```

これらのメッセージは、MySQL のベースディレクトリまたはデータディレクトリが、デフォルトの場所 (それぞれ C:\Program Files\MySQL\MySQL Server 5.6 および C:\Program Files\MySQL\MySQL Server 5.6\data) と異なる場所にインストールされている場合によく生じます。

この状況は、MySQL をアップグレードして新しい場所にインストールしたが、構成ファイルが新しい場所を反映するように更新されていない場合に生じます。さらに、新旧の構成ファイルが対立している場合もあります。MySQL をアップグレードする際は必ず旧構成ファイルを削除するか名前を変更してください。

MySQL を C:\Program Files\MySQL\MySQL Server 5.6 以外のディレクトリにインストールした場合は、構成 (my.ini) ファイルを使用して MySQL Server がそのことを認識できるようにしてください。my.ini ファイルを Windows ディレクトリ (通常 C:\WINDOWS) に置きます。WINDIR 環境変数の値から正確な場所を割り出すためには、コマンドプロンプトから次のコマンドを発行します。

```
C:\> echo %WINDIR%
```

オプションファイルはノートパッドなどのテキスト エディタで作成および変更できます。たとえば、MySQL が E:\mysql にインストールされていて、データディレクトリが D:\MySQLdata にある場合は、オプションファイルを作成して [mysqld] セクションを設定し、basedir および datadir オプションの値を指定できます。

```
[mysqld]
# set basedir to your installation path
basedir=E:/mysql
# set datadir to the location of your data directory
datadir=D:/MySQLdata
```

Microsoft Windows のパス名は、オプションファイル内でバックスラッシュではなく (フォワード) スラッシュを使用して指定されます。バックスラッシュを使用する場合は、2 つ使用します。

```
[mysqld]
# set basedir to your installation path
basedir=C:\\Program Files\\MySQL\\MySQL Server 5.6
# set datadir to the location of your data directory
datadir=D:\\MySQLdata
```

オプションファイル値でのバックスラッシュの使用に関するルールは セクション4.2.6「オプションファイルの使用」 にあります。

MySQL 構成ファイルの datadir の値を変更する場合は、MySQL Server を再起動する前に既存の MySQL データディレクトリの内容を移動する必要があります。

セクション2.3.5.2「オプションファイルの作成」 を参照してください。

- まず既存の MySQL サービスを停止して削除してから MySQL Installer を使用して MySQL をインストールせずに、MySQL を再インストールまたはアップグレードすると、次のエラーが生じる場合があります。

```
Error: Cannot create Windows service for MySql. Error: 0
```

これは Configuration Wizard がサービスをインストールしようとしたときに既存のサービスが同じ名前で存在する場合に発生します。

この問題の解決方法の 1 つは、Configuration Wizard を使用する際にmysql 以外のサービス名を選択することです。これにより、新しいサービスが正しくインストールされ、古いサービスはそのままにできます。これは特に問題はありませんが、使用しない古いサービスは削除したほうがよいでしょう。

古い mysql サービスを完全に削除するには、管理者権限を持つユーザーとして、コマンド行で次のコマンドを実行します。

```
C:\> sc delete mysql
[SC] DeleteService SUCCESS
```

使用している Windows のバージョンで sc ユーティリティーが利用できない場合は、delsrv ユーティリティーを http://www.microsoft.com/windows2000/techinfo/reskit/tools/existing/delsrv-o.asp からダウンロードして、delsrv mysql 構文を使用します。

2.3.7 Windows 上の MySQL をアップグレードする

Windows 上の MySQL をアップグレードするには、次の手順に従います。

1. セクション2.11.1「MySQL のアップグレード」 の、Windows に特化しない MySQL のアップグレードに関する詳細を参照してください。
2. アップグレードを実行する前に、必ず現在の MySQL インストールをバックアップしてください。セクション7.2「データベースバックアップ方法」を参照してください。
3. MySQL の最新の Windows 配布を https://dev.mysql.com/downloads/ からダウンロードします。
4. MySQL をアップグレードする前に、サーバーを停止します。サーバーをサービスとしてインストールしている場合は、コマンドプロンプトから次のコマンドでサービスを停止します。

   ```
   C:\> NET STOP MySQL
   ```

   MySQL Server をサービスとして実行していない場合は、mysqladmin を使用して停止します。たとえば、MySQL 5.5 から 5.6 にアップグレードする前に、mysqladmin を MySQL 5.5 から次のように使用します。

   ```
   C:\> "C:\Program Files\MySQL\MySQL Server 5.5\bin\mysqladmin" -u root shutdown
   ```

   > **注記**
   >
   > MySQL の root ユーザーアカウントにパスワードが設定されている場合は、mysqladmin を -p オプションを使って起動し、要求されたらパスワードを入力します。

5. MySQL 4.1.5 より前のバージョンから 5.6 へのアップグレード、または Zip アーカイブからインストールされた MySQL のバージョンから MySQL Installation Wizard でインストールされる MySQL のバージョンへのアップグレードの場合は、その前にまず手動で以前のインストールおよび MySQL サービス (サーバーがサービスとしてインストールされている場合) を削除しなければなりません。

   MySQL サービスを削除するには次のコマンドを使用します。

   ```
   C:\> C:\mysql\bin\mysqld --remove
   ```

   既存のサービスを削除しないと、MySQL Installation Wizard は新しい MySQL サービスを適切にインストールできない場合があります。

6. MySQL Installer を使用する場合は、セクション2.3.3「MySQL Installer を使用した MySQL の Microsoft Windows へのインストール」 の説明に従って起動します。
7. MySQL を Zip アーカイブからアップグレードする場合は、アーカイブを抽出します。既存の MySQL のインストールを上書きする (通常 C:\mysql にあります) か、あるいは C:\mysql5 などの異なるディレクトリにインストールします。既存のインストールの上書きをお勧めしています。しかし、(はじめてインストールする場

合に対して) アップグレードの場合は、現在のデータファイルが置換されるのを避けるため、既存の MySQL インストールからデータディレクトリを削除しなければなりません。そのためには、次の手順に従います。

a. Zip アーカイブを、現在の MySQL インストールとは別の場所に解凍します。

b. データディレクトリを削除します。

c. Zip アーカイブを再圧縮します。

d. 変更した Zip アーカイブを既存のインストールの上に解凍します。

別の方法

a. Zip アーカイブを、現在の MySQL インストールとは別の場所に解凍します。

b. データディレクトリを削除します。

c. データディレクトリを、現在の MySQL インストールから今削除したデータディレクトリに移動します。

d. 現在の MySQL インストールを削除します。

e. 解凍したインストールを今削除したインストールの場所に移動します。

8. MySQL を Windows のサービスとして使用していて、この手順で以前にサービスを削除した場合は、サービスを再インストールします。(セクション2.3.5.7「Windows のサービスとして MySQL を起動する」 を参照してください。)

9. サーバーを再起動します。MySQL をサービスとして使用している場合は NET START MySQL などを使用するか、あるいは mysqld を直接実行します。

10. 管理者として mysql_upgrade を実行してテーブルをチェックし、必要に応じて修復を試み、付与テーブルが変更されている場合は更新して新機能を利用できるようにします。セクション4.4.7「mysql_upgrade — MySQL テーブルのチェックとアップグレード」を参照してください。

11. エラーが発生する場合は セクション2.3.6「Microsoft Windows MySQL Server インストールのトラブルシューティング」 を参照します。

2.3.8 Windows でのインストール後の手順

後述のタスクのほとんどを実行する次のような GUI ツールがあります。

- MySQL Installer: MySQL 製品のインストールとアップグレードに使用されます。
- MySQL Workbench: MySQL Server の管理および SQL クエリーの編集を行います。
- MySQL Notifier: MySQL Server の起動、停止、または再起動を行い、ステータスをモニターします。
- MySQL for Excel: Microsoft Excel で MySQL データを編集します。

Windows では、データディレクトリおよび付与テーブルを作成する必要はありません。MySQL の Windows の配布には、データディレクトリの mysql データベースに初期化されたアカウントセットの付与テーブルが含まれています。パスワードに関しては、MySQL を MySQL Installer を使用してインストールした場合は、すでにパスワードをアカウントに割り当てた可能性があります。(セクション2.3.3「MySQL Installer を使用した MySQL の Microsoft Windows へのインストール」 を参照してください)。そうでない場合には、セクション2.10.2「最初の MySQL アカウントのセキュリティー設定」 にあるパスワード割り当て手順を使用します。

パスワードを設定する前にクライアントプログラムを実行して、サーバーに接続でき、サーバーが適切に動作していることを確認するとよいでしょう。サーバーが稼働していること (セクション2.3.5.4「サーバーをはじめて起動する」 参照) を確認してから、次のコマンドを発行してサーバーから情報を取り出せることを確認します。コマンド行で、C:\mysql\bin とは異なるディレクトリを指定する必要がある場合があります。MySQL を MySQL Installer を使用してインストールした場合は、デフォルトディレクトリは C:\Program Files\MySQL\MySQL Server 5.6 で、mysql および mysqlshow クライアントプログラムはC:\Program Files\MySQL\MySQL Server 5.6\bin にあります。詳細については、セクション2.3.3「MySQL Installer を使用した MySQL の Microsoft Windows へのインストール」 を参照してください。

どのようなデータベースが存在するかを表示するには、mysqlshow を使用します。

```
C:\> C:\mysql\bin\mysqlshow
```

```
+--------------------+
|    Databases       |
+--------------------+
| information_schema |
| mysql              |
| test               |
+--------------------+
```

インストール済みのデータベースのリストは異なる場合がありますが、最低でも mysql および information_schema を含みます。多くの場合、test データベースも自動的にインストールされます。

前述のコマンド (および mysql などのその他の MySQL プログラムのコマンド) は、正しい MySQL アカウントが存在しないと機能しない場合があります。たとえば、プログラムがエラーで終了する場合やすべてのデータベースを表示できない場合があります。MySQL を MySQL Installer を使用してインストールした場合は、指定したパスワードで root ユーザーが自動的に作成されています。この場合、-u root および -p オプションを使用してください。(MySQL 初期アカウントをすでにセキュアにした場合も、-u root および -p オプションを使用する必要があります)。-p を使用すると、root パスワードを求められます。例:

```
C:\> C:\mysql\bin\mysqlshow -u root -p
Enter password: (enter root password here)
+--------------------+
|    Databases       |
+--------------------+
| information_schema |
| mysql              |
| test               |
+--------------------+
```

データベース名を指定すると、mysqlshow はそのデータベース内のテーブルのリストを表示します。

```
C:\> C:\mysql\bin\mysqlshow mysql
Database: mysql
+---------------------------+
|         Tables            |
+---------------------------+
| columns_priv              |
| db                        |
| event                     |
| func                      |
| help_category             |
| help_keyword              |
| help_relation             |
| help_topic                |
| host                      |
| plugin                    |
| proc                      |
| procs_priv                |
| servers                   |
| tables_priv               |
| time_zone                 |
| time_zone_leap_second     |
| time_zone_name            |
| time_zone_transition      |
| time_zone_transition_type |
| user                      |
+---------------------------+
```

mysql プログラムを使用して、mysql データベース内のテーブルから情報を選択します。

```
C:\> C:\mysql\bin\mysql -e "SELECT Host,Db,User FROM mysql.db"
+------+--------+------+
| host | db     | user |
+------+--------+------+
| %    | test   |      |
| %    | test_% |      |
+------+--------+------+
```

mysqlshow および mysql の詳細は、セクション4.5.6「mysqlshow — データベース、テーブル、およびカラム情報の表示」 および セクション4.5.1「mysql — MySQL コマンド行ツール」 を参照してください。

サービスをサポートする Windows のバージョンを実行している場合は、MySQL Server を Windows の起動時に自動的に実行するようにセットアップできます。セクション2.3.5.7「Windows のサービスとして MySQL を起動する」を参照してください。

2.4 OS X に MySQL をインストールする

MySQL Server がサポートする OS X のバージョンのリストについては、http://www.mysql.com/support/supportedplatforms/database.htmlを参照してください。

MySQL for OS X は複数の形式で利用可能です。

- ネイティブパッケージインストーラ形式。これはネイティブ OS X インストーラ (DMG) を使用して、MySQL のインストール全体をガイドします。詳細は、セクション2.4.2「ネイティブパッケージを使用して OS X に MySQL をインストールする」を参照してください。OS X のパッケージインストーラを使用できます。インストールの実行に使用するユーザーは、管理者権限を持っていなければなりません。

- tar パッケージ形式。これは Unix の tar コマンドと gzip コマンドを使用してパッケージ化されたファイルを使用します。この方法を使用するには、ターミナルウィンドウを開く必要があります。この方法を使用して、MySQL Server をどこにでもインストールできるため、この方法を使用する場合、管理者権限は必要ありません。この方法の使用については、tarball の使用に関する一般的な説明セクション2.2「一般的なバイナリを使用した MySQL の Unix/Linux へのインストール」を使用できます。

  Package Installer には、コアインストールに加えてセクション2.4.4「MySQL 起動アイテムのインストール」およびセクション2.4.5「MySQL Preference Pane のインストールと使用」が含まれます。いずれもインストールの管理が簡潔になります。

OS X での MySQL の使用に関する追加情報は、セクション2.4.1「OS X への MySQL のインストールに関する一般的な注記」を参照してください。

2.4.1 OS X への MySQL のインストールに関する一般的な注記

次の問題と注記に留意してください。

- 起動アイテムは、起動デーモンを優先するため OS X 10.4 で非推奨になりました。OS X 10.10 (Yosemite) では起動アイテムは機能しません。このため、起動アイテムよりも起動デーモンを使用することが推奨されます。

- MySQL ディレクトリおよびデータを所有する特定の mysql ユーザーの作成が必要となる (またはしたほうがよい) 場合があります。これは、 ディレクトリユーティリティーを通じて実行可能であり、mysql ユーザーがすでに存在している必要があります。シングルユーザーモードで使用するには、_mysql (アンダースコアのプリフィクスに注意してください) のエントリが /etc/passwd システムファイルにすでに存在している必要があります。

- MySQL の起動時に「insecure startup item disabled」エラーが生じる場合は、次の手順に従ってください。パス名はシステムに合わせて適切に調整してください。

  1. 次のコマンドを使用して mysql.script を変更します (1 行に入力します)。

     ```
     shell> sudo /Applications/TextEdit.app/Contents/MacOS/TextEdit
      /usr/local/mysql/support-files/mysql.server
     ```

  2. basedir の値を定義しているオプションファイルを探し、次の行を含むように変更します。

     ```
     basedir=/usr/local/mysql
     datadir=/usr/local/mysql/data
     ```

     /Library/StartupItems/MySQLCOM/ ディレクトリで、グループ ID を次のように staff から wheel に変更します。

     ```
     shell> sudo chgrp wheel MySQLCOM StartupParameters.plist
     ```

  3. 「システム環境設定」または Terminal.app からサーバーを起動します。

- MySQL Package Installer は、MySQL のコンテンツをバージョンとプラットフォームに固有のディレクトリにインストールするため、これを利用して、アップグレードしてバージョン間でデータベースの移行を実行できます。data ディレクトリを古いバージョンから新しいバージョンにコピーするか、あるいは代替の datadir 値を指定してデータディレクトリの場所を設定することが必要になります。デフォルトでは、MySQL ディレクトリは /usr/local/ の下にインストールされます。

- シェルのリソースファイルにエイリアスを追加すると、コマンド行から mysql および mysqladmin などのよく使用するプログラムに容易にアクセスできます。bash での構文は次のようになります。

  ```
  alias mysql=/usr/local/mysql/bin/mysql
  alias mysqladmin=/usr/local/mysql/bin/mysqladmin
  ```

  tcsh では次を使用します。

```
alias mysql /usr/local/mysql/bin/mysql
alias mysqladmin /usr/local/mysql/bin/mysqladmin
```

/usr/local/mysql/bin to your PATH 環境変数を追加するとさらによいでしょう。これを行うには、シェルの適切な起動ファイルを変更します。詳細については、セクション4.2.1「MySQL プログラムの起動」を参照してください。

- 古いインストールからの MySQL データベースファイルのコピーが終了して新しいサーバーの起動が完了したら、ディスクスペースを節約するために古いインストールファイルを削除するとよいでしょう。さらに、/Library/Receipts/mysql-VERSION.pkg にある、古いバージョンのパッケージを入れたディレクトリも削除する必要があります。
- OS X 10.7 より前では、MySQL Server は OS X Server にバンドルされていました。

2.4.2 ネイティブパッケージを使用して OS X に MySQL をインストールする

パッケージは、ディスクイメージ (.dmg) ファイルにあり、最初に Finder でそのアイコンをダブルクリックしてマウントする必要があります。すると、イメージがマウントされ、そのコンテンツが表示されます。

> **注記**
>
> インストールに進む前に、MySQL Manager Application (OS X Server 上で) またはコマンド行で mysqladmin shutdown を使用して、実行中の MySQL Server インスタンスを必ずすべて停止してください。

パッケージバージョンからインストールする場合、MySQL Preference Pane もインストールできます。これにより、MySQL Server の起動と実行を「システム環境設定」から管理できるようになります。詳細については、セクション2.4.5「MySQL Preference Pane のインストールと使用」を参照してください。

Package Installer を使用してインストールする場合、ファイルは /usr/local 内の、インストールするバージョンとプラットフォームに一致する名前のディレクトリにインストールされます。たとえば、インストーラファイル mysql-5.6-osx10.8-x86_64.dmg は、MySQL を /usr/local/mysql-5.6-osx10.8-x86_64/ にインストールします。次の表は、インストールディレクトリのレイアウトを示しています。

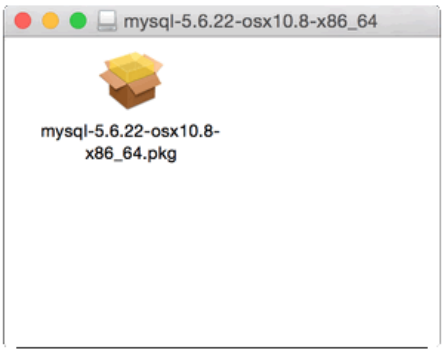
表 2.5 OS X 上での MySQL のインストールレイアウト

ディレクトリ	ディレクトリの内容
bin	クライアントプログラムおよび mysqld サーバー
data	ログファイル、データベース
docs	リリースノートおよびビルド情報などのヘルパードキュメント
include	インクルード (ヘッダー) ファイル
lib	ライブラリ
man	Unix マニュアルページ
mysql-test	MySQL テスト項目
scripts	mysql_install_db
share	エラーメッセージ、サンプル構成ファイル、データベースインストールのための SQL を含む種々のサポートファイル
sql-bench	ベンチマーク
support-files	スクリプトとサンプル構成ファイル
/tmp/mysql.sock	MySQL Unix ソケットの場所

Package Installer のプロセス中に、/usr/local/mysql から、インストール中に作成されたバージョン/プラットフォーム固有のディレクトリへのシンボリックリンクが自動的に作成されます。

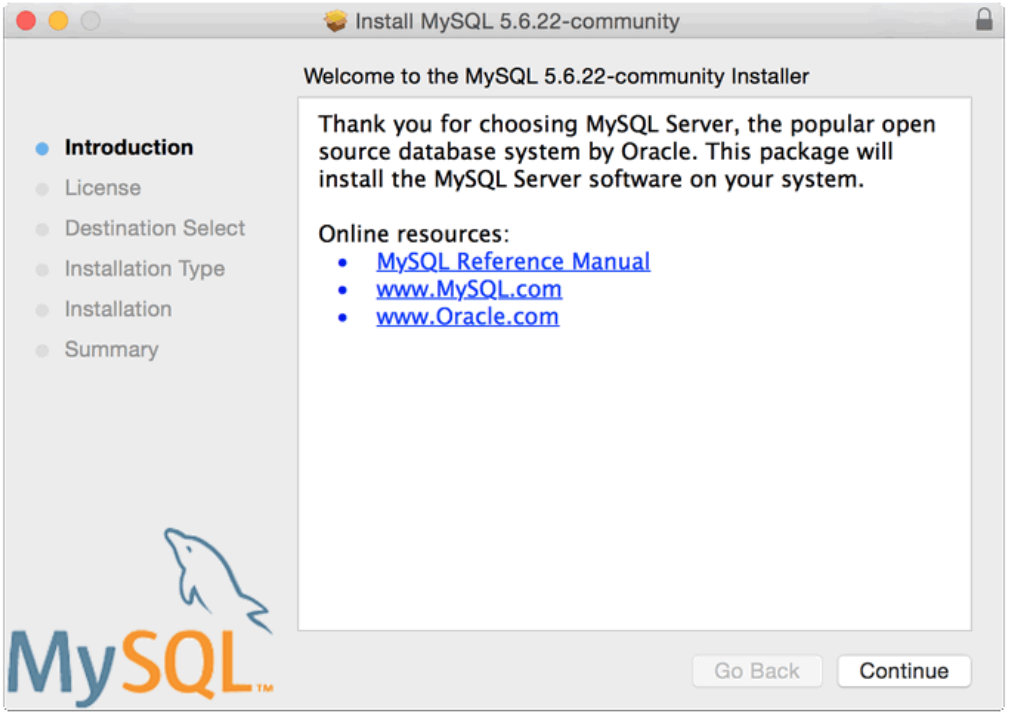
1. MySQL Package Installer をダウンロードして開きます。これは、メインの MySQL インストールパッケージファイルを含むディスクイメージ (.dmg) にあります。ディスクイメージをダブルクリックして開きます。

図 2.40 MySQL Package Installer: DMG のコンテンツ



2. MySQL Installer パッケージをダブルクリックします。ダウンロードした MySQL のバージョンに従って命名されます。たとえば、MySQL Server 5.6.23 をダウンロードした場合は、mysql-5.6.23-osx-10.8-x86_64.pkg をダブルクリックします。
3. インストーラのオープニングダイアログが表示されます。「Continue」をクリックしてインストールを開始します。

図 2.41 MySQL Package Installer: Introduction



4. MySQL のコミュニティーバージョンをダウンロードした場合は、関連する GNU General Public License のコピーが表示されます。「Continue」、「Agree」の順にクリックして続行します。

5. 「Installation Type」ページから、「Install」をクリックしてすべてデフォルトを使用してインストールウィザードを実行するか、「Customize」をクリックしてインストールするコンポーネントを変更するか (MySQL Server、Startup Item、Preference Pane は、デフォルトではすべて有効です)、または「Change Installation

Location」をクリックして、インストールのタイプをすべてのユーザー、インストーラを実行しているユーザーのみ、またはカスタム場所の定義に変更できます。

図 2.42 MySQL Package Installer: Installation Type

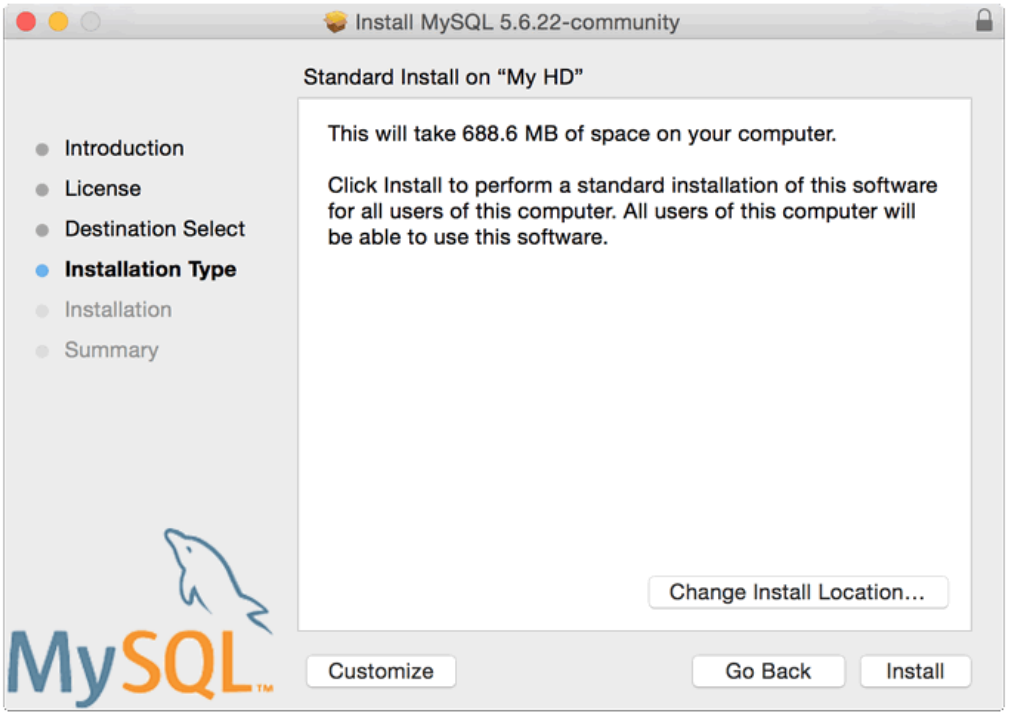
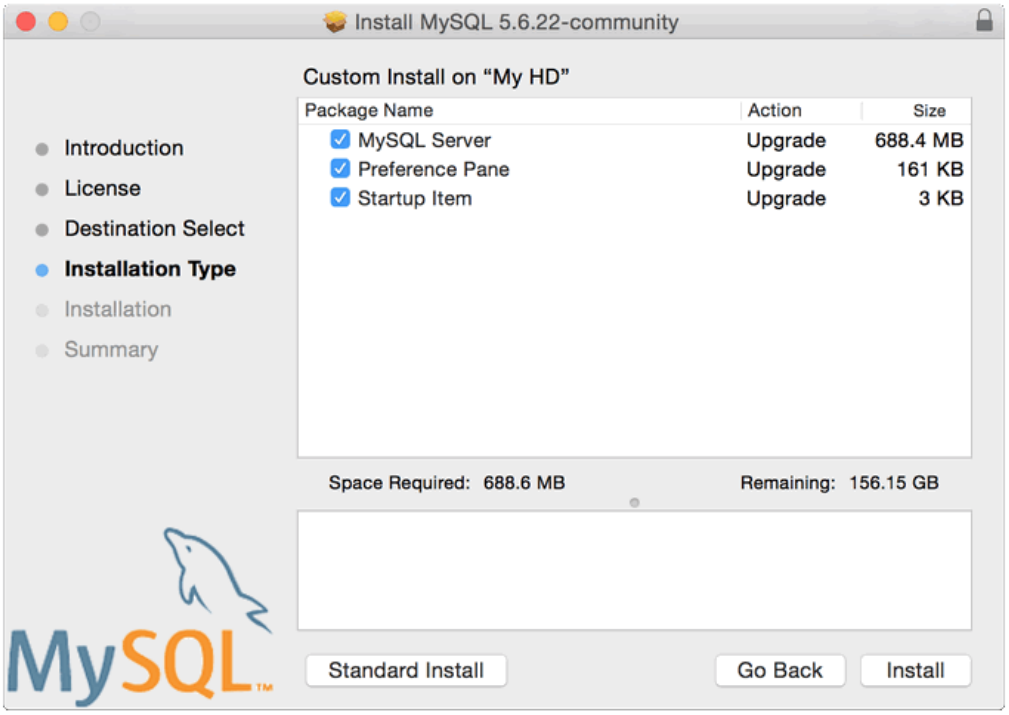


図 2.43 MySQL Package Installer: Destination Select (インストール場所の変更)

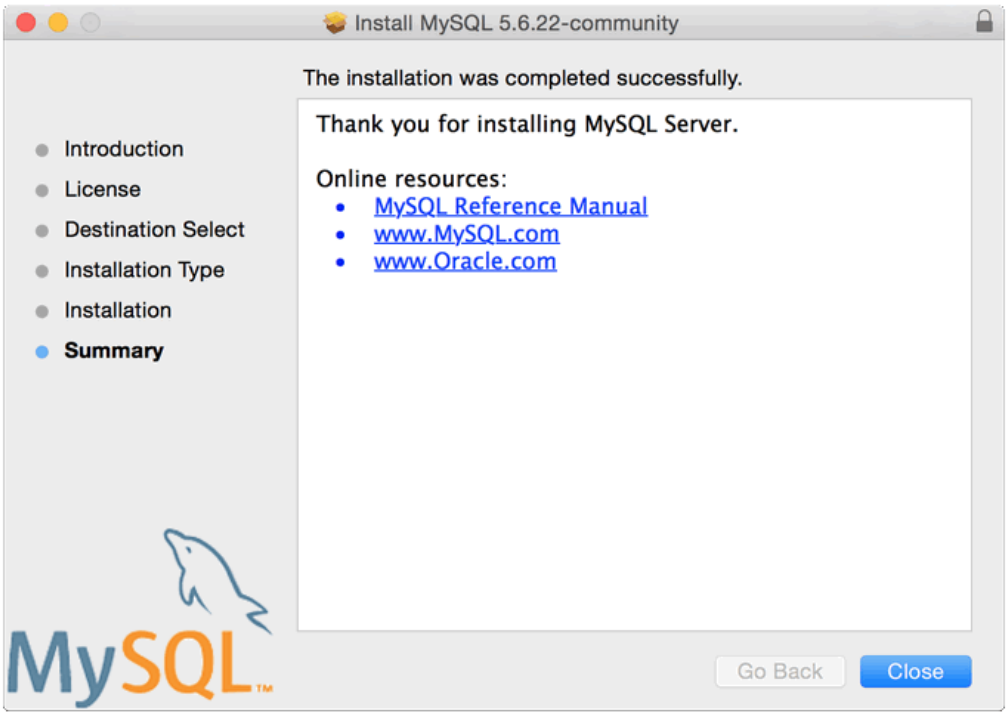
図 2.44 MySQL Package Installer: カスタマイズ



6. 「Install」をクリックしてインストールプロセスを開始します。

7. インストールが正常に完了すると、「Install Succeeded」メッセージが短いサマリーとともに表示されます。ここで、ウィザードを「Close」して MySQL Server の使用を開始します。

図 2.45 MySQL Package Installer: Summary



利便性のため、起動デーモンと Preference Pane をインストールしてもよいでしょう。セクション2.4.3「MySQL 起動デーモンのインストール」、セクション2.4.5「MySQL Preference Pane のインストールと使用」を参照してください。

2.4.3 MySQL 起動デーモンのインストール

OS X は、起動デーモンを使用して、MySQL などのプロセスやアプリケーションの起動、停止、および管理を自動的に行います。OS X では、起動アイテムよりも起動デーモンを使用することが推奨されます。

> **注記**
>
> 起動アイテムは、起動デーモンを優先するため OS X 10.4 で非推奨になりました。OS X 10.10 (Yosemite) では起動アイテムは機能しません。このため、起動アイテムよりも起動デーモンを使用することが推奨されます。

MySQL を起動する launchd サンプルファイルを次に示します。

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE plist PUBLIC "-//Apple//DTD PLIST 1.0//EN"
"http://www.apple.com/DTDs/PropertyList-1.0.dtd">
<plist version="1.0">
  <dict>
    <key>KeepAlive</key>
    <true/>
    <key>Label</key>
    <string>com.mysql.mysqld</string>
    <key>ProgramArguments</key>
    <array>
    <string>/usr/local/mysql/bin/mysqld_safe</string>
    <string>--user=mysql</string>
    </array>
```

```
  </dict>
</plist>
```

ProgramArguments 配列はシステムに合わせて調整します。たとえば mysqld_safe へのパスが異なる場合があります。適切に調整したら、次を行います。

- XML を /Library/LaunchDaemons/com.mysql.mysql.plist という名前のファイルとして保存します。
- ファイルアクセス許可を、Apple 推奨の所有者「root」、所有グループ「wheel」、ファイルアクセス許可「644」を使用して調整します。

```
shell> sudo chown root:wheel /Library/LaunchDaemons/com.mysql.mysql.plist
shell> sudo chmod 644 /Library/LaunchDaemons/com.mysql.mysql.plist
```

- この新しい MySQL サービスを有効にします。

```
shell> sudo launchctl load -w /Library/LaunchDaemons/com.mysql.mysql.plist
```

MySQL デーモンが実行され、システムがリブートされると自動的に起動します。

2.4.4 MySQL 起動アイテムのインストール

MySQL インストールパッケージには、MySQL の自動起動および停止に使用できる起動アイテムが含まれています。

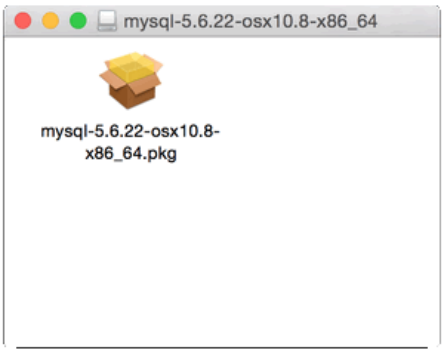
> **重要**
>
> 起動アイテムは、起動デーモンを優先するため非推奨になりました。追加情報についてはセクション2.4.3「MySQL 起動デーモンのインストール」を参照してください。

MySQL 起動アイテムをインストールするには:

1. MySQL Package Installer をダウンロードして開きます。これは、メインの MySQL インストールパッケージを含むディスクイメージ (.dmg) にあります。

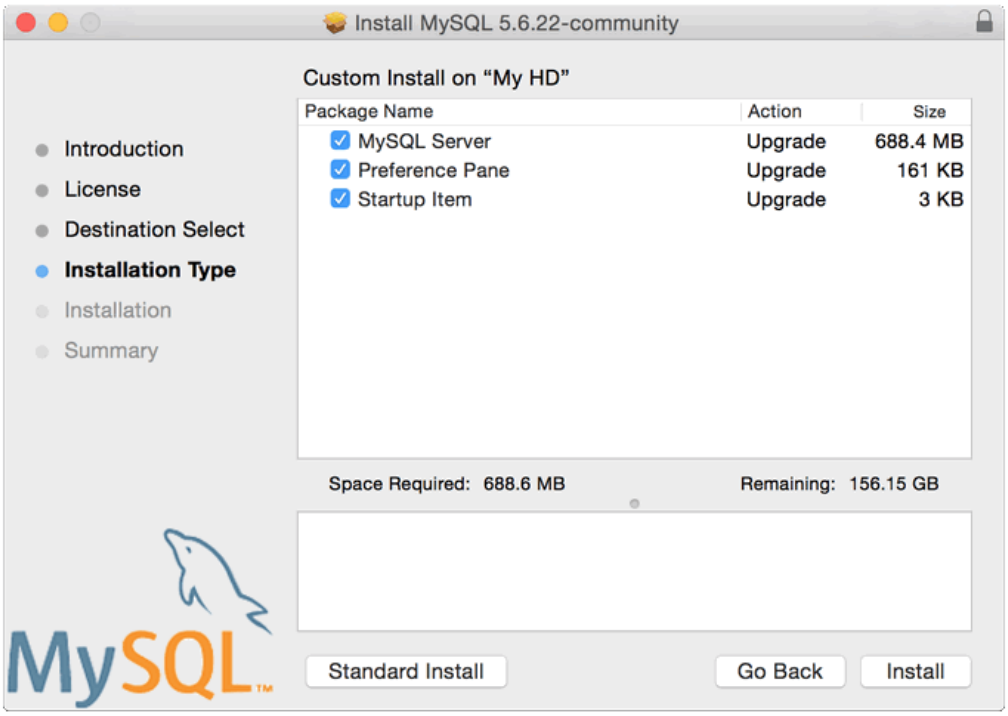
   > **注記**
   >
   > 以前は、OS X パッケージには個別の MySQLStartupItem.pkg および MySQL.prefPane ファイルが含まれていました。それ以降、これらはメインパッケージファイルにマージされています。

   図 2.46 MySQL Package Installer: DMG のコンテンツ

2. MySQL Server のインストールプロセスを、セクション2.4.2「ネイティブパッケージを使用して OS X に MySQL をインストールする」のドキュメントの記述に従って実行します。
3. 「Installation Type」の手順で、「Customize」をクリックします。「Startup Item」オプションがそこに表示され、デフォルトで有効です。

図 2.47 OS X の MySQL Installer: カスタマイズ



4. MySQL Server インストールプロセスを完了します。

MySQL の起動アイテムは /Library/StartupItems/MySQLCOM にインストールされます。起動アイテムのインストールにより、変数 MYSQLCOM=-YES- がシステム構成ファイル /etc/hostconfig に追加されます。MySQL の自動起動を無効にする場合は、この変数を MYSQLCOM=-NO- に変更します。

> **注記**
>
> MySQL Preference Pane から 「Automatically Start MySQL Server on Startup」 の選択を解除すると、MYSQLCOM 変数が -NO-. に設定されます。

インストール後、MySQL Preference Pane から (推奨)、または次のコマンドをターミナルウィンドウで実行することにより、MySQL Server を起動および停止できます。これらのタスクを実行するには管理者権限が必要で、パスワードを要求される場合があります。

起動アイテムをインストールした場合は、次のコマンドを使用してサーバーを開始します。

```
shell> sudo /Library/StartupItems/MySQLCOM/MySQLCOM start
```

起動アイテムをインストールした場合は、次のコマンドを使用してサーバーを停止します。

```
shell> sudo /Library/StartupItems/MySQLCOM/MySQLCOM stop
```

2.4.5 MySQL Preference Pane のインストールと使用

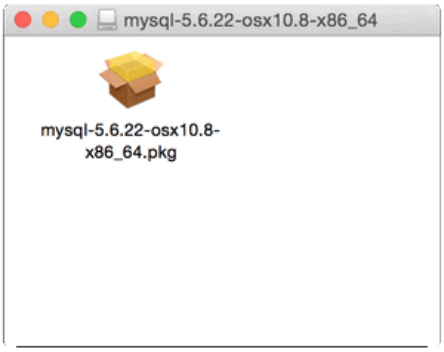
MySQL Installation Package には、MySQL インストールのブート中に MySQL の起動、停止、および自動起動の制御を可能にする MySQL Preference Pane が含まれます。

MySQL Preference Pane をインストールするには:

1. MySQL Package Installer をダウンロードして開きます。これは、メインの MySQL インストールパッケージを含むディスクイメージ (.dmg) にあります。

> **注記**
>
> 以前は、OS X パッケージには個別の MySQLStartupItem.pkg および MySQL.prefPane ファイルが含まれていました。それ以降、これらはメインパッケージファイルにマージされています。

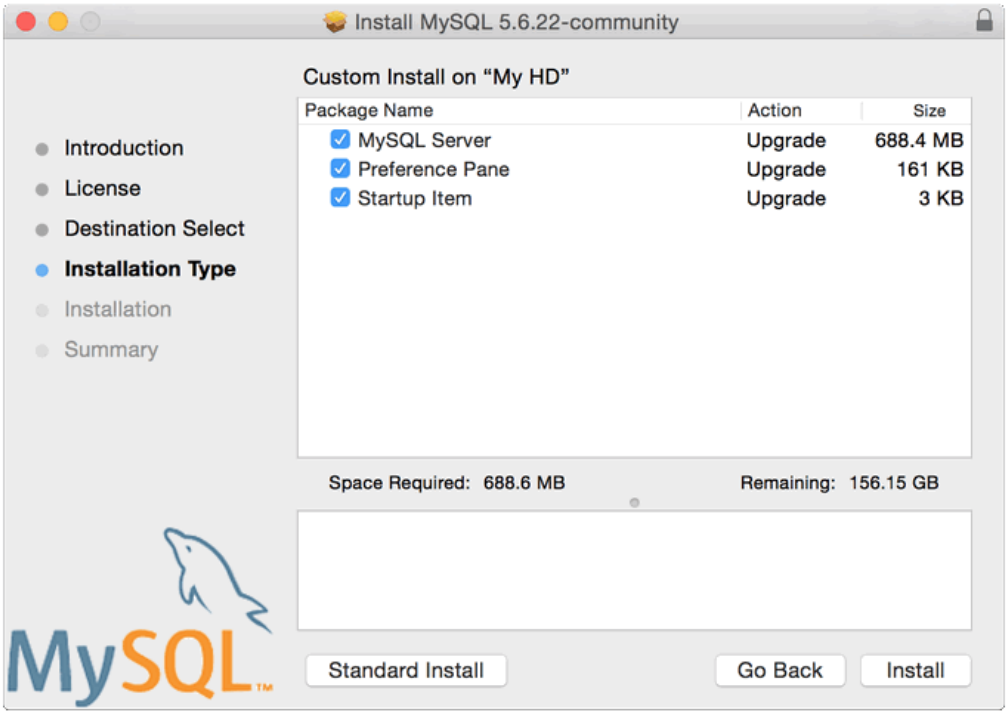
図 2.48 MySQL Package Installer: DMG のコンテンツ



2. MySQL Server のインストールプロセスを、セクション2.4.2「ネイティブパッケージを使用して OS X に MySQL をインストールする」のドキュメントの記述に従って実行します。

3. 「Installation Type」の手順で、「Customize」をクリックします。「Preference Pane」オプションがそこに表示され、デフォルトで有効です。

図 2.49 OS X の MySQL Installer: カスタマイズ

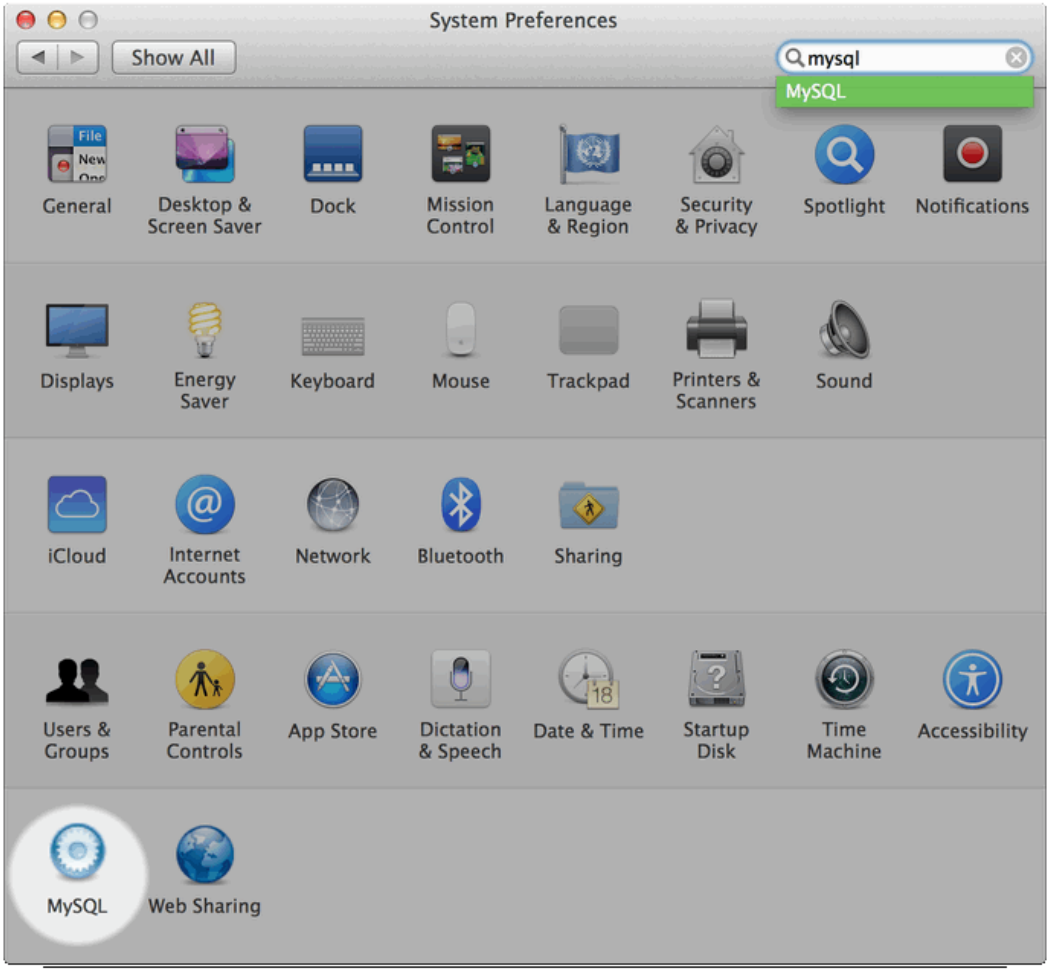


4. MySQL Server インストールプロセスを完了します。

> **注記**
>
> MySQL Preference Pane は、デフォルトの場所にインストールされた MySQL パッケージインストールからインストールされた MySQL インストールのみを起動および停止します。

MySQL Preference Pane がインストールされると、Preference Pane を使用して MySQL Server インスタンスを制御できます。Preference Pane を使用するには、「システム環境設定...」を Apple メニューから開きます。プリファレンスペインのリストの下のセクションにある MySQL 口ゴをクリックして MySQL Preference Pane を選択します。

図 2.50 MySQL Preference Pane: 場所



MySQL Preference Pane は、MySQL Server の現在のステータスを表示します。サーバーが実行していない場合は (赤で) 「stopped」 、サーバーがすでに起動している場合は (緑で) 「running」 と表示します。Preference Pane は、MySQL Server が自動的に起動するように設定されているかどうかに関する現在の設定も表示します。

- MySQL Server を Preference Pane を使用して起動するには:

  「Start MySQL Server」をクリックします。MySQL Server を起動するために、管理者権限を持つユーザーのユーザー名とパスワードを要求される場合があります。

- MySQL Server を Preference Pane を使用して停止するには:

  「Stop MySQL Server」をクリックします。MySQL Server を停止するために、管理者権限を持つユーザーのユーザー名とパスワードを要求される場合があります。

- システムのブート時に自動的に MySQL Server を起動するには:

  「Automatically Start MySQL Server on Startup」の隣のチェックボックスにチェックを付けます。

- システムブート時の MySQL Server の自動起動を無効にするには:

  「Automatically Start MySQL Server on Startup」の隣のチェックボックスのチェックをオフにします。

設定が完了したら、「システム環境設定...」ウィンドウを閉じます。

2.5 Linux に MySQL をインストールする

Linux は、MySQL をインストールするための何種類かのソリューションをサポートします。オラクルからの配布のいずれかを使用することを推奨します。いくつかのインストール方法が使用可能です。

- MySQL Yum リポジトリ を使用して、Yum でインストールします。詳細は、セクション2.5.1「MySQL Yum リポジトリを使用して MySQL を Linux にインストールする」を参照してください。
- MySQL APT リポジトリ を使用して、APT でインストールします。詳細は、セクション2.5.3「MySQL APT リポジトリを使用して MySQL を Linux にインストールする」を参照してください。
- MySQL SLES リポジトリ を使用して、Zypper でインストールします。詳細は、セクション2.5.4「MySQL SLES リポジトリを使用して MySQL を Linux にインストールする」を参照してください。
- プリコンパイル済み RPM パッケージを使用してインストールします。詳細は、セクション2.5.5「RPM パッケージを使用して MySQL を Linux にインストールする」を参照してください。
- プリコンパイル済み Debian パッケージを使用してインストールします。詳細は、セクション2.5.6「オラクルからの Debian パッケージを使用して MySQL を Linux にインストールする」を参照してください。
- .tar.gz 形式の一般的なバイナリパッケージからインストールします。詳細は、セクション2.2「一般的なバイナリを使用した MySQL の Unix/Linux へのインストール」を参照してください。
- オラクルの Unbreakable Linux Network (ULN) を使用してインストールします。詳細は、セクション 2.6「Unbreakable Linux Network (ULN) を使用した MySQL のインストール」を参照してください。
- MySQL をソース配布から抽出してコンパイルします。詳細な手順については、セクション2.9「ソースから MySQL をインストールする」を参照してください。

代わりにシステムのパッケージマネージャーを使用して、Linux 配布のネイティブソフトウェアリポジトリからのパッケージで、MySQL を自動的にダウンロードしてインストールできます。これらのネイティブパッケージは多くの場合、使用可能な最新のリリースから数バージョン遅れています。また、開発マイルストーンリリース (DMR) は、通常ネイティブリポジトリで使用可能にはならないため、これらをインストールすることも通常できません。ネイティブパッケージインストーラの使用の詳細は、セクション2.5.7「ネイティブソフトウェアリポジトリから MySQL を Linux にインストールする」を参照してください。

> 注記
>
> 多くの Linux インストールでは、マシンが起動するときに自動的に MySQL を起動するように設定するとよいでしょう。多くのネイティブパッケージインストールではこの操作は自動的に行われますが、ソース、バイナリ、および RPM ソリューションでは個別の設定が必要になる場合があります。必要なスクリプト mysql.server は、MySQL インストールディレクトリの下の support-files ディレクトリ、または MySQL ソースツリーにあります。自動的に起動およびシャットダウンするには、それを /etc/init.d/mysql としてインストールします。セクション2.10.1.2「MySQL を自動的に起動および停止する」を参照してください。

2.5.1 MySQL Yum リポジトリを使用して MySQL を Linux にインストールする

MySQL は、Yum スタイルのソフトウェアリポジトリを次の Linux プラットフォームで提供します。

- EL5、EL6、および EL7 ベースのプラットフォーム (Red Hat Enterprise Linux、Oracle Linux、および CentOS の対応するバージョンなど)
- Fedora 20 および 21

現在、前述のプラットフォームの MySQL Yum リポジトリ は、MySQL Server、クライアント、MySQL Workbench、MySQL Utilities (EL5 ベースのプラットフォームでは利用できません)、Connector/ODBC、および Connector/Python (EL5 ベースのプラットフォームでは利用できません) をインストールするための RPM パッケージを提供しています。

開始する前に

MySQL は、広く使用されるオープンソースのソフトウェアとして、オリジナルの形式または再パッケージされた形式で、さまざまなダウンロードサイトやソフトウェアリポジトリなどのさまざまなソースから、多くのシステムに広くインストールされています。次の説明では、システムには (オラクルまたはそれ以外から配布される) どのバージョンの MySQL もインストールされていないことを前提とします。そうでない場合は、セクション 2.11.1.1「MySQL Yum リポジトリを使用する MySQL のアップグレード」またはセクション2.5.2「サードパーティーの MySQL 配布を MySQL Yum リポジトリを使用して置換する」を参照してください。

MySQL のインストールを最初から実行する手順

MySQL の最新の GA バージョンを MySQL Yum リポジトリでインストールするには、次の手順に従ってください。

MySQL Yum リポジトリの追加

まず、MySQL Yum リポジトリをシステムのリポジトリリストに追加します。この操作は一度だけ必要で、MySQL が提供する RPM をインストールすることで実行できます。次の手順に従います。

a. MySQL Developer Zone の「Download MySQL Yum Repository」ページ (https://dev.mysql.com/downloads/repo/yum/) に移動します。

b. 使用しているプラットフォーム用のリリースパッケージを選択してダウンロードします。

c. ダウンロードしたリリースパッケージを、次のコマンドでインストールします (EL5 ベースのシステムを除きます)。このとき、platform-and-version-specific-package-name をダウンロードした RPM パッケージの名前に変更します。

```
shell> sudo yum localinstall platform-and-version-specific-package-name.rpm
```

EL6 ベースのシステムでは、コマンドは次の形式です。

```
shell> sudo yum localinstall mysql-community-release-el6-{version-number}.noarch.rpm
```

EL7 ベースのシステムの場合:

```
shell> sudo yum localinstall mysql-community-release-el7-{version-number}.noarch.rpm
```

Fedora 20 の場合:

```
shell> sudo yum localinstall mysql-community-release-fc20-{version-number}.noarch.rpm
```

Fedora 21 の場合:

```
shell> sudo yum localinstall mysql-community-release-fc21-{version-number}.noarch.rpm
```

EL5 ベースのシステムでは、代わりに次のコマンドを使用します。

```
shell> sudo rpm -Uvh mysql-community-release-el5-{version-number}.noarch.rpm
```

インストールコマンドにより、MySQL Yum リポジトリがシステムのリポジトリリストに追加され、ソフトウェアパッケージの完全性をチェックするために GnuPG 鍵がダウンロードされます。GnuPG 鍵チェックの詳細は、セクション2.1.4.2「GnuPG を使用した署名確認」を参照してください。

MySQL Yum リポジトリが正常に追加されたことは、次のコマンドでチェックできます。

```
shell> yum repolist enabled | grep "mysql.*-community.*"
```

> **注記**
>
> システムで MySQL Yum リポジトリが有効になると、yum update コマンドによるシステム全体の更新によって、システム上の MySQL パッケージがアップグレードされ、ネイティブのサードパーティーパッケージがある場合は、Yum が MySQL Yum リポジトリ中に代替を検索できればそれらで置換されます。セクション2.11.1.1「MySQL Yum リポジトリを使用する MySQL のアップグレード」を参照してください。また、システムへの影響の可能性に関する議論については、Upgrading to the Shared Client Librariesを参照してください。

リリースシリーズの選択

> **注記**
>
> MySQL Yum リポジトリを使用する場合、インストールにはデフォルトで MySQL の最新の GA リリースが選択されます。それでよい場合は、次の手順のYum での MySQL のインストールに進んでください。

MySQL Yum リポジトリ内では、MySQL Community Server の異なるリリースシリーズが、異なるサブリポジトリにホストされています。最新の GA シリーズ (現在は 5.6) のサブリポジトリは デフォルトで有効であり、

その他のすべてのシリーズ (たとえば、現在まだデベロッパマイルストーンリリース (DMR) ステータスの 5.7 シリーズなど) のサブリポジトリはデフォルトで無効です。次のコマンドを使用して、MySQL Yum リポジトリのすべてのサブリポジトリを表示し、どれが有効でどれが無効かを確認します。

```
shell> yum repolist all | grep mysql
```

最新の GA シリーズの最新のリリースをインストールする場合は、構成は不要です。最新の GA シリーズ以外の特定のシリーズの最新リリースをインストールするには、インストールコマンドを実行する前に、最新の GA シリーズのサブリポジトリを無効にし、その特定のシリーズのサブリポジトリを有効にします。使用しているプラットフォームで yum-config-manager がサポートされている場合は、次のコマンドを発行することで実行できます。このコマンドは、5.6 シリーズのサブリポジトリを無効にし、5.7 シリーズのサブリポジトリを有効にします。

```
shell> sudo yum-config-manager --disable mysql56-community
shell> sudo yum-config-manager --enable mysql57-community-dmr
```

yum-config-manager を使用する以外に、/etc/yum.repos.d/mysql-community.repo ファイルを手動で編集してリリースシリーズを選択することもできます。ファイル内の、リリースシリーズのサブリポジトリの典型的なエントリを次に示します。

```
# Enable to use MySQL 5.6
[mysql56-community]
name=MySQL 5.6 Community Server
baseurl=//repo.mysql.com/yum/mysql-5.6-community/el/5/$basearch/
enabled=1
gpgcheck=1
gpgkey=file:/etc/pki/rpm-gpg/RPM-GPG-KEY-mysql
```

構成したいサブリポジトリのエントリを探し、enabled オプションを編集します。enabled=0 を指定してサブリポジトリを無効にするか、enabled=1 を指定してサブリポジトリを有効にします。たとえば、最新の 5.7 DMR をインストールするには、前述の MySQL 5.6 のサブリポジトリのエントリを enabled=0 に設定し、5.7 シリーズのエントリを enabled=1 にしたことを確認します。

```
# Note: MySQL 5.7 is currently in development. For use at your own risk.
# Please read with sub pages: https://dev.mysql.com/doc/relnotes/mysql/5.7/en/
[mysql57-community-dmr]
name=MySQL 5.7 Community Server Development Milestone Release
baseurl=http://repo.mysql.com/yum/mysql-5.7-community/el/6/$basearch/
enabled=1
gpgcheck=1
gpgkey=file:/etc/pki/rpm-gpg/RPM-GPG-KEY-mysql
```

一度に有効にするサブリポジトリは、 1 つのリリースシリーズのもののみにしてください。複数のリリースシリーズのサブリポジトリが有効になっている場合は、Yum はもっとも新しいシリーズを使用します。

次のコマンドを実行して出力をチェックして、サブリポジトリが正しく有効または無効にされていることを確認します。

```
shell> yum repolist enabled | grep mysql
```

Yum での MySQL のインストール

次のコマンドで MySQL をインストールします。

```
shell> sudo yum install mysql-community-server
```

これにより、MySQL Server (mysql-community-server)、クライアントのパッケージ (mysql-community-client)、クライアントとサーバー用の一般的なエラーメッセージと文字セット (mysql-community-common)、および共有クライアントライブラリ (mysql-community-libs) など、サーバーを実行するために必要なコンポーネントのパッケージがインストールされます。

MySQL サーバーの起動と停止

次のコマンドで MySQL Server を起動します。

```
shell> sudo service mysqld start
```

前述のコマンドの出力例を次に示します。

```
Starting mysqld:[ OK ]
```

次のコマンドで MySQL Server のステータスをチェックできます。

```
shell> sudo service mysqld status
```

前述のコマンドの出力例を次に示します。

```
mysqld (pid 3066) is running.
```

次のコマンドで MySQL Server を停止します。

```
shell> sudo service mysqld stop
```

MySQL インストールのセキュリティー設定

プログラム mysql_secure_installation を使用すると、ルートパスワードの設定、匿名ユーザーの削除など、重要な操作を実行できます。必ず実行して MySQL インストールをセキュアにしてください。

```
shell> mysql_secure_installation
```

設定したルートパスワードを記憶しておくことが重要です。詳細は、セクション 4.4.5「mysql_secure_installation — MySQL インストールのセキュリティー改善」を参照してください。

インストール後の手順については、セクション2.10「インストール後のセットアップとテスト」を参照してください。

> 注記
>
> EL7 ベースのプラットフォームの互換性情報: プラットフォームのネイティブソフトウェアリポジトリからの次の RPM パッケージは、MySQL Server をインストールする MySQL Yum リポジトリからのパッケージとは互換性がありません。MySQL Yum リポジトリを使用して MySQL をインストールすると、これらのパッケージはインストールできなくなります (その逆も同様です)。
>
> - akonadi-mysql
> - ocsinventory

追加の MySQL 製品およびコンポーネントの Yum でのインストール

Yum を使用して、MySQL の個々のコンポーネントのインストールおよび管理を実行できます。これらのコンポーネントの一部は、MySQL Yum リポジトリのサブリポジトリにホストされています。たとえば、MySQL Connectors は MySQL Connectors Community サブリポジトリ、MySQL Workbench は MySQL Tools Community にあります。 次のコマンドを使用して、使用しているプラットフォームで、MySQL Yum リポジトリから使用可能な MySQL コンポーネントのすべてのパッケージをリストできます。

```
shell> sudo yum --disablerepo=\* --enablerepo='mysql*-community*' list available
```

次のコマンドで package-name をパッケージ名に置換して、任意のパッケージをインストールします。

```
shell> sudo yum install package-name
```

たとえば MySQL Workbench をインストールするには:

```
shell> sudo yum install mysql-workbench-community
```

共有クライアントライブラリをインストールするには:

```
shell> sudo yum install mysql-community-libs
```

Yum での MySQL の更新

インストールのほかに、MySQL 製品およびコンポーネントの更新も、MySQL Yum リポジトリを使用して実行できます。詳細は、セクション2.11.1.1「MySQL Yum リポジトリを使用する MySQL のアップグレード」を参照してください。

2.5.2 サードパーティーの MySQL 配布を MySQL Yum リポジトリを使用して置換する

サポートされる Yum ベースのプラットフォーム (リストについてはセクション2.5.1「MySQL Yum リポジトリを使用して MySQL を Linux にインストールする」を参照してください) では、MySQL Yum リポジトリを使用して、サードパーティーの MySQL 配布を MySQL からの最新の GA リリースに置換できます。サードパーティーの MySQL 配布がインストールされた方法によって、手順が異なります。

ネイティブのサードパーティーの MySQL 配布の置換

サードパーティーの MySQL 配布をネイティブソフトウェアリポジトリ (つまり、ご自分の Linux 配布で提供されたソフトウェアリポジトリ) からインストールした場合は、次の手順に従います。

データベースのバックアップ

データの損失を避けるため、MySQL Yum リポジトリを使用して MySQL インストールを置換する前に、必ずデータベースをバックアップしてください。データベースのバックアップの方法については、第7章「バックアップとリカバリ」を参照してください。

MySQL Yum リポジトリの追加

Adding the MySQL Yum Repositoryに記載の説明に従って、MySQL Yum リポジトリをシステムのリポジトリリストに追加します。

Yum アップデートでのネイティブのサードパーティーの配布の置換

設計により、yum update または yum update mysql-server コマンドをシステムで実行すると、MySQL Yum リポジトリはネイティブのサードパーティー MySQL を置換します。

Yum リポジトリを使用して MySQL を更新したあとも、古いバージョンの共有クライアントライブラリを使用してコンパイルされたアプリケーションは機能するはずです。しかし、アプリケーションを再コンパイルして、更新されたライブラリに動的にリンクする場合は、特に考慮するべき点についてUpgrading to the Shared Client Librariesを参照してください。

ネイティブ以外のサードパーティーの MySQL 配布の置換

サードパーティーの MySQL 配布をネイティブでないソフトウェアリポジトリ (つまり、ご自分の Linux 配布で提供されていないソフトウェアリポジトリ) からインストールした場合は、次の手順に従います。

データベースのバックアップ

データの損失を避けるため、MySQL Yum リポジトリを使用して MySQL インストールを置換する前に、必ずデータベースをバックアップしてください。データベースのバックアップの方法については、第7章「バックアップとリカバリ」を参照してください。

Yum による、サードパーティーのネイティブでないリポジトリからの MySQL パッケージの受信を停止

MySQL をインストールするために MySQL Yum リポジトリを使用する前に、システムがサードパーティーのネイティブでない Yum リポジトリから MySQL パッケージを受信するのを停止する必要があります。

たとえば、MariaDB をそのソフトウェアリポジトリからインストールした場合は、次のコマンドを使用して、インストール済みの MariaDB パッケージのリストを取得します。

```
shell> yum list installed mariadb\*
```

コマンドの出力例を次に示します。

```
MariaDB-common.i686             10.0.4-1             @mariadb
MariaDB-compat.i686             10.0.4-1             @mariadb
MariaDB-server.i686             10.0.4-1             @mariadb
```

コマンド出力から、インストール済みのパッケージ (MariaDB-common、MariaDB-compat、および MariaDB-server) およびそのソース (mariadb という名前のネイティブでないソフトウェアリポジトリ) を特定できます。

別の例として、Percona をそのソフトウェアリポジトリからインストールした場合は、次のコマンドを使用して、インストール済みの Percona パッケージのリストを取得します。

```
shell> yum list installed Percona\*
```

コマンドの出力例を次に示します。

```
Percona-Server-client-55.i686     5.5.39-rel36.0.el6         @percona-release-i386
Percona-Server-server-55.i686     5.5.39-rel36.0.el6         @percona-release-i386
Percona-Server-shared-55.i686     5.5.39-rel36.0.el6         @percona-release-i386
percona-release.noarch            0.1-3                      @/percona-release-0.1-3.noarch
```

コマンド出力から、インストール済みのパッケージ (Percona-Server-client、Percona-Server-server、Percona-Server-shared、および percona-release.noarch) およびそのソース (percona-release という名前のネイティブでないソフトウェアリポジトリ) を特定できます。

どのサードパーティーの MySQL フォークをインストールしたかはっきりしない場合は、このコマンドでそれがわかります。インストール済みの RPM パッケージおよびパッケージを提供しているサードパーティーリポジトリがリストされます。

```
shell> yum --disablerepo=\* provides mysql\*
```

次の手順は、Yum がサードパーティーのネイティブでないリポジトリからパッケージを受信するのを停止することです。使用しているプラットフォームで yum-config-manager ユーティリティーがサポートされている場合は、たとえば MariaDB では次のコマンドを使用できます。

```
shell> sudo yum-config-manager --disable mariadb
```

また、Percona では次のコマンドを使用します。

```
shell> sudo yum-config-manager --disable percona-release
```

/etc/yum.repos.d/ ディレクトリにあるリポジトリファイルの 1 つに存在するソフトウェアリポジトリのエントリを削除することで、同じタスクを実行できます。MariaDB での通常のエントリを次に示します。

```
[mariadb] name = MariaDB
 baseurl = [base URL for repository]
 gpgkey = [URL for GPG key]
 gpgcheck =1
```

エントリは通常、MariaDB では /etc/yum.repos.d/MariaDB.repo にあります。ファイルを削除するか、そのファイルから (またはエントリが見つかったファイルから) エントリを削除します。

> **注記**
>
> Yum リポジトリリリースパッケージ (Percona など) を使用して構成されたインストールでは、次の手順 3 の Percona のアンインストールコマンドのように、リリースパッケージ (Percona の場合 percona-release.noarch) を削除する予定である場合は、この手順は不要です。

ネイティブ以外のサードパーティーの MySQL 配布のアンインストール

MySQL Yum リポジトリを使用して MySQL をインストールする前に、ネイティブでないサードパーティーの MySQL 配布をアンインストールする必要があります。前記の手順 2 の MariaDB パッケージの場合、次のコマンドでアンインストールします。

```
shell> sudo yum remove MariaDB-common MariaDB-compat MariaDB-server
```

前記の手順 2 の Percona パッケージの場合:

```
shell> sudo yum remove Percona-Server-client-55 Percona-Server-server-55 \
  Percona-Server-shared-55.i686 percona-release
```

MySQL Yum リポジトリを使用する MySQL のインストール

次に、セクション2.5.1「MySQL Yum リポジトリを使用して MySQL を Linux にインストールする」に示す手順に従って、MySQL を MySQL Yum リポジトリを使用してインストールします。

> **重要**
>
> - サードパーティーの MySQL 配布を MySQL Yum リポジトリからの新しいバージョンの MySQL に置換する場合は、サーバーの起動後に mysql_upgrade を実行して、古いデータとアップグレードされたソフトウェアとの間の非互換性をチェックし、あれば解決します。mysql_upgrade はその他の機能も実行します。詳細は、セクション4.4.7「mysql_upgrade — MySQL テーブルのチェックとアップグレード」を参照してください。
> - EL7 ベースのプラットフォームの場合: Compatibility Information for EL7-based platforms [126]を参照してください。

2.5.3 MySQL APT リポジトリを使用して MySQL を Linux にインストールする

MySQL APT リポジトリは、MySQL Server、クライアント、その他のコンポーネントを次の Linux プラットフォームにインストールして管理するために、deb パッケージを提供します。

- Debian 7.x (「wheezy」)
- Ubuntu 12.04 LTS (「Precise Pangolin」)
- Ubuntu 14.04 LTS (「Trusty Tahr」)
- Ubuntu 14.10 (「Utopic Unicorn」)

MySQL APT リポジトリの使用に関する説明は、A Quick Guide to Using the MySQL APT Repositoryにあります。

> **注記**
>
> MySQL APT リポジトリは現在開発リリースです。ご試用の上フィードバックをお聞かせください。バグまたは不整合にお気づきの場合は、バグデータベースで報告してください。

2.5.4 MySQL SLES リポジトリを使用して MySQL を Linux にインストールする

MySQL SLES リポジトリは、MySQL Server、クライアント、その他のコンポーネントを SUSE Enterprise Linux Server にインストールして管理するために、RPM パッケージを提供します。

MySQL SLES リポジトリの使用に関する説明は、A Quick Guide to Using the MySQL SLES Repositoryにあります。

> **注記**
>
> MySQL SLES リポジトリは現在開発リリースです。ご試用の上フィードバックをお聞かせください。バグまたは不整合にお気づきの場合は、バグデータベースで報告してください。

2.5.5 RPM パッケージを使用して MySQL を Linux にインストールする

> **注記**
>
> MySQL 5.6.11 のインストールまたはアップグレードを行うには、このセクションの最後にある特記事項を必ずお読みください。

glibc を使用する RPM ベースの Linux 配布に MySQL をインストールするために推奨される方法は、MySQL が提供する RPM パッケージを使用することです。RPM パッケージの Community バージョンを取得するには 2 つのソースがあります。

- 次のプラットフォームでは、MySQL ソフトウェアリポジトリから:
  - EL5、EL6、または EL7 ベースのプラットフォームおよび Fedora 20 または 21 では、MySQL Yum リポジトリを使用します (詳細はセクション2.5.1「MySQL Yum リポジトリを使用して MySQL を Linux にインストールする」を参照してください)。
  - SUSE Enterprise Linux Server では、MySQL SLES リポジトリを使用します (詳細はセクション 2.5.4「MySQL SLES リポジトリを使用して MySQL を Linux にインストールする」を参照してください)。

- MySQL Developer Zone の MySQL Downloads ページ から。ここでは、異なるプラットフォームで機能するさまざまな RPM パッケージを提供しています。

このセクションの議論は、MySQL Developer Zone から直接ダウンロードした RPM パッケージのみに適用されます。これらのパッケージで作成されたインストールは、次の表に示すシステムディレクトリのファイルになります。

表 2.6 MySQL Developer Zone からの Linux RPM パッケージの MySQL インストールレイアウト

ディレクトリ	ディレクトリの内容
/usr/bin	クライアントプログラムおよびスクリプト
/usr/sbin	mysqld サーバー
/var/lib/mysql	ログファイル、データベース
/usr/share/info	Info 形式のドキュメント
/usr/share/man	Unix マニュアルページ
/usr/include/mysql	インクルード (ヘッダー) ファイル
/usr/lib/mysql	ライブラリ
/usr/share/mysql	エラーメッセージ、文字セットファイル、サンプル構成ファイル、データベースインストールのための SQL を含む種々のサポートファイル
/usr/share/sql-bench	ベンチマーク

> 注記
>
> MySQL の RPM 配布はほかのベンダーからも提供されています。それらは、弊社によってビルドされたものとは (通信セットアップなどの) 機能や規則が異なる場合があり、それらのインストールにはこのマニュアルの説明が必ずしも適用されないことに注意してください。代わりに、ベンダーの説明書を参照してください。これらの違いのため、弊社がビルドした RPM パッケージは、ほかのベンダーによってビルドされたこのような RPM がインストールされているかどうかをチェックします。その場合、RPM はインストールを実行せず、そのことを説明するメッセージを生成します。
>
> ほかのベンダーからの RPM がすでにインストールされている場合、どのファイルがサーバーに属し、どれがクライアントライブラリに属するかに関するベンダーの表記が、Oracle パッケージで使用される内訳と異なるなどの不一致が生じる場合があります。このような場合は、Oracle RPM を rpm -i でインストールしようとすると、インストールされる RPM 内のファイルが、インストール済みのパッケージ (次の段落では mysql-libs と表されます) と不一致であるというメッセージになります。
>
> 弊社は、MySQL の各リリースに MySQL-shared-compat パッケージを提供します。このパッケージは mysql-libs に代わるものであり、古い MySQL シリーズに置換可能な互換クライアントライブラリを提供します。MySQL-shared-compat は、mysql-libs を無効にするようにセットアップされていますが、rpm は、(-U とは異なり) -i とともに起動された場合は無効パッケージを置換することを明示的に拒否します。rpm -i によるインストールが不一致を生ずるのはこのためです。
>
> MySQL-shared-compat は、ライブラリが別の場所にインストールされるため、mysql-libs とともに安全にインストールできます。したがって、shared-compat を先にインストールしてから、インストールを続行する前に手動で mysql-libs を削除できます。mysql-libs が削除されたあと、ダイナミックリンカーは mysql-libs がクライアントライブラリを配置する場所でそれらを探すのを停止し、MySQL-shared-compat パッケージによって提供されるライブラリが引き継ぎます。
>
> もう 1 つの選択肢は、yum を使用してパッケージをインストールすることです。MySQL リリースのすべての RPM パッケージを含むディレクトリで、yum install MySQL*rpm を実行すると、一度で不一致なくそれらを正しい順序でインストールし、mysql-libs を削除します。

ほとんどの場合、機能する MySQL インストールを取得するためにインストールする必要があるのは MySQL-server および MySQL-client パッケージのみです。その他のパッケージは、標準インストールには必要ありません。

MySQL 5.6.8 では、RPM の (アップグレードではなく) インストール操作は、mysql_install_db をよりセキュアな MySQL インストールを提供する --random-passwords オプションで起動します。--random-passwords を付けて mysql_install_db を呼び出すと、ランダムなパスワードを MySQL root アカウントに割り当て、これらのアカウントについて「パスワードが期限切れである」ことを示すフラグを設定し、匿名ユーザー MySQL アカウントを作成しません。インストール後、サーバーを起動して $HOME/.mysql_secret ファイルに書き込まれたパスワードを使用して root として接続し、新しい root パスワードを割り当てることが必要になります。これを行うまで、root はそれ以外何もできません。これは、使用するすべての root アカウントについて実行する必要があります。パスワードを変更するには、SET PASSWORD ステートメントを (たとえば mysql クライアントとともに) 使用します。mysqladmin または mysql_secure_installation も使用できます。追加情報についてはセクション4.4.3「mysql_install_db — MySQL データディレクトリの初期化」を参照してください。(Unbreakable Linux Network での RPM を使用したインストール操作は mysql_install_db を使用しないため、影響されません。)

> **重要**
>
> MySQL クラスタ用の RPM　MySQL によってビルドされた標準の MySQL Server RPM は、NDBCLUSTER ストレージエンジンをサポートしません。RPM から MySQL Cluster をインストールする方法の詳細は、セクション18.2「MySQL Cluster のインストール」を参照してください。
>
> MySQL Cluster RPM インストールをアップグレードするときは、サーバーおよびクライアントの RPM を含め、インストールしたすべての RPM をアップグレードする必要があります。

もともと複数の RPM パッケージをインストールして作成されたインストールの場合は、アップグレードでは、一部だけでなくすべてのパッケージをアップグレードするのが最善です。たとえば、以前にサーバーおよびクライアントの RPM をインストールした場合は、サーバーの RPM だけをアップグレードすることはしないでください。

RPM インストール時にデータディレクトリが存在する場合は、インストールプロセスは既存のデータを変更しません。これは、たとえば付与テーブル内のアカウントがアカウントのデフォルトセットに初期化されないという効果があります。

MySQL パッケージをインストールしようとして依存関係エラーが生じる場合は (たとえば、error: removing these packages would break dependencies: libmysqlclient.so.10 is needed by ...)、MySQL-shared-compat パッケージもインストールしてください。これには後方互換性のための古いリリースの共有ライブラリが含まれます。

次のリストに示す RPM パッケージが利用できます。ここに示した名前は、.linux_glibc2.5.i386.rpm というサフィクスを使用しますが、特定のパッケージではサフィクスが異なる場合があります。以降で説明します。複数の RPM パッケージをインストールする計画の場合は、RPM バンドル tar ファイルを代わりにダウンロードするとよいでしょう。これには複数の RPM パッケージが含まれるため、個別にダウンロードする必要がありません。

- MySQL-server-VERSION.linux_glibc2.5.i386.rpm

  MySQL Server。別のマシンで動作している MySQL サーバーに接続するのみでなければ、これが必要です。

- MySQL-client-VERSION.linux_glibc2.5.i386.rpm

  標準の MySQL クライアントプログラム。通常は、このパッケージをインストールします。

- MySQL-devel-VERSION.linux_glibc2.5.i386.rpm

  Perl モジュールなど、ほかの MySQL クライアントをコンパイルする場合に必要なライブラリおよびインクルードファイル。C API アプリケーションをコンパイルする予定である場合はこの RPM をインストールします。

- MySQL-shared-VERSION.linux_glibc2.5.i386.rpm

  このパッケージには、MySQL を動的にロードして使用するために、一部の言語およびアプリケーションが必要とする共有ライブラリ (libmysqlclient.so*) が含まれます。それにはシングルスレッドとスレッドセーフのライブラリが含まれます。共有クライアントライブラリに依存する C API アプリケーションをコンパイルして実行する予定である場合はこの RPM をインストールします。

- MySQL-shared-compat-VERSION.linux_glibc2.5.i386.rpm

  このパッケージは古いリリースの共有ライブラリを含みますが、現在のリリースのライブラリは含みません。それにはシングルスレッドとスレッドセーフのライブラリが含まれます。古いバージョンの MySQL にダイナ

ミックリンクするアプリケーションをインストールしていて、ライブラリの依存関係を壊すことなく最新バージョンにアップグレードする場合には、このパッケージをインストールします。

MySQL 5.6.5 では、MySQL-shared-compat RPM パッケージにより、Red Hat が提供する mysql-*-5.1 RPM パッケージのユーザーが、オラクルが提供する MySQL-*-5.5 パッケージに移行できます。MySQL-shared-compat は、後者のパッケージの libmysqlclient.so ファイルを置換して、mysql-libs へのほかのパッケージの依存関係を満たすことによって、Red Hat mysql-libs パッケージを置換します。この変更は、Red Hat (または Red Hat 互換の) RPM パッケージのユーザーのみに影響します。Oracle RPM パッケージのユーザーにとっては何も変更はありません。

- MySQL-embedded-VERSION.linux_glibc2.5.i386.rpm

  組み込み MySQL サーバーライブラリ。

- MySQL-test-VERSION.linux_glibc2.5.i386.rpm

  このパッケージには、MySQL のテストスイートが含まれています。

- MySQL-VERSION.src.rpm

  これにはすべての旧パッケージのソースコードが含まれています。ほかのアーキテクチャー (たとえば、Alpha あるいは SPARC) で RPM をビルドする際にも使用できます。

RPM パッケージ名のサフィクス (VERSION 値に続く) の構文は次のとおりです。

```
.PLATFORM.CPU.rpm
```

PLATFORM および CPU 値は、パッケージのビルド対象になっているシステムのタイプを示します。PLATFORM はプラットフォームを示し、CPU はプロセッサのタイプまたはファミリを示します。

すべてのパッケージは glibc 2.5 にダイナミックリンクされます。PLATFORM 値は、次の表に示すように、パッケージがプラットフォームから独立しているか、または特定のプラットフォームを目的としているかを示します。

表 2.7 MySQL Linux インストールパッケージ

PLATFORM 値	使用目的
linux_glibc25	プラットフォームから独立。glibc 2.5 をサポートする任意の Linux 配布で動作します。
rhel5、rhel6	Red Hat Enterprise Linux 5 または 6
el6、el7	Enterprise Linux 6 または 7
sles10、sles11	SUSE Linux Enterprise Server 10 または 11

MySQL 5.6 では、現在 linux_glibc2.5 パッケージのみ使用可能です。

CPU 値は、パッケージのビルド対象になっているプロセッサのタイプまたはファミリを示します。

表 2.8 Linux 用 MySQL インストールパッケージの CPU 識別子

CPU 値	対象のプロセッサタイプまたはファミリ
i386、i586、i686	Pentium プロセッサ以上、32 ビット
x86_64	64 ビットの x86 プロセッサ
ia64	Itanium (IA-64) プロセッサ

RPM パッケージ (MySQL-server RPM など) のすべてのファイルを表示するには、次のようなコマンドを実行します。

```
shell> rpm -qpl MySQL-server-VERSION.linux_glibc2.5.i386.rpm
```

標準の最低限のインストールには、サーバーとクライアント RPM をインストールします。

```
shell> rpm -i MySQL-server-VERSION.linux_glibc2.5.i386.rpm
shell> rpm -i MySQL-client-VERSION.linux_glibc2.5.i386.rpm
```

クライアントプログラムのみをインストールする場合は、クライアント RPM のみインストールします。

```
shell> rpm -i MySQL-client-VERSION.linux_glibc2.5.i386.rpm
```

RPM には、インストールするパッケージの完全性および信憑性を検証する機能があります。この機能の詳細は、セクション2.1.4「MD5 チェックサムまたは GnuPG を用いたパッケージの完全性の確認」を参照してください。

サーバー RPM はデータを /var/lib/mysql ディレクトリに格納します。RPM はまた、MySQL Server を実行するために使用するユーザー mysql という名前のログインアカウントを作成し (存在しない場合)、サーバーがブート時に自動的に起動するように適切なエントリを /etc/init.d/ に作成します。(これは、以前にインストールを実行し、スタートアップスクリプトに変更を加えた場合、新しい RPM をインストールするときにスクリプトを失わないように、コピーをとる方がよいことを意味します。)システム起動時に MySQL を自動的に起動する方法の詳細は、セクション2.10.1.2「MySQL を自動的に起動および停止する」を参照してください。

MySQL 5.6 では、新規インストール中にサーバーブートスクリプトがインストールされますが、MySQL Server はインストールの最後には起動されません。これは、無人インストール中のサーバーのステータスがわからないためです。

MySQL 5.6 では、RPM パッケージを使用するアップグレードインストール中に、アップグレードが行われたときに MySQL Server が実行中である場合は、MySQL Server は停止され、アップグレードが実行され、MySQL Server が再起動されます。RPM アップグレードの実行時に MySQL Server がすでに実行中ではなかった場合は、インストールの最後に MySQL Server は起動されません。

> 注記
>
> MySQL のコミュニティーバージョンから商用バージョンにアップグレードする場合は (これは実際には、まずコミュニティーバージョンをアンインストールしてから商用バージョンをインストールする必要があります)、アップグレード後に手動でサーバーを再起動する必要があります。

うまくいかない場合には、バイナリのインストールのセクションに詳細情報があります。セクション2.2「一般的なバイナリを使用した MySQL の Unix/Linux へのインストール」を参照してください。

> 注記
>
> MySQL の付与テーブルにリストされているアカウントには、最初はパスワードがありません。サーバーの起動後に、セクション2.10「インストール後のセットアップとテスト」の説明に従って、それらにパスワードを設定する必要があります。

RPM のインストール時に、mysql というユーザーと mysql というグループがシステムに作成されます。これは、useradd、groupadd、および usermod コマンドを使用して行われます。これらのコマンドは適切な管理者権限を必要とします。これは、root によって実行される RPM インストールプロセスによってローカルに管理されるユーザーおよびグループ (/etc/passwd および /etc/group ファイルにリストされています) に必要です。

mysql ユーザーとしてログインすると、MySQL が「Invalid (old?) table or database name」エラーを表示し、.mysqlgui、lost +found、.mysqlgui、.bash_history、.fonts.cache-1、.lesshst、.mysql_history、.profile、.viminfo、および MySQL またはオペレーティングシステムのユーティリティーによって作成された同様のファイルが示されます。これらのエラーメッセージは無視しても、エラーメッセージの原因となっているファイルまたはディレクトリが不要であれば削除しても、安全です。

ローカル以外のユーザー管理 (LDAP、NIS、など) では、管理ツールが追加の認証 (パスワードなど) を要求することがあり、インストールを行うユーザーがその認証情報を提供しない場合は失敗します。失敗しても、RPM インストールは中止されずに正常に行われます。これは意図したものです。失敗した場合は意図した所有権の移動の一部が実行されておらず、システム管理者は、適切なユーザーおよびグループが存在することを手動で確認し、RPM spec ファイルのアクションに従って所有権を手動で移動することが推奨されます。

MySQL 5.6.11 では RPM spec ファイルが更新され、次のような影響があります。

- アップグレードでないインストール (既存の MySQL バージョンはインストールされていない) の場合、MySQL を yum を使用してインストールできます。
- アップグレードの場合、それまでの MySQL インストールがあればクリーンアップする必要があります。実際には、アップデートは古いインストールを削除して新しいものをインストールすることによって実行されます。

追加の詳細は、あとの方で説明します。

MySQL 5.6.11 以降のアップグレードでないインストールの場合、MySQL を yum を使用してインストールできます。

```
shell> yum install MySQL-server-NEWVERSION.linux_glibc2.5.i386.rpm
```

MySQL 5.6.11 以降へのアップグレードの場合、アップグレードは古いインストールを削除して新しいものをインストールすることによって実行されます。そうするには、次の手順に従います。

1. 既存の 5.6.X インストールを削除します。OLDVERSION は削除するバージョンです。

   ```
   shell> rpm -e MySQL-server-OLDVERSION.linux_glibc2.5.i386.rpm
   ```

   この手順をすべてのインストール済み MySQL RPM に対して繰り返します。

2. 新しいバージョンをインストールします。NEWVERSION はインストールするバージョンです。

   ```
   shell> rpm -ivh MySQL-server-NEWVERSION.linux_glibc2.5.i386.rpm
   ```

あるいは、削除とインストールは yum を使用して実行できます。

```
shell> yum remove MySQL-server-OLDVERSION.linux_glibc2.5.i386.rpm
shell> yum install MySQL-server-NEWVERSION.linux_glibc2.5.i386.rpm
```

2.5.6 オラクルからの Debian パッケージを使用して MySQL を Linux にインストールする

オラクルは、MySQL を Debian や Debian のと類似の Linux システムにインストールするための Debian パッケージを提供します。このパッケージは 2 種類のチャネルから入手可能です。

- MySQL APT Repository。Debian 7、Ubuntu 12、および Ubuntu 14 の各プラットフォームをサポートします。詳細は、セクション2.5.3「MySQL APT リポジトリを使用して MySQL を Linux にインストールする」を参照してください。
- MySQL Developer Zone のダウンロード領域。詳細は、セクション2.1.3「MySQL の取得方法」を参照してください。そこで入手可能な Debian パッケージと、インストールの手順についての情報を次に示します。
  - libaio ライブラリがシステムにない場合は、それもインストールする必要がある場合があります。

    ```
    shell> sudo apt-get install libaio1
    ```

  - Debian 7、Ubuntu 12、および Ubuntu 14 の場合:
    - MySQL のさまざまなコンポーネントをインストールするために、MySQL Developer Zone ではさまざまな Debian パッケージが提供されています。推奨される方法は tarball バンドルを使用するものです。これには MySQL の基本的なセットアップに必要なパッケージが含まれます。tarball バンドルは、mysql-server_MVER-DVER_CPU.deb-bundle.tar という形式の名前を持ちます。MVER は MySQL のバージョン、DVER は Linux 配布のバージョンです。次の表に示すように、CPU 値は、パッケージのビルド対象になっているプロセッサのタイプまたはファミリを示します。

      表 2.9 MySQL Debian 7 および Ubuntu インストールパッケージの CPU 識別子

CPU 値	対象のプロセッサタイプまたはファミリ
i386	Pentium プロセッサ以上、32 ビット
amd64	64 ビットの x86 プロセッサ

    - tarball のダウンロード後、次のコマンドでアンパックします。

      ```
      shell> tar -xvf mysql-server_MVER-DVER_CPU.deb-bundle.tar
      ```

    - 一般的には、次のコマンドで tarball からアンパックした deb パッケージをインストールします (サーバーパッケージのインストールに必要な追加の手順は、次の説明を参照してください)。

      ```
      shell> sudo dpkg -i package-name.deb
      ```

      4 つのパッケージをインストールします。

      - データベースの共通ファイル (このパッケージはほかのものより前にインストールします):

```
shell> sudo dpkg -i mysql-common_MVER-DVER_CPU.deb
```

- MySQL Server:

  まず、データベースの共通ファイルのパッケージをインストールし (直前の行頭記号を参照してください)てから、次のコマンドでサーバーインストールを事前構成します。

  ```
  shell> dpkg-preconfigure mysql-community-server_MVER-DVER_CPU.deb
  ```

  次に、2 つのリクエストがあります。

  - MySQL インストールの root ユーザーのパスワードを提供します。

    > **重要**
    >
    > 設定した root パスワードを必ず記憶してください。あとでパスワードを設定するユーザーは、ダイアログボックスの password フィールドをブランクのままにして、「OK」のみを押します。ただし、データベースの root アカウントをパスワードでセキュアにするまで、MySQL Server への匿名でのアクセスが可能なため、プログラム mysql_secure_installation を使用してすぐにパスワードを設定することが非常に重要です。

  - テストデータベースをインストールするかどうかを、「Yes」または「No」で示します。本番環境では、テストデータベースをインストールすることが推奨されます。

  次に、次のコマンドでサーバーパッケージをインストールします。

  ```
  shell> sudo dpkg -i mysql-community-server_MVER-DVER_CPU.deb
  ```

- MySQL クライアント:

  ```
  shell> sudo dpkg -i mysql-community-client_MVER-DVER_CPU.deb
  ```

- MySQL 共有クライアントライブラリ:

  ```
  shell> sudo dpkg -i libmysqlclient18_MVER-DVER_CPU.deb
  ```

ファイルがシステムにインストールされる場所は次のとおりです。

- すべての構成ファイル (my.cnf など) は /etc の下
- すべてのバイナリ、ライブラリ、ヘッダーなどは /usr の下
- データディレクトリは /var の下

- Debian 6 の場合:

  - MySQL Developer Zone から直接ダウンロードされた Debian パッケージファイルは、mysql-MVER-DVER-CPU.deb という形式の名前を持ちます。MVER は MySQL のバージョン、DVER は Debian のバージョンです。次の表に示すように、CPU 値は、パッケージのビルド対象になっているプロセッサのタイプまたはファミリを示します。

    表 2.10 MySQL Debian 6 インストールパッケージの CPU 識別子

CPU 値	対象のプロセッサタイプまたはファミリ
i686	Pentium プロセッサ以上、32 ビット
x86_64	64 ビットの x86 プロセッサ

  - Debian パッケージのダウンロード後、次のコマンドでインストールします。

    ```
    shell> dpkg -i mysql-MVER-DVER-CPU.deb
    ```

    Debian パッケージは、ファイルを /opt/mysql/server-5.6 ディレクトリにインストールします。

    > **注記**
    >
    > MySQL の Debian 配布はほかのベンダーからも提供されています。それらは、オラクルによってビルドされたものとは (通信セットアップなどの) 機能や規則が異なる場合があり、それらのインストールにはこのマニュアルの説明が必ずしも適用されないことに注意してください。代わりに、ベンダーの説明書を参照してください。

2.5.7 ネイティブソフトウェアリポジトリから MySQL を Linux にインストールする

多くの Linux 配布は、ネイティブソフトウェアリポジトリの中に 1 つのバージョンの MySQL Server、クライアントツール、および開発コンポーネントを含み、プラットフォームの標準パッケージ管理システムでインストールできます。このセクションでは、これらのパッケージ管理システムを使用して MySQL をインストールするための基本的な手順を説明します。

> **重要**
>
> ネイティブパッケージは多くの場合、使用可能な最新のリリースから数バージョン遅れています。また、開発マイルストーンリリース (DMR) は、通常ネイティブリポジトリで使用可能にはならないため、これらをインストールすることも通常できません。次に進む前に、セクション2.5「Linux に MySQL をインストールする」に記述されているその他のインストールオプションを確認することが推奨されます。

配布固有の手順を次に示します。

- Red Hat Linux、Fedora、CentOS

> **注記**
>
> EL5、EL6、または EL7 ベースの Linux プラットフォームおよび Fedora 20 または 21 では、プラットフォームのネイティブソフトウェアリポジトリの代わりに MySQL Yum リポジトリを使用して MySQL をインストールできます。詳細は、セクション 2.5.1「MySQL Yum リポジトリを使用して MySQL を Linux にインストールする」 を参照してください。

Red Hat および同様の配布では、MySQL 配布はいくつかの個別のパッケージ (クライアントツール用の mysql、サーバーおよび関連ツール用の mysql-server、およびライブラリ用の mysql-libs) にわかれています。Perl や Python など、異なる言語および環境からの接続性を提供する場合は、ライブラリは必須です。

インストールするには、yum コマンドを使用してインストールするパッケージを指定します。例:

```
root-shell> yum install mysql mysql-server mysql-libs mysql-server
Loaded plugins: presto, refresh-packagekit
Setting up Install Process
Resolving Dependencies
--> Running transaction check
---> Package mysql.x86_64 0:5.1.48-2.fc13 set to be updated
---> Package mysql-libs.x86_64 0:5.1.48-2.fc13 set to be updated
---> Package mysql-server.x86_64 0:5.1.48-2.fc13 set to be updated
--> Processing Dependency: perl-DBD-MySQL for package: mysql-server-5.1.48-2.fc13.x86_64
--> Running transaction check
---> Package perl-DBD-MySQL.x86_64 0:4.017-1.fc13 set to be updated
--> Finished Dependency Resolution

Dependencies Resolved

================================================================================
 Package             Arch         Version              Repository     Size
================================================================================
Installing:
 mysql               x86_64       5.1.48-2.fc13        updates       889 k
 mysql-libs          x86_64       5.1.48-2.fc13        updates       1.2 M
 mysql-server        x86_64       5.1.48-2.fc13        updates       8.1 M
Installing for dependencies:
 perl-DBD-MySQL      x86_64       4.017-1.fc13         updates       136 k

Transaction Summary
================================================================================
Install       4 Package(s)
Upgrade       0 Package(s)

Total download size: 10 M
Installed size: 30 M
Is this ok [y/N]: y
Downloading Packages:
Setting up and reading Presto delta metadata
Processing delta metadata
Package(s) data still to download: 10 M
(1/4): mysql-5.1.48-2.fc13.x86_64.rpm                 | 889 kB     00:04
(2/4): mysql-libs-5.1.48-2.fc13.x86_64.rpm            | 1.2 MB     00:06
```

```
(3/4): mysql-server-5.1.48-2.fc13.x86_64.rpm            | 8.1 MB     00:40
(4/4): perl-DBD-MySQL-4.017-1.fc13.x86_64.rpm           | 136 kB     00:00
--------------------------------------------------------------------------------
Total                                   201 kB/s |  10 MB     00:52
Running rpm_check_debug
Running Transaction Test
Transaction Test Succeeded
Running Transaction
  Installing     : mysql-libs-5.1.48-2.fc13.x86_64                       1/4
  Installing     : mysql-5.1.48-2.fc13.x86_64                            2/4
  Installing     : perl-DBD-MySQL-4.017-1.fc13.x86_64                    3/4
  Installing     : mysql-server-5.1.48-2.fc13.x86_64                     4/4

Installed:
  mysql.x86_64 0:5.1.48-2.fc13           mysql-libs.x86_64 0:5.1.48-2.fc13
  mysql-server.x86_64 0:5.1.48-2.fc13

Dependency Installed:
  perl-DBD-MySQL.x86_64 0:4.017-1.fc13

Complete!
```

MySQL および MySQL Server はすでにインストールされています。サンプル構成ファイルは /etc/my.cnf にインストールされます。サーバーの起動および停止のための init スクリプトは、/etc/init.d/mysqld にインストールされています。MySQL Server の起動には service を使用します。

```
root-shell> service mysqld start
```

ブート中にサーバーを自動的に起動および停止できるようにするには、chkconfig を使用します。

```
root-shell> chkconfig --levels 235 mysqld on
```

これにより、MySQL Server が指定した実行レベルで自動的に起動 (および停止) されます。

データベーステーブルがまだ存在していない場合は、自動的に作成されます。ただし、サーバーで mysql_secure_installation を実行して root のパスワードを設定してください。

- Debian、Ubuntu、Kubuntu

  > **注記**
  >
  > Debian 7、Ubuntu 12、および Ubuntu 14 では、プラットフォームのネイティブソフトウェアリポジトリの代わりに MySQL APT リポジトリを使用して MySQL をインストールできます。詳細は、セクション2.5.3「MySQL APT リポジトリを使用して MySQL を Linux にインストールする」 を参照してください。

  Debian および関連する配布では、ソフトウェアリポジトリに MySQL のパッケージが mysql-client と mysql-server の 2 つあります。それぞれクライアントコンポーネントとサーバーコンポーネントです。希望するバージョンの MySQL を確実にインストールするため、mysql-client-5.1 のように明示的にバージョンを指定するとよいでしょう。

  依存関係を含めてダウンロードしてインストールするには、インストールするパッケージを指定して apt-get コマンドを使用します。

  > **注記**
  >
  > 入手可能な最新バージョンを確実にダウンロードするために、インストールの前に apt-get インデックスファイルを更新してください。

  MySQL パッケージのサンプルインストールは次のようになります (わかりやすくするため一部のセクションは省略しています)。

```
root-shell> apt-get install mysql-client-5.1 mysql-server-5.1
Reading package lists... Done
Building dependency tree
Reading state information... Done
The following packages were automatically installed and are no longer required:
  linux-headers-2.6.28-11 linux-headers-2.6.28-11-generic
Use 'apt-get autoremove' to remove them.
The following extra packages will be installed:
  bsd-mailx libdbd-mysql-perl libdbi-perl libhtml-template-perl
  libmysqlclient15off libmysqlclient16 libnet-daemon-perl libplrpc-perl mailx
  mysql-common postfix
Suggested packages:
```

```
  dbishell libipc-sharedcache-perl tinyca procmail postfix-mysql postfix-pgsql
  postfix-ldap postfix-pcre sasl2-bin resolvconf postfix-cdb
The following NEW packages will be installed
  bsd-mailx libdbd-mysql-perl libdbi-perl libhtml-template-perl
  libmysqlclient15off libmysqlclient16 libnet-daemon-perl libplrpc-perl mailx
  mysql-client-5.1 mysql-common mysql-server-5.1 postfix
0 upgraded, 13 newly installed, 0 to remove and 182 not upgraded.
Need to get 1907kB/25.3MB of archives.
After this operation, 59.5MB of additional disk space will be used.
Do you want to continue [Y/n]? Y
Get: 1 http://gb.archive.ubuntu.com jaunty-updates/main mysql-common 5.1.30really5.0.75-0ubuntu10.5 [63.6kB]
Get: 2 http://gb.archive.ubuntu.com jaunty-updates/main libmysqlclient15off 5.1.30really5.0.75-0ubuntu10.5 [1843kB]
Fetched 1907kB in 9s (205kB/s)
Preconfiguring packages ...
Selecting previously deselected package mysql-common.
(Reading database ... 121260 files and directories currently installed.)
...
Processing 1 added doc-base file(s)...
Registering documents with scrollkeeper...
Setting up libnet-daemon-perl (0.43-1) ...
Setting up libplrpc-perl (0.2020-1) ...
Setting up libdbi-perl (1.607-1) ...
Setting up libmysqlclient15off (5.1.30really5.0.75-0ubuntu10.5) ...

Setting up libdbd-mysql-perl (4.008-1) ...
Setting up libmysqlclient16 (5.1.31-1ubuntu2) ...

Setting up mysql-client-5.1 (5.1.31-1ubuntu2) ...

Setting up mysql-server-5.1 (5.1.31-1ubuntu2) ...
 * Stopping MySQL database server mysqld
   ...done.
100825 11:46:15  InnoDB: Started; log sequence number 0 46409
100825 11:46:15  InnoDB: Starting shutdown...
100825 11:46:17  InnoDB: Shutdown completed; log sequence number 0 46409
100825 11:46:17 [Warning] Forcing shutdown of 1 plugins
 * Starting MySQL database server mysqld
   ...done.
 * Checking for corrupt, not cleanly closed and upgrade needing tables.
...
Processing triggers for libc6 ...
ldconfig deferred processing now taking place
```

> **注記**
>
> apt-get コマンドは、典型的なツールおよびアプリケーション環境を提供するために、MySQL Server をはじめとするいくつかのパッケージをインストールします。これは、メインの MySQL パッケージに加えて多数のパッケージをインストールするということを意味します。

インストール中、初期データベースが作成され、MySQL root のパスワード (および確認) を求められます。構成ファイルが /etc/mysql/my.cnf に作成されています。init スクリプトが /etc/init.d/mysql に作成されています。

サーバーはすでに起動されています。次を使用してサーバーを自動的に起動および停止できます。

```
root-shell> service mysql [start|stop]
```

このサービスは自動的に 2、3、および 4 の実行レベルに追加され、stop スクリプトは単独のシャットダウンおよび再起動レベルにあります。

- Gentoo Linux

  ソースベースの配布として、Gentoo への MySQL のインストールには、ソースのダウンロード、Gentoo 固有のパッチの適用、および MySQL Server のコンパイルとインストールが含まれます。このプロセスは、emerge コマンドにより自動的に処理されます。インストールする MySQL のバージョンによっては、選択したプラットフォームの固有のバージョンをマスク解除する必要がある場合があります。

  MySQL Server およびクライアントツールは、単一のパッケージ dev-db/mysql で提供されます。インストールに使用できるバージョンのリストは、パッケージの portage ディレクトリを見ることで取得できます。

```
root-shell> ls /usr/portage/dev-db/mysql/mysql-5.1*
mysql-5.1.39-r1.ebuild
mysql-5.1.44-r1.ebuild
mysql-5.1.44-r2.ebuild
mysql-5.1.44-r3.ebuild
mysql-5.1.44.ebuild
```

```
mysql-5.1.45-r1.ebuild
mysql-5.1.45.ebuild
mysql-5.1.46.ebuild
```

特定の MySQL バージョンをインストールするには、アトム全体を指定する必要があります。例:

```
root-shell> emerge =dev-db/mysql-5.1.46
```

より簡単な方法は、virtual/mysql-5.1 パッケージを使用することで、最新のバージョンがインストールされます。

```
root-shell> emerge =virtual/mysql-5.1
```

パッケージが (現在のプラットフォームに対してテストまたは認証されていないため) マスクされている場合は、ACCEPT_KEYWORDS 環境変数を使用します。例:

```
root-shell> ACCEPT_KEYWORDS="~x86" emerge =virtual/mysql-5.1
```

インストール後、mysql_install_db を使用して新規データベースを作成し、MySQL の root ユーザーのパスワードを設定してください。構成インタフェースを使用してパスワードを設定し、初期データベースを作成できます。

```
root-shell> emerge --config =dev-db/mysql-5.1.46
```

サンプル構成ファイルが、自動的に /etc/mysql/my.cnf に作成され、init スクリプトが /etc/init.d/mysql に作成されています。

MySQL を、通常の (デフォルト) 実行レベルで自動的に起動できるようにするには、次を使用できます。

```
root-shell> rc-update add mysql default
```

2.6 Unbreakable Linux Network (ULN) を使用した MySQL のインストール

セクション2.5「Linux に MySQL をインストールする」で説明するように、Linux は、MySQL をインストールするための何種類かのソリューションをサポートします。1 つの方法はこのセクションで説明しますが、オラクルの Unbreakable Linux Network (ULN) からインストールすることです。Oracle Linux および ULN の詳細は、http://linux.oracle.com/にあります。

ULN を使用するには、ULN ログインを取得し、ULN でのインストールに使用するマシンを登録する必要があります。これは、ULN FAQ で詳細に説明しています。このページでは、パッケージのインストールと更新の方法も説明しています。MySQL パッケージは、ULN の、使用しているシステムアーキテクチャーの 「MySQL for Oracle Linux 6」 および 「MySQL for Oracle Linux 7」 のチャネルにあります。

> **注記**
>
> 本書の記述の時点では、ULN は Oracle Linux 6 および Oracle Linux 7 の MySQL 5.6 を提供します。

ULN を使用して MySQL をインストールしたあと、このセクション、特にセクション2.5.5「RPM パッケージを使用して MySQL を Linux にインストールする」に、サーバーの起動および停止に関する情報があります。

既存の MySQL インストールを ULN を使用するインストールに更新している場合は、mysqldump を使用してデータをエクスポートし、既存のインストールを削除し、MySQL を ULN からインストールし、エクスポートしたデータを新しくインストールされた MySQL にロードするという手順が推奨されます。

アップグレードしようとしている既存の MySQL インストールが以前のリリースシリーズ (MySQL 5.6 より前) である場合は、MySQL のアップグレードに関するセクション、セクション2.11.1「MySQL のアップグレード」を必ずお読みください。

2.7 Solaris および OpenSolaris に MySQL をインストールする

Solaris および OpenSolaris の MySQL は、さまざまな形式で使用可能です。

- ネイティブの Solaris PKG 形式を使用するインストールの詳細は、セクション2.7.1「Solaris PKG を使用して Solaris に MySQL をインストールする」を参照してください。
- OpenSolaris では、SMF 管理コマンドを使用してインストールを制御できるように、標準パッケージリポジトリに、Service Management Framework (SMF) のエントリを含む OpenSolaris 用に特別にビルドされた MySQL

パッケージが含まれます。詳細については、セクション2.7.2「IPS を使用して MySQL を OpenSolaris にインストールする」を参照してください。

- 標準の tar バイナリインストールを使用するには、セクション2.2「一般的なバイナリを使用した MySQL の Unix/Linux へのインストール」の注記を参照してください。インストール前後に必要となる可能性のある、Solaris 固有の注記については、このセクションの最後の注記およびヒントをチェックしてください。

tarball または PKG 形式の Solaris 用のバイナリ MySQL 配布を取得するには、https://dev.mysql.com/downloads/mysql/5.6.html にアクセスしてください。

MySQL を Solaris にインストールして使用する上で知っておくべき追加の注記:

- MySQL を mysql ユーザーおよびグループで使用する場合は、groupadd コマンドおよび useradd コマンドを使用します。

```
groupadd mysql
useradd -g mysql mysql
```

- MySQL をバイナリの tarball 配布で Solaris にインストールする場合、MySQL の配布をアンパックする前にすでに問題が生じることがあります。これは Solaris の tar が長いファイル名を扱えないためです。これは、MySQL をアンパックするときにエラーが表示される場合があることを意味します。

  これが発生したら、GNU tar (gtar) を使用して配布をアンパックする必要があります。Solaris 10 および OpenSolaris では、gtar は通常 /usr/sfw/bin/gtar に置かれていますが、デフォルトのパス定義に含まれていない場合があります。

- Solaris 10 を x86_64 で使用する場合、InnoDB ファイルを forcedirectio オプションで保持するファイルシステムをマウントするようにしてください。(デフォルトではこのオプションなしでマウントされます。)そうしないと InnoDB ストレージエンジンをこのプラットフォームで使用した場合性能が大幅に劣化します。

- MySQL を自動的に起動する場合は、support-files/mysql.server を /etc/init.d にコピーして、それに /etc/rc3.d/S99mysql.server という名前のシンボリックリンクを作成します。

- あまりにも多くのプロセスが急激に mysqld に接続を試みた場合、MySQL ログに次のエラーが記録されます。

```
Error in accept: Protocol error
```

  この問題の回避策としてサーバーを --back_log=50 オプションで起動するとよいでしょう。

- Solaris でコアファイルの生成を構成するには、coreadm コマンドを使用します。setuid() アプリケーションでコアを生成するとセキュリティー上の問題があるため、デフォルトでは Solaris は setuid() プログラムではコアファイルをサポートしません。ただし、この動作は coreadm を使用して変更できます。現在のユーザーに対して setuid() コアファイルを有効にすると、モード 600 を使用して生成され、スーパーユーザーによって所有されます。

2.7.1 Solaris PKG を使用して Solaris に MySQL をインストールする

バイナリ tarball 配布ではなく、ネイティブの Solaris PKG 形式を使用して、バイナリパッケージを使用して MySQL を Solaris および OpenSolaris にインストールできます。

このパッケージを使用するには、対応する mysql-VERSION-solaris10-PLATFORM.pkg.gz ファイルをダウンロードしてから圧縮解除します。例:

```
shell> gunzip mysql-5.6.23-solaris10-x86_64.pkg.gz
```

新しいパッケージをインストールするには、pkgadd を使用して画面の指示に従います。この操作を実行するにはルート権限が必要です。

```
shell> pkgadd -d mysql-5.6.23-solaris10-x86_64.pkg

The following packages are available:
  1  mysql     MySQL Community Server (GPL)
               (i86pc) 5.6.23

Select package(s) you wish to process (or 'all' to process
all packages). (default: all) [?,??,q]:
```

PKG インストーラは、必要なすべてのファイルおよびツールをインストールし、そのあとデータベースが存在しない場合はデータベースを初期化します。インストールを完了するには、インストールの最後の指示に示されるように、MySQL のルートパスワードを設定してください。あるいは、インストールに同梱されている mysql_secure_installation スクリプトを実行することもできます。

デフォルトでは、PKG パッケージは MySQL をルートパス /opt/mysql にインストールします。インストールのルートパスを変更できるのは pkgadd を使用している場合のみで、これを使用すると MySQL を異なる Solaris ゾーンにインストールできます。特定のディレクトリにインストールする必要がある場合は、バイナリの tar ファイルの配布を使用してください。

pkg インストーラは、適切な MySQL スタートアップスクリプトを /etc/init.d/mysql にコピーします。MySQL を自動的にスタートアップおよびシャットダウンできるようにするためには、このファイルと init スクリプトディレクトリとの間にリンクを作成してください。たとえば、MySQL の安全なスタートアップおよびシャットダウンを確実にするには、次のコマンドを使用して適切なリンクを追加します。

```
shell> ln /etc/init.d/mysql /etc/rc3.d/S91mysql
shell> ln /etc/init.d/mysql /etc/rc0.d/K02mysql
```

MySQL を削除する場合、インストールされるパッケージ名は mysql です。これを pkgrm コマンドと合わせて使用してインストールを削除できます。

Solaris パッケージファイル形式を使用している場合にアップグレードするには、更新パッケージをインストールする前に既存のインストールを削除する必要があります。パッケージを削除しても既存のデータベース情報は削除されません。サーバー、バイナリ、およびサポートファイルのみが削除されます。したがって、通常のアップグレード手順は次のようになります。

```
shell> mysqladmin shutdown
shell> pkgrm mysql
shell> pkgadd -d mysql-5.6.23-solaris10-x86_64.pkg
shell> mysqld_safe &
shell> mysql_upgrade
```

アップグレードを実行する前に、セクション2.11「MySQL のアップグレードとダウングレード」の注記を確認してください。

2.7.2 IPS を使用して MySQL を OpenSolaris にインストールする

OpenSolaris では、MySQL の標準パッケージはコアリポジトリにあります。MySQL パッケージは MySQL の特定のリリースに基づいており、定期的に更新されます。最新のリリースは、ネイティブの Solaris PKG、tar、またはソースインストールを使用する必要があります。ネイティブ OpenSolaris パッケージは、ネイティブサービス管理ツールを使用して、自動スタートアップおよびリカバリなど MySQL インストールを容易に制御できるように、SMF ファイルを含みます。

MySQL を OpenSolaris にインストールするには、pkg コマンドを使用します。root としてログインする必要があります。あるいは、次の例に示すように pfexec ツールを使用します。

```
shell> pfexec pkg install SUNWmysql56
```

パッケージセットは、MySQL クライアントライブラリを含む SUNWmysql56lib、SMF および構成ファイルなどのルートコンポーネントを含む SUNWmysql56r、およびスクリプト、バイナリツール、およびその他のファイルを含む SUNWmysql56u の、3 つの個別のパッケージをインストールします。これらのパッケージは、対応するコンポーネントのみが必要な場合は個別にインストールできます。

MySQL ファイルは /usr/mysql にインストールされ、サブディレクトリ (bin、lib、など.) をバージョン固有のディレクトリにシンボリックリンクします。MySQL 5.6 では、完全なインストールは /usr/mysql/5.6 に置かれます。デフォルトのデータディレクトリは /var/mysql/5.6/data です。構成ファイルは /etc/mysql/5.6/my.cnf にインストールされます。このレイアウトにより、MySQL の複数のバージョンを、ほかのバージョンからのデータおよびバイナリを上書きすることなくインストールできます。

インストールが終了したら、mysql_install_db を実行してデータベースを初期化し、mysql_secure_installation を使用してインストールをセキュアにします。

SMF を使用する MySQL インストールの管理

インストールが終了したら、インストール済みの SMF 構成を使用して MySQL Server を起動および停止できます。サービス名は mysql、あるいは複数のバージョンをインストールしている場合は、mysql:version_56 のようにフルバージョン名を使用してください。MySQL を起動して、ブート時に起動できるようにするには:

```
shell> svcadm enable mysql
```

ブート時の MySQL の起動を無効にし、MySQL Server が実行している場合はシャットダウンするには、次を使用します。

```
shell> svcadm disable mysql
```

たとえば構成ファイルの変更後などに MySQL を再起動するには、restart オプションを使用します。

```
shell> svcadm restart mysql
```

SMF を使用してデータディレクトリを構成し、フル 64 ビットモードを有効にすることも可能です。たとえば、MySQL が使用するデータディレクトリを設定するには:

```
shell> svccfg
svc:> select mysql:version_56
svc:/application/database/mysql:version_56> setprop mysql/data=/data0/mysql
```

デフォルトでは、32 ビットバイナリが使用されます。64 ビットプラットフォームで 64 ビットサーバーを有効にするには、enable_64bit パラメータを設定します。例:

```
svc:/application/database/mysql:version_56> setprop mysql/enable_64bit=1
```

これらのオプションの設定後、SMF のリフレッシュが必要です。

```
shell> svcadm refresh mysql
```

2.8 FreeBSD に MySQL をインストールする

このセクションでは、FreeBSD Unix のバリアントへの MySQL のインストールに関する情報を提供します。

オラクルが提供するバイナリ配布を使用して、MySQL を FreeBSD にインストールできます。詳細については、セクション2.2「一般的なバイナリを使用した MySQL の Unix/Linux へのインストール」を参照してください。

MySQL のインストールでもっとも容易な (かつ推奨される) 方法は mysql-server ポートおよび mysql-client ポートを使用することです。それらは http://www.freebsd.org/ で入手できます。これらのポートを使用することで次のメリットがあります。

- 使用している FreeBSD バージョンで動作することが知られている、すべての最適化を行なった実働可能な MySQL。
- 自動構成およびビルド。
- /usr/local/etc/rc.d にインストールされたスタートアップスクリプト。
- どのファイルがインストールされているかを確認するための pkg_info -L の使用。
- 使用しているマシンで MySQL が必要でなくなった場合にそれを削除できる pkg_delete の使用。

MySQL ビルドプロセスが機能するには GNU make (gmake) が必要です。GNU make が利用できない場合、MySQL をコンパイルする前にインストールする必要があります。

ポートシステムを使用してインストールするには:

```
# cd /usr/ports/databases/mysql51-server
# make
...
# cd /usr/ports/databases/mysql51-client
# make
...
```

標準ポートインストールでは、サーバーは /usr/local/libexec/mysqld に置かれ、MySQL Server のスタートアップスクリプトは /usr/local/etc/rc.d/mysql-server に置かれます。

BSD 実装に関する追加の注記:

- ポートシステムを使用して MySQL をインストールしたあとに削除するには:

  ```
  # cd /usr/ports/databases/mysql51-server
  # make deinstall
  ...
  # cd /usr/ports/databases/mysql51-client
  # make deinstall
  ...
  ```

- MySQL の現在の日付に問題がある場合には、TZ 変数の設定が役に立ちます。セクション2.12「環境変数」を参照してください。

2.9 ソースから MySQL をインストールする

MySQL をソースコードからビルドすると、ビルドパラメータ、コンパイラ最適化、およびインストールの場所をカスタマイズできます。MySQL を実行できることがわかっているシステムのリストは、https://www.mysql.com/support/supportedplatforms/database.htmlを参照してください。

ソースからのインストールに進む前に、使用しているプラットフォーム用に事前コンパイルされたバイナリ配布をオラクルが作成しているかどうか、およびそれがニーズに適合しているかどうかをチェックしてください。弊社は、パフォーマンスを最適化するために、最善のオプションでバイナリをビルドすることを保証するべく多大な努力をしています。バイナリ配布のインストール手順は、セクション2.2「一般的なバイナリを使用した MySQL の Unix/Linux へのインストール」を参照してください。

ソースのインストール方法

MySQL をソースからインストールするには、2 つの方法があります。

- 標準の MySQL ソース配布を使用します。標準の配布を取得するには、セクション2.1.3「MySQL の取得方法」を参照してください。標準の配布からのビルドの詳細は、セクション2.9.2「標準ソース配布を使用して MySQL をインストールする」を参照してください。

  標準の配布は、圧縮 tar ファイル、Zip アーカイブ、または RPM パッケージとして入手可能です。配布ファイルは、mysql-VERSION.tar.gz, mysql-VERSION.zip、または mysql-VERSION.rpm という形式の名前を持ちます。ここで、VERSION は 5.6.23 などの番号です。ソース配布のファイル名は汎用でプラットフォーム名を含まないのに対し、バイナリ配布名にはその配布が対象とするシステムのタイプを示すプラットフォーム名が含まれる (たとえば、pc-linux-i686 または winx64) ことから、ソースファイルを事前コンパイル済みのバイナリ配布と区別できます。

- MySQL 開発ツリーを使用します。開発ツリーからのビルドの詳細は、セクション2.9.3「開発ソースツリーを使用して MySQL をインストールする」を参照してください。

ソースインストールのシステム要件

MySQL をソースからインストールするためには、いくつかの開発ツールが必要です。これらのツールの一部は、標準ソース配布または開発ソースツリーのいずれを使用するかにかかわらず必要です。その他のツールは、どちらのインストール方法を使用するかによって必要かどうかが決まります。

MySQL をソースからインストールするには、インストール方法にかかわらず次のツールがシステムになければなりません。

- CMake。これはすべてのプラットフォームでビルドフレームワークとして使用されます。CMake は http://www.cmake.org からダウンロードできます。

- 優良な make プログラム。一部のプラットフォームには独自の make 実装が備わっていますが、GNU make 3.75 以降を使用することを強くお勧めします。これは、使用しているシステムですでに gmake として使用可能になっている場合があります。GNU make は、http://www.gnu.org/software/make/ から入手可能です。

- 動作する ANSI C++ コンパイラ。GCC 4.2.1 以降、Sun Studio 12 以降、Visual Studio 2010 以降、および多くの最新のベンダー供給のコンパイラが機能することが知られています。

- テストスクリプトを実行する場合は、Perl が必要です。ほとんどの Unix 類似システムには Perl が含まれます。Windows では、ActiveState Perl などのバージョンが使用できます。

MySQL を標準ソース配布からインストールするには、配布ファイルをアンパックするために次のツールのいずれかが必要です。

- .tar.gz で圧縮された tar ファイルの場合: 配布を圧縮解除するための GNU gunzip、およびそれをアンパックするための妥当な tar。使用している tar プログラムが z オプションをサポートする場合は、ファイルの展開とアンパックの両方を実行できます。

  GNU tar が機能することが知られています。一部のオペレーティングシステムで提供される標準の tar は、MySQL 配布内の長いファイル名をアンパックできません。GNU tar をダウンロードしてインストールするか、プリインストールバージョンの GNU tar が利用可能であればそれを使用します。通常、これは gnutar、gtar、または tar という名前です (/usr/sfw/bin または /usr/local/bin などの GNU または Free Software ディレクトリ内)。GNU tar は、http://www.gnu.org/software/tar/ から入手可能です。

- .zip Zip アーカイブの場合: WinZip または .zip ファイルを読み取ることができるその他のツール。

- .rpm RPM パッケージの場合: 配布のビルドに使用される rpmbuild プログラムでアンパックできます。

MySQL を開発ソースツリーからインストールするには、次の追加ツールが必要です。

- 開発ソースコードを取得するには、次のリビジョン管理システムが必要です。
  - Git: GitHub ヘルプには、さまざまなプラットフォームに Git をダウンロードしてインストールするための説明があります。MySQL は、2014 年 9 月に正式に GitHub に参加しました。MySQL の GitHub への移動の詳細は、MySQL Release Engineering ブログ MySQL on GitHub のお知らせを参照してください。
  - Bazaar: Bazaar VCS Web サイトには、さまざまなプラットフォームに Bazaar をダウンロードしてインストールするための説明があります。Bazaar は、Python をサポートするすべてのプラットフォームでサポートされ、そのためすべての Linux、Unix、Windows、または OS X ホストと互換です。
- bison 2.1 以降。http://www.gnu.org/software/bison/ から入手可能です。(バージョン 1 はサポートされなくなりました。)可能であれば bison の最新バージョンを使用します。問題が生じた場合は、以前のバージョンに戻るのではなく、より新しいバージョンにアップグレードします。

  bison は、http://www.gnu.org/software/bison/ から入手可能です。bison for Windows は、http://gnuwin32.sourceforge.net/packages/bison.htm からダウンロードできます。「Complete package, excluding sources」 のラベルのパッケージをダウンロードします。Windows では、bison のデフォルトの場所は C:\Program Files\GnuWin32 ディレクトリです。ディレクトリ名にスペースが含まれるため、一部のユーティリティーでは bison を検索できない場合があります。また、パスにスペースがある場合 Visual Studio がフリーズすることがあります。これらの問題は、たとえば C:\GnuWin32 など、スペースを含まないディレクトリにインストールすることで解決できます。
- OpenSolaris および Solaris Express では、bison のほかに m4 もインストールされていなければなりません。m4 は http://www.gnu.org/software/m4/ から入手可能です。

  > **注記**
  >
  > プログラムをインストールする必要がある場合は、PATH 環境変数を、プログラムが置かれるディレクトリを含むように変更します。セクション4.2.10「環境変数の設定」を参照してください。

問題が発生してバグをレポートする必要が生じた場合には、セクション1.6「質問またはバグをレポートする方法」 の手順に従ってください。

2.9.1 ソースインストールの MySQL のレイアウト

デフォルトでは、MySQL をソースからコンパイルしたあとにインストールすると、インストール手順によりファイルが /usr/local/mysql にインストールされます。インストールディレクトリの下のコンポーネントの場所は、バイナリ配布の場合と同じです。表2.3「一般的な Unix/Linux バイナリパッケージの MySQL インストールのレイアウト」、セクション2.3.1「Microsoft Windows 上での MySQL のインストールレイアウト」を参照してください。デフォルトとは異なるインストールの場所を構成するには、セクション2.9.4「MySQL ソース構成オプション」で説明されるオプションを使用します。

2.9.2 標準ソース配布を使用して MySQL をインストールする

標準ソース配布を使用して MySQL をインストールするには:

1. システムが、セクション2.9「ソースから MySQL をインストールする」にリストされるツール要件を満たすことを確認します。
2. 配布ファイルを セクション2.1.3「MySQL の取得方法」 の説明に従って取得します。
3. このセクションの説明に従って、配布の構成、ビルド、およびインストールを実行します。
4. セクション2.10「インストール後のセットアップとテスト」の説明に従って、インストール後の手順を実行します。

MySQL 5.6 では、すべてのプラットフォームで CMake がビルドフレームワークとして使用されます。ここに記載する説明で、動作するインストールを作成できるでしょう。CMake を使用して MySQL をビルドする方法の詳細は、CMake による MySQL Server のビルド方法を参照してください。

ソース RPM から開始する場合は、インストールするバイナリ RPM を、次のコマンドを使用して作成します。rpmbuild がない場合は代わりに rpm を使用します。

```
shell> rpmbuild --rebuild --clean MySQL-VERSION.src.rpm
```

結果として 1 つまたは複数の RPM パッケージが生成されます。セクション2.5.5「RPM パッケージを使用して MySQL を Linux にインストールする」の指示に従ってインストールします。

圧縮 tar ファイルまたは Zip アーカイブソース配布からのインストールのシーケンスは、ソース配布はすべてのプラットフォームで使用されること、および配布を構成してコンパイルする手順が含まれる点を除き、一般的なバイナリ配布のプロセスと同様です (セクション2.2「一般的なバイナリを使用した MySQL の Unix/Linux へのインストール」を参照してください)。たとえば、Unix の圧縮 tar ファイルのソース配布では、基本的なインストールコマンドシーケンスは次のようになります。

```
# Preconfiguration setup
shell> groupadd mysql
shell> useradd -r -g mysql mysql
# Beginning of source-build specific instructions
shell> tar zxvf mysql-VERSION.tar.gz
shell> cd mysql-VERSION
shell> cmake .
shell> make
shell> make install
# End of source-build specific instructions
# Postinstallation setup
shell> cd /usr/local/mysql
shell> chown -R mysql .
shell> chgrp -R mysql .
shell> scripts/mysql_install_db --user=mysql
shell> chown -R root .
shell> chown -R mysql data
shell> bin/mysqld_safe --user=mysql &
# Next command is optional
shell> cp support-files/mysql.server /etc/init.d/mysql.server
```

mysql_install_db は、my.cnf という名前のデフォルトオプションファイルを基本インストールディレクトリに作成します。このファイルは my-default.cnf という名前の配布パッケージに含まれるテンプレートから作成されます。詳細は、セクション5.1.2.2「サンプルのデフォルトサーバー構成ファイルの使用」 を参照してください。

このあと、ソースビルド固有の手順について、より詳細に説明します。

> **注記**
>
> ここに示した手順では、MySQL アカウントにパスワードは設定しません。その手順のあとは、セクション2.10「インストール後のセットアップとテスト」に進み、インストール後のセットアップとテストを実行します。

構成前のセットアップの実行

Unix では、MySQL Server を実行してデータベースディレクトリを所有する、mysql ユーザーおよびグループをセットアップします。詳細は、セクション2.2「一般的なバイナリを使用した MySQL の Unix/Linux へのインストール」の mysql システムユーザーおよびグループの作成を参照してください。次に、指定された場合を除き、mysql ユーザーとして次の手順を実行します。

配布の取得とアンパック

配布をアンパックするディレクトリを選択してそこに移動します。

配布ファイルを セクション2.1.3「MySQL の取得方法」 の説明に従って取得します。

配布を現在のディレクトリにアンパックします。

- 圧縮 tar ファイルをアンパックするには、tar が z オプションをサポートする場合、このコマンドで配布の圧縮解除とアンパックを実行できます。

  ```
  shell> tar zxvf mysql-VERSION.tar.gz
  ```

  tar が z オプションをサポートしない場合は、gunzip を使用して配布をパックし、tar を使用してアンパックします。

  ```
  shell> gunzip < mysql-VERSION.tar.gz | tar xvf -
  ```

  あるいは、CMake を使用して配布を圧縮解除してアンパックすることも可能です。

  ```
  shell> cmake -E tar zxvf mysql-VERSION.tar.gz
  ```

- Zip アーカイブをアンパックするには、WinZip または .zip ファイルを読み取ることができるその他のツールを使用します。

配布ファイルをアンパックすると、mysql-VERSION という名前のディレクトリが作成されます。

配布の構成

アンパックした配布のトップレベルのディレクトリに場所を変更します。

```
shell> cd mysql-VERSION
```

ソースディレクトリを構成します。最低限の構成を行うコマンドには、構成のデフォルトをオーバーライドするオプションは含まれません。

```
shell> cmake .
```

Windows では、開発環境を指定します。たとえば、次のコマンドはそれぞれ 32 ビットまたは 64 ビッドの MySQL ビルドを構成します。

```
shell> cmake . -G "Visual Studio 10 2010"
shell> cmake . -G "Visual Studio 10 2010 Win64"
```

OS X で Xcode IDE を使用するには:

```
shell> cmake . -G Xcode
```

cmake を実行する場合、コマンド行にオプションを追加するとよいでしょう。次にいくつかの例を示します。

- -DBUILD_CONFIG=mysql_release: オラクルが公式な MySQL リリースのバイナリ配布を生成するために使用するのと同じビルドオプションでソースを構成します。
- -DCMAKE_INSTALL_PREFIX=dir_name: 特定の場所にインストールするように配布を構成します。
- -DCPACK_MONOLITHIC_INSTALL=1: make package が、複数のファイルではなく単独のインストールファイルを生成するようにします。
- -DWITH_DEBUG=1: 配付をデバッグサポート付きでビルドします。

オプションのより詳細なリストは、セクション2.9.4「MySQL ソース構成オプション」を参照してください。

構成オプションをリストするには、次のコマンドのいずれかを使用します。

```
shell> cmake . -L   # overview
shell> cmake . -LH  # overview with help text
shell> cmake . -LAH # all params with help text
shell> ccmake .     # interactive display
```

CMake が失敗する場合は、異なるオプションで再実行して再構成する必要がある場合があります。再構成を行う場合は、次に注意してください。

- CMake を以前に実行したあとで実行すると、以前の起動時に収集した情報を使用する場合があります。この情報は CMakeCache.txt に格納されています。CMake は、起動時にそのファイルを探し、存在すればその情報がまだ正しいと仮定して内容を読み込みます。この仮定は再構成した場合には無効です。
- CMake を実行するたびに、make を再実行して再コンパイルする必要があります。しかし、前のビルドの古いオブジェクトファイルが異なる構成オプションでコンパイルされている場合、それらを最初に削除する場合もあります。

古いオブジェクトファイルまたは構成情報が使用されることを予防するために、Unix では CMake を再実行する前に次のコマンドを実行します。

```
shell> make clean
shell> rm CMakeCache.txt
```

あるいは、Windows の場合:

```
shell> devenv MySQL.sln /clean
shell> del CMakeCache.txt
```

ソースツリーからビルドする場合 (あとで説明します)、CMakeCache.txt ファイルおよびすべてのビルドファイルはビルドディレクトリにあるため、オブジェクトファイルおよびキャッシュされた構成情報を削除するには、そのディレクトリを削除します。

MySQL メーリングリストにメールを送信して構成の支援を依頼する場合は、CMakeFiles ディレクトリのファイルで、失敗に関する有用な情報をまずチェックしてください。バグレポートを提出する際は、セクション1.6「質問またはバグをレポートする方法」の説明に従ってください。

配布のビルド

Unix の場合:

```
shell> make
shell> make VERBOSE=1
```

2 番目のコマンドは、コンパイル済みの各ソースに対するコマンドを表示するため、VERBOSE をセットします。

GNU make を使用し、それが gmake としてインストールされているシステムでは、代わりに gmake を使用します。

Windows の場合:

```
shell> devenv MySQL.sln /build RelWithDebInfo
```

ソースツリーからビルドして、ツリーをクリーンな状態に保つことができます。トップレベルのソースディレクトリが、現在の作業ディレクトリの下の mysql-src という名前のディレクトリの場合、次のように、同じレベルの bld という名前のディレクトリにビルドできます。

```
shell> mkdir bld
shell> cd bld
shell> cmake ../mysql-src
```

ビルドディレクトリは、実際にソースツリーの外部になければならないわけではありません。たとえば、ディレクトリ内にビルドするには、トップレベルのソースツリーの下の bld という名前のディレクトリにビルドできます。そのためには、現在の作業ディレクトリとして mysql-src から開始します。

```
shell> mkdir bld
shell> cd bld
shell> cmake ..
```

複数のソースツリーが同じレベルにある場合 (たとえば、複数のバージョンの MySQL をビルドする場合)、2 番目の方法が有利です。最初の方法では、すべてのビルドディレクトリを同じレベルに置くため、それぞれに一意の名前が必要です。2 番目の方法では、各ソースツリー内のビルドディレクトリに同じ名前を使用できます。

コンパイル段階に進んだが、配布がビルドされない場合は、セクション2.9.5「MySQL のコンパイルに関する問題」を参照してください。それでも問題が解決しない場合は、セクション1.6「質問またはバグをレポートする方法」を参照して、それをバグデータベースに入力してください。必要なツールの最新バージョンをインストール済みで、構成ファイルを処理しようとしてそれらのツールがクラッシュする場合は、それもレポートしてください。ただし、コマンドが見つかりませんという エラーや、必要なツールに関して同様の問題がある場合は、レポートしないでください。代わりに、必要なツールがすべてインストール済みであり、シェルがそれらを検索できるように PATH 変数が正しく設定されていることを確認してください。

配付のインストール

Unix の場合:

```
shell> make install
```

これは、ファイルを構成されたインストールディレクトリ (デフォルトでは /usr/local/mysql) にインストールします。コマンドを root として実行する必要がある場合があります。

特定のディレクトリにインストールするには、コマンド行に DESTDIR パラメータを追加します。

```
shell> make install DESTDIR="/opt/mysql"
```

あるいは、任意の場所にインストールできるインストールパッケージファイルを生成します。

```
shell> make package
```

この操作は、一般的なバイナリ配布パッケージのようにインストールできる、1 つまたは複数の .tar.gz ファイルを生成します。セクション2.2「一般的なバイナリを使用した MySQL の Unix/Linux へのインストール」を参照してください。CMake を -DCPACK_MONOLITHIC_INSTALL=1 を使用して実行すると、その操作で単独のファイルが作成されます。そうでない場合は複数のファイルが作成されます。

Windows では、データディレクトリを生成してから、.zip アーカイブインストールパッケージを作成します。

```
shell> devenv MySQL.sln /build RelWithDebInfo /project initial_database
shell> devenv MySQL.sln /build RelWithDebInfo /project package
```

結果の .zip アーカイブは任意の場所にインストールできます。セクション2.3.5「非インストール Zip アーカイブを使用して Microsoft Windows に MySQL をインストールする」を参照してください。

インストール後のセットアップの実行

インストールプロセスの残りの部分は、構成ファイルのセットアップ、コアデータベースの作成、および MySQL Server の開始などです。手順は、セクション2.10「インストール後のセットアップとテスト」を参照してください。

> **注記**
>
> MySQL の付与テーブルにリストされているアカウントには、最初はパスワードがありません。サーバーの起動後に、セクション2.10「インストール後のセットアップとテスト」の説明に従って、それらにパスワードを設定する必要があります。

2.9.3 開発ソースツリーを使用して MySQL をインストールする

このセクションでは、MySQL を最新の開発ソースコードからインストールする方法を説明します。最新の開発ソースコードは現在、GitHub および Launchpad の両方にホストされています。MySQL Server のソースコードをこれらのリポジトリホスティングサービスのいずれかから取得するには、ローカルの MySQL Git リポジトリまたはローカルの MySQL Bazaar ブランチをセットアップします。

- GitHub では、MySQL Server およびその他の MySQL プロジェクトは MySQL ページにあります。MySQL Server プロジェクトは、MySQL 5.5、5.6、および 5.7 のブランチを含む単独のリポジトリです。

  MySQL は、2014 年 9 月に正式に GitHub に参加しました。MySQL の GitHub への移動の詳細は、MySQL Release Engineering ブログ MySQL on GitHub のお知らせを参照してください。

- Launchpad では、MySQL Server、MySQL Workbench などを含む MySQL プロジェクトは、Oracle/MySQL エンジニアリングページにあります。MySQL Server のみに関係するリポジトリについては、MySQL Server ページを参照してください。

開発ソースからのインストールの前提条件

MySQL を開発ソースツリーからインストールするには、セクション2.9「ソースから MySQL をインストールする」に概説されるツール要件をシステムが満たしていなければなりません。

MySQL Git リポジトリのセットアップ

MySQL Git リポジトリをマシンにセットアップするには、次の手順を使用します。

1. MySQL Git リポジトリをマシンにクローニングします。次のコマンドは、MySQL Git リポジトリを mysql-server という名前のディレクトリにクローニングします。ダウンロードサイズは約 437M バイトです。接続の速度によっては、最初のダウンロードを完了するのにしばらく時間がかかります。

   ```
   ~$ git clone https://github.com/mysql/mysql-server.git
   Cloning into 'mysql-server'...
   remote: Counting objects: 1035465, done.
   remote: Total 1035465 (delta 0), reused 0 (delta 0)
   Receiving objects: 100% (1035465/1035465), 437.48 MiB | 5.10 MiB/s, done.
   Resolving deltas: 100% (855607/855607), done.
   Checking connectivity... done.
   Checking out files: 100% (21902/21902), done.
   ```

2. クローニング操作が完了すると、ローカルの MySQL Git リポジトリは次のようになります。

   ```
   ~$ cd mysql-server

   ~/mysql-server$ ls
   BUILD           COPYING           libmysqld    regex         tests
   BUILD-CMAKE     dbug              libservices  scripts       unittest
   client          Docs              man          sql           VERSION
   cmake           extra             mysql-test   sql-bench     vio
   CMakeLists.txt  include           mysys        sql-common    win
   cmd-line-utils  INSTALL-SOURCE    packaging    storage       zlib
   config.h.cmake  INSTALL-WIN-SOURCE plugin      strings
   ```

```
configure.cmake  libmysql          README      support-files
```

3. git branch -r コマンドを使用して、MySQL リポジトリのリモート追跡ブランチを表示します。

```
~/mysql-server$ git branch -r
  origin/5.5
  origin/5.6
  origin/5.7
  origin/HEAD -> origin/5.7
```

4. ローカルリポジトリにチェックアウトされたブランチを表示するには、git branch コマンドを発行します。MySQL Git リポジトリをクローニングしたとき、MySQL 5.7 ブランチが自動的にチェックアウトされています。アスタリスクにより、5.7 ブランチがアクティブなブランチであることが示されます。

```
~/mysql-server$ git branch
* 5.7
```

5. 異なる MySQL ブランチをチェックアウトするには、ブランチ名を指定して git checkout コマンド実行します。たとえば、MySQL 5.6 ブランチをチェックアウトするには:

```
~/mysql-server$ git checkout 5.6
Branch 5.6 set up to track remote branch 5.6 from origin.
Switched to a new branch '5.6'
```

6. git branch をふたたび実行して、MySQL 5.6 ブランチが存在することを確認します。最後にチェックアウトした MySQL 5.6 にアスタリスクがマークされ、それがアクティブなブランチであることが示されています。

```
~/mysql-server$ git branch
* 5.6
  5.7
```

   git checkout コマンドも、ブランチの切り替えに使用されます。たとえば、ふたたび MySQL 5.7 をアクティブブランチにするには、git checkout 5.7 を実行します。

7. MySQL Git リポジトリを最初にセットアップしたあとの変更を取得するには、更新したいブランチに切り替えて git pull コマンドを発行します。

```
~/mysql-server$ git checkout 5.6
~/mysql-server$ git pull
```

   コミット履歴を調べるには、git log オプションを使用します。

```
~/mysql-server$ git log
```

   GitHub MySQL サイトでコミット履歴とソースコードを参照することもできます。

   変更や疑問のあるコードがある場合は、電子メールを MySQL internals メーリングリストに送信してください。セクション1.5.1「MySQL メーリングリスト」 を参照してください。パッチの貢献については、MySQL Server への貢献を参照してください。

8. MySQL Git リポジトリをクローニングしてビルドするブランチをチェックアウトしたら、MySQL Server をソースコードからビルドできます。説明はセクション2.9.2「標準ソース配布を使用して MySQL をインストールする」にありますが、配布の取得およびアンパックに関する部分はスキップします。

   本番環境マシン上の配布ソースツリーからのビルドをインストールする場合は注意が必要です。インストールコマンドが、ライブリリースインストールを上書きすることがあります。MySQL がインストール済みであり、それを上書きしたくない場合は、CMAKE_INSTALL_PREFIX、MYSQL_TCP_PORT、および MYSQL_UNIX_ADDR オプションの値を、本番環境サーバーで使用したものとは異なるものにして、CMake を実行します。複数のサーバーが相互に干渉することを防ぐ方法の詳細は、セクション5.3「1 つのマシン上での複数の MySQL インスタンスの実行」を参照してください。

   新しいインストールを厳密に検証します。たとえば、新機能をクラッシュさせてみてください。make test の実行から始めてみてください。セクション24.1.2「MySQL テストスイート」を参照してください。

MySQL Bazaar ブランチのセットアップ

MySQL Bazaar ブランチをマシンにセットアップするには、次の手順を使用します。

1. Launchpad にホストされている MySQL 開発ソースコードのコピーを取得するには、新しい Bazaar ブランチを作成します。まだ Bazaar リポジトリのディレクトリをセットアップしていない場合は、新しいディレクトリを初期化する必要があります。

```
shell> mkdir mysql-server
shell> bzr init-repo --trees mysql-server
```

これは 1 回かぎりの操作です。

2. 初期化されたリポジトリディレクトリがあることを前提に、パブリック MySQL Server リポジトリをブランチして、ローカルのソースツリーを作成します。特定のバージョンのブランチを作成するには:

```
shell> cd mysql-server
shell> bzr branch lp:mysql-server/5.6 mysql-5.6
```

これはソースツリーごとに一度だけの操作です。mysql-server ディレクトリで、複数のバージョンの MySQL のソースツリーをブランチできます。

接続の速度によっては、最初のダウンロードを完了するのにしばらく時間がかかります。最初のツリーをダウンロードすれば、追加のツリーのダウンロードにかかる時間は大幅に短くなるはずです。

3. Bazaar ブランチからビルドするときは、アクティブなブランチのコピーを作成すると、元のブランチの内容に影響を及ぼさずに構成その他の変更を行えます。これを行うには、元のブランチからブランチ処理を実行します。

```
shell> bzr branch mysql-5.6 mysql-5.6-build
```

4. ブランチを最初にセットアップしたあとの変更を取得するには、定期的に pull オプションを使用して更新します。このコマンドは、ローカルコピーのトップレベルディレクトリで使用します。

```
shell> bzr pull
```

ツリーの変更セットのコメントを調べるには、bzr に log オプションを使用します。

```
shell> bzr log
```

変更セット、コメント、およびソースコードは、Launchpad MySQL Server ページでオンラインでも参照できます。

diff (変更) や疑問のあるコードがある場合は、お気軽に電子メールを MySQL internals メーリングリストに送信してください。セクション1.5.1「MySQL メーリングリスト」 を参照してください。パッチの貢献については、「Contributing to MySQL Server」を参照してください。

5. ローカルブランチを作成したら、MySQL Server をソースコードからビルドできます。説明はセクション 2.9.2「標準ソース配布を使用して MySQL をインストールする」にありますが、配布の取得およびアンパックに関する部分はスキップします。

本番環境マシン上の配布ソースツリーからのビルドをインストールする場合は注意が必要です。インストールコマンドが、ライブリリースインストールを上書きすることがあります。MySQL がインストール済みであり、それを上書きしたくない場合は、CMAKE_INSTALL_PREFIX、MYSQL_TCP_PORT、および MYSQL_UNIX_ADDR オプションの値を、本番環境サーバーで使用したものとは異なるものにして、CMake を実行します。複数のサーバーが相互に干渉することを防ぐ方法の詳細は、セクション5.3「1 つのマシン上での複数の MySQL インスタンスの実行」を参照してください。

新しいインストールを厳密に検証します。たとえば、新機能をクラッシュさせてみてください。make test の実行から始めてみてください。セクション24.1.2「MySQL テストスイート」を参照してください。

2.9.4 MySQL ソース構成オプション

CMake プログラムを使用すると、MySQL ソース配布の構成方法を大幅に制御できます。通常これは、CMake コマンド行でオプションを使用して行います。CMake がサポートするオプションの詳細は、トップレベルのソースディレクトリで次のコマンドのいずれかを実行します。

```
shell> cmake . -LH
shell> ccmake .
```

特定の環境変数を使用して CMake に影響を与えることもできます。セクション2.12「環境変数」を参照してください。

次の表は、使用可能な CMake オプションを示しています。デフォルトカラムで、PREFIX は CMAKE_INSTALL_PREFIX オプションの値を意味します。これは、インストールのベースディレクトリを指定します。この値は、一部のインストールサブディレクトリの親の場所として使用されます。

表 2.11 MySQL ソース構成オプションリファレンス (CMake)

形式	説明	デフォルト	導入	削除
BUILD_CONFIG	公式なリリースと同じビルドオプションを使用			
CMAKE_BUILD_TYPE	生成するビルドのタイプ	RelWithDebInfo		
CMAKE_CXX_FLAGS	C++ コンパイラのフラグ			
CMAKE_C_FLAGS	C コンパイラのフラグ			
CMAKE_INSTALL_PREFIX	インストールの基本ディレクトリ	/usr/local/mysql		
COMPILATION_COMMENT	コンパイル環境に関するコメント			
CPACK_MONOLITHIC_INSTALL	パッケージビルドが単独のファイルを生成するかどうか	OFF		
DEFAULT_CHARSET	デフォルトのサーバー文字セット	latin1		
DEFAULT_COLLATION	デフォルトのサーバー照合順序	latin1_swedish_ci		
ENABLED_LOCAL_INFILE	LOAD DATA INFILE に LOCAL を有効にするかどうか	OFF		
ENABLED_PROFILING	クエリープロファイリングコードを有効にするかどうか	ON		
ENABLE_DEBUG_SYNC	Debug Sync サポートを有効にするかどうか	ON		
ENABLE_DOWNLOADS	オプションファイルをダウンロードするかどうか	OFF		
ENABLE_DTRACE	DTrace サポートをインクルードするかどうか			
ENABLE_GCOV	gcov サポートをインクルードするかどうか		5.6.3	
ENABLE_GPROF	gprof の有効化 (Linux ビルドのみに最適化)	OFF	5.6.6	
IGNORE_AIO_CHECK	-DBUILD_CONFIG=mysql_release とともに、libaio チェックを無視する	OFF	5.6.1	
INNODB_PAGE_ATOMIC_REF_COUNT	アトミックページ参照のカウントを有効または無効にする	ON	5.6.16	
INSTALL_BINDIR	ユーザー実行可能ファイルディレクトリ	PREFIX/bin		
INSTALL_DOCDIR	ドキュメントディレクトリ	PREFIX/docs		
INSTALL_DOCREADMEDIR	README ファイルディレクトリ	PREFIX		
INSTALL_INCLUDEDIR	Header ファイルディレクトリ	PREFIX/include		
INSTALL_INFODIR	Info ファイルディレクトリ	PREFIX/docs		
INSTALL_LAYOUT	事前定義インストールレイアウトの選択	STANDALONE		
INSTALL_LIBDIR	Library ファイルディレクトリ	PREFIX/lib		
INSTALL_MANDIR	Manual ページディレクトリ	PREFIX/man		
INSTALL_MYSQLSHAREDIR	共有データディレクトリ	PREFIX/share		
INSTALL_MYSQLTESTDIR	mysql-test ディレクトリ	PREFIX/mysql-test		
INSTALL_PLUGINDIR	Plugin ディレクトリ	PREFIX/lib/plugin		
INSTALL_SBINDIR	サーバー実行可能ファイルディレクトリ	PREFIX/bin		

形式	説明	デフォルト	導入	削除
INSTALL_SCRIPTDIR	Scripts ディレクトリ	PREFIX/scripts		
INSTALL_SHAREDIR	aclocal/mysql.m4 インストールディレクトリ	PREFIX/share		
INSTALL_SQLBENCHDIR	sql-bench ディレクトリ	PREFIX		
INSTALL_SUPPORTFILESDIR	その他のサポートファイルディレクトリ	PREFIX/support-files		
MEMCACHED_HOME	memcached へのパス	[none]		
MYSQL_DATADIR	データディレクトリ			
MYSQL_MAINTAINER_MODE	MySQL 管理者固有の開発環境を有効にするかどうか	OFF		
MYSQL_PROJECT_NAME	Windows/OS X プロジェクト名	3306	5.6.5	
MYSQL_TCP_PORT	TCP/IP ポート番号	3306		
MYSQL_UNIX_ADDR	Unix ソケットファイル	/tmp/mysql.sock		
ODBC_INCLUDES	ODBC インクルードディレクトリ			
ODBC_LIB_DIR	ODBC ライブラリディレクトリ			
OPTIMIZER_TRACE	オプティマイザのトレースをサポートするかどうか		5.6.3	
SUNPRO_CXX_LIBRARY	Solaris 10+ のクライアントリンクライブラリ		5.6.20	
SYSCONFDIR	オプションファイルディレクトリ			
TMPDIR	tmpdir のデフォルト値		5.6.16	
WITHOUT_SERVER	サーバーをビルドしない	OFF		
WITHOUT_xxx_STORAGE_ENGINE	ストレージエンジン xxx をビルドから除外			
WITH_ASAN	AddressSanitizer 有効	OFF	5.6.15	
WITH_BUNDLED_LIBEVENT	ndbmemcache のビルド時にバンドルされた libevent を使用	ON		
WITH_BUNDLED_MEMCACHED	ndbmemcache のビルド時にバンドルされた memcached を使用	ON		
WITH_CLASSPATH	Java 用 MySQL Cluster Connector のビルド時に使用するクラスパス。デフォルトは空の文字列です。			
WITH_DEBUG	デバッグサポートをインクルードするかどうか	OFF		
WITH_DEFAULT_COMPILER_OPTIONS	デフォルトのコンパイラオプションを使用するかどうか	ON	5.6.6	
WITH_DEFAULT_FEATURE_SET	デフォルトの機能セットを使用するかどうか	ON	5.6.6	
WITH_EDITLINE	どの libedit/editline ライブラリを使用するか	bundled	5.6.12	
WITH_EMBEDDED_SERVER	組み込みサーバーをビルドするかどうか	OFF		
WITH_EMBEDDED_SHARED_LIBRARY	共有組み込みサーバーライブラリをビルドするかどうか	OFF	5.6.17	
WITH_ERROR_INSERT	NDB ストレージエンジンのエラーインジェクションを有効	OFF		

形式	説明	デフォルト	導入	削除
	化。本番用のバイナリのビルドには使用しないでください。			
WITH_EXTRA_CHARSETS	どの追加文字セットをインクルードするか	all		
WITH_INNODB_MEMCACHED	memcached 共有ライブラリを生成するかどうか。	OFF		
WITH_LIBEDIT	バンドルされた libedit ライブラリを使用	ON		5.6.12
WITH_LIBEVENT	どの libevent ライブラリを使用するか	bundled	5.6.6	
WITH_LIBWRAP	libwrap (TCP ラッパー) サポートをインクルードするかどうか	OFF		
WITH_NDBCLUSTER_STORAGE_ENGINE	NDB ストレージエンジンのビルド	ON		
WITH_NDBMTD	マルチスレッドのデータノードをビルド。	ON		
WITH_NDB_BINLOG	mysqld によるバイナリのロギングをデフォルトで有効化。	ON		
WITH_NDB_DEBUG	テストまたはトラブルシューティング用のデバッグビルドを生成。	OFF		
WITH_NDB_JAVA	Java および ClusterJ のビルドのサポートを有効化。デフォルトで有効。MySQL Cluster のみでサポート。	ON		
WITH_NDB_PORT	このオプションでビルドされた管理サーバーが使用するデフォルトポート。ビルドにこのオプションが使用されなかった場合、管理サーバーのデフォルトポートは 1186 です。	[none]		
WITH_NDB_TEST	NDB API テストプログラムをインクルード。	OFF		
WITH_READLINE	バンドルされた readline ライブラリを使用	OFF		5.6.5
WITH_SSL	SSL サポートのタイプ	bundled		
WITH_UNIXODBC	unixODBC サポートの有効化	OFF		
WITH_VALGRIND	Valgrind ヘッダーファイルをコンパイルするかどうか	OFF		
WITH_ZLIB	zlib サポートのタイプ	system		
WITH_xxx_STORAGE_ENGINE	ストレージエンジン xxx をサーバーに静的にコンパイル			

次のセクションでは、CMake オプションについてより詳しく説明します。

- General Options
- Installation Layout Options
- Feature Options
- CMake Options for Compiling MySQL Cluster
- Compiler Flags

boolean オプションでは、値を 1 または ON に指定してオプションを有効にするか、0 または OFF に指定して無効にします。

多くのオプションはコンパイル時のデフォルトを構成し、それらはサーバー起動時にオーバーライドできます。たとえば、デフォルトのインストールベースディレクトリ、TCP/IP ポート番号、および Unix ソケットファイルを構成する CMAKE_INSTALL_PREFIX、MYSQL_TCP_PORT、および MYSQL_UNIX_ADDR の各オプションは、サーバー起動時に mysqld の --basedir、--port、および --socket オプションで変更できます。該当する場合は、構成オプションの説明で対応する mysqld スタートアップオプションを示します。

一般オプション

- -DBUILD_CONFIG=mysql_release

  このオプションは、オラクルが公式な MySQL リリースのバイナリ配布を生成するために使用するのと同じビルドオプションでソース配布を構成します。

- -DCMAKE_BUILD_TYPE=type

  生成するビルドのタイプ

  - RelWithDebInfo: 最適化を有効にし、デバッグ情報を生成します。これはデフォルトの MySQL ビルドタイプです。
  - Debug: 最適化を無効にし、デバッグ情報を生成します。このビルドタイプは、WITH_DEBUG オプションが有効な場合も使用されます。つまり、-DWITH_DEBUG=1 は -DCMAKE_BUILD_TYPE=Debug と同じ効果を持ちます。

- -DCPACK_MONOLITHIC_INSTALL=bool

  このオプションは、make package 操作が複数のインストールパッケージファイルを作成するか、単独のファイルを作成するかに影響します。無効の場合、この操作は複数のインストールパッケージファイルを作成します。これは完全な MySQL インストールのサブセットのみをインストールする場合に便利です。有効の場合、すべてをインストールするための単独のファイルを作成します。

インストールレイアウトオプション

CMAKE_INSTALL_PREFIX オプションは、ベースインストールディレクトリを示します。コンポーネントの場所を示す、INSTALL_xxx という形式の名前を持つその他のオプションは、プリフィクスに相対的なものとして解釈され、その値は相対パス名です。それらの値はプリフィクスを含みません。

- -DCMAKE_INSTALL_PREFIX=dir_name

  インストールのベースディレクトリ。

  この値は、サーバー起動時に --basedir オプションで設定できます。

- -DINSTALL_BINDIR=dir_name

  ユーザープログラムをインストールする場所。

- -DINSTALL_DOCDIR=dir_name

  ドキュメントをインストールする場所。

- -DINSTALL_DOCREADMEDIR=dir_name

  README ファイルをインストールする場所。

- -DINSTALL_INCLUDEDIR=dir_name

  ヘッダーファイルをインストールする場所。

- -DINSTALL_INFODIR=dir_name

  Info ファイルをインストールする場所。

- -DINSTALL_LAYOUT=name

  事前定義インストールレイアウトを選択します。

  - STANDALONE: .tar.gz および .zip パッケージで使用されるのと同じレイアウト。これはデフォルトです。
  - RPM: RPM パッケージと同様のレイアウト。

  - SVR4: Solaris パッケージレイアウト。
  - DEB: DEB パッケージレイアウト (実験的)。

  事前定義のレイアウトを選択できますが、ほかのオプションを指定することによって、個々のコンポーネントのインストール場所を変更できます。例:

```
shell> cmake . -DINSTALL_LAYOUT=SVR4 -DMYSQL_DATADIR=/var/mysql/data
```

- -DINSTALL_LIBDIR=dir_name

  ライブラリファイルをインストールする場所。

- -DINSTALL_MANDIR=dir_name

  マニュアルページをインストールする場所。

- -DINSTALL_MYSQLSHAREDIR=dir_name

  共有データファイルをインストールする場所。

- -DINSTALL_MYSQLTESTDIR=dir_name

  mysql-test ディレクトリをインストールする場所。MySQL 5.6.12 では、このディレクトリのインストールを抑制するには、オプションを明示的に空の値にセットします (-DINSTALL_MYSQLTESTDIR=)。

- -DINSTALL_PLUGINDIR=dir_name

  プラグインディレクトリの場所。

  この値は、サーバー起動時に --plugin_dir オプションで設定できます。

- -DINSTALL_SBINDIR=dir_name

  mysqld サーバーをインストールする場所。

- -DINSTALL_SCRIPTDIR=dir_name

  mysql_install_db をインストールする場所。

- -DINSTALL_SHAREDIR=dir_name

  aclocal/mysql.m4 をインストールする場所。

- -DINSTALL_SQLBENCHDIR=dir_name

  sql-bench ディレクトリをインストールする場所。このディレクトリのインストールを抑制するには、オプションを明示的に空の値にセットします (-DINSTALL_SQLBENCHDIR=)。

- -DINSTALL_SUPPORTFILESDIR=dir_name

  追加のサポートファイルをインストールする場所。

- -DMYSQL_DATADIR=dir_name

  MySQL データディレクトリの場所。

  この値は、サーバー起動時に --datadir オプションで設定できます。

- -DODBC_INCLUDES=dir_name

  ODBC インクルードディレクトリの場所。Connector/ODBC の構成中に使用されることがあります。

- -DODBC_LIB_DIR=dir_name

  ODBC ライブラリディレクトリの場所。Connector/ODBC の構成中に使用されることがあります。

- -DSYSCONFDIR=dir_name

  デフォルトの my.cnf オプションファイルディレクトリ。

この場所はサーバー起動時にはセットできませんが、--defaults-file=file_name オプションを使用して、指定されたオプションファイルでサーバーを起動できます。ここで、file_name はファイルへのフルパス名です。

- -DTMPDIR=dir_name

  tmpdir システム変数に使用されるデフォルトの場所。指定しない場合は、値はデフォルトで P_tmpdir in <stdio.h> になります。このオプションは MySQL 5.6.16 で追加されました。

ストレージエンジンオプション

ストレージエンジンはプラグインとしてビルドされます。プラグインは、静的モジュール (サーバー内にコンパイル) または動的モジュール (使用する前に、INSTALL PLUGIN ステートメントまたは --plugin-load オプションを使用してサーバーにインストールする必要のあるダイナミックライブラリとしてビルド) としてビルドできます。一部のプラグインは、静的または動的ビルドをサポートしない場合があります。

MyISAM、MERGE、MEMORY、および CSV エンジンは必須 (必ずサーバー内にコンパイル) で、明示的にインストールする必要はありません。

ストレージエンジンをサーバー内に静的にコンパイルするには、-DWITH_engine_STORAGE_ENGINE=1 を使用します。許可される engine の値は、ARCHIVE、BLACKHOLE、EXAMPLE、FEDERATED、INNOBASE (InnoDB)、NDB または NDBCLUSTER (NDB)、PARTITION (パーティション化のサポート)、および PERFSCHEMA (Performance Schema) です。例:

```
-DWITH_INNOBASE_STORAGE_ENGINE=1
-DWITH_ARCHIVE_STORAGE_ENGINE=1
-DWITH_BLACKHOLE_STORAGE_ENGINE=1
-DWITH_PERFSCHEMA_STORAGE_ENGINE=1
```

> 注記
>
> WITH_NDBCLUSTER_STORAGE_ENGINE は、MySQL Cluster を MySQL Cluster ソースを使用してビルドする場合のみサポートされます。ほかの MySQL ソースツリーまたは配布でのクラスタ化のサポートを有効にするために使用することはできません。MySQL Cluster ソース配布では、デフォルトで有効です。詳しくは、セクション18.2.2.3「Linux でのソースからの MySQL Cluster のビルド」およびセクション18.2.3.2「Windows でのソースからの MySQL Cluster のコンパイルとインストール」を参照してください。

ストレージエンジンをビルドから除外するには、-DWITHOUT_engine_STORAGE_ENGINE=1 を使用します。例:

```
-DWITHOUT_EXAMPLE_STORAGE_ENGINE=1
-DWITHOUT_FEDERATED_STORAGE_ENGINE=1
-DWITHOUT_PARTITION_STORAGE_ENGINE=1
```

あるストレージエンジンに対して -DWITH_engine_STORAGE_ENGINE も -DWITHOUT_engine_STORAGE_ENGINE も指定されていない場合、そのエンジンは共有モジュールとしてビルドされるか、あるいは共有モジュールとしてビルドできない場合は除外されます。

機能オプション

- -DCOMPILATION_COMMENT=string

  コンパイル環境に関する説明コメント。

- -DDEFAULT_CHARSET=charset_name

  サーバーの文字セット。デフォルトでは、MySQL は latin1 (cp1252 西部ヨーロッパ) 文字セットを使用します。

  charset_name は、binary、armscii8、ascii、big5、cp1250、cp1251、cp1256、cp1257、cp850、cp852、cp866、cp932、dec8、eucjpms、のいずれかにできます。許可される文字セットは、cmake/character_sets.cmake ファイルに CHARSETS_AVAILABLE の値としてリストされています。

  この値は、サーバー起動時に --character_set_server オプションで設定できます。

- -DDEFAULT_COLLATION=collation_name

  サーバーの照合順序。デフォルトでは、MySQL は latin1_swedish_ci を使用します。各文字セットにどの照合順序を使用するかを決めるには SHOW COLLATION ステートメントを使用します。

この値は、サーバー起動時に --collation_server オプションで設定できます。

- -DENABLE_DEBUG_SYNC=bool

  Debug Sync 機能をサーバーにコンパイルするかどうか。この機能はテストとデバッグに使用されます。このオプションはデフォルトで有効ですが、MySQL でデバッグが有効に構成されていない場合は効果はありません。デバッグが有効で、Debug Sync を無効にする場合は、-DENABLE_DEBUG_SYNC=0 を使用します。

  コンパイルされる場合、実行時には Debug Sync はデフォルトで無効です。有効にするには、mysqld を --debug-sync-timeout=N オプションを使用して起動します。ここで、N は 0 より大きいタイムアウト値です。(デフォルト値は 0 で、Debug Sync を無効にします。)N はそれぞれの同期ポイントのデフォルトのタイムアウトになります。

  Debug Sync 機能および同期点の使用方法についての説明は、「MySQL Internals: Test Synchronization」を参照してください。

- -DENABLE_DOWNLOADS=bool

  オプションファイルをダウンロードするかどうか。たとえば、このオプションを有効にすると CMake は、ユニットテストを実行するためにテストスイートが使用する Google Test 配布をダウンロードします。

- -DENABLE_DTRACE=bool

  DTrace プローブのサポートをインクルードするかどうか。DTrace については、セクション5.4「DTrace を使用した mysqld のトレース」を参照してください。

- -DENABLE_GCOV=bool

  gcov サポートをインクルードするかどうか (Linux のみ)。

- -DENABLE_GPROF=bool

  gprof を有効にするかどうか (最適化された Linux ビルドのみ)。このオプションは MySQL 5.6.6 で追加されました。

- -DENABLED_LOCAL_INFILE=bool

  LOCAL 機能を LOAD DATA INFILE のクライアントライブラリで有効にするかどうか。

  このオプションは、クライアント側の LOCAL 機能を制御しますが、この機能は --local-infile オプションを使用して、サーバー側でサーバーの起動時に設定できます。セクション6.1.6「LOAD DATA LOCAL のセキュリティーの問題」を参照してください。

- -DENABLED_PROFILING=bool

  クエリープロファイリングコードを有効にするかどうか (SHOW PROFILE および SHOW PROFILES ステートメントで)。

- -DIGNORE_AIO_CHECK=bool

  Linux で -DBUILD_CONFIG=mysql_release オプションが与えられた場合、デフォルトで libaio ライブラリがリンクされていなければなりません。libaio がない場合、またはインストールしない場合、-DIGNORE_AIO_CHECK=1 を指定するとそのチェックを抑制できます。このオプションは MySQL 5.6.1 で追加されました。

- -DINNODB_PAGE_ATOMIC_REF_COUNT=bool

  アトミックページ参照のカウントを有効にするか無効にするか。バッファープールからページをフェッチおよび解放してページの状態を追跡するのは、負荷が高く複雑な操作です。ページの相互排他ロックを使用してこれらの操作を追跡すると、うまく拡張しません。INNODB_PAGE_ATOMIC_REF_COUNT=ON (デフォルト) では、フェッチと解放は可能であればアトミックを使用して追跡されます。アトミッ

クをサポートしないプラットフォームでは、アトミックページ参照のカウントを無効にするように INNODB_PAGE_ATOMIC_REF_COUNT=OFF を設定します。

アトミックページ参照のカウントが有効の場合 (デフォルト)、サーバーのスタートアップ時に「[Note] InnoDB: Using atomics to ref count buffer pool pages」がエラーログに出力されます。アトミックページ参照のカウントが無効の場合、代わりに「[Note] InnoDB: Using mutexes to ref count buffer pool pages」が出力されます。

INNODB_PAGE_ATOMIC_REF_COUNT は、MySQL Bug #68079 の解決策として導入されました。このオプションは MySQL 5.7.5 では削除されました。アトミックのサポートは、MySQL を MySQL 5.7.5 としてビルドするために必要です。そのためこのオプションは非推奨です。

- -DMYSQL_MAINTAINER_MODE=bool

  MySQL 管理者固有の開発環境を有効にするかどうか。有効の場合、このオプションによりコンパイラの警告はエラーになります。

- -DMYSQL_PROJECT_NAME=name

  Windows または OS X では、プロジェクトファイル名に組み込むプロジェクト名。このオプションは MySQL 5.6.5 で追加されました。

- -DMYSQL_TCP_PORT=port_num

  サーバーが TCP/IP 接続を待機するポート番号。デフォルトは 3306 です。

  この値は、サーバー起動時に --port オプションで設定できます。

- -DMYSQL_UNIX_ADDR=file_name

  サーバーがソケット接続を待機する Unix ソケットファイルのパス。これは絶対パス名でなければなりません。デフォルトは /tmp/mysql.sock です。

  この値は、サーバー起動時に --socket オプションで設定できます。

- -DOPTIMIZER_TRACE=bool

  オプティマイザのトレースをサポートするかどうか。「MySQL Internals: Tracing the Optimizer」を参照してください。このオプションは MySQL 5.6.3 で追加されました。

- -DWITH_ASAN=bool

  AddressSanitizer を有効にするかどうか (サポートするコンパイラの場合)。デフォルトはオフです。このオプションは MySQL 5.6.15 で追加されました。

- -DWITH_DEBUG=bool

  デバッグサポートを含めるかどうか。

  デバッグサポートを有効にして MySQL を構成することにより、サーバーを起動するときに --debug="d,parser_debug" オプションを使用できるようになります。これにより、SQL ステートメントの処理に使用される Bison パーサーが、パーサートレースをサーバーの標準エラー出力にダンプします。一般的に、この出力はエラーログに書き込まれます。

- -DWITH_DEFAULT_FEATURE_SET=bool

  cmake/build_configurations/feature_set.cmake からのフラグを使用するかどうか。このオプションは MySQL 5.6.6 で追加されました。

- -DWITH_EDITLINE=value

  どの libedit/editline ライブラリを使用するか。許可される値は、bundled (デフォルト) および system です。

  WITH_EDITLINE は MySQL 5.6.12 で追加されました。これは、削除された WITH_LIBEDIT に代わるものです。

- -DWITH_EMBEDDED_SERVER=bool

  libmysqld 組み込みサーバーライブラリをビルドするかどうか。

- -DWITH_EMBEDDED_SHARED_LIBRARY=bool

共有 libmysqld 組み込みサーバーライブラリをビルドするかどうか。このオプションは MySQL 5.6.17 で追加されました。

- -DWITH_EXTRA_CHARSETS=name

  どの追加文字セットをインクルードするか。

  - all: すべての文字セット。これはデフォルトです。
  - complex: 複雑な文字セット。
  - none: 追加の文字セットなし。

- -DWITH_INNODB_MEMCACHED=bool

  memcached 共有ライブラリ (libmemcached.so および innodb_engine.so) を生成するかどうか。

- -DWITH_LIBEVENT=string

  どの libevent ライブラリを使用するか。許可される値は、bundled (デフォルト)、system、および yes です。system または yes を指定した場合、システム libevent ライブラリが存在すればそれが使用されます。システムライブラリが見つからない場合は、バンドルの libevent ライブラリが使用されます。libevent ライブラリは、InnoDB memcached が必要とします。

- -DWITH_LIBEDIT=bool

  配布にバンドルされている libedit ライブラリを使用するかどうか。

  WITH_LIBEDIT は、MySQL 5.6.12 で削除されました。代わりに WITH_EDITLINE を使用します。

- -DWITH_LIBWRAP=bool

  libwrap (TCP ラッパー) サポートを含めるかどうか。

- -DWITH_READLINE=bool

  配布にバンドルされている readline ライブラリを使用するかどうか。readline はバンドルされなくなったため、このオプションは MySQL 5.6.5 で削除されました。

- -DWITH_SSL={ssl_type|path_name}

- 含める SSL サポートのタイプ (ある場合) または使用する OpenSSL インストールへのパス名。

  - ssl_type には、次の値のいずれかを指定できます。

    - no: SSL サポートなし。これは MySQL 5.6.6 以前のデフォルトです。5.6.6 では、これは許可される値ではなくなりました。デフォルトは bundled です。
    - yes: システムの SSL ライブラリが存在すればそれを使用します。そうでない場合は配布にバンドルされたライブラリを使用します。
    - bundled: 配布にバンドルされた SSL ライブラリを使用します。これは MySQL 5.6.6 のデフォルトです。
    - system: システムの SSL ライブラリを使用します。

  - path_name, MySQL 5.6.7 以降で許可されます。使用する OpenSSL インストールへのパス名です。CMake が、システムにインストールされた古いまたは誤った OpenSSL バージョンを検出して使用するのを防ぐことができるため、system の ssl_type 値を使用するよりもこちらの方が推奨されます。(同じことをするためのもう 1 つの許可される方法は、CMAKE_PREFIX_PATH オプションを path_name にセットすることです。)

  SSL サポートの使用の詳細は、セクション6.3.10「セキュアな接続のための SSL の使用」を参照してください。

- -DWITH_UNIXODBC=1

  Connector/ODBC に関して、unixODBC サポートを有効にします。

- -DWITH_VALGRIND=bool

Valgrind ヘッダーファイルをコンパイルするかどうか。これにより、Valgrind API が MySQL コードに公開されます。デフォルトは OFF です。

Valgrind 対応のデバッグビルドを生成するには、-DWITH_VALGRIND=1 は通常 -DWITH_DEBUG=1 と組み合わされます。Building Debug Configurations を参照してください。

- -DWITH_ZLIB=zlib_type

  一部の機能では、サーバーが COMPRESS() および UNCOMPRESS() 関数などの圧縮ライブラリサポートでビルドされていること、およびクライアント/サーバープロトコルの圧縮を必要とします。WITH_ZLIB は zlib サポートのソースを示します。

  - bundled: 配布にバンドルされた zlib ライブラリを使用します。
  - system: システムの zlib ライブラリを使用します。これはデフォルトです。

- -DWITHOUT_SERVER=bool

  MySQL Server なしでビルドするかどうか。デフォルトは OFF で、サーバーをビルドします。

コンパイラフラグ

- -DCMAKE_C_FLAGS="flags"

  C コンパイラのフラグ。

- -DCMAKE_CXX_FLAGS="flags"

  C++ コンパイラのフラグ。

- -DWITH_DEFAULT_COMPILER_OPTIONS=bool

  cmake/build_configurations/compiler_options.cmake からのフラグを使用するかどうか。このオプションは MySQL 5.6.6 で追加されました。

  > **注記**
  >
  > すべての最適化フラグは、MySQL ビルドチームによって慎重に選択およびテストされています。オーバーライドすると予期しない結果になることがあります。自己責任において行なってください。

- -DSUNPRO_CXX_LIBRARY="lib_name"

  Solaris 10 以降で stlport4 の代わりに libCstd へのリンクを可能にします。サーバーは C++98 に依存するため、これはクライアントコードのみに機能します。使用例:

  ```
  cmake -DWITHOUT_SERVER=1 -DSUNPRO_CXX_LIBRARY=Cstd
  ```

  このオプションは MySQL 5.6.20 で追加されました。

独自の C および C++ コンパイラフラグを指定するには、最適化に影響しないフラグの場合は CMAKE_C_FLAGS および CMAKE_CXX_FLAGS CMake オプションを使用します。

独自のコンパイラフラグを提供する場合、CMAKE_BUILD_TYPE も指定するとよいでしょう。

たとえば、32 ビットリリースビルドを 64 ビット Linux マシンに作成するには次のようにします。

```
shell> mkdir bld
shell> cd bld
shell> cmake .. -DCMAKE_C_FLAGS=-m32 \
       -DCMAKE_CXX_FLAGS=-m32 \
       -DCMAKE_BUILD_TYPE=RelWithDebInfo
```

最適化に影響するフラグ (-Onumber) をセットする場合は、CMAKE_C_FLAGS_build_type および/または CMAKE_CXX_FLAGS_build_type オプションをセットする必要があります。ここで、build_type は CMAKE_BUILD_TYPE 値に対応します。デフォルトのビルドタイプ (RelWithDebInfo) に異なる最適化を指定するには、CMAKE_C_FLAGS_RELWITHDEBINFO および CMAKE_CXX_FLAGS_RELWITHDEBINFO オプションをセットします。たとえば、Linux で -O3 とデバッグシンボルを使用してコンパイルするには、次のようにします。

```
shell> cmake .. -DCMAKE_C_FLAGS_RELWITHDEBINFO="-O3 -g" \
       -DCMAKE_CXX_FLAGS_RELWITHDEBINFO="-O3 -g"
```

MySQL Cluster のコンパイルのための CMake オプション

次のオプションは、MySQL Cluster を MySQL Cluster ソースでビルドする場合に使用されます。現在、MySQL 5.6 Server ツリーからのソースを使用する場合は、サポートされていません。

- -DMEMCACHED_HOME=path

  path で示されるシステムディレクトリにインストールされた memcached (バージョン 1.6 以降) を使用してビルドを実行します。ビルドに使用されるこのインストールからのファイルは、memcached バイナリ、ヘッダーファイル、およびライブラリに加えて、memcached_utilities ライブラリおよびヘッダーファイル engine_testapp.h を含みます。

  ndbmemcache を、バンドルの memcached ソースを使用してビルドする場合 (WITH_BUNDLED_MEMCACHED オプション)、このオプションはセットしないでください。すなわち、デフォルトでバンドルのソースが使用されます。

  このオプションは MySQL Cluster NDB 7.2.2 で追加されました。

  SASL 承認および dtrace サポートの提供など、追加の CMake オプションは、外部ソースから memcached をコンパイルするときに使用可能ですが、これらのオプションは現在、MySQL Cluster にバンドルされる memcached ソースには有効にはなっていません。

- -DWITH_BUNDLED_LIBEVENT={ON|OFF}

  MySQL Cluster を ndbmemcached サポート (MySQL Cluster NDB 7.2.2 以降) でビルドする際には、MySQL Cluster ソースに含まれる libevent を使用します。デフォルトで有効。OFF にすると、システムの libevent が代わりに使用されます。

- -DWITH_BUNDLED_MEMCACHED={ON|OFF}

  MySQL Cluster ソースツリー (MySQL Cluster NDB 7.2.3 以降) に含まれる memcached ソースをビルドし、ndbmemcache エンジンをビルドするときにその結果の memcached サーバーを使用します。この場合、make install は memcached バイナリをインストールの bin ディレクトリに置き、ndbmemcache エンジン共有オブジェクトファイル ndb_engine.so をインストールの lib ディレクトリに置きます。

  このオプションはデフォルトで ON です。

- -DWITH_CLASSPATH=path

  Java 用 MySQL Cluster Connector のビルド時に使用するクラスパスをセットします。デフォルトは空です。MySQL Cluster NDB 7.2.9 以降では、-DWITH_NDB_JAVA=OFF が使用されている場合はこのオプションは無視されます。

- -DWITH_ERROR_INSERT={ON|OFF}

  NDB カーネルのエラーインジェクションを有効化します。テスト専用です。本番環境のバイナリのビルドに使用することは意図していません。デフォルトは OFF です。

- -DWITH_NDBCLUSTER_STORAGE_ENGINE={ON|OFF}

  mysqld で NDB (NDBCLUSTER) ストレージエンジンのサポートのビルドおよびリンク。デフォルトは ON です。

- -DWITH_NDBCLUSTER={ON|OFF}

  これは WITH_NDBCLUSTER_STORAGE_ENGINE のエイリアスです。

- -DWITH_NDBMTD={ON|OFF}

  マルチスレッドのデータノード実行ファイル ndbmtd をビルド。デフォルトは ON です。

- -DWITH_NDB_BINLOG={ON|OFF}

  このオプションを使用して、mysqld ビルド内でデフォルトでバイナリロギングを有効にします。デフォルトで ON です。

- -DWITH_NDB_DEBUG={ON|OFF}

MySQL Cluster バイナリのデバッグバージョンのビルドを有効にします。デフォルトで OFF です。

- -DWITH_NDB_JAVA={ON|OFF}

  ClusterJ を含め、MySQL Cluster の Java サポート付きでのビルドを有効にします。

  このオプションは MySQL Cluster NDB 7.2.9 で追加され、デフォルトで ON です。MySQL Cluster を Java サポート付きでコンパイルしない場合は、CMake の実行時に -DWITH_NDB_JAVA=OFF を指定することによって明示的に無効にする必要があります。そうしないと、Java が検出できない場合にビルドの構成が失敗します。

- -DWITH_NDB_PORT=port

  ビルドされた MySQL Cluster 管理サーバー (ndb_mgmd) が、この port をデフォルトで使用するようにします。このオプションがセットされていない場合、結果の管理サーバーはデフォルトでポート 1186 を使用しようとします。

- -DWITH_NDB_TEST={ON|OFF}

  有効の場合、NDB API テストプログラムをインクルードします。デフォルトは OFF です。

2.9.5 MySQL のコンパイルに関する問題

多くの問題の解決方法には再構成が含まれます。再構成を行う場合は、次に注意してください。

- CMake を以前に実行したあとで実行すると、以前の起動時に収集した情報を使用する場合があります。この情報は CMakeCache.txt に格納されています。CMake は、起動時にそのファイルを探し、存在すればその情報がまだ正しいと仮定して内容を読み込みます。この仮定は再構成した場合には無効です。
- CMake を実行するたびに、make を再実行して再コンパイルする必要があります。しかし、前のビルドの古いオブジェクトファイルが異なる構成オプションでコンパイルされている場合、それらを最初に削除する場合もあります。

古いオブジェクトファイルまたは構成情報が使用されることを予防するために、CMake を再実行する前に次のコマンドを実行します。

Unix の場合:

```
shell> make clean
shell> rm CMakeCache.txt
```

Windows の場合:

```
shell> devenv MySQL.sln /clean
shell> del CMakeCache.txt
```

ソースツリー外でビルドする場合、CMake を再実行する前にビルドディレクトリを削除して再作成します。ソースツリー外でのビルドに関する説明は、Build MySQL Server を CMake でビルドする方法を参照してください。

一部のシステムでは、システムインクルードファイルの違いにより警告が生じる場合があります。次のリストは、MySQL のコンパイル時にもっともよく生じることがよくわかっているその他の問題を記述しています。

- どの C および C++ コンパイラを使用するかを定義するために、CC および CXX 環境変数を使用できます。例:

  ```
  shell> CC=gcc
  shell> CXX=g++
  shell> export CC CXX
  ```

  独自の C および C++ コンパイラフラグを指定するには、CMAKE_C_FLAGS および CMAKE_CXX_FLAGS CMake オプションを使用します。Compiler Flagsを参照してください。

  指定する必要がある可能性のあるフラグを確認するには、mysql_config を、--cflags および --cxxflags オプションを使用して起動します。

- コンパイル段階でどのコマンドが実行されるかを確認するには、CMake を使用して MySQL を構成したあと、単なる make ではなく make VERBOSE=1 を実行します。
- コンパイルに失敗する場合は、MYSQL_MAINTAINER_MODE オプションが有効かどうかをチェックします。このモードでは、コンパイラの警告はエラーになるため、無効にすることによりコンパイルを続行できる場合があります。

- コンパイルが次のいずれかのエラーで失敗した場合、make のバージョンを GNU make にアップグレードする必要があります。

```
make: Fatal error in reader: Makefile, line 18:
Badly formed macro assignment
```

または:

```
make: file `Makefile' line 18: Must be a separator (:
```

または:

```
pthread.h: No such file or directory
```

Solaris および FreeBSD は、make プログラムに問題が多いことが知られています。

GNU make 3.75 は動作が確認されています。

- sql_yacc.cc ファイルは sql_yacc.yy から生成されます。通常、MySQL には事前生成された sql_yacc.cc が付属しているため、ビルドプロセスでコピーを作成する必要はありません。しかし、それを再度作成する必要がある場合、次のエラーに遭遇する場合もあります。

```
"sql_yacc.yy", line xxx fatal: default action causes potential...
```

これは yacc のバージョンに問題があることを意味しています。おそらく、代わりに bison (the GNU version of yacc) の最近のバージョンをインストールして使用する必要があります。

bison 1.75 以前のバージョンでは次のエラーがレポートされる場合があります。

```
sql_yacc.yy:#####: fatal error: maximum table size (32767) exceeded
```

テーブルの最大サイズを実際には超えていなくても、bison の旧バージョンのバグでエラーが発生します。

ツールの取得または更新の詳細は、セクション2.9「ソースから MySQL をインストールする」のシステム要件を参照してください。

2.9.6 MySQL の構成とサードパーティーツール

MySQL のソースから MySQL のバージョンを判断する必要のあるサードパーティーツールは、トップレベルソースディレクトリの VERSION ファイルを読み取ることができます。このファイルには、バージョンの各部分が個別にリストされています。たとえば、バージョンが MySQL 5.7.4-m14 の場合、このファイルは次のようになります。

```
MYSQL_VERSION_MAJOR=5
MYSQL_VERSION_MINOR=7
MYSQL_VERSION_PATCH=4
MYSQL_VERSION_EXTRA=-m14
```

ソースが General Availablility (GA) リリースでない場合は、MYSQL_VERSION_EXTRA 値は空ではありません。この例の場合、値はマイルストーン 14 に対応します。

バージョンのコンポーネントから 5 桁の数字を構成するには、次の式を使用します。

```
MYSQL_VERSION_MAJOR*10000 + MYSQL_VERSION_MINOR*100 + MYSQL_VERSION_PATCH
```

2.10 インストール後のセットアップとテスト

このセクションでは、Unix 類似システムでのインストール後の項目について説明します。Windows を使用している場合は、セクション2.3.8「Windows でのインストール後の手順」を参照してください。

MySQL のインストール後、対処するべき項目がいくつかあります。例:

- セクション2.10.1「Unix 類似システムでのインストール後の手順」で説明されているように、データディレクトリを初期化して MySQL 付与テーブルを作成してください。
- セキュリティー上の重要な問題点は、付与テーブル内の初期アカウントにはパスワードがないことです。MySQL Server への権限のないアクセスを防ぐにはパスワードを割り当てる必要があります。その手順は、セクション2.10.2「最初の MySQL アカウントのセキュリティー設定」を参照してください。
- オプションで、タイムゾーンテーブルを作成して名前付きタイムゾーンの認識を有効にできます。その手順は、セクション4.4.6「mysql_tzinfo_to_sql — タイムゾーンテーブルのロード」を参照してください。

- サーバーの起動に問題がある場合は、セクション2.10.1.3「MySQL サーバーの起動とトラブルシューティング」を参照してください。
- 新たにユーザーアカウントを作成する用意ができたら、セクション6.2「MySQL アクセス権限システム」およびセクション6.3「MySQL ユーザーアカウントの管理」にある MySQL アクセス管理システムおよびアカウント管理に関する情報を参照して、ユーザーアカウントを設定します。

2.10.1 Unix 類似システムでのインストール後の手順

Unix 類似システムへの MySQL のインストール後、付与テーブルを初期化してサーバーを起動し、サーバーが満足に機能することを確認する必要があります。サーバーを、システムの起動および停止とともに自動的に起動および停止させることもできます。付与テーブルでアカウントにパスワードの割り当ても行なってください。

Unix 類似システムでは、付与テーブルは mysql_install_db プログラムによってセットアップされます。一部のインストール方法では、既存のデータベースが検出できない場合このプログラムが自動的に実行されます。

- RPM 配布を使用して MySQL を Linux にインストールする場合、サーバー RPM が mysql_install_db を実行します。
- Debian Linux、Ubuntu Linux、Gentoo Linux などの多くのプラットフォームでは、ネイティブのパッケージングシステムを使用すると、mysql_install_db コマンドが自動的に実行されます。
- DMG 配布を使用して MySQL を OS X にインストールする場合、インストーラが mysql_install_db を実行します。

一般的なバイナリおよびソースインストールを含むその他のプラットフォームおよびインストールタイプでは、mysql_install_db を実行する必要があります。

次の手順は、付与テーブルを初期化して (まだ実行されていない場合)、サーバーを起動する方法を説明します。同時に、サーバーへのアクセスおよびサーバーの正常動作のテストに使用できるいくつかのコマンドも示してあります。サーバーの自動起動および停止に関する情報は、セクション2.10.1.2「MySQL を自動的に起動および停止する」を参照してください。

手順を完了してサーバーが実行できたら、mysql_install_db によって作成されたアカウントにパスワードを割り当て、テストデータベースへのアクセスを制限するとよいでしょう。その手順は、セクション2.10.2「最初の MySQL アカウントのセキュリティー設定」を参照してください。

次の例では、サーバーは mysql のログインアカウントのユーザー ID で動作します。これはそのようなアカウントが存在することを前提にしています。存在しない場合は作成するか、あるいはサーバーの実行に使用する計画の別の既存のログインアカウントの名前を代用します。アカウント作成の詳細は、セクション2.2「一般的なバイナリを使用した MySQL の Unix/Linux へのインストール」のmysql システムユーザーおよびグループの作成を参照してください。

1. MySQL インストールのトップレベルのディレクトリ (ここでは BASEDIR で示されます) に場所を変更します。

   ```
   shell> cd BASEDIR
   ```

   BASEDIR は、MySQL インスタンスのインストールディレクトリです。それは、/usr/local/mysql、/usr/local、または /usr/bin (MySQL Yum リポジトリまたはその他の方法によるインストールの場合、) などのようになります。次の手順では、このディレクトリに移動していることを前提とします。

   BASEDIR ディレクトリにはいくつかのファイルとサブディレクトリがあります。インストールにもっとも重要なものは bin サブディレクトリおよび scripts サブディレクトリです。

   - bin ディレクトリにはクライアントプログラムとサーバーが含まれています。シェルが MySQL プログラムを正しく見つけることができるように、このディレクトリの完全なパス名を PATH 環境変数に追加してください。セクション2.12「環境変数」を参照してください。
   - scripts ディレクトリには、サーバーアクセス権限を格納する付与テーブルを含む mysql データベースの初期化に使用される、mysql_install_db プログラムが含まれます。

2. 必要に応じて、mysql が配布のコンテンツにアクセスできることを確認します。配布を mysql としてインストールした場合は、追加のアクションは不要です。配布を root としてインストールした場合は、そのコンテンツは root が所有します。インストールディレクトリで次のコマンドを root として実行することで、所有権を mysql に変更します。最初のコマンドはファイルの所有者属性を mysql ユーザーに変更します。2 番目のコマンドはグループ属性を mysql グループに変更します。

```
shell> chown -R mysql .
shell> chgrp -R mysql .
```

3. 必要に応じて mysql_install_db プログラムを実行して、ユーザーによるサーバーへの接続許可を決定する権限を含む、初期 MySQL 付与テーブルをセットアップします。これは、インストール手順でこのプログラムが自動的に実行されないような配布タイプを使用した場合には必要になります。

   ```
   shell> scripts/mysql_install_db --user=mysql
   ```

   一般的には、mysql_install_db は MySQL を最初にインストールする際にのみ実行する必要があるため、既存のインストールをアップグレードした場合にはこのステップをスキップできます。しかし、mysql_install_db は既存の権限テーブルを上書きしないので、どのような場合でもこれを実行するのが安全でしょう。

   mysql_install_db がインストールディレクトリまたはデータディレクトリの正しい場所を特定しない場合は、--basedir や --datadir などのほかのオプションを指定する必要がある場合があります。例:

   ```
   shell> scripts/mysql_install_db --user=mysql \
          --basedir=/opt/mysql/mysql \
          --datadir=/opt/mysql/mysql/data
   ```

   mysql_install_db プログラムは、mysql を所有者としてサーバーのデータディレクトリを作成します。データディレクトリの下に、付与テーブルを保持する mysql データベースと MySQL のテストに使用できる test データベースが作成されます。そのスクリプトはまた、root および匿名ユーザーアカウントの権限テーブルのエントリを作成します。これらのアカウントには、最初パスワードがありません。セクション2.10.2「最初の MySQL アカウントのセキュリティー設定」で、初期権限について説明しています。瞬間的に、これらの権限により MySQL root ユーザーはあらゆることを実行可能であり、だれでも test という名前または test_ で始まる名前のデータベースを作成したり使用したりできます。付与テーブルの完全なリストと説明については、セクション6.2「MySQL アクセス権限システム」を参照してください。

   インストールをよりセキュアにするには、mysql_install_db を --random-passwords オプションで起動します。これにより、ランダムパスワードが MySQL root アカウントに割り当てられ、これらのアカウントに対して「有効期限切れパスワード」フラグが設定され、匿名ユーザー MySQL アカウントが削除されます。追加情報についてはセクション4.4.3「mysql_install_db — MySQL データディレクトリの初期化」を参照してください。(Unbreakable Linux Network での RPM を使用したインストール操作は mysql_install_db を使用しないため、影響されません。)

   あとでサーバーを起動したときに、サーバーがデータベースのディレクトリおよびファイルに対して読み取りアクセスおよび書き込みアクセスを持つように、それらが mysql のログインアカウントの所有になっていることを確認しておくことが重要です。これを確認するには、mysql_install_db を root として実行する場合は、示されるように --user オプションを含めます。そうでない場合は、mysql として口グインしている間にそのスクリプトを実行する必要があります。その場合 --user オプションをコマンドから除外できます。

   test データベースを保持しない場合は、サーバーを起動してから、セクション2.10.2「最初の MySQL アカウントのセキュリティー設定」の指示に従って削除できます。

   この段階で mysql_install_db の問題に遭遇した場合、セクション2.10.1.1「mysql_install_db 実行の問題」を参照してください。

4. ほとんどの MySQL インストールは、必要に応じて root を所有者にできます。例外は、データディレクトリを mysql が所有する必要があることです。これを行うには、次のコマンドをインストールディレクトリで root として実行します。

   ```
   shell> chown -R root .
   shell> chown -R mysql data
   ```

5. プラグインディレクトリ (plugin_dir システム変数によって指定されるディレクトリ) が、サーバーによる書き込みが可能な場合、ユーザーが SELECT ... INTO DUMPFILE を使用してそのディレクトリ内のファイルに実行可能なコードを書き込むことができる場合があります。これを防ぐために、plugin_dir をサーバーに対して読み取り専用にしたり、SELECT 書き込みが安全に実行できるディレクトリに --secure-file-priv を設定したりできます。

6. ソース配布を使用して MySQL をインストールした場合、提供される構成ファイルをオプションで support-files ディレクトリからご自身の /etc ディレクトリにコピーするとよいでしょう。さまざまなユースケース、サーバータイプ、および CPU と RAM の構成に対して、さまざまなサンプル構成ファイルがあります。これらの標準ファイルのいずれかを使用する場合、それを /etc/my.cnf または /etc/mysql/my.cnf にコピーして、最初に MySQL Server を起動する前に構成を編集してチェックするとよいでしょう。

   標準構成ファイルのいずれかをコピーしない場合、MySQL Server はデフォルト設定で起動されます。

マシンをブートしたときに MySQL を自動的に起動する場合は、support-files/mysql.server をシステムの起動ファイルがある場所にコピーします。詳細情報は、mysql.server スクリプト自体、およびセクション 2.10.1.2「MySQL を自動的に起動および停止する」にあります。

7. MySQL Server を起動します。

```
shell> bin/mysqld_safe --user=mysql &
```

MySQL Server を権限のない (非 root) ログインアカウントで起動することが重要です。これを確認するには、mysqld_safe を root として実行する場合は、示されるように --user オプションを含めます。または、mysql としてログインしている間にそのスクリプトを実行する必要があります。その場合 --user オプションをコマンドから除外できます。

MySQL を権限なしのユーザーとして実行する方法の詳細は、セクション6.1.5「MySQL を通常ユーザーとして実行する方法」を参照してください。

コマンドがただちに失敗して mysqld ended を出力する場合は、エラーログ (デフォルトではデータディレクトリ内の host_name.err ファイル) で情報を探してください。

この手順に進む前に mysql_install_db を実行して付与テーブルの作成を行わなかった場合は、サーバーの起動時にエラーログファイルに次のメッセージが現れます。

```
mysqld: Can't find file: 'host.frm'
```

このエラーは、mysql_install_db を root として実行し、--user オプションを使用しなかった場合にも生じます。data ディレクトリを削除して、前述のように mysql_install_db を --user オプションを使用して実行します。

サーバーの起動時にほかの問題が発生した場合には、セクション2.10.1.3「MySQL サーバーの起動とトラブルシューティング」を参照してください。mysqld_safe の詳細は、セクション4.3.2「mysqld_safe — MySQL サーバー起動スクリプト」を参照してください。

8. mysqladmin を使用してサーバーが動作していることを確認します。次のコマンドは、サーバーの起動および接続を確認する簡単なテストを提供します。

```
shell> bin/mysqladmin version
shell> bin/mysqladmin variables
```

mysqladmin version の出力は、プラットフォームおよび MySQL のバージョンによって多少異なりますが、次に示すものに類似します。

```
shell> bin/mysqladmin version
mysqladmin  Ver 14.12 Distrib 5.6.23, for pc-linux-gnu on i686
...

Server version          5.6.23
Protocol version        10
Connection              Localhost via UNIX socket
UNIX socket             /var/lib/mysql/mysql.sock
Uptime:                 14 days 5 hours 5 min 21 sec

Threads: 1  Questions: 366  Slow queries: 0
Opens: 0  Flush tables: 1  Open tables: 19
Queries per second avg: 0.000
```

mysqladmin のほかに機能を表示するには、それを --help オプションで起動します。

9. サーバーをシャットダウンできることを確認します。

```
shell> bin/mysqladmin -u root shutdown
```

10. サーバーを再度起動できることを確認します。これは、mysqld_safe を使用するか、あるいは mysqld を直接起動して行います。例:

```
shell> bin/mysqld_safe --user=mysql &
```

mysqld_safe が失敗する場合は、セクション2.10.1.3「MySQL サーバーの起動とトラブルシューティング」を参照します。

11. 簡単なテストをいくつか実行して、サーバーから情報を取り出せることを確認します。その出力は次に示すものと類似しているはずです。

```
shell> bin/mysqlshow
+--------------------+
|     Databases      |
+--------------------+
| information_schema |
| mysql              |
| test               |
+--------------------+

shell> bin/mysqlshow mysql
Database: mysql
+---------------------------+
|          Tables           |
+---------------------------+
| columns_priv              |
| db                        |
| event                     |
| func                      |
| help_category             |
| help_keyword              |
| help_relation             |
| help_topic                |
| host                      |
| plugin                    |
| proc                      |
| procs_priv                |
| servers                   |
| tables_priv               |
| time_zone                 |
| time_zone_leap_second     |
| time_zone_name            |
| time_zone_transition      |
| time_zone_transition_type |
| user                      |
+---------------------------+

shell> bin/mysql -e "SELECT Host,Db,User FROM db" mysql
+------+--------+------+
| host | db     | user |
+------+--------+------+
| %    | test   |      |
| %    | test_% |      |
+------+--------+------+
```

12. sql-bench ディレクトリ (MySQL インストールディレクトリの下) に、異なるプラットフォームでの MySQL の動作の違いを比較したベンチマークスイートがあります。ベンチマークスイートは Perl で書かれています。それにはさまざまなデータベースへのデータベース非依存型インタフェースを提供する Perl DBI、およびその他いくつかの Perl モジュールが必要です。

```
DBI
DBD::mysql
Data::Dumper
Data::ShowTable
```

これらのモジュールは CPAN (http://www.cpan.org/) で入手できます。セクション2.13.1「Unix に Perl をインストールする」も参照してください。

sql-bench/Results ディレクトリには、さまざまなデータベースおよびプラットフォームでの多くの実行結果が含まれています。すべてのテストを行うには、次のコマンドを実行します。

```
shell> cd sql-bench
shell> perl run-all-tests
```

sql-bench ディレクトリがない場合は、おそらくソース RPM 以外の RPM ファイルを使用して MySQL をインストールしました。(ソース RPM には sql-bench ベンチマークディレクトリが含まれています。)この場合、ベンチマークスイートを使用する前に、まずそれをインストールする必要があります。mysql-

bench-VERSION.i386.rpm の名前の個別のベンチマーク RPM ファイルがあり、その中にはベンチマークのコードおよびデータが含まれています。

ソース配布の場合には、テストも tests サブディレクトリにあり、実行できます。たとえば、auto_increment.tst を実行するには、このコマンドをソース配布のトップレベルのディレクトリから実行します。

```
shell> mysql -vvf test < ./tests/auto_increment.tst
```

想定されるテストの結果は ./tests/auto_increment.res ファイルにあります。

13. この段階では、サーバーが稼働しているはずです。しかし、初期 MySQL アカウントにはいずれもパスワードが設定されておらず、サーバーはテストデータベースへのアクセスを寛容に許可します。セキュリティーを厳しくするには、セクション2.10.2「最初の MySQL アカウントのセキュリティー設定」の手順に従ってください。

MySQL 5.6 インストール手順では、mysql データベースにタイムゾーンテーブルが作成されますが、データは移入されません。そのためには、セクション10.6「MySQL Server でのタイムゾーンのサポート」を参照してください。

インストールディレクトリの下の bin ディレクトリにインストールされたプログラムの起動をより便利にするために、そのディレクトリを PATH 環境変数設定に追加できます。それにより、プログラムのパス名全体ではなく名前のみを入力して実行できます。セクション4.2.10「環境変数の設定」を参照してください。

2.10.1.1 mysql_install_db 実行の問題

mysql_install_db プログラムの目的は、新しい MySQL 権限テーブルを生成することです。それは既存の MySQL 権限テーブルを上書きせず、ほかのデータに影響を及ぼすことはありません。

権限のテーブルを再度作成する場合は、mysqld サーバーが稼働していればそれを停止します。次にデータディレクトリの mysql ディレクトリを保存するために名前変更を行い、次に mysql_install_db を実行します。現在のディレクトリが MySQL のインストールディレクトリで、mysql_install_db が bin ディレクトリにあり、データディレクトリ名が data であるとします。mysql データベースの名前変更を行い mysql_install_db を実行するには、次のコマンドを使用します。

```
shell> mv data/mysql data/mysql.old
shell> scripts/mysql_install_db --user=mysql
```

mysql_install_db を実行すると、次の問題が発生する場合があります。

- mysql_install_db が付与テーブルのインストールに失敗する

  mysql_install_db が付与テーブルのインストールに失敗し、次のメッセージの表示後に終了する場合があります。

  ```
  Starting mysqld daemon with databases from XXXXXX
  mysqld ended
  ```

  この場合、エラーのログファイルを非常に慎重に調べる必要があります。ログは、エラーメッセージで指定されるディレクトリ XXXXXX にあるはずで、mysqld が起動しなかった理由を示しているはずです。どのような問題が発生したかわからない場合は、バグレポートを投稿するときにログを一緒に送ります。セクション1.6「質問またはバグをレポートする方法」を参照してください。

- mysqld プロセスが実行されている

  これはサーバーが稼働していることを示しています。この場合、付与テーブルはおそらくすでに作成されています。その場合、mysql_install_db を起動する必要があるのは一度だけ (最初に MySQL をインストールしたとき) であるため、それを起動する必要はまったくありません。

- 1 台のサーバーの起動中に 2 台目の mysqld サーバーのインストールができない

  これは既存の MySQL のインストールがあり、異なる場所に新たにインストールを行う場合に起こることがあります。たとえば、本番用のインストールがすでにあるが、テスト目的に 2 台目のインストールを作成する場合です。一般的に 2 台目のサーバーを起動しようとしたときに起こる問題は、1 台目のサーバーが使用しているネットワークインタフェースを 2 台目のサーバーが使用しようとした場合に発生します。この場合、次のエラーメッセージのいずれかが表示されるはずです。

  ```
  Can't start server: Bind on TCP/IP port:
  Address already in use
  ```

```
Can't start server: Bind on unix socket...
```

複数のサーバーの設定に関する説明は、セクション5.3「1 つのマシン上での複数の MySQL インスタンスの実行」を参照してください。

- /tmp ディレクトリに対する書き込みアクセス権がない

  デフォルトの場所 (/tmp ディレクトリ) または TMP_DIR 環境変数 (設定されている場合) に、一時ファイルまたは Unix ソケットファイルを作成するための書き込みアクセス権がない場合、mysql_install_db または mysqld サーバーを起動するとエラーが生じます。

  mysql_install_db あるいは mysqld を起動する前に次のコマンドを実行して、一時ディレクトリおよび Unix のソケットファイルに別の場所を指定できます。ここで、some_tmp_dir は書き込み許可のあるディレクトリへのフルパス名です。

```
shell> TMPDIR=/some_tmp_dir/
shell> MYSQL_UNIX_PORT=/some_tmp_dir/mysql.sock
shell> export TMPDIR MYSQL_UNIX_PORT
```

  そうすれば、mysql_install_db を起動して次のコマンドでサーバーを起動できるはずです。

```
shell> scripts/mysql_install_db --user=mysql
shell> bin/mysqld_safe --user=mysql &
```

  mysql_install_db が scripts ディレクトリにある場合には、最初のコマンドを scripts/mysql_install_db に変更します。

  セクションB.5.4.5「MySQL の UNIX ソケットファイルを保護または変更する方法」、およびセクション2.12「環境変数」を参照してください。

MySQL の配布で提供される mysql_install_db プログラムを実行するには、いくつかの方法があります。

- 初期権限を標準のデフォルトと異なるものにする場合、mysql_install_db を実行する前に変更できます。しかし、付与テーブルを設定したあとにGRANT および REVOKE を使用して権限を変更することが推奨されます。つまり、mysql_install_db を起動し、次に mysql -u root mysql を使用して MySQL root ユーザーとしてサーバーに接続します。これで必要な GRANT および REVOKE ステートメントを発行できます。

  同じ権限で複数のマシンに MySQL をインストールする場合、GRANT および REVOKE ステートメントを 1 つのファイルに入れ、mysql_install_db を起動後に、mysql を使用してそのファイルをスクリプトとして実行できます。例:

```
shell> scripts/mysql_install_db --user=mysql
shell> bin/mysql -u root < your_script_file
```

  このようにすることで、各マシンでステートメントを手動で発行する必要がなくなります。

- 付与テーブルを作成したあとで、完全に再作成できます。単に GRANT および REVOKE の使用方法を学習していて、mysql_install_db の実行後にあまりに多くの変更を行なったためテーブルを消去してやり直す場合には、このようにするとよいでしょう。

  付与テーブルを再度作成するには、mysql データベースディレクトリにあるすべての .frm、.MYI、および .MYD ファイルを削除します。次に、mysql_install_db プログラムを再度実行します。

- mysqld を --skip-grant-tables オプションを使用して手動で起動し、mysql を使用してご自身で権限情報を追加することができます。

```
shell> bin/mysqld_safe --user=mysql --skip-grant-tables &
shell> bin/mysql mysql
```

  mysql から、mysql_install_db の SQL コマンドを手動で実行します。mysqladmin flush-privileges あるいは mysqladmin reload を必ずあとで実行して、付与テーブルをリロードするようサーバーに指示します。

  mysql_install_db を使用しなかった場合は、付与テーブルに手動でデータを移入しなければならないのみではなく、まず付与テーブルを作成する必要があります。

2.10.1.2 MySQL を自動的に起動および停止する

一般的には、mysqld サーバーは次のいずれかの方法で起動します。

- 直接 mysqld を呼び出します。これはどのプラットフォームでも機能します。

- mysqld_safe を実行します。これは、mysqld の適切なオプションを判断してそれらのオプションで実行します。このスクリプトは Unix および Unix 類似システムで使用されます。セクション4.3.2「mysqld_safe — MySQL サーバー起動スクリプト」を参照してください。
- mysql.server を呼び出します。このスクリプトは、主に System V スタイルの実行ディレクトリ (すなわち、/etc/init.d および実行レベル固有のディレクトリ) を使用するシステムで、システムの起動時およびシャットダウン時に使用されます。通常、mysql という名前でインストールされます。mysql.server スクリプトは mysqld_safe を実行してサーバーを起動します。セクション4.3.3「mysql.server — MySQL サーバー起動スクリプト」を参照してください。
- OS X では、システム起動時の MySQL の自動起動を有効にする別個の MySQL Startup Item パッケージをインストールします。Startup Item は mysql.server を実行することでサーバーを起動します。詳細はセクション2.4.4「MySQL 起動アイテムのインストール」を参照してください。MySQL Preference Pane でも、System Preferences を介して MySQL の起動と停止を制御できます。セクション2.4.5「MySQL Preference Pane のインストールと使用」を参照してください。
- Solaris/OpenSolaris の SMF (service management framework) システムを使用して、MySQL の起動を開始および管理します。詳細については、セクション2.7.2「IPS を使用して MySQL を OpenSolaris にインストールする」を参照してください。

mysqld_safe スクリプトおよび mysql.server スクリプト、Solaris/OpenSolaris SMF、および OS X Startup Item (または MySQL Preference Pane) を使用して、サーバーを手動で、あるいはシステムの起動時に自動的に起動できます。mysql.server および Startup Item は、サーバーの停止にも使用できます。

mysql.server スクリプトを使用してサーバーを手動で起動または停止するには、サーバーを start 引数または stop 引数を使用して呼び出します。

```
shell> mysql.server start
shell> mysql.server stop
```

mysql.server は、サーバーを起動する前に場所を MySQL インストールディレクトリに変更し、次に mysqld_safe を実行します。サーバーを特定のユーザーとして起動するには、このセクションで後述するように、適切な user オプションを /etc/my.cnf オプションファイルの [mysqld] グループに追加します。(バイナリ配布の MySQL を標準と異なる場所にインストールした場合には mysql.server の編集が必要になる場合があります。mysqld_safe を実行する前に場所を適切なディレクトリに変更するように、それを修正します。これを行うと、将来 MySQL をアップグレードすると変更したバージョンの mysql.server が上書きされる場合がありますので、編集したバージョンを再インストールできるようにコピーを取っておくとよいでしょう。)

mysql.server stop は、サーバーに信号を送って停止します。mysqladmin shutdown を実行してサーバーを手動で停止することもできます。

サーバー上の MySQL を自動的に起動および停止するには、起動および停止のコマンドを /etc/rc* ファイルの適切な場所に加える必要があります。

Linux サーバーの RPM パッケージ (MySQL-server-VERSION.rpm) またはネイティブの Linux パッケージインストールを使用する場合は、mysql.server スクリプトは /etc/init.d のディレクトリに mysql という名前でインストールされる場合があります。Linux RPM パッケージに関する詳細は、セクション2.5.5「RPM パッケージを使用して MySQL を Linux にインストールする」を参照してください。

ベンダーによっては、起動スクリプトを mysqld のような別名でインストールする RPM パッケージを提供している場合もあります。

MySQL をソース配布から、あるいは mysql.server を自動的にインストールしないバイナリの配布形式を使用してインストールする場合、それを手動でインストールできます。そのスクリプトは、MySQL インストールディレクトリの support-files ディレクトリあるいは MySQL のソースツリーにあります。

mysql.server を手動でインストールするには、それを /etc/init.d ディレクトリに mysql の名前でコピーし、次にそれを実行ファイルにします。それには、mysql.server のある適切なディレクトリに場所を変更して、次のコマンドを実行します。

```
shell> cp mysql.server /etc/init.d/mysql
shell> chmod +x /etc/init.d/mysql
```

> **注記**
>
> 古い Red Hat システムは /etc/init.d ではなく /etc/rc.d/init.d ディレクトリを使用しています。前述のコマンドを適宜調整します。または、最初に /etc/init.d を /etc/rc.d/init.d を指すシンボリックリンクとして作成します。

```
shell> cd /etc
shell> ln -s rc.d/init.d .
```

スクリプトをインストールしたあと、それを有効にしてシステムの起動時に実行するために必要なコマンドは、使用しているオペレーティングシステムによって異なります。Linux では、chkconfig を使用します。

```
shell> chkconfig --add mysql
```

一部の Linux システムでは、mysql スクリプトを完全に有効にするには次のコマンドも必要になる場合があります。

```
shell> chkconfig --level 345 mysql on
```

FreeBSD では、起動スクリプトは通常 /usr/local/etc/rc.d/ にあります。rc(8) のマニュアルページでは、このディレクトリのスクリプトはそれらのベース名が *.sh シェルファイル名のパターンに一致したときのみ実行できると記載されています。そのディレクトリの、その他のファイルあるいおよびディレクトリは警告なしで無視されます。つまり FreeBSD では、自動起動を有効にするには mysql.server スクリプトを /usr/local/etc/rc.d/mysql.server.sh としてインストールするべきです。

前述の設定の代案として、オペレーティングシステムの中には /etc/rc.local あるいは /etc/init.d/boot.local を使用して起動時に追加のサービスを起動するものもあります。この方法で MySQL を起動するには、次のようなコマンドを適切な起動ファイルに追加します。

```
/bin/sh -c 'cd /usr/local/mysql; ./bin/mysqld_safe --user=mysql &'
```

ほかのシステムの起動スクリプトのインストール方法についてはそのオペレーティングシステムのドキュメントをお読みください。

mysql.server のオプションを、グローバル /etc/my.cnf ファイルに追加できます。一般的な /etc/my.cnf ファイルは次のようになります。

```
[mysqld]
datadir=/usr/local/mysql/var
socket=/var/tmp/mysql.sock
port=3306
user=mysql

[mysql.server]
basedir=/usr/local/mysql
```

The mysql.server スクリプトは、basedir、datadir、および pid-file の各オプションをサポートしています。指定する場合は、コマンド行ではなくオプションファイルに配置する必要があります。mysql.server は、start および stop のみをコマンド行の引数としてサポートしています。

次の表は、サーバーおよび各起動スクリプトがオプションファイルから読み取るオプショングループを示しています。

表 2.12 MySQL 起動スクリプトとサポートされるサーバーオプショングループ

スクリプト	オプショングループ
mysqld	[mysqld]、[server]、[mysqld-major_version]
mysqld_safe	[mysqld]、[server]、[mysqld_safe]
mysql.server	[mysqld]、[mysql.server]、[server]

[mysqld-major_version] は、[mysqld-5.5] および [mysqld-5.6] のような名前のグループが、5.5.x、5.6.x、などのバージョンのサーバーによって読み取られることを意味します。この機能は、所定のリリースシリーズのサーバーによってのみ読み取られるオプションを指定するために使用できます。

下位互換性のため、mysql.server は[mysql_server] グループも読み取り、mysqld_safe は[safe_mysqld] グループも読み取ります。ただし、MySQL 5.6 を使用する場合は、代わりに [mysql.server] および [mysqld_safe] グループを使用するようにオプションファイルを更新するとよいでしょう。

MySQL 構成ファイルとその構造とコンテンツについては、セクション4.2.6「オプションファイルの使用」を参照してください。

2.10.1.3 MySQL サーバーの起動とトラブルシューティング

このセクションでは、Unix 類似システムでのサーバー起動の問題に関するトラブルシューティングの提案を示します。Windows を使用している場合には、セクション2.3.6「Microsoft Windows MySQL Server インストールのトラブルシューティング」を参照してください。

サーバーの起動時に問題がある場合、次を試してみます。

- エラーログをチェックして、サーバーが起動しない原因を確認します。ログファイルはデータディレクトリ (通常、Windows では C:\Program Files\MySQL\MySQL Server 5.6\data、Unix/Linux バイナリ配布では /usr/local/mysql/data、Unix/Linux ソース配布では /usr/local/var) にあります。データディレクトリの host_name.err および host_name.log の形式のファイル名のファイルを探します。ここで、host_name はサーバーホストの名です。次にこれらのファイルの最後の数行を調べます。それらを表示するには tail を使用します。

  ```
  shell> tail host_name.err
  shell> tail host_name.log
  ```

- 使用しているストレージエンジンに必要なオプションがあれば指定します。my.cnf ファイルを作成して、使用するエンジンの起動オプションを指定します。トランザクションテーブルをサポートするストレージエンジン (InnoDB、NDB) を使用する予定である場合、サーバーを起動する前に、それらが適切に構成されていることを確認します。InnoDB テーブルを使用している場合は、ガイドラインについてはセクション14.3「InnoDB の構成」、オプションの構文についてはセクション14.12「InnoDB の起動オプションおよびシステム変数」を参照してください。

  ストレージエンジンは、省略したオプションについてはデフォルト値を使用しますが、使用可能なオプションを確認し、デフォルト値がインストールに対して適切でないようなオプションがあれば明示的に値を指定することを推奨します。

- サーバーが、データディレクトリを検索する場所を認識していることを確認してください。mysqld サーバーはこのディレクトリを現在のディレクトリとして使用します。そこでデータベースを探し、ログファイルに書き込むことを想定しています。サーバーはデータディレクトリで pid (プロセス ID) ファイルも書き込みます。

  デフォルトのデータディレクトリの場所は、サーバーのコンパイル時にハードコードされます。デフォルトのパス設定を確認するには、mysqld を --verbose オプションおよび --help オプションで起動します。データディレクトリがシステム上の別の場所にある場合は、その場所を、コマンド行またはオプションファイルで --datadir オプションを使用して mysqld または mysqld_safe に指定します。そうしないと、サーバーは適切に動作しません。--datadir オプションの代替として、MySQL がインストールされているベースディレクトリの場所を、mysqld に --basedir で指定できます。そうすれば、mysqld はそこで data ディレクトリを検索します。

  パスオプションの指定の結果を知るには、mysqld をそれらのオプションで実行し次に--verbose および --help オプションを実行します。たとえば、場所を mysqld をインストールしたディレクトリに変更してから次のコマンドを実行すると、ベースディレクトリ /usr/local でサーバーを起動した結果が表示されます。

  ```
  shell> ./mysqld --basedir=/usr/local --verbose --help
  ```

  --datadir のようなオプションも同様に指定できますが、--verbose および --help は最後のオプションでなければなりません。

  任意のパスを設定したあと、サーバーを --verbose および --help を使用しないで起動します。

  mysqld が動作しているときは、次のコマンドを実行してどのパス設定が使用されているか確認できます。

  ```
  shell> mysqladmin variables
  ```

  または:

  ```
  shell> mysqladmin -h host_name variables
  ```

  host_name は MySQL サーバーのホスト名です。

- サーバーが、データディレクトリにアクセスできることを確認してください。データディレクトリとそのコンテンツの所有権と許可は、サーバーがそれらを読み取って修正できるようなものでなければなりません。

  mysqld の起動時に Errcode 13 (許可が却下されたことを意味します) を受け取る場合は、データディレクトリまたはそのコンテンツの特権が、サーバーアクセスを許可しないものであることを意味します。この場合、関連するファイルおよびディレクトリの権限を変更して、サーバーがそれらを使用する権限を持つようにします。サーバーを root から起動することもできますが、この場合セキュリティーの問題が生じるため、避けるべきです。

場所をデータディレクトリに変更し、データディレクトリとそのコンテンツの所有権をチェックして、サーバーにアクセス権があることを確認します。たとえば、データディレクトリが /usr/local/mysql/var の場合は、次のコマンドを使用します。

```
shell> ls -la /usr/local/mysql/var
```

データディレクトリあるいはそのファイルまたはサブディレクトリが、サーバーを稼働するためのログインアカウントの所有でない場合、それらの所有権をそのアカウントに変更します。そのアカウントが mysql の場合、次のコマンドを使用します。

```
shell> chown -R mysql /usr/local/mysql/var
shell> chgrp -R mysql /usr/local/mysql/var
```

所有権が適切な場合でも、ファイルシステムのさまざまな部分へのアプリケーションのアクセスを管理するようなセキュリティーソフトウェアがシステム上で実行されている場合、MySQL が起動に失敗することがあります。この場合は、そのソフトウェアを再構成して mysqld が通常の動作中に使用するディレクトリにアクセスできるようにします。

- サーバーが使用するネットワークインタフェースが利用できることを確認します。

  次のいずれかのエラーが発生した場合、ほかのプログラム (おそらく別の mysqld サーバー) が、mysqld が使用する TCP/IP ポートあるいは Unix のソケットファイルを使用していることを意味します。

  ```
  Can't start server: Bind on TCP/IP port: Address already in use
  Can't start server: Bind on unix socket...
  ```

  ps を使用して、別の mysqld サーバーが稼働しているかどうか判断します。その場合、mysqld を再度起動する前にサーバーをシャットダウンします。(別のサーバーが動作中で、複数のサーバーを稼働させる必要がある場合は、その方法に関する情報は、セクション5.3「1 つのマシン上での複数の MySQL インスタンスの実行」にあります。)

  別のサーバーが稼働していない場合は、コマンド telnet your_host_name tcp_ip_port_number を実行します。(デフォルトの MySQL ポート番号は 3306 です。)次に Enter を数回押します。telnet: Unable to connect to remote host: Connection refused などのエラーメッセージが表示されない場合は、mysqld が使用しようとしている TCP/IP ポートをほかのプログラムが使用しています。それがどのプログラムなのかを追跡して無効にするか、--port オプションを使用して、mysqld に別のポートで待機するよう指示します。この場合、TCP/IP を使用してサーバーに接続するときに、クライアントプログラムに同じデフォルト以外のポート番号を指定します。

  ポートにアクセスできない別の理由に、ファイアウォールがその接続をブロックしている場合があります。その場合は、ファイアウォールの設定を、ポートへのアクセスを許可するように変更します。

  サーバーは起動するがそれに接続できない場合は、/etc/hosts に次のようなエントリがあることを確認します。

  ```
  127.0.0.1       localhost
  ```

- mysqld を起動できない場合は、--debug オプションを使用して、トレースファイルを作成して問題を発見してみてください。セクション24.4.3「DBUG パッケージ」を参照してください。

2.10.2 最初の MySQL アカウントのセキュリティー設定

MySQL インストールプロセスの一環として、付与テーブルを含む mysql データベースを設定します。

- Windows 配布は事前に初期化された付与テーブルを含みます。
- Unix では、mysql_install_db プログラムが付与テーブルに移入します。一部のインストール方法ではこのプログラムは自動的に実行されます。その他では、手動でそれを実行する必要があります。詳細については、セクション2.10.1「Unix 類似システムでのインストール後の手順」を参照してください。

mysql.user 付与テーブルは、初期 MySQL ユーザーアカウントとそのアクセス権限を定義します。

- いくつかのアカウントはユーザー名が root です。これらは、すべての権限を持ち、どのようなことも可能なスーパーユーザーアカウントです。root アカウントの初期パスワードは空です。そのため、だれでもパスワードなしで root として MySQL Server に接続して、すべての権限を付与されることができます。
  - Windows では、ローカルホストからの接続のみを許可する root アカウントが作成されます。ホスト名 localhost、IP アドレス 127.0.0.1、または IPv6 アドレス ::1 を指定することで接続できます。ユーザーが、インストール中に「Enable root access from remote machines」オプションを選択した場合は、Windows インストーラは、に荷のホストからの接続を許可する別の root アカウントを作成します。

  - Unix では、各 root アカウントはローカルホストからの接続を許可します。ホスト名 localhost、IP アドレス 127.0.0.1、または IPv6 アドレス ::1、あるいは実際のホスト名または IP アドレスを指定することで接続できます。

  ホスト 127.0.0.1 への接続を試みると、通常 localhost アカウントに解決します。ただし、サーバーが --skip-name-resolve オプションを使用して稼働されている場合は、これは失敗します。そのため、その場合には 127.0.0.1 アカウントが便利です。::1 アカウントは IPv6 接続に使用されます。

- 一部のアカウントは匿名ユーザー用です。これらは空のユーザー名を持ちます。匿名のアカウントにはパスワードがないため、だれでもそのアカウントを使用して MySQL サーバーに接続できます。

  - Windows では、ローカルホストからの接続を許可する匿名アカウントが 1 つあります。localhost のホスト名を指定することで接続できます。

  - Unix では、各匿名アカウントはローカルホストからの接続を許可します。アカウントの 1 つのホスト名 localhost を指定するか、その他のアカウントの実際のホスト名または IP アドレスを指定することで接続できます。

mysql.user テーブルにどのアカウントが存在するか、それらのパスワードが空かどうかを表示するには、次のステートメントを使用します。

```
mysql> SELECT User, Host, Password FROM mysql.user;
+------+-------------------+----------+
| User | Host              | Password |
+------+-------------------+----------+
| root | localhost         |          |
| root | myhost.example.com |          |
| root | 127.0.0.1         |          |
| root | ::1               |          |
|      | localhost         |          |
|      | myhost.example.com |          |
+------+-------------------+----------+
```

この出力は、いくつかの root アカウントと匿名ユーザーアカウントがあり、いずれにもパスワードがないことを示しています。使用しているシステムでは出力が異なる場合がありますが、空のパスワードのアカウントが存在するということは、何らかの対処をするまで MySQL インストールが保護されていないということを意味します。

- それぞれの MySQL root アカウントにパスワードを割り当てるべきです。

- クライアントが匿名ユーザーとしてパスワードなしで接続することを防ぐには、それぞれの匿名のアカウントにパスワードを割り当てるか、あるいはそれらのアカウントを削除するとよいでしょう。

さらに、mysql.db テーブルにはすべてのアカウントが test データベースおよび test_ で始まる名前を持つその他のデータベースにアクセスすることを許可する行が含まれます。これは、デフォルトの匿名アカウントのように、そうでなければ特別な権限を持たないアカウントにも当てはまります。これはテスト用には便利ですが、本番サーバーでは推奨されません。データベースへのアクセスを、その目的のために明示的に許可を付与されたアカウントのみに制限する場合は、管理者は mysql.db テーブルのこれらの行を削除するとよいでしょう。

次の手順では、初期 MySQL アカウントにパスワードを設定する方法を、最初に root アカウント、次に匿名アカウントの順で説明します。この手順では、匿名アクセスをまったく許可しない場合に、匿名アカウントを削除する方法にも触れ、データベースをテストするための寛容なアクセスを削除する方法も説明します。例の中の newpwd は使用するパスワードに置き換えてください。host_name はサーバーのホスト名に置き換えます。この名前は、前述の SELECT ステートメントの出力から判断できます。示された出力では、host_name は myhost.example.com です。

> **注記**
>
> パスワード設定の詳細は、セクション6.3.5「アカウントパスワードの割り当て」を参照してください。root パスワードを設定したあとでそれを忘れた場合は、セクション B.5.4.1「root のパスワードをリセットする方法」を参照してください。

追加のセットアップやテストを実行する間、パスワードを指定する必要を避けるために、パスワードの設定を遅らせたい場合があります。しかし、インストールを本番用に使用する前にはそれらを忘れずに設定してください。

追加のアカウントをセットアップするには、セクション6.3.2「ユーザーアカウントの追加」を参照してください。

root アカウントのパスワードの割り当て

root アカウントのパスワードは、いくつかの方法でセットできます。次の説明では 3 種類の方法を示します。

- SET PASSWORD ステートメントを使用する
- UPDATE ステートメントを使用する
- mysqladmin コマンド行のクライアントプログラムを使用する

SET PASSWORD を使用してパスワードを割り当てるには、サーバーに root として接続し、SET PASSWORD ステートメントを、mysql.user テーブルにリストされている各 root アカウントに対して発行します。PASSWORD() 関数を使用してパスワードを忘れずに暗号化します。

Windows では、次のようにします。

```
shell> mysql -u root
mysql> SET PASSWORD FOR 'root'@'localhost' = PASSWORD('newpwd');
mysql> SET PASSWORD FOR 'root'@'127.0.0.1' = PASSWORD('newpwd');
mysql> SET PASSWORD FOR 'root'@'::1' = PASSWORD('newpwd');
mysql> SET PASSWORD FOR 'root'@'%' = PASSWORD('newpwd');
```

mysql.user テーブルに、ホスト値が % である root アカウントがない場合は、最後のステートメントは不要です。

Unix では、次のようにします。

```
shell> mysql -u root
mysql> SET PASSWORD FOR 'root'@'localhost' = PASSWORD('newpwd');
mysql> SET PASSWORD FOR 'root'@'127.0.0.1' = PASSWORD('newpwd');
mysql> SET PASSWORD FOR 'root'@'::1' = PASSWORD('newpwd');
mysql> SET PASSWORD FOR 'root'@'host_name' = PASSWORD('newpwd');
```

UPDATE を使用して mysql.user テーブルを直接変更することで、すべての root アカウントにパスワードを割り当てる単独のステートメントを使用することもできます。この方法はどのプラットフォームでも機能します。

```
shell> mysql -u root
mysql> UPDATE mysql.user SET Password = PASSWORD('newpwd')
    ->     WHERE User = 'root';
mysql> FLUSH PRIVILEGES;
```

FLUSH ステートメントにより、サーバーが付与テーブルを再度読み取ります。それがないと、パスワードの変更はサーバーを再起動するまでサーバーには認識されません。

パスワードを root アカウントに mysqladmin を使用して割り当てるには、次のコマンドを実行します。

```
shell> mysqladmin -u root password "newpwd"
shell> mysqladmin -u root -h host_name password "newpwd"
```

これらのコマンドは Windows および Unix の両方に該当します。パスワードを囲む二重引用符は必ずしも常に必要なわけではありませんが、パスワードにスペースその他のコマンドインタプリタにとって特殊な文字が含まれる場合は使用するべきです。

mysqladmin を使用して root アカウントのパスワードを設定する方法は、'root'@'127.0.0.1' or 'root'@'::1' アカウントでは機能しません。前述の SET PASSWORD の方法を使用します。

root の設定後は、root としてサーバーに接続するたびに適切なパスワードを提供する必要があります。たとえば、mysqladmin でサーバーをシャットダウンするには、次のコマンドを使用します。

```
shell> mysqladmin -u root -p shutdown
Enter password: (enter root password here)
```

匿名アカウントのパスワードの割り当て

次の手順の mysql コマンドは、前述の手順を使用して root のパスワードを設定し、サーバーへの接続時にそのパスワードを指定しなければならないことを前提として、-p オプションを含んでいます。

匿名アカウントにパスワードを割り当てるには、サーバーに root として接続してから、SET PASSWORD または UPDATE を使用します。PASSWORD() 関数を使用してパスワードを忘れずに暗号化します。

Windows で SET PASSWORD を使用するには、次のようにします。

```
shell> mysql -u root -p
```

```
Enter password: (enter root password here)
mysql> SET PASSWORD FOR ''@'localhost' = PASSWORD('newpwd');
```

Unix で SET PASSWORD を使用するには、次のようにします。

```
shell> mysql -u root -p
Enter password: (enter root password here)
mysql> SET PASSWORD FOR ''@'localhost' = PASSWORD('newpwd');
mysql> SET PASSWORD FOR ''@'host_name' = PASSWORD('newpwd');
```

単独の UPDATE ステートメントで匿名ユーザーアカウントのパスワードを設定するには、次のようにします (任意のプラットフォームで)。

```
shell> mysql -u root -p
Enter password: (enter root password here)
mysql> UPDATE mysql.user SET Password = PASSWORD('newpwd')
    ->     WHERE User = '';
mysql> FLUSH PRIVILEGES;
```

FLUSH ステートメントにより、サーバーが付与テーブルを再度読み取ります。それがないと、パスワードの変更はサーバーを再起動するまでサーバーには認識されません。

匿名アカウントの削除

匿名アカウントにパスワードを割り当てるのではなく、削除する場合は、Windows では次のようにします。

```
shell> mysql -u root -p
Enter password: (enter root password here)
mysql> DROP USER ''@'localhost';
```

Unix では、次のように匿名アカウントを削除します。

```
shell> mysql -u root -p
Enter password: (enter root password here)
mysql> DROP USER ''@'localhost';
mysql> DROP USER ''@'host_name';
```

テストデータベースのセキュリティー設定

デフォルトでは、mysql.db テーブルには任意のユーザーによる test データベースおよび test_ で始まる名前を持つその他のデータベースへのアクセスを許可する行が含まれます。(これらの行は空の User カラム値を持ち、アクセスチェックのために任意のユーザー名に一致します。)これは、そうでなければ何の権限も持たないアカウントでさえも、このようなデータベースを使用できることを意味します。テストデータベースへの任意のユーザーによるアクセスを削除する場合は、次のようにします。

```
shell> mysql -u root -p
Enter password: (enter root password here)
mysql> DELETE FROM mysql.db WHERE Db LIKE 'test%';
mysql> FLUSH PRIVILEGES;
```

FLUSH ステートメントにより、サーバーが付与テーブルを再度読み取ります。それがないと、権限の変更はサーバーを再起動するまでサーバーには認識されません。

前述の変更により、グローバルデータベース権限、または test データベース用に明示的に付与された権限を持つユーザーのみがそれを使用できます。ただし、データベースがまったく不要である場合は、ドロップします。

```
mysql> DROP DATABASE test;
```

> **注記**
>
> Windows では、MySQL Installer (セクション2.3.3「MySQL Installer を使用した MySQL の Microsoft Windows へのインストール」を参照してください) でのインストール中に、このセクションで説明した手順を実行することもできます。すべてのプラットフォームで、MySQL 配布には、MySQL インストールをセキュアにするプロセスの大部分を自動化するコマンド行ユーティリティー mysql_secure_installation が含まれます。また、MySQL Workbench はすべてのプラットフォームで使用可能で、ユーザーアカウントの管理機能も提供します (第26章「MySQL Workbench」を参照してください)。

2.11 MySQL のアップグレードとダウングレード

このセクションでは、MySQL インストールのアップグレードまたはダウングレードの手順を説明します。

MySQL の同じリリースシリーズ内でのバグフィックスや、MySQL の主なリリース間の重大な機能を入手する場合など、アップグレードは一般的な手順です。この手順は、テストシステムでまず実行して、すべてが問題なく動作することを確認してから本番システムで実行します。

ダウングレードはそれほど一般的ではありません。通常、最初にテストシステムでアップグレードを確認したときには発見されなかった互換性またはパフォーマンスの問題が本番システムで発生したために、アップグレードを元に戻します。アップグレードの手順と同様に、ダウングレードの手順は本番システム上で使用する前に、まずテストシステム上で実行して確認します。

2.11.1 MySQL のアップグレード

原則として、あるリリースシリーズから別のものにアップグレードするには、シリーズをスキップせずに次のシリーズに移動します。MySQL 5.5 より前のリリースシリーズからアップグレードするには、MySQL 5.5 に到達するまで順に次のリリースシリーズにアップグレードしてから、MySQL 5.6 へのアップグレードに進みます。たとえば、現在 MySQL 5.1 を稼働していて、新しいシリーズにアップグレードする場合は、MySQL 5.6 にアップグレードする前にまず 5.5 にアップグレードする、などです。MySQL 5.5 へのアップグレードの詳細は、『MySQL 5.5 リファレンスマニュアル』を参照してください。

MySQL 5.6 にアップグレードするには、次のチェックリストの項目を参考にしてください。

- アップグレードの前に、付与テーブルを含む mysql データベースなどのデータベースのバックアップを作成します。セクション7.2「データベースバックアップ方法」を参照してください。
- セクション2.11.1.3「MySQL 5.5 から 5.6 へのアップグレード」のすべての注釈をお読みください。これらの注釈によって、現在の MySQL インストールに適用されるアップグレードの問題を識別できます。そのセクションで説明されている一部の互換性の欠如は、アップグレードの前に注意する必要があります。アップグレードのあとにアクションが必要なものもあります。
- リリースノートもお読みください。ここには、MySQL 5.6 の新機能や MySQL の以前のリリースとは異なるものに関する情報があります。
- MySQL の新しいバージョンにアップグレードしたら、mysql_upgrade を実行します (セクション 4.4.7「mysql_upgrade — MySQL テーブルのチェックとアップグレード」を参照してください)。このプログラムはテーブルを確認し、必要に応じて修復を試みます。また、付与テーブルを更新して最新の構造を持つようにし、新しい機能を活用できるようにします。(MySQL のリリースの中には、新たに権限や機能を加えるために付与テーブルの構造に変更を加えているものもあります。)

  mysql_upgrade では、ヘルプテーブルの内容はアップグレードされません。アップグレードの手順については、セクション5.1.10「サーバー側のヘルプ」を参照してください。

  mysql_upgrade は、サーバーが --gtid-mode=ON で稼働している場合には使用しないでください。使用すると、mysql データベース内の非トランザクションシステムテーブルが変更される場合があります。これらの多くは MyISAM で、異なるストレージエンジンを使用するように変更することはできません。GTID モードと mysql_upgradeを参照してください。
- MySQL Server を Windows 上で使用する場合は、セクション2.3.7「Windows 上の MySQL をアップグレードする」を参照してください。
- レプリケーションを使用する場合は、レプリケーションのセットアップのアップグレードについてセクション 17.4.3「レプリケーションセットアップをアップグレードする」を参照してください。
- InnoDB を使用する場合は、サーバーをシャットダウンしてアップグレードする前に、innodb_fast_shutdown を 0 に設定することを考慮します。innodb_fast_shutdown を 0 に設定すると、InnoDB はシャットダウンの前に低速シャットダウン、完全なパージ、および挿入バッファーのマージを実行します。これにより、アップグレードプロセスでファイル形式が変更される場合に備えて、すべてのデータファイルが完全に準備されます。
- もともと複数の RPM パッケージをインストールして生成したインストールをアップグレードする場合は、一部だけでなくすべてのパッケージをアップグレードするのが最善です。たとえば、以前にサーバーおよびクライアントの RPM をインストールした場合は、サーバーの RPM だけをアップグレードすることはしないでください。
- ユーザー定義関数 (UDF) を所定の名前で作成したあとで、MySQL を同じ名前の組み込み関数を実装した新しいバージョンにアップグレードした場合、その UDF はアクセスできなくなります。これを修正するには、DROP FUNCTION を使用して UDF を削除してから、CREATE FUNCTION を使用して競合しない別の名前で UDF を再作成します。新しいバージョンの MySQL が、既存のストアドファンクションと同名の組み込み関数を実装している場合にも、同じとこが言えます。各種の関数への参照をサーバーが解釈する方法を記述したルールについては、セクション9.2.4「関数名の構文解析と解決」を参照してください。

EL5、EL6、または EL7 ベースの Linux プラットフォームおよび Fedora 20 または 21 では、MySQL およびそのコンポーネントを、MySQL Yum リポジトリでインプレースアップグレードできます。セクション 2.11.1.1「MySQL Yum リポジトリを使用する MySQL のアップグレード」を参照してください。

Debian 7、Ubuntu 12、および Ubuntu 14 では、MySQL およびそのコンポーネントを、MySQL APT リポジトリでインプレースアップグレードできます。セクション2.11.1.2「MySQL APT リポジトリを使用する MySQL のアップグレード」を参照してください。

MySQL リリースシリーズの一般提供ステータスに到達したバージョン間のアップグレードでは、MySQL 形式のファイルおよびデータファイルを、同じアーキテクチャーを持つシステム上の異なるバージョン間で移動できます。MySQL リリースシリーズの開発ステータスにあるバージョンへのアップグレードでは、必ずしもそうではありません。開発リリースは自己責任でご使用ください。

新しいバージョンの使用に慎重な場合には、新しいバージョンをインストールする前に、常に古い mysqld の名前を変更できます。たとえば、MySQL 5.5 のバージョンを使用していて 5.6 にアップグレードする場合は、現在のサーバーを mysqld から mysqld-5.5 に名前変更します。新しい mysqld に予想外の問題が発生した場合は、単にそれをシャットダウンして古い mysqld を起動するだけで済みます。

新しい mysqld サーバーが起動しない、あるいはパスワードなしで接続できないなどの問題が発生した場合には、前のインストールの古い my.cnf ファイルが残っていないか確認します。これは --print-defaults オプション (たとえば、mysqld --print-defaults) で確認できます。このコマンドがプログラム名以外の何かを表示した場合、サーバーあるいはクライアントオペレーションに影響を及ぼすアクティブな my.cnf ファイルがあることになります。

アップグレード後に、コンパイル済みのクライアントプログラムに Commands out of sync または予測外のコアダンプなどの問題が発生した場合は、プログラムのコンパイル時に、古いヘッダーファイルまたはライブラリファイルを使用した可能性があります。この場合、mysql.h ファイルおよび libmysqlclient.a ライブラリの日付をチェックして、それらが新しい MySQL 配布のものであることを確認します。そうでない場合には、プログラムを新しいヘッダーおよびライブラリで再度コンパイルします。(libmysqlclient.so.15 から libmysqlclient.so.16 など) ライブラリのメジャーバージョン番号が変更された場合は、共有クライアントライブラリを使用してコンパイルされたプログラムも再コンパイルが必要になることがあります。

MySQL インストールに、インプレースアップグレード後の変換に長い時間がかかる可能性のある大量のデータが含まれている場合は、必要な変換とその実行に関係する作業を評価するための「仮の」データベースインスタンスを作成すると役立つことがあります。mysql データベースの完全なコピーと、データを含まないその他のすべてのデータベースを含む MySQL インスタンスのコピーを作成します。このダミーのインスタンスに対してアップグレードの手順を実行し、必要になるアクションを確認して、元のデータベースインスタンスで実際のデータ変換を実行するときに関係する作業をよりよく評価できるようにします。

新しいリリースの MySQL をインストールした際は必ず Perl DBD::mysql モジュールを再構築して再インストールするのがいいでしょう。同じことが、PHP mysql 拡張や Python MySQLdb モジュールなどの、ほかの MySQL インタフェースにも言えます。

2.11.1.1 MySQL Yum リポジトリを使用する MySQL のアップグレード

サポートされる Yum ベースのプラットフォーム (リストについてはセクション2.5.1「MySQL Yum リポジトリを使用して MySQL を Linux にインストールする」を参照してください) では、MySQL Yum リポジトリを使用して、MySQL のインプレースアップグレード (つまり、古いバージョンを置換して、古いデータファイルに対して新しいバージョンを実行) を実行できます。

> **メモ**
>
> - MySQL に対して更新を実行する前に、セクション2.11.1「MySQL のアップグレード」の手順に慎重に従ってください。そこで議論されている手順のうち、更新の前にデータベースのバックアップを取ることが特に重要です。
> - 次の手順では、MySQL Yum リポジトリを使用して MySQL をインストールしたことを前提としています。そうでない場合は、セクション2.5.2「サードパーティーの MySQL 配布を MySQL Yum リポジトリを使用して置換する」の手順に従ってください。

ターゲット シリーズの選択

デフォルトでは、MySQL Yum リポジトリは、インストール中に選択したリリースシリーズ内の最新バージョンに MySQL を更新します (詳細はSelecting a Release Seriesを参照してください)。つまり、たとえば 5.6.x インストールは 5.7.x リリースに自動的には更新されません。別のリリースシリーズに更新するには、(デフォルトで、または自分で) 選択したシリーズのサブリポジトリを無効にし、ターゲットシリーズのサブリポジトリを有効にする必要があります。それをするには、 Selecting a Release Seriesで説明されている、/etc/

yum.repos.d/mysql-community.repo ファイル内のサブリポジトリのエントリを編集するための手順に従います。

原則として、あるリリースシリーズから別のものにアップグレードするには、シリーズをスキップせずに次のシリーズに移動します。たとえば、現在 MySQL 5.5 を稼働していて、5.7 にアップグレードする場合は、MySQL 5.7 にアップグレードする前にまず 5.6 にアップグレードします。

> **重要**
>
> MySQL 5.6 から 5.7 へのアップグレードに関する重要な情報については、「Upgrading from MySQL 5.6 to 5.7」を参照してください。

MySQL のアップグレード

次のコマンドで MySQL およびそのコンポーネントをアップグレードします。

```
shell> sudo yum update mysql-server
```

または、Yum に対してシステム上のすべてを更新するように指示することで、MySQL を更新することもできます (これにはかなり時間がかかる場合があります)。

```
shell> sudo yum update
```

MySQL の再起動

MySQL Server は、Yum による更新後必ず再起動します。サーバーの再起動後、mysql_upgrade を実行して、古いデータとアップグレードされたソフトウェアとの間の非互換性をチェックし、あれば解決します。mysql_upgrade はその他の機能も実行します。詳細は、セクション4.4.7「mysql_upgrade — MySQL テーブルのチェックとアップグレード」を参照してください。

特定のコンポーネントのみを更新することもできます。次のコマンドを使用して、MySQL コンポーネントのインストール済みのすべてのパッケージをリストできます。

```
shell> sudo yum list installed | grep "^mysql"
```

選択したコンポーネントのパッケージ名を特定したら、次のコマンドで package-name をそのパッケージ名に置換して、パッケージを更新します。

```
shell> sudo yum update package-name
```

共有クライアントライブラリのアップグレード

Yum リポジトリを使用して MySQL を更新したあとも、古いバージョンの共有クライアントライブラリを使用してコンパイルされたアプリケーションは機能するはずです。

アプリケーションを再コンパイルして、更新されたライブラリに動的にリンクする場合: 一般的に、新しいバージョンの共有ライブラリでは新旧ライブラリ間でシンボルバージョン管理に違いや追加があるため (たとえば、新しい標準 5.6 共有クライアントライブラリと、Linux 配布のソフトウェアリポジトリによってネイティブで出荷された、またはその他のソースからの一部の古い (以前のまたはバリアントの) バージョンの共有ライブラリとの間で)、更新された新しい共有ライブラリを使用してコンパイルされたアプリケーションは、アプリケーションが配備されるシステム上で更新済みのライブラリを必要とします。また、これらのライブラリがない場合は、共有ライブラリを必要とするアプリケーションは予想どおり失敗します。したがって、MySQL からの共有ライブラリのパッケージを、必ずこれらのシステムに配備してください。これは、MySQL Yum リポジトリをシステムに追加し (Adding the MySQL Yum Repositoryを参照してください)、Installing Additional MySQL Products and Components with Yumに記載の手順を使用して最新の共有ライブラリをインストールすることによって実行できます。

2.11.1.2 MySQL APT リポジトリを使用する MySQL のアップグレード

Debian 7、Ubuntu 12、および Ubuntu 14 では、MySQL およびそのコンポーネントを、MySQL APT リポジトリでインプレースアップグレードできます。「A Quick Guide to Using the MySQL APT Repository」の、「Upgrading MySQL with the MySQL APT Repository」を参照してください。

2.11.1.3 MySQL 5.5 から 5.6 へのアップグレード

> **注記**
>
> MySQL 5.6.6 以降では、MySQL Server のいくつかのパラメータのデフォルト値が、前のリリースと異なっています。このセクションで後述するこれらの変更に関する注記、

特に後方互換性の維持が懸念事項である場合は、それらのオーバーライドに関する注記を参照してください。

注記

新しいバージョンのソフトウェアをインストールする前にデータのバックアップを必ず励行してください。MySQL は、高水準の品質を保証するため多大な努力を行なっていますが、バックアップを取ってデータを保護してください。

以前のバージョンから 5.6 にアップグレードするには、アップグレードの前に mysqldump でテーブルをダンプし、アップグレード後にダンプファイルをリロードすることを、MySQL は推奨します。すべてのデータベースをダンプに含めるには、--all-databases オプションを使用します。データベースにストアドプログラムが含まれる場合は、--routines オプションおよび --events オプションも使用します。

一般的に、MySQL 5.5 から 5.6 へのアップグレードを行う場合は次を実行します。

- 次のセクションにあるすべての項目を読み、いずれかの項目がアプリケーションに影響を及ぼすかどうか確認します。
  - セクション2.11.1「MySQL のアップグレード」に、更新に関する一般的な情報があります。
  - このセクションで後述する変更リスト内の項目によって、現在の MySQL インストールに該当するアップグレードの問題を識別できます。そこで説明されている互換性の欠如の一部は、アップグレードの前に注意する必要があります。ほかの点はアップグレードのあとで扱うようにしてください。
  - MySQL 5.6 リリースノートでは、5.6 で使用できる主な新機能や、MySQL の以前のリリースとは異なるものについて説明しています。これらの変更の一部には互換性がない場合があります。

  既知の問題または非互換の変更というマークが付いている変更に特に注意してください。以前のバージョンの MySQL との互換性がないこれらの変更では、アップグレードする前に注意を払う必要があることがあります。弊社の目的はそれらの変更を避けることですが、各リリースの間の非適合性よりもさらに深刻な問題を修正するために必要である場合もあります。使用しているインストールに当てはまるアップグレードの問題に、特別な処置を必要とする互換性の欠如が関係している場合は、互換性の欠如の説明にある手順に従ってください。これには、テーブルのダンプとリロード、あるいは CHECK TABLE や REPAIR TABLE などのステートメントの使用が含まれる場合があります。

  ダンプとリロードの手順については、セクション2.11.4「テーブルまたはインデックスの再作成または修復」を参照してください。USE_FRM オプションを使う REPAIR TABLE に関係する手順は、アップグレードの前に実行する必要があります。テーブルの作成に使用したものとは異なるバージョンの MySQL でこのステートメントを使用すると (つまり、アップグレード後にこのステートメントを使用すると)、テーブルが破損することがあります。セクション13.7.2.5「REPAIR TABLE 構文」を参照してください。

- 新しいバージョンの MySQL にアップグレードする前に、現在使用している MySQL のバージョンとアップグレードするバージョンとの間でテーブルの形式または文字セットまたは照合順序に変更があったかどうかを、セクション2.11.3「テーブルまたはインデックスの再構築が必要かどうかのチェック」で確認してください。その場合、それらの変更により MySQL バージョン間に互換性の欠如が生じている場合は、セクション2.11.4「テーブルまたはインデックスの再作成または修復」の手順を使用して影響されるテーブルをアップグレードすることが必要になります。

- MySQL の新しいバージョンにアップグレードしたら、mysql_upgrade を実行します (セクション4.4.7「mysql_upgrade — MySQL テーブルのチェックとアップグレード」を参照してください)。このプログラムはテーブルを確認し、必要に応じて修復を試みます。また、付与テーブルを更新して最新の構造を持つようにし、新しい機能を活用できるようにします。(MySQL のリリースの中には、新たに権限や機能を加えるために付与テーブルの構造に変更を加えているものもあります。)

  mysql_upgrade では、ヘルプテーブルの内容はアップグレードされません。アップグレードの手順については、セクション5.1.10「サーバー側のヘルプ」を参照してください。

- MySQL Server を Windows 上で使用する場合は、セクション2.3.7「Windows 上の MySQL をアップグレードする」を参照してください。

- レプリケーションを使用する場合は、レプリケーションのセットアップのアップグレードについてセクション17.4.3「レプリケーションセットアップをアップグレードする」を参照してください。

MySQL インストールに、インプレースアップグレード後の変換に長い時間がかかる可能性のある大量のデータが含まれている場合は、必要な変換とその実行に関係する作業を評価するための「仮の」データベースインスタンスを作成すると役立つことがあります。mysql データベースの完全なコピーと、データを含まないその他のすべ

てのデータベースを含む MySQL インスタンスのコピーを作成します。このダミーのインスタンスに対してアップグレードの手順を実行し、必要になるアクションを確認して、元のデータベースインスタンスで実際のデータ変換を実行するときに関係する作業をよりよく評価できるようにします。

次からのセクションにあるすべての項目を読み、いずれかの項目がアプリケーションに影響を及ぼすかどうか確認します。

構成の変更

- MySQL 5.6.6 以降では、MySQL Server のいくつかのパラメータのデフォルト値が、前のリリースと異なっています。これらの変更の目的は、初期設定のままで優れたパフォーマンスを提供し、データベース管理者が設定を手動で変更する必要性を低下させることです。これらの変更は、フィードバックの取得に伴い将来のリリースでのリビジョンにつながる場合があります。

  パラメータが異なる静的なデフォルト値を持つ場合もあります。あるいは、静的な値を使用するのではなく、ほかの関連するパラメータやサーバーホスト構成に基づく公式を使用して、サーバーが起動時にパラメータを自動サイズ設定する場合もあります。たとえば、back_log の現在の設定は、以前のデフォルトの 50 で、max_connections の値に比例する量に応じて上方に調整されます。自動サイズ設定の背後には、固定値よりも適切だと思われるパラメータ設定について、決定を下すために利用できる情報をサーバーが持つ場合、その決定を下すという考え方があります。

  次の表は、デフォルトへの変更を要約したものです。これらはすべて、サーバー起動時に明示的な値を指定することによってオーバーライドできます。

パラメータ	古いデフォルト	新しいデフォルト
back_log	50	max_connections を使用した自動サイズ設定
binlog_checksum	NONE	CRC32
--binlog-row-event-max-size	1024	8192
flush_time	1800 (Windows の場合)	0
innodb_autoextend_increment	8	64
innodb_buffer_pool_instances	1	8 (プラットフォームに依存)
innodb_checksum_algorithm	INNODB	CRC32
innodb_concurrency_tickets	500	5000
innodb_file_per_table	0	1
innodb_old_blocks_time	0	1000
innodb_open_files	300	innodb_file_per_table、table_open_cache を使用した自動サイズ設定
innodb_stats_on_metadata	ON	OFF
join_buffer_size	128KB	256KB
max_allowed_packet	1MB	4MB
max_connect_errors	10	100
sync_master_info	0	10000
sync_relay_log	0	10000
sync_relay_log_info	0	10000

  以前のリリースとの互換性に関して、もっとも重要な変更は:

  - innodb_file_per_table は有効です (以前は無効)。
  - innodb_checksum_algorithm は CRC32 です (以前は INNODB)。
  - binlog_checksum は CRC32 です (以前は NONE)。

  したがって、既存の MySQL インストールからアップグレードする場合で、これらのパラメータの値を以前のデフォルトからまだ変更しておらず、後方互換性が懸念事項である場合は、これらを以前のデフォルト値に明示的にセットするとよいでしょう。たとえば、サーバーのオプションファイルに次の行を挿入します。

```
[mysqld]
innodb_file_per_table=0
innodb_checksum_algorithm=INNODB
binlog_checksum=NONE
```

これらの設定により、互換性が次のように維持されます。

- innodb_file_per_table の新しいデフォルトは有効で、アップグレード後に ALTER TABLE 操作を行うとシステムのテーブルスペース内の InnoDB テーブルが個々の .ibd ファイルに移動されます。innodb_file_per_table=0 を使用するとこれを防ぐことができます。
- innodb_checksum_algorithm=INNODB を設定することにより、このリリースにアップグレードしたあとでバイナリのダウングレードが可能になります。CRC32 を設定することにより、InnoDB は MySQL の古いバージョンが使用できないチェックサムを使用します。
- binlog_checksum=NONE により、バイナリログチェックサムを理解しない古いスレーブが失敗することなく、サーバーをレプリケーションマスターとして使用できます。

- MySQL 5.6.5 では、4.1 より前のパスワードと mysql_old_password 認証プラグインは非推奨です。MySQL 4.1 より前で使用される古いハッシュ形式で格納されているパスワードは、ネイティブのパスワードハッシュ方式を使用するパスワードよりもセキュアでないため、使用しないようにしてください。4.1 より前のパスワードハッシュを持つアカウントを使用して接続することを防ぐために、secure_auth システム変数はデフォルトで有効になりました。(このようなパスワードハッシュを持つアカウントに対して接続を許可するには、--secure_auth=0 を使用してサーバーを起動してください。)

  データベース管理者は、mysql_old_password 認証プラグインを使用するアカウントを、代わりに mysql_native_password を使用するように変換することが推奨されます。アカウントのアップグレード手順については、セクション6.3.8.3「4.1 よりも前のパスワードハッシュ方式と mysql_old_password プラグインからの移行」を参照してください。

  既知の問題: MySQL 5.6 の以前の開発バージョンの一部 (5.6.6 から 5.6.10) では、サーバーが、パスワードハッシュと認証プラグインが一致しないアカウントを作成することがあります。たとえば、デフォルトの認証プラグインが mysql_native_password の場合、次のステートメントのシーケンスにより、プラグインが mysql_native_password だが、4.1 より前のパスワードハッシュ (mysql_old_password で使用される形式) であるアカウントになります。

  ```
  SET old_passwords = 1;
  CREATE USER 'jeffrey'@'localhost' IDENTIFIED BY 'mypass';
  ```

  この不一致により、MySQL Server に接続できない、SET PASSWORD を OLD_PASSWORD() または old_passwords=1 で使用できない、などの現象が発生します。

  MySQL 5.6.11 では、この不一致は発生しなくなりました。代わりに、サーバーがエラーを生成します。

  ```
  mysql> SET old_passwords = 1;
  mysql> CREATE USER 'jeffrey'@'localhost' IDENTIFIED BY 'mypass';
  ERROR 1827 (HY000): The password hash doesn't have the expected
  format. Check if the correct password algorithm is being used with
  the PASSWORD() function.
  ```

  不一致の影響を受けているアカウントを処理するには、データベース管理者はそのアカウントの mysql.user テーブルの行の plugin カラムまたは Password カラムを、ほかのカラムと一致するように修正できます。

  - old_passwords を 0 に設定してから、SET PASSWORD および PASSWORD() を使用してアカウントに新しいパスワードを割り当てます。これにより、Password カラムが 4.1 パスワードハッシュを持つようになり、mysql_native_password プラグインと一致します。これが、推奨されるアカウントの修正方法です。
  - あるいは、データベース管理者はプラグインがパスワードハッシュ形式に一致するように mysql_old_password を変更してから、権限をフラッシュできます。mysql_old_password プラグインおよび 4.1 より前のパスワードハッシュは非推奨であり、MySQL の将来のバージョンではサポートされなくなるため、この方法は推奨されません。

サーバーの変更

- 互換性のない変更: あるカラムの DEFAULT 値が、テーブル作成時の sql_mode 値に対しては有効だが、行の挿入または更新を行うときの sql_mode 値に対しては無効であるということが起こり得ます。例:

  ```
  SET sql_mode = '';
  CREATE TABLE t (d DATE DEFAULT 0);
  ```

```
SET sql_mode = 'NO_ZERO_DATE,STRICT_ALL_TABLES';
INSERT INTO t (d) VALUES(DEFAULT);
```

この場合、0 は CREATE TABLE に対しては受け入れられるべきですが、INSERT に対しては受け入れられるべきではありません。しかし、サーバーは挿入または更新時の DEFAULT 値を現在の sql_mode に対して評価しませんでした。この例では、INSERT は成功し、'0000-00-00' を DATE カラムに挿入します。

MySQL 5.6.13 では、サーバーは適切な sql_mode チェックを適用して、挿入時または更新時に警告またはエラーを生成します。

その結果、ステートメントベースのロギング (binlog_format=STATEMENT) を使用している場合に、レプリケーションに関する非互換性が生じます。スレーブがアップグレードされた場合、アップグレードされていないマスターが次の例をエラーなしで実行しますが、INSERT はスレーブで失敗し、レプリケーションは停止します。

これに対処するには、マスター上のすべての新しいステートメントを停止し、スレーブが追い付くのを待ちます。そのあと、スレーブに続いてマスターをアップグレードします。あるいは、新しいステートメントを停止できない場合は、マスターで一時的に行ベースのロギングに変更し (binlog_format=ROW)、すべてのスレーブが、この変更の時点までに生成されたすべてのバイナリログを処理するまで待ちます。そのあと、スレーブに続いてマスターをアップグレードし、マスターをステートメントベースのロギングに戻します。

- 互換性のない変更: MySQL 5.6.11 以降では、CREATE TABLE ... [SUB]PARTITION BY ALGORITHM=n [LINEAR] KEY (...) をサポートします。これは、KEY のパーティション化が MySQL 5.1 Server と互換性を持つテーブルを作成するのに使用できます (n=1)。(Bug #14521864、Bug #66462) この構文は MySQL 5.5.31 以降の MySQL 5.5 ではサポートされていますが、MySQL 5.6.10 以前では受け入れられません。MySQL 5.5.31 以降の MySQL 5.5 リリースの mysqldump には、このオプションを使用してテーブルをダンプするときに ALGORITHM オプションが含まれますが、次のように条件コメントで囲みます。

```
CREATE TABLE t1 (a INT)
/*!50100 PARTITION BY KEY */ /*!50531 ALGORITHM = 1 */ /*!50100 ()
    PARTITIONS 3 */
```

このような CREATE TABLE ステートメントを含むダンプを MySQL 5.6.10 以前の MySQL 5.6 Server にインポートする場合、バージョン付きコメントは無視されず構文エラーが発生します。したがって、このようなダンプファイルをインポートする前に、MySQL 5.6 Server がコメントを無視するように変更するか (文字列 !50531 が出現するたびにそれを削除するか、!50611 に置換することによって)、あるいは削除します。

これは、MySQL 5.6.11 以降を使用して作成されたダンプファイルでは問題になりません。ここでは ALGORITHM オプションは /*!50611 ... */ を使用して書き込まれています。

- 互換性のない変更: TIME、DATETIME、および TIMESTAMP カラムについては、MySQL 5.6.4 より前に作成されたテーブルに必要なストレージは、5.6.4 以降で作成されたテーブルに必要なストレージとは異なります。これは、5.6.4 で、これらの時間型が少数部を持つことを許可するように変更されたためです。MySQL 5.5 から MySQL 5.6.4 以降にアップグレードしたあと、MySQL 5.5 から MySQL 5.6 の TIME、DATETIME、および TIMESTAMP 型にもアップグレードすることを推奨します。ALTER TABLE は現在、MySQL 5.5 および MySQL 5.6.4 (以降) のバイナリ形式の時間カラムを含むテーブルの作成を許可しますが、このため、.frm ファイルが利用できない場合にテーブルを再作成するのがより困難になります。さらに、MySQL 5.6.4 では、前述の時間型はスペース効率が改善されています。MySQL 5.6.4 での時間型の変更の詳細は、Storage Requirements for Date and Time Typesを参照してください。

  MySQL 5.6.16 では、ALTER TABLE は、ADD COLUMN、CHANGE COLUMN、MODIFY COLUMN、ADD INDEX、および FORCE 操作に関して、古い時間カラムを 5.6 形式にアップグレードします。したがって、次のステートメントは、古い形式のカラムを含むテーブルをアップグレードします。

  ```
  ALTER TABLE tbl_name FORCE;
  ```

  テーブルを再構築しなければならないため、この変換は INPLACE アルゴリズムを使用して実行することはできません。そのため、これらの場合に ALGORITHM=INPLACE を指定すると エラーになります。必要であれば、ALGORITHM=COPY を指定します。

  ALTER TABLE は、時間形式の変換を実行しない場合には、SHOW WARNINGS: TIME/TIMESTAMP/DATETIME columns of old format have been upgraded to the new format で表示できるメッセージを生成します。

- 前記の互換性のない変更で説明した時間型の変更のため、DATETIME 型および TIMESTAMP 型を含む、MySQL 5.6.4 より前のテーブルを MySQL 5.6.4 (以降) にインポートすると失敗します。これらの時間型を

持つ MySQL 5.5 テーブルを MySQL 5.6.4 (以降) にインポートする場合が、この問題が発生する可能性がもっとも高いシナリオです。

次の手順では、元の MySQL 5.6.4 より前の .frm ファイルを使用して、5.6.4 (以降) と互換性のある行構造を持つテーブルを再作成する回避策について説明します。この手順には、.frm ファイルを目的のインスタンスのデータディレクトリにコピーし、ALTER TABLE を使用してテーブルのストレージエンジンタイプを InnoDB に戻すことで、MySQL 5.6.4 より前の元の .frm ファイルを、InnoDB の代わりに Memory ストレージエンジンを使用するように変更することが含まれます。テーブルに外部キーがない場合は最初の手順を使用します。テーブルに外部キーが含まれる場合は、追加の手順を含む 2 番目の手順を使用します。

テーブルに外部キーがない場合:

1. テーブルの元の .frm ファイルを、テーブルスペースをインポートするサーバーのデータディレクトリにコピーします。
2. テーブルの .frm ファイルを、InnoDB ストレージエンジンの代わりに Memory ストレージエンジンを使用するように変更します。この変更には、.frm ファイル内の、テーブルのストレージエンジンタイプを定義する 7 バイトを変更する必要があります。16 進の編集ツールを使用する場合:
   - 次に示すように、オフセット位置 0003 のバイト legacy_db_type を、「0c」 (InnoDB) から 「06」 (Memory) に変更します。

     ```
     00000000  fe 01 09 06 03 00 00 10  01 00 00 30 00 00 10 00
     ```

   - 残りの 6 バイトには固定のオフセットはありません。.frm ファイルで 「InnoDB」 を検索し、その他の 6 バイトを含む行を探します。行は次に示すようになります。

     ```
     00001010  ff 00 00 00 00 00 00 06  00 49 6e 6e 6f 44 42 00  |.........InnoDB.|
     ```

   - 行が次のようになるようにバイトを変更します。

     ```
     00001010  ff 00 00 00 00 00 00 06 00 4d 45 4d 4f 52 59 00
     ```

3. ALTER TABLE ... ENGINE=INNODB を実行して、テーブル定義を InnoDB データディクショナリに追加します。これにより、新しい形式の時間データ型を持つ InnoDB テーブルが作成されます。ALTER TABLE 操作が正常に完了するためには、.frm ファイルがテーブルスペースに対応していなければなりません。
4. ALTER TABLE ... IMPORT TABLESPACE を使用してテーブルをインポートします。

テーブルに外部キーがある場合:

1. SHOW CREATE TABLE の出力からテーブルの定義を使用して、外部キーのあるテーブルを再作成します。この時点では、正しくない時間カラムフォーマットは問題ではありません。
2. INFORMATION_SCHEMA.TABLE_CONSTRAINTS および INFORMATION_SCHEMA.KEY_COLUMN_USAGE から外部キー情報を選択して、すべての外部キー定義をテキストファイルにダンプします。
3. すべてのテーブルを削除し、外部キーのないテーブルに関する前述の手順のステップ 1 から 4 で説明されているテーブルのインポートプロセスを完了します。
4. インポート操作が完了したら、テキストファイルに保存した外部キー定義から外部キーを追加します。

- 互換性のない変更: MySQL 5.6 では、character_set_server が ucs2、utf16、utf16le、または utf32 の場合、全文ストップワードファイルがロードされ、latin1 を使用して検索されます。サーバーの文字セットが ucs2、utf16、utf16le、または utf32 である間に、FULLTEXT インデックスを持つテーブルが作成された場合は、次のステートメントを使用して修復してください。

  ```
  REPAIR TABLE tbl_name QUICK;
  ```

- 互換性のない変更: MySQL 5.6.20 では、Bug #69477 に対するパッチにより、BLOB で書き込まれる Redo ログのサイズが Redo ログファイルサイズの 10% に制限されます。この新しい制限の結果として、innodb_log_file_size は、テーブルの行で見つかった最大の BLOB データサイズの 10 倍よりも大きい値に設定するべきです。innodb_log_file_size 設定がすでに、最大の BLOB データサイズの 10 倍になっているか、テーブルに BLOB データが含まれない場合は、アクションは不要です。

  MySQL 5.6.22 では、Redo ログ BLOB の書き込み制限は 合計 Redo ログサイズ (innodb_log_file_size * innodb_log_files_in_group) の 10% に緩められました。(Bug #19498877)

SQL の変更

- MySQL 5.5 では予約されていなかった一部のキーワードが、MySQL 5.6 では予約されている場合があります。セクション9.3「予約語」を参照してください。
- YEAR(2) データ型には、使用する前に考慮する必要のある特定の問題があります。MySQL 5.6.6 以降、YEAR(2) は非推奨です。既存のテーブル内の YEAR(2) カラムは以前のとおりに扱われますが、新規または変更したテーブルでは YEAR(2) は YEAR(4) に変換されます。詳細は、セクション11.3.4「YEAR(2) の制限と YEAR(4) への移行」を参照してください。
- MySQL 5.6.6 では、値 DEFAULT をストアドプロシージャーまたはストアドファンクションのパラメータや、ストアドプログラムのローカル変数に割り当てることは (SET var_name = DEFAULT ステートメント)、明示的に不許可です。これは、以前にはサポートされておらず、許可されるという説明もありませんでしたが、何らかの都合で既存のコードでこの構造が使用されていた場合のために、互換性のない変更としてフラグが付けられました。システム変数に DEFAULT を割り当てることは以前と同様に許可されますが、DEFAULT をパラメータやローカル変数に割り当てると、構文エラーになります。

  MySQL 5.6.6 以降へのアップグレード後、この構造を使用する既存のストアドプログラムが起動されると、構文エラーになります。5.6.5 以前の mysqldump ファイルが 5.6.6 以降にロードされると、ロード操作は失敗し、影響を受けるストアドプログラム定義を変更する必要があります。
- MySQL では、TIMESTAMP データ型は非標準的な方式であるという点でほかのデータ型と異なります。
  - NULL 属性で明示的に宣言されない TIMESTAMP カラムには、NOT NULL 属性が割り当てられます。(ほかのデータ型のカラムは、NOT NULL として明示的に宣言されない場合、NULL 値が許可されます。)そのようなカラムを NULL に設定すると、カラムは現在のタイムスタンプに設定されます。
  - テーブル内の最初の TIMESTAMP カラムは、NULL 属性や明示的な DEFAULT または ON UPDATE 句で宣言されない場合、DEFAULT CURRENT_TIMESTAMP および ON UPDATE CURRENT_TIMESTAMP 属性が自動的に割り当てられます。
  - 最初のカラムに続く TIMESTAMP カラムは、NULL 属性または明示的な DEFAULT 句で宣言されない場合、DEFAULT '0000-00-00 00:00:00' (「ゼロ」タイムスタンプ) が自動的に割り当てられます。そのような カラムに対して明示的な値を指定しない挿入された行については、カラムに '0000-00-00 00:00:00' が自動的に割り当てられて、警告は発生しません。

  これらの非標準の動作は TIMESTAMP についてはデフォルトのままですが、MySQL 5.6.6 以降では非推奨となり、起動時に次の警告が表示されます。

```
[Warning] TIMESTAMP with implicit DEFAULT value is deprecated.
Please use --explicit_defaults_for_timestamp server option (see
documentation for more details).
```

  警告が示すように、非標準の動作をオフにするには、新しい explicit_defaults_for_timestamp システム変数を起動時に有効にします。この変数を有効にすると、サーバーは TIMESTAMP を、代わりに次のように処理します。

  - 明示的に NOT NULL として宣言されない TIMESTAMP カラムでは、NULL 値が許可されます。そのような カラムを NULL に設定することで、カラムは現在のタイムスタンプではなく NULL に設定されます。
  - TIMESTAMP カラムに DEFAULT CURRENT_TIMESTAMP または ON UPDATE CURRENT_TIMESTAMP 属性が自動的に割り当てられません。これらの属性は、明示的に指定する必要があります。
  - NOT NULL として宣言され、明示的な DEFAULT 句を持たない TIMESTAMP カラムは、デフォルト値を持たないものとして処理されます。そのようなカラムについて明示的な値を指定しない挿入された行の場合、結果は SQL モードによって異なります。厳密 SQL モードが有効である場合、エラーが発生します。厳密 SQL

モードが有効でない場合、カラムには暗黙的なデフォルトの '0000-00-00 00:00:00' が割り当てられ、警告が発生します。これは、MySQL が DATETIME などのほかの時間型を処理する方法に類似しています。

レプリケーションに使用されるサーバーをアップグレードする場合は、まずスレーブをアップグレードしてからマスターをアップグレードします。マスターとそのスレーブとの間のレプリケーションは、すべてが explicit_defaults_for_timestamp の同じ値を使用しているという条件で、機能するはずです。

1. スレーブを停止してアップグレードし、explicit_defaults_for_timestamp の任意の値で構成してから起動します。

   スレーブは、マスターから受信したバイナリログの形式から、マスターが古い (explicit_defaults_for_timestamp の導入に先行する) こと、およびマスターからの TIMESTAMP カラムが古い TIMESTAMP 動作を使用していることを認識します。

2. マスターを停止してアップグレードし、スレーブに使用されるのと同じ explicit_defaults_for_timestamp の値で構成してから起動します。

2.11.2 MySQL のダウングレード

このセクションでは、新しいバージョンよりも以前のバージョンの方がよく動作するというまれな場合のために、MySQL の古いバージョンにダウングレードする方法について説明します。

ダウングレードが失敗してインスタンスが不安定な状態になった場合に備えて、前もってバックアップを取るとよいでしょう。

同じリリースシリーズ内の一般提供 (GA) ステータスのバージョン間でダウングレードする場合は、通常は新しいバイナリを古いものの上にインストールするだけで、データベースには何も変更を加えません。

同じリリースシリーズ内のマイルストーンリリース間 (または GA リリースからマイルストーンリリースへ) のダウングレードはサポートされておらず、問題が発生する場合があります。

次の項目は、ダウングレードを実行する際に必ず行うことのチェックリストです。

- ダウングレード元のリリースシリーズのアップグレードのセクションを読み、実際に必要な機能がないことを確認します。セクション2.11.1「MySQL のアップグレード」を参照してください。
- そのバージョンのダウングレードのセクションがある場合は、それもお読みください。
- ダウングレード先のバージョンと現在使用しているバージョンとの間で、どの新機能が追加されたかを確認するには、リリースノートを参照してください。
- 現在使用している MySQL のバージョンとダウングレード先のバージョンとの間でテーブルの形式または文字セットまたは照合順序に変更があったかどうかを、セクション2.11.3「テーブルまたはインデックスの再構築が必要かどうかのチェック」で確認してください。その場合、それらの変更により MySQL バージョン間に互換性の欠如が生じている場合は、セクション2.11.4「テーブルまたはインデックスの再作成または修復」の手順を使用して、影響されるテーブルをダウングレードすることが必要になります。

ほとんどの場合、MySQL の同じリリースシリーズのバージョン内にとどまっているかぎり、MySQL 形式のファイルおよびデータファイルは、同じアーキテクチャーの異なる GA バージョン間で移動できます。

1 つのリリースシリーズから別のリリースシリーズにダウングレードする場合、テーブルストレージフォーマットの互換性が取れなくなる場合があります。その場合は、ダウングレードする前に mysqldump を使用してテーブルをダンプします。ダウングレードしたら、mysql あるいは mysqlimport を使用してダンプファイルをロードし、テーブルを再作成します。参考例は、セクション2.11.5「MySQL データベースのほかのマシンへのコピー」を参照してください。

ダウンロードした際の、下位互換性のないテーブル形式の変更の一般的な現象は、テーブルを開くことができないことです。そのような場合には、次の手順に従います。

1. ダウングレード先の古い MySQL サーバーを停止します。
2. ダウングレード元の新しい MySQL Server を再起動します。
3. mysqldump を使用して、古いサーバーからアクセスできなかったテーブルをすべてダンプしてダンプファイルを作成します。
4. 新しい MySQL Server を停止して古いサーバーを再起動します。
5. 古いサーバーにダンプファイルをリロードします。これでテーブルにアクセスできるはずです。

mysql データベース内のシステムテーブルが変更された場合は、ダウングレードによって一部の機能が失われたり、調整が必要になったりする場合があります。次にいくつかの例を示します。

- MySQL 5.1 では、トリガー作成には TRIGGER 権限が必要です。MySQL 5.0 では、TRIGGER 権限はなく、代わりに SUPER が必要です。MySQL 5.1 から 5.0 にダウングレードする場合は、5.1 で TRIGGER 権限を持っていたアカウントに SUPER 権限を付与する必要があります。
- トリガーは MySQL 5.0 で追加されたため、5.0 から 4.1 にダウングレードする場合は、トリガーをまったく使用できません。
- mysql.proc.comment カラム定義は、MySQL 5.1 と 5.5 との間で変更されました。5.5 から 5.1 へのダウングレード後、このテーブルは壊れていて修復が必要であるとみなされます。この問題を回避するには、ダウングレード先の MySQL のバージョンから mysql_upgrade を実行します。

2.11.2.1 MySQL 5.5 へのダウングレード

MySQL 5.6 から MySQL 5.5 にダウングレードする場合、MySQL 5.5 とは異なる MySQL 5.6 の動作または機能に関係する次の問題に注意してください。

システムテーブル

- MySQL 5.6 の mysql.user テーブルには password_expired カラムがあります。MySQL 5.5 の mysql.user テーブルにはありません。これは、MySQL 5.6 でパスワードが期限切れになっているアカウントは MySQL 5.5 では正常に機能しないことを意味します。

データ型

- TIME、DATETIME、および TIMESTAMP カラムについては、MySQL 5.6.4 より前に作成されたテーブルに必要なストレージは、5.6.4 以降で作成されたテーブルに必要なストレージとは異なります。これは、5.6.4 で、これらの時間型が少数部を持つことを許可するように変更されたためです。5.6.4 より前のバージョンにダウングレードする場合は、ダウングレードする前に mysqldump を使用して影響されるテーブルをダンプし、ダウングレード後にテーブルをリロードします。

InnoDB

- InnoDB 検索インデックス (型が FULLTEXT のもの) は MySQL 5.6.4 で導入され、5.6 シリーズの以前のリリースを含む、MySQL の以前のバージョンとは互換性がありません。ダウングレードを実行する前に、このようなインデックスは削除します。
- innodb_page_size 構成オプションで指定される InnoDB の小さいページサイズは MySQL 5.6.4 で導入され、5.6 シリーズの以前のリリースを含む、MySQL の以前のバージョンとは互換性がありません。小さい InnoDB ページサイズを使用するインスタンス内のすべての InnoDB テーブルをダンプし、テーブルを削除し、ダウングレード後に再作成してリロードします。

レプリケーション

- MySQL 5.6 では、relay-log.info ファイルは行カウントおよびレプリケーション遅延値を含むため、古いバージョンとはファイル形式が異なります。セクション17.2.2.2「スレーブステータスログ」を参照してください。スレーブサーバーを MySQL 5.6 より前のバージョンにダウングレードすると、古いサーバーはファイルを正しく読み取りません。これに対処するには、テキストエディタでファイルを変更し、行数を含む最初の行を削除します。
- MySQL 5.6.6 より、MySQL Server はバイナリログへの書き込みに Version 2 バイナリログイベントを使用します。Version 2 ログイベントを使用して書き込まれたバイナリログは、前のバージョンの MySQL Server では読み取れません。古いサーバーが読み取ることができる、Version 1 ログイベントを使用して書き込まれたバイナリログを生成するには、MySQL 5.6.6 以降のサーバーを --log-bin-use-v1-row-events=1 を使用して起動します。これは、バイナリログの書き込みに Version 1 イベントを使用することをサーバーに強制します。

2.11.3 テーブルまたはインデックスの再構築が必要かどうかのチェック

バイナリアップグレードまたはバイナリダウングレードでは、テーブルをダンプしてリロードせずに、あるバージョンの MySQL を既存のバージョンの上に「インプレース」でインストールします。

1. 既存のバージョンのサーバーが動作している場合は、それを停止します。
2. 別のバージョンの MySQL をインストールします。新しいバージョンが元のバージョンより新しい場合はアップグレードになり、古い場合はダウングレードになります。

3. 新しいバージョンのサーバーを起動します。

多くの場合、MySQL の以前のバージョンのテーブルは新しいバージョンで問題なく使用できます。ただし、このセクションで説明するように、テーブルまたはテーブルのインデックスの再構築が必要になるような変更が生じることもあります。ここで説明する問題のいずれかの影響を受けるテーブルがある場合は、必要に応じてそのテーブルまたはインデックスをセクション2.11.4「テーブルまたはインデックスの再作成または修復」に記載の手順を使用して再構築します。

テーブルの非互換性

ARCHIVE テーブルを含む MySQL 5.0 インストールから MySQL 5.1 へのバイナリアップグレードのあと、これらのテーブルにアクセスすると、mysql_upgrade または CHECK TABLE ... FOR UPGRADE を実行済みでもサーバーがクラッシュします。この問題を回避するには、アップグレードの前に mysqldump を使用してすべての ARCHIVE テーブルをダンプし、アップグレード後に MySQL 5.1 にリロードします。MySQL 5.1 から 5.0 へのバイナリダウングレードでも同じ問題が生じます。

アップグレードの問題は MySQL 5.6.4 で修正されています。サーバーは MySQL 5.0 で作成された ARCHIVE テーブルを開けます。ただし、引き続き推奨されるアップグレード手順は、アップグレードの前に 5.0 の ARCHIVE テーブルをダンプし、アップグレード後にリロードする手順です。

インデックスの非互換性

MySQL 5.6.3 では、ROW_FORMAT=DYNAMIC または ROW_FORMAT=COMPRESSED を使用する InnoDB テーブルについては、インデックスプリフィクスキーの長さ制限は 767 バイトから 3072 バイトに増加しました。詳細は、セクション14.6.7「InnoDB テーブル上の制限」を参照してください。この変更は、MySQL 5.5.14 にも移植されています。これら以上のリリースのいずれかから、長さ制限が短い以前のリリースにダウングレードする場合、インデックスプリフィクスキーを 767 バイトに切り捨てることができます。そうしないとダウングレードが失敗することがあります。この問題は、ダウングレードするサーバー上で構成オプション innodb_large_prefix が有効であった場合にのみ生じる可能性があります。

テーブルのダンプおよびリロードを実行せずにバイナリアップグレードを実行する場合、MySQL 4.1 から 5.1 以上に直接アップグレードすることはできません。これは、MySQL 5.0 の MyISAM テーブルインデックスの形式の非互換な変更によるものです。MySQL 4.1 から 5.0 にアップグレードし、すべての MyISAM テーブルを修復してください。その後、MySQL 5.0 から 5.1 にアップグレードし。テーブルをチェックして修復します。

文字セットまたは照合順序の処理の変更により、文字セットの順序が変更されることがあり、そのために、影響を受ける文字セットまたは照合順序を使用するインデックスのエントリの順序が正しくなくなることがあります。こうした変更から、いくつかの問題が生じることがあります。

- 比較の結果が前の結果と異なる
- インデックスエントリの順序付けが間違っているために一部のインデックス値を見つけることができない
- ORDER BY の結果の順序付けが間違っている
- CHECK TABLE によってレポートされるテーブルを修復する必要がある

これらの問題の解決方法は、インデックスを削除して再作成するか、テーブル全体をダンプしてリロードすることにより、影響を受ける文字セットまたは照合順序を使用するインデックスを再構築することです。場合によっては、影響を受けるカラムを変更して、別の照合順序を使用するようにすることができます。インデックスの再構築については、セクション2.11.4「テーブルまたはインデックスの再作成または修復」を参照してください。

再構築する必要のあるインデックスがテーブルにあるかどうかを確認するには、次のリストを参照してください。このリストは、インデックスの再構築が必要な文字セットまたは照合順序の変更が導入された MySQL のバージョンを示しています。各エントリは、変更が行われたバージョンと、その変更の影響を受ける文字セットまたは照合順序を示しています。変更が特定のバグレポートに関連付けられている場合は、そのバグ番号が示されています。

このリストは、バイナリアップグレードとバイナリダウングレードの両方に適用されます。たとえば、Bug #27877 は MySQL 5.1.24 で修正されたため、5.1.24 より前のバージョンから 5.1.24 以上のバージョンへのアップグレード、および 5.1.24 以上から 5.1.24 より前のバージョンへのダウングレードに適用されます。

多くの場合、CHECK TABLE ... FOR UPGRADE を使用して、インデックスの再構築が必要なテーブルを識別できます。次のメッセージがレポートされます。

```
Table upgrade required.
Please do "REPAIR TABLE `tbl_name`" or dump/reload to fix it!
```

これらの場合には、mysqlcheck --check-upgrade または mysql_upgrade も使用できます。これらは CHECK TABLE を実行します。ただし、CHECK TABLE を使用できるのは、ダウングレードではなくアップグレードのあとだけです。また、CHECK TABLE は、すべてのストレージエンジンに適用できるとは限りません。CHECK TABLE がサポートするストレージエンジンの詳細は、セクション13.7.2.2「CHECK TABLE 構文」を参照してください。

これらの変更により、インデックスの再構築が必要になります。

- MySQL 5.1.24 (Bug #27877)

  'ß' LATIN SMALL LETTER SHARP S (ドイツ語) を含むカラムの utf8_general_ci または ucs2_general_ci 照合順序を使用するインデックスに影響します。このバグフィックスでは、元の照合順序のエラーが修正されましたが、'ß' が、以前は異なるという比較結果だった文字と同じという比較結果になるという、非互換性が導入されました。

  影響を受けるテーブルは、MySQL 5.1.30 では CHECK TABLE ... FOR UPGRADE で検出できます (Bug #40053 を参照してください)。

  この問題の回避策は、MySQL 5.1.62、5.5.21、および 5.6.5 で実装されています。回避策には、影響されるカラムを utf8_general_mysql500_ci および ucs2_general_mysql500_ci 照合順序を使用するように変更することが含まれます。この照合順序では、元の 5.1.24 より前の utf8_general_ci および ucs2_general_ci の順序付けを維持します。

- MySQL 5.0.48、5.1.23 (Bug #27562)

  '`' GRAVE ACCENT、'[' LEFT SQUARE BRACKET、'\' REVERSE SOLIDUS、']' RIGHT SQUARE BRACKET、'~' TILDE のいずれかの文字を含むカラムで ascii_general_ci 照合順序を使用するインデックスに影響があります

  影響を受けるテーブルは、MySQL 5.1.29 では CHECK TABLE ... FOR UPGRADE で検出できます (Bug #39585 を参照してください)。

- MySQL 5.0.48、5.1.21 (Bug #29461)

  eucjpms、euc_kr、gb2312、latin7、macce、ujis のいずれかの文字セットを使用するカラムのインデックスに影響があります

  影響を受けるテーブルは、MySQL 5.1.29 では CHECK TABLE ... FOR UPGRADE で検出できます (Bug #39585 を参照してください)。

2.11.4 テーブルまたはインデックスの再作成または修復

このセクションでは、データ型または文字セットの処理方法に関する MySQL の変更後の、テーブルの再構築方法を説明します。たとえば、照合順序のエラーが修正され、その照合順序を使用する文字カラムのインデックスを更新するためにテーブルの再構築が必要になることがあります。(例については、セクション2.11.3「テーブルまたはインデックスの再構築が必要かどうかのチェック」を参照してください。)CHECK TABLE、mysqlcheck、または mysql_upgrade によって実行されるようなテーブルチェック操作によって示されるように、テーブルを修復またはアップグレードする必要がある場合があります。

テーブルを再構築する方法には、テーブルをダンプしてリロードする方法や、ALTER TABLE または REPAIR TABLE を使用する方法があります。REPAIR TABLE は MyISAM、ARCHIVE、および CSV の各テーブルのみに適用されます。

> 注記
>
> バイナリ (インプレース) アップグレードまたはダウングレード後に、MySQL の異なるバージョンがテーブルを処理しないためにテーブルを再構築する場合は、ダンプしてリロードする方法を使用する必要があります。アップグレードまたはダウングレードの前に、元のバージョンの MySQL を使用してテーブルをダンプします。次に、アップグレードまたはダウングレードのあとに、テーブルをリロードします。
>
> インデックスを再構築する目的のためだけにダンプしてリロードする方法を使ってテーブルを再構築する場合は、アップグレードまたはダウングレードの前でもあとでもダンプを実行できます。その場合でも、リロードはあとで行う必要があります。

テーブルをダンプしてリロードすることによって再構築するには、mysqldump を使用してダンプファイルを作成し、mysql でファイルをリロードします。

```
shell> mysqldump db_name t1 > dump.sql
shell> mysql db_name < dump.sql
```

単独のデータベース内のテーブルをすべて再構築する場合は、データベース名を、そのあとにテーブル名なしで指定します。

```
shell> mysqldump db_name > dump.sql
shell> mysql db_name < dump.sql
```

すべてのデータベース内のすべてのテーブルをリロードするには、--all-databases オプションを使用してください。

```
shell> mysqldump --all-databases > dump.sql
shell> mysql < dump.sql
```

ALTER TABLE でテーブルを再構築する場合は、「null」 変更を使用します。すなわち、すでに使用しているストレージエンジンを使用するように、テーブルを「変更」する ALTER TABLE ステートメントです。たとえば、t1 が InnoDB テーブルである場合、次のステートメントを利用します。

```
mysql> ALTER TABLE t1 ENGINE = InnoDB;
```

ALTER TABLE ステートメントで指定するストレージエンジンがわからない場合は、SHOW CREATE TABLE を使用してテーブル定義を表示します。

CHECK TABLE 操作でテーブルのアップグレードが必要であることが示されるために InnoDB テーブルを再構築する必要がある場合は、前述のように mysqldump を使用してダンプファイルを作成し、mysql でファイルをリロードします。CHECK TABLE 操作で、破損があることが示されたり InnoDB が失敗したりする場合は、innodb_force_recovery オプションを使用して InnoDB を再起動する方法について、セクション 14.19.2「InnoDB のリカバリの強制的な実行」を参照してください。CHECK TABLE が遭遇している問題のタイプを理解するには、セクション13.7.2.2「CHECK TABLE 構文」の InnoDB に関する注記を参照してください。

MyISAM、ARCHIVE、または CSV テーブルについては、テーブルチェック操作で破損があることやアップグレードが必要なことが示された場合は、REPAIR TABLE を使用できます。たとえば、MyISAM テーブルを修復するには、次のステートメントを使用します。

```
mysql> REPAIR TABLE t1;
```

mysqlcheck --repair は、コマンド行で REPAIR TABLE ステートメントへのアクセスを提供します。--databases オプションまたは --all-databases オプションをそれぞれ使用して、特定のデータベースまたはすべてのデータベースのすべてのテーブルを修復できるため、テーブル修復の方法として、より便利な場合があります。

```
shell> mysqlcheck --repair --databases db_name ...
shell> mysqlcheck --repair --all-databases
```

MySQL 5.1.24 で、Bug #27877 に対する修正 (utf8_general_ci および ucs2_general_ci 照合順序を修正しました) によって導入された非互換性については、MySQL 5.1.62、5.5.21、および 5.6.5 で回避策が実装されています。これらのいずれかのバージョンにアップグレードしてから、影響を受ける各テーブルを次の方法のいずれかを使用して変換します。どの場合も、回避策には、影響されるカラムを utf8_general_mysql500_ci および ucs2_general_mysql500_ci 照合順序を使用するように変更することが含まれます。この照合順序では、元の 5.1.24 より前の utf8_general_ci および ucs2_general_ci の順序付けを維持します。

- テーブルファイルをそのまま残すバイナリアップグレード後に、影響を受けるテーブルを変換するには、新しい照合順序を使用するようにテーブルを変更します。テーブル t1 に 1 つまたは複数の問題のある utf8 カラムがあるとします。テーブルをテーブルレベルで変換するには、次のようなステートメントを使用します。

  ```
  ALTER TABLE t1
  CONVERT TO CHARACTER SET utf8 COLLATE utf8_general_mysql500_ci;
  ```

  カラム固有ベースで変更を適用するには、次のようなステートメントを使用します (COLLATE 句を除き、最初に指定されていたカラム定義を必ず繰り返します)。

  ```
  ALTER TABLE t1
  MODIFY c1 CHAR(N) CHARACTER SET utf8 COLLATE utf8_general_mysql500_ci;
  ```

- ダンプしてリロードする手順を使用してテーブルをアップグレードする場合は、mysqldump を使用してテーブルをダンプし、ダンプファイル内の CREATE TABLE ステートメントを新しい照合順序を使用するように変更して、テーブルをリロードします。

適切な変更を行うと、CHECK TABLE はエラーをレポートしないはずです。

2.11.5 MySQL データベースのほかのマシンへのコピー

データベースを異なるアーキテクチャー間で移動する必要がある場合、mysqldump を使用して SQL ステートメントを含むファイルを作成します。次にそのファイルを別のマシンに転送して mysql クライアントへの入力として扱います。

> **注記**
>
> 同じ浮動小数点形式をサポートしている異なるアーキテクチャー間で、MyISAM テーブルの .frm、.MYI、および .MYD ファイルをコピーできます。(MySQL はバイトスワッピング問題を処理します。)セクション15.2「MyISAM ストレージエンジン」を参照してください。

利用できるオプションを表示するには mysqldump --help を使用します。

データベースを 2 つのマシンで間で移動するもっとも容易な (ただし、速くはない) 方法は、データベースを搭載したマシン上で次のコマンドを実行することです。

```
shell> mysqladmin -h 'other_hostname' create db_name
shell> mysqldump db_name | mysql -h 'other_hostname' db_name
```

データベースをリモートマシンから速度の遅いネットワークを介してコピーするには、次のコマンドを使用できます。

```
shell> mysqladmin create db_name
shell> mysqldump -h 'other_hostname' --compress db_name | mysql db_name
```

ダンプをファイルに保存して、そのファイルをターゲットマシンに転送し、そのファイルをそこのデータベースにロードすることもできます。たとえば、データベースをソースマシンの圧縮ファイルに次のようにダンプできます。

```
shell> mysqldump --quick db_name | gzip > db_name.gz
```

データベースのコンテンツを含んだファイルをターゲットマシンに転送し、そこで次のコマンドを実行します。

```
shell> mysqladmin create db_name
shell> gunzip < db_name.gz | mysql db_name
```

データベースの転送に mysqldump および mysqlimport を使用することもできます。大きなテーブルの場合、これは単に mysqldump を使用するよりも非常に速いです。次のコマンドで、DUMPDIR は mysqldump の出力の保存に使用されるディレクトリのフルパス名です。

最初に、その出力ファイルのディレクトリを作成してデータベースをダンプします。

```
shell> mkdir DUMPDIR
shell> mysqldump --tab=DUMPDIR db_name
```

次に DUMPDIR ディレクトリのファイルをターゲットマシンの相当するディレクトリに転送して、そのファイルをそこの MySQL にロードします。

```
shell> mysqladmin create db_name          # create database
shell> cat DUMPDIR/*.sql | mysql db_name   # create tables in database
shell> mysqlimport db_name DUMPDIR/*.txt   # load data into tables
```

mysql データベースをコピーすることを忘れないでください。付与テーブルがそこに格納されているからです。新しいマシンで mysql データベースが用意できるまで、コマンドを MySQL root ユーザーとして実行しなければならない場合があります。

mysql データベースを新しいマシンにインポートしたら、mysqladmin flush-privileges を実行してサーバーに付与テーブルの情報をロードさせます。

2.12 環境変数

このセクションでは、MySQL で直接的あるいは間接的に使用されるすべての環境変数について説明します。これらの多くは本ドキュメントの別の場所にもあります。

コマンド行のどのようなオプションも、オプションファイルおよび環境変数で指定された値に優先し、オプションファイルの値は環境変数の値に優先することにご留意ください。

多くの場合、MySQL の動作を変更するには、環境変数ではなくオプションファイルを使用する方が好ましいといえます。セクション4.2.6「オプションファイルの使用」を参照してください。

変数	説明
CXX	C++ コンパイラの名前 (CMake 実行用)。
CC	C コンパイラの名前 (CMake 実行用)。
DBI_USER	Perl DBI のデフォルトユーザー名。
DBI_TRACE	Perl DBI のトレースオプション。
HOME	mysql 履歴ファイルのデフォルトのパスは $HOME/.mysql_history です。
LD_RUN_PATH	libmysqlclient.so の位置指定に使用。
LIBMYSQL_ENABLE_CLEARTEXT_PLUGIN	mysql_clear_password 認証プラグインを有効化。セクション6.3.8.7「クライアント側のクリアテキスト認証プラグイン」を参照してください。
LIBMYSQL_PLUGIN_DIR	クライアントプラグインを検索するディレクトリ。
LIBMYSQL_PLUGINS	プリロードするクライアントプラグイン。
MYSQL_DEBUG	デバッグ中のデバッグトレースオプション。
MYSQL_GROUP_SUFFIX	オプショングループのサフィクスの値 (--defaults-group-suffix などの指定)。
MYSQL_HISTFILE	mysql 履歴ファイルへのパス。この変数を設定すると、その値は $HOME/.mysql_history のデフォルトをオーバーライドします。
MYSQL_HISTIGNORE	$HOME/.mysql_history にログ記録しないステートメントを指定するパターン。
MYSQL_HOME	サーバー固有の my.cnf ファイルが存在するディレクトリへのパス。
MYSQL_HOST	mysql コマンド行クライアントが使用するデフォルトのホスト名。
MYSQL_PS1	mysql コマンド行クライアントで使用するコマンドプロンプト。
MYSQL_PWD	mysqld に接続する際のデフォルトのパスワード。これを使用することはセキュアではありません。セクション6.1.2.1「パスワードセキュリティーのためのエンドユーザーガイドライン」を参照してください。
MYSQL_TCP_PORT	デフォルトの TCP/IP ポート番号。
MYSQL_TEST_LOGIN_FILE	.mylogin.cnf ログインファイルの名前。
MYSQL_UNIX_PORT	localhost への接続に使用される、デフォルトの Unix ソケットファイル名。
PATH	シェルが MySQL プログラムの検索に使用します。
TMPDIR	一時ファイルが作成されるディレクトリ。
TZ	ローカルタイムゾーンに設定するようにしてください。セクションB.5.4.6「タイムゾーンの問題」を参照してください。
UMASK	ファイルを作成する際のユーザーファイル作成モード。表のあとにある注釈を参照してください。
UMASK_DIR	ディレクトリを作成する際のユーザーディレクトリ作成モード。表のあとにある注釈を参照してください。
USER	mysqld に接続する際の Windows のデフォルトのユーザー名。

mysql の履歴ファイルの詳細は、セクション4.5.1.3「mysql のロギング」を参照してください。

MYSQL_TEST_LOGIN_FILE は、ログインファイル (mysql_config_editor によって作成されるファイル) のパス名です。設定されていない場合、デフォルト値は Windows では %APPDATA%\MySQL\.mylogin.cnf ディレクトリ、Windows 以外のシステムでは $HOME/.mylogin.cnf です。セクション4.6.6「mysql_config_editor — MySQL 構成ユーティリティー」を参照してください。

UMASK 変数および UMASK_DIR 変数は、その名前にもかかわらず、マスクではなくモードとして使用されます。

- UMASK が設定されている場合、mysqld は ($UMASK | 0600) をファイル作成のモードとして使用し、新しく作成されるファイルのモードは 0600 から 0666 の範囲になります (すべて 8 進数の値)。
- UMASK_DIR が設定されている場合、mysqld は ($UMASK_DIR | 0700) をディレクトリ作成のベースモードとして使用し、次に ~(~$UMASK & 0666) との AND が取られます。そのため新しく作成されるファイルのモードは 0700 から 0777 の範囲になります (すべて 8 進数の値)。AND 演算によってディレクトリモードから読み取り/書き込み権が削除されることがありますが、実行権が削除されることはありません。

MySQL では、UMASK または UMASK_DIR の値がゼロで始まる場合、その値は 8 進数と見なされます。

2.13 Perl のインストールに関する注釈

PerlDBIモジュールはデータベースアクセスのための一般的なインタフェースを提供します。変更なしで多くの異なったデータベースエンジンで作動する DBI スクリプトを書くことができます。DBI を使用するには、DBI モジュール、および DataBase Driver (DBD) モジュールを、アクセスする各データベースサーバーのタイプごとにインストールする必要があります。MySQL の場合、このドライバは DBD::mysql モジュールです。

MySQL ベンチマークスクリプトを実行する場合は、Perl および DBI の DBD::MySQL モジュールがインストールされていなければなりません。セクション8.12.2「MySQL ベンチマークスイート」を参照してください。

> 注記
>
> MySQL の配布には Perl のサポートは含まれていません。必要なモジュールは、Unix の場合は http://search.cpan.org から、Windows では ActiveState ppm プログラムを使用して取得できます。次のセクションでその方法について説明します。

DBI/DBD インタフェースは Perl 5.6.0 を必要とし、5.6.1 以降が推奨されます。それより古いバージョンの Perl の場合、DBI は機能しません。DBD::mysql 4.009 以上を使用するようにしてください。それより前のバージョンは使用可能ですが、MySQL 5.6 の全機能をサポートしているわけではありません。

2.13.1 Unix に Perl をインストールする

MySQL の Perl のサポートには MySQL のクライアントプログラムサポート (ライブラリおよびヘッダーファイル) をインストールする必要があります。ほとんどのインストール方法で、必要なファイルがインストールされます。MySQL を RPM ファイルからインストールする場合、開発者 RPM もインストールしてください。クライアントプログラムはクライアント RPM にありますが、クライアントプログラムサポートは開発者 RPM にあります。

Perl のサポートに必要なファイルは、http://search.cpan.org の CPAN (Comprehensive Perl Archive Network) から入手できます。

Unix に Perl モジュールをインストールするには CPAN モジュールを使用するのがいちばん簡単です。例:

```
shell> perl -MCPAN -e shell
cpan> install DBI
cpan> install DBD::mysql
```

DBD::mysql インストールは多くのテストを実行します。これらのテストでは、デフォルトのユーザー名とパスワードを使用してローカルの MySQL サーバーに接続を試みます。(デフォルトのユーザー名は、Unix ではログイン名であり、Windows では ODBC です。デフォルトのパスワードは「パスワードなし」です)。サーバーにそれらの値で接続できない場合 (たとえば、アカウントにパスワードを設定している場合)、テストは失敗します。force install DBD::mysql を使用すると、失敗したテストを無視できます。

DBI には Data::Dumper モジュールが必要です。それはインストールされている場合があります。もしされていない場合、DBI をインストールする前にそれをインストールするようにしてください。

モジュールの配布を圧縮 tar アーカイブの形でダウンロードして、モジュールを手動でビルドすることもできます。たとえば、DBI 配布をアンパックしてビルドするには、次のような手順に従います。

1. 配布を現在のディレクトリにアンパックします。

   ```
   shell> gunzip < DBI-VERSION.tar.gz | tar xvf -
   ```

   このコマンドは、DBI-VERSION という名前のディレクトリを作成します。

2. アンパックした配布のトップレベルのディレクトリに場所を変更します。

   ```
   shell> cd DBI-VERSION
   ```

3. 配布をビルドしてすべてをコンパイルします。

   ```
   shell> perl Makefile.PL
   shell> make
   shell> make test
   shell> make install
   ```

make test コマンドはモジュールが動作していることを確認するために重要です。DBD::mysql のインストール中にそのコマンドを実行してインタフェースのコードを実行するには、MySQL サーバーが動作していなければなりません。そうしないとそのテストは失敗します。

新しいリリースの MySQL をインストールする際は必ず DBD::mysql 配布を再構築して再インストールするのがいいでしょう。これにより、MySQL クライアントライブラリの最新バージョンが正しくインストールされていることが保証されます。

Perl モジュールをシステムディレクトリにインストールするアクセス権がない場合、またはローカルの Perl をインストールする場合、次のリファレンスが有用でしょう。http://learn.perl.org/faq/perlfaq8.html#How-do-I-keep-my-own-module-library-directory-。

2.13.2 Windows に ActiveState Perl をインストールする

Windows 上で、MySQL DBD モジュールを ActiveState Perl でインストールするには次の手順に従います。

1. http://www.activestate.com/Products/ActivePerl/ から ActiveState Perl を入手してインストールします。
2. コンソールウィンドウを開きます。
3. 必要に応じて HTTP_proxy 変数を設定します。たとえば、次のような設定を試してみます。

   ```
   C:\> set HTTP_proxy=my.proxy.com:3128
   ```

4. PPM プログラムを起動します。

   ```
   C:\> C:\perl\bin\ppm.pl
   ```

5. まだ DBI をインストールしていない場合は、インストールします。

   ```
   ppm> install DBI
   ```

6. これが成功したら、次のコマンドを実行します。

   ```
   ppm> install DBD-mysql
   ```

この手順は ActiveState Perl 5.6 以降で機能するはずです。

手順が機能しない場合は、代わりに ODBC ドライバをインストールして ODBC から MySQL サーバーに接続するようにしてください。

```
use DBI;
$dbh= DBI->connect("DBI:ODBC:$dsn",$user,$password) ||
  die "Got error $DBI::errstr when connecting to $dsn\n";
```

2.13.3 Perl DBI/DBD インタフェース使用の問題

Perl が ../mysql/mysql.so モジュールを見つけることができない場合、問題はおそらく Perl が libmysqlclient.so 共有ライブラリの場所がわからないことです。この問題は次の方法のいずれかで解決できるはずです。

- libmysqlclient.so をほかの共有ライブラリがある (おそらく /usr/lib あるいは /lib) ディレクトリにコピーします。
- DBD::mysql のコンパイルに使用される -L オプションを、libmysqlclient.so の実際の場所を反映するように変更します。
- Linux では、libmysqlclient.so があるディレクトリのパス名を /etc/ld.so.conf ファイルに追加できます。
- libmysqlclient.so があるディレクトリのパス名を LD_RUN_PATH 環境変数に追加します。システムの中には LD_LIBRARY_PATH を使用しているものもあります。

リンカーが見つけられないほかのライブラリがある場合も、-L オプションを変更する必要がある場合があります。たとえば、libc が /lib にあり、リンクコマンドが -L/usr/lib を指定しているためにリンカーがそれを見つけられない場合、-L オプションを -L/lib に変更するかあるいは -L/lib を既存のリンクコマンドに追加します。

DBD::mysql から次のエラーが表示された場合、おそらく gcc (あるいは gcc でコンパイルされた旧バイナリ) を使用しているでしょう。

```
/usr/bin/perl: can't resolve symbol '__moddi3'
/usr/bin/perl: can't resolve symbol '__divdi3'
```

mysql.so ライブラリがビルドされるときに、-L/usr/lib/gcc-lib/... -lgcc をリンクコマンドに追加します (Perl クライアントのコンパイル時に、make の出力で mysql.so を確認します)。-L オプションは、システム上の libgcc.a があるディレクトリのパス名を指定するようにしてください。

この問題の別の原因は Perl および MySQL が両方とも gcc でコンパイルされていない場合です。この場合、両方を gcc でコンパイルすることでこの不一致を解決できます。

第 3 章 チュートリアル

目次

3.1 サーバーへの接続とサーバーからの切断 197
3.2 クエリーの入力 198
3.3 データベースの作成と使用 200
3.3.1 データベースの作成と選択 201
3.3.2 テーブルの作成 202
3.3.3 テーブルへのデータのロード 203
3.3.4 テーブルからの情報の取り出し 204
3.4 データベースとテーブルに関する情報の取得 215
3.5 バッチモードでの MySQL の使用 216
3.6 一般的なクエリーの例 217
3.6.1 カラムの最大値 217
3.6.2 特定のカラムの最大値が格納されている行 218
3.6.3 グループごとのカラムの最大値 218
3.6.4 特定のカラムのグループごとの最大値が格納されている行 218
3.6.5 ユーザー定義の変数の使用 219
3.6.6 外部キーの使用 219
3.6.7 2 つのキーを使用した検索 221
3.6.8 日ごとの訪問数の計算 221
3.6.9 AUTO_INCREMENT の使用 221
3.7 Apache での MySQL の使用 223

この章では、mysql クライアントプログラムを使用して、簡単なデータベースを作成して使用する方法を示すことで、MySQL のチュートリアルを提供します。mysql (「端末モニター」または単に「モニター」と呼ばれることもあります) は、MySQL Server への接続、クエリーの実行、および結果の表示を可能にするインタラクティブなプログラムです。mysql は、バッチモードでも使用できます。クエリーを前もってファイルに入れておき、mysql にファイルのコンテンツを実行するよう指示します。ここでは、mysql の両方の使用方法について説明します。

mysql で提供されているオプションのリストを表示するには、--help オプションを指定して mysql を起動します。

```
shell> mysql --help
```

この章では、mysql がマシンにインストールされていることと、接続可能な MySQL Server が存在することを前提にしています。そうでない場合は、MySQL 管理者に問い合わせてください。(ご自身が管理者の場合は、このドキュメントの第5章「MySQL サーバーの管理」などの関連部分を参照してください。)

この章では、データベースの設定と使用の手順全体について説明します。既存のデータベースへのアクセス手順だけを知りたい場合は、データベースおよびそれに含まれるテーブルの作成方法に関するセクションはスキップすることもできます。

この章はチュートリアルであるため、必然的に多くの詳細が省略されています。ここで説明されているトピックの詳細については、ドキュメントの関連セクションを参照してください。

3.1 サーバーへの接続とサーバーからの切断

サーバーに接続するには、通常は mysql の起動時に MySQL ユーザー名を入力し、多くの場合はパスワードも入力する必要があります。ログインするマシンとは異なるマシンでサーバーが稼働している場合は、ホスト名も指定する必要があります。接続に使用する接続パラメータ (使用するホスト、ユーザー名、およびパスワード) については、管理者に問い合わせてください。正しいパラメータがわかったら、次の方法で接続できます。

```
shell> mysql -h host -u user -p
Enter password: ********
```

host は MySQL サーバーが稼働しているホストの名前、user はユーザーの MySQL アカウントのユーザー名です。設定に応じて適切な値で置き換えてください。******** はユーザーのパスワードです。mysql で「Enter password:」というプロンプトが表示されたら入力してください。

それが機能すると、照会情報に続いて mysql> プロンプトが表示されるはずです。

```
shell> mysql -h host -u user -p
Enter password: ********
Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 25338 to server version: 5.6.23-standard

Type 'help;' or '\h' for help. Type '\c' to clear the buffer.

mysql>
```

mysql> プロンプトは、mysql がコマンド入力の準備が整っていることを示します。

MySQL が稼働しているマシンと同じマシンにログインしている場合は、ホストを省略して次を使用できます。

```
shell> mysql -u user -p
```

ログイン時に次のようなエラーメッセージが表示される場合があります。 ERROR 2002 (HY000): Can't connect to local MySQL server through socket '/tmp/mysql.sock' (2)。これは、MySQL サーバーデーモン (Unix) またはサービス (Windows) が起動していないことを示しています。管理者に問い合わせるか、オペレーティングシステムに応じて第2章「MySQL のインストールと更新」の適切なセクションを参照してください。

ログイン時によく発生するほかの問題については、セクションB.5.2「MySQL プログラム使用時の一般的なエラー」を参照してください。

一部の MySQL インストールでは、ローカルホストで稼働しているサーバーに、ユーザーが匿名 (名前を指定しない) ユーザーとして接続できます。これに該当するマシンでは、オプションを何も指定せずに mysql を起動すると、そのサーバーに接続できるはずです。

```
shell> mysql
```

接続に成功したら、QUIT (または \q) と mysql> プロンプトに入力することで、いつでも切断できます。

```
mysql> QUIT
Bye
```

Unix では、Control+D を押しても切断できます。

以降のセクションに示すほとんどの例では、サーバーに接続していることを前提にしています。このことは、mysql> プロンプトによって示されます。

3.2 クエリーの入力

前のセクションで説明したように、サーバーに接続していることを確認します。それだけでは操作するデータベースは選択されていませんが、それでかまいません。この時点では、テーブルの作成、テーブルへのデータのロード、テーブルからのデータの取り出しに今すぐ進むのではなく、クエリーの発行方法を学ぶことが重要です。このセクションでは、mysql の動作に習熟するために、いくつかのクエリーを使用してコマンド入力の基本原則について説明します。

サーバーにバージョン番号と現在の日付を照会する単純なコマンドを次に示します。mysql> プロンプトに続けて、次に示すように入力してから Enter を押します。

```
mysql> SELECT VERSION(), CURRENT_DATE;
+--------------+--------------+
| VERSION()    | CURRENT_DATE |
+--------------+--------------+
| 5.6.1-m4-log | 2010-08-06   |
+--------------+--------------+
1 row in set (0.01 sec)
mysql>
```

このクエリーには、mysql に関するいくつかの事項が示されています。

- 通常、コマンドは SQL ステートメントとそれに続くセミコロンで構成されます。(セミコロンを省略できる例外もいくつかあります。前述の QUIT もその 1 つです。ほかのものについてはあとで説明します。)
- コマンドを発行すると、mysql はそれをサーバーに送信して実行し、結果を表示してから、もう 1 つの mysql> プロンプトを出力して、次のコマンドの準備ができたことを示します。
- mysql は、クエリーの出力を表形式 (行とカラム) で表示します。最初の行には、カラムのラベルが表示されます。以降の行はクエリーの結果です。通常、カラムラベルは、データベーステーブルからフェッチされるカラムの名前です。次の例のように、テーブルカラムではなく式の値を取得する場合、mysql ではカラムのラベルとして式自体が使用されます。

- mysql では、返された行の数とクエリーの実行にかかった時間が表示されるため、サーバーの大まかなパフォーマンスがわかります。これらの値は、CPU 時間やマシン時間ではなく時計の時間で表されるため、また、サーバー負荷やネットワーク待機時間などの要因に影響されるため、正確ではありません。(簡略化のために、この章のほかの例では「rows in set」の行が省略されている場合もあります。)

キーワードは大文字でも小文字でも入力できます。次のクエリーは同等です。

```
mysql> SELECT VERSION(), CURRENT_DATE;
mysql> select version(), current_date;
mysql> SeLeCt vErSiOn(), current_DATE;
```

次に、別のクエリーを示します。これは、mysql を簡単な計算機として使用できることを示しています。

```
mysql> SELECT SIN(PI()/4), (4+1)*5;
+------------------+---------+
| SIN(PI()/4)      | (4+1)*5 |
+------------------+---------+
| 0.70710678118655 |      25 |
+------------------+---------+
1 row in set (0.02 sec)
```

これまでに示したクエリーは、比較的に短い、1 行のステートメントでした。1 行に複数のステートメントを入力することもできます。各ステートメントの末尾にはセミコロンを付加してください。

```
mysql> SELECT VERSION(); SELECT NOW();
+-------------+
| VERSION()   |
+-------------+
| 5.6.1-m4-log |
+-------------+
1 row in set (0.00 sec)
+---------------------+
| NOW()               |
+---------------------+
| 2010-08-06 12:17:13 |
+---------------------+
1 row in set (0.00 sec)
```

1 つのコマンド全体を 1 行に入力する必要はないため、複数の行を必要とする長いコマンドでも問題ありません。mysql は、入力行の末尾を探すのではなく、終端のセミコロンを探すことによってステートメントの末尾を判定します。(つまり、mysql は自由形式の入力を受け入れます。入力行を収集しますが、セミコロンが見つかるまでは実行しません。)

次に、単純な複数行ステートメントを示します。

```
mysql> SELECT
    -> USER()
    -> ,
    -> CURRENT_DATE;
+---------------+--------------+
| USER()        | CURRENT_DATE |
+---------------+--------------+
| jon@localhost | 2010-08-06   |
+---------------+--------------+
```

この例では、複数行にわたるクエリーの最初の行を入力したあと、プロンプトが mysql> から -> に変化することに注意してください。これは、mysql がまだ完全なステートメントを検出しておらず、残りの入力を待機していることを示しています。プロンプトは貴重なフィードバックを提供してくれる味方です。そのフィードバックを使用すれば、mysql が何を待機しているのかを、常に認識することができます。

入力途中のコマンドの実行を取り消すには、\c と入力します。

```
mysql> SELECT
    -> USER()
    -> \c
mysql>
```

ここでもプロンプトに注目してください。\c の入力後に mysql> に戻り、mysql が新しいコマンドに対して準備できていることを示すフィードバックが与えられます。

次の表に、表示される各プロンプトとそれらが示す mysql の状態をまとめます。

プロンプト	意味
mysql>	新しいコマンドを入力できます。

プロンプト	意味
->	複数行コマンドの次の行を待機しています。
'>	単一引用符 (「'」) で始まった文字列が完了するまで次の行を待機しています。
">	二重引用符 (「"」) で始まった文字列が完了するまで次の行を待機しています。
`>	逆引用符 (「`」) で始まった識別子が完了するまで次の行を待機しています。
/*>	/* で始まったコメントが完了するまで次の行を待機しています。

一般に、1 行のコマンドを発行しようとして終端のセミコロンを忘れると、複数行ステートメントが発生します。この場合、mysql は追加の入力を待機します。

```
mysql> SELECT USER()
    ->
```

これが起きた場合 (ステートメントを入力したのに -> プロンプトの応答が示される) は、mysql がセミコロンを待機している可能性が高くなります。プロンプトの意味に気付かなければ、何をするべきか理解するまでに時間がかかるかもしれません。セミコロンを入力してステートメントを完了すると、mysql で実行されます。

```
mysql> SELECT USER()
    -> ;
+---------------+
| USER()        |
+---------------+
| jon@localhost |
+---------------+
```

'> および "> プロンプトは、文字列の収集中に発生します (MySQL が文字列の完了を待機しているという意味です)。MySQL では、「'」 文字または 「"」 文字で囲んで文字列を記述でき (たとえば、'hello' や "goodbye" など)、mysql では、複数行にわたって文字列を入力することができます。'> または "> プロンプトが表示される場合、それは 「'」 または 「"」 引用符で始まる文字列を含む行を入力したが、その文字列を終了させる対応する引用符をまだ入力していないことを意味します。このような場合、引用符の入力を忘れていることがよくあります。例:

```
mysql> SELECT * FROM my_table WHERE name = 'Smith AND age < 30;
    '>
```

この SELECT ステートメントを入力し、Enter を押して結果を待っても、何も起きません。このクエリーがなぜそれほど時間がかかるのかと不思議に思うのではなく、'> プロンプトで示される手がかりに注目してください。このプロンプトは、mysql が完了していない文字列の残りを待機していることを示しています。(このステートメントの間違いに気付きましたか。文字列 'Smith には、2 番目の一重引用符がありません。)

では、どうしますか。もっとも簡単なのは、コマンドを取り消すことです。ただし、この場合は単に \c と入力することはできません。なぜなら、mysql で収集中の文字列の一部とみなされるからです。代わりに、閉じる引用符を入力して (つまり、文字列が完了したことを mysql に認識させ)、次に \c と入力します。

```
mysql> SELECT * FROM my_table WHERE name = 'Smith AND age < 30;
    '> '\c
mysql>
```

プロンプトが mysql> に戻り、mysql が新しいコマンドに対して準備できていることが示されます。

`> プロンプトは '> および "> プロンプトと同様ですが、逆引用符を開始したがまだ終了していないことを示しています。

'>、">、および `> の各プロンプトが何を意味するかを理解することが重要です。なぜなら、誤って未終了の文字列を入力した場合、それ以降の入力行は QUIT を含む行も含めて、mysql に無視されるように見えるからです。特に、現在のコマンドを取り消すにはまず終端の引用符を入力する必要があることを理解していないと、これにはかなり困惑する可能性があります。

3.3 データベースの作成と使用

コマンドの入力方法を理解したら、データベースにアクセスすることができます。

何匹かのペット (menagerie、動物) を家で飼っているとします。これらのペットについてさまざまな種類の情報の記録をつける場合を考えます。これは、データを保持するためのテーブルを作成し、必要な情報をテーブルにロードすることで実現できます。次に、テーブルからデータを取り出すことで、ペットに関する各種の質問に回答できます。このセクションでは、次の操作の実行方法について説明します。

- データベースを作成する
- テーブルを作成する
- テーブルにデータをロードする
- さまざまな方法でテーブルからデータを取り出す
- 複数のテーブルを使用する

menagerie データベースは意図的に単純になっていますが、類似のデータベースを使用するような実際の状況を考えることは難しくありません。たとえば、これに似たデータベースを使用して農場主が家畜の記録をつけたり、獣医が患者の記録をつけたりできます。以降のセクションで使用するいくつかのクエリーとサンプルデータを含む menagerie の配布を、MySQL の Web サイトから入手できます。圧縮 tar ファイル形式および Zip 形式の両方で、https://dev.mysql.com/doc/ で入手できます。

SHOW ステートメントを使用して、サーバーに現在どのようなデータベースが存在するかを調べます。

```
mysql> SHOW DATABASES;
+----------+
| Database |
+----------+
| mysql    |
| test     |
| tmp      |
+----------+
```

mysql データベースは、ユーザーのアクセス権限を記述します。test データベースは、多くの場合、ユーザーが各種の試行をするための作業スペースとして用意されています。

このステートメントで表示されるデータベースのリストは、マシンによって異なる場合があります。ユーザーが SHOW DATABASES 権限を持っていない場合、ユーザーにまったく権限のないデータベースは SHOW DATABASES で表示されません。セクション13.7.5.15「SHOW DATABASES 構文」を参照してください。

test データベースが存在する場合は、アクセスしてみます。

```
mysql> USE test
Database changed
```

QUIT と同様に、USE にもセミコロンは必要ありません。(これらのステートメントの末尾にセミコロンを付加しても問題はありません。) USE ステートメントは、1 行に記述する必要があるという点でも特殊です。

test データベースへのアクセス権限を持っている場合は以降の例に使用できますが、そのデータベース内に作成した内容は、アクセス権限を持っているほかのユーザーによって削除される可能性があります。この理由から、自分専用のデータベースを使用する許可を MySQL 管理者に依頼するとよいでしょう。自分のデータベースに menagerie という名前を付けるとします。管理者は次のようなコマンドを実行する必要があります。

```
mysql> GRANT ALL ON menagerie.* TO 'your_mysql_name'@'your_client_host';
```

ここで、your_mysql_name はユーザーに割り当てられた MySQL ユーザー名、your_client_host はサーバーに接続するホストです。

3.3.1 データベースの作成と選択

管理者がユーザーのアクセス権を設定する際にユーザー用のデータベースを作成した場合、ユーザーはそのデータベースを使用し始めることができます。そうでない場合は、ユーザー自身で作成する必要があります。

```
mysql> CREATE DATABASE menagerie;
```

Unix の場合、SQL キーワードとは異なり、データベース名では大文字と小文字が区別されるため、このデータベースは常に menagerie と表記する必要があり、Menagerie や MENAGERIE などの異なる表記は使用できません。これはテーブル名にも当てはまります。(Windows の場合、この制限は適用されませんが、データベースとテーブルの表記は 1 つのクエリー内では統一する必要があります。ただし、さまざまな理由から、データベースの作成時に使用した表記を常に使用することをお勧めします。)

> **注記**
>
> データベースの作成時に ERROR 1044 (42000): Access denied for user 'monty'@'localhost' to database 'menagerie' のようなエラーが表示される場合があります。これは、ユーザーのアカウントには必要な権限がないことを示しています。これに

> ついては管理者に問い合わせるか、セクション6.2「MySQL アクセス権限システム」を参照してください。

データベースを作成しても、そのデータベースは選択されません。使用するには明示的に選択する必要があります。menagerie を現在のデータベースにするには、次のコマンドを使用します。

```
mysql> USE menagerie
Database changed
```

データベースの作成は 1 回だけ必要ですが、使用するには mysql セッションを開始するたびにデータベースを選択する必要があります。そのためには、例のように USE ステートメントを発行します。または、mysql を起動するときにコマンド行でデータベースを選択できます。必要な接続パラメータをすべて指定したあとに、データベースの名前を指定します。例:

```
shell> mysql -h host -u user -p menagerie
Enter password: ********
```

> **重要**
>
> このコマンド内の menagerie はパスワードではありません。コマンド行で -p オプションのあとにパスワードを入力する場合は、間にスペースを入れずに入力する必要があります (たとえば、-p mypassword ではなく、-pmypassword)。ただし、コマンド行にパスワードを入力することは、同じマシンにログインしているほかのユーザーに覗き見られるおそれがあるため、お勧めできません。

> **注記**
>
> SELECT DATABASE() を使用すると、現在どのデータベースが選択されているかをいつでも調べることができます。

3.3.2 テーブルの作成

データベースの作成は簡単な部分ですが、SHOW TABLES が示すとおり、この時点では空です。

```
mysql> SHOW TABLES;
Empty set (0.00 sec)
```

難しい部分は、データベースの構造、つまり、どのようなテーブルが必要で各テーブルにどのようなカラムを含めるかを決定することです。

各ペットの記録を格納するテーブルが必要です。このテーブルには、pet という名前を付けることができ、少なくとも各ペットの名前を含めるべきです。名前自体には深い意味はないため、このテーブルにはほかの情報も含めるとよいでしょう。たとえば、家族の複数のメンバーがペットを飼っている場合は、各ペットの所有者を記録することができます。種や性別などの基本的な説明も記録できます。

年齢はどうでしょうか。重要な情報ではありますが、データベースに格納するには適しません。年齢は時間の経過によって変化するため、記録を頻繁に更新する必要が生じます。代わりに、生年月日などの固定値を格納する方が適切です。そうしておけば、年齢が必要になったときに、現在の日付と生年月日の差として計算することができます。MySQL には日付演算を行う関数が用意されているため、これは難しくありません。年齢の代わりに生年月日を格納することには、ほかの利点もあります。

- たとえば、ペットの誕生日が近づいたらリマインダを生成するといったタスクにデータベースを使用できます。(このようなクエリーには意味がないと感じる場合、現在の週や月で誕生祝いのメッセージを送信する必要のあるクライアントを特定するためにビジネスデータベースを使用する場合と、コンピュータ支援の接客という点で同じ問題であることに注目してください。)
- 現在の日付ではなく、別の日付を基にして年齢を計算することもできます。たとえば、データベースに死亡日を格納すると、ペットが何歳で死んだかを簡単に計算できます。

pet テーブルに含めると役立ちそうな情報はほかにもあるでしょうが、ここまでに挙げた名前 (name)、所有者 (owner)、種 (species)、性別 (sex)、生年月日 (birth)、および死亡年月日 (death) で十分です。

CREATE TABLE ステートメントを使用して、テーブルのレイアウトを指定します。

```
mysql> CREATE TABLE pet (name VARCHAR(20), owner VARCHAR(20),
    -> species VARCHAR(20), sex CHAR(1), birth DATE, death DATE);
```

name、owner、および species の各カラムではカラム値の長さが変化するため、これらのカラムに VARCHAR を選択することは適切です。これらのカラム定義の長さは、すべて同じである必要はなく、20 である必要もありま

せん。通常は、1 から 65535 までの範囲で、もっとも適切と思われる任意の長さを選択できます。選択が適切でなく、あとでより長いフィールドが必要になった場合は、MySQL の ALTER TABLE ステートメントを利用できます。

ペットのレコードで性別を表す値としては、'm' と 'f'、または 'male' と 'female' など、いくつかの種類を選択できます。1 文字の 'm' と 'f' を使用するのがもっとも簡単です。

birth カラムと death カラムに DATE データ型を使用することはかなり明白な選択です。

テーブルを作成したあとは、SHOW TABLES で何か出力されるはずです。

```
mysql> SHOW TABLES;
+--------------------+
| Tables in menagerie |
+--------------------+
| pet                |
+--------------------+
```

想定どおりにテーブルが作成されたことを確認するには、DESCRIBE ステートメントを使用します。

```
mysql> DESCRIBE pet;
+---------+-------------+------+-----+---------+-------+
| Field   | Type        | Null | Key | Default | Extra |
+---------+-------------+------+-----+---------+-------+
| name    | varchar(20) | YES  |     | NULL    |       |
| owner   | varchar(20) | YES  |     | NULL    |       |
| species | varchar(20) | YES  |     | NULL    |       |
| sex     | char(1)     | YES  |     | NULL    |       |
| birth   | date        | YES  |     | NULL    |       |
| death   | date        | YES  |     | NULL    |       |
+---------+-------------+------+-----+---------+-------+
```

DESCRIBE ステートメントは、テーブル内のカラムの名前や型を忘れた場合などに、いつでも使用できます。

MySQL のデータ型に関する詳細は、第11章「データ型」を参照してください。

3.3.3 テーブルへのデータのロード

テーブルを作成したら、データを移入する必要があります。これには、LOAD DATA ステートメントと INSERT ステートメントが役立ちます。

ペットのレコードを次のように記述できると仮定します。(MySQL では、日付の形式は 'YYYY-MM-DD' と想定されています。これはユーザーが通常使用する形式とは異なる場合があります。)

name	owner	species	sex	birth	death
Fluffy	Harold	cat	f	1993-02-04	
Claws	Gwen	cat	m	1994-03-17	
Buffy	Harold	dog	f	1989-05-13	
Fang	Benny	dog	m	1990-08-27	
Bowser	Diane	dog	m	1979-08-31	1995-07-29
Chirpy	Gwen	bird	f	1998-09-11	
Whistler	Gwen	bird		1997-12-09	
Slim	Benny	snake	m	1996-04-29	

空のテーブルから始めているため、データを移入する簡単な方法は、各ペットに対応する行を記述したテキストファイルを作成してから、1 つのステートメントでそのファイルの内容をテーブルにロードすることです。

たとえば、テキストファイル pet.txt を作成し、1 行に 1 レコードを記述します。値は、CREATE TABLE ステートメントに指定したカラムの順序に従い、タブで区切って指定します。性別が不明な場合やまだ生きているペットの死亡年月日など、不足している値には NULL 値を使用できます。テキストファイルでこれらを表現するには、\N (バックスラッシュと大文字の N) を使用します。たとえば、Whistler という鳥のレコードは次のようになります (値の間の空白は 1 つのタブ文字です)。

```
Whistler        Gwen    bird    \N      1997-12-09      \N
```

テキストファイル pet.txt を pet テーブルにロードするには、次のステートメントを使用します。

```
mysql> LOAD DATA LOCAL INFILE '/path/pet.txt' INTO TABLE pet;
```

このファイルを Windows で作成した場合、作成に使用した工ディタで \r\n が行ターミネータとして使用されているときは、代わりに次のステートメントを使用します。

```
mysql> LOAD DATA LOCAL INFILE '/path/pet.txt' INTO TABLE pet
    -> LINES TERMINATED BY '\r\n';
```

(OS X を実行している Apple マシンでは、LINES TERMINATED BY '\r' を使用するとよいでしょう。)

カラム値の区切り文字と行末マーカーは、必要に応じて LOAD DATA ステートメントで明示的に指定できますが、デフォルトではタブと改行です。ステートメントで pet.txt ファイルを正しく読み取るにはこれで十分です。

ステートメントが失敗する場合、使用している MySQL インストールではローカルファイル機能がデフォルトで有効になっていない可能性があります。これを変更する方法については、セクション6.1.6「LOAD DATA LOCAL のセキュリティーの問題」を参照してください。

新しいレコードを 1 つずつ追加する場合は、INSERT ステートメントが役立ちます。もっとも単純な形式では、CREATE TABLE ステートメントに指定したカラムの順序に従って、各カラムの値を入力します。Diane が、「Puffball」という名前の新しいハムスターを手に入れたとします。次のように、INSERT ステートメントを使用して新しいレコードを追加できます。

```
mysql> INSERT INTO pet
    -> VALUES ('Puffball','Diane','hamster','f','1999-03-30',NULL);
```

ここでは、文字値と日付値を、引用符付きの文字列で指定しています。また、INSERT では、不足している値を表す NULL を直接挿入することができます。LOAD DATA の場合のように \N を使用することはありません。

この例からわかるとおり、初期レコードをロードするために複数の INSERT ステートメントを使用すると、1 つの LOAD DATA ステートメントを使用する場合よりもかなり多くの入力が必要になります。

3.3.4 テーブルからの情報の取り出し

テーブルから情報を取り出すには、SELECT ステートメントを使用します。このステートメントの一般的な形式は次のとおりです。

```
SELECT what_to_select
FROM which_table
WHERE conditions_to_satisfy;
```

what_to_select は取得する対象です。これには、カラムのリストか、「すべてのカラム」を表す * を指定できます。which_table は、データを取り出すテーブルです。WHERE 句はオプションです。指定する場合は、取得対象となる行の条件を 1 つまたは複数 conditions_to_satisfy に指定します。

3.3.4.1 すべてのデータの選択

SELECT のもっとも単純な形式では、テーブルのすべての内容が取り出されます。

```
mysql> SELECT * FROM pet;
+----------+--------+---------+------+------------+------------+
| name     | owner  | species | sex  | birth      | death      |
+----------+--------+---------+------+------------+------------+
| Fluffy   | Harold | cat     | f    | 1993-02-04 | NULL       |
| Claws    | Gwen   | cat     | m    | 1994-03-17 | NULL       |
| Buffy    | Harold | dog     | f    | 1989-05-13 | NULL       |
| Fang     | Benny  | dog     | m    | 1990-08-27 | NULL       |
| Bowser   | Diane  | dog     | m    | 1979-08-31 | 1995-07-29 |
| Chirpy   | Gwen   | bird    | f    | 1998-09-11 | NULL       |
| Whistler | Gwen   | bird    | NULL | 1997-12-09 | NULL       |
| Slim     | Benny  | snake   | m    | 1996-04-29 | NULL       |
| Puffball | Diane  | hamster | f    | 1999-03-30 | NULL       |
+----------+--------+---------+------+------------+------------+
```

SELECT のこの形式は、たとえばテーブルに初期データセットをロードした直後など、テーブル全体を取り出すときに役立ちます。たとえば、Bowser の生年月日が正しくないようだと気付いたとします。血統書の原本を確認すると、正しい生年は 1979 年ではなく 1989 年であるとわかりました。

これを修正するには、少なくとも 2 つの方法があります。

- ファイル pet.txt を編集して間違いを修正したあと、DELETE と LOAD DATA を使用してテーブルを空にしてからリロードします。

```
mysql> DELETE FROM pet;
mysql> LOAD DATA LOCAL INFILE 'pet.txt' INTO TABLE pet;
```

ただし、この方法では、Puffball のレコードも再度入力する必要があります。

- UPDATE ステートメントを使用して、間違ったレコードだけを修正します。

```
mysql> UPDATE pet SET birth = '1989-08-31' WHERE name = 'Bowser';
```

UPDATE は該当するレコードだけを変更するため、テーブルをリロードする必要はありません。

3.3.4.2 特定の行の選択

前のセクションで説明したとおり、テーブル全体を取り出すことは簡単です。SELECT ステートメントから WHERE 句を省略するだけで済みます。ただし、特にテーブルが大きくなってくると、テーブル全体を取り出すことは通常ありません。通常は特定の質問に対する回答が必要であり、そのために、取得する情報に関していくつかの制約を指定します。ペットに関する質問に対して回答を得る選択クエリーをいくつか見てみましょう。

テーブルから特定の行だけを選択することができます。たとえば、Bowser の誕生日に加えた変更を確認するには、次のように Bowser のレコードを選択します。

```
mysql> SELECT * FROM pet WHERE name = 'Bowser';
+--------+-------+---------+------+------------+------------+
| name   | owner | species | sex  | birth      | death      |
+--------+-------+---------+------+------------+------------+
| Bowser | Diane | dog     | m    | 1989-08-31 | 1995-07-29 |
+--------+-------+---------+------+------------+------------+
```

出力から、1979 年ではなく 1989 年と正しく記録されていることが確認されます。

通常、文字列の比較では大文字と小文字が区別されないため、名前の指定には 'bowser' や 'BOWSER' などを使用できます。クエリーの結果は同じになります。

name だけでなく、任意のカラムに条件を指定できます。たとえば、1998 年以降に生まれたペットを調べるには、birth カラムを検査します。

```
mysql> SELECT * FROM pet WHERE birth >= '1998-1-1';
+----------+-------+---------+------+------------+-------+
| name     | owner | species | sex  | birth      | death |
+----------+-------+---------+------+------------+-------+
| Chirpy   | Gwen  | bird    | f    | 1998-09-11 | NULL  |
| Puffball | Diane | hamster | f    | 1999-03-30 | NULL  |
+----------+-------+---------+------+------------+-------+
```

条件を組み合わせて、たとえば雌の犬を特定することができます。

```
mysql> SELECT * FROM pet WHERE species = 'dog' AND sex = 'f';
+-------+--------+---------+------+------------+-------+
| name  | owner  | species | sex  | birth      | death |
+-------+--------+---------+------+------------+-------+
| Buffy | Harold | dog     | f    | 1989-05-13 | NULL  |
+-------+--------+---------+------+------------+-------+
```

前のクエリーでは、AND 論理演算子が使用されています。OR 演算子もあります。

```
mysql> SELECT * FROM pet WHERE species = 'snake' OR species = 'bird';
+----------+-------+---------+------+------------+-------+
| name     | owner | species | sex  | birth      | death |
+----------+-------+---------+------+------------+-------+
| Chirpy   | Gwen  | bird    | f    | 1998-09-11 | NULL  |
| Whistler | Gwen  | bird    | NULL | 1997-12-09 | NULL  |
| Slim     | Benny | snake   | m    | 1996-04-29 | NULL  |
+----------+-------+---------+------+------------+-------+
```

AND と OR は一緒に使用できますが、AND は OR よりも高い優先順位を持っています。両方の演算子を一緒に使用する場合は、括弧を使用して条件の組み合わせ方を明示的に示すことをお勧めします。

```
mysql> SELECT * FROM pet WHERE (species = 'cat' AND sex = 'm')
    -> OR (species = 'dog' AND sex = 'f');
+-------+--------+---------+------+------------+-------+
| name  | owner  | species | sex  | birth      | death |
+-------+--------+---------+------+------------+-------+
| Claws | Gwen   | cat     | m    | 1994-03-17 | NULL  |
| Buffy | Harold | dog     | f    | 1989-05-13 | NULL  |
```

```
+--------+--------+---------+------+------------+-------+
```

3.3.4.3 特定のカラムの選択

テーブルの行全体ではなく特定のカラムを取得するには、カラムの名前をカンマで区切って指定します。たとえば、ペットの生年月日を調べるには、name カラムと birth カラムを選択します。

```
mysql> SELECT name, birth FROM pet;
+----------+------------+
| name     | birth      |
+----------+------------+
| Fluffy   | 1993-02-04 |
| Claws    | 1994-03-17 |
| Buffy    | 1989-05-13 |
| Fang     | 1990-08-27 |
| Bowser   | 1989-08-31 |
| Chirpy   | 1998-09-11 |
| Whistler | 1997-12-09 |
| Slim     | 1996-04-29 |
| Puffball | 1999-03-30 |
+----------+------------+
```

ペットの所有者を調べるには、次のクエリーを使用します。

```
mysql> SELECT owner FROM pet;
+--------+
| owner  |
+--------+
| Harold |
| Gwen   |
| Harold |
| Benny  |
| Diane  |
| Gwen   |
| Gwen   |
| Benny  |
| Diane  |
+--------+
```

このクエリーは各レコードから owner カラムを取り出すだけなため、一部は複数回出現しています。出力を最小化するには、キーワード DISTINCT を追加して、一意の出力レコードをそれぞれ 1 回だけ取り出すようにします。

```
mysql> SELECT DISTINCT owner FROM pet;
+--------+
| owner  |
+--------+
| Benny  |
| Diane  |
| Gwen   |
| Harold |
+--------+
```

WHERE 句を使用して、行の選択とカラムの選択を組み合わせることができます。たとえば、犬と猫だけについて生年月日を調べるには、次のクエリーを使用します。

```
mysql> SELECT name, species, birth FROM pet
    -> WHERE species = 'dog' OR species = 'cat';
+--------+---------+------------+
| name   | species | birth      |
+--------+---------+------------+
| Fluffy | cat     | 1993-02-04 |
| Claws  | cat     | 1994-03-17 |
| Buffy  | dog     | 1989-05-13 |
| Fang   | dog     | 1990-08-27 |
| Bowser | dog     | 1989-08-31 |
+--------+---------+------------+
```

3.3.4.4 行のソート

前の例で、結果の行の表示には特定の順序がないことに気付いたでしょう。多くの場合、クエリーの出力は、行を何らかの意味のある順序でソートすると確認しやすくなります。結果をソートするには、ORDER BY 句を使用します。

次に、ペットの生年月日を日付でソートしたものを示します。

```
mysql> SELECT name, birth FROM pet ORDER BY birth;
+----------+------------+
| name     | birth      |
+----------+------------+
| Buffy    | 1989-05-13 |
| Bowser   | 1989-08-31 |
| Fang     | 1990-08-27 |
| Fluffy   | 1993-02-04 |
| Claws    | 1994-03-17 |
| Slim     | 1996-04-29 |
| Whistler | 1997-12-09 |
| Chirpy   | 1998-09-11 |
| Puffball | 1999-03-30 |
+----------+------------+
```

文字型のカラムでは、ソートはほかのすべての比較演算と同様に、通常大文字小文字の区別なしで実行されます。したがって、大文字と小文字の違いしかないカラムの場合、順序は未定義になります。カラムのソートで大文字と小文字を区別するには、BINARY を使用し、ORDER BY BINARY col_name のように指定します。

デフォルトのソート順序は昇順で、最小値が最初になります。逆順 (降順) でソートするには、ソートするカラムの名前に DESC キーワードを加えてください。

```
mysql> SELECT name, birth FROM pet ORDER BY birth DESC;
+----------+------------+
| name     | birth      |
+----------+------------+
| Puffball | 1999-03-30 |
| Chirpy   | 1998-09-11 |
| Whistler | 1997-12-09 |
| Slim     | 1996-04-29 |
| Claws    | 1994-03-17 |
| Fluffy   | 1993-02-04 |
| Fang     | 1990-08-27 |
| Bowser   | 1989-08-31 |
| Buffy    | 1989-05-13 |
+----------+------------+
```

複数のカラムでソートでき、ソートの方向はカラムごとに変えることができます。たとえば、ペットの種類で昇順にソートしてから、同じ種類の中では生年月日で降順に (若い順に) ソートするには、次のクエリーを使用します。

```
mysql> SELECT name, species, birth FROM pet
    -> ORDER BY species, birth DESC;
+----------+---------+------------+
| name     | species | birth      |
+----------+---------+------------+
| Chirpy   | bird    | 1998-09-11 |
| Whistler | bird    | 1997-12-09 |
| Claws    | cat     | 1994-03-17 |
| Fluffy   | cat     | 1993-02-04 |
| Fang     | dog     | 1990-08-27 |
| Bowser   | dog     | 1989-08-31 |
| Buffy    | dog     | 1989-05-13 |
| Puffball | hamster | 1999-03-30 |
| Slim     | snake   | 1996-04-29 |
+----------+---------+------------+
```

DESC キーワードはその直前のカラム名 (birth) だけに適用されます。species カラムのソート順序には影響を与えません。

3.3.4.5 日付の計算

MySQL には、年齢の計算や日付の一部の抽出など、日付の計算に使用できる関数がいくつか用意されています。

それぞれのペットが何歳なのかを判別するには、TIMESTAMPDIFF() 関数を使用します。引数は、結果を表現する単位、および差を求める 2 つの日付です。次のクエリーでは、各ペットの生年月日、現在の日付、および年齢が表示されます。出力の最後のカラムラベルに意味を持たせるために、エイリアス (age) が使用されています。

```
mysql> SELECT name, birth, CURDATE(),
    -> TIMESTAMPDIFF(YEAR,birth,CURDATE()) AS age
    -> FROM pet;
+----------+------------+------------+------+
| name     | birth      | CURDATE()  | age  |
+----------+------------+------------+------+
| Fluffy   | 1993-02-04 | 2003-08-19 |   10 |
| Claws    | 1994-03-17 | 2003-08-19 |    9 |
```

```
| Buffy    | 1989-05-13 | 2003-08-19 |   14 |
| Fang     | 1990-08-27 | 2003-08-19 |   12 |
| Bowser   | 1989-08-31 | 2003-08-19 |   13 |
| Chirpy   | 1998-09-11 | 2003-08-19 |    4 |
| Whistler | 1997-12-09 | 2003-08-19 |    5 |
| Slim     | 1996-04-29 | 2003-08-19 |    7 |
| Puffball | 1999-03-30 | 2003-08-19 |    4 |
+----------+------------+------------+------+
```

このクエリーは正しく動作しますが、結果の行を何らかの順序で表示すると確認しやすくなるでしょう。そのためには、ORDER BY name 句を追加することで、出力を名前でソートできます。

```
mysql> SELECT name, birth, CURDATE(),
    -> TIMESTAMPDIFF(YEAR,birth,CURDATE()) AS age
    -> FROM pet ORDER BY name;
+----------+------------+------------+------+
| name     | birth      | CURDATE()  | age  |
+----------+------------+------------+------+
| Bowser   | 1989-08-31 | 2003-08-19 |   13 |
| Buffy    | 1989-05-13 | 2003-08-19 |   14 |
| Chirpy   | 1998-09-11 | 2003-08-19 |    4 |
| Claws    | 1994-03-17 | 2003-08-19 |    9 |
| Fang     | 1990-08-27 | 2003-08-19 |   12 |
| Fluffy   | 1993-02-04 | 2003-08-19 |   10 |
| Puffball | 1999-03-30 | 2003-08-19 |    4 |
| Slim     | 1996-04-29 | 2003-08-19 |    7 |
| Whistler | 1997-12-09 | 2003-08-19 |    5 |
+----------+------------+------------+------+
```

出力を name ではなく age でソートするには、異なる ORDER BY 句を使用します。

```
mysql> SELECT name, birth, CURDATE(),
    -> TIMESTAMPDIFF(YEAR,birth,CURDATE()) AS age
    -> FROM pet ORDER BY age;
+----------+------------+------------+------+
| name     | birth      | CURDATE()  | age  |
+----------+------------+------------+------+
| Chirpy   | 1998-09-11 | 2003-08-19 |    4 |
| Puffball | 1999-03-30 | 2003-08-19 |    4 |
| Whistler | 1997-12-09 | 2003-08-19 |    5 |
| Slim     | 1996-04-29 | 2003-08-19 |    7 |
| Claws    | 1994-03-17 | 2003-08-19 |    9 |
| Fluffy   | 1993-02-04 | 2003-08-19 |   10 |
| Fang     | 1990-08-27 | 2003-08-19 |   12 |
| Bowser   | 1989-08-31 | 2003-08-19 |   13 |
| Buffy    | 1989-05-13 | 2003-08-19 |   14 |
+----------+------------+------------+------+
```

類似のクエリーを使用して、死んだペットの死亡時の年齢を求めることができます。どのペットなのかを判断するには、death 値が NULL かどうかを確認します。次に、NULL でない値について、death 値と birth 値の差を計算します。

```
mysql> SELECT name, birth, death,
    -> TIMESTAMPDIFF(YEAR,birth,death) AS age
    -> FROM pet WHERE death IS NOT NULL ORDER BY age;
+--------+------------+------------+------+
| name   | birth      | death      | age  |
+--------+------------+------------+------+
| Bowser | 1989-08-31 | 1995-07-29 |    5 |
+--------+------------+------------+------+
```

このクエリーでは、death <> NULL ではなく death IS NOT NULL を使用します。NULL は通常の比較演算子を使用して比較することができない特殊な値であるためです。これについてはあとで説明します。セクション 3.3.4.6「NULL 値の操作」を参照してください。

来月誕生日を迎えるペットを調べるにはどうしますか。このような計算の場合、年と日は無関係で、birth カラムの月の部分を抽出するだけで済みます。MySQL には、YEAR()、MONTH()、DAYOFMONTH() など、日付の一部を抽出する関数がいくつか用意されています。ここでは MONTH() 関数が適しています。動作の仕組みを確認するために、birth と MONTH(birth) の両方の値を表示する単純なクエリーを実行します。

```
mysql> SELECT name, birth, MONTH(birth) FROM pet;
+----------+------------+--------------+
| name     | birth      | MONTH(birth) |
+----------+------------+--------------+
| Fluffy   | 1993-02-04 |            2 |
| Claws    | 1994-03-17 |            3 |
| Buffy    | 1989-05-13 |            5 |
```

```
| Fang     | 1990-08-27 |           8 |
| Bowser   | 1989-08-31 |           8 |
| Chirpy   | 1998-09-11 |           9 |
| Whistler | 1997-12-09 |          12 |
| Slim     | 1996-04-29 |           4 |
| Puffball | 1999-03-30 |           3 |
+----------+------------+-------------+
```

来月誕生日を迎えるペットを見つけることも簡単です。今月は 4 月だとします。月の値は 4 であるため、5 月 (月 5) に生まれたペットは次のように探すことができます。

```
mysql> SELECT name, birth FROM pet WHERE MONTH(birth) = 5;
+-------+------------+
| name  | birth      |
+-------+------------+
| Buffy | 1989-05-13 |
+-------+------------+
```

今月が 12 月の場合は多少複雑になります。月の番号 (12) に単に 1 を加算して 13 月に生まれたペットを探すということはできません。そのような月は存在しないからです。代わりに、1 月 (月 1) に生まれたペットを探します。

現在が何月であっても機能するクエリーを記述すると、特定の月の番号を使用する必要がなくなります。DATE_ADD() を使用すると、所定の日付に時間間隔を加算できます。CURDATE() の値に 1 か月を加算してから、月の部分を MONTH() で抽出すると、誕生日を調べる月が得られます。

```
mysql> SELECT name, birth FROM pet
    -> WHERE MONTH(birth) = MONTH(DATE_ADD(CURDATE(),INTERVAL 1 MONTH));
```

現在の月の値が 12 の場合はモジュロ関数 (MOD) を適用して 0 に折り返してから、1 を加算する方法でも、同じタスクを達成できます。

```
mysql> SELECT name, birth FROM pet
    -> WHERE MONTH(birth) = MOD(MONTH(CURDATE()), 12) + 1;
```

MONTH() は 1 から 12 までの数値を返します。また、MOD(something,12) は 0 から 11 までの数値を返します。したがって、MOD() のあとで加算を行わないと、11 月から 1 月に進んでしまいます。

3.3.4.6 NULL 値の操作

NULL 値に慣れるまでは驚くかもしれません。概念的には、NULL は「存在しない不明な値」を意味し、ほかの値とは多少異なる方法で扱われます。

NULL を調べるために、次に示すように IS NULL および IS NOT NULL 演算子を使用します。

```
mysql> SELECT 1 IS NULL, 1 IS NOT NULL;
+-----------+---------------+
| 1 IS NULL | 1 IS NOT NULL |
+-----------+---------------+
|         0 |             1 |
+-----------+---------------+
```

=、<、または <> などの算術比較演算子を使用して NULL をテストすることはできません。これを自分で確認するために、次のクエリーを実行してみてください。

```
mysql> SELECT 1 = NULL, 1 <> NULL, 1 < NULL, 1 > NULL;
+----------+-----------+----------+----------+
| 1 = NULL | 1 <> NULL | 1 < NULL | 1 > NULL |
+----------+-----------+----------+----------+
|     NULL |      NULL |     NULL |     NULL |
+----------+-----------+----------+----------+
```

NULL に関する算術比較は、結果もすべて NULL になるため、このような比較から意味のある結果を得ることはできません。

MySQL では、0 や NULL は false を意味し、それ以外はすべて true を意味します。ブール演算のデフォルトの真理値は 1 です。

NULL がこのように特殊な方法で扱われているため、前のセクションで、どの動物がもう生きていいないのかを判断するために、death <> NULL ではなく death IS NOT NULL を使用することが必要だったのです。

GROUP BY では、2 つの NULL 値は等しいとみなされます。

ORDER BY を実行する場合、NULL 値は ORDER BY ... ASC では最初に表示され、ORDER BY ... DESC では最後に表示されます。

NULL を操作するときによくある間違いは、NOT NULL と定義されたカラムにはゼロや空の文字列は挿入できないと想定することです。これらは実際に値ですが、一方 NULL は「値がない」ことを意味します。このことは、次に示すように IS [NOT] NULL を使用してとても簡単にテストできます。

```
mysql> SELECT 0 IS NULL, 0 IS NOT NULL, '' IS NULL, '' IS NOT NULL;
+-----------+---------------+------------+----------------+
| 0 IS NULL | 0 IS NOT NULL | '' IS NULL | '' IS NOT NULL |
+-----------+---------------+------------+----------------+
|         0 |             1 |          0 |              1 |
+-----------+---------------+------------+----------------+
```

このように、ゼロや空の文字列は実際に NOT NULL であるため、NOT NULL カラムに挿入することができます。セクションB.5.5.3「NULL 値に関する問題」を参照してください。

3.3.4.7 パターンマッチング

MySQL では、標準の SQL パターンマッチングに加え、vi、grep、sed などの Unix ユーティリティーで使用されるものに似た拡張正規表現に基づくパターンマッチング形式が提供されています。

SQL のパターンマッチングを使用すると、「_」で任意の単一の文字、「%」で任意の数の文字 (ゼロ文字を含む) に一致させることができます。MySQL のデフォルトでは、SQL パターンでは大文字と小文字が区別されません。次にいくつかの例を示します。SQL パターンを使用するときには、= や <> は使用しません。代わりに、LIKE または NOT LIKE 比較演算子を使用します。

「b」で始まる名前を探すには:

```
mysql> SELECT * FROM pet WHERE name LIKE 'b%';
+--------+--------+---------+------+------------+------------+
| name   | owner  | species | sex  | birth      | death      |
+--------+--------+---------+------+------------+------------+
| Buffy  | Harold | dog     | f    | 1989-05-13 | NULL       |
| Bowser | Diane  | dog     | m    | 1989-08-31 | 1995-07-29 |
+--------+--------+---------+------+------------+------------+
```

「fy」で終わる名前を探すには:

```
mysql> SELECT * FROM pet WHERE name LIKE '%fy';
+--------+--------+---------+------+------------+-------+
| name   | owner  | species | sex  | birth      | death |
+--------+--------+---------+------+------------+-------+
| Fluffy | Harold | cat     | f    | 1993-02-04 | NULL  |
| Buffy  | Harold | dog     | f    | 1989-05-13 | NULL  |
+--------+--------+---------+------+------------+-------+
```

「w」を含む名前を探すには:

```
mysql> SELECT * FROM pet WHERE name LIKE '%w%';
+----------+-------+---------+------+------------+------------+
| name     | owner | species | sex  | birth      | death      |
+----------+-------+---------+------+------------+------------+
| Claws    | Gwen  | cat     | m    | 1994-03-17 | NULL       |
| Bowser   | Diane | dog     | m    | 1989-08-31 | 1995-07-29 |
| Whistler | Gwen  | bird    | NULL | 1997-12-09 | NULL       |
+----------+-------+---------+------+------------+------------+
```

ちょうど 5 文字の名前を探すには、「_」パターン文字の 5 つのインスタンスを使用します。

```
mysql> SELECT * FROM pet WHERE name LIKE '_____';
+-------+--------+---------+------+------------+-------+
| name  | owner  | species | sex  | birth      | death |
+-------+--------+---------+------+------------+-------+
| Claws | Gwen   | cat     | m    | 1994-03-17 | NULL  |
| Buffy | Harold | dog     | f    | 1989-05-13 | NULL  |
+-------+--------+---------+------+------------+-------+
```

MySQL で提供されているもう 1 種類のパターンマッチングは、拡張正規表現を使用します。このタイプのパターンについて一致をテストする場合は、REGEXP 演算子と NOT REGEXP 演算子 (またはシノニムである RLIKE と NOT RLIKE) を使用します。

次の表に、拡張正規表現の特徴の一部を示します。

- 「.」は任意の 1 文字に一致します。
- 文字クラス「[...]」は、括弧内のいずれかの文字に一致します。たとえば、「[abc]」は「a」、「b」、または「c」に一致します。文字の範囲を指定するには、ダッシュを使用します。「[a-z]」は任意の英字に一致し、「[0-9]」は任意の数字に一致します。
- 「*」は直前の文字の 0 個以上のインスタンスに一致します。たとえば、「x*」は任意の数の「x」文字に一致し、「[0-9]*」は任意の数の数字に一致し、「.*」は任意の数の任意の文字に一致します。
- REGEXP パターンマッチングは、テストする値のいずれかの部分にパターンが一致すれば成功です。(これとは異なり、LIKE パターンマッチングは、パターンが値全体に一致する場合のみ成功です。)
- テストする値の先頭または末尾にパターンが一致するように指定するには、パターンの先頭に「^」またはパターンの末尾に「$」を使用します。

拡張正規表現動作の仕組みを確認するために、前出の LIKE クエリーを REGEXP で書き直したものを次に示します。

「b」で始まる名前を探すには、「^」を使用して名前の先頭に一致するように指定します。

```
mysql> SELECT * FROM pet WHERE name REGEXP '^b';
+--------+--------+---------+------+------------+------------+
| name   | owner  | species | sex  | birth      | death      |
+--------+--------+---------+------+------------+------------+
| Buffy  | Harold | dog     | f    | 1989-05-13 | NULL       |
| Bowser | Diane  | dog     | m    | 1989-08-31 | 1995-07-29 |
+--------+--------+---------+------+------------+------------+
```

REGEXP 比較で大文字と小文字を区別するには、BINARY キーワードを使用して、文字列の 1 つをバイナリ文字列にします。次のクエリーは、名前の先頭にある小文字の「b」だけに一致します。

```
mysql> SELECT * FROM pet WHERE name REGEXP BINARY '^b';
```

「fy」で終わる名前を探すには、「$」を使用して名前の末尾に一致するように指定します。

```
mysql> SELECT * FROM pet WHERE name REGEXP 'fy$';
+--------+--------+---------+------+------------+-------+
| name   | owner  | species | sex  | birth      | death |
+--------+--------+---------+------+------------+-------+
| Fluffy | Harold | cat     | f    | 1993-02-04 | NULL  |
| Buffy  | Harold | dog     | f    | 1989-05-13 | NULL  |
+--------+--------+---------+------+------------+-------+
```

「w」を含む名前を探すには、次のクエリーを使用します。

```
mysql> SELECT * FROM pet WHERE name REGEXP 'w';
+----------+-------+---------+------+------------+------------+
| name     | owner | species | sex  | birth      | death      |
+----------+-------+---------+------+------------+------------+
| Claws    | Gwen  | cat     | m    | 1994-03-17 | NULL       |
| Bowser   | Diane | dog     | m    | 1989-08-31 | 1995-07-29 |
| Whistler | Gwen  | bird    | NULL | 1997-12-09 | NULL       |
+----------+-------+---------+------+------------+------------+
```

正規表現パターンは値のいずれかの部分に見つかれば一致するため、このクエリーでは、SQL パターンを使用する場合のようにパターンの両側にワイルドカードを付加してパターンを値全体と一致させる必要はありません。

ちょうど 5 文字の名前を探すには、「^」と「$」を使用して名前の先頭と末尾に一致するように指定し、間に「.」を 5 つ使用します。

```
mysql> SELECT * FROM pet WHERE name REGEXP '^.....$';
+-------+--------+---------+------+------------+-------+
| name  | owner  | species | sex  | birth      | death |
+-------+--------+---------+------+------------+-------+
| Claws | Gwen   | cat     | m    | 1994-03-17 | NULL  |
| Buffy | Harold | dog     | f    | 1989-05-13 | NULL  |
+-------+--------+---------+------+------------+-------+
```

このクエリーは、{n} (「n 回繰り返し」) 演算子を使用して記述することもできます。

```
mysql> SELECT * FROM pet WHERE name REGEXP '^.{5}$';
+-------+--------+---------+------+------------+-------+
| name  | owner  | species | sex  | birth      | death |
+-------+--------+---------+------+------------+-------+
```

```
| Claws | Gwen   | cat     | m    | 1994-03-17 | NULL  |
| Buffy | Harold | dog     | f    | 1989-05-13 | NULL  |
+-------+--------+---------+------+------------+-------+
```

セクション12.5.2「正規表現」で、正規表現の構文の詳細について説明しています。

3.3.4.8 行のカウント

データベースは、「テーブルの中に、特定のタイプのデータがどの程度の頻度で現れるか」という質問に答えるために使用されることがよくあります。たとえば、何匹ペットを飼っているのか、それぞれの所有者が何匹のペットを所有しているかを調べたり、または動物に対してさまざまな個体数調査を実施したりすることがあるでしょう。

ペットの総数をカウントすることは、「pet テーブルには何行あるか」という質問と同等です。このテーブルにはペットごとに 1 つのレコードが存在するからです。COUNT(*) は行数をカウントするため、ペットの数をカウントするクエリーは次のようになります。

```
mysql> SELECT COUNT(*) FROM pet;
+----------+
| COUNT(*) |
+----------+
|        9 |
+----------+
```

前に、ペットの所有者の名前を取得しました。COUNT() を使用して、各所有者のペットの数を調べることができます。

```
mysql> SELECT owner, COUNT(*) FROM pet GROUP BY owner;
+--------+----------+
| owner  | COUNT(*) |
+--------+----------+
| Benny  |        2 |
| Diane  |        2 |
| Gwen   |        3 |
| Harold |        2 |
+--------+----------+
```

このクエリーは GROUP BY を使用して各 owner のすべてのレコードをグループ化しています。COUNT() を GROUP BY とともに使用すると、さまざまなグループ化の下でデータの特徴を示すことができます。次の例では、ペットの個体数調査を実行するさまざまな方法を示します。

種ごとのペット数

```
mysql> SELECT species, COUNT(*) FROM pet GROUP BY species;
+---------+----------+
| species | COUNT(*) |
+---------+----------+
| bird    |        2 |
| cat     |        2 |
| dog     |        3 |
| hamster |        1 |
| snake   |        1 |
+---------+----------+
```

性別ごとのペット数

```
mysql> SELECT sex, COUNT(*) FROM pet GROUP BY sex;
+------+----------+
| sex  | COUNT(*) |
+------+----------+
| NULL |        1 |
| f    |        4 |
| m    |        4 |
+------+----------+
```

(この出力で、NULL は性別不明を示します。)

種と性別の組み合わせごとのペット数

```
mysql> SELECT species, sex, COUNT(*) FROM pet GROUP BY species, sex;
+---------+------+----------+
| species | sex  | COUNT(*) |
+---------+------+----------+
| bird    | NULL |        1 |
| bird    | f    |        1 |
```

```
| cat     | f    |       1 |
| cat     | m    |       1 |
| dog     | f    |       1 |
| dog     | m    |       2 |
| hamster | f    |       1 |
| snake   | m    |       1 |
+---------+------+----------+
```

COUNT() を使用するときにテーブル全体を取り出す必要はありません。たとえば、前のクエリーを犬と猫だけに対して実行する場合は、次のようになります。

```
mysql> SELECT species, sex, COUNT(*) FROM pet
    -> WHERE species = 'dog' OR species = 'cat'
    -> GROUP BY species, sex;
+---------+------+----------+
| species | sex  | COUNT(*) |
+---------+------+----------+
| cat     | f    |       1 |
| cat     | m    |       1 |
| dog     | f    |       1 |
| dog     | m    |       2 |
+---------+------+----------+
```

または、性別のわかっているペットについてのみ性別ごとのペット数を調べるには:

```
mysql> SELECT species, sex, COUNT(*) FROM pet
    -> WHERE sex IS NOT NULL
    -> GROUP BY species, sex;
+---------+------+----------+
| species | sex  | COUNT(*) |
+---------+------+----------+
| bird    | f    |       1 |
| cat     | f    |       1 |
| cat     | m    |       1 |
| dog     | f    |       1 |
| dog     | m    |       2 |
| hamster | f    |       1 |
| snake   | m    |       1 |
+---------+------+----------+
```

COUNT() 値に加え、選択するカラムを指定する場合は、それらのカラムを GROUP BY 句で指定する必要があります。そうでない場合は、次のようになります。

- ONLY_FULL_GROUP_BY SQL モードが有効である場合は、エラーが発生します。

```
mysql> SET sql_mode = 'ONLY_FULL_GROUP_BY';
Query OK, 0 rows affected (0.00 sec)

mysql> SELECT owner, COUNT(*) FROM pet;
ERROR 1140 (42000): Mixing of GROUP columns (MIN(),MAX(),COUNT()...)
with no GROUP columns is illegal if there is no GROUP BY clause
```

- ONLY_FULL_GROUP_BY が有効でない場合、このクエリーはすべての行を 1 つのグループとみなして処理されますが、指定した各カラムに選択される値は不確定です。サーバーによって任意の行の値が自由に選択されます。

```
mysql> SET sql_mode = '';
Query OK, 0 rows affected (0.00 sec)

mysql> SELECT owner, COUNT(*) FROM pet;
+--------+----------+
| owner  | COUNT(*) |
+--------+----------+
| Harold |        8 |
+--------+----------+
1 row in set (0.00 sec)
```

セクション12.19.3「MySQL での GROUP BY の処理」も参照してください。

3.3.4.9 複数のテーブルの使用

pet テーブルはペットの記録を保持します。獣医の診察や出産といったペットの生涯におけるイベントなど、ペットに関するほかの情報を記録するには、別のテーブルが必要です。このテーブルはどのようなものにしたらよいでしょうか。次の情報を含める必要があります。

- 各イベントがどのペットに関するものかを示すためのペット名。

- イベントがいつ発生したかを示す日付。
- イベントを説明するフィールド。
- イベントを分類できるようにする場合は、イベントタイプのフィールド。

これらを考慮すると、event テーブルの CREATE TABLE ステートメントは次のようになります。

```
mysql> CREATE TABLE event (name VARCHAR(20), date DATE,
    -> type VARCHAR(15), remark VARCHAR(255));
```

pet テーブルの場合と同様に、初期レコードをロードするもっとも簡単な方法として、次の情報を記述したタブ区切りのテキストファイルを作成します。

name	date	type	remark
Fluffy	1995-05-15	litter	4 kittens, 3 female, 1 male
Buffy	1993-06-23	litter	5 puppies, 2 female, 3 male
Buffy	1994-06-19	litter	3 puppies, 3 female
Chirpy	1999-03-21	vet	needed beak straightened
Slim	1997-08-03	vet	broken rib
Bowser	1991-10-12	kennel	
Fang	1991-10-12	kennel	
Fang	1998-08-28	birthday	Gave him a new chew toy
Claws	1998-03-17	birthday	Gave him a new flea collar
Whistler	1998-12-09	birthday	First birthday

次のようにレコードをロードします。

```
mysql> LOAD DATA LOCAL INFILE 'event.txt' INTO TABLE event;
```

pet テーブルで実行したクエリーから学んだことを基にすれば、原則は同じであるため event テーブルのレコードも取得できるはずです。ただし、event テーブルだけでは質問に回答できない場合はどのようなときでしょうか。

各ペットの出産時の年齢を調べるとします。前に、2 つの日付から年齢を計算する方法を学びました。ペットの出産日は event テーブルにありますが、その日付での年齢を計算するには生年月日が必要で、それは pet テーブルにあります。したがって、このクエリーには両方のテーブルが必要です。

```
mysql> SELECT pet.name,
    -> (YEAR(date)-YEAR(birth)) - (RIGHT(date,5)<RIGHT(birth,5)) AS age,
    -> remark
    -> FROM pet INNER JOIN event
    ->   ON pet.name = event.name
    -> WHERE event.type = 'litter';
+--------+------+-----------------------------+
| name   | age  | remark                      |
+--------+------+-----------------------------+
| Fluffy |    2 | 4 kittens, 3 female, 1 male |
| Buffy  |    4 | 5 puppies, 2 female, 3 male |
| Buffy  |    5 | 3 puppies, 3 female         |
+--------+------+-----------------------------+
```

このクエリーには注目するべき点がいくつかあります。

- このクエリーは両方のテーブルから情報を取り出す必要があるため、FROM 句で 2 つのテーブルを結合しています。
- 複数のテーブルの情報を組み合わせる (結合する) 場合、1 つのテーブルのレコードとほかのテーブルのレコードがどのように対応するかを指定する必要があります。両方のテーブルに name カラムがあるため、これは簡単です。このクエリーは、ON 句を使用して、2 つのテーブルのレコードを name 値に基づいて対応させています。

  このクエリーは INNER JOIN を使用してテーブルを結合しています。INNER JOIN では、ON 句で指定された条件を両方のテーブルが満たす場合にかぎって、結果にテーブルの行が許可されます。この例では、pet テーブルの name カラムと event テーブルの name カラムが一致する必要があると ON 句で指定しています。名前が一方のテーブルにあって他方にはない場合、ON 句の条件が満たされないため、その行は結果に表示されません。

- name カラムは両方のテーブルにあるため、このカラムを参照するときはどちらのテーブルのものかを明確に示す必要があります。そのためには、カラム名の前にテーブル名を付加します。

2 つの異なるテーブルでなくても結合は実行できます。テーブル内のレコードをその同じテーブル内のほかのレコードと比較する場合に、テーブルをそれ自体に結合すると役立つことがあります。たとえば、繁殖のつがいにするペットを選ぶ場合、pet テーブルをそれ自体に結合して、同種の雄と雌のつがい候補を生成できます。

```
mysql> SELECT p1.name, p1.sex, p2.name, p2.sex, p1.species
    -> FROM pet AS p1 INNER JOIN pet AS p2
    ->   ON p1.species = p2.species AND p1.sex = 'f' AND p2.sex = 'm';
+--------+------+--------+------+---------+
| name   | sex  | name   | sex  | species |
+--------+------+--------+------+---------+
| Fluffy | f    | Claws  | m    | cat     |
| Buffy  | f    | Fang   | m    | dog     |
| Buffy  | f    | Bowser | m    | dog     |
+--------+------+--------+------+---------+
```

このクエリーでは、テーブル名のエイリアスを指定してカラムを参照し、各カラムがテーブルのどちらのインスタンスに関連するかを必ず明確にしています。

3.4 データベースとテーブルに関する情報の取得

データベースやテーブルの名前を忘れた場合や、特定のテーブルの構造 (カラムの名前など) を忘れた場合はどうしますか。MySQL では、この問題に対処するために、サポートしているデータベースとテーブルについて情報を提供するステートメントがいくつか用意されています。

前出の SHOW DATABASES は、サーバーで管理されているデータベースのリストを表示します。現在どのデータベースが選択されているかを調べるには、DATABASE() 関数を使用します。

```
mysql> SELECT DATABASE();
+------------+
| DATABASE() |
+------------+
| menagerie  |
+------------+
```

まだどのデータベースも選択していない場合、結果は NULL になります。

テーブルの名前がはっきりしない場合などに、デフォルトのデータベースにどのようなテーブルが含まれているかを調べるには、次のコマンドを使用します。

```
mysql> SHOW TABLES;
+---------------------+
| Tables_in_menagerie |
+---------------------+
| event               |
| pet                 |
+---------------------+
```

このステートメントで生成される出力のカラム名は常に Tables_in_db_name になります。ここで、db_name はデータベースの名前です。詳細については、セクション13.7.5.38「SHOW TABLES 構文」を参照してください。

テーブルの構造を知りたい場合は、DESCRIBE ステートメントが役に立ちます。このステートメントはテーブルの各カラムの情報を表示します。

```
mysql> DESCRIBE pet;
+---------+-------------+------+-----+---------+-------+
| Field   | Type        | Null | Key | Default | Extra |
+---------+-------------+------+-----+---------+-------+
| name    | varchar(20) | YES  |     | NULL    |       |
| owner   | varchar(20) | YES  |     | NULL    |       |
| species | varchar(20) | YES  |     | NULL    |       |
| sex     | char(1)     | YES  |     | NULL    |       |
| birth   | date        | YES  |     | NULL    |       |
| death   | date        | YES  |     | NULL    |       |
+---------+-------------+------+-----+---------+-------+
```

Field はカラム名、Type はそのカラムのデータ型、NULL はカラムに NULL 値を含められるかどうか、Key はカラムにインデックスが設定されているかどうか、Default はカラムのデフォルト値を示します。Extra には、カラムに関する特殊な情報が表示されます。カラムが AUTO_INCREMENT オプションで作成された場合、値は空ではなく auto_increment になります。

DESC は DESCRIBE の省略形式です。詳細については、セクション13.8.1「DESCRIBE 構文」を参照してください。

既存のテーブルを作成するために必要な CREATE TABLE ステートメントを、SHOW CREATE TABLE ステートメントを使用して取得できます。セクション13.7.5.12「SHOW CREATE TABLE 構文」を参照してください。

テーブルにインデックスが設定されている場合は、SHOW INDEX FROM tbl_name でその情報を表示できます。このステートメントの詳細については、セクション13.7.5.23「SHOW INDEX 構文」を参照してください。

3.5 バッチモードでの MySQL の使用

前のセクションでは、mysql をインタラクティブに使用してクエリーを入力し、結果を表示しました。mysql をバッチモードで実行することもできます。そのためには、実行するコマンドをファイルに記述し、そのファイルから入力を読み取るように mysql に指示します。

```
shell> mysql < batch-file
```

mysql を Windows で実行する場合に、ファイル内の一部の特殊文字によって問題が発生するときは、次のように実行できます。

```
C:\> mysql -e "source batch-file"
```

コマンド行で接続パラメータを指定する必要がある場合、コマンドは次のようになります。

```
shell> mysql -h host -u user -p < batch-file
Enter password: ********
```

この方法で mysql を使用する場合は、スクリプトファイルを作成してから、そのスクリプトを実行することになります。

スクリプト内の一部のステートメントでエラーが発生してもスクリプトを続行する場合は、--force コマンド行オプションを使用します。

なぜスクリプトを使用するのでしょうか。いくつかの理由を次に示します。

- クエリーを繰り返し実行する場合 (毎日、毎週など)、スクリプトにすると、実行するたびに入力し直す必要がなくなります。
- 既存のクエリーのスクリプトファイルをコピーして編集することによって、類似の新しいクエリーを作成できます。
- バッチモードはクエリーの開発時にも役立ちます。特に、複数行にわたるコマンドまたは複数ステートメントによるコマンドシーケンスを使用する場合に便利です。間違いがあっても、すべてを入力し直す必要はありません。スクリプトを編集して間違いを修正してから、mysql で再度実行するだけで済みます。
- 多量の出力を生成するクエリーの場合、画面でスクロールアップする出力を見る代わりに、pager を介して出力できます。

  ```
  shell> mysql < batch-file | more
  ```

- あとで処理できるように出力をファイルに取り込むことができます。

  ```
  shell> mysql < batch-file > mysql.out
  ```

- スクリプトを配布すると、ほかのユーザーも同じコマンドを実行できます。
- cron ジョブからクエリーを実行する場合など、インタラクティブには使用できないことがあります。この場合はバッチモードを使用する必要があります。

mysql をバッチモードで実行したときのデフォルトの出力形式は、インタラクティブに使用した場合とは異なり、より簡潔になります。たとえば、mysql をインタラクティブに実行すると、SELECT DISTINCT species FROM pet の出力は次のようになります。

```
+---------+
| species |
+---------+
| bird    |
| cat     |
| dog     |
| hamster |
```

```
| snake   |
+---------+
```

これに対し、バッチモードの出力は次のようになります。

```
species
bird
cat
dog
hamster
snake
```

バッチモードで、インタラクティブ出力形式のデータを取得するには、mysql -t を使用します。実行したコマンドを出力にエコーするには、mysql -vvv を使用します。

source コマンドまたは \. コマンドを使用すると、mysql プロンプトからでもスクリプトを使用できます。

```
mysql> source filename;
mysql> \. filename
```

詳細については、セクション4.5.1.5「テキストファイルから SQL ステートメントを実行する」を参照してください。

3.6 一般的なクエリーの例

ここでは、MySQL に関する一般的な問題を解決する方法の例を示します。

一部の例では、テーブル shop を使用します。このテーブルには、業者 (ディーラー) の物品 (品番) ごとの価格が格納されます。各業者は物品ごとに 1 つの定価を付けていると仮定すると、(article, dealer) がレコードの主キーになります。

コマンド行ツール mysql を起動し、データベースを選択します。

```
shell> mysql your-database-name
```

(ほとんどの MySQL インストールで、test というデータベースを使用できます)。

次のステートメントを実行すると、テーブルを作成し、データを移入できます。

```
CREATE TABLE shop (
    article INT(4) UNSIGNED ZEROFILL DEFAULT '0000' NOT NULL,
    dealer  CHAR(20)               DEFAULT ''     NOT NULL,
    price   DOUBLE(16,2)           DEFAULT '0.00' NOT NULL,
    PRIMARY KEY(article, dealer));
INSERT INTO shop VALUES
    (1,'A',3.45),(1,'B',3.99),(2,'A',10.99),(3,'B',1.45),
    (3,'C',1.69),(3,'D',1.25),(4,'D',19.95);
```

これらのステートメントを発行したあと、テーブルには次の内容が格納されています。

```
SELECT * FROM shop;

+---------+--------+-------+
| article | dealer | price |
+---------+--------+-------+
|    0001 | A      |  3.45 |
|    0001 | B      |  3.99 |
|    0002 | A      | 10.99 |
|    0003 | B      |  1.45 |
|    0003 | C      |  1.69 |
|    0003 | D      |  1.25 |
|    0004 | D      | 19.95 |
+---------+--------+-------+
```

3.6.1 カラムの最大値

「もっとも大きい品番は?」

```
SELECT MAX(article) AS article FROM shop;

+---------+
| article |
+---------+
```

```
|      4 |
+---------+
```

3.6.2 特定のカラムの最大値が格納されている行

タスク: もっとも高価な物品の品番、業者、および価格を調べます。

これはサブクエリーを使用して簡単に実行できます。

```
SELECT article, dealer, price
FROM   shop
WHERE  price=(SELECT MAX(price) FROM shop);

+---------+--------+-------+
| article | dealer | price |
+---------+--------+-------+
|    0004 | D      | 19.95 |
+---------+--------+-------+
```

ほかにも、LEFT JOIN を使用する方法や、すべての行を価格の降順でソートしてから MySQL 固有の LIMIT 句を使用して最初の行だけを取得する方法もあります。

```
SELECT s1.article, s1.dealer, s1.price
FROM shop s1
LEFT JOIN shop s2 ON s1.price < s2.price
WHERE s2.article IS NULL;

SELECT article, dealer, price
FROM shop
ORDER BY price DESC
LIMIT 1;
```

> **注記**
>
> 最高価格のものが複数あり、価格が 19.95 の場合、LIMIT を使用した方法では、その中の 1 つしか取得できません。

3.6.3 グループごとのカラムの最大値

タスク: 物品ごとの最高値を調べます。

```
SELECT article, MAX(price) AS price
FROM   shop
GROUP BY article;

+---------+-------+
| article | price |
+---------+-------+
|    0001 |  3.99 |
|    0002 | 10.99 |
|    0003 |  1.69 |
|    0004 | 19.95 |
+---------+-------+
```

3.6.4 特定のカラムのグループごとの最大値が格納されている行

タスク: 物品ごとに最高値を付けている業者 (複数可) を調べます。

この問題は、次のようなサブクエリーを使用して解決できます。

```
SELECT article, dealer, price
FROM   shop s1
WHERE  price=(SELECT MAX(s2.price)
              FROM shop s2
              WHERE s1.article = s2.article);

+---------+--------+-------+
| article | dealer | price |
+---------+--------+-------+
|    0001 | B      |  3.99 |
|    0002 | A      | 10.99 |
|    0003 | C      |  1.69 |
|    0004 | D      | 19.95 |
+---------+--------+-------+
```

この例では相関サブクエリーを使用していますが、これは十分でない場合があります (セクション13.2.10.7「相関サブクエリー」を参照してください)。この問題を解決する別の方法は、FROM 句または LEFT JOIN で非相関サブクエリーを使用することです。

非相関サブクエリー

```
SELECT s1.article, dealer, s1.price
FROM shop s1
JOIN (
  SELECT article, MAX(price) AS price
  FROM shop
  GROUP BY article) AS s2
  ON s1.article = s2.article AND s1.price = s2.price;
```

LEFT JOIN:

```
SELECT s1.article, s1.dealer, s1.price
FROM shop s1
LEFT JOIN shop s2 ON s1.article = s2.article AND s1.price < s2.price
WHERE s2.article IS NULL;
```

LEFT JOIN の動作は、s1.price が最大値を取るときに、それより大きい値の s2.price は存在せず、s2 行の値は NULL になることに基づいています。セクション13.2.9.2「JOIN 構文」を参照してください。

3.6.5 ユーザー定義の変数の使用

MySQL ユーザー変数を使用すると、クライアント側で一時変数を使用せずに結果を記憶することができます。(セクション9.4「ユーザー定義変数」を参照してください。)

たとえば、最高値および最安値が付けられている物品を取得するには、次を実行します。

```
mysql> SELECT @min_price:=MIN(price),@max_price:=MAX(price) FROM shop;
mysql> SELECT * FROM shop WHERE price=@min_price OR price=@max_price;
+---------+--------+-------+
| article | dealer | price |
+---------+--------+-------+
|    0003 | D      |  1.25 |
|    0004 | D      | 19.95 |
+---------+--------+-------+
```

> **注記**
>
> また、テーブルやカラムといったデータベースオブジェクトの名前をユーザー変数に格納してから、この変数を SQL ステートメントで使用することもできます。ただし、これにはプリペアドステートメントを使用する必要があります。詳細は、セクション13.5「準備済みステートメントのための SQL 構文」を参照してください。

3.6.6 外部キーの使用

MySQL では、InnoDB テーブルで外部キー制約の確認をサポートしています。第14章「InnoDB ストレージエンジン」、セクション1.7.2.4「外部キーの違い」を参照してください。

2 つのテーブルを結合するだけの場合は、外部キー制約は必要ありません。InnoDB 以外のストレージエンジンの場合、カラムを定義するときに REFERENCES tbl_name(col_name) 句を使用できます。これは実際の効果はありませんが、現在定義しようとしているカラムが別のテーブルのカラムを参照する予定であるという自分のメモまたはコメントとして役立ちます。この構文を使用するときは、次の点を理解しておくことが非常に重要です。

- MySQL は、col_name が実際に tbl_name に存在するか (また、その tbl_name 自体が存在するか) を確認するためのどのような CHECK も実行しません。
- MySQL は、tbl_name に対してどのようなアクションも実行しません。たとえば、定義しようとしているテーブルの行に実行されたアクションに対応して行を削除することなどはありません。つまり、この構文にはどのような ON DELETE 動作や ON UPDATE 動作もありません。(REFERENCES 句の一部として ON DELETE 句や ON UPDATE 句を記述することはできますが、これらも無視されます。)
- この構文はカラムを作成します。どのようなインデックスやキーも作成しません。

このように作成したカラムを、次のように結合カラムとして使用できます。

```
CREATE TABLE person (
```

```
    id SMALLINT UNSIGNED NOT NULL AUTO_INCREMENT,
    name CHAR(60) NOT NULL,
    PRIMARY KEY (id)
);

CREATE TABLE shirt (
    id SMALLINT UNSIGNED NOT NULL AUTO_INCREMENT,
    style ENUM('t-shirt', 'polo', 'dress') NOT NULL,
    color ENUM('red', 'blue', 'orange', 'white', 'black') NOT NULL,
    owner SMALLINT UNSIGNED NOT NULL REFERENCES person(id),
    PRIMARY KEY (id)
);

INSERT INTO person VALUES (NULL, 'Antonio Paz');

SELECT @last := LAST_INSERT_ID();

INSERT INTO shirt VALUES
(NULL, 'polo', 'blue', @last),
(NULL, 'dress', 'white', @last),
(NULL, 't-shirt', 'blue', @last);

INSERT INTO person VALUES (NULL, 'Lilliana Angelovska');

SELECT @last := LAST_INSERT_ID();

INSERT INTO shirt VALUES
(NULL, 'dress', 'orange', @last),
(NULL, 'polo', 'red', @last),
(NULL, 'dress', 'blue', @last),
(NULL, 't-shirt', 'white', @last);

SELECT * FROM person;
+----+--------------------+
| id | name               |
+----+--------------------+
|  1 | Antonio Paz        |
|  2 | Lilliana Angelovska |
+----+--------------------+

SELECT * FROM shirt;
+----+---------+--------+-------+
| id | style   | color  | owner |
+----+---------+--------+-------+
|  1 | polo    | blue   |     1 |
|  2 | dress   | white  |     1 |
|  3 | t-shirt | blue   |     1 |
|  4 | dress   | orange |     2 |
|  5 | polo    | red    |     2 |
|  6 | dress   | blue   |     2 |
|  7 | t-shirt | white  |     2 |
+----+---------+--------+-------+


SELECT s.* FROM person p INNER JOIN shirt s
  ON s.owner = p.id
 WHERE p.name LIKE 'Lilliana%'
  AND s.color <> 'white';

+----+-------+--------+-------+
| id | style | color  | owner |
+----+-------+--------+-------+
|  4 | dress | orange |     2 |
|  5 | polo  | red    |     2 |
|  6 | dress | blue   |     2 |
+----+-------+--------+-------+
```

この方法で使用する場合、REFERENCES 句は SHOW CREATE TABLE や DESCRIBE の出力に表示されません。

```
SHOW CREATE TABLE shirt\G
*************************** 1. row ***************************
Table: shirt
Create Table: CREATE TABLE `shirt` (
`id` smallint(5) unsigned NOT NULL auto_increment,
`style` enum('t-shirt','polo','dress') NOT NULL,
`color` enum('red','blue','orange','white','black') NOT NULL,
`owner` smallint(5) unsigned NOT NULL,
PRIMARY KEY  (`id`)
) ENGINE=MyISAM DEFAULT CHARSET=latin1
```

REFERENCES をこのようにカラム定義のコメントまたは「リマインダ」として使用する方法は、MyISAM テーブルで機能します。

3.6.7 2 つのキーを使用した検索

1 つのキーを使用した OR の処理は、AND の処理と同様にかなり最適化されています。

注意が必要なのは、OR で結合された 2 つの異なるキーを使用して検索する場合です。

```
SELECT field1_index, field2_index FROM test_table
WHERE field1_index = '1' OR  field2_index = '1'
```

この場合は最適化されています。セクション8.2.1.4「インデックスマージの最適化」を参照してください。

2 つの異なる SELECT ステートメントの出力を結合する UNION を使用することでも、この問題を効率的に解決できます。セクション13.2.9.4「UNION 構文」を参照してください。

各 SELECT は 1 つのキーだけを検索するため、最適化できます。

```
SELECT field1_index, field2_index
   FROM test_table WHERE field1_index = '1'
UNION
SELECT field1_index, field2_index
   FROM test_table WHERE field2_index = '1';
```

3.6.8 日ごとの訪問数の計算

ビットグループ関数を使用して、あるユーザーが Web ページを訪問した月ごとの日数を計算する方法の例を次に示します。

```
CREATE TABLE t1 (year YEAR(4), month INT(2) UNSIGNED ZEROFILL,
           day INT(2) UNSIGNED ZEROFILL);
INSERT INTO t1 VALUES(2000,1,1),(2000,1,20),(2000,1,30),(2000,2,2),
          (2000,2,23),(2000,2,23);
```

このテーブルには、ユーザーがページを訪問した日付を表す年月日の値が格納されています。月ごとの訪問日数を取得するには、次のクエリーを実行します。

```
SELECT year,month,BIT_COUNT(BIT_OR(1<<day)) AS days FROM t1
      GROUP BY year,month;
```

次の結果が表示されます。

```
+------+-------+------+
| year | month | days |
+------+-------+------+
| 2000 |    01 |    3 |
| 2000 |    02 |    2 |
+------+-------+------+
```

このクエリーでは、年と月の組み合わせに対して異なる日付が何回テーブルに出現するかを、自動的に重複エントリを除去することによって計算しています。

3.6.9 AUTO_INCREMENT の使用

AUTO_INCREMENT 属性を使用すると、新しい行に一意の識別子を生成できます。

```
CREATE TABLE animals (
     id MEDIUMINT NOT NULL AUTO_INCREMENT,
     name CHAR(30) NOT NULL,
     PRIMARY KEY (id)
);

INSERT INTO animals (name) VALUES
    ('dog'),('cat'),('penguin'),
    ('lax'),('whale'),('ostrich');

SELECT * FROM animals;
```

次の結果が表示されます。

```
+----+---------+
| id | name    |
```

```
+----+---------+
| 1 | dog     |
| 2 | cat     |
| 3 | penguin |
| 4 | lax     |
| 5 | whale   |
| 6 | ostrich |
+----+---------+
```

AUTO_INCREMENT カラムには値が指定されなかったため、MySQL が自動的にシーケンス番号を割り当てました。カラムに明示的に 0 を割り当ててシーケンス番号を生成することもできます。カラムが NOT NULL と宣言されている場合は、NULL 割り当ててシーケンス番号を生成することもできます。

SQL 関数 LAST_INSERT_ID() または C API 関数 mysql_insert_id() を使用すると、最後に生成した AUTO_INCREMENT の値を取得できます。これらの関数は接続に固有の関数であるため、別の接続が同様に挿入を実行していても、戻り値は影響を受けません。

AUTO_INCREMENT カラムには、必要な最大のシーケンス値を保持するのに十分な大きさを持つ最小の整数データ型を使用します。カラムがデータ型の上限値に到達すると、次にシーケンス番号を生成しようとしたときには失敗します。可能であれば、より広い範囲を可能にするために UNSIGNED 属性を使用します。たとえば、TINYINT を使用する場合、許可される最大のシーケンス番号は 127 です。TINYINT UNSIGNED の場合は最大値は 255 です。すべての整数型の範囲は、セクション11.2.1「整数型 (真数値) - INTEGER、INT、SMALLINT、TINYINT、MEDIUMINT、BIGINT」を参照してください。

> **注記**
>
> 複数行を同時に挿入する場合、LAST_INSERT_ID() と mysql_insert_id() は、実際には最初に挿入した行の AUTO_INCREMENT キーを返します。これにより、レプリケーションセットアップで複数行の挿入を別のサーバーで正しく再現できます。

1 以外の AUTO_INCREMENT 値で開始するには、次のように、その値を CREATE TABLE または ALTER TABLE でセットします。

```
mysql> ALTER TABLE tbl AUTO_INCREMENT = 100;
```

InnoDB の注意

InnoDB テーブルでは、一連の INSERT ステートメントの途中で自動インクリメント値を含むカラムを修正する場合は注意が必要です。たとえば、UPDATE ステートメントを使用して、自動インクリメントカラムに新しくより大きい値を入れると、後続の INSERT は「重複」エラーになる場合があります。DELETE を実行したあとでさらに INSERT ステートメントが続く場合、またはトランザクションを COMMIT したが UPDATE ステートメントのあとではない場合に、自動インクリメント値がすでに存在するかどうかのテストが行われます。

MyISAM の注意

- MyISAM テーブルには、マルチカラムインデックス内のセカンダリカラムに AUTO_INCREMENT を指定することができます。この場合、AUTO_INCREMENT カラムに生成される値は、MAX(auto_increment_column) + 1 WHERE prefix=given-prefix として計算されます。これは、データを順序付きのグループに分割する場合に便利です。

```
CREATE TABLE animals (
    grp ENUM('fish','mammal','bird') NOT NULL,
    id MEDIUMINT NOT NULL AUTO_INCREMENT,
    name CHAR(30) NOT NULL,
    PRIMARY KEY (grp,id)
) ENGINE=MyISAM;

INSERT INTO animals (grp,name) VALUES
    ('mammal','dog'),('mammal','cat'),
    ('bird','penguin'),('fish','lax'),('mammal','whale'),
    ('bird','ostrich');

SELECT * FROM animals ORDER BY grp,id;
```

次の結果が表示されます。

```
+--------+----+---------+
| grp    | id | name    |
+--------+----+---------+
| fish   |  1 | lax     |
| mammal |  1 | dog     |
| mammal |  2 | cat     |
```

```
| mammal |  3 | whale   |
| bird   |  1 | penguin |
| bird   |  2 | ostrich |
+--------+----+---------+
```

この場合 (AUTO_INCREMENT カラムがマルチカラムインデックスの一部として使用されている場合)、グループ内で最大の AUTO_INCREMENT 値を持つ行を削除すると、そのグループで同じ AUTO_INCREMENT 値が再使用されることになります。これは、通常は AUTO_INCREMENT 値が再使用されることのない MyISAM テーブルの場合にも発生します。

- AUTO_INCREMENT カラムが複合インデックスの一部である場合、MySQL は AUTO_INCREMENT カラムで始まるインデックスを使用してシーケンス値を生成します (ある場合)。たとえば、animals テーブルにインデックス PRIMARY KEY (grp, id) と INDEX (id) が含まれている場合、MySQL はシーケンス値の生成で PRIMARY KEY を無視します。その結果、テーブルには grp 値ごとに 1 つのシーケンスではなく、単一のシーケンスが含まれることになります。

参照情報

AUTO_INCREMENT に関する詳細の参照先を次に示します。

- カラムに AUTO_INCREMENT 属性を割り当てる方法: セクション13.1.17「CREATE TABLE 構文」、およびセクション13.1.7「ALTER TABLE 構文」。
- AUTO_INCREMENT の、NO_AUTO_VALUE_ON_ZERO SQL モードによる動作の違い: セクション5.1.7「サーバー SQL モード」。
- LAST_INSERT_ID() 関数を使用して最新の AUTO_INCREMENT 値を見つける方法: セクション12.14「情報関数」。
- 使用する AUTO_INCREMENT 値の設定: セクション5.1.4「サーバーシステム変数」。
- AUTO_INCREMENT とレプリケーション: セクション17.4.1.1「レプリケーションと AUTO_INCREMENT」。
- レプリケーションに使用できる AUTO_INCREMENT 関連のサーバーシステム変数 (auto_increment_increment と auto_increment_offset): セクション5.1.4「サーバーシステム変数」。

3.7 Apache での MySQL の使用

MySQL データベースを使用してユーザーを認証し、ログファイルを MySQL のテーブルに書き込むプログラムがあります。

Apache 構成ファイルに次を追加することで、MySQL に簡単に読み込めるように Apache のロギング形式を変更することができます。

```
LogFormat \
    "\"%h\",%{%Y%m%d%H%M%S}t,%>s,\"%b\",\"%{Content-Type}o\",  \
    \"%U\",\"%{Referer}i\",\"%{User-Agent}i\""
```

この形式のログファイルを MySQL にロードするには、次のようなステートメントを使用します。

```
LOAD DATA INFILE '/local/access_log' INTO TABLE tbl_name
FIELDS TERMINATED BY ',' OPTIONALLY ENCLOSED BY '"' ESCAPED BY '\\'
```

LogFormat 行がログファイルに書き込むデータに対応して、指定するテーブルのカラムを作成する必要があります。

第 4 章 MySQL プログラム

目次

4.1 MySQL プログラムの概要 226
4.2 MySQL プログラムの使用 230
4.2.1 MySQL プログラムの起動 230
4.2.2 MySQL サーバーへの接続 230
4.2.3 プログラムオプションの指定 233
4.2.4 コマンド行でのオプションの使用 234
4.2.5 プログラムオプション修飾子 235
4.2.6 オプションファイルの使用 236
4.2.7 オプションファイルの処理に影響するコマンド行オプション 240
4.2.8 プログラム変数の設定へのオプションの使用 241
4.2.9 オプションのデフォルト、値を想定するオプション、および = 記号 242
4.2.10 環境変数の設定 244
4.3 MySQL サーバーとサーバー起動プログラム 245
4.3.1 mysqld — MySQL サーバー 245
4.3.2 mysqld_safe — MySQL サーバー起動スクリプト 246
4.3.3 mysql.server — MySQL サーバー起動スクリプト 250
4.3.4 mysqld_multi — 複数の MySQL サーバーの管理 250
4.4 MySQL インストール関連のプログラム 254
4.4.1 comp_err — MySQL エラーメッセージファイルのコンパイル 254
4.4.2 mysqlbug — バグレポートの生成 255
4.4.3 mysql_install_db — MySQL データディレクトリの初期化 255
4.4.4 mysql_plugin — MySQL サーバープラグインの構成 258
4.4.5 mysql_secure_installation — MySQL インストールのセキュリティー改善 260
4.4.6 mysql_tzinfo_to_sql — タイムゾーンテーブルのロード 260
4.4.7 mysql_upgrade — MySQL テーブルのチェックとアップグレード 261
4.5 MySQL クライアントプログラム 266
4.5.1 mysql — MySQL コマンド行ツール 266
4.5.2 mysqladmin — MySQL サーバーの管理を行うクライアント 286
4.5.3 mysqlcheck — テーブル保守プログラム 293
4.5.4 mysqldump — データベースバックアッププログラム 300
4.5.5 mysqlimport — データインポートプログラム 316
4.5.6 mysqlshow — データベース、テーブル、およびカラム情報の表示 321
4.5.7 mysqlslap — 負荷エミュレーションクライアント 325
4.6 MySQL 管理プログラムおよびユーティリティープログラム 332
4.6.1 innochecksum — オフライン InnoDB ファイルチェックサムユーティリティー 332
4.6.2 myisam_ftdump — 全文インデックス情報の表示 333
4.6.3 myisamchk — MyISAM テーブルメンテナンスユーティリティー 334
4.6.4 myisamlog — MyISAM ログファイルの内容の表示 349
4.6.5 myisampack — 圧縮された読み取り専用の MyISAM テーブルの生成 350
4.6.6 mysql_config_editor — MySQL 構成ユーティリティー 355
4.6.7 mysqlaccess — アクセス権限をチェックするクライアント 360
4.6.8 mysqlbinlog — バイナリログファイルを処理するためのユーティリティー 362
4.6.9 mysqldumpslow — スロークエリーログファイルの要約 379
4.6.10 mysqlhotcopy — データベースバックアッププログラム 381
4.6.11 mysql_convert_table_format — 指定されたストレージエンジンを使用するためのテーブルの変換 384
4.6.12 mysql_find_rows — ファイルからの SQL ステートメントの抽出 384
4.6.13 mysql_fix_extensions — テーブルファイル名の拡張子の正規化 385
4.6.14 mysql_setpermission — 付与テーブルに許可をインタラクティブに設定 385
4.6.15 mysql_waitpid — プロセスを強制終了して終了を待機 386
4.6.16 mysql_zap — パターンに一致するプロセスを強制終了 387
4.7 MySQL プログラム開発ユーティリティー 387
4.7.1 msql2mysql — mSQL プログラムを MySQL で使用するために変換 387
4.7.2 mysql_config — クライアントのコンパイル用オプションの表示 388
4.7.3 my_print_defaults — オプションファイルからのオプションの表示 389
4.7.4 resolve_stack_dump — 数値スタックトレースダンプをシンボルに解決 390
4.8 その他のプログラム 390

4.8.1 perror — エラーコードの説明 390
4.8.2 replace — 文字列置換ユーティリティー 391
4.8.3 resolveip — ホスト名と IP アドレスの解決 392

この章では、Oracle Corporation が提供する MySQL コマンド行プログラムの簡単な概要を説明します。また、これらのプログラムを実行するときに、オプションを指定するための一般的な構文についても説明します。ほとんどのプログラムには、その動作に固有のオプションがありますが、オプションの構文はそれらすべてに関して同様です。最後に、この章では個々のプログラムが認識するオプションを含め、詳細を説明します。

4.1 MySQL プログラムの概要

MySQL インストールには多くのさまざまなプログラムがあります。このセクションでは、それらの概要を簡単に説明します。そのあとのセクションで、それぞれをより詳細に説明しますが、MySQL Cluster プログラムは例外です。各プログラムの説明では、呼び出し構文とサポートされるオプションを示します。セクション 18.4「MySQL Cluster プログラム」では、MySQL Cluster に固有のプログラムについて説明します。

ほとんどの MySQL 配布には、これらのプログラムが (プラットフォーム固有のプログラムを除き) すべて含まれます。(たとえば、サーバー起動スクリプトは Windows では使用されません。)例外は、RPM 配布はより専門化されているということです。サーバー用に 1 つの RPM があり、クライアントプログラム用にもう 1 つ、などです。もし 1 つまたは複数のプログラムが欠けているようであれば、第2章「MySQL のインストールと更新」を参照して、配布のタイプとその内容を確認してください。配布がすべてのプログラムを含んではおらず、追加のパッケージをインストールする必要がある場合もあります。

各 MySQL プログラムは多くのさまざまなオプションを取ります。ほとんどのプログラムには --help オプションがあり、プログラムのさまざまなオプションの説明を取得するために使用できます。たとえば、mysql --help を試してみてください。

MySQL プログラムのデフォルトのオプション値は、コマンド行のオプションまたはオプションファイルを指定して、オーバーライドできます。プログラムの起動と、プログラムオプションの指定に関する一般的な情報については、セクション4.2「MySQL プログラムの使用」を参照してください。

MySQL Server mysqld は、MySQL インストールのほとんどの作業を実行するメインプログラムです。サーバーには、サーバーの起動と停止を支援する、関係するいくつかのスクリプトが付随します。

- mysqld

  SQL デーモン (すなわち MySQL Server)。クライアントは、サーバーに接続することでデータベースへアクセスするため、クライアントプログラムを使用するには、mysqld が稼働していなければなりません。セクション 4.3.1「mysqld — MySQL サーバー」を参照してください。

- mysqld_safe

  サーバー起動スクリプト。mysqld_safe は mysqld を起動しようとします。セクション4.3.2「mysqld_safe — MySQL サーバー起動スクリプト」を参照してください。

- mysql.server

  サーバー起動スクリプト。このスクリプトは、特定の実行レベルのシステムサービスを起動するスクリプトを含む、System V スタイルの実行ディレクトリを使用するシステム上で使用されます。これは、MySQL サーバーを起動するために、mysqld_safe を呼び出します。セクション4.3.3「mysql.server — MySQL サーバー起動スクリプト」を参照してください。

- mysqld_multi

  システムにインストールされている複数のサーバーの起動または停止を実行できるサーバー起動スクリプト。セクション4.3.4「mysqld_multi — 複数の MySQL サーバーの管理」を参照してください。

いくつかのプログラムは、MySQL のインストール中またはアップグレード中に、セットアップ操作を実行します。

- comp_err

  このプログラムは MySQL のビルド/インストールプロセス中に使用されます。これは、エラーソースファイルからエラーメッセージをコンパイルします。セクション4.4.1「comp_err — MySQL エラーメッセージファイルのコンパイル」を参照してください。

- mysql_install_db

このプログラムは、MySQL データベースを作成し、付与テーブルをデフォルト権限で初期化し、InnoDB システムテーブルスペースをセットアップします。通常、MySQL をはじめてシステムにインストールするときに一度だけ実行されます。セクション4.4.3「mysql_install_db — MySQL データディレクトリの初期化」、セクション2.10.1「Unix 類似システムでのインストール後の手順」、およびセクション4.4.3「mysql_install_db — MySQL データディレクトリの初期化」を参照してください。

- mysql_plugin

  このプログラムは MySQL Server プラグインを構成します。セクション4.4.4「mysql_plugin — MySQL サーバープラグインの構成」を参照してください。

- mysql_secure_installation

  このプログラムにより、MySQL インストールのセキュリティーを改善できます。SQL。セクション4.4.5「mysql_secure_installation — MySQL インストールのセキュリティー改善」を参照してください。

- mysql_tzinfo_to_sql

  このプログラムは、ホストシステムの zoneinfo データベース (タイムゾーンを記述しているファイルセット) の内容を使用して、タイムゾーンテーブルを mysql データベースにロードします。SQL。セクション4.4.6「mysql_tzinfo_to_sql — タイムゾーンテーブルのロード」を参照してください。

- mysql_upgrade

  このプログラムは MySQL のアップグレード操作のあとで使用されます。テーブルの非互換性をチェックして必要に応じて修復し、新しいバージョンの MySQL で行われた変更に合わせて付与テーブルを更新します。セクション4.4.7「mysql_upgrade — MySQL テーブルのチェックとアップグレード」を参照してください。

MySQL サーバーに接続するクライアントプログラム。

- mysql

  インタラクティブに SQL ステートメントを書き込む、またはバッチモードでファイルを使用してステートメントを実行するためのコマンド行ツールです。セクション4.5.1「mysql — MySQL コマンド行ツール」を参照してください。

- mysqladmin

  データベースの作成と削除、付与テーブルのリロード、テーブルのディスクへのフラッシュ、およびログファイルの再オープン等、管理操作を実行するクライアント。mysqladmin は、サーバーからバージョン、プロセス、およびステータス情報を取得するためにも使用できます。セクション4.5.2「mysqladmin — MySQL サーバーの管理を行うクライアント」を参照してください。

- mysqlcheck

  テーブルのチェック、修復、分析、および最適化を行うテーブルメンテナンスのクライアント。セクション4.5.3「mysqlcheck — テーブル保守プログラム」を参照してください。

- mysqldump

  MySQL データベースを SQL、テキスト、または XML としてファイルにダンプするクライアント。セクション4.5.4「mysqldump — データベースバックアッププログラム」を参照してください。

- mysqlimport

  LOAD DATA INFILE を使用して、テキストファイルを対応するテーブルにインポートするクライアント。セクション4.5.5「mysqlimport — データインポートプログラム」を参照してください。

- mysqlshow

  データベース、テーブル、カラム、およびインデックスの情報を表示するクライアント。セクション4.5.6「mysqlshow — データベース、テーブル、およびカラム情報の表示」を参照してください。

- mysqlslap

  MySQL Server のクライアント負荷を工ミュレートし、各ステージのタイミングをレポートするクライアント。複数のクライアントがサーバーにアクセスしているかのように作動します。セクション4.5.7「mysqlslap — 負荷工ミュレーションクライアント」を参照してください。

MySQL 管理プログラムおよびユーティリティープログラム:

- innochecksum

  オフライン InnoDB オフラインファイルのチェックサムユーティリティー。セクション4.6.1「innochecksum — オフライン InnoDB ファイルチェックサムユーティリティー」を参照してください。

- myisam_ftdump

  MyISAM テーブル内の全文インデックスの情報を表示するユーティリティー。セクション 4.6.2「myisam_ftdump — 全文インデックス情報の表示」を参照してください。

- myisamchk

  MyISAM テーブルの記述、チェック、最適化、および修復を行うユーティリティー。セクション 4.6.3「myisamchk — MyISAM テーブルメンテナンスユーティリティー」を参照してください。

- myisamlog

  MyISAM ログファイルの内容を処理するユーティリティー。セクション4.6.4「myisamlog — MyISAM ログファイルの内容の表示」を参照してください。

- myisampack

  MyISAM テーブルを圧縮してより小さい読み取り専用のテーブルを生成するユーティリティー。セクション 4.6.5「myisampack — 圧縮された読み取り専用の MyISAM テーブルの生成」を参照してください。

- mysql_config_editor

  .mylogin.cnf という名前のセキュアで暗号化されたログインファイルに認証情報を格納できるようにするユーティリティー。セクション4.6.6「mysql_config_editor — MySQL 構成ユーティリティー」を参照してください。

- mysqlaccess

  ホスト名、ユーザー名、およびデータベースの組み合わせに対してアクセス権限をチェックするスクリプト。セクション4.6.7「mysqlaccess — アクセス権限をチェックするクライアント」を参照してください。

- mysqlbinlog

  バイナリログからステートメントを読み取るためのユーティリティー。クラッシュ状態からリカバリするために、バイナリログファイルに含まれる実行ステートメントのログを使用することができます。セクション 4.6.8「mysqlbinlog — バイナリログファイルを処理するためのユーティリティー」を参照してください。

- mysqldumpslow

  スロークエリーログの内容を読み取って要約するユーティリティー。セクション4.6.9「mysqldumpslow — スロークエリーログファイルの要約」を参照してください。

- mysqlhotcopy

  サーバーの稼働中に MyISAM テーブルのバックアップを迅速に作成するユーティリティー。セクション 4.6.10「mysqlhotcopy — データベースバックアッププログラム」を参照してください。

- mysql_convert_table_format

  データベース内のテーブルを、所定のストレージエンジンを使用するように変換するユーティリティー。セクション4.6.11「mysql_convert_table_format — 指定されたストレージエンジンを使用するためのテーブルの変換」を参照してください。

- mysql_find_rows

  SQL ステートメント (更新ログなど) を含むファイルを読み取り、所定の正規表現に一致するステートメントを抽出するユーティリティー。セクション4.6.12「mysql_find_rows — ファイルからの SQL ステートメントの抽出」を参照してください。

- mysql_fix_extensions

  MyISAM テーブルファイルの拡張子を小文字に変換するユーティリティー。これは、ファイル名の大文字と小文字を区別しないシステムから、ファイル名の大文字と小文字を区別するシステムにファイルを転送したあと

で便利です。セクション4.6.13「mysql_fix_extensions — テーブルファイル名の拡張子の正規化」を参照してください。

- mysql_setpermission

  MySQL 付与テーブルにインタラクティブに許可を設定するためのユーティリティー。セクション 4.6.14「mysql_setpermission — 付与テーブルに許可をインタラクティブに設定」を参照してください。

- mysql_waitpid

  指定されたプロセス ID のプロセスを強制終了させるユーティリティー。セクション4.6.15「mysql_waitpid — プロセスを強制終了して終了を待機」を参照してください。

- mysql_zap

  パターンと一致するプロセスを強制終了させるユーティリティー。セクション4.6.16「mysql_zap — パターンに一致するプロセスを強制終了」を参照してください。

MySQL プログラム開発ユーティリティー:

- msql2mysql

  mSQL プログラムを MySQL に変換するシェルスクリプト。これはすべてのケースを扱うわけではありませんが、変換するときのスタートとしては便利です。セクション4.7.1「msql2mysql — mSQL プログラムを MySQL で使用するために変換」を参照してください。

- mysql_config

  MySQL プログラムをコンパイルするときに必要なオプション値を生成するシェルスクリプト。セクション 4.7.2「mysql_config — クライアントのコンパイル用オプションの表示」を参照してください。

- my_print_defaults

  オプションファイルのオプショングループにどのオプションがあるかを表示するユーティリティー。セクション4.7.3「my_print_defaults — オプションファイルからのオプションの表示」を参照してください。

- resolve_stack_dump

  数値スタックトレースダンプをシンボルに解決するユーティリティープログラム。セクション 4.7.4「resolve_stack_dump — 数値スタックトレースダンプをシンボルに解決」を参照してください。

その他のユーティリティー:

- perror

  システムあるいは MySQL エラーコードの意味を表示するユーティリティー。セクション4.8.1「perror — エラーコードの説明」を参照してください。

- replace

  入力テキストで文字列を置換するユーティリティープログラム。セクション4.8.2「replace — 文字列置換ユーティリティー」を参照してください。

- resolveip

  ホスト名を IP アドレスに、または IP アドレスをホスト名に解決するユーティリティープログラム。セクション4.8.3「resolveip — ホスト名と IP アドレスの解決」を参照してください。

Oracle Corporation は、MySQL Workbench GUI も提供します。これは、MySQL サーバーおよびデータベースの管理、クエリーの作成、実行、評価、およびほかのリレーショナルデータベース管理システムからのスキーマおよびデータを MySQL で使用するための移行を行うために使用されます。その他の GUI ツールには、MySQL Notifier および MySQL for Excel などがあります。

MySQL クライアント/サーバーライブラリを使用してサーバーと通信する MySQL クライアントプログラムは、次の環境変数を使用します。

環境変数	意味
MYSQL_UNIX_PORT	localhost への接続に使用される、デフォルト Unix ソケットファイル
MYSQL_TCP_PORT	TCP/IP 接続に使用されるデフォルトのポート番号

環境変数	意味
MYSQL_PWD	デフォルトパスワード
MYSQL_DEBUG	デバッグ中のデバッグトレースオプション
TMPDIR	一時テーブルや一時ファイルが作成されるディレクトリ

MySQL プログラムが使用する環境変数の全リストについては、セクション2.12「環境変数」を参照してください。

MYSQL_PWD の使用はセキュアではありません。セクション6.1.2.1「パスワードセキュリティーのためのエンドユーザーガイドライン」を参照してください。

4.2 MySQL プログラムの使用

4.2.1 MySQL プログラムの起動

MySQL プログラムをコマンド行から (つまり、シェルまたはコマンドプロンプトから) 起動するには、プログラム名に続けて、実行させたいことをプログラムに指示するために必要な任意のオプションまたはその他の引数を入力します。次のコマンドは、プログラム起動の例です。「shell>」 は、コマンドインタプリタのプロンプトを示します。この部分は入力には含まれません。表示される具体的なプロンプトは、コマンドインタプリタによって異なります。通常のプロンプトは、sh、ksh、または bash では $、csh または tcsh では %、Windows command.com または cmd.exe コマンドインタプリタでは C:\> です。

```
shell> mysql --user=root test
shell> mysqladmin extended-status variables
shell> mysqlshow --help
shell> mysqldump -u root personnel
```

単一または二重のダッシュ (「-」、「--」) で始まる引数は、プログラムオプションを指定します。オプションは通常、プログラムが行うサーバーへの接続のタイプを示したり、動作モードに影響したりします。オプション構文についてはセクション4.2.3「プログラムオプションの指定」で説明します。

オプションではない引数 (先頭にダッシュのない引数) は、プログラムに追加情報を提供します。たとえば、mysql プログラムは最初のオプションではない引数をデータベース名として解釈するため、コマンド mysql --user=root test は test データベースを使用することを示します。

個々のプログラムについて説明しているこのあとのセクションでは、プログラムがどのオプションをサポートするかを示し、追加のオプションではない引数があればその意味を説明しています。

一部のオプションは、複数のプログラムで共通です。これらのうち、もっともよく使用されるのは、接続パラメータを指定する --host (または -h)、--user (または -u)、および --password (または -p) の各オプションです。これらは、MySQL サーバーが稼働しているホストの名前、MySQL アカウントのユーザー名とパスワードを示します。すべての MySQL クライアントプログラムは、これらのオプションを理解します。これらのオプションにより、どのサーバーに接続するか、およびそのサーバーで使用するアカウントを指定できます。その他の接続オプションとしては、TCP/IP ポート番号を指定する --port (または -P)、そして Unix 上で Unix ソケットファイル (Windows では名前付きパイプ名) を指定する --socket (または -S) があります。接続オプションを指定するオプションの詳細は、セクション4.2.2「MySQL サーバーへの接続」を参照してください。

MySQL プログラムを起動するために、プログラムがインストールされている bin ディレクトリのパス名を使用する必要がある場合があります。bin ディレクトリ以外のディレクトリから MySQL を起動しようとすると毎回「program not found」 というエラーが表示される場合は、この場合に該当する可能性があります。MySQL プログラムをより使いやすくするために、MySQL の bin ディレクトリのパス名を PATH 環境変数設定に追加できます。それにより、プログラムのパス名全体ではなく名前のみを入力して実行できます。たとえば、mysql が /usr/local/mysql/bin にインストールされている場合、mysql と呼び出すことでプログラムを実行可能であり、/usr/local/mysql/bin/mysql と呼び出す必要はありません。

PATH 変数の設定する際の手順については、コマンドインタプリタのドキュメントを参照してください。環境変数を設定するための構文は、インタプリタ固有です。(これに関する説明は、セクション4.2.10「環境変数の設定」で述べられています。)PATH 設定を変更したら、設定が反映されるように、Windows の場合は新しいコンソールウィンドウを開き、Unix の場合はログインしなおします。

4.2.2 MySQL サーバーへの接続

クライアントプログラムが MySQL サーバーに接続できるためには、サーバーが稼働しているホストの名前および MySQL アカウントのユーザー名とパスワードなどの、適切な接続パラメータを使用する必要があります。各

接続パラメータにはデフォルト値がありますが、必要に応じてコマンド行またはオプションファイルでプログラムオプションを指定することによってオーバーライドできます。

次の例は mysql クライアントプログラムを使用しますが、原則は mysqldump、mysqladmin、または mysqlshow など、ほかのクライアントにも適用されます。

次のコマンドは、接続パラメータを明示的に指定せずに mysql を呼び出します。

```
shell> mysql
```

パラメータオプションがないため、デフォルト値が適用されます。

- デフォルトのホスト名は localhost です。Unix では、後述するようにこれには特別な意味があります。
- デフォルトのユーザー名は、Windows では ODBC、Unix では Unix のログイン名です。
- -p も --password も指定しない場合は、パスワードは送信されません。
- mysql では、最初のオプションではない引数はデフォルトデータベースの名前とみなされます。そのようなオプションがない場合は、mysql はデフォルトデータベースを選択しません。

パスワードのほか、ホスト名およびユーザー名を明示的に指定するには、コマンド行で適切なオプションを指定します。

```
shell> mysql --host=localhost --user=myname --password=mypass mydb
shell> mysql -h localhost -u myname -pmypass mydb
```

パスワードオプションについては、パスワード値はオプションです。

- -p または --password オプションを使用してパスワード値を指定する場合、-p または --password とそれに続くパスワードとの間にスペースがあってはいけません。
- -p または --password オプションを使用するがパスワード値を指定しない場合、パスワードを入力するようクライアントプログラムが要求します。パスワードは入力時に表示されません。これは、コマンド行でパスワードを提供するよりもセキュアです。システム上のほかのユーザーが ps auxw などのコマンドを実行して、コマンド行で指定されたパスワードを見ることができる場合があります。セクション6.1.2.1「パスワードセキュリティーのためのエンドユーザーガイドライン」を参照してください。

今述べたように、コマンド行にパスワード値を含めるのはセキュリティー上のリスクになり得ます。この問題を解決するには、パスワード値を入れずに --password または -p オプションを指定します。

```
shell> mysql --host=localhost --user=myname --password mydb
shell> mysql -h localhost -u myname -p mydb
```

パスワードオプションにパスワード値がない場合、クライアントプログラムはプロンプトを出力して、ユーザーがパスワードを入力するのを待機します。(これらの例では、mydb はその前のパスワードオプションとスペースで区切られているため、パスワードとは解釈されません。)

システムによっては、MySQL がパスワードを要求するために使用するライブラリルーチンが、自動的にパスワードを 8 文字に制限します。これはシステムライブラリの問題であり、MySQL の問題ではありません。内部的には、MySQL にはパスワードの長さに関する制限はありません。この問題を回避するには、MySQL のパスワードを 8 文字以下の値に変更するか、またはオプションファイルにパスワードを指定します。

Unix では、MySQL プログラムはホスト名 localhost を、ほかのネットワークベースのプログラムと比較して想定されるのとはおそらく異なる、特別な方法で扱います。localhost への接続で、MySQL プログラムは Unix ソケットファイルを使用してローカルサーバーに接続しようとします。これは、ポート番号を指定するために --port または -P オプションが与えられた場合にも生じます。クライアントがローカルサーバーに TCP/IP 接続を行うことを保証するには、--host または -h を使用して、ホスト名の値 127.0.0.1、またはローカルサーバーの IP アドレスまたは名前を指定します。--protocol=TCP オプションを使用して、localhost に対しても、接続プロトコルを明示的に指定することもできます。例:

```
shell> mysql --host=127.0.0.1
shell> mysql --protocol=TCP
```

--protocol オプションを使用すると、その他のオプションが通常デフォルトでほかのプロトコルになる場合でも、特定のタイプの接続を確立できます。

サーバーが IPv6 接続を受け付けるように構成されている場合、クライアントは --host=::1 を使用して IPv6 で接続できます。セクション5.1.9「IPv6 サポート」を参照してください。

Windows では、--pipe または --protocol=PIPE オプションを指定するか、または .(ピリオド) をホスト名として指定することによって、MySQL クライアントが名前付けパイプ接続を使用することを強制できます。名前付きパイプ接続が有効にされていない場合は、エラーが発生します。デフォルトのパイプ名を使用しない場合には --socket オプションを使用してパイプ名を指定します。

リモートサーバーへの接続では、常に TCP/IP が使用されます。次のコマンドは、デフォルトのポート番号 (3306) を使用して remote.example.com で動作するサーバーに接続します。

```
shell> mysql --host=remote.example.com
```

ポート番号を明示的に指定するには、--port オプションまたは -P オプションを使用します。

```
shell> mysql --host=remote.example.com --port=13306
```

ローカルサーバーへの接続にもポート番号を指定できます。ただし前述のように、Unix では localhost への接続にはデフォルトでソケットファイルが使用されます。前述のように TCP/IP 接続を強制しないと、ポート番号を指定するすべてのオプションは無視されます。

次のコマンドでは、Unix ではプログラムはソケットファイルを使用し、--port オプションは無視されます。

```
shell> mysql --port=13306 --host=localhost
```

ポート番号を使用するようにするには、プログラムを次のいずれかの方法で呼び出します。

```
shell> mysql --port=13306 --host=127.0.0.1
shell> mysql --port=13306 --protocol=TCP
```

次のリストは、クライアントプログラムがサーバーに接続する方法を制御するために使用できるオプションの概要です。

- --host=host_name, -h host_name

  サーバーが稼働しているホスト。デフォルト値は localhost です。

- --password[=pass_val], -p[pass_val]

  MySQL アカウントのパスワード。前述のように、パスワード値はオプションですが、指定する場合は -p または --password= とそれに続くパスワードとの間にスペースがあってはいけません。デフォルトではパスワードを送信しません。

- --pipe, -W

  Windows で、名前付きパイプを使用してサーバーに接続します。名前付きパイプ接続を可能にするには、サーバーは --enable-named-pipe オプションで起動する必要があります。

- --port=port_num, -P port_num

  TCP/IP を使用する接続で、接続に使用するポート番号。デフォルトのポート番号は 3306 です。

- --protocol={TCP|SOCKET|PIPE|MEMORY}

  このオプションは、サーバーへの接続に使用するプロトコルを明示的に指定します。このオプションは、ほかの接続パラメータによって、必要なプロトコル以外のものが通常使用される場合に役立ちます。たとえば、Unix では localhost への接続はデフォルトでは Unix ソケットファイルを使用して行われます。

  ```
  shell> mysql --host=localhost
  ```

  代わりに TCP/IP 接続を強制するには、--protocol オプションを指定します。

  ```
  shell> mysql --host=localhost --protocol=TCP
  ```

  次の表は、許可される --protocol オプション値を示し、各値を使用できるプラットフォームを示しています。値は大文字と小文字が区別されません。

--protocol の値	接続プロトコル	許可されるオペレーティングシステム
TCP	ローカルまたはリモートのサーバーへの TCP/IP 接続	すべて
SOCKET	ローカルサーバーへの Unix ソケットファイル接続	Unix のみ

--protocol の値	接続プロトコル	許可されるオペレーティングシステム
PIPE	ローカルまたはリモートのサーバーへの名前付きパイプ接続	Windows のみ
MEMORY	ローカルサーバーへの共有メモリー接続	Windows のみ

- --shared-memory-base-name=name

  Windows で、共有メモリーを使用して作成されるローカルサーバーへの接続の共有メモリー名。デフォルト値は MYSQL です。共有メモリー名では大文字と小文字を区別します。

  共有メモリー接続を可能にするには、サーバーは --shared-memory オプションで起動する必要があります。

- --socket=file_name, -S file_name

  Unix で、名前付きパイプを使用して行われるローカルサーバーへの接続で使用する Unix ソケットファイルの名前。デフォルトの Unix ソケットファイル名は /tmp/mysql.sock です。

  Windows の場合、ローカルサーバーへの接続で、使用する名前付きパイプの名前。デフォルトの Windows パイプ名は MySQL です。パイプ名では大文字と小文字を区別しません。

  名前付きパイプ接続を可能にするには、サーバーは --enable-named-pipe オプションで起動する必要があります。

- --ssl*

  --ssl で始まるオプションは、サーバーが SSL サポートで構成されている場合には、SSL を使用してサーバーへのセキュアな接続を確立するために使用されます。詳細については、セクション6.3.10.4「SSL コマンドのオプション」を参照してください。

- --user=user_name, -u user_name

  使用する MySQL アカウントのユーザー名。デフォルトのユーザー名は、Windows では ODBC、Unix では Unix のログイン名です。

クライアントプログラムを呼び出すたびに異なるデフォルト値を入力しなくてもいいように、コマンド行で接続を行うときに、それらを使用するように指定できます。これはいくつかの方法で実行できます。

- 接続パラメータをオプションファイルの [client] セクションに指定できます。このファイルの関係セクションは次のようになります。

```
[client]
host=host_name
user=user_name
password=your_pass
```

  セクション4.2.6「オプションファイルの使用」でオプションファイルの詳細を記述しています。

- 一部の接続パラメータは、環境変数を使用して指定できます。mysql のホストは、MYSQL_HOST で指定できます。MySQL ユーザー名は、USER を使用して指定できます (これは Windows のみです)。パスワードは、MYSQL_PWD を使用して指定できます (ただし、これはセキュアではありません。セクション6.1.2.1「パスワードセキュリティーのためのエンドユーザーガイドライン」を参照してください)。変数の一覧は、セクション2.12「環境変数」を参照してください。

4.2.3 プログラムオプションの指定

MySQL プログラムに対してオプションを指定する方法はいくつかあります。

- コマンド行で、プログラム名に続けてオプションをリストします。これは、プログラムの特定の起動に適用されるオプションに共通です。
- プログラムが起動するときに読み取るオプションファイルにオプションをリストします。これは、プログラムを実行するとき毎回使用するオプションに共通です。
- 環境変数にオプションをリストします (セクション4.2.10「環境変数の設定」を参照してください)。この方法は、プログラムを実行するとき毎回適用するオプションに便利です。実際には、このためにはオプションファイルを使用する方が一般的ですが、環境変数が非常に役立つ状況については、セクション5.3.3「Unix 上での複

数の MySQL インスタンスの実行」で説明しています。そこでは、このような変数を使用して、サーバーとクライアントプログラムに対して TCP/IP のポート番号および Unix ソケットファイルを指定する便利なテクニックについて説明しています。

オプションは順に処理されるため、あるオプションが複数回指定されている場合、最後のものが優先されます。次のコマンドでは、mysql は localhost で稼働しているサーバーに接続します。

```
shell> mysql -h example.com -h localhost
```

矛盾するオプションまたは関係するオプションが指定された場合、あとのオプションが先のオプションより優先されます。次のコマンドは mysql を「カラム名なし」モードで起動します。

```
shell> mysql --column-names --skip-column-names
```

MySQL プログラムは、まず環境変数を調べ、次にオプションファイルを読み取り、その次にコマンド行をチェックすることで、どのオプションが指定されたかを判断します。すなわち、環境変数はもっとも優先順位が低く、コマンド行のオプションがもっとも高くなります。

MySQL プログラムがオプションを処理する方法を利用して、プログラムのデフォルトオプション値をオプションファイルに指定できます。それにより、プログラムを実行するたびに入力するのを避けることができると同時に、必要に応じてコマンド行オプションを使用することにより、デフォルトをオーバーライドできます。

オプションは、すべて記述するか、またはあいまいでないプリフィクスとして指定できます。たとえば--compress オプションは --compr として mysqldump に指定できますが、--comp としては指定できません。後者はあいまいなためです。

```
shell> mysqldump --comp
mysqldump: ambiguous option '--comp' (compatible, compress)
```

オプションのプリフィクスを使用すると、プログラムに対して新しいオプションが実装された場合に問題となることがあることに注意してください。現在あいまいでないプリフィクスが、将来あいまいになる可能性があります。

> **注記**
>
> MySQL 5.6.13 以降、あいまいでないプリフィクスは非推奨です。あいまいでないプリフィクスが指定された場合、フィードバックを提供するため警告が発生します。オプションのプリフィクスは MySQL 5.7 ではサポートされなくなりました。完全なオプションのみが受け入れられます。

4.2.4 コマンド行でのオプションの使用

コマンド行で指定されるプログラムオプションは次のルールに従います。

- オプションはコマンド名のあとに指定します。
- オプション引数は、それがオプション名の短い形式か長い形式かによって、1 つまたは 2 つのダッシュで始まります。多くのオプションには短い形式と長い形式の両方があります。たとえば、-? と --help は、 MySQL プログラムに対してヘルプメッセージの表示を指示するオプションの短い形式と長い形式です。
- オプション名では大文字と小文字が区別されます。-v と -V はいずれも正当で、異なる意味を持ちます。(これらは、--verbose および --version オプションに対応する短い形式です。)
- 一部のオプションは、オプション名に続いて値を取ります。たとえば、-h localhost または --host=localhost は、クライアントプログラムに対して MySQL サーバーホストを示します。オプション値は、プログラムに対して MySQL サーバーが稼働しているホストの名前を示します。
- 値を取る長いオプションでは、オプション名と値を 「=」 記号で区切ります。値を取る短いオプションでは、オプション値はオプション文字の直後に続けるか、または間にスペースがあってもかまいません。-hlocalhost と -h localhost は同等です。このルールの例外は、MySQL のパスワードを指定するオプションです。このオプションは、長い形式で --password=pass_val として、または --password として指定できます。後者の場合 (パスワード値を指定しない場合)、プログラムはパスワードを要求します。パスワードオプションは、短い形式で -ppass_val または -p としても指定できます。しかし短い形式の場合、パスワード値を指定する場合は、間にスペースを入れずにオプション文字に続けなければいけません。この理由は、オプション文字にスペースが続く場合、続く引数がパスワード値なのかほかの種類の引数なのかをプログラムが判断する方法がないためです。その結果、次の 2 つのコマンドは 2 つのまったく異なる意味を持ちます。

```
shell> mysql -ptest
shell> mysql -p test
```

最初のコマンドは、パスワード値 test を使用することを mysql に指示しますが、デフォルトデータベースの指定は行いません。2 番目は、パスワード値を要求し、test をデフォルトデータベースとして使用することを mysql に指示します。

- オプション名内では、ダッシュ (「-」) と下線 (「_」) を区別なく使用できます。たとえば、--skip-grant-tables と --skip_grant_tables は同等です。(ただし、先頭のダッシュは下線で指定することはできません。)
- 数値を取るオプションについては、値は、1024、1024^2 または 1024^3 の乗数を示すために、K、M または G (大文字または小文字) のサフィクスで指定できます。たとえば、次のコマンドは mysqladmin に対し、サーバーに 1024 回 ping を実行し、各 ping の間に 10 秒間スリープすることを指示します。

```
mysql> mysqladmin --count=1K --sleep=10 ping
```

スペースを含むオプション値をコマンド行で指定する場合は、引用符で囲まなければなりません。たとえば、--execute (または -e) オプションを mysql とともに使用して、SQL ステートメントをサーバーに渡すことができます。このオプションが使用されると、mysql はオプション値のステートメントを実行して終了します。ステートメントは引用符で囲む必要があります。たとえば、次のコマンドを使用してユーザーアカウントのリストを取得できます。

```
mysql> mysql -u root -p --execute="SELECT User, Host FROM mysql.user"
Enter password: ******
+------+-----------+
| User | Host      |
+------+-----------+
|      | gigan     |
| root | gigan     |
|      | localhost |
| jon  | localhost |
| root | localhost |
+------+-----------+
shell>
```

長い形式 (--execute) には、等号 (=) が続くことに注意してください。

ステートメント内で引用符に囲まれた値を使用する場合は、内側の引用符をエスケープするか、ステートメント内ではステートメント自体を囲むのに使用する引用符とは異なるタイプの引用符を使用する必要があります。一重引用符または二重引用符を使用できるかどうか、および引用符文字をエスケープするための構文は、コマンドプロセッサの機能により異なります。たとえば、コマンドプロセッサが一重引用符または二重引用符の使用をサポートする場合、ステートメントの周囲には二重引用符を使用し、ステートメント内の引用符で囲む値には一重引用符を使用できます。

コマンド行のオプション値に複数の SQL ステートメントをセミコロンで区切って渡すことができます。

```
shell> mysql -u root -p -e "SELECT VERSION();SELECT NOW()"
Enter password: ******
+-----------------+
| VERSION()       |
+-----------------+
| 5.1.5-alpha-log |
+-----------------+
+---------------------+
| NOW()               |
+---------------------+
| 2006-01-05 21:19:04 |
+---------------------+
```

4.2.5 プログラムオプション修飾子

一部のオプションは「ブール値」で、オンまたはオフにできる動作を制御します。たとえば、mysql クライアントは、クエリーの結果の先頭にカラム名の行を表示するかどうかを決定するオプション --column-names をサポートします。デフォルトでは、このオプションは有効です。ただし、初期ヘッダーラインを想定せずデータのみを想定するような別のプログラムに mysql の出力を送信する場合などは、この機能を無効にするとよいでしょう。

カラム名を無効にするには、次のいずれかの形式を使用してオプションを指定します。

```
--disable-column-names
--skip-column-names
--column-names=0
```

--disable プリフィクスと --skip プリフィクス、および =0 サフィクスはすべて同じ効果を持ち、オプションをオフにします。

オプションを「有効」にする形式は、次のうちのいずれかで指定できます。

```
--column-names
--enable-column-names
--column-names=1
```

MySQL 5.6.2 以降では、ON、TRUE、OFF、および FALSE もブール値オプションとみなされます (大文字小文字は区別しません)。

オプションに --loose プリフィクスがある場合、プログラムがオプションを認知できない場合にはエラーで終了せず、警告のみを発行します。

```
shell> mysql --loose-no-such-option
mysql: WARNING: unknown option '--loose-no-such-option'
```

--loose プリフィクスは、MySQL を同じマシン上に複数回インストールした状態でプログラムを起動し、オプションファイルにオプションをリストする場合に役立ちます。プログラムのすべてのバージョンには認知されないようなオプションを --loose プリフィクス (またはオプションファイル内では loose) を使用して指定できます。オプションを認識するバージョンのプログラムは通常どおりに処理し、認識しないバージョンは警告を発行して無視します。

mysqld では、クライアントプログラムが設定できる動的システム変数の大きさに制限を設けることができます。これを実行するには、--maximum プリフィクスを変数名とともに使用します。たとえば、--maximum-query_cache_size=4M は、クライアントが 4M バイトを超えるクエリーキャッシュサイズを作成できなくなります。

4.2.6 オプションファイルの使用

ほとんどの MySQL プログラムはオプションファイル (構成ファイルと呼ばれることもあります) から起動オプションを読み取ることができます。オプションファイルは、よく使用されるオプションを指定するための便利な方法を提供し、プログラムを実行するたびにコマンド行で入力する必要がなくなります。MySQL サーバーには MySQL によって多数の事前構成済みのオプションファイルが提供されます。

プログラムがオプションファイルを読み取るかどうかを判断するには、--help オプションを使用してプログラムを呼び出します。(mysqld では、--verbose および --help を使用します。)プログラムがオプションファイルを読み取る場合は、どのファイルを探すのか、およびどのオプショングループを認識するのかが、ヘルプメッセージに示されます。

ログインパスオプションを含む .mylogin.cnf ファイルは、mysql_config_editor ユーティリティーによって作成されます。セクション4.6.6「mysql_config_editor — MySQL 構成ユーティリティー」を参照してください。「ログインパス」は、host、user、および password という限定されたオプションのセットのみを許可するオプショングループです。クライアントプログラムは、.mylogin.cnf からどのログインパスを読み取るのかを、--login-path オプションを使用して指定します。

別のファイル名を指定するには、MYSQL_TEST_LOGIN_FILE 環境変数を設定します。この変数は mysql-test-run.pl テストユーティリティーが使用しますが、mysql_config_editor および mysql、mysqladmin、などの MySQL クライアントによっても認識されます。

> **注記**
>
> MySQL Cluster プログラムで使用されるオプションファイルは、セクション 18.3「MySQL Cluster の構成」で説明します。

Windows では、MySQL プログラムは起動オプションを次のファイルから指定された順 (トップの項目が最初に使用されます) で読み取ります。

ファイル名	目的
%PROGRAMDATA%\MySQL\MySQL Server 5.6\my.ini, %PROGRAMDATA%\MySQL\MySQL Server 5.6\my.cnf	グローバルオプション
%WINDIR%\my.ini, %WINDIR%\my.cnf	グローバルオプション
C:\my.ini, C:\my.cnf	グローバルオプション

ファイル名	目的
INSTALLDIR\my.ini, INSTALLDIR\my.cnf	グローバルオプション
defaults-extra-file	--defaults-extra-file=path によって指定されるファイル (ある場合)
%APPDATA%\MySQL\.mylogin.cnf	ログインパスオプション

%PROGRAMDATA% は、ホスト上のすべてのユーザーのアプリケーションデータを含むファイルシステムディレクトリを示します。このパスはデフォルトで、Microsoft Windows Vista 以降では C:\ProgramData、Microsoft Windows のそれ以前のバージョンでは C:\Documents and Settings\All Users\Application Data です。

%WINDIR% は、Windows ディレクトリの場所を示します。これは一般には C:\WINDOWS です。次のコマンドを使用して WINDIR 環境変数の値から正確な場所を割り出すことができます。

```
C:\> echo %WINDIR%
```

INSTALLDIR は、MySQL のインストールディレクトリを示します。これは通常、MySQL 5.6 がインストールウィザードおよび構成ウィザードを使用してインストールされた場合には、C:\PROGRAMDIR\MySQL\MySQL 5.6 Server です。ここで、PROGRAMDIR はプログラムディレクトリ (英語バージョンの Windows では通常 Program Files) です。セクション2.3.3「MySQL Installer を使用した MySQL の Microsoft Windows へのインストール」を参照してください。

%APPDATA% は、Windows アプリケーションデータディレクトリの値を示します。次のコマンドを使用して APPDATA 環境変数の値から正確な場所を判断できます。

```
C:\> echo %APPDATA%
```

Unix、Linux、および OS X では、MySQL プログラムは起動オプションを次のファイルから指定された順 (トップの項目が最初に使用されます) で読み取ります。

ファイル名	目的
/etc/my.cnf	グローバルオプション
/etc/mysql/my.cnf	グローバルオプション
SYSCONFDIR/my.cnf	グローバルオプション
$MYSQL_HOME/my.cnf	サーバー固有のオプション
defaults-extra-file	--defaults-extra-file=path によって指定されるファイル (ある場合)
~/.my.cnf	ユーザー固有のオプション
~/.mylogin.cnf	ログインパスオプション

~ は、現在のユーザーのホームディレクトリ ($HOME の値) を示します。

SYSCONFDIR は、MySQL がビルドされたときに SYSCONFDIR オプションとともに CMake に指定されたディレクトリを示します。デフォルトでは、これはコンパイル済みのインストールディレクトリの下にある etc ディレクトリです。

MYSQL_HOME はサーバー固有の my.cnf ファイルが存在するディレクトリへのパスを含む環境変数です。MYSQL_HOME がセットされていない状態で mysqld_safe プログラムを使ってサーバーを起動すると、mysqld_safe は次のように MYSQL_HOME をセットしようとします。

- BASEDIR および DATADIR が、それぞれ MySQL ベースディレクトリとデータディレクトリのパス名を示すようにします。
- DATADIR には my.cnf ファイルが存在し、BASEDIR には存在しない場合、mysqld_safe は MYSQL_HOME を DATADIR にセットします。
- そうでない場合は、MYSQL_HOME がセットされておらず、my.cnf ファイルが DATADIR に存在しない場合、mysqld_safe は BASEDIR に MYSQL_HOME をセットします。

MySQL 5.6 では、DATADIR を my.cnf の場所として使用することは非推奨です。

一般的に、DATADIR はバイナリインストールでは /usr/local/mysql/data、ソースインストールでは /usr/local/var です。これは構成時に指定されたデータディレクトリの場所であって、mysqld が起動したとき --datadir オプショ

ンで指定されるものではないということに注意してください。実行時に --datadir を使用しても、サーバーがオプションファイルを探す際に何の影響ももたらしません。これはサーバーがオプションを処理する前にオプションファイルを探すからです。

MySQL は、前述の順序でオプションファイルを検索し、存在するものをすべて読み取ります。使用したいオプションファイルが存在しない場合は、プレーンテキストエディタで作成します。

所定のオプションが複数ある場合は、最後のインスタンスが優先されます。1 つ例外があります。mysqld では、オプションファイルで指定されたユーザーがコマンド行でオーバーライドされるのを防ぐため、安全対策として --user オプションの最初のインスタンスが使用されます。

> **注記**
>
> Unix プラットフォームでは、MySQL はだれでも書き込める構成ファイルを無視します。これはセキュリティー対策として意図的なものです。

MySQL プログラムを実行する際にコマンド行で指定できるすべての長いオプションは、オプションファイルでも指定できます。プログラムに対して使用可能なオプションのリストを取得するには、--help オプションを使用してそのプログラムを実行します。

オプションファイルでオプションを指定する構文は、コマンド行構文と同様です (セクション4.2.4「コマンド行でのオプションの使用」を参照してください)。ただしオプションファイルでは、先頭の 2 つのダッシュはオプション名から省略し、1 行で 1 つのオプションのみを指定します。たとえば、コマンド行での --quick および --host=localhost は、オプションファイルでは独立した行にある quick および host=localhost として指定するようにしてください。--loose-opt_name 形式のオプションをオプションファイルで指定するには、loose-opt_name として作成します。

オプションファイルの空の行は無視されます。空でない行は次のいずれかの形式を取ることができます。

- #comment, ;comment

  「#」 または 「;」 で始まるコメント行。「#」 を使用するコメントは、行の途中で開始することもできます。

- [group]

  group はオプションを設定するプログラムまたはグループの名前です。グループ行のあと、すべてのオプション設定行は、オプションファイルが終了するか、または別のグループ行が指定されるまで、名前を指定したグループに適用されます。オプショングループ名では、大文字と小文字は区別されません。

- opt_name

  これは、コマンド行の --opt_name と同等です。

- opt_name=value

  これは、コマンド行の --opt_name=value と同等です。オプションファイルでは、「=」 文字の周囲にスペースを置くことができます。これはコマンド行ではできません。オプションで、値を一重引用符または二重引用符で囲むことができます。これは、値が 「#」 コメント文字を含む場合に便利です。

先頭および末尾のスペースは、自動的にオプション名および値から削除されます。

エスケープシーケンス 「\b」、「\t」、「\n」、「\r」、「\\」、および 「\s」 をオプション値で使用して、バックスペース、タブ、改行、復帰改行、バックスラッシュ、およびスペース文字を表すことができます。オプションファイルでのエスケープのルール:

- バックスラッシュに有効なエスケープシーケンス文字が続く場合、そのシーケンスはシーケンスが表す文字に変換されます。たとえば、「\s」 はスペースに変換されます。
- バックスラッシュに有効なエスケープシーケンス文字が続かない場合は、変更されません。たとえば、「\S」 はそのままです。

前述のルールは、バックスラッシュそのものは 「\\」 で指定できることを意味します。または、有効なエスケープシーケンス文字が続かなければ 「\」 で指定できます。

オプションファイルにおけるエスケープシーケンスのルールは、SQL ステートメントにおける文字列リテラルのエスケープシーケンスのルールとは若干異なります。後者の場合は、「x」 が有効なエスケープシーケンス文字ではない場合、「\x」 は 「\x」 ではなく 「x」 になります。セクション9.1.1「文字列リテラル」を参照してください。

オプションファイル値のエスケープのルールは、「\」 をパス名区切り文字として使用する Windows パス名に特に関係します。Windows パス名の区切り文字は、エスケープシーケンス文字が続く場合は 「\\」 と記述する必要があります。そうでない場合は 「\\」 または 「\」 と記述できます。または、Windows パス名に 「/」 を使用することもでき、「\」 として扱われます。オプションファイルでベースディレクトリ C:\Program Files\MySQL\MySQL Server 5.6 を指定するとします。これはいくつかの方法で実行できます。例:

```
basedir="C:\Program Files\MySQL\MySQL Server 5.6"
basedir="C:\\Program Files\\MySQL\\MySQL Server 5.6"
basedir="C:/Program Files/MySQL/MySQL Server 5.6"
basedir=C:\\Program\sFiles\\MySQL\\MySQL\sServer\s5.6
```

オプショングループ名がプログラム名と同じである場合、グループ内のオプションは特にそのプログラムに適用されます。たとえば、[mysqld] グループおよび [mysql] グループは、それぞれ mysqld サーバーおよび mysql クライアントプログラムに適用されます。

[client] オプショングループは、すべてのクライアントプログラムによって読み取られます (しかし、mysqld には読み取られません)。このため、すべてのクライアントに適用されるオプションを指定できます。たとえば、[client] は、サーバーへの接続に使用するパスワードを指定するために使用するのに最適なグループです。(ただし、ほかの人にパスワードを知られないように、オプションファイルは必ず自分にしか読み書きできないようにしてください。)使用するすべてのクライアントプログラムが [client] グループを認識しないかぎり、オプションに置かないようにしてください。そのオプションを理解しないプログラムを実行しようとすると、そのプログラムはエラーメッセージを表示してから終了します。

一般的なグローバルオプションファイルを次に示します。

```
[client]
port=3306
socket=/tmp/mysql.sock

[mysqld]
port=3306
socket=/tmp/mysql.sock
key_buffer_size=16M
max_allowed_packet=8M

[mysqldump]
quick
```

前述のオプションファイルは、key_buffer_size 変数および max_allowed_packet 変数をセットする行に var_name=value 構文を使用します。

一般的なユーザーオプションファイルを次に示します。

```
[client]
# The following password will be sent to all standard MySQL clients
password="my_password"

[mysql]
no-auto-rehash
connect_timeout=2

[mysqlhotcopy]
interactive-timeout
```

MySQL の特定のリリースシリーズの mysqld サーバーのみに読み取らせるべきオプショングループを作成する場合は、[mysqld-5.5]、[mysqld-5.6]、などの名前のグループを使用することで実行できます。次のグループは、--new オプションが 5.6.x バージョン番号の MySQL サーバーのみに使用されるべきであることを示します。

```
[mysqld-5.6]
new
```

オプションファイルで、!include ディレクティブを使用してほかのオプションファイルをインクルードしたり、!includedir を使用して特定のディレクトリでオプションファイルを検索したりできます。たとえば、/home/mydir/myopt.cnf ファイルをインクルードするには、次のディレクティブを使用します。

```
!include /home/mydir/myopt.cnf
```

/home/mydir ディレクトリを検索してそこで見つかったオプションファイルを読み取るには、次のディレクティブを使用します。

```
!includedir /home/mydir
```

ディレクトリ内のオプションファイルが読み取られる順序は保証されません。

> **注記**
>
> 現在、Unix オペレーティングシステムで !includedir ディレクティブを使用して検索およびインクルードされるファイルは、.cnf で終わるファイル名を持っていなくてはなりません。Windows においては、このディレクティブは .ini または .cnf 拡張子を持つファイルをチェックします。

インクルードされるオプションファイルの内容は、ほかのオプションファイルと同様に記述します。すなわち、オプションのグループを含み、それぞれの前にオプションが適用されるプログラムを示す [group] 行があるようにしてください。

インクルードされるファイルの処理中、現在のプログラムが検索するグループ内のオプションのみが使用されます。その他のグループは無視されます。my.cnf ファイルに次の行が含まれるとします。

```
!include /home/mydir/myopt.cnf
```

また、/home/mydir/myopt.cnf は次のようであるとします。

```
[mysqladmin]
force

[mysqld]
key_buffer_size=16M
```

my.cnf が mysqld によって処理される場合、/home/mydir/myopt.cnf 内の [mysqld] グループのみが使用されます。このファイルが mysqladmin によって処理される場合、[mysqladmin] グループのみが使用されます。このファイルがその他のプログラムによって処理される場合、/home/mydir/myopt.cnf のオプションは使用されません。

!includedir ディレクティブは同様に処理されますが、指名されたディレクトリ内のすべてのオプションファイルが読み取られる点が異なります。

4.2.7 オプションファイルの処理に影響するコマンド行オプション

オプションファイルをサポートするほとんどの MySQL プログラムは、次のオプションを処理します。それらはオプションファイルの処理に影響するため、オプションファイルではなくコマンド行で指定する必要があります。これらのオプションはそれぞれ、正しく機能するためにはほかのオプションより前に指定する必要があります。ただし、次の例外があります。

- --print-defaults は、--defaults-file または --defaults-extra-file の直後に使用できます。
- Windows では、サーバーが --defaults-file および --install オプションで起動される場合、--install が最初でなければなりません。セクション2.3.5.7「Windows のサービスとして MySQL を起動する」を参照してください。

ファイル名を指定する場合、「~」シェルメタキャラクタは想定どおり解釈されない場合があるため、使用を避けるようにしてください。

- --defaults-extra-file=file_name

  このオプションファイルは、グローバルオプションファイルのあとに読み取りますが、(UNIX では) ユーザーオプションファイルの前に読み取るようにしてください。ファイルが存在しないかアクセスできない場合、エラーが発生します。file_name は、フルパス名でなく相対パス名として指定された場合、現行ディレクトリを基準にして解釈されます。

- --defaults-file=file_name

  指定されたオプションファイルのみ使用します。ファイルが存在しないかアクセスできない場合、エラーが発生します。file_name は、フルパス名でなく相対パス名として指定された場合、現行ディレクトリを基準にして解釈されます。

- --defaults-group-suffix=str

  通常のオプショングループだけでなく、通常の名前に str のサフィクスが付いたグループも読み取ります。たとえば、mysql クライアントは通常 [client] グループおよび [mysql] グループを読み取ります。--defaults-group-suffix=_other オプションを指定した場合、mysql は [client_other] グループおよび [mysql_other] グループも読み取ります。

- --login-path=name

  指名されたログインパスから .mylogin.cnf ログインファイルのオプションを読み取ります。「ログインパス」は、host、user、および password という限定されたオプションのセットのみを許可するオプショングルー

プです。ログインパスは、サーバーホストおよびそのサーバーで認証するための認証情報を示す値のセットであると考えてください。ログインパスファイルを作成するには、mysql_config_editor ユーティリティーを使用します。セクション4.6.6「mysql_config_editor — MySQL 構成ユーティリティー」を参照してください。このオプションは MySQL 5.6.6 で追加されました。

- --no-defaults

  オプションファイルを読み取りません。オプションファイルから不明のオプションを読み取ることが原因でプログラムの起動に失敗する場合、--no-defaults を使用して、オプションを読み取らないようにすることができます。

  例外として、.mylogin.cnf ファイルは、存在する場合はすべての場合に読み取られます。これにより、--no-defaults が使用された場合に、コマンド行よりも安全な方法でパスワードを指定できます。(.mylogin.cnf は mysql_config_editor ユーティリティーによって作成されます。セクション4.6.6「mysql_config_editor — MySQL 構成ユーティリティー」を参照してください)。

- --print-defaults

  プログラム名と、オプションファイルから受け取るすべてのオプションを出力します。

4.2.8 プログラム変数の設定へのオプションの使用

多くの MySQL プログラムには内部変数があり、実行時に SET ステートメントを使用して設定できます。セクション13.7.4「SET 構文」およびセクション5.1.5「システム変数の使用」を参照してください。

これらのプログラム変数のほとんどは、プログラムオプションの指定に適用されるのと同じ構文を使用して、サーバーの起動時にも設定できます。たとえば、mysql には通信バッファーの最大サイズを制御する max_allowed_packet 変数があります。mysql に対して max_allowed_packet 変数を 16M バイトの値にセットするには、次のコマンドのいずれかを使用してください。

```
shell> mysql --max_allowed_packet=16777216
shell> mysql --max_allowed_packet=16M
```

最初のコマンドは値をバイトで指定します。2 番目は値を M バイトで指定します。数値を取る変数については、1024、1024^2 または 1024^3 の乗数を示すために、K、M または G (大文字または小文字) のサフィクスで値を指定できます。(たとえば、max_allowed_packet を設定するために使用される場合、サフィクスは K バイト、M バイトまたは G バイトの単位を示します。)

オプションファイルでは、変数の設定は先頭のダッシュなしで指定されます。

```
[mysql]
max_allowed_packet=16777216
```

または:

```
[mysql]
max_allowed_packet=16M
```

お好みで、変数名内の下線をダッシュとして指定できます。次のオプショングループは同等です。どちらもサーバーのキーバッファーを 512M バイトに設定します。

```
[mysqld]
key_buffer_size=512M

[mysqld]
key-buffer-size=512M
```

変数は、すべて記述するか、またはあいまいでないプリフィクスとして指定できます。たとえば、max_allowed_packet 変数は mysql に対して --max_a として設定できますが、--max としては指定できません。後者はあいまいなためです。

```
shell> mysql --max=1000000
mysql: ambiguous option '--max=1000000' (max_allowed_packet, max_join_size)
```

変数のプリフィクスを使用すると、プログラムに対して新しい変数が実装された場合に問題となることがあることに注意してください。現在あいまいでないプリフィクスが、将来あいまいになる可能性があります。

値乗数を指定するサフィクスは、サーバーの起動時に変数を設定するときに使用できますが、実行時に SET で値を設定するためには使用できません。一方、SET を使用すると、式を使用して変数の値を割り当てることができ

ますが、サーバーの起動時に変数を設定するときには使用できません。たとえば、サーバーの起動時に次の 1 行目は有効ですが 2 行目は無効です。

```
shell> mysql --max_allowed_packet=16M
shell> mysql --max_allowed_packet=16*1024*1024
```

逆に、実行時に次の 2 行目は有効ですが 1 行目は無効です。

```
mysql> SET GLOBAL max_allowed_packet=16M;
mysql> SET GLOBAL max_allowed_packet=16*1024*1024;
```

4.2.9 オプションのデフォルト、値を想定するオプション、および = 記号

慣例により、値を割り当てるオプションの長い形式は、次のように等号 (=) で記述されます。

```
shell> mysql --host=tonfisk --user=jon
```

値を必要とするオプション (つまり、デフォルトの値がないもの) では等号は必要ではないため、次も有効です。

```
shell> mysql --host tonfisk --user jon
```

どちらの場合も、mysql クライアントは 「tonfisk」 という名前のホストで稼働している MySQL サーバーに、ユーザー名 「jon」 のアカウントを使用して接続しようとします。

この動作のため、値を想定するオプションに値を指定しない場合に、時として問題が生じることがあります。次の例を見てください。ユーザーがホスト tonfisk で稼働している MySQL サーバーに、ユーザー jon として接続します。

```
shell> mysql --host 85.224.35.45 --user jon
Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 3
Server version: 5.6.23 Source distribution

Type 'help;' or '\h' for help. Type '\c' to clear the buffer.

mysql> SELECT CURRENT_USER();
+----------------+
| CURRENT_USER() |
+----------------+
| jon@%          |
+----------------+
1 row in set (0.00 sec)
```

これらのオプションの 1 つに対して必要な値を省略すると、次に示すような エラーが生じます。

```
shell> mysql --host 85.224.35.45 --user
mysql: option '--user' requires an argument
```

この場合、mysql は、コマンド行で --user オプションのあとに何もないため、値を見つけられませんでした。ただし、使用される最後のオプションではないオプションの値を省略すると、想定外の別の エラーが生じます。

```
shell> mysql --host --user jon
ERROR 2005 (HY000): Unknown MySQL server host '--user' (1)
```

mysql はコマンド行で --host に続く任意の文字列をホスト名と想定するため、--host --user は --host=--user と解釈され、クライアントは 「--user」 という名前のホストで稼働している MySQL サーバーに接続しようとします。

デフォルト値を持つオプションは、値を割り当てる場合には必ず等号が必要です。そうしないとエラーになります。たとえば、MySQL サーバー --log-error オプションはデフォルト値 host_name.err を持ちます。ここで、host_name は MySQL が稼働しているホストの名前です。ホスト名が 「tonfisk」 であるコンピュータ上で MySQL を稼働しているとします。次のように mysqld_safe を呼び出した場合を考えます。

```
shell> mysqld_safe &
[1] 11699
shell> 080112 12:53:40 mysqld_safe Logging to '/usr/local/mysql/var/tonfisk.err'.
080112 12:53:40 mysqld_safe Starting mysqld daemon with databases from /usr/local/mysql/var
shell>
```

サーバーのシャットダウン後、次のように再起動します。

```
shell> mysqld_safe --log-error &
[1] 11699
shell> 080112 12:53:40 mysqld_safe Logging to '/usr/local/mysql/var/tonfisk.err'.
```

```
080112 12:53:40 mysqld_safe Starting mysqld daemon with databases from /usr/local/mysql/var
shell>
```

コマンド行で --log-error にはほかに何も続いておらず、独自のデフォルト値が供給されるため、結果は同じです。(& 文字は、オペレーティングシステムに対して MySQL をバックグランドで実行することを指示します。MySQL 自身はこれを無視します。)ここで、エラーを my-errors.err という名前のファイルに記録するとします。--log-error my-errors でサーバーを起動しようとする可能性がありますが、これは次に示すように、意図した効果を持ちません。

```
shell> mysqld_safe --log-error my-errors &
[1] 31357
shell> 080111 22:53:31 mysqld_safe Logging to '/usr/local/mysql/var/tonfisk.err'.
080111 22:53:32 mysqld_safe Starting mysqld daemon with databases from /usr/local/mysql/var
080111 22:53:34 mysqld_safe mysqld from pid file /usr/local/mysql/var/tonfisk.pid ended

[1]+  Done                    ./mysqld_safe --log-error my-errors
```

サーバーは /usr/local/mysql/var/tonfisk.err をエラーログとして起動しようとしましたが、シャットダウンしました。このファイルの最後の数行を調べると理由がわかります。

```
shell> tail /usr/local/mysql/var/tonfisk.err
080111 22:53:32  InnoDB: Started; log sequence number 0 46409
/usr/local/mysql/libexec/mysqld: Too many arguments (first extra is 'my-errors').
Use --verbose --help to get a list of available options
080111 22:53:32 [ERROR] Aborting

080111 22:53:32  InnoDB: Starting shutdown...
080111 22:53:34  InnoDB: Shutdown completed; log sequence number 0 46409
080111 22:53:34 [Note] /usr/local/mysql/libexec/mysqld: Shutdown complete

080111 22:53:34 mysqld_safe mysqld from pid file /usr/local/mysql/var/tonfisk.pid ended
```

--log-error オプションはデフォルト値を供給するため、次に示すように、別の値を割り当てるには等号を使用する必要があります。

```
shell> mysqld_safe --log-error=my-errors &
[1] 31437
shell> 080111 22:54:15 mysqld_safe Logging to '/usr/local/mysql/var/my-errors.err'.
080111 22:54:15 mysqld_safe Starting mysqld daemon with databases from /usr/local/mysql/var

shell>
```

今度はサーバーは正常に起動し、エラーをファイル /usr/local/mysql/var/my-errors.err に口ギングしています。

オプションファイルでオプション値を指定する場合も、同様の問題が生じる可能性があります。たとえば、次の内容の my.cnf ファイルを考えます。

```
[mysql]

host
user
```

mysql クライアントがこのファイルを読み取ると、これらのエントリは --host --user または --host=--user と解釈され、次に示すような結果になります。

```
shell> mysql
ERROR 2005 (HY000): Unknown MySQL server host '--user' (1)
```

ただし、オプションファイルでは等号は想定されません。my.cnf ファイルが次に示すようになっているとします。

```
[mysql]

user jon
```

この場合、mysql を起動しようとすると別のエラーになります。

```
shell> mysql
mysql: unknown option '--user jon'
```

host=tonfisk ではなく host tonfisk とオプションファイルに記述した場合も同様のエラーが生じます。代わりに、等号を使用しなくてはなりません。

```
[mysql]

user=jon
```

これでログインが成功します。

```
shell> mysql
Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 5
Server version: 5.6.23 Source distribution

Type 'help;' or '\h' for help. Type '\c' to clear the buffer.

mysql> SELECT USER();
+---------------+
| USER()        |
+---------------+
| jon@localhost |
+---------------+
1 row in set (0.00 sec)
```

これはコマンド行での動作と同じではありません。コマンド行では等号は必要ではありません。

```
shell> mysql --user jon --host tonfisk
Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 6
Server version: 5.6.23 Source distribution

Type 'help;' or '\h' for help. Type '\c' to clear the buffer.

mysql> SELECT USER();
+---------------+
| USER()        |
+---------------+
| jon@tonfisk   |
+---------------+
1 row in set (0.00 sec)
```

MySQL 5.6 では、オプションファイルに、値を必要とするオプションを値なしで指定すると、サーバーがエラーで中止します。my.cnf に次の内容が含まれるとします。

```
[mysqld]
log_error
relay_log
relay_log_index
```

これは、次に示すようにサーバーが起動に失敗します。

```
shell> mysqld_safe &

090514 09:48:39 mysqld_safe Logging to '/home/jon/bin/mysql/var/tonfisk.err'.
090514 09:48:39 mysqld_safe Starting mysqld daemon with databases from /home/jon/bin/mysql/var
090514 09:48:39 mysqld_safe mysqld from pid file /home/jon/bin/mysql/var/tonfisk.pid ended
```

--log-error オプションは引数を必要としません。しかし、エラーログ (指定された値がないため、デフォルトの datadir/hostname.err になります) に示されるように、--relay-log オプションには必要です。

```
shell> tail -n 3 ../var/tonfisk.err

090514 09:48:39 mysqld_safe Starting mysqld daemon with databases from /home/jon/bin/mysql/var
090514  9:48:39 [ERROR] /home/jon/bin/mysql/libexec/mysqld: option '--relay-log' requires an argument
090514  9:48:39 [ERROR] Aborting
```

これは以前の動作からは変更されています。以前は、サーバーは my.cnf ファイルの最後の 2 行を --relay-log=relay_log_index と解釈し、「relay_log_index」 をベース名としてリレーログファイルを作成しました。(Bug #25192)

4.2.10 環境変数の設定

環境変数は、コマンドプロセッサの現在の起動に影響するようにコマンドプロンプトで設定したり、将来の起動に影響するように永続的に設定したりできます。変数を永続的に設定するには、起動ファイルで設定するか、またはこのためにシステムが提供するインタフェースを使用して設定できます。具体的な詳細は、コマンドインタプリタのドキュメントを参照してください。セクション2.12「環境変数」には、MySQL プログラムの動作に影響するすべての環境変数のリストがあります。

環境変数の値を指定するには、コマンドプロセッサに適した構文を使用します。たとえば Windows では、USER 変数を設定して MySQL アカウント名を指定できます。そうするには、次の構文を使用します。

```
SET USER=your_name
```

Unix での構文はシェルに依存します。MYSQL_TCP_PORT 変数を使用して TCP/IP ポート番号を指定するとします。(sh、ksh、bash、zsh、などの) 典型的な構文は次のとおりです。

```
MYSQL_TCP_PORT=3306
export MYSQL_TCP_PORT
```

最初のコマンドが変数を設定し、その値が MySQL およびほかのプロセスにアクセス可能になるように、export コマンドがシェル環境に変数をエクスポートします。

csh および tcsh については、シェル変数を環境で使用可能にするために setenv を使用してください。

```
setenv MYSQL_TCP_PORT 3306
```

環境変数を設定するコマンドは、コマンドプロンプトで実行してただちに有効にできますが、その設定はログアウトするまでしか持続しません。ログインするたびに設定を有効にするには、システムが提供するインタフェースを使用するか、コマンドインタプリタが、起動時に毎回読み取る起動ファイルに適切なコマンドを配置します。

Windows では、システム コントロール パネル (詳細設定の下) で環境変数を設定できます。

Unix では、通常のシェル起動ファイルは bash では .bashrc または .bash_profile、または tcsh では .tcshrc です。

MySQL プログラムが /usr/local/mysql/bin にインストールされ、これらのプログラムを呼び出しやすくしたいとします。そのためには、PATH 環境変数の値をそのディレクトリを含むように設定します。たとえば、シェルが bash の場合、.bashrc ファイルに次の行を追加します。

```
PATH=${PATH}:/usr/local/mysql/bin
```

bash はログインシェルと非ログインシェルで異なる起動ファイルを使用するため、ログインシェルに対しては設定を .bashrc、非ログインシェルに対しては .bash_profile に追加して、いずれの場合にも確実に PATH が設定されるようにするとよいでしょう。

シェルが tcsh の場合、.tcshrc ファイルに次の行を追加します。

```
setenv PATH ${PATH}:/usr/local/mysql/bin
```

ホームディレクトリに適切な起動ファイルが存在しない場合、テキストエディタで作成します。

PATH 設定を変更したら、設定が反映されるように、Windows の場合は新しいコンソールウィンドウを開き、Unix の場合はログインしなおします。

4.3 MySQL サーバーとサーバー起動プログラム

このセクションでは、MySQL サーバー mysqld、およびサーバーを起動するために使用されるいくつかのプログラムについて説明します。

4.3.1 mysqld — MySQL サーバー

mysqld は、MySQL サーバーとも呼ばれ、MySQL インストールでのほとんどの作業を実行するメインプログラムです。MySQL サーバーは、データベースおよびテーブルを含む MySQL データディレクトリへのアクセスを管理します。データディレクトリは、ログファイルおよびステータスファイルなど、その他の情報のデフォルトの場所でもあります。

MySQL サーバーが起動するとき、クライアントプログラムからのネットワーク接続を待機し、それらのクライアントに代わってデータベースへのアクセスを管理します。

mysqld プログラムには、起動時に指定できる多くのオプションがあります。すべてのオプションをリストするには、次のコマンドを実行します。

```
shell> mysqld --verbose --help
```

MySQL サーバーには、稼働時の動作に影響する一連のシステム変数もあります。システム変数はサーバーの起動時に設定でき、多くは実行時に変更して動的なサーバー再構成に影響を与えることができます。MySQL サーバーには、動作に関する情報を提供する一連のステータス変数もあります。これらのステータス変数をモニターして、実行時のパフォーマンス特性にアクセスできます。

MySQL サーバーのコマンドオプション、システム変数、およびステータス変数の詳細な説明は、セクション 5.1「MySQL Server」を参照してください。MySQL のインストールおよび初期構成のセットアップについては、第2章「MySQL のインストールと更新」を参照してください。

4.3.2 mysqld_safe — MySQL サーバー起動スクリプト

mysqld_safe は、Unix で mysqld サーバーを起動するための推奨される方法です。mysqld_safe は、エラー発生時の再起動やランタイム情報のエラーログファイルへのロギングなど、いくつかの安全機能を追加します。エラーのロギングについては、このセクションで後ほど説明します。

mysqld_safe は、mysqld という名前の実行可能ファイルを起動しようとします。デフォルトの動作をオーバーライドして、起動するサーバーの名前を明示的に指定するには、mysqld_safe で --mysqld オプションまたは --mysqld-version オプションを指定します。--ledir オプションを使用して、mysqld_safe がサーバーを検索するディレクトリを指定することもできます。

mysqld_safe のオプションの多くは、mysqld のオプションと同じです。セクション5.1.3「サーバーコマンドオプション」を参照してください。

mysqld_safe が理解できないオプションは、コマンド行で指定された場合は mysqld に渡されますが、オプションファイルの [mysqld_safe] グループに指定された場合は無視されます。セクション4.2.6「オプションファイルの使用」を参照してください。

mysqld_safe は、オプションファイルの [mysqld]、[server]、[mysqld_safe] の各セクションからすべてのオプションを読み取ります。たとえば、[mysqld] セクションを次のように指定すると、mysqld_safe は --log-error オプションを検出して使用します。

```
[mysqld]
log-error=error.log
```

下位互換性のため、mysqld_safe は [safe_mysqld] セクションも読み取りますが、[safe_mysqld] セクションを [mysqld_safe] セクションに名前変更することが推奨されます。

mysqld_safe は次のオプションをサポートします。また、オプションファイルを読み取り、セクション4.2.7「オプションファイルの処理に影響するコマンド行オプション」に説明されている、それらを処理するためのオプションもサポートします。

表 4.1 mysqld_safe のオプション

オプション名	説明
--basedir	MySQL インストールディレクトリへのパス
--core-file-size	mysqld が作成できるべきコアファイルのサイズ
--datadir	データディレクトリへのパス
--defaults-extra-file	通常のオプションファイルに加えてオプションファイルを読み取る
--defaults-file	指名されたオプションファイルのみを読み取る
--help	ヘルプメッセージを表示して終了
--ledir	サーバーが置かれているディレクトリへのパス
--log-error	指定されたファイルにエラーログを書き込み
--malloc-lib	mysqld で使用する代替 malloc ライブラリ
--mysqld	起動するサーバープログラム名 (ledir ディレクトリの)
--mysqld-version	サーバープログラム名のサフィクス
--nice	サーバーのスケジュール設定の優先順位を設定するために nice プログラムを使用
--no-defaults	オプションファイルを読み取らない
--open-files-limit	mysqld が開くことができるファイル数
--pid-file	プロセス ID ファイルのパス名
--plugin-dir	プラグインがインストールされているディレクトリ
--port	TCP/IP 接続を待機するポート番号
--skip-kill-mysqld	迷子の mysqld プロセスの kill を試行しない
--skip-syslog	syslog にエラーメッセージを書き込まず、エラーログファイルを使用
--socket	Unix ソケット接続を待機するソケットファイル
--syslog	syslog にエラーメッセージを書き込み

オプション名	説明
--syslog-tag	syslog に書き込まれるメッセージのタグサフィクス
--timezone	TZ タイムゾーン環境変数を指定された値に設定
--user	mysqld を、名前が user_name または数値のユーザー ID が user_id であるユーザーとして実行

- --help

  ヘルプメッセージを表示して終了します。

- --basedir=path

  MySQL インストールディレクトリへのパス。

- --core-file-size=size

  mysqld で作成できるべきコアファイルのサイズ。このオプション値は ulimit -c に渡されます。

- --datadir=path

  データディレクトリへのパス。

- --defaults-extra-file=path

  通常のオプションファイルに加えて読み取られるオプションファイルの名前。これを使用する場合は、コマンド行の最初のオプションでなければなりません。ファイルが存在しない場合やその他の理由でアクセスできない場合は、サーバーはエラーで終了します。

- --defaults-file=file_name

  通常のオプションファイルの代わりに読み取られるオプションファイルの名前。これを使用する場合は、コマンド行の最初のオプションでなければなりません。

- --ledir=path

  mysqld_safe がサーバーを検出できない場合、このオプションを使用してサーバーが置かれているディレクトリのパス名を示します。

- --log-error=file_name

  指定されたファイルにエラーログを書き込みます。セクション5.2.2「エラーログ」 を参照してください。

- --malloc-lib=[lib_name]

  システムの malloc() ライブラリの代わりに、メモリー割り当てに使用されるライブラリの名前。パス名を指定することによって任意のライブラリを使用できますが、MySQL 5.6 では Linux 向けのバイナリの MySQL 配布に同梱の tcmalloc ライブラリを使用できるようにするためのショートカット形式があります。特定の構成ではショートカット形式が機能しない可能性があります。その場合は代わりにパス名を指定するようにしてください。

  --malloc-lib オプションは、mysqld が起動するときに、ダイナミックリンクに影響を与え、ローダーがメモリー割り当てライブラリを検出できるように、LD_PRELOAD 環境値を変更します。

  - このオプションが指定されない場合、または値なしで指定された場合 (--malloc-lib=)、LD_PRELOAD は変更されず、tcmalloc を使用する試みは行われません。
  - オプションが、--malloc-lib=tcmalloc と指定された場合、mysqld_safe は tcmalloc ライブラリを /usr/lib から検索し、次に MySQL pkglibdir の場所 (たとえば、/usr/local/mysql/lib または適切なもの) から検索します。tmalloc が検出された場合は、そのパス名が LD_PRELOAD 値の先頭に追加されて mysqld に渡されます。tcmalloc が見つからない場合は、mysqld_safe は エラーで中止されます。
  - オプションが --malloc-lib=/path/to/some/library と指定された場合、そのフルパスが LD_PRELOAD 値の先頭に追加されます。そのフルパスが、存在しないかまたは読み取り不能なファイルを指している場合は、mysqld_safe は エラーで中止されます。
  - mysqld_safe がパス名を LD_PRELOAD に追加する場合は、その変数がすでに持っている、既存の値の先頭にパスを追加します。

Linux ユーザーは、次の行を my.cnf ファイルに追加することにより、バイナリパッケージに含まれる libtcmalloc_minimal.so を使用できます。

```
[mysqld_safe]
malloc-lib=tcmalloc
```

任意のプラットフォームで tcmalloc パッケージを /usr/lib にインストールしたユーザーにも、これらの行は十分です。特定の tcmalloc ライブラリを使用するには、そのフルパス名を指定します。例:

```
[mysqld_safe]
malloc-lib=/opt/lib/libtcmalloc_minimal.so
```

- --mysqld=prog_name

  ledir ディレクトリにある、起動するサーバープログラムの名前。MySQL のバイナリ配布を使用するが、バイナリ配布以外のデータディレクトリがある場合には、このオプションが必要です。mysqld_safe がサーバーを検出できない場合は、--ledir オプションを使用してサーバーのあるディレクトリへのパス名を示します。

- --mysqld-version=suffix

  このオプションは --mysqld オプションと類似していますが、サーバーのプログラム名のサフィクスだけを指定します。ベース名は mysqld であるとみなされます。たとえば、--mysqld-version=debug を使用すると、mysqld_safe は ledir ディレクトリの mysqld-debug プログラムを起動します。--mysqld-version の引数が空の場合、mysqld_safe は ledir ディレクトリの mysqld を使用します。

- --nice=priority

  サーバーのスケジュール優先順位を任意の値に設定するには、nice プログラムを使用します。

- --no-defaults

  オプションファイルを読み取りません。これを使用する場合は、コマンド行の最初のオプションでなければなりません。

- --open-files-limit=count

  mysqld が開くことができるファイル数。オプション値は ulimit -n に渡されます。

  > **注記**
  >
  > これが正しく機能するためには、mysqld_safe を root として起動する必要があります。

- --pid-file=file_name

  プロセス ID ファイルのパス名。

- --plugin-dir=path

  プラグインディレクトリのパス名。

- --port=port_num

  サーバーが TCP/IP 接続を待機するときに使用するポート番号。サーバーが root システムユーザーで開始されている場合を除き、ポート番号は 1024 以上にする必要があります。

- --skip-kill-mysqld

  起動時に、未処理の mysqld プロセスの強制終了を試行しません。このオプションは、Linux のみで機能します。

- --socket=path

  サーバーがローカル接続を待機するときに使用する Unix ソケットファイル。

- --syslog, --skip-syslog

  --syslog は、logger プログラムをサポートするシステムで、エラーメッセージが syslog に送信されるようにします。--skip-syslog は、syslog の使用を抑制し、メッセージはエラーログファイルに書き込まれます。

syslog が使用される場合、daemon.err syslog 機能/重要度がすべてのログメッセージで使用されます。

- --syslog-tag=tag

  syslog への口ギングで、mysqld_safe および mysqld からのメッセージはそれぞれ mysqld_safe および mysqld 識別子を付けて書き込まれます。識別子のサフィクスを指定するには、--syslog-tag=tag を使用します。これにより、識別子は mysqld_safe-tag および mysqld-tag に変更されます。

- --timezone=timezone

  TZ タイムゾーン環境変数を、指定されたオプション値に設定します。正当なタイムゾーン指定形式は、オペレーティングシステムのドキュメントを参照してください。

- --user={user_name|user_id}

  mysqld サーバーを、名前 user_name または数字ユーザー ID user_id を持つユーザーとして実行します。(このコンテキストでの「ユーザー」は、システムログインアカウントであり、付与テーブルにリストされている MySQL ユーザーではありません。)

mysqld_safe を --defaults-file オプションまたは --defaults-extra-file オプションで起動してオプションファイルを指定する場合、そのオプションはコマンド行で指定されるうちの最初のものでなければなりません。そうでない場合は、オプションファイルは使用されません。たとえば、次のコマンドは指定されたオプションファイルを使用しません。

```
mysql> mysqld_safe --port=port_num --defaults-file=file_name
```

代わりに、次のコマンドを使用します。

```
mysql> mysqld_safe --defaults-file=file_name --port=port_num
```

MySQL のソースまたはバイナリ配布は通常、サーバーを若干異なる場所にインストールしますが、mysqld_safe スクリプトは、どちらからインストールしたサーバーでも正常に立ち上げることができるように作成されています。(セクション2.1.5「インストールのレイアウト」を参照してください。) mysqld_safe は、次の条件のいずれかが満たされていることを想定しています。

- サーバーとデータベースは作業ディレクトリ (mysqld_safe が呼び出されたディレクトリ) から相対的に検索できること。バイナリ配布の場合、mysqld_safe は作業ディレクトリの下の bin ディレクトリおよび data ディレクトリを検索します。ソース配布の場合は、libexec ディレクトリおよび var ディレクトリを検索します。mysqld_safe を MySQL インストールディレクトリ (バイナリ配布の場合は /usr/local/mysql) から起動した場合には、この条件が満たされるはずです。
- サーバーとデータベースが作業ディレクトリから相対的に検出できない場合は、mysqld_safe は絶対パス名での検索を試みます。通常の場所は /usr/local/libexec および /usr/local/var です。実際の場所は、配布のビルド時に構成される値から決定されます。構成時に指定された場所に MySQL がインストールされていれば、これは正しいはずです。

mysqld_safe はサーバーおよびデータベースを作業ディレクトリから相対的に検索しようとするため、mysqld_safe を MySQL インストールディレクトリから起動するかぎり、MySQL のバイナリ配布は任意の場所にインストールできます。

```
shell> cd mysql_installation_directory
shell> bin/mysqld_safe &
```

mysqld_safe を MySQL インストールディレクトリから呼び出しても失敗する場合は、--ledir オプションおよび --datadir オプションを指定して、システムのサーバーとデータベースがあるディレクトリを指定します。

MySQL 5.6.5 以降では、mysqld_safe は sleep および date システムユーティリティーを使用して、今回の起動までに何回起動しようとしたかを判断します。これらが存在し、5 回より多い場合は、再起動するまでに強制的に 1 秒間待機させられます。これは、連続して失敗する場合に過度に CPU を使用することを防ぐためのものです。(Bug #11761530、Bug #54035)

mysqld_safe を使用して mysqld を起動する場合、mysqld_safe は自身と mysqld からのエラー (および通知) メッセージが同じ出力先に送信されるよう手配します。

これらのメッセージの出力先を制御するための mysqld_safe オプションがいくつかあります。

- --log-error=file_name: エラーメッセージを指定されたエラーファイルに書き込みます。
- --syslog: logger プログラムをサポートするシステムで、syslog にエラーメッセージを書き込みます。

- --skip-syslog: エラーメッセージを syslog に書き込みません。メッセージは、デフォルトのエラーログファイル (データディレクトリの host_name.err)、または --log-error オプションが指定された場合は、指定されたファイルに書き込まれます。

これらのオプションが指定されていない場合は、デフォルトは --skip-syslog です。

--log-error と --syslog が両方指定された場合は、警告が発行され --log-error が優先されます。

mysqld_safe がメッセージを書き込む場合、通知はロギングの出力先 (syslog またはエラーログファイル) および stdout に送られます。エラーはロギングの出力先と stderr に送られます。

4.3.3 mysql.server — MySQL サーバー起動スクリプト

Unix の MySQL 配布には、mysql.server というスクリプトがあります。System V スタイルの実行ディレクトリを使用してシステムサービスの起動および停止を行う、Linux および Solaris などのシステムで使用できます。MySQL の OS X Startup Item でも使用されます。

mysql.server は、MySQL インストールディレクトリの support-files または MySQL のソース配布にあります。

Linux の RPM パッケージ (MySQL-server-VERSION.rpm) を使用する場合は、mysql.server スクリプトは /etc/init.d ディレクトリに mysql という名前でインストールされます。それを手動でインストールする必要はありません。Linux RPM パッケージに関する詳細は、セクション2.5.5「RPM パッケージを使用して MySQL を Linux にインストールする」を参照してください。

ベンダーによっては、起動スクリプトを mysqld のような別名でインストールする RPM パッケージを提供している場合もあります。

MySQL をソース配布から、あるいは mysql.server を自動的にインストールしないバイナリの配布形式を使用してインストールする場合、それを手動でインストールできます。手順はセクション2.10.1.2「MySQL を自動的に起動および停止する」を参照してください。

mysql.server は、オプションファイルの [mysql.server] セクションおよび [mysqld] セクションからオプションを読み取ります。下位互換性のため、[mysql_server] セクションも読み取りますが、MySQL 5.6 を使用する場合は、このようなセクションは [mysql.server] に名前変更するようにしてください。

mysql.server は次のオプションをサポートします。

- --basedir=path

  MySQL インストールディレクトリへのパス。

- --datadir=path

  MySQL データディレクトリへのパス。

- --pid-file=file_name

  サーバーがプロセス ID を書き込むべきファイルのパス名。

- --service-startup-timeout=seconds

  サーバーの起動を確認するために待機する秒数。サーバーがこの時間内に起動しない場合、mysql.server は エラーで終了します。デフォルト値は 900 です。0 の値は、起動をまったく待機しないことを意味します。負の値はいつまでも (タイムアウトなしで) 待機することを意味します。

- --use-mysqld_safe

  mysqld_safe を使用してサーバーを起動します。これはデフォルトです。

- --user=user_name

  mysqld を実行する際に使用するログインユーザー名。

4.3.4 mysqld_multi — 複数の MySQL サーバーの管理

mysqld_multi は、Unix ソケットファイルや TCP/IP ポートでの接続を待機する複数の mysqld プロセスを管理するためのものです。サーバーの起動または停止、現在のステータスのレポートを実行できます。

mysqld_multi は [mysqldN] という名前のグループを my.cnf (または --defaults-file オプションで指定されたファイル) から検索します。N は任意の正の整数です。この数字は、次の説明ではオプショングループ番号、または GNR といいます。グループ番号は、それぞれのオプショングループを識別し、起動、停止、またはステータスレポート取得の対象となるサーバーを指定するために、mysqld_multi の引数として使用されます。これらのグループにリストされるオプションは、mysqld を起動するために [mysqld] グループで使用するものと同じです。(たとえばセクション2.10.1.2「MySQL を自動的に起動および停止する」などを参照してください。)ただし、複数のサーバーを使用する場合は、Unix ソケットファイルや TCP/IP ポート番号などに関してそれぞれが独自のオプション値を使用する必要があります。複数サーバーの環境で、どのオプションを一意とする必要があるのかについては、セクション5.3「1 つのマシン上での複数の MySQL インスタンスの実行」を参照してください。

mysqld_multi を呼び出すには、次の構文を使用します。

```
shell> mysqld_multi [options] {start|stop|reload|report} [GNR[,GNR] ...]
```

start、stop、reload (停止して再起動)、および report は、実行する操作を示します (reload は MySQL 5.6.3 で使用可能です。)オプション名に続く GNR リストに従って、指定した操作を単一のサーバーまたは複数のサーバーで実行できます。リストがない場合は、mysqld_multi はオプションファイル内のすべてのサーバーに対して操作を実行します。

各 GNR 値は、オプショングループ番号またはグループ番号の範囲を表します。値は、オプションファイル内のグループ名の最後の数字とします。たとえば、[mysqld17] という名前のグループの GNR は 17 です。番号の範囲を指定するには、最初と最後の番号をダッシュで区切ります。GNR 値 10-13 は、[mysqld10] から [mysqld13] のグループを表します。コマンド行で、複数のグループまたはグループの範囲を、カンマで区切って指定できます。GNR リストには空白文字 (スペースまたはタブ) を使用してはいけません。空白文字からあとにあるものはすべて無視されます。

次のコマンドは [mysqld17] というオプショングループを使用して単一のサーバーを起動します。

```
shell> mysqld_multi start 17
```

次のコマンドは、[mysqld8] および [mysqld10] から [mysqld13] までのオプショングループを使用して複数のサーバーを停止します。

```
shell> mysqld_multi stop 8,10-13
```

オプションファイルをセットアップする方法の例として、次のコマンドを使用します。

```
shell> mysqld_multi --example
```

mysqld_multi は次のようにオプションファイルを検索します。

- --no-defaults では、オプションファイルは読み取りません。
- --defaults-file=file_name では、指定されたファイルのみを読み取ります。
- それ以外の場合は、--defaults-extra-file=file_name オプションが指定された場合はそこで指定されたファイルを含めて、標準の場所のリストにあるオプションファイルが読み取られます。(オプションが複数回指定された場合は、最後の値が使用されます。)

読み取られるオプションファイルでは、[mysqld_multi] および [mysqldN] オプショングループが検索されます。[mysqld_multi] グループは、mysqld_multi 自身へのオプションに使用できます。[mysqldN] グループは、mysqld の特定のインスタンスに渡されるオプションに使用できます。

[mysqld] グループまたは [mysqld_safe] グループは、mysqld または mysqld_safe のすべてのインスタンスによって読み取られる共通オプションに使用できます。--defaults-file=file_name オプションを指定して、そのインスタンスに異なる構成ファイルを使用できます。この場合、そのファイルの[mysqld] または [mysqld_safe] グループがそのインスタンスに使用されます。

mysqld_multi は次のオプションをサポートします。

- --help

  ヘルプメッセージを表示して終了します。

- --example

  サンプルのオプションファイルを表示します。

- --log=file_name

ログファイルの名前を指定します。ファイルが存在する場合は、ログ出力はそこに追加されます。

- --mysqladmin=prog_name

  サーバーの停止に使用する mysqladmin バイナリ。

- --mysqld=prog_name

  使用する mysqld バイナリ。このオプションの値として mysqld_safe を指定できます。mysqld_safe を使用してサーバーを起動する場合は、対応する [mysqldN] オプショングループに mysqld オプションまたは ledir オプションを含めることができます。これらのオプションは、mysqld_safe が起動するべきサーバーの名前と、サーバーがあるディレクトリのパス名を示します。(これらのオプションに関する説明は、セクション 4.3.2「mysqld_safe — MySQL サーバー起動スクリプト」を参照してください。)例:

```
[mysqld38]
mysqld = mysqld-debug
ledir  = /opt/local/mysql/libexec
```

- --no-log

  ログファイルではなく、stdout にログを出力します。デフォルトでは、出力はログファイルに送られます。

- --password=password

  mysqladmin を呼び出すときに使う MySQL アカウントのパスワード。ほかの MySQL プログラムとは異なり、このオプションではパスワード値はオプションではありません。

- --silent

  サイレントモード。警告を無効にします。

- --tcp-ip

  Unix ソケットファイルではなく TCP/IP ポートを介して各 MySQL サーバーに接続します。(ソケットファイルがない場合でもサーバーは稼働している可能性がありますが、TCP/IP ポートからのみアクセスできます。)接続はデフォルトでは Unix ソケットファイルを使用して行われます。このオプションは stop 操作と report 操作に影響します。

- --user=user_name

  mysqladmin を呼び出すときに使う MySQL アカウントのユーザー名。

- --verbose

  より詳細になります。

- --version

  バージョン情報を表示して終了します。

mysqld_multi に関する注意:

- もっとも重要: mysqld_multi を使用する前に、mysqld サーバーに渡されるオプションの意味と、なぜ独立した mysqld プロセスが必要なのかを確実に理解してください。同じデータディレクトリで複数の mysqld サーバーを使用することの危険性に注意してください。特別な意図がないかぎり、独立のデータディレクトリを使用してください。スレッドを使用するシステムでは、同じデータディレクトリで複数のサーバーを起動してもパフォーマンスは改善されません。セクション5.3「1 つのマシン上での複数の MySQL インスタンスの実行」 を参照してください。

- > **重要**
  >
  > 各サーバーのデータディレクトリが、その特定の mysqld を開始した Unix アカウントから完全にアクセス可能であることを確認してください。特別な意図がないかぎり、Unix root アカウントをこれに使用しないでください。セクション6.1.5「MySQL を通常ユーザーとして実行する方法」を参照してください。

- mysqld サーバーを (mysqladmin プログラムで) 停止するのに使用する MySQL アカウントが、各サーバーに対して同じユーザー名とパスワードを持つことを確認してください。また、そのアカウントには SHUTDOWN 権限があることも確かめてください。管理対象のサーバーの管理アカウントのユーザー名またはパスワードが異

なる場合は、各サーバーに同じユーザー名とパスワードを持つアカウントを作成するとよいでしょう。たとえば、次のコマンドをそれぞれのサーバーで実行することにより、共通の multi_admin アカウントをセットアップできます。

```
shell> mysql -u root -S /tmp/mysql.sock -p
Enter password:
mysql> GRANT SHUTDOWN ON *.*
    -> TO 'multi_admin'@'localhost' IDENTIFIED BY 'multipass';
```

セクション6.2「MySQL アクセス権限システム」を参照してください。これは、それぞれの mysqld サーバーで行う必要があります。それぞれに接続する場合、接続パラメータを適切に変更します。アカウント名のホスト名部分は、 mysqld_multi を実行するホストから multi_admin として接続できるものでなければなりません。

- Unix ソケットファイルと TCP/IP ポート番号は、すべての mysqld で異なる必要があります。(または、ホストが複数のネットワークアドレスを持つ場合、--bind-address を使用して、異なるサーバーが異なるインタフェースを待機するようにできます。)

- mysqld_safe を使用して mysqld を起動している場合 (たとえば --mysqld=mysqld_safe)、--pid-file オプションは非常に重要です。すべての mysqld が独自のプロセス ID ファイルを持っているべきです。mysqld ではなく、mysqld_safe を使用することの利点は、mysqld_safe は mysqld のプロセスをモニターして、kill -9 を使用したシグナル送信や、セグメンテーション違反などその他の原因でプロセスが終了した場合に、再起動するということです。mysqld_safe スクリプトでは、特定の場所から起動することが必要な場合があります。これは、mysqld_multi を実行する前に、特定のディレクトリに場所を変更しなければならない可能性があることを意味します。起動時に問題がある場合は、mysqld_safe スクリプトを調べてください。特に次の行をチェックします。

  ```
  ----------------------------------------------------------------------
  MY_PWD=`pwd`
  # Check if we are starting this relative (for the binary release)
  if test -d $MY_PWD/data/mysql -a \
    -f ./share/mysql/english/errmsg.sys -a \
    -x ./bin/mysqld
  ----------------------------------------------------------------------
  ```

  これらの行によって実行されるテストは成功するはずです。そうでないと問題が生じます。セクション4.3.2「mysqld_safe — MySQL サーバー起動スクリプト」を参照してください。

- --user オプションを mysqld に対して使用する場合がありますが、そのためには mysqld_multi スクリプトを Unix のスーパーユーザー (root) として実行する必要があります。このオプションがオプションファイルにあるかどうかは問題ではなく、もしスーパーユーザーではない人が、mysqld プロセスを自分の Unix アカウントで起動すると、警告が出ます。

次の例は、mysqld_multi とともに使用するオプションファイルの設定方法を示します。mysqld プログラムが起動または終了する順序は、オプションファイルで指定する順序によります。グループ番号は、切れ目のないシーケンスの形式にする必要はありません。例では、最初と 5 番目の [mysqldN] グループは意図的に省略しています。これは、オプションファイルで「ギャップ」があっても構わないことを示しています。これにより、柔軟性が高まります。

```
# This file should probably be in your home dir (~/.my.cnf)
# or /etc/my.cnf
# Version 2.1 by Jani Tolonen

[mysqld_multi]
mysqld     = /usr/local/bin/mysqld_safe
mysqladmin = /usr/local/bin/mysqladmin
user       = multi_admin
password   = multipass

[mysqld2]
socket     = /tmp/mysql.sock2
port       = 3307
pid-file   = /usr/local/mysql/var2/hostname.pid2
datadir    = /usr/local/mysql/var2
language   = /usr/local/share/mysql/english
user       = john

[mysqld3]
socket     = /tmp/mysql.sock3
port       = 3308
pid-file   = /usr/local/mysql/var3/hostname.pid3
datadir    = /usr/local/mysql/var3
language   = /usr/local/share/mysql/swedish
user       = monty
```

```
[mysqld4]
socket     = /tmp/mysql.sock4
port       = 3309
pid-file   = /usr/local/mysql/var4/hostname.pid4
datadir    = /usr/local/mysql/var4
language   = /usr/local/share/mysql/estonia
user       = tonu

[mysqld6]
socket     = /tmp/mysql.sock6
port       = 3311
pid-file   = /usr/local/mysql/var6/hostname.pid6
datadir    = /usr/local/mysql/var6
language   = /usr/local/share/mysql/japanese
user       = jani
```

セクション4.2.6「オプションファイルの使用」を参照してください。

4.4 MySQL インストール関連のプログラム

このセクションのプログラムは、MySQL のインストールまたはアップグレードに使用されます。

4.4.1 comp_err — MySQL エラーメッセージファイルのコンパイル

comp_err は、mysqld がさまざまなエラーコードに対して表示するエラーメッセージを判断するために使用する、errmsg.sys ファイルを作成します。comp_err は、通常 MySQL のビルド時に自動的に実行されます。MySQL ソース配布の sql/share/errmsg.txt にあるプレーンテキストファイルから errmsg.sys ファイルをコンパイルします。

comp_err は、mysqld_error.h、mysqld_ername.h、および sql_state.h の各ヘッダーファイルも生成します。

エラーメッセージの定義方法の詳細は、「MySQL Internals Manual」を参照してください。

comp_err は次のように呼び出します。

```
shell> comp_err [options]
```

comp_err は次のオプションをサポートします。

- --help, -?

  ヘルプメッセージを表示して終了します。

- --charset=path, -C path

  文字セットディレクトリ。デフォルトは ../sql/share/charsets です。

- --debug=debug_options, -# debug_options

  デバッグのログを書き込みます。一般的な debug_options 文字列は d:t:O,file_name です。デフォルトは d:t:O,/tmp/comp_err.trace です。

- --debug-info, -T

  プログラムの終了時に、デバッグ情報を出力します。

- --header_file=file_name, -H file_name

  エラーヘッダーファイルの名前です。デフォルトは mysqld_error.h です。

- --in_file=file_name, -F file_name

  入力ファイルの名前です。デフォルトは ../sql/share/errmsg.txt です。

- --name_file=file_name, -N file_name

  エラー名ファイルの名前です。デフォルトは mysqld_ername.h です。

- --out_dir=path, -D path

  出力ベースディレクトリの名前です。デフォルトは ../sql/share/ です。

- --out_file=file_name, -O file_name

  出力ファイルの名前です。デフォルトは errmsg.sys です。

- --statefile=file_name, -S file_name

  SQLSTATE ヘッダーファイルの名前です。デフォルトは sql_state.h です。

- --version, -V

  バージョン情報を表示して終了します。

4.4.2 mysqlbug — バグレポートの生成

このプログラムは非推奨です。これは MySQL 5.6.19 以降で非推奨となり、MySQL 5.7 で削除されています。

通常バグをレポートする場合は、http://bugs.mysql.com/ にアクセスしてください。これはバグデータベースのアドレスです。このバグデータベースは一般に公開されているので、だれでも参照および検索することができます。システムにログインすると、新しいレポートを入力できます。

4.4.3 mysql_install_db — MySQL データディレクトリの初期化

mysql_install_db は MySQL データディレクトリを初期化し、システムテーブルを作成します (システムテーブルがない場合)。また、InnoDB テーブルの管理に必要な、システムのテーブルスペースおよび関係するデータ構造体も初期化します。MySQL 5.6.8 では、mysql_install_db は Perl スクリプトで、Perl がインストールされた任意のシステムで使用できます。5.6.8 より前ではシェルスクリプトで、Unix プラットフォームでのみ使用可能です。

MySQL 5.6.8 以降では、mysql_install_db は UNIX プラットフォーム上で、my.cnf という名前のデフォルトオプションファイルを基本インストールディレクトリに作成します。このファイルは my-default.cnf という名前の配布パッケージに含まれるテンプレートから作成されます。テンプレートは基本インストールディレクトリ内またはその配下から見つけることができます。mysqld_safe を使用して開始すると、サーバーはデフォルトで my.cnf ファイルを使用します。my.cnf がすでに存在する場合、mysql_install_db はそのファイルが使用中だと認識し、my-new.cnf という名前の新しいファイルを代わりに書き込みます。

1 つの例外を除き、デフォルトのオプションファイル内の設定はコメント化されて無効になっています。例外は、このファイルが sql_mode システム変数を NO_ENGINE_SUBSTITUTION,STRICT_TRANS_TABLES に変更することです。この設定により、トランザクションテーブルを変更する操作での不良データに対して警告でなくエラーを発生させるサーバー構成が提供されます。セクション5.1.7「サーバー SQL モード」を参照してください。

mysql_install_db を呼び出すには、次の構文を使用します。

```
shell> mysql_install_db [options]
```

MySQL サーバー mysqld があとで起動されるとき、データディレクトリにアクセスしなければならないため、mysqld の起動に使用するのと同じシステムアカウントから mysql_install_db を起動するか、または root として実行し、--user オプションを指定して mysqld を実行するユーザー名を指示します。mysql_install_db がインストールディレクトリまたはデータディレクトリの正しい場所を使用しない場合は、--basedir または --datadir などの、ほかのオプションを指定しなければならない場合があります。例:

```
shell> scripts/mysql_install_db --user=mysql \
       --basedir=/opt/mysql/mysql \
       --datadir=/opt/mysql/mysql/data
```

> 注記
>
> mysql_install_db が InnoDB システムテーブルスペース をセットアップしたあと、テーブルスペースの特性を変更するにはまったく新しいインスタンスをセットアップする必要があります。これには、システムテーブルスペース内の最初のファイル名および undo ログの数が含まれます。デフォルト値を使用しない場合は、mysql_install_db を実行する前に、innodb_data_file_path および innodb_log_file_size の各構成パラメータの設定が MySQL 構成ファイル内で適切な場所にあることを確認してください。また、innodb_data_home_dir および innodb_log_group_home_dir などの、InnoDB ファイルの作成および場所に影響するその他のパラメータを、必要に応じて指定してください。
>
> これらのオプションが構成ファイルにあるが、そのファイルが MySQL がデフォルトで読み取る場所にない場合は、mysql_install_db の実行時に --defaults-extra-file オプションを使用してファイルの場所を指定します。

> **注記**
>
> インストールを実行する際にカスタムの TMPDIR 環境変数を設定し、指定されたディレクトリにアクセスできない場合、mysql_install_db は失敗する場合があります。その場合は、TMPDIR の設定を取り消すか、または TMPDIR をシステムの一時ディレクトリ (通常 /tmp) を指すように設定します。

mysql_install_db は次のオプションをサポートします。これらはコマンド行またはオプションファイルの [mysql_install_db] グループで指定できます。(mysqld に共通なオプションは、[mysqld] グループにも指定できます。)その他のオプションは mysqld に渡されます。MySQL プログラムによって使用されるオプションファイルの詳細については、セクション4.2.6「オプションファイルの使用」を参照してください。

表 4.2 mysql_install_db オプション

オプション名	説明	導入	非推奨
--basedir	ベースディレクトリへのパス		
--builddir	ビルドディレクトリへのパス (ソースからのビルドで)		
--cross-bootstrap	内部使用		
--datadir	データディレクトリへのパス		
--defaults-extra-file	通常のオプションファイルに加えてオプションファイルを読み取る		
--defaults-file	指名されたオプションファイルのみを読み取る		
--force	DNS が機能しない場合でも実行		
--help	ヘルプメッセージを表示して終了		
--keep-my-cnf	既存の my.cnf file を保持し、新規に作成しない	5.6.20	5.6.20
--ldata	--datadir のシノニム		
--no-defaults	オプションファイルを読み取らない		
--random-passwords	管理アカウントにランダムパスワードを生成	5.6.8	
--rpm	内部使用		
--skip-name-resolve	付与テーブルにホスト名ではなく IP アドレスを使用		
--srcdir	内部使用		
--user	mysqld を実行するシステムログインユーザー		
--verbose	冗長モード		
--windows	内部使用		

- --help

  ヘルプメッセージを表示して終了します。

- --basedir=path

  MySQL インストールディレクトリへのパス。

- --builddir=path

  --srcdir およびソースからのビルドとともに使用します。これは、ビルドされたファイルがあるディレクトリの場所にセットします。

- --cross-bootstrap

  内部使用。このオプションは、あるホスト用のシステムテーブルを別のホストでビルドするために使用されます。

- --datadir=path

  MySQL データディレクトリへのパス。MySQL 5.6.8 以降、mysql_install_db はオプション値に関してより厳密になりました。パスが存在しない場合は、パス名の最後のコンポーネントのみが作成されます。親ディレクトリはすでに存在する必要があります。存在しない場合はエラーが発生します。

- --defaults-extra-file=file_name

このオプションファイルは、グローバルオプションファイルのあとに読み取りますが、(UNIX では) ユーザーオプションファイルの前に読み取るようにしてください。ファイルが存在しないかアクセスできない場合、エラーが発生します。file_name は、フルパス名でなく相対パス名として指定された場合、現行ディレクトリを基準にして解釈されます。

- --defaults-file=file_name

  指定されたオプションファイルのみ使用します。ファイルが存在しないかアクセスできない場合、エラーが発生します。file_name は、フルパス名でなく相対パス名として指定された場合、現行ディレクトリを基準にして解釈されます。

- --force

  DNS が機能しない場合でも mysql_install_db を実行します。通常ホスト名を使用して作成される付与テーブルエントリが、代わりに IP アドレスを使用します。

- --keep-my-cnf

  mysql_install_db に対し、既存の my.cnf ファイルがあれば、新しいデフォルトの my.cnf ファイルを作成せずに既存のものを維持するように指示します。このオプションは MySQL 5.6.20 で追加されました。

- --ldata=path

  --datadir のシノニム。

- --no-defaults

  オプションファイルを読み取りません。オプションファイルから不明のオプションを読み取ることが原因でプログラムの起動に失敗する場合、--no-defaults を使用して、オプションを読み取らないようにすることができます。

- --random-passwords

  Unix プラットフォームで、このオプションはよりセキュアな MySQL インストールを提供します。mysql_install_db を --random-passwords で起動すると、通常の動作に加えて次のアクションを実行するようになります。

  - インストールプロセスはランダムなパスワードを作成して、それを初期 MySQL root アカウントに割り当て、これらのアカウントに対して「有効期限切れパスワード」フラグを設定します。
  - 初期のランダムな root パスワードは、HOME 環境変数によって指定されるディレクトリの .mysql_secret ファイルに書き込まれます。オペレーティングシステムによっては、sudo などのコマンドを使用するとHOME の値が root システムユーザーのホームディレクトリを参照するようになる場合があります。

    .mysql_secret がすでに存在する場合は、新しいパスワード情報はそれに追加されます。各パスワードエントリにはタイムスタンプが含まれるため、複数のインストール操作の場合に、それぞれに関連するパスワードを判断できます。

    .mysql_secret は、作成対象のシステムユーザーのみがアクセスできるように、モード 600 で作成されます。
  - 匿名ユーザー MySQL アカウントは作成されません。

  これらのアクションの結果、インストール後、サーバーを起動して .mysql_secret ファイルに書き込まれたパスワードを使用して root として接続し、新しい root パスワードを指定することが必要です。これを行うまで、root はそれ以外何もできません。これは、使用するすべての root アカウントについて実行する必要があります。パスワードを変更するには、SET PASSWORD ステートメントを (たとえば mysql クライアントとともに) 使用します。mysqladmin または mysql_secure_installation も使用できます。

  (アップグレードではなく) 新しい RPM インストール操作は、mysql_install_db を --random-passwords オプションで起動します。(Unbreakable Linux Network での RPM を使用したインストール操作は mysql_install_db を使用しないため、影響されません。)

  MySQL 5.6.9 では、(アップグレードではなく) 新しい Solaris PKG インストール操作は、mysql_install_db を --random-passwords オプションで起動します。

  バイナリの .tar.gz 配布またはソース配布を使用するインストール操作では、mysql_install_db を --random-passwords オプションで手動で起動して、MySQL インストールをよりセキュアにできます。これは、機密データのあるサイトでは特に推奨されます。

このオプションは MySQL 5.6.8 で追加されました。

- --rpm

  内部使用。このオプションは、RPM パッケージを使用して実行されるインストール操作で、MySQL のインストールプロセス中に使用されます。

- --skip-name-resolve

  付与テーブルのエントリ作成時にホスト名ではなく IP アドレスを使用します。このオプションは、DNS が機能しない場合に便利です。

- --srcdir=path

  内部使用。このオプションは、mysql_install_db がエラーメッセージファイルおよびヘルプテーブルの移入用ファイルなどのサポートファイルを検索するディレクトリを指定します。

- --user=user_name

  mysqld を実行する際に使用するシステム (ログイン) ユーザー名。mysqld によって作成されるファイルおよびディレクトリは、このユーザーが所有します。このオプションを使用するには、システム root ユーザーでなければなりません。デフォルトでは、mysqld は現在のログイン名を使用して稼働し、作成されるファイルおよびディレクトリはそのユーザーの所有となります。

- --verbose

  冗長モード。プログラムの動作についてより多くの情報を出力します。

- --windows

  内部使用。このオプションは、Windows 配布の作成に使用されます。

4.4.4 mysql_plugin — MySQL サーバープラグインの構成

mysql_plugin ユーティリティーを使用すると、MySQL 管理者は MySQL サーバーがどのプラグインをロードするかを管理できます。サーバーの起動時に手動で --plugin-load オプションを指定したり、INSTALL PLUGIN および UNINSTALL PLUGIN ステートメントを実行時に使用したりする代わりに使用できます。mysql_plugin は MySQL 5.6.3 で使用可能です。

mysql_plugin は、プラグインを有効にするために呼び出されたか無効にするために呼び出されたかによって、プラグインレジストリとして機能する mysql.plugin テーブルの行を挿入または削除します。(この操作を実行するために、mysql_plugin は MySQL サーバーをブートストラップモードで呼び出します。これは、サーバーがすでに稼働していてはならないことを意味します。)通常のサーバー起動では、サーバーは mysql.plugin にリストされているプラグインを自動的にロードして有効にします。プラグインの有効化をさらに管理するには、セクション 5.1.8.1「プラグインのインストールおよびアンインストール」で説明するように、特定のプラグインを指名する --plugin_name オプションを使用します。

mysql_plugin は、呼び出されるたびに構成ファイルを読み取り、単一のプラグインライブラリオブジェクトファイルに含まれるプラグインを構成する方法を決定します。mysql_plugin を呼び出すには、次の構文を使用します。

```
mysql_plugin [options] plugin {ENABLE|DISABLE}
```

plugin は構成するプラグインの名前です。ENABLE または DISABLE (大文字と小文字は区別されません) は、構成ファイルで指名されたプラグインライブラリのコンポーネントを有効にするか無効にするかを指定します。plugin 引数および ENABLE 引数または DISABLE 引数の順序はどちらでもかまいません。

たとえば、Linux で myplugins.so、または Windows で myplugins.dll という名前のプラグインライブラリファイルのコンポーネントを構成するには、plugin の値を myplugins に指定します。このプラグインライブラリに、plugin1、plugin2、および plugin3 という 3 つのプラグインが含まれ、これらすべてを mysql_plugin コントロールに構成するとします。規則では、構成ファイルは .ini というサフィクスおよびプラグインライブラリと同じベース名を持つため、このプラグインライブラリのデフォルト構成ファイル名は myplugins.ini です。構成ファイルの内容は次のようになります。

```
myplugins
plugin1
plugin2
```

```
plugin3
```

myplugins.ini ファイルの最初の行は、.so または .dll などの拡張子のないライブラリオブジェクトファイルの名前です。残りの行は、有効または無効にするコンポーネントの名前です。ファイル内の各値は、独立した行に記述するようにしてください。'#' の文字で始まる行は、コメントとみなされ無視されます。

構成ファイルにリストされたプラグインを有効にするには、mysql_plugin を次のように呼び出します。

```
shell> mysql_plugin myplugins ENABLE
```

プラグインを無効にするには、ENABLE ではなく DISABLE を使用します。

mysql_plugin が構成ファイルもプラグインライブラリファイルも検出できない場合、または mysql_plugin が MySQL サーバーを起動できない場合は、エラーが発生します。

mysql_plugin は次のオプションをサポートします。これらはコマンド行または任意のオプションファイルの [mysqld] グループで指定できます。[mysqld] グループに指定されるオプションに関しては、mysql_plugin は --basedir、--datadir、および --plugin-dir の各オプションを認識し、その他は無視します。MySQL プログラムによって使用されるオプションファイルの詳細については、セクション4.2.6「オプションファイルの使用」を参照してください。

表 4.3 mysql_plugin オプション

オプション名	説明
--basedir	サーバーのベースディレクトリ
--datadir	サーバーのデータディレクトリ
--help	ヘルプメッセージを表示して終了
--my-print-defaults	my_print_defaults へのパス
--mysqld	サーバーへのパス
--no-defaults	構成ファイルを読み取らない
--plugin-dir	プラグインがインストールされているディレクトリ
--plugin-ini	プラグイン構成ファイル
--print-defaults	構成ファイルのデフォルトを表示
--verbose	冗長モード
--version	バージョン情報を表示して終了

- --help, -?

  ヘルプメッセージを表示して終了します。

- --basedir=path, -b path

  サーバーのベースディレクトリ。

- --datadir=path, -d path

  サーバーのデータディレクトリ。

- --my-print-defaults=path, -b path

  my_print_defaults プログラムへのパス。

- --mysqld=path, -b path

  mysqld サーバーへのパス。

- --no-defaults, -p

  構成ファイルから値を読み取りません。このオプションにより、管理者は構成ファイルからデフォルトを読み取るのをスキップできます。

  mysql_plugin では、--no-defaults をサポートするほかのほとんどの MySQL プログラムとは異なり、このオプションをコマンド行の最初に指定する必要はありません。

- --plugin-dir=path, -p path

  サーバーのプラグインディレクトリ。

- --plugin-ini=file_name, -i file_name

  mysql_plugin 構成ファイル。相対パス名は、現在のディレクトリに相対的に解釈されます。このオプションを指定しない場合、デフォルトはプラグインディレクトリの plugin.ini です。ここで、plugin はコマンド行の plugin 引数です。

- --print-defaults, -P

  構成ファイルからのデフォルト値を表示します。このオプションを使用すると、mysql_plugin は--basedir、--datadir、および --plugin-dir のデフォルトが構成ファイルにあればそれらを出力します。変数の値が見つからない場合は何も表示されません。

  mysql_plugin では、--print-defaults をサポートするほかのほとんどの MySQL プログラムとは異なり、このオプションをコマンド行の最初に指定する必要はありません。

- --verbose, -v

  冗長モード。プログラムの動作についてより多くの情報を出力します。このオプションは情報量を増加させるために複数回使用できます。

- --version, -V

  バージョン情報を表示して終了します。

4.4.5 mysql_secure_installation — MySQL インストールのセキュリティー改善

このプログラムにより、次の方法で MySQL インストールのセキュリティーを改善できます。

- root アカウントのパスワードを設定できます。
- ローカルホスト以外からアクセス可能な root アカウントを削除できます。
- 匿名ユーザーアカウントを削除できます。
- test データベース (デフォルトでは、匿名ユーザーであっても、すべてのユーザーがアクセスできます)、および test_ で始まる名前を持つデータベースへのアクセスを許可する権限を削除できます。

mysql_secure_installation により、セクション2.10.2「最初の MySQL アカウントのセキュリティー設定」に述べられているのと同様のセキュリティーの推奨事項を実装しやすくなります。

mysql_secure_installation は引数なしで呼び出します。

```
shell> mysql_secure_installation
```

このスクリプトを実行すると、どのアクションを実行するか決定するように求められます。

4.4.6 mysql_tzinfo_to_sql — タイムゾーンテーブルのロード

mysql_tzinfo_to_sql プログラムは、mysql データベースに、タイムゾーンテーブルをロードします。zoneinfo データベース (タイムゾーンを記述するファイルのセット) があるシステムで使用します。このようなシステムには、Linux、FreeBSD、Solaris、OS X などがあります。これらのファイルの適切な場所の 1 つは /usr/share/zoneinfo ディレクトリです (Solaris では /usr/share/lib/zoneinfo)。zoneinfo データベースがないシステムの場合には、セクション10.6「MySQL Server でのタイムゾーンのサポート」で説明するダウンロード可能なパッケージを使用できます。

mysql_tzinfo_to_sql はいくつかの方法で呼び出せます。

```
shell> mysql_tzinfo_to_sql tz_dir
shell> mysql_tzinfo_to_sql tz_file tz_name
shell> mysql_tzinfo_to_sql --leap tz_file
```

最初の呼び出し構文は、zoneinfo ディレクトリのパス名を mysql_tzinfo_to_sql に渡し、出力を mysql プログラムに送信します。例:

```
shell> mysql_tzinfo_to_sql /usr/share/zoneinfo | mysql -u root mysql
```

mysql_tzinfo_to_sql は、システムのタイムゾーンファイルを読み取り、そのファイルから SQL ステートメントを生成します。mysql はこれらのステートメントを処理して、タイムゾーンテーブルをロードします。

2 番目の構文は、mysql_tzinfo_to_sql がタイムゾーン名 tz_name に対応する単一のタイムゾーンファイル tz_file をロードします。

```
shell> mysql_tzinfo_to_sql tz_file tz_name | mysql -u root mysql
```

うるう秒を考慮する必要がある場合は、3 番目の構文を使用して mysql_tzinfo_to_sql を呼び出します。これはうるう秒の情報を初期化します。tz_file はタイムゾーンファイルの名前です。

```
shell> mysql_tzinfo_to_sql --leap tz_file | mysql -u root mysql
```

mysql_tzinfo_to_sql の実行後、以前にキャッシュしたすべてのタイムゾーンデータを使用し続けないように、サーバーを再起動することをお勧めします。

4.4.7 mysql_upgrade — MySQL テーブルのチェックとアップグレード

mysql_upgrade はすべてのデータベースのすべてのテーブルに対して、MySQL サーバーの現在のバージョンとの非互換性を調べます。また、mysql_upgrade は、システムテーブルをアップグレードして、追加された可能性のある新しい権限または機能を利用できるようにします。

mysql_upgrade は、テーブルに非互換性がある可能性が見つかった場合はテーブルのチェックを実行し、問題が検出された場合はテーブルの修復を試みます。テーブルを修復できない場合は、手動でテーブルを修復する方法について、セクション2.11.4「テーブルまたはインデックスの再作成または修復」を参照してください。

mysql_upgrade は、MySQL をアップグレードするたびに実行するようにしてください。

Linux で、MySQL を RPM パッケージからインストールする場合は、サーバーとクライアントの RPM をインストールする必要があります。mysql_upgrade はサーバー RPM に含まれていますが、クライアント RPM に mysqlcheck が含まれるため、クライアント RPM が必要です。(セクション2.5.5「RPM パッケージを使用して MySQL を Linux にインストールする」を参照してください。)

> **注記**
>
> Windows Server 2008、Vista、およびそれ以降では、mysql_upgrade を管理者権限で実行する必要があります。そのためには、コマンドプロンプトを管理者として実行してそのコマンドを実行します。そうしないと、アップグレードが正しく実行されない場合があります。

> **注意**
>
> アップグレードを実行する前に、必ず現在の MySQL インストールをバックアップするようにしてください。セクション7.2「データベースバックアップ方法」を参照してください。
>
> 一部のアップグレードの非互換性により、MySQL インストールをアップグレードして mysql_upgrade を実行する前に特殊な処理が必要な場合があります。このような非互換性が、使用しているインストールに該当するかどうかの判断、およびその対処方法については、セクション2.11.1「MySQL のアップグレード」を参照してください。

mysql_upgrade を使用するには、サーバーが稼働していることを確認してください。そのあと、次のように呼び出します。

```
shell> mysql_upgrade [options]
```

mysql_upgrade の実行後、システムテーブルに行われた変更が有効になるように、サーバーを停止して再起動します。

複数の MySQL サーバーインスタンスが稼働している場合は、目的のサーバーに適した接続パラメータを使用して mysql_upgrade を呼び出します。たとえば、ローカルホストで 3306 から 3308 までのポートでサーバーが稼働している場合、適切なポートに接続してそれぞれをアップグレードします。

```
shell> mysql_upgrade --protocol=tcp -P 3306 [other_options]
shell> mysql_upgrade --protocol=tcp -P 3307 [other_options]
shell> mysql_upgrade --protocol=tcp -P 3308 [other_options]
```

Unix でのローカルホスト接続では、--protocol=tcp オプションを使用すると、Unix ソケットファイルではなく TCP/IP を強制的に使用して接続が行われます。

mysql_upgrade は次のコマンドを実行して、テーブルのチェックと修復およびシステムテーブルのアップグレードを実行します。

```
mysqlcheck --no-defaults --databases
 --fix-db-names --fix-table-names mysql
mysqlcheck --no-defaults --check-upgrade --databases
 --auto-repair mysql
mysql < fix_priv_tables
mysqlcheck --no-defaults --all-databases
 --skip-database=mysql --fix-db-names --fix-table-names
mysqlcheck --no-defaults --check-upgrade --all-databases
 --skip-database=mysql --auto-repair
```

前述のコマンドに関する注意:

- mysql_upgrade はまた、mysql_upgrade コマンドに --write-binlog オプションが指定されたかどうかに応じて、--write-binlog または --skip-write-binlog を mysqlcheck コマンドに追加します。
- mysql_upgrade は mysqlcheck を --all-databases オプションで呼び出すため、すべてのデータベースのすべてのテーブルが処理され、完了するのに長時間かかる場合があります。各テーブルはロックされるため、処理中にほかのセッションで使用することはできません。チェックと修復の処理には時間がかかることがあり、特に大きなテーブルでは長い時間を要する可能性があります。
- --check-upgrade オプションで実行されるチェックの詳細は、CHECK TABLE ステートメントの FOR UPGRADE オプションの説明を参照してください (セクション13.7.2.2「CHECK TABLE 構文」を参照してください)。
- fix_priv_tables は、mysql_upgrade により内部的に生成され、mysql データベース内のテーブルをアップグレードするための SQL ステートメントを含むスクリプトを示します。

チェックおよび修復が行われたすべてのテーブルは、現在の MySQL バージョン番号でマークされます。これにより、次に同じバージョンのサーバーで mysql_upgrade を立ち上げるときに、そのテーブルを再度チェックして修正する必要があるかどうかが確実にわかります。

mysql_upgrade はまた、MySQL バージョン番号をデータディレクトリの mysql_upgrade_info という名前のファイルに保存します。これは、テーブルのチェックをスキップできるように、すべてのテーブルがこのリリースに関してチェック済みかどうかを迅速にチェックするために使用されます。このファイルを無視してとにかくチェックを実行するには、--force オプションを使用します。

mysql_upgrade では、ヘルプテーブルの内容はアップグレードされません。アップグレードの手順については、セクション5.1.10「サーバー側のヘルプ」を参照してください。

デフォルトでは、mysql_upgrade は MySQL root ユーザーとして実行されます。mysql_upgrade の実行時に root のパスワードが期限切れの場合、パスワードが期限切れであり、その結果 mysql_upgrade が失敗したというメッセージが表示されます。これを修正するには、root のパスワードを期限が切れていないものにリセットして、mysql_upgrade をふたたび実行します。

```
shell> mysql -u root -p
Enter password: ****  <- enter root password here
mysql> SET PASSWORD = PASSWORD('root-password');
mysql> quit

shell> mysql_upgrade [options]
```

mysql_upgrade は次のオプションをサポートします。これらはコマンド行またはオプションファイルの [mysql_upgrade] グループおよび [client] グループで指定できます。その他のオプションは mysqlcheck に渡されます。オプションファイルの詳細は、セクション4.2.6「オプションファイルの使用」を参照してください。

表 4.4 mysql_upgrade オプション

オプション名	説明	導入
--basedir	使用しない	
--character-sets-dir	文字セットがインストールされているディレクトリ	
--compress	クライアントとサーバー間で送信される情報をすべて圧縮	
--datadir	使用しない	
--debug	デバッグのログを書き込み	
--debug-check	プログラムの終了時にデバッグ情報を出力	

オプション名	説明	導入
--debug-info	プログラムの終了時に、デバッグ情報、メモリー、および CPU の統計を出力	
--default-auth	使用する認証プラグイン	5.6.2
--default-character-set	デフォルト文字セットを指定	
--defaults-extra-file	通常のオプションファイルに加えてオプションファイルを読み取る	
--defaults-file	指名されたオプションファイルのみを読み取る	
--defaults-group-suffix	オプショングループのサフィクス値	
--force	MySQL の現在のバージョンに対して mysql_upgrade を実行済みでも強制的に実行	
--help	ヘルプメッセージを表示して終了	
--host	指定されたホスト上で MySQL サーバーに接続	
--login-path	ログインパスオプションを .mylogin.cnf から読み取り	5.6.6
--no-defaults	オプションファイルを読み取らない	
--password	サーバーに接続する際に使用するパスワード	
--pipe	Windows で、名前付きパイプを使用してサーバーに接続	
--plugin-dir	プラグインがインストールされているディレクトリ	5.6.2
--port	接続に使用する TCP/IP ポート番号	
--print-defaults	デフォルトを出力	
--protocol	使用する接続プロトコル	
--shared-memory-base-name	共有メモリー接続に使用する共有メモリーの名前	
--socket	ローカルホストへの接続で、使用する Unix ソケットファイル	
--ssl	接続に SSL を有効化	
--ssl-ca	信頼された SSL CA のリストを含むファイルのパス	
--ssl-capath	信頼された SSL CA の PEM 形式の証明書を含むディレクトリのパス	
--ssl-cert	PEM 形式の X509 証明書を含むファイルのパス	
--ssl-cipher	SSL の暗号化に使用される、許可された暗号のリスト	
--ssl-crl	証明書失効リストを含むファイルのパス	5.6.3
--ssl-crlpath	証明書失効リストファイルを含むディレクトリのパス	5.6.3
--ssl-key	PEM 形式の X509 鍵を含むファイルのパス	
--ssl-verify-server-cert	サーバーへの接続時に、サーバーの証明書内のコモンネーム値をホスト名に対して検証	
--tmpdir	一時ファイルのディレクトリ	
--upgrade-system-tables	システムテーブルのみを更新し、データは更新しない	
--user	サーバーへの接続時に使用する MySQL ユーザー名	
--verbose	冗長モード	
--version-check	適切なサーバーバージョンをチェック	5.6.12
--write-binlog	すべてのステートメントをバイナリログに書き込み	

- --help

  短いヘルプメッセージを表示して終了します。

- --basedir=dir_name

  MySQL インストールディレクトリへのパス。このオプションは、後方互換性のため受け付けられますが無視されます。MySQL 5.7 では削除されました。

- --character-sets-dir=path

文字セットがインストールされているディレクトリ。セクション10.5「文字セットの構成」を参照してください。

- --compress

  クライアントとサーバーの両方が圧縮をサポートしている場合、その間で送受信される情報をすべて圧縮します。

- --datadir=dir_name

  データディレクトリへのパス。このオプションは、後方互換性のため受け付けられますが無視されます。MySQL 5.7 では削除されました。

- --debug[=debug_options], -# [debug_options]

  デバッグのログを書き込みます。一般的な debug_options 文字列は d:t:o,file_name です。デフォルトは d:t:O,/tmp/mysql_upgrade.trace です。

- --debug-check

  プログラムの終了時に、デバッグ情報を出力します。

- --debug-info, -T

  プログラムの終了時に、デバッグ情報とメモリーおよび CPU 使用率の統計を出力します。

- --default-auth=plugin

  使用するクライアント側の認証プラグイン。セクション6.3.7「プラガブル認証」を参照してください。

  このオプションは MySQL 5.6.2 で追加されました。

- --default-character-set=charset_name

  charset_name をデフォルト文字セットとして使用します。セクション10.5「文字セットの構成」を参照してください。

- --defaults-extra-file=file_name

  このオプションファイルは、グローバルオプションファイルのあとに読み取りますが、(UNIX では) ユーザーオプションファイルの前に読み取るようにしてください。ファイルが存在しないかアクセスできない場合、エラーが発生します。file_name は、フルパス名でなく相対パス名として指定された場合、現行ディレクトリを基準にして解釈されます。

- --defaults-file=file_name

  指定されたオプションファイルのみ使用します。ファイルが存在しないかアクセスできない場合、エラーが発生します。file_name は、フルパス名でなく相対パス名として指定された場合、現行ディレクトリを基準にして解釈されます。

- --defaults-group-suffix=str

  通常のオプショングループだけでなく、通常の名前に str のサフィクスが付いたグループも読み取ります。たとえば、mysql_upgrade は通常 [client] グループおよび [mysql_upgrade] グループを読み取ります。--defaults-group-suffix=_other オプションを指定した場合、mysql_upgrade は [client_other] グループおよび [mysql_upgrade_other] グループも読み取ります。

- --force

  mysql_upgrade_info ファイルを無視し、MySQL の現在のバージョンに対して mysql_upgrade を実行済みでも強制的に実行します。

- --host=host_name, -h host_name

  指定されたホストの MySQL サーバーに接続します。

- --login-path=name

  指名されたログインパスから .mylogin.cnf ログインファイルのオプションを読み取ります。「ログインパス」は、host、user、および password という限定されたオプションのセットのみを許可するオプショングルー

プです。ログインパスは、サーバーホストおよびそのサーバーで認証するための認証情報を示す値のセットであると考えてください。ログインパスファイルを作成するには、mysql_config_editor ユーティリティーを使用します。セクション4.6.6「mysql_config_editor — MySQL 構成ユーティリティー」を参照してください。このオプションは MySQL 5.6.6 で追加されました。

- --no-defaults

  オプションファイルを読み取りません。オプションファイルから不明のオプションを読み取ることが原因でプログラムの起動に失敗する場合、--no-defaults を使用して、オプションを読み取らないようにすることができます。

  例外として、.mylogin.cnf ファイルは、存在する場合はすべての場合に読み取られます。これにより、 --no-defaults が使用されたとしても、コマンド行よりも安全な方法でパスワードを指定できます。(.mylogin.cnf は mysql_config_editor ユーティリティーによって作成されます。セクション4.6.6「mysql_config_editor — MySQL 構成ユーティリティー」を参照してください)。

- --password[=password], -p[password]

  サーバーに接続する際に使用するパスワードです。短いオプション形式 (-p) を使用した場合は、オプションとパスワードの間にスペースを置くことはできません。コマンド行で、--password オプションまたは -p オプションに続けて password の値を指定しなかった場合、mysql_upgrade はそれを要求します。

  コマンド行でのパスワード指定は、セキュアでないと考えるべきです。セクション6.1.2.1「パスワードセキュリティーのためのエンドユーザーガイドライン」を参照してください。オプションファイルを使用すれば、コマンド行でパスワードを指定することを回避できます。

- --pipe, -W

  Windows で、名前付きパイプを使用してサーバーに接続します。このオプションは、サーバーが名前付きパイプ接続をサポートしている場合にのみ適用されます。

- --plugin-dir=path

  プラグインを検索するディレクトリ。--default-auth オプションを使用して認証プラグインを指定したが、mysql_upgrade がそれを検出できない場合は、このオプションを指定しなければならない可能性があります。セクション6.3.7「プラガブル認証」を参照してください。

  このオプションは MySQL 5.6.2 で追加されました。

- --port=port_num, -P port_num

  接続に使用する TCP/IP ポート番号。

- --print-defaults

  プログラム名と、オプションファイルから受け取るすべてのオプションを出力します。

- --protocol={TCP|SOCKET|PIPE|MEMORY}

  サーバーへの接続に使用する接続プロトコル。このオプションは、ほかの接続パラメータによって、必要なプロトコル以外のものが通常使用される場合に役立ちます。許可される値の詳細は、セクション4.2.2「MySQL サーバーへの接続」を参照してください。

- --shared-memory-base-name=name

  Windows で、共有メモリーを使用して作成されるローカルサーバーへの接続の共有メモリー名。デフォルト値は MYSQL です。共有メモリー名では大文字と小文字を区別します。

  共有メモリー接続を可能にするには、サーバーは --shared-memory オプションで起動する必要があります。

- --socket=path, -S path

  localhost への接続用に使用する、Unix ソケットファイル、または Windows では使用する名前付きパイプの名前。

- --ssl*

  --ssl で始まるオプションは、SSL を使用してサーバーに接続することを許可するかどうかを指定し、SSL 鍵および証明書を検索する場所を指定します。セクション6.3.10.4「SSL コマンドのオプション」を参照してください。

- --tmpdir=dir_name, -t path

  一時ファイルを作成するために使用するディレクトリのパス名。

- --upgrade-system-tables, -s

  システムテーブルのみをアップグレードし、データはアップグレードしません。

- --user=user_name, -u user_name

  サーバーへの接続時に使用する MySQL ユーザー名。デフォルトのユーザー名は root です。

- --verbose

  冗長モード。プログラムの動作についてより多くの情報を出力します。

- --version-check, -k

  mysql_upgrade の接続先のサーバーのバージョンをチェックして、mysql_upgrade がビルドされたバージョンと同じであることを確認します。そうでない場合は mysql_upgrade は終了します。このオプションはデフォルトで有効にされています。--skip-version-check を使用して無効化してください。このオプションは MySQL 5.6.12 で追加されました。

- --write-binlog

  mysql_upgrade の実行中にバイナリロギングを有効にします。MySQL 5.6.6 以前では、これはデフォルトの動作でした。(アップグレード中にバイナリロギングを無効にするには、プログラムを --skip-write-binlog で起動することによって、このオプションの逆を使用する必要がありました。)MySQL 5.6.7 から、mysql_upgrade によるバイナリロギングはデフォルトで無効です (Bug #14221043)。アクションをバイナリログに書き込む場合は、明示的に --write-binlog を使用してプログラムを呼び出します。(また、MySQL 5.6.7 から、--skip-write-binlog オプションは実際には何もしません。)

  グローバルトランザクション識別子を有効にして稼働している MySQL サーバーでは、mysql_upgrade を実行することは推奨されません (Bug #13833710)。これは、GTID を有効にするということは、mysql_upgrade がシステムテーブルに対して、MyISAM などの非トランザクションストレージエンジンを使用して実行する必要のある更新が、失敗することを意味するからです。詳細は、セクション17.1.3.4「GTID ベースレプリケーションの制約」を参照してください。

4.5 MySQL クライアントプログラム

このセクションでは、MySQL サーバーに接続するクライアントプログラムについて説明します。

4.5.1 mysql — MySQL コマンド行ツール

mysql は、入力行の編集機能を備えた簡単な SQL シェルです。インタラクティブおよび非インタラクティブでの使用をサポートします。インタラクティブで使用した場合、クエリー結果は ASCII 表形式で提示されます。非インタラクティブ (フィルタとしてなど) で使用する場合、結果はタブ区切り形式で表示されます。出力形式は、コマンドオプションを使用して変更できます。

大きな結果セット用のメモリーが足りないことで問題が発生する場合は、--quick オプションを使用します。これにより mysql は、全結果セットを表示前に取得してメモリー内でバッファリングする代わりに、サーバーから 1 行ずつ結果を取得することを強制されます。これは、mysql_store_result() ではなく、クライアント/サーバーライブラリ内の mysql_use_result() C API 関数を使用して結果セットを返すことで実行されます。

mysql は非常に簡単に使用できます。次のように、コマンドインタプリタのプロンプトから起動してください。

```
shell> mysql db_name
```

または:

```
shell> mysql --user=user_name --password=your_password db_name
```

次に SQL ステートメントを入力し、「;」、\g、または \G で終わらせ Enter を押します。

Control+C と入力すると、mysql は現在のステートメントの強制終了を試行します。これが実行できない場合、またはステートメントを強制終了する前に Control+C が再度入力された場合は、mysql は終了します。

SQL ステートメントは次のように、スクリプトファイル (バッチファイル) で実行できます。

```
shell> mysql db_name < script.sql > output.tab
```

Unix では、mysql クライアントはインタラクティブに実行されたステートメントを履歴ファイルにログ記録します。セクション4.5.1.3「mysql のロギング」を参照してください。

4.5.1.1 mysql のオプション

mysql は次のオプションをサポートします。これらはコマンド行またはオプションファイルの [mysql] グループおよび [client] グループで指定できます。MySQL プログラムによって使用されるオプションファイルの詳細については、セクション4.2.6「オプションファイルの使用」を参照してください。

表 4.5 mysql のオプション

オプション名	説明	導入
--auto-rehash	自動リハッシュを有効化	
--auto-vertical-output	結果セットの垂直形式での表示を有効化	
--batch	履歴ファイルを使用しない	
--binary-mode	\r\n の \n への変換および \0 をクエリーの終端として処理することを無効化	5.6.3
--bind-address	指定されたネットワークインタフェースを使用して MySQL サーバーに接続	5.6.1
--character-sets-dir	文字セットがインストールされているディレクトリ	
--column-names	結果にカラム名を記述	
--column-type-info	結果セットのメタデータを表示	
--comments	サーバーに送信されたステートメント内のコメントを保持するか削除するか	
--compress	クライアントとサーバー間で送信される情報をすべて圧縮	
--connect-expired-password	クライアントが期限切れパスワードのサンドボックスモードを処理できることをサーバーに指示。	5.6.12
--connect_timeout	接続タイムアウトまでの秒数	
--database	使用されるべきデータベース	
--debug	デバッグのログを書き込み	
--debug-check	プログラムの終了時にデバッグ情報を出力	
--debug-info	プログラムの終了時に、デバッグ情報、メモリー、および CPU の統計を出力	
--default-auth	使用する認証プラグイン	
--default-character-set	デフォルト文字セットを指定	
--defaults-extra-file	通常のオプションファイルに加えてオプションファイルを読み取る	
--defaults-file	指名されたオプションファイルのみを読み取る	
--defaults-group-suffix	オプショングループのサフィクス値	
--delimiter	ステートメント区切り文字を設定	
--enable-cleartext-plugin	平文の認証プラグインを有効化	5.6.7
--execute	ステートメントを実行して終了	
--force	SQL エラーが発生しても続行	
--help	ヘルプメッセージを表示して終了	
--histignore	ロギングに関してどのステートメントを無視するかを指定するパターン	5.6.8
--host	指定されたホスト上で MySQL サーバーに接続	
--html	HTML 出力を生成	
--ignore-spaces	関数名のあとのスペースを無視	
--init-command	接続後に実行する SQL ステートメント	
--line-numbers	エラーの行番号を書き込み	

オプション名	説明	導入
--local-infile	LOAD DATA INFILE で LOCAL 機能を有効化または無効化	
--login-path	ログインパスオプションを .mylogin.cnf から読み取り	5.6.6
--max_allowed_packet	サーバーとの間で送受信するパケットの最大長	
--max_join_size	--safe-updates を使用する結合で、自動的に設定される行の制限	
--named-commands	名前付き mysql コマンドを有効化	
--net_buffer_length	TCP/IP とソケット通信のバッファーサイズ	
--no-auto-rehash	自動リハッシュを無効化	
--no-beep	エラー時に音を発生させない	
--no-defaults	オプションファイルを読み取らない	
--one-database	コマンド行で指定されたデフォルトデータベースに対するステートメント以外を無視	
--pager	クエリー出力のページングに指定されたコマンドを使用	
--password	サーバーに接続する際に使用するパスワード	
--pipe	Windows で、名前付きパイプを使用してサーバーに接続	
--plugin-dir	プラグインがインストールされているディレクトリ	
--port	接続に使用する TCP/IP ポート番号	
--print-defaults	デフォルトを出力	
--prompt	プロンプトを指定された形式に設定	
--protocol	使用する接続プロトコル	
--quick	各クエリーの結果をキャッシュしない	
--raw	カラム値をエスケープの変換なしで書き込み	
--reconnect	サーバーとの接続が失われたとき、再接続を自動的に試行	
--safe-updates, --i-am-a-dummy	キー値を指定する UPDATE ステートメントおよび DELETE ステートメントのみを許可	
--secure-auth	古い (4.1.1 より前の) 形式でサーバーにパスワードを送信しない	
--select_limit	--safe-updates 使用時に自動的に設定される SELECT ステートメントの制限	
--server-public-key-path	RSA 公開鍵を含むファイルへのパス名	5.6.6
--shared-memory-base-name	共有メモリー接続に使用する共有メモリーの名前	
--show-warnings	各ステートメントのあとに警告があれば表示	
--sigint-ignore	SIGINT 信号を無視 (通常、Control+C を入力した結果)	
--silent	サイレントモード	
--skip-auto-rehash	自動リハッシュを無効化	
--skip-column-names	結果にカラム名を記述しない	
--skip-line-numbers	エラーの行番号を省略	
--skip-named-commands	名前付き mysql コマンドを無効化	
--skip-pager	ページングを無効化	
--skip-reconnect	再接続を無効化	
--socket	ローカルホストへの接続で、使用する Unix ソケットファイル	
--ssl	接続に SSL を有効化	
--ssl-ca	信頼された SSL CA のリストを含むファイルのパス	
--ssl-capath	信頼された SSL CA の PEM 形式の証明書を含むディレクトリのパス	
--ssl-cert	PEM 形式の X509 証明書を含むファイルのパス	

オプション名	説明	導入
--ssl-cipher	SSL の暗号化に使用される、許可された暗号のリスト	
--ssl-crl	証明書失効リストを含むファイルのパス	5.6.3
--ssl-crlpath	証明書失効リストファイルを含むディレクトリのパス	5.6.3
--ssl-key	PEM 形式の X509 鍵を含むファイルのパス	
--ssl-verify-server-cert	サーバーへの接続時に、サーバーの証明書内のコモンネーム値をホスト名に対して検証	
--table	出力を表形式で表示	
--tee	出力のコピーを指定されたファイルに追加	
--unbuffered	各クエリーのあとでバッファーをフラッシュ	
--user	サーバーへの接続時に使用する MySQL ユーザー名	
--verbose	冗長モード	
--version	バージョン情報を表示して終了	
--vertical	クエリー出力行を垂直形式で出力 (カラム値ごとに 1 行)	
--wait	接続が確立できない場合、中止せずに待機してからリトライ	
--xml	XML 出力を生成	

- --help, -?

  ヘルプメッセージを表示して終了します。

- --auto-rehash

  自動リハッシュを有効にします。このオプションはデフォルトでオンになっており、データベース、テーブル、およびカラムの名前補完が可能になります。リハッシュを無効にするには、--disable-auto-rehash を使用します。これにより、mysql の起動がより高速になりますが、名前補完を使用する場合は rehash コマンドまたはそのショートカット \# を発行する必要があります。

  名前を補完するには、最初の部分を入力して Tab を押します。名前があいまいでない場合、mysql がその名前を補完します。そうでない場合は、Tab をもう一度押して、これまでに入力した値で始まる、考えられる名前を表示できます。デフォルトのデータベースがない場合、補完は行われません。

  > **注記**
  >
  > この機能では、readline ライブラリでコンパイルされた MySQL クライアントが必要です。通常、readline ライブラリは Windows では使用できません。

- --auto-vertical-output

  結果セットが現在のウィンドウに対して大きすぎる場合には縦に表示され、そうでない場合は通常の表形式が使用されるようになります。(これは、; または \G で終了するステートメントに適用されます。)

- --batch, -B

  カラム区切り文字としてタブを使用し、各行を新しい行に出力します。このオプションでは、mysql は履歴ファイルを使用しません。

  バッチモードでは、表形式でない出力が生成され、特殊文字のエスケープ処理が行われます。raw モードを使用すれば、エスケープ処理を無効にできます。--raw オプションの説明を参照してください。

- --binary-mode

  このオプションは、BLOB 値を含む可能性のある mysqlbinlog の出力を処理する場合に便利です。mysql は、デフォルトではステートメント文字列内の \r\n を \n に変換し、\0 をステートメント終端記号と解釈します。--binary-mode ではこの両方の機能が無効になります。また、非インタラクティブモード (mysql にパイプされた入力または source コマンドでロードされた入力) の charset および delimiter を除いて、すべての mysql コマンドも無効になります。

  このオプションは MySQL 5.6.3 で追加されました。

- --bind-address=ip_address

複数のネットワークインタフェースを持つコンピュータで、このオプションを使用して、MySQL サーバーへの接続に使用するインタフェースを選択します。

このオプションは MySQL 5.6.1 からサポートされています。

- --character-sets-dir=path

  文字セットがインストールされているディレクトリ。セクション10.5「文字セットの構成」を参照してください。

- --column-names

  結果にカラム名を記述します。

- --column-type-info, -m

  結果セットのメタデータを表示します。

- --comments, -c

  サーバーに送信されたステートメント内のコメントを保持するかどうかを指定します。デフォルトは --skip-comments (コメントを破棄) であり、--comments (コメントを保持) で有効になります。

- --compress, -C

  クライアントとサーバーの両方が圧縮をサポートしている場合、その間で送受信される情報をすべて圧縮します。

- --connect-expired-password

  サーバーに対し、接続に使用されたアカウントのパスワードの期限が切れている場合に、クライアントがサンドボックスモードを処理できることを示します。通常サーバーは、パスワードの期限の切れたアカウントを使用して接続を試みる非インタラクティブなクライアントを切断するため、これは非インタラクティブに mysql を起動する場合に便利です。(セクション6.3.6「パスワードの期限切れとサンドボックスモード」を参照してください。)このオプションは MySQL 5.6.12 で追加されました。

- --database=db_name, -D db_name

  使用するデータベース。これは主に、オプションファイルで便利です。

- --debug[=debug_options], -# [debug_options]

  デバッグのログを書き込みます。一般的な debug_options 文字列は d:t:o,file_name です。デフォルトは d:t:o,/tmp/mysql.trace です。

- --debug-check

  プログラムの終了時に、デバッグ情報を出力します。

- --debug-info, -T

  プログラムの終了時に、デバッグ情報とメモリーおよび CPU 使用率の統計を出力します。

- --default-auth=plugin

  使用するクライアント側の認証プラグイン。セクション6.3.7「プラガブル認証」を参照してください。

- --default-character-set=charset_name

  クライアントおよび接続で、charset_name をデフォルト文字セットとして使用します。

  オペレーティングシステムが utf8 またはその他のマルチバイト文字セットを使用する場合に生じる一般的な問題は、MySQL クライアントがデフォルトで latin1 文字セットを使用するため、mysql クライアントからの出力が正しくフォーマットされないことです。このような問題は通常、このオプションを使用して、クライアントが代わりにシステムの文字セットを使用するように強制することで解決できます。

  詳細は、セクション10.5「文字セットの構成」を参照してください。

- --defaults-extra-file=file_name

このオプションファイルは、グローバルオプションファイルのあとに読み取りますが、(UNIX では) ユーザーオプションファイルの前に読み取るようにしてください。ファイルが存在しないかアクセスできない場合、エラーが発生します。file_name は、フルパス名でなく相対パス名として指定された場合、現行ディレクトリを基準にして解釈されます。

- --defaults-file=file_name

  指定されたオプションファイルのみ使用します。ファイルが存在しないかアクセスできない場合、エラーが発生します。file_name は、フルパス名でなく相対パス名として指定された場合、現行ディレクトリを基準にして解釈されます。

- --defaults-group-suffix=str

  通常のオプショングループだけでなく、通常の名前に str のサフィクスが付いたグループも読み取ります。たとえば、mysql は通常 [client] グループおよび [mysql] グループを読み取ります。--defaults-group-suffix=_other オプションを指定した場合、mysql は [client_other] グループおよび [mysql_other] グループも読み取ります。

- --delimiter=str

  ステートメント区切り文字を設定します。デフォルトはセミコロン (「;」) 文字です。

- --disable-named-commands

  名前付きコマンドを無効化します。* 形式のみを使用します。または、セミコロン (「;」) で終わる行の先頭でのみ名前付きコマンドを使用します。mysql の起動時に、このオプションはデフォルトで有効になっています。ただし、このオプションを使用しても、ロング形式コマンドは最初の行から効果を発揮します。セクション4.5.1.2「mysql コマンド」を参照してください。

- --enable-cleartext-plugin

  mysql_clear_password 平文認証プラグインを有効にします。(セクション6.3.8.7「クライアント側のクリアテキスト認証プラグイン」を参照してください。)このオプションは MySQL 5.6.7 で追加されました。

- --execute=statement, -e statement

  ステートメントを実行して、終了します。デフォルトの出力形式は、--batch で生成されるものと同様です。例については、セクション4.2.4「コマンド行でのオプションの使用」を参照してください。このオプションでは、mysql は履歴ファイルを使用しません。

- --force, -f

  SQL エラーが発生しても続行します。

- --histignore

  ロギングのために、どのステートメントを無視するかを指定する、コロン区切りの 1 つまたは複数のパターンのリスト。これらのパターンはデフォルトのパターンリスト ("*IDENTIFIED*:*PASSWORD*") に追加されます。このオプションに指定された値は、履歴ファイルに書き込まれるステートメントの口ギングに影響します。詳細は、セクション4.5.1.3「mysql の口ギング」を参照してください。このオプションは MySQL 5.6.8 で追加されました。

- --host=host_name, -h host_name

  指定されたホストの MySQL サーバーに接続します。

- --html, -H

  HTML 出力を生成します。

- --ignore-spaces, -i

  関数名の後ろのスペースを無視します。これの効果は、IGNORE_SPACE SQL モード (セクション5.1.7「サーバー SQL モード」を参照してください) に関する議論で説明されています。

- --init-command=str

  サーバーへの接続後に実行する SQL ステートメント。auto-reconnect が有効の場合、ステートメントは再接続が生じたあとに再度実行されます。

- --line-numbers

エラーの行番号を書き込みます。--skip-line-numbers で無効にできます。

- --local-infile[={0|1}]

  LOAD DATA INFILE で LOCAL 機能を有効または無効にします。値がない場合、オプションは LOCAL を有効にします。このオプションは、LOCAL を明示的に有効または無効にするため、--local-infile=0 または --local-infile=1 として指定できます。サーバーも LOCAL をサポートしていない場合、有効にしても効果はありません。

- --login-path=name

  指名されたログインパスから .mylogin.cnf ログインファイルのオプションを読み取ります。「ログインパス」は、host、user、および password という限定されたオプションのセットのみを許可するオプショングループです。ログインパスは、サーバーホストおよびそのサーバーで認証するための認証情報を示す値のセットであると考えてください。ログインパスファイルを作成するには、mysql_config_editor ユーティリティーを使用します。セクション4.6.6「mysql_config_editor — MySQL 構成ユーティリティー」を参照してください。このオプションは MySQL 5.6.6 で追加されました。

- --named-commands, -G

  名前付き mysql コマンドを有効にします。ショート形式のコマンドのみではなく、ロング形式のコマンドが許可されます。たとえば、quit と \q は両方認識されます。名前付きコマンドを無効化するには、--skip-named-commands を使用します。セクション4.5.1.2「mysql コマンド」を参照してください。

- --no-auto-rehash, -A

  これは --skip-auto-rehash と同様の効果を持ちます。--auto-rehash の説明を参照してください。

- --no-beep, -b

  エラー時に音を発生させません。

- --no-defaults

  オプションファイルを読み取りません。オプションファイルから不明のオプションを読み取ることが原因でプログラムの起動に失敗する場合、--no-defaults を使用して、オプションを読み取らないようにできます。

  例外として、.mylogin.cnf ファイルは、存在する場合はすべての場合に読み取られます。これにより、--no-defaults が使用されたとしても、コマンド行よりも安全な方法でパスワードを指定できます。(.mylogin.cnf は mysql_config_editor ユーティリティーによって作成されます。セクション4.6.6「mysql_config_editor — MySQL 構成ユーティリティー」を参照してください)。

- --one-database, -o

  デフォルトデータベースがコマンド行で指定されたものである間に生じるステートメント以外を無視します。このオプションは初歩的なものであり、注意して使用してください。ステートメントのフィルタリングは USE ステートメントのみに基づいています。

  最初、mysql は入力内のステートメントを実行します。これは、データベース db_name をコマンド行で指定することは USE db_name を入力の最初に挿入することと同等であるためです。次に、USE ステートメントを検出するたびに、mysql は、指名されたデータベースがコマンド行のものであるかによって、後続のステートメントを受け入れまたは拒否します。ステートメントの内容は重要ではありません。

  mysql が次の一連のステートメントを処理するために起動されたとします。

```
DELETE FROM db2.t2;
USE db2;
DROP TABLE db1.t1;
CREATE TABLE db1.t1 (i INT);
USE db1;
INSERT INTO t1 (i) VALUES(1);
CREATE TABLE db2.t1 (j INT);
```

  コマンド行が mysql --force --one-database db1 の場合、mysql は入力を次のように処理します。

  - DELETE ステートメントでは別のデータベースのテーブルが指名されていますが、デフォルトデータベースは db1 であるため、このステートメントは実行されます。

  - DROP TABLE ステートメントおよび CREATE TABLE ステートメントでは db1 のテーブルを指名していますが、デフォルトデータベースが db1 ではないため、これらのステートメントは実行されません。
  - CREATE TABLE ステートメントでは別のデータベースのテーブルが指名されていますが、デフォルトデータベースは db1 であるため、INSERT ステートメントおよび CREATE TABLE ステートメントは実行されます。

- --pager[=command]

  クエリー出力のページングに、指定されたコマンドを使用します。このコマンドが省略された場合、デフォルトのページャーは PAGER 環境変数の値となります。有効なページャーは、less、more、cat [> filename]、などです。このオプションは Unix でインタラクティブモードの場合のみ機能します。ページングを無効化するには、--skip-pager を使用してください。セクション4.5.1.2「mysql コマンド」には出力のページングの詳細説明があります。

- --password[=password], -p[password]

  サーバーに接続する際に使用するパスワードです。短いオプション形式 (-p) を使用した場合は、オプションとパスワードの間にスペースを置くことはできません。コマンド行で、--password オプションまたは -p オプションに続けて password の値を指定しなかった場合、mysql はそれを要求します。

  コマンド行でのパスワード指定は、セキュアでないと考えるべきです。セクション6.1.2.1「パスワードセキュリティーのためのエンドユーザーガイドライン」を参照してください。オプションファイルを使用すれば、コマンド行でパスワードを指定することを回避できます。

- --pipe, -W

  Windows で、名前付きパイプを使用してサーバーに接続します。このオプションは、サーバーが名前付きパイプ接続をサポートしている場合にのみ適用されます。

- --plugin-dir=path

  プラグインを検索するディレクトリ。--default-auth オプションを使用して認証プラグインを指定したが、mysql がそれを検出できない場合は、このオプションを指定しなければならない可能性があります。セクション6.3.7「プラガブル認証」を参照してください。

- --port=port_num, -P port_num

  接続に使用する TCP/IP ポート番号。

- --print-defaults

  プログラム名と、オプションファイルから受け取るすべてのオプションを出力します。

- --prompt=format_str

  プロンプトを指定された形式に設定します。デフォルトは mysql> です。プロンプト内に含めることができる特別なシーケンスは、セクション4.5.1.2「mysql コマンド」で説明されています。

- --protocol={TCP|SOCKET|PIPE|MEMORY}

  サーバーへの接続に使用する接続プロトコル。このオプションは、ほかの接続パラメータによって、必要なプロトコル以外のものが通常使用される場合に役立ちます。許可される値の詳細は、セクション4.2.2「MySQL サーバーへの接続」を参照してください。

- --quick, -q

  各クエリー結果をキャッシュせず、各行を受信ししだい出力します。出力が遅延された場合、サーバーの速度が低下することがあります。このオプションでは、mysql は履歴ファイルを使用しません。

- --raw, -r

  表形式の出力では、カラムを「枠で囲む」ことにより、1 つのカラム値を別のカラム値と区別できます。表形式でない出力 (バッチモードで生成される場合や --batch または --silent オプションを指定した場合など) では、簡単に識別できるように特殊文字が出力時に工スケープされます。改行、タブ、NUL、およびバックスラッ

シュはそれぞれ、\n、\t、\0、および \\ と記述されます。--raw オプションを指定すると、この文字のエスケープ処理は無効になります。

次の例は、表形式と表形式でない出力の違い、および raw モードを使用してエスケープ処理を無効にした場合を示しています。

```
% mysql
mysql> SELECT CHAR(92);
+----------+
| CHAR(92) |
+----------+
| \        |
+----------+

% mysql -s
mysql> SELECT CHAR(92);
CHAR(92)
\\

% mysql -s -r
mysql> SELECT CHAR(92);
CHAR(92)
\
```

- --reconnect

  サーバーとの接続が失われたとき、再接続を自動的に試行します。接続が失われるたびに一度再接続が試みられます。再接続動作を抑制するには、--skip-reconnect を使用します。

- --safe-updates, --i-am-a-dummy, -U

  キー値を使用して変更する行を指定する UPDATE ステートメントおよび DELETE ステートメントのみを許可します。このオプションをオプションファイル内で設定した場合、--safe-updates をコマンド行で使用することでオーバーライドできます。このオプションについては、セクション4.5.1.6「mysql のヒント」を参照してください。

- --secure-auth

  古い (4.1 より前の) 形式でサーバーにパスワードを送信しません。これにより、新しいパスワード形式を使用するサーバー以外への接続を防ぎます。MySQL 5.6.7 では、このオプションはデフォルトで有効です。無効にするには --skip-secure-auth を使用します。

  > **注記**
  >
  > 4.1 より前のハッシュ方式を使用するパスワードはネイティブのパスワードハッシュ方式を使用するパスワードよりもセキュアでないため、使用しないようにしてください。4.1 よりも前のパスワードは非推奨であり、これらのサポートは今後の MySQL リリースで削除される予定です。アカウントのアップグレード手順については、セクション6.3.8.3「4.1 よりも前のパスワードハッシュ方式と mysql_old_password プラグインからの移行」を参照してください。

- --server-public-key-path=file_name

  サーバー RSA 公開鍵を含むファイルへのパス名。ファイルは PEM 形式である必要があります。公開鍵は、sha256_password プラグインによって認証するアカウントを使用して、サーバーへの接続を作成するためのクライアントパスワードの RSA 暗号化に使用されます。このオプションは、そのプラグインによって認証しないクライアントアカウントに対しては無視されます。さらに、クライアントが SSL 接続を使用して、サーバーに接続する場合のように、パスワード暗号化が必要でない場合も無視されます。

  サーバーは必要に応じて公開鍵をクライアントに送信するため、RSA パスワードの暗号化が行われるように、このオプションを使う必要はありません。そうすることで、サーバーは鍵を送信する必要がないため、効率が向上します。

  RSA 公開鍵の取得方法を含め、sha256_password プラグインの使用に関する追加の説明については、セクション6.3.8.4「SHA-256 認証プラグイン」を参照してください。

  このオプションは、MySQL が OpenSSL を使用してビルドされている場合のみ利用できます。MySQL 5.6.6 で --server-public-key という名前で追加され、5.6.7 で --server-public-key-path に名前変更されました。

- --shared-memory-base-name=name

  Windows で、共有メモリーを使用して作成されるローカルサーバーへの接続の共有メモリー名。デフォルト値は MYSQL です。共有メモリー名では大文字と小文字を区別します。

  共有メモリー接続を可能にするには、サーバーは --shared-memory オプションで起動する必要があります。

- --show-warnings

  警告が存在する場合、各ステートメント後に表示させます。このオプションはインタラクティブとバッチモードにのみ対応しています。

- --sigint-ignore

  SIGINT 信号を無視します (通常、Control+C を入力した結果)。

- --silent, -s

  サイレントモード。出力の生成を少なくします。このオプションを複数回指定して、出力の生成をさらに少なくできます。

  このオプションでは、表形式でない出力が生成され、特殊文字のエスケープ処理が行われます。raw モードを使用すれば、エスケープ処理を無効にできます。--raw オプションの説明を参照してください。

- --skip-column-names, -N

  結果にカラム名を記述しません。

- --skip-line-numbers, -L

  エラーの行番号を書き込みません。エラーメッセージを含む結果ファイルを比較する場合に便利です。

- --socket=path, -S path

  localhost への接続用に使用する、Unix ソケットファイル、または Windows では使用する名前付きパイプの名前。

- --ssl*

  --ssl で始まるオプションは、SSL を使用してサーバーに接続することを許可するかどうかを指定し、SSL 鍵および証明書を検索する場所を指定します。セクション6.3.10.4「SSL コマンドのオプション」を参照してください。

- --table, -t

  出力を表形式で表示します。インタラクティブに使用する場合はこれがデフォルトですが、バッチモードで表形式の出力を生成するのにも使用できます。

- --tee=file_name

  出力のコピーを指定されたファイルに追加します。このオプションはインタラクティブモードの場合のみ機能します。セクション4.5.1.2「mysql コマンド」で、tee ファイルについて詳細に説明しています。

- --unbuffered, -n

  各クエリー後にバッファーをフラッシュします。

- --user=user_name, -u user_name

  サーバーへの接続時に使用する MySQL ユーザー名。

- --verbose, -v

  冗長モード。プログラムの動作についてより多くの出力を生成します。このオプションを複数回指定して、さらに多くの出力を生成できます。(たとえば、-v -v -v ではバッチモードでも表形式の出力を生成します。)

- --version, -V

  バージョン情報を表示して終了します。

- --vertical, -E

クエリー出力行を縦に出力します (カラム値ごとに一行)。このオプションを使用しない場合、個々のステートメントを \G で終了させることにより、縦の出力を指定できます。

- --wait, -w

  接続が確立できない場合、中止せずに待機してからリトライします。

- --xml, -X

  XML 出力を生成します。

```
<field name="column_name">NULL</field>
```

  --xml が mysql とともに使用された場合の出力は、mysqldump --xml の出力と一致します。詳細は、セクション 4.5.4「mysqldump — データベースバックアッププログラム」を参照してください。

  次に示すように、XML 出力は XML 名前空間も使用します。

```
shell> mysql --xml -uroot -e "SHOW VARIABLES LIKE 'version%'"
<?xml version="1.0"?>

<resultset statement="SHOW VARIABLES LIKE 'version%'" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
<row>
<field name="Variable_name">version</field>
<field name="Value">5.0.40-debug</field>
</row>

<row>
<field name="Variable_name">version_comment</field>
<field name="Value">Source distribution</field>
</row>

<row>
<field name="Variable_name">version_compile_machine</field>
<field name="Value">i686</field>
</row>

<row>
<field name="Variable_name">version_compile_os</field>
<field name="Value">suse-linux-gnu</field>
</row>
</resultset>
```

  (Bug #25946 を参照してください。)

--var_name=value を使用して、次の変数も設定できます。

- connect_timeout

  接続タイムアウトまでの秒数。(デフォルト値は 0 です。)

- max_allowed_packet

  クライアント/サーバー通信用のバッファーの最大サイズ。デフォルトは 16M バイト、最大は 1G バイトです。

- max_join_size

  --safe-updates 使用時の、自動的に設定される結合内の行の制限。(デフォルト値は 1,000,000 です。)

- net_buffer_length

  TCP/IP とソケット通信のバッファーサイズ。(デフォルト値は 16K バイトです。)

- select_limit

  --safe-updates 使用時の、自動的に設定される SELECT ステートメントの制限。(デフォルト値は 1,000 です。)

4.5.1.2 mysql コマンド

mysql は、ユーザーが発行する各 SQL ステートメントを、実行のためサーバーに送信します。mysql 自体が解釈するコマンドもあります。これらのコマンドのリストを表示するには、mysql> プロンプトで help または \h と入力します。

```
mysql> help

List of all MySQL commands:
Note that all text commands must be first on line and end with ';'
?         (\?) Synonym for `help'.
clear     (\c) Clear command.
connect   (\r) Reconnect to the server. Optional arguments are db and host.
delimiter (\d) Set statement delimiter.
edit      (\e) Edit command with $EDITOR.
ego       (\G) Send command to mysql server, display result vertically.
exit      (\q) Exit mysql. Same as quit.
go        (\g) Send command to mysql server.
help      (\h) Display this help.
nopager   (\n) Disable pager, print to stdout.
notee     (\t) Don't write into outfile.
pager     (\P) Set PAGER [to_pager]. Print the query results via PAGER.
print     (\p) Print current command.
prompt    (\R) Change your mysql prompt.
quit      (\q) Quit mysql.
rehash    (\#) Rebuild completion hash.
source    (\.) Execute an SQL script file. Takes a file name as an argument.
status    (\s) Get status information from the server.
system    (\!) Execute a system shell command.
tee       (\T) Set outfile [to_outfile]. Append everything into given
               outfile.
use       (\u) Use another database. Takes database name as argument.
charset   (\C) Switch to another charset. Might be needed for processing
               binlog with multi-byte charsets.
warnings  (\W) Show warnings after every statement.
nowarning (\w) Don't show warnings after every statement.

For server side help, type 'help contents'
```

mysql が --binary-mode オプションで起動された場合、非インタラクティブモードの charset および delimiter (mysql にパイプされた入力または source コマンドでロードされた入力) を除いて、すべての mysql コマンドは無効になります。

各コマンドにはそれぞれロング形式とショート形式があります。ロング形式では大文字小文字は区別されませんが、ショート形式では区別されます。ロング形式にはオプションのセミコロン終端記号があとに続くこともありますが、ショート形式ではありません。

複数行の /* ... */ コメント内でショート形式のコマンドを使用することはサポートされていません。

- help [arg], \h [arg], \? [arg], ? [arg]

  使用可能な mysql コマンドの一覧を示すヘルプメッセージを表示します。

  help コマンドに引数を指定した場合、mysql は、サーバー側のヘルプにアクセスして MySQL リファレンスマニュアルの内容から検索するための文字列として引数を使用します。詳細は、セクション4.5.1.4「mysql サーバー側のヘルプ」を参照してください。

- charset charset_name, \C charset_name

  デフォルトの文字セットを変更し、SET NAMES ステートメントを発行します。これにより、自動再接続が有効になっている状態で mysql が稼働中の場合 (これは推奨されていません) に、クライアントとサーバーの間で文字セットの同期が保持されます。これは、指定された文字セットが再接続に使用されるためです。

- clear, \c

  現在の入力をクリアします。これは、入力しているステートメントの実行を取りやめる場合に使用します。

- connect [db_name host_name]], \r [db_name host_name]]

  サーバーに再接続します。オプションのデータベース名およびホスト名の引数を指定して、デフォルトのデータベースまたはサーバーが稼働しているホストを指定することもできます。省略した場合は、現在の値が使用されます。

- delimiter str, \d str

  mysql が SQL ステートメント間の区切り文字として解釈する文字列を変更します。デフォルトはセミコロン (「;」) 文字です。

  区切り文字の文字列は、delimiter コマンド行で引用符ありまたはなしの引数として指定できます。単一引用符 ('')、二重引用符 ("")、または逆引用符 (`) 文字で囲むことができます。引用符で囲まれた文字列内に引用符を含

めるには、別の引用符で文字列を囲むか、またはバックスラッシュ (「\」) 文字で引用符を工スケープ処理します。バックスラッシュは MySQL の工スケープ文字であるため、引用符で囲まれた文字列の外側では使用しないようにしてください。引用符で囲まれていない引数については、区切り文字は最初のスペースまたは行の最後まで読み取られます。引用符で囲まれた引数の場合、区切り文字はその行の対応する引用符まで読み取られます。

mysql は、区切り文字列のインスタンスがどこで検出されても (引用符で囲まれた文字列内を除いて)、ステートメントの区切り文字として解釈します。ほかの語に出現する可能性のある区切り文字を定義しないように注意してください。たとえば、区切り文字を X と定義すると、ステートメントで INDEX という語を使用できなくなります。mysql はこれを INDE に区切り文字 X が続くと解釈します。

mysql によって認識される区切り文字がデフォルトの 「;」 以外の何かに設定されている場合、その文字のインスタンスは解釈されずにサーバーに送信されます。しかし、サーバー自体は引き続き 「;」 をステートメントの区切り文字として解釈し、その解釈に従ってステートメントを処理します。サーバー側でのこの動作は、複数ステートメントの実行 (セクション23.7.17「複数ステートメント実行の C API サポート」を参照してください) や、ストアドプロシージャーおよび関数の本体、トリガー、およびイベントの解析 (セクション20.1「ストアドプログラムの定義」を参照してください) に効果があります。

- edit, \e

  現在の入力ステートメントを編集します。mysql では、EDITOR および VISUAL 環境変数の値を確認して、どのエディタを使用するかを判断します。どちらの変数も設定されていない場合、デフォルトの工ディタは vi です。

  edit コマンドは Unix でのみ機能します。

- ego, \G

  現在のステートメントを、実行するためにサーバーに送信し、結果を縦の形式で表示します。

- exit, \q

  mysql を終了します。

- go, \g

  現在のステートメントを、実行するためにサーバーに送信します。

- nopager, \n

  出力のページングを無効にします。pager の説明を参照してください。

  nopager コマンドは Unix でのみ機能します。

- notee, \t

  tee ファイルへの出力コピーを無効にします。tee の説明を参照してください。

- nowarning, \w

  各ステートメントのあとの警告の表示を無効にします。

- pager [command], \P [command]

  出力のページングを有効にします。mysql を呼び出すときに --pager オプションを使用することで、less、more、またはその他の同様のプログラムなどの Unix プログラムを使って、インタラクティブモードでクエリー結果を参照または検索できます。オプションで値を特定しない場合、mysql は PAGER 環境変数の値を確認し、ページャーをその値に設定します。ページャー機能はインタラクティブモードの場合のみ機能します。

  出力ページングは pager コマンドでインタラクティブに有効にでき、nopager で無効にできます。このコマンドはオプションの引数を取ります。指定された場合、ページングプログラムはそれに設定されます。引数がない場合、ページャーはコマンド行で設定されたもの、またはページャーが指定されていない場合は stdout になります。

  出力ページングは Unix 上でのみ機能します。これは Windows では存在しない popen() 関数を使用するからです。Windows では、クエリー出力の保存に tee オプションを代わりに使用できますが、これは場合によっては、出力のブラウズには pager ほど便利ではありません。

- print, \p

  現在の入力ステートメントを実行しないで出力します。

- prompt [str], \R [str]

  mysql プロンプトを指定の文字列に再構成します。プロンプトで使用できる特殊文字シーケンスについては、このセクションのあとの方で説明します。

  引数なしで prompt コマンドを指定すると、mysql はプロンプトをデフォルトの mysql> にリセットします。

- quit, \q

  mysql を終了します。

- rehash, \#

  ステートメントの入力中にデータベース、テーブル、およびカラムの名前補完を可能にする補完ハッシュを再構築します。(--auto-rehash オプションの説明を参照してください。)

- source file_name, \. file_name

  指定されたファイルを読み取り、その中に含まれているステートメントを実行します。Windows では、パス名区切り文字を / または \\ に指定できます。

- status, \s

  使用中の接続とサーバーに関するステータス情報を表示します。--safe-updates モードで稼働中の場合、status はクエリーに影響する mysql 変数の値も出力します。

- system command, \! command

  デフォルトのコマンドインタプリタを使って指定のコマンドを実行します。

  system コマンドは Unix でのみ機能します。

- tee [file_name], \T [file_name]

  mysql の呼び出し時に --tee オプションを使用することで、ステートメントとその出力をログに記録できます。画面上に表示されるデータはすべて指定されたファイルに追加されます。これはデバッグを行う際にも非常に便利です。mysql では、各ステートメントが終わって次のプロンプトが表示される直前に、結果をファイルにフラッシュします。tee 機能はインタラクティブモードの場合のみ機能します。

  tee コマンドを使用すれば、この機能をインタラクティブに有効にできます。パラメータがない場合、以前のファイルが使用されます。tee ファイルを無効にするには、notee コマンドを使用します。tee を実行すると口ギングが再度有効になります。

- use db_name, \u db_name

  db_name をデフォルトデータベースとして使用します。

- warnings, \W

  各ステートメントのあとの警告の表示を有効にします (存在する場合)。

pager コマンドのヒントを次に記します。

- これを使用してファイルに書き込むと、結果はファイルにのみ送られます。

```
mysql> pager cat > /tmp/log.txt
```

  ページャーとして使用するプログラムのオプションを渡すこともできます。

```
mysql> pager less -n -i -S
```

- 前の例の、-S オプションに注意してください。幅の広いクエリー結果のブラウズの際に非常に便利です。非常に幅の広い結果セットは、画面上では読みにくい場合があります。less に対して -S オプションを指定すると、左右の方向キーを使用して横にスクロールできるため、結果セットが読みやすくなります。また、-S を less 内でインタラクティブに使用して、水平方向のブラウズモードをオン/オフにできます。詳細は、less マニュアルページを参照してください。

```
shell> man less
```

- -F および -X オプションを less で使用すると、出力が 1 画面に収まる場合にプログラムを終了させることができ、これはスクロールが不要なときに便利です。

```
mysql> pager less -n -i -S -F -X
```

- クエリー出力の取り扱いに関する非常に複雑なページャーコマンドを指定できます。

```
mysql> pager cat | tee /dr1/tmp/res.txt \
    | tee /dr2/tmp/res2.txt | less -n -i -S
```

この例では、コマンドはクエリーの結果を /dr1 および /dr2 にマウントされた 2 つの異なるファイルシステムの 2 つの異なるディレクトリ内の 2 つのファイルに送信し、さらに less を使用して結果を画面に表示します。

tee 関数と pager 関数を組み合わせることもできます。tee ファイルを有効にし、pager を less に設定してあれば、less プログラムを使って結果をブラウズしつつ、同時にすべてをファイルに追加できます。pager コマンドと一緒に使用する Unix tee と、mysql に組み込みの tee コマンドの違いは、組み込みの tee は Unix tee がなくても機能するということです。また、組み込みの tee は画面に出力されるものすべてをログに記録しますが、pager と一緒に使用される Unix tee はそこまでログに記録しません。さらに、tee ファイルのロギングは mysql 内からインタラクティブにオン/オフできます。これは一部のクエリーのみをファイルにログするときに有効です。

prompt コマンドはデフォルトの mysql> プロンプトを再構成します。プロンプトを定義するための文字列には、次の特殊なシーケンスを含めることができます。

オプション	説明
\c	ステートメントを発行するたびにインクリメントするカウンタ
\D	現在の日付 (フルで)
\d	デフォルトデータベース
\h	サーバーホスト
\l	現在の区切り文字
\m	現在の時間の分
\n	改行文字
\O	3 文字の形式の現在の月 (Jan、Feb、…)
\o	数字形式の現在の月
\P	am/pm
\p	現在の TCP/IP ポートまたはソケットファイル
\R	現在の時間、24 時間表記 (0-23)
\r	現在の時間、12 時間表記 (1-12)
\S	セミコロン
\s	現在の時間の秒
\t	タブ文字
\U	完全な user_name@host_name アカウント名
\u	ユーザー名
\v	サーバーバージョン
\w	3 文字の形式の現在の曜日 (Mon, Tue, …)
\Y	現在の年 (4 桁)
\y	現在の年 (2 桁)
_	スペース
\	スペース (バックスラッシュのあとにスペースがあります)
\'	単一引用符
\"	二重引用符
\\	リテラルの 「\」 バックスラッシュ文字
\x	x (上記にないすべての 「x」)

プロンプトはいくつかの方法でセットできます。

- 環境変数を使用します。MYSQL_PS1 環境変数をプロンプト文字列に設定できます。例:

```
shell> export MYSQL_PS1="(\u@\h) [\d]> "
```

- コマンド行オプションを使用します。コマンド行で、--prompt オプションを mysql に設定できます。例:

```
shell> mysql --prompt="(\u@\h) [\d]> "
(user@host) [database]>
```

- オプションファイルを使用します。prompt オプションを、ホームディレクトリの /etc/my.cnf または .my.cnf ファイルなど、任意の MySQL オプションファイルの [mysql] グループに設定できます。例:

```
[mysql]
prompt=(\\u@\\h) [\\d]>\\_
```

この例では、バックスラッシュが 2 つあることに注意してください。オプションファイルで prompt オプションを使用してプロンプトを設定する場合、特別なプロンプトオプションを使用するときはバックスラッシュを 2 つ使用することをお勧めします。許可されるプロンプトオプションのセットと、オプションファイルで認識される特殊なエスケープシーケンスのセットには、重複があります。(オプションファイルでのエスケープシーケンスに関するルールはセクション4.2.6「オプションファイルの使用」にリストされています。)単一のバックスラッシュを使用している場合、この重複が問題となる可能性があります。たとえば、\s は現在の秒の値としてではなく、スペースとして解釈されます。次の例は、現在の時間を HH:MM:SS> 形式で含めるように、オプションファイルでプロンプトを定義する方法を示しています。

```
[mysql]
prompt="\\r:\\m:\\s> "
```

- プロンプトをインタラクティブに設定します。prompt コマンド (または \R コマンド) を使用すると、プロンプトをインタラクティブに変更できます。例:

```
mysql> prompt (\u@\h) [\d]>\_
PROMPT set to '(\u@\h) [\d]>\_'
(user@host) [database]>
(user@host) [database]> prompt
Returning to default PROMPT of mysql>
mysql>
```

4.5.1.3 mysql のロギング

Unix では、mysql クライアントはインタラクティブに実行されたステートメントを履歴ファイルにログ記録します。デフォルトでは、このファイルは .mysql_history という名前で、ユーザーのホームディレクトリにあります。別のファイルを指定する場合は、MYSQL_HISTFILE 環境変数値を設定します。

ロギングの方法

ステートメントのロギングは次のように行われます。

- ステートメントは、インタラクティブに実行された場合のみログに記録されます。ステートメントは、たとえばファイルまたはパイプから読み取られる場合はインタラクティブではありません。--batch オプションまたは --execute オプションを使用することによって、ステートメントのロギングを抑制することもできます。
- ステートメントは、「ignore」 リスト内のパターンのいずれかに一致する場合には無視され、ログに記録されません。このリストについてはあとで説明します。
- mysql は、無視されず空でない各ステートメント行を個別にログに記録します。
- 無視されないステートメントが複数の行にまたがる場合 (終端区切り文字を含まない)、mysql は行を連結して完全なステートメントを形成し、改行をスペースに対応付け、結果に区切り文字を付け加えてログに記録します。

その結果、複数行にまたがる入力ステートメントが 2 回ログに記録されることがあります。次の入力について考えます。

```
mysql> SELECT
   -> 'Today is'
   -> ,
   -> CONCAT()
   -> ;
```

この場合、mysql は 「SELECT」、「'Today is'」、「,」、「CONCAT()」、および 「;」 の各行を読み取りながらログに記録します。また、SELECT\n'Today is'\n,\nCURDATE() を SELECT 'Today is' , CURDATE() に対応付け

たあと、区切り文字を追加した完全なステートメントもログに記録します。したがって、ログに記録される出力には次の行があります。

```
SELECT
'Today is'
,
CURDATE()
;
SELECT 'Today is' , CURDATE();
```

MySQL 5.6.8 では、mysql は口ギングのために 「ignore」 リスト内の任意のパターンに一致するステートメントを無視します。デフォルトでは、パターンのリストは "*IDENTIFIED*:*PASSWORD*" で、パスワードを参照するステートメントを無視します。パターンマッチングでは大文字と小文字を区別しません。パターン内では、特殊文字が 2 つあります。

- ? は任意の 1 文字に一致します。
- * はゼロ個以上の文字の任意のシーケンスに一致します。

追加のパターンを指定するには、--histignore オプションを使用するか、または MYSQL_HISTIGNORE 環境変数を設定します。(両方を指定した場合はオプション値が優先されます。)値は、コロン区切りの 1 つまたは複数のパターンのリストで、デフォルトのパターンリストに追加されます。

コマンド行で指定されたパターンは。コマンドインタプリタで特殊な扱いを受けることを防ぐために、引用符で囲むかエスケープ処理を行う必要がある場合があります。たとえば、パスワードを参照するステートメントに加えて UPDATE ステートメントおよび DELETE ステートメントの口ギングを抑制するには、mysql を次のように呼び出します。

```
shell> mysql --histignore="*UPDATE*:*DELETE*"
```

履歴ファイルの制御

.mysql_history ファイルには、パスワードを含む SQL ステートメントのテキストなどの機密情報が書き込まれる場合があるため、制限付きアクセスモードで保護するようにしてください。セクション6.1.2.1「パスワードセキュリティーのためのエンドユーザーガイドライン」を参照してください。

履歴ファイルを維持しない場合は、.mysql_history が存在すればまずそれを削除します。次の手法のいずれかを使用してふたたび作成されないようにします。

- MYSQL_HISTFILE 環境変数を /dev/null に設定します。ログインするたびにこの設定が有効になるようにするには、これをシェルの起動ファイルに置きます。
- .mysql_history を /dev/null へのシンボリックリンクとして作成します。これは一度のみの実行で済みます。

  ```
  shell> ln -s /dev/null $HOME/.mysql_history
  ```

4.5.1.4 mysql サーバー側のヘルプ

```
mysql> help search_string
```

help コマンドに引数を指定した場合、mysql は、サーバー側のヘルプにアクセスして MySQL リファレンスマニュアルの内容から検索するための文字列として引数を使用します。このコマンドを適切に操作するには、mysql データベース内のヘルプテーブルがヘルプトピック情報で初期化されていることが必要です (セクション5.1.10「サーバー側のヘルプ」を参照してください)。

検索文字列に一致するものがない場合、検索は失敗に終わります。

```
mysql> help me

Nothing found
Please try to run 'help contents' for a list of all accessible topics
```

ヘルプカテゴリのリストを閲覧するには help contents を使用してください。

```
mysql> help contents
You asked for help about help category: "Contents"
For more information, type 'help <item>', where <item> is one of the
following categories:
   Account Management
   Administration
   Data Definition
   Data Manipulation
   Data Types
```

```
  Functions
  Functions and Modifiers for Use with GROUP BY
  Geographic Features
  Language Structure
  Plugins
  Storage Engines
  Stored Routines
  Table Maintenance
  Transactions
  Triggers
```

検索文字列に一致するものが複数ある場合は、mysql は一致するトピックのリストを表示します。

```
mysql> help logs
Many help items for your request exist.
To make a more specific request, please type 'help <item>',
where <item> is one of the following topics:
  SHOW
  SHOW BINARY LOGS
  SHOW ENGINE
  SHOW LOGS
```

トピックのヘルプエントリを閲覧するには、そのトピックを検索文字列として使用してください。

```
mysql> help show binary logs
Name: 'SHOW BINARY LOGS'
Description:
Syntax:
SHOW BINARY LOGS
SHOW MASTER LOGS

Lists the binary log files on the server. This statement is used as
part of the procedure described in [purge-binary-logs], that shows how
to determine which logs can be purged.

mysql> SHOW BINARY LOGS;
+---------------+-----------+
| Log_name      | File_size |
+---------------+-----------+
| binlog.000015 |    724935 |
| binlog.000016 |    733481 |
+---------------+-----------+
```

検索文字列にはワイルドカード文字 「%」 および 「_」 を含めることができます。これらは LIKE 演算子で実行されるパターンマッチング演算と同じ意味を持ちます。たとえば、HELP rep% は rep で始まるトピックのリストを返します。

```
mysql> HELP rep%
Many help items for your request exist.
To make a more specific request, please type 'help <item>',
where <item> is one of the following
topics:
  REPAIR TABLE
  REPEAT FUNCTION
  REPEAT LOOP
  REPLACE
  REPLACE FUNCTION
```

4.5.1.5 テキストファイルから SQL ステートメントを実行する

mysql クライアントは、通常次のようにインタラクティブに使用されます。

```
shell> mysql db_name
```

しかし、SQL ステートメントをファイルに入れ、mysql にその入力をファイルから読み取るように指示することも可能です。そのためには、実行するステートメントを含む text_file を作成します。それから、次に示されるように mysql を起動します。

```
shell> mysql db_name < text_file
```

ファイルの最初のステートメントとして USE db_name ステートメントを配置する場合、コマンド行でデータベース名を指定する必要はありません。

```
shell> mysql < text_file
```

mysql がすでに稼働している場合は、source コマンドまたは \. コマンドを使用して SQL スクリプトファイルを実行できます。

```
mysql> source file_name
mysql> \. file_name
```

スクリプトでユーザーに進行状況を表示する場合があります。このためには、次のステートメントを挿入できます。

```
SELECT '<info_to_display>' AS ' ';
```

示されたステートメントは <info_to_display> を出力します。

また、--verbose オプションを付けて mysql を呼び出すこともでき、生成される結果の前に各ステートメントが表示されるようになります。

mysql は、入力ファイルの先頭にある Unicode バイト順マーク (BOM) 文字を無視します。以前は、それらを読み取ってサーバーに送信していたため、構文エラーが発生していました。BOM が存在しても、mysql はデフォルトの文字セットを変更しません。これを行うには、--default-character-set=utf8 などのオプションを付けて mysql を呼び出します。

バッチモードの詳細は、セクション3.5「バッチモードでの MySQL の使用」を参照してください。

4.5.1.6 mysql のヒント

このセクションでは、mysql をさらに効果的に使用するテクニックを紹介します。

入力行の編集

mysql は入力行の編集をサポートし、現在の入力行を修正したり以前の入力行を呼び出したりできます。たとえば、「左矢印」キーおよび「右矢印」キーで現在の入力行内を横方向に移動し、「上矢印」キーおよび「下矢印」キーで以前に入力した行を上下に移動できます。「バックスペース」でカーソルの前の文字を削除でき、新しい文字を入力するとカーソルの位置に挿入されます。行を入力するには、「Enter」を押します。

Windows では、編集キーシーケンスはコンソールウィンドウでコマンドの編集に関してサポートされているものと同じです。Unix では、キーシーケンスは mysql のビルドに使用された入力ライブラリ (たとえば、libedit または readline ライブラリ) に依存します。

libedit ライブラリおよび readline ライブラリのドキュメントは、オンラインで入手できます。所定の入力ライブラリで許可されるキーシーケンスのセットを変更するには、ライブラリ起動ファイルでキーバインドを定義します。これはホームディレクトリにあるファイルで、.editrc は libedit 用、.inputrc は readline 用です。

たとえば libedit では、Control+W は現在のカーソル位置の前にあるものをすべて削除し、Control+U は行全体を削除します。readline では、Control+W はカーソルの前の単語を削除し、Control+U は現在のカーソル位置の前にあるものをすべて削除します。mysql が libedit を使用してビルドされた場合は、これら 2 つのキーに関して readline の動作を好むユーザーは、.editrc ファイルに次の行を置くことができます (必要に応じてファイルを作成します)。

```
bind "^W" ed-delete-prev-word
bind "^U" vi-kill-line-prev
```

現在のキーバインドのセットを表示するには、一時的に bind のみの行を .editrc の最後に置きます。mysql は起動時にバインドを表示します。

Windows における Unicode のサポート

Windows は、コンソールからの読み取りおよび書き込みに、UTF-16LE に基づく API を提供します。MySQL 5.6.2 では、Windows の mysql クライアントはこれらの API を使用できます。5.6.3 では、Windows インストーラは MySQL メニューに MySQL command line client - Unicode という項目を作成します。この項目は、Unicode を使用して MySQL サーバーにコンソール経由で通信するように設定されたプロパティーで mysql クライアントを呼び出します。

このサポートを手動で利用するためには、互換性のある Unicode フォントを使用するコンソール内で mysql を実行し、デフォルト文字セットをサーバーとの通信でサポートされる Unicode 文字セットに設定します。

1. コンソールウィンドウを開きます。
2. コンソールウィンドウプロパティーに移動して「フォント」タブを選択し、Lucida Console またはその他の互換性のある Unicode フォントを選択します。コンソールウィンドウはデフォルトでは Unicode に不適切な DOS ラスターフォントを使用するため、これが必要です。
3. mysql.exe を --default-character-set=utf8 (または utf8mb4) オプションで実行します。utf16le は接続文字セットとしてはサポートされていないため、このオプションが必要です。

これらの変更により、mysql は Windows API を使用して、UTF-16LE を使用してコンソールと通信し、サーバーとは UTF-8 を使用して通信します。(前述のメニュー項目は、フォントと文字セットを今説明したように設定します。)

mysql を起動するたびにこれらのステップを実行しなくてもいいように、mysql.exe を呼び出すショートカットを作成できます。このショートカットは、コンソールフォントを Lucida Console またはその他の互換性のある Unicode フォントに設定し、--default-character-set=utf8 (または utf8mb4) オプションを mysql.exe に渡すようにしてください。

または、コンソールフォントの設定のみを行うショートカットを作成し、文字セットは my.ini ファイルの [mysql] グループで設定します。

```
[mysql]
default-character-set=utf8
```

クエリー結果を縦に表示する

クエリー結果の中には、縦表示の方が、通常の横向きの表形式よりもはるかに読みやすい場合があります。セミコロンの代わりに \G でクエリーを終了することで、クエリーを縦に表示できます。たとえば、多くの場合、改行を含む長いテキスト値は縦の出力の方がはるかに読みやすくなります。

```
mysql> SELECT * FROM mails WHERE LENGTH(txt) < 300 LIMIT 300,1\G
*************************** 1. row ***************************
  msg_nro: 3068
     date: 2000-03-01 23:29:50
time_zone: +0200
mail_from: Monty
    reply: monty@no.spam.com
  mail_to: "Thimble Smith" <tim@no.spam.com>
      sbj: UTF-8
      txt: >>>>> "Thimble" == Thimble Smith writes:

Thimble> Hi.  I think this is a good idea.  Is anyone familiar
Thimble> with UTF-8 or Unicode? Otherwise, I'll put this on my
Thimble> TODO list and see what happens.

Yes, please do that.

Regards,
Monty
     file: inbox-jani-1
     hash: 190402944
1 row in set (0.09 sec)
```

--safe-updates オプションの使用

初心者にとって、使いやすい起動オプションは --safe-updates (または同じ効果のある --i-am-a-dummy) です。これは DELETE FROM tbl_name ステートメントを発行したが、WHERE 句を忘れてしまった場合に役立ちます。通常、このようなステートメントはテーブルからすべての行を削除します。--safe-updates を使用すると、行を特定するキー値を指定しないと、行を消去できません。これにより、事故を予防します。

--safe-updates オプションを使用すると、mysql は MySQL サーバーに接続した際に次のステートメントを発行します。

```
SET sql_safe_updates=1, sql_select_limit=1000, max_join_size=1000000;
```

セクション5.1.4「サーバーシステム変数」を参照してください。

SET ステートメントには次の効果があります。

- UPDATE ステートメントまたは DELETE ステートメントは、WHERE 句にキーの制約を指定するか、LIMIT 句を提供するか (または両方) しないかぎり実行を許可されません。例:

  ```
  UPDATE tbl_name SET not_key_column=val WHERE key_column=val;

  UPDATE tbl_name SET not_key_column=val LIMIT 1;
  ```

- サーバーは、ステートメントに LIMIT 句が含まれていないかぎり、すべての大規模な SELECT の結果を 1,000 行に制限します。
- サーバーは、1,000,000 を超える行の組み合わせを確認しなければいけない可能性のある、複数テーブルの SELECT ステートメントを中止します。

1,000 および 1,000,000 以外の制限を指定するには、--select_limit オプションおよび --max_join_size オプションを使用してデフォルトをオーバーライドできます。

```
shell> mysql --safe-updates --select_limit=500 --max_join_size=10000
```

mysql の自動再接続を無効にする

ステートメントの送信中にサーバーとの接続が切断された場合、mysql クライアントはただちに自動的にサーバーに一度再接続してステートメントを再度送信しようとします。ただし、mysql が再接続に成功しても、最初の接続は終了し、前セッションのオブジェクトと設定は失われます。この中には、一時テーブル、自動コミットモード、およびユーザー定義変数やセッション変数が含まれます。また、現トランザクションはロールバックします。この動作は危険な場合があります。たとえば、次の例では、サーバーはユーザーの了解なしに、最初のステートメントと 2 番目のステートメントの間にシャットダウンして再起動させられています。

```
mysql> SET @a=1;
Query OK, 0 rows affected (0.05 sec)

mysql> INSERT INTO t VALUES(@a);
ERROR 2006: MySQL server has gone away
No connection. Trying to reconnect...
Connection id:    1
Current database: test

Query OK, 1 row affected (1.30 sec)

mysql> SELECT * FROM t;
+------+
| a    |
+------+
| NULL |
+------+
1 row in set (0.05 sec)
```

@a ユーザー変数は接続とともに失われ、再接続後は未定義です。接続が失われた際に、mysql が エラーで終了することが望ましい場合、mysql クライアントを --skip-reconnect オプションで起動できます。

自動再接続および再接続時の状態情報への影響の詳細は、セクション23.7.16「自動再接続動作の制御」を参照してください。

4.5.2 mysqladmin — MySQL サーバーの管理を行うクライアント

mysqladmin は管理操作を実行するためのクライアントです。サーバーの構成や現在のステータスの確認、データベースの作成および削除、およびその他の用途に使用できます。

mysqladmin は次のように起動します。

```
shell> mysqladmin [options] command [command-arg] [command [command-arg]] ...
```

mysqladmin は次のコマンドをサポートします。コマンドの中にはコマンド名のあとに引数を取るものもあります。

- create db_name

  db_name という名前の新しいデータベースを作成します。

- debug

  エラーログにデバッグ情報を書き込むようにサーバーに指示します。この情報の形式と内容は変更されることがあります。

  これにはイベントスケジューラの情報が含まれます。セクション20.4.5「イベントスケジューラのステータス」を参照してください。

- drop db_name

  db_name という名前のデータベースとそのテーブルをすべて削除します。

- extended-status

  サーバーステータス変数とその値を表示します。

- flush-hosts

  ホストキャッシュ内の情報をすべてフラッシュします。

- flush-logs

ログをすべてフラッシュします。

- flush-privileges

  付与テーブルをリロードします (reload と同じ)。

- flush-status

  ステータス変数をクリアします。

- flush-tables

  テーブルをすべてフラッシュします。

- flush-threads

  スレッドキャッシュをフラッシュします。

- kill id,id,...

  サーバースレッドを強制終了します。複数のスレッド ID 値を指定する場合、リストにはスペースが存在してはいけません。

- old-password new-password

  これは password コマンドと似ていますが、古い (4.1 以前) パスワードハッシュ形式を使用してパスワードを保存します。(セクション6.1.2.4「MySQL でのパスワードハッシュ」を参照してください。)

- password new-password

  新しいパスワードを設定します。これにより、サーバーへの接続に使う mysqladmin のアカウントパスワードを new-password に変更します。したがって、次に同じアカウントを使用して mysqladmin (またはほかのクライアントプログラム) を起動するとき、新しいパスワードを指定することが必要になります。

  new-password の値がスペースまたはコマンドインタプリタにとって特殊なその他の文字を含んでいる場合、引用符で囲む必要があります。Windows では、単一引用符ではなく二重引用符を必ず使用してください。単一引用符はパスワードから取り除かれず、むしろパスワードの一部として解釈されます。例:

```
shell> mysqladmin password "my new password"
```

  MySQL 5.6 では、password コマンドに続いて新しいパスワードを省略できます。この場合、mysqladmin はパスワード値を要求し、パスワードをコマンド行で指定するのを避けることができます。パスワード値は、password が mysqladmin コマンド行の最後のコマンドである場合にかぎって省略できます。そうでない場合、次の引数がパスワードとみなされます。

  > **注意**
  >
  > サーバーが --skip-grant-tables オプションを使用して起動された場合は、このコマンドを使用しないでください。パスワードの変更は適用されません。これは、同じコマンド行で password コマンドの前に flush-privileges コマンドを指定して付与テーブルを再度有効にする場合にも当てはまります (フラッシュ処理は接続後に行われるため)。ただし、mysqladmin flush-privileges コマンドを実行して付与テーブルを再度有効にしてから、個別に mysqladmin password コマンドを実行してパスワードを変更することは可能です。

- ping

  サーバーが使用可能かどうかをチェックします。サーバーが稼働中の場合は mysqladmin のリターンステータスは 0 になり、稼働していない場合は 1 になります。Access denied のようなエラーの場合でも 0 となります。これは、サーバーは稼働しているが接続を拒否したことを意味しており、サーバーが稼働していない状態とは異なるからです。

- processlist

  アクティブなサーバースレッドのリストを表示します。これは SHOW PROCESSLIST ステートメントの出力と同様です。--verbose オプションが指定されている場合、出力は SHOW FULL PROCESSLIST の出力と同様です。(セクション13.7.5.30「SHOW PROCESSLIST 構文」を参照してください。)

- reload

付与テーブルをリロードします。

- refresh

  全テーブルをフラッシュし、ログファイルを閉じて、開きます。

- shutdown

  サーバーを停止します。

- start-slave

  スレーブサーバーのレプリケーションを開始します。

- status

  短いサーバーステータスメッセージを表示します。

- stop-slave

  スレーブサーバーのレプリケーションを停止します。

- variables

  サーバーシステム変数とその値を表示します。

- version

  サーバーからのバージョン情報を表示します。

すべてのコマンドは一意のプリフィクスに省略できます。例:

```
shell> mysqladmin proc stat
+----+-------+-----------+----+---------+------+-------+------------------+
| Id | User  | Host      | db | Command | Time | State | Info             |
+----+-------+-----------+----+---------+------+-------+------------------+
| 51 | monty | localhost |    | Query   | 0    |       | show processlist |
+----+-------+-----------+----+---------+------+-------+------------------+
Uptime: 1473624  Threads: 1  Questions: 39487
Slow queries: 0  Opens: 541  Flush tables: 1
Open tables: 19  Queries per second avg: 0.0268
```

mysqladmin status コマンドの結果は次の値を表示します。

- Uptime

  MySQL サーバーが稼働している秒数。

- Threads

  アクティブスレッド (クライアント) の数です。

- Questions

  サーバーが起動して以来クライアントから寄せられた質問 (クエリー) の数です。

- Slow queries

  long_query_time 秒よりも時間を要したクエリーの数。セクション5.2.5「スロークエリーログ」を参照してください。

- Opens

  サーバーによって開かれたテーブルの数。

- Flush tables

  サーバーが実行した flush-*、refresh、および reload コマンドの数です。

- Open tables

  現在開いているテーブルの数。

Unix ソケットファイルを使用してローカルサーバーに接続する際に mysqladmin shutdown を実行した場合、mysqladmin はサーバーのプロセス ID ファイルが取り除かれるまで待ちます。これはサーバーが正しく停止したことを確認するためです。

mysqladmin は次のオプションをサポートします。これらはコマンド行またはオプションファイルの [mysqladmin] グループおよび [client] グループで指定できます。MySQL プログラムによって使用されるオプションファイルの詳細については、セクション4.2.6「オプションファイルの使用」を参照してください。

表 4.6 mysqladmin のオプション

オプション名	説明	導入
--bind-address	指定されたネットワークインタフェースを使用して MySQL サーバーに接続	5.6.1
--compress	クライアントとサーバー間で送信される情報をすべて圧縮	
--connect_timeout	接続タイムアウトまでの秒数	
--count	繰り返されるコマンド実行での反復回数	
--debug	デバッグのログを書き込み	
--debug-check	プログラムの終了時にデバッグ情報を出力	
--debug-info	プログラムの終了時に、デバッグ情報、メモリー、および CPU の統計を出力	
--default-auth	使用する認証プラグイン	
--default-character-set	デフォルト文字セットを指定	
--defaults-extra-file	通常のオプションファイルに加えてオプションファイルを読み取る	
--defaults-file	指名されたオプションファイルのみを読み取る	
--defaults-group-suffix	オプショングループのサフィクス値	
--enable-cleartext-plugin	平文の認証プラグインを有効化	5.6.7
--force	SQL エラーが発生しても続行	
--help	ヘルプメッセージを表示して終了	
--host	指定されたホスト上で MySQL サーバーに接続	
--login-path	ログインパスオプションを .mylogin.cnf から読み取り	5.6.6
--no-beep	エラー時に音を発生させない	
--no-defaults	オプションファイルを読み取らない	
--password	サーバーに接続する際に使用するパスワード	
--pipe	Windows で、名前付きパイプを使用してサーバーに接続	
--plugin-dir	プラグインがインストールされているディレクトリ	
--port	接続に使用する TCP/IP ポート番号	
--print-defaults	デフォルトを出力	
--protocol	使用する接続プロトコル	
--relative	--sleep オプションとともに使用された場合、現在値と以前の値の差異を表示	
--secure-auth	古い (4.1.1 より前の) 形式でサーバーにパスワードを送信しない	5.6.17
--shared-memory-base-name	共有メモリー接続に使用する共有メモリーの名前	
--shutdown_timeout	サーバーのシャットダウンを待機する最大秒数	
--silent	サイレントモード	
--sleep	コマンドを反復実行し、その間に delay 秒間スリープ	
--socket	ローカルホストへの接続で、使用する Unix ソケットファイル	
--ssl	接続に SSL を有効化	
--ssl-ca	信頼された SSL CA のリストを含むファイルのパス	

オプション名	説明	導入
--ssl-capath	信頼された SSL CA の PEM 形式の証明書を含むディレクトリのパス	
--ssl-cert	PEM 形式の X509 証明書を含むファイルのパス	
--ssl-cipher	SSL の暗号化に使用される、許可された暗号のリスト	
--ssl-crl	証明書失効リストを含むファイルのパス	5.6.3
--ssl-crlpath	証明書失効リストファイルを含むディレクトリのパス	5.6.3
--ssl-key	PEM 形式の X509 鍵を含むファイルのパス	
--ssl-verify-server-cert	サーバーへの接続時に、サーバーの証明書内のコモンネーム値をホスト名に対して検証	
--user	サーバーへの接続時に使用する MySQL ユーザー名	
--verbose	冗長モード	
--version	バージョン情報を表示して終了	
--vertical	クエリー出力行を垂直形式で出力 (カラム値ごとに 1 行)	
--wait	接続が確立できない場合、中止せずに待機してからリトライ	

- --help, -?

  ヘルプメッセージを表示して終了します。

- --bind-address=ip_address

  複数のネットワークインタフェースを持つコンピュータで、このオプションを使用して、MySQL サーバーへの接続に使用するインタフェースを選択します。

  このオプションは MySQL 5.6.1 からサポートされています。

- --character-sets-dir=path

  文字セットがインストールされているディレクトリ。セクション10.5「文字セットの構成」を参照してください。

- --compress, -C

  クライアントとサーバーの両方が圧縮をサポートしている場合、その間で送受信される情報をすべて圧縮します。

- --count=N, -c N

  --sleep オプションが指定されている場合、繰り返し実行されるコマンドの反復実行の数。

- --debug[=debug_options], -# [debug_options]

  デバッグのログを書き込みます。一般的な debug_options 文字列は d:t:o,file_name です。デフォルトは d:t:o,/tmp/mysqladmin.trace です。

- --debug-check

  プログラムの終了時に、デバッグ情報を出力します。

- --debug-info

  プログラムの終了時に、デバッグ情報とメモリーおよび CPU 使用率の統計を出力します。

- --default-auth=plugin

  使用するクライアント側の認証プラグイン。セクション6.3.7「プラガブル認証」を参照してください。

- --default-character-set=charset_name

  charset_name をデフォルト文字セットとして使用します。セクション10.5「文字セットの構成」を参照してください。

- --defaults-extra-file=file_name

このオプションファイルは、グローバルオプションファイルのあとに読み取りますが、(UNIX では) ユーザーオプションファイルの前に読み取るようにしてください。ファイルが存在しないかアクセスできない場合、エラーが発生します。file_name は、フルパス名でなく相対パス名として指定された場合、現行ディレクトリを基準にして解釈されます。

- --defaults-file=file_name

  指定されたオプションファイルのみ使用します。ファイルが存在しないかアクセスできない場合、エラーが発生します。file_name は、フルパス名でなく相対パス名として指定された場合、現行ディレクトリを基準にして解釈されます。

- --defaults-group-suffix=str

  通常のオプショングループだけでなく、通常の名前に str のサフィクスが付いたグループも読み取ります。たとえば、mysqladmin は通常 [client] グループおよび [mysqladmin] グループを読み取ります。--defaults-group-suffix=_other オプションを指定した場合、mysqladmin は [client_other] グループおよび [mysqladmin_other] グループも読み取ります。

- --enable-cleartext-plugin

  mysql_clear_password 平文認証プラグインを有効にします。(セクション6.3.8.7「クライアント側のクリアテキスト認証プラグイン」を参照してください。)このオプションは MySQL 5.6.7 で追加されました。

- --force, -f

  drop db_name コマンドの確認を求めません。複数のコマンドで、エラーが発生しても続けます。

- --host=host_name, -h host_name

  指定されたホストの MySQL サーバーに接続します。

- --login-path=name

  指名されたログインパスから .mylogin.cnf ログインファイルのオプションを読み取ります。「ログインパス」は、host、user、および password という限定されたオプションのセットのみを許可するオプショングループです。ログインパスは、サーバーホストおよびそのサーバーで認証するための認証情報を示す値のセットであると考えてください。ログインパスファイルを作成するには、mysql_config_editor ユーティリティーを使用します。セクション4.6.6「mysql_config_editor — MySQL 構成ユーティリティー」を参照してください。このオプションは MySQL 5.6.6 で追加されました。

- --no-beep, -b

  サーバーへの接続に失敗するなどのエラーの際にデフォルトで鳴らされる警告音を抑制します。

- --no-defaults

  オプションファイルを読み取りません。オプションファイルから不明のオプションを読み取ることが原因でプログラムの起動に失敗する場合、--no-defaults を使用して、オプションを読み取らないようにすることができます。

  例外として、.mylogin.cnf ファイルは、存在する場合はすべての場合に読み取られます。これにより、--no-defaults が使用された場合でも、コマンド行よりも安全な方法でパスワードを指定できます。(.mylogin.cnf は mysql_config_editor ユーティリティーによって作成されます。セクション4.6.6「mysql_config_editor — MySQL 構成ユーティリティー」を参照してください)。

- --password[=password], -p[password]

  サーバーに接続する際に使用するパスワードです。短いオプション形式 (-p) を使用した場合は、オプションとパスワードの間にスペースを置くことはできません。コマンド行で、--password オプションまたは -p オプションに続けて password の値を指定しなかった場合、mysqladmin はそれを要求します。

  コマンド行でのパスワード指定は、セキュアでないと考えるべきです。セクション6.1.2.1「パスワードセキュリティーのためのエンドユーザーガイドライン」を参照してください。オプションファイルを使用すれば、コマンド行でパスワードを指定することを回避できます。

- --pipe, -W

  Windows で、名前付きパイプを使用してサーバーに接続します。このオプションは、サーバーが名前付きパイプ接続をサポートしている場合にのみ適用されます。

- --plugin-dir=path

  プラグインを検索するディレクトリ。--default-auth オプションを使用して認証プラグインを指定したが、mysqladmin がそれを検出できない場合は、このオプションを指定しなければならない可能性があります。セクション6.3.7「プラガブル認証」を参照してください。

- --port=port_num, -P port_num

  接続に使用する TCP/IP ポート番号。

- --print-defaults

  プログラム名と、オプションファイルから受け取るすべてのオプションを出力します。

- --protocol={TCP|SOCKET|PIPE|MEMORY}

  サーバーへの接続に使用する接続プロトコル。このオプションは、ほかの接続パラメータによって、必要なプロトコル以外のものが通常使用される場合に役立ちます。許可される値の詳細は、セクション4.2.2「MySQL サーバーへの接続」を参照してください。

- --relative, -r

  --sleep オプションとともに使用された場合、現在値と以前の値の差異を表示します。このオプションは extended-status コマンドとのみ機能します。

- --secure-auth

  古い (4.1 より前の) 形式でサーバーにパスワードを送信しません。これにより、新しいパスワード形式を使用するサーバー以外への接続を防ぎます。このオプションはデフォルトで有効です。無効にするには --skip-secure-auth を使用します。このオプションは MySQL 5.6.17 で追加されました。

  > **注記**
  >
  > 4.1 より前のハッシュ方式を使用するパスワードはネイティブのパスワードハッシュ方式を使用するパスワードよりもセキュアでないため、使用しないようにしてください。4.1 よりも前のパスワードは非推奨であり、これらのサポートは今後の MySQL リリースで削除される予定です。アカウントのアップグレード手順については、セクション6.3.8.3「4.1 よりも前のパスワードハッシュ方式と mysql_old_password プラグインからの移行」を参照してください。

- --shared-memory-base-name=name

  Windows で、共有メモリーを使用して作成されるローカルサーバーへの接続の共有メモリー名。デフォルト値は MYSQL です。共有メモリー名では大文字と小文字を区別します。

  共有メモリー接続を可能にするには、サーバーは --shared-memory オプションで起動する必要があります。

- --silent, -s

  サーバーとの接続が確立できない場合、警告なしで終了します。

- --sleep=delay, -i delay

  コマンドを繰り返し実行し、その間 delay 秒間スリープします。--count オプションは反復回数を決定します。--count が指定されていない場合は、mysqladmin は中断されるまでいつまでもコマンドを実行します。

- --socket=path, -S path

  localhost への接続用に使用する、Unix ソケットファイル、または Windows では使用する名前付きパイプの名前。

- --ssl*

  --ssl で始まるオプションは、SSL を使用してサーバーに接続することを許可するかどうかを指定し、SSL 鍵および証明書を検索する場所を指定します。セクション6.3.10.4「SSL コマンドのオプション」を参照してください。

- --user=user_name, -u user_name

  サーバーへの接続時に使用する MySQL ユーザー名。

- --verbose, -v

  冗長モード。プログラムの動作についてより多くの情報を出力します。

- --version, -V

  バージョン情報を表示して終了します。

- --vertical, -E

  出力を縦に出力します。これは --relative と同様ですが、出力を縦に出力します。

- --wait[=count], -w[count]

  接続が確立できない場合、中止せずに待機してからリトライします。count 値が指定されている場合、リトライ回数を示します。デフォルトは 1 回です。

--var_name=value を使用して、次の変数も設定できます。

- connect_timeout

  接続タイムアウトまでの最大秒数。デフォルト値は 43200 秒 (12 時間) です。

- shutdown_timeout

  サーバーのシャットダウンを待機する最大秒数。デフォルト値は 3600 秒 (1 時間) です。

4.5.3 mysqlcheck — テーブル保守プログラム

mysqlcheck クライアントでは、テーブルの保守 (テーブルの検査、修復、最適化、分析) を実行します。

各テーブルは処理中にロックされるため、ほかのセッションでは利用できません。ただし、検査操作ではテーブルは READ ロックでのみロックされます (READ ロックおよび WRITE ロックの詳細は、セクション 13.3.5「LOCK TABLES および UNLOCK TABLES 構文」を参照してください)。テーブルの保守処理は、特に大きなテーブルでは長い時間を要する可能性があります。--databases オプションまたは --all-databases オプションを使用して 1 つまたは複数のデータベースに含まれるすべてのテーブルを処理する場合は、mysqlcheck の呼び出しに長い時間がかかる可能性があります。(mysql_upgrade についても同じことが言えますが、これはこのプログラムが mysqlcheck を呼び出してすべてのテーブルを検査し、必要に応じて修復するからです。)

mysqlcheck の機能は myisamchk と同様ですが、作動方法が異なります。主な作動方法の違いは、mysqlcheck は mysqld サーバーが稼働中のときに使用されなければならないのに対し、myisamchk はこのサーバーが稼働していないときに使用されるべきであるという点です。mysqlcheck を使用することの利点は、テーブルの保守を行うためにサーバーを停止する必要がないことです。

mysqlcheck は SQL ステートメント CHECK TABLE、REPAIR TABLE、ANALYZE TABLE、および OPTIMIZE TABLE をユーザーにとって便利な方法で使用します。実行する操作に対してどのステートメントを使用するか決定し、実行のためサーバーにステートメントを送信します。各ステートメントがどのストレージエンジンと機能するかは、セクション13.7.2「テーブル保守ステートメント」のステートメントの説明を参照してください。

MyISAM ストレージエンジンでは 4 つの保守処理をすべてサポートしているため、mysqlcheck を使用すると、MyISAM テーブルに対してそれらのどの保守処理も実行できます。ほかのストレージエンジンは必ずしもすべての操作をサポートしているとはかぎりません。そのような場合、エラーメッセージが表示されます。たとえば、test.t が MEMORY テーブルの場合、検査しようとすると次の結果が生成されます。

```
shell> mysqlcheck test t
test.t
note     : The storage engine for the table doesn't support check
```

mysqlcheck がテーブルを修復できない場合、手動でテーブルを修復する方法についてはセクション2.11.4「テーブルまたはインデックスの再作成または修復」を参照してください。たとえば、InnoDB テーブルがこれに当たります。このテーブルは CHECK TABLE で検査できますが、REPAIR TABLE で修復はできません。

> **注意**
>
> テーブルの修復操作を実行する前に、テーブルのバックアップを作成することをお勧めします。状況によっては、この操作のためにデータ損失が発生することがあります。考えられる原因としては、ファイルシステムのエラーなどがありますがこれに限りません。

一般的に、mysqlcheck を起動するには 3 つの方法があります。

```
shell> mysqlcheck [options] db_name [tbl_name ...]
shell> mysqlcheck [options] --databases db_name ...
shell> mysqlcheck [options] --all-databases
```

db_name のあとにテーブルを指定しない場合、または--databases オプションまたは --all-databases オプションを使用している場合、データベース全体が検査されます。

ほかのクライアントプログラムに比べ、mysqlcheck は特別な機能があります。テーブル検査のデフォルト動作 (--check) はバイナリの名前を変更することで変えられます。テーブルをデフォルトで修復するツールが必要な場合、mysqlrepair という名前で mysqlcheck のコピーを作成するか、mysqlrepair という名前で mysqlcheck へのシンボリックリンクを作成してください。mysqlrepair を起動すれば、テーブルが修復されます。

次の表に示す名前は、mysqlcheck のデフォルト動作を変更するために使用できます。

コマンド	意味
mysqlrepair	デフォルトオプションは --repair
mysqlanalyze	デフォルトオプションは --analyze
mysqloptimize	デフォルトオプションは --optimize

mysqlcheck は次のオプションをサポートします。これらはコマンド行またはオプションファイルの [mysqlcheck] グループおよび [client] グループで指定できます。MySQL プログラムによって使用されるオプションファイルの詳細については、セクション4.2.6「オプションファイルの使用」を参照してください。

表 4.7 mysqlcheck のオプション

オプション名	説明	導入
--all-databases	データベース内のテーブルをすべて確認	
--all-in-1	各データベースに対して、そのデータベースのすべてのテーブルを指定する単一のステートメントを実行	
--analyze	テーブルを分析	
--auto-repair	確認されたテーブルが破損していた場合、自動的に修復	
--bind-address	指定されたネットワークインタフェースを使用して MySQL サーバーに接続	5.6.1
--character-sets-dir	文字セットがインストールされているディレクトリ	
--check	テーブルにエラーがないか確認	
--check-only-changed	最後に行われた検査以降に変更されたテーブルのみをチェック	
--check-upgrade	CHECK TABLE を FOR UPGRADE オプションで呼び出し	
--compress	クライアントとサーバー間で送信される情報をすべて圧縮	
--databases	指定されたデータベース内のテーブルをすべて処理	
--debug	デバッグの口グを書き込み	
--debug-check	プログラムの終了時にデバッグ情報を出力	
--debug-info	プログラムの終了時に、デバッグ情報、メモリー、および CPU の統計を出力	
--default-auth	使用する認証プラグイン	5.6.2
--default-character-set	デフォルト文字セットを指定	
--defaults-extra-file	通常のオプションファイルに加えてオプションファイルを読み取る	
--defaults-file	指名されたオプションファイルのみを読み取る	
--defaults-group-suffix	オプショングループのサフィクス値	
--extended	テーブルをチェックして修復	
--fast	正しく閉じられていないテーブルのみを確認	
--fix-db-names	データベース名を 5.1 形式に変換	
--fix-table-names	テーブル名を 5.1 形式に変換	
--force	SQL エラーが発生しても続行	

オプション名	説明	導入
--help	ヘルプメッセージを表示して終了	
--host	指定されたホスト上で MySQL サーバーに接続	
--login-path	ログインパスオプションを .mylogin.cnf から読み取り	5.6.6
--medium-check	--extended 操作よりも速いチェックを実行	
--no-defaults	オプションファイルを読み取らない	
--optimize	テーブルを最適化	
--password	サーバーに接続する際に使用するパスワード	
--pipe	Windows で、名前付きパイプを使用してサーバーに接続	
--plugin-dir	プラグインがインストールされているディレクトリ	5.6.2
--port	接続に使用する TCP/IP ポート番号	
--print-defaults	デフォルトを出力	
--protocol	使用する接続プロトコル	
--quick	最速のチェック方法	
--repair	一意ではではない一意なキー以外のほぼすべてを修正できる修復を実行	
--secure-auth	古い (4.1.1 より前の) 形式でサーバーにパスワードを送信しない	5.6.17
--shared-memory-base-name	共有メモリー接続に使用する共有メモリーの名前	
--silent	サイレントモード	
--skip-database	実行される操作からこのデータベースを除外	5.6.11
--socket	ローカルホストへの接続で、使用する Unix ソケットファイル	
--ssl	接続に SSL を有効化	
--ssl-ca	信頼された SSL CA のリストを含むファイルのパス	
--ssl-capath	信頼された SSL CA の PEM 形式の証明書を含むディレクトリのパス	
--ssl-cert	PEM 形式の X509 証明書を含むファイルのパス	
--ssl-cipher	SSL の暗号化に使用される、許可された暗号のリスト	
--ssl-crl	証明書失効リストを含むファイルのパス	5.6.3
--ssl-crlpath	証明書失効リストファイルを含むディレクトリのパス	5.6.3
--ssl-key	PEM 形式の X509 鍵を含むファイルのパス	
--ssl-verify-server-cert	サーバーへの接続時に、サーバーの証明書内のコモンネーム値をホスト名に対して検証	
--tables	--databases オプションまたは -B オプションをオーバーライド	
--use-frm	MyISAM テーブルの修復操作用	
--user	サーバーへの接続時に使用する MySQL ユーザー名	
--verbose	冗長モード	
--version	バージョン情報を表示して終了	
--write-binlog	ANALYZE ステートメント、OPTIMIZE ステートメント、REPAIR ステートメントをバイナリログに記録。--skip-write-binlog は、NO_WRITE_TO_BINLOG をこれらのステートメントに追加します。	

- --help, -?

  ヘルプメッセージを表示して終了します。

- --all-databases, -A

  データベース内のテーブルをすべて検査します。これは、INFORMATION_SCHEMA データベースおよび performace_schema データベースがダンプされない点を除き、I--databases オプションを使用してコマンド行にすべてのデータベースを指定するのと同じです。これらは、--databases オプションで明示的に指名することでダンプできます。

- --all-in-1, -1

  各テーブルに対してステートメントを発行する代わりに、各データベースに対して、そのデータベースから処理されるすべてのテーブルを指名する単一のステートメントを実行します。

- --analyze, -a

  テーブルを分析します。

- --auto-repair

  確認されたテーブルが壊れていた場合、自動的に修復します。必要な修復はすべてのテーブルが確認されたあとに実行されます。

- --bind-address=ip_address

  複数のネットワークインタフェースを持つコンピュータで、このオプションを使用して、MySQL サーバーへの接続に使用するインタフェースを選択します。

  このオプションは MySQL 5.6.1 からサポートされています。

- --character-sets-dir=path

  文字セットがインストールされているディレクトリ。セクション10.5「文字セットの構成」を参照してください。

- --check, -c

  テーブルにエラーがないか確認します。これはデフォルトの動作です。

- --check-only-changed, -C

  最後に行われた検査以降に変更されたテーブル、または適切に閉じられなかったテーブルのみを検査します。

- --check-upgrade, -g

  CHECK TABLE を FOR UPGRADE オプションで呼び出し、サーバーの現在のバージョンとの互換性のないテーブルがあるか検査します。このオプションにより、--fix-db-names オプションおよび --fix-table-names オプションが自動的に有効になります。

- --compress

  クライアントとサーバーの両方が圧縮をサポートしている場合、その間で送受信される情報をすべて圧縮します。

- --databases, -B

  指定されたデータベース内のテーブルをすべて処理します。通常、mysqlcheck はコマンド行の最初の名前引数をデータベース名として、それに続く名前をテーブル名として処理します。このオプションを使用すると、名前引数をすべてデータベース名として処理します。

  このオプションは、INFORMATION_SCHEMA データベースおよび performace_schema データベースのダンプに使用できます。これらは通常、--all-databases オプションを指定してもダンプされません。(--skip-lock-tables オプションも使用してください。)

- --debug[=debug_options], -# [debug_options]

  デバッグの口グを書き込みます。一般的な debug_options 文字列は d:t:o,file_name です。デフォルトは d:t:o です。

- --debug-check

  プログラムの終了時に、デバッグ情報を出力します。

- --debug-info

  プログラムの終了時に、デバッグ情報とメモリーおよび CPU 使用率の統計を出力します。

- --default-character-set=charset_name

  charset_name をデフォルト文字セットとして使用します。セクション10.5「文字セットの構成」を参照してください。

- --defaults-extra-file=file_name

  このオプションファイルは、グローバルオプションファイルのあとに読み取りますが、(UNIX では) ユーザーオプションファイルの前に読み取るようにしてください。ファイルが存在しないかアクセスできない場合、エラーが発生します。file_name は、フルパス名でなく相対パス名として指定された場合、現行ディレクトリを基準にして解釈されます。

- --defaults-file=file_name

  指定されたオプションファイルのみ使用します。ファイルが存在しないかアクセスできない場合、エラーが発生します。file_name は、フルパス名でなく相対パス名として指定された場合、現行ディレクトリを基準にして解釈されます。

- --defaults-group-suffix=str

  通常のオプショングループだけでなく、通常の名前に str のサフィクスが付いたグループも読み取ります。たとえば、mysqlcheck は通常 [client] グループおよび [mysqlcheck] グループを読み取ります。--defaults-group-suffix=_other オプションを指定した場合、mysqlcheck は [client_other] グループおよび [mysqlcheck_other] グループも読み取ります。

- --extended, -e

  テーブルの検査にこのオプションを使用している場合、100% 整合性があることが保証されますが、時間がかかります。

  このオプションを使用してテーブルを修復している場合、修復作業に時間がかかるだけでなく、大量のガベージ行を生成することもあります。

- --default-auth=plugin

  使用するクライアント側の認証プラグイン。セクション6.3.7「プラガブル認証」を参照してください。

  このオプションは MySQL 5.6.2 で追加されました。

- --fast, -F

  正しく閉じられていないテーブルのみを検査します。

- --fix-db-names

  データベース名を 5.1 形式に変換します。特殊文字を含むデータベース名のみ影響を受けます。

- --fix-table-names

  テーブル名を 5.1 形式に変換します。特殊文字を含むテーブル名のみ影響を受けます。このオプションはビューにも適用されます。

- --force, -f

  SQL エラーが発生しても続行します。

- --host=host_name, -h host_name

  指定されたホストの MySQL サーバーに接続します。

- --login-path=name

  指名されたログインパスから .mylogin.cnf ログインファイルのオプションを読み取ります。「ログインパス」は、host、user、および password という限定されたオプションのセットのみを許可するオプショングループです。ログインパスは、サーバーホストおよびそのサーバーで認証するための認証情報を示す値のセットであると考えてください。ログインパスファイルを作成するには、mysql_config_editor ユーティリティーを使用

します。セクション4.6.6「mysql_config_editor — MySQL 構成ユーティリティー」を参照してください。このオプションは MySQL 5.6.6 で追加されました。

- --medium-check, -m

  --extended 操作よりも高速な検査を実行します。これはすべてのエラーの 99.99% のみを確認し、ほとんどの場合はこれで十分でしょう。

- --no-defaults

  オプションファイルを読み取りません。オプションファイルから不明のオプションを読み取ることが原因でプログラムの起動に失敗する場合、--no-defaults を使用して、オプションを読み取らないようにできます。

  例外として、.mylogin.cnf ファイルは、存在する場合はすべての場合に読み取られます。これにより、--no-defaults が使用された場合にも、コマンド行よりも安全な方法でパスワードを指定できます。(.mylogin.cnf は mysql_config_editor ユーティリティーによって作成されます。セクション4.6.6「mysql_config_editor — MySQL 構成ユーティリティー」を参照してください)。

- --optimize, -o

  テーブルを最適化します。

- --password[=password], -p[password]

  サーバーに接続する際に使用するパスワードです。短いオプション形式 (-p) を使用した場合は、オプションとパスワードの間にスペースを置くことはできません。コマンド行で、--password オプションまたは -p オプションに続けて password の値を指定しなかった場合、mysqlcheck はそれを要求します。

  コマンド行でのパスワード指定は、セキュアでないと考えるべきです。セクション6.1.2.1「パスワードセキュリティーのためのエンドユーザーガイドライン」を参照してください。オプションファイルを使用すれば、コマンド行でパスワードを指定することを回避できます。

- --pipe, -W

  Windows で、名前付きパイプを使用してサーバーに接続します。このオプションは、サーバーが名前付きパイプ接続をサポートしている場合にのみ適用されます。

- --plugin-dir=path

  プラグインを検索するディレクトリ。--default-auth オプションを使用して認証プラグインを指定したが、mysqlcheck がそれを検出できない場合は、このオプションを指定しなければならない可能性があります。セクション6.3.7「プラガブル認証」を参照してください。

  このオプションは MySQL 5.6.2 で追加されました。

- --port=port_num, -P port_num

  接続に使用する TCP/IP ポート番号。

- --print-defaults

  プログラム名と、オプションファイルから受け取るすべてのオプションを出力します。

- --protocol={TCP|SOCKET|PIPE|MEMORY}

  サーバーへの接続に使用する接続プロトコル。このオプションは、ほかの接続パラメータによって、必要なプロトコル以外のものが通常使用される場合に役立ちます。許可される値の詳細は、セクション4.2.2「MySQL サーバーへの接続」を参照してください。

- --quick, -q

  このオプションを使用してテーブルを検査している場合、正しくないリンクを検査するために行のスキャンを行いません。これが最速の検査方法です。

  このオプションを使用してテーブルを修復している場合、インデックスツリーのみの修復を試みます。これが最速の修復方法です。

- --repair, -r

  一意ではないユニークキー以外のすべてを修正できる修復を実行します。

- --secure-auth

  古い (4.1 より前の) 形式でサーバーにパスワードを送信しません。これにより、新しいパスワード形式を使用するサーバー以外への接続を防ぎます。このオプションはデフォルトで有効です。無効にするには --skip-secure-auth を使用します。このオプションは MySQL 5.6.17 で追加されました。

  > **注記**
  >
  > 4.1 より前のハッシュ方式を使用するパスワードはネイティブのパスワードハッシュ方式を使用するパスワードよりもセキュアでないため、使用しないようにしてください。4.1 よりも前のパスワードは非推奨であり、これらのサポートは今後の MySQL リリースで削除される予定です。アカウントのアップグレード手順については、セクション6.3.8.3「4.1 よりも前のパスワードハッシュ方式と mysql_old_password プラグインからの移行」を参照してください。

- --shared-memory-base-name=name

  Windows で、共有メモリーを使用して作成されるローカルサーバーへの接続の共有メモリー名。デフォルト値は MYSQL です。共有メモリー名では大文字と小文字を区別します。

  共有メモリー接続を可能にするには、サーバーは --shared-memory オプションで起動する必要があります。

- --silent, -s

  サイレントモード。エラーメッセージのみを出力します。

- --skip-database=db_name

  指名されたデータベース (大文字小文字は区別されます) を、mysqlcheck が実行する操作に含めません。このオプションは MySQL 5.6.11 で追加されました。

- --socket=path, -S path

  localhost への接続用に使用する、Unix ソケットファイル、または Windows では使用する名前付きパイプの名前。

- --ssl*

  --ssl で始まるオプションは、SSL を使用してサーバーに接続するかどうかを指定し、SSL 鍵および証明書を検索する場所を指定します。セクション6.3.10.4「SSL コマンドのオプション」を参照してください。

- --tables

  --databases オプションまたは -B オプションをおオーバーライドします。オプションに続くすべての名前引数はテーブル名とみなされます。

- --use-frm

  MyISAM テーブルの修復操作で、.frm ファイルからテーブル構造を取得することで .MYI ヘッダーが壊れていてもテーブルが修復できます。

- --user=user_name, -u user_name

  サーバーへの接続時に使用する MySQL ユーザー名。

- --verbose, -v

  冗長モード。プログラム処理のさまざまな段階についての情報を出力します。

- --version, -V

  バージョン情報を表示して終了します。

- --write-binlog

  このオプションはデフォルトで有効で、mysqlcheck によって生成される ANALYZE TABLE、OPTIMIZE TABLE、および REPAIR TABLE の各ステートメントがバイナリログに書き込まれます。--skip-write-binlog を使用すると、ステートメントに NO_WRITE_TO_BINLOG が追加され、ログに記録されなくなります。これらのステートメントがレプリケーションスレーブに送信されるべきではない場合またはバイナリログをバックアップからのリカバリに使用している場合には、--skip-write-binlog を使用します。

4.5.4 mysqldump — データベースバックアッププログラム

mysqldump クライアントは、論理バックアップ を実行するユーティリティーで、元のスキーマオブジェクト、テーブルデータ、または両方を再現するために実行できる SQL ステートメントのセットを生成します。バックアップまたは別の SQL サーバーへの転送のため、1 つまたは複数の MySQL データベースをダンプします。mysqldump コマンドは、CSV、その他の区切り文字で区切られたテキスト、または XML 形式でも出力を生成できます。

mysqldump では、ダンプされるテーブルに対する SELECT 権限、ダンプされるビューに対する SHOW VIEW、ダンプされるトリガーに対する TRIGGER、および --single-transaction オプションが使用されない場合には LOCK TABLES が少なくとも必要です。オプションの説明に示すように、一部のオプションではその他の権限が必要な場合があります。

ダンプファイルをリロードするには、CREATE ステートメントを手動で発行してダンプされる各オブジェクトを作成するために必要な権限と同じものを持っていなければなりません。

mysqldump 出力には、データベースの照合順序を変更する ALTER DATABASE ステートメントを含めることができます。これらは、ストアドプログラムをダンプする際に文字のエンコードを維持するために使用できます。このようなステートメントを含むダンプファイルをリロードするには、影響されるデータベースに対する ALTER 権限が必要です。

パフォーマンスおよびスケーラビリティーに関する考慮事項

mysqldump の利点には、リストアする前に出力を表示して編集もできるという便利さと柔軟性があります。開発およびデータベース管理用にデータベースのクローンを作成したり、テスト用に既存のデータベースとわずかに異なるデータベースを作成したりできます。大量のデータのバックアップのための、高速でスケーラブルなソリューションを意図したものではありません。データサイズが大量の場合、バックアップのステップにかかる時間が妥当だとしても、SQL ステートメントの再現には、挿入やインデックスの作成などのディスク I/O が含まれるため、データのリストアに非常に長い時間がかかることがあります。

大規模なバックアップとリストアでは、データファイルを高速でリストアできる元の形式でコピーする、物理バックアップの方が適切です。

- テーブルが主に InnoDB テーブルである場合、またはInnoDB テーブルと MyISAM テーブルが混在する場合は、MySQL Enterprise Backup 製品の mysqlbackup コマンドを使用することを検討してください。(Enterprise サブスクリプションの一部として含まれています。)これにより、最低限の中断でもっともパフォーマンスのよい InnoDB のバックアップを実行できます。また、MyISAM およびその他のストレージエンジンからのテーブルもバックアップでき、さまざまなバックアップシナリオに対応するための便利なオプションを多数提供します。セクション25.2「MySQL Enterprise Backup」を参照してください。
- テーブルが主に MyISAM テーブルである場合は、代わりに mysqlhotcopy を使用することを検討してください。mysqldump のバックアップおよびリストア操作よりも優れたパフォーマンスを提供します。セクション4.6.10「mysqlhotcopy — データベースバックアッププログラム」を参照してください。

mysqldump は、テーブルの内容を 1 行ずつ取得してダンプすることも、ダンプする前にテーブルからすべての内容を取得して、メモリーにバッファリングすることもできます。大きなテーブルをダンプしている場合、メモリーへのバッファリングが問題になる場合があります。テーブルを 1 行ずつダンプする場合、--quick オプションを使用してください (または --opt を指定すれば --quick が有効になります)。--opt オプションは (したがって --quick も) デフォルトで有効なため、メモリーへのバッファリングを有効にするには、--skip-quick を使用します。

最近のバージョンの mysqldump を使用して、非常に古い MySQL サーバーにリロードされるダンプを生成する場合は、--opt オプションまたは --extended-insert オプションの代わりに --skip-opt オプションを使用します。

mysqldump の詳細は、セクション7.4「バックアップへの mysqldump の使用」を参照してください。

構文

次に示すように、一般に mysqldump を使用するには、1 つまたは複数のテーブルのセットのダンプ、1 つまたは複数の完全なデータベースのセット、または MySQL サーバー全体の 3 つの方法があります。

```
shell> mysqldump [options] db_name [tbl_name ...]
shell> mysqldump [options] --databases db_name ...
shell> mysqldump [options] --all-databases
```

データベース全体をダンプするには、db_name に続けてテーブルを指名しないか、または --databases オプションまたは --all-databases オプションを使用します。

使用しているバージョンの mysqldump がサポートするオプションのリストを表示するには、コマンド mysqldump --help を発行します。

オプション構文 - アルファベット順のサマリー

mysqldump は次のオプションをサポートします。これらはコマンド行またはオプションファイルの [mysqldump] グループおよび [client] グループで指定できます。MySQL プログラムによって使用されるオプションファイルの詳細については、セクション4.2.6「オプションファイルの使用」を参照してください。

表 4.8 mysqldump のオプション

オプション名	説明	導入
--add-drop-database	DROP DATABASE ステートメントを CREATE DATABASE ステートメントの前に追加	
--add-drop-table	DROP TABLE ステートメントを CREATE TABLE ステートメントの前に追加	
--add-drop-trigger	DROP TRIGGER ステートメントを CREATE TRIGGER ステートメントの前に追加	
--add-locks	LOCK TABLES と UNLOCK TABLES ステートメントで各テーブルダンプを囲む	
--all-databases	すべてのデータベース内のすべてのテーブルをダンプ	
--allow-keywords	キーワードであるカラム名の作成を許可	
--apply-slave-statements	CHANGE MASTER ステートメントの前に STOP SLAVE を含め、START SLAVE を出力の最後に含める	
--bind-address	指定されたネットワークインタフェースを使用して MySQL サーバーに接続	5.6.1
--comments	ダンプファイルにコメントを追加	
--compact	よりコンパクトな出力を生成	
--compatible	古い MySQL サーバーやほかのデータベースシステムとの互換性がより高い出力を生成	
--complete-insert	カラム名を含む完全な INSERT ステートメントを使用	
--create-options	すべての MySQL に固有なテーブルオプションを CREATE TABLE ステートメントに含める	
--databases	複数のデータベースをダンプ	
--debug	デバッグのログを書き込み	
--debug-check	プログラムの終了時にデバッグ情報を出力	
--debug-info	プログラムの終了時に、デバッグ情報、メモリー、および CPU の統計を出力	
--default-auth	使用する認証プラグイン	
--default-character-set	デフォルト文字セットを指定	
--defaults-extra-file	通常のオプションファイルに加えてオプションファイルを読み取る	
--defaults-file	指名されたオプションファイルのみを読み取る	
--defaults-group-suffix	オプショングループのサフィクス値	
--delayed-insert	INSERT ステートメントではなく INSERT DELAYED ステートメントを書き出す	
--delete-master-logs	マスターレプリケーションサーバーで、ダンプ操作の実行後にバイナリログを削除	
--disable-keys	各テーブルについて、キーを無効にするステートメントおよび有効にするステートメントで INSERT ステートメントを囲む	
--dump-date	--comments が指定された場合、ダンプ日を "Dump completed on" コメントとして含める	
--dump-slave	スレーブのマスターのバイナリログ座標をリストする CHANGE MASTER ステートメントを含める	
--events	ダンプされたデータベースからイベントをダンプ	

オプション名	説明	導入
--extended-insert	複数の VALUES リストを含む、複数行 INSERT 構文を使用	
--fields-enclosed-by	このオプションは --tab オプションとともに使用され、LOAD DATA INFILE の対応する句と同じ意味を持つ	
--fields-escaped-by	このオプションは --tab オプションとともに使用され、LOAD DATA INFILE の対応する句と同じ意味を持つ	
--fields-optionally-enclosed-by	このオプションは --tab オプションとともに使用され、LOAD DATA INFILE の対応する句と同じ意味を持つ	
--fields-terminated-by	このオプションは --tab オプションとともに使用され、LOAD DATA INFILE の対応する句と同じ意味を持つ	
--flush-logs	ダンプを始める前に MySQL サーバーログファイルをフラッシュ	
--flush-privileges	mysql データベースのダンプ後、FLUSH PRIVILEGES ステートメントを発行	
--help	ヘルプメッセージを表示して終了	
--hex-blob	バイナリカラムを 16 進変換表記法を使用してダンプ (たとえば、'abc' は 0x616263)	
--host	接続先のホスト (IP アドレスまたはホスト名)	
--ignore-table	指定されたテーブルをダンプしない	
--include-master-host-port	--dump-slave とともに生成された CHANGE MASTER ステートメントに MASTER_HOST/MASTER_PORT オプションを含める	
--insert-ignore	INSERT ステートメントではなく INSERT IGNORE ステートメントを書き出す	
--lines-terminated-by	このオプションは --tab オプションとともに使用され、LOAD DATA INFILE の対応する句と同じ意味を持つ	
--lock-all-tables	データベース内のテーブルをすべてロック	
--lock-tables	テーブルをダンプする前にすべてロック	
--log-error	警告およびエラーを指名されたファイルに追加	
--login-path	ログインパスオプションを .mylogin.cnf から読み取り	5.6.6
--master-data	バイナリログファイルの名前と場所を出力に書き込む	
--max_allowed_packet	サーバーとの間で送受信するパケットの最大長	
--net_buffer_length	TCP/IP とソケット通信のバッファーサイズ	
--no-autocommit	ダンプされたテーブルごとに、INSERT ステートメントを SET autocommit = 0 ステートメントと COMMIT ステートメントで囲む	
--no-create-db	このオプションは CREATE DATABASE ステートメントを抑制	
--no-create-info	各ダンプされたテーブルを再作成する CREATE TABLE ステートメントを書き出さない	
--no-data	テーブルの内容をダンプしない	
--no-defaults	オプションファイルを読み取らない	
--no-set-names	--skip-set-charset と同じ	
--no-tablespaces	CREATE LOGFILE GROUP ステートメントおよび CREATE TABLESPACE ステートメントを出力に書き出さない	
--opt	--add-drop-table --add-locks --create-options --disable-keys --extended-insert --lock-tables --quick --set-charset の短縮形。	
--order-by-primary	各テーブルの行を、主キーまたは最初の一意のインデックスでソートしてダンプ	
--password	サーバーに接続する際に使用するパスワード	

オプション名	説明	導入
--pipe	Windows で、名前付きパイプを使用してサーバーに接続	
--plugin-dir	プラグインがインストールされているディレクトリ	
--port	接続に使用する TCP/IP ポート番号	
--print-defaults	デフォルトを出力	
--protocol	使用する接続プロトコル	
--quick	サーバーからのテーブルについて、一度に 1 行ずつ取得	
--quote-names	識別子を逆引用符文字で囲む	
--replace	INSERT ステートメントではなく REPLACE ステートメントを書き出す	
--result-file	指定されたファイルに出力	
--routines	ダンプされたデータベースからストアドルーチン (プロシージャーおよび関数) をダンプ	
--secure-auth	古い (4.1.1 より前の) 形式でサーバーにパスワードを送信しない	5.6.17
--set-charset	SET NAMES default_character_set を出力に追加	
--set-gtid-purged	SET @@GLOBAL.GTID_PURGED を出力に追加するかどうか	5.6.9
--shared-memory-base-name	共有メモリー接続に使用する共有メモリーの名前	
--single-transaction	このオプションは、サーバーからデータをダンプする前に BEGIN SQL ステートメントを発行	
--skip-add-drop-table	DROP TABLE ステートメントを CREATE TABLE ステートメントの前に追加しない	
--skip-add-locks	ロックを追加しない	
--skip-comments	ダンプファイルにコメントを追加しない	
--skip-compact	よりコンパクトな出力を生成しない	
--skip-disable-keys	キーを無効にしない	
--skip-extended-insert	extended-insert をオフにする	
--skip-opt	--opt で設定されたオプションをオフにする	
--skip-quick	サーバーからのテーブルについて、一度に 1 行ずつ取得しない	
--skip-quote-names	識別子を引用符で囲まない	
--skip-set-charset	SET NAMES ステートメントを抑制	
--skip-triggers	トリガーをダンプしない	
--skip-tz-utc	tz-utc をオフにする	
--socket	ローカルホストへの接続で、使用する Unix ソケットファイル	
--ssl	接続に SSL を有効化	
--ssl-ca	信頼された SSL CA のリストを含むファイルのパス	
--ssl-capath	信頼された SSL CA の PEM 形式の証明書を含むディレクトリのパス	
--ssl-cert	PEM 形式の X509 証明書を含むファイルのパス	
--ssl-cipher	SSL の暗号化に使用される、許可された暗号のリスト	
--ssl-crl	証明書失効リストを含むファイルのパス	5.6.3
--ssl-crlpath	証明書失効リストファイルを含むディレクトリのパス	5.6.3
--ssl-key	PEM 形式の X509 鍵を含むファイルのパス	
--ssl-verify-server-cert	サーバーへの接続時に、サーバーの証明書内のコモンネーム値をホスト名に対して検証	
--tab	タブ区切りのデータファイルを生成	

オプション名	説明	導入
--tables	--databases オプションまたは -B オプションをオーバーライド	
--triggers	ダンプされた各テーブルについて、トリガーをダンプする	
--tz-utc	SET TIME_ZONE='+00:00' をダンプファイルに追加	
--user	サーバーへの接続時に使用する MySQL ユーザー名	
--verbose	冗長モード	
--version	バージョン情報を表示して終了	
--where	所定の WHERE 条件で選択された行のみダンプ	
--xml	XML 出力を生成	

接続オプション

mysqldump コマンドは MySQL サーバーにログインして情報を抽出します。次のオプションは、同じマシンまたはリモートシステム上の MySQL サーバーに接続する方法を指定します。

- --bind-address=ip_address

  複数のネットワークインタフェースを持つコンピュータで、このオプションを使用して、MySQL サーバーへの接続に使用するインタフェースを選択します。

  このオプションは MySQL 5.6.1 からサポートされています。

- --compress, -C

  クライアントとサーバーの両方が圧縮をサポートしている場合、その間で送受信される情報をすべて圧縮します。

- --default-auth=plugin

  使用するクライアント側の認証プラグイン。セクション6.3.7「プラガブル認証」を参照してください。

- --host=host_name, -h host_name

  与えられたホスト上の MySQL サーバーからデータをダンプします。デフォルトホストは localhost です。

- --login-path=name

  指名されたログインパスから .mylogin.cnf ログインファイルのオプションを読み取ります。「ログインパス」は、host、user、および password という限定されたオプションのセットのみを許可するオプショングループです。ログインパスは、サーバーホストおよびそのサーバーで認証するための認証情報を示す値のセットであると考えてください。ログインパスファイルを作成するには、mysql_config_editor ユーティリティーを使用します。セクション4.6.6「mysql_config_editor — MySQL 構成ユーティリティー」を参照してください。このオプションは MySQL 5.6.6 で追加されました。

- --password[=password], -p[password]

  サーバーに接続する際に使用するパスワードです。短いオプション形式 (-p) を使用した場合は、オプションとパスワードの間にスペースを置くことはできません。コマンド行で、--password オプションまたは -p オプションに続けて password の値を指定しなかった場合、mysqldump はそれを要求します。

  コマンド行でのパスワード指定は、セキュアでないと考えるべきです。セクション6.1.2.1「パスワードセキュリティーのためのエンドユーザーガイドライン」を参照してください。オプションファイルを使用すれば、コマンド行でパスワードを指定することを回避できます。

- --pipe, -W

  Windows で、名前付きパイプを使用してサーバーに接続します。このオプションは、サーバーが名前付きパイプ接続をサポートしている場合にのみ適用されます。

- --plugin-dir=path

  プラグインを検索するディレクトリ。--default-auth オプションを使用して認証プラグインを指定したが、mysqldump がそれを検出できない場合は、このオプションを指定しなければならない可能性があります。セクション6.3.7「プラガブル認証」を参照してください。

- --port=port_num, -P port_num

  接続に使用する TCP/IP ポート番号。

- --protocol={TCP|SOCKET|PIPE|MEMORY}

  サーバーへの接続に使用する接続プロトコル。このオプションは、ほかの接続パラメータによって、必要なプロトコル以外のものが通常使用される場合に役立ちます。許可される値の詳細は、セクション4.2.2「MySQL サーバーへの接続」を参照してください。

- --secure-auth

  古い (4.1 より前の) 形式でサーバーにパスワードを送信しません。これにより、新しいパスワード形式を使用するサーバー以外への接続を防ぎます。このオプションはデフォルトで有効です。無効にするには --skip-secure-auth を使用します。このオプションは MySQL 5.6.17 で追加されました。

  > **注記**
  >
  > 4.1 より前のハッシュ方式を使用するパスワードはネイティブのパスワードハッシュ方式を使用するパスワードよりもセキュアでないため、使用しないようにしてください。4.1 よりも前のパスワードは非推奨であり、これらのサポートは今後の MySQL リリースで削除される予定です。アカウントのアップグレード手順については、セクション6.3.8.3「4.1 よりも前のパスワードハッシュ方式と mysql_old_password プラグインからの移行」を参照してください。

- --socket=path, -S path

  localhost への接続用に使用する、Unix ソケットファイル、または Windows では使用する名前付きパイプの名前。

- --ssl*

  --ssl で始まるオプションは、SSL を使用してサーバーに接続することを許可するかどうかを指定し、SSL 鍵および証明書を検索する場所を指定します。セクション6.3.10.4「SSL コマンドのオプション」を参照してください。

- --user=user_name, -u user_name

  サーバーへの接続時に使用する MySQL ユーザー名。

--var_name=value 構文を使用すれば、次の変数も設定できます。

- max_allowed_packet

  クライアント/サーバー通信用のバッファーの最大サイズ。デフォルトは 24M バイト、最大は 1G バイトです。

- net_buffer_length

  クライアント/サーバー通信用のバッファーの初期サイズ。複数行の INSERT ステートメント (--extended-insert オプションまたは --opt オプションを使用する場合など) を作成する場合、mysqldump は net_buffer_length までの長さの行を作成します。この変数を増加させる場合は、MySQL サーバー内の net_buffer_length 変数が少なくともこの大きさであることを確認してください。

オプションファイルオプション

これらのオプションは、どのオプションファイルを読み取るかを制御するために使用されます。

- --defaults-extra-file=file_name

  このオプションファイルは、グローバルオプションファイルのあとに読み取りますが、(UNIX では) ユーザーオプションファイルの前に読み取るようにしてください。ファイルが存在しないかアクセスできない場合、エラーが発生します。file_name は、フルパス名でなく相対パス名として指定された場合、現行ディレクトリを基準にして解釈されます。

- --defaults-file=file_name

  指定されたオプションファイルのみ使用します。ファイルが存在しないかアクセスできない場合、エラーが発生します。file_name は、フルパス名でなく相対パス名として指定された場合、現行ディレクトリを基準にして解釈されます。

- --defaults-group-suffix=str

  通常のオプショングループだけでなく、通常の名前に str のサフィクスが付いたグループも読み取ります。たとえば、mysqldump は通常 [client] グループおよび [mysqldump] グループを読み取ります。--defaults-group-suffix=_other オプションを指定した場合、mysqldump は [client_other] グループおよび [mysqldump_other] グループも読み取ります。

- --no-defaults

  オプションファイルを読み取りません。オプションファイルから不明のオプションを読み取ることが原因でプログラムの起動に失敗する場合、--no-defaults を使用して、オプションを読み取らないようにできます。

  例外として、.mylogin.cnf ファイルは、存在する場合はすべての場合に読み取られます。これにより、--no-defaults が使用された場合にも、コマンド行よりも安全な方法でパスワードを指定できます。(.mylogin.cnf は mysql_config_editor ユーティリティーによって作成されます。セクション4.6.6「mysql_config_editor — MySQL 構成ユーティリティー」を参照してください)。

- --print-defaults

  プログラム名と、オプションファイルから受け取るすべてのオプションを出力します。

DDL オプション

mysqldump の使用シナリオには、新しい MySQL インスタンス全体 (データベーステーブルを含む) のセットアップ、および既存のインスタンス内部のデータを既存のデータベースおよびテーブルで置換することが含まれます。次のオプションを使用すると、ダンプファイル内にさまざまな DDL ステートメントをエンコードすることによって、ダンプをリストアする際に何を削除し何をセットアップするのかを指定できます。

- --add-drop-database

  DROP DATABASE ステートメントを各 CREATE DATABASE ステートメントの前に追加します。通常このオプションは、--all-databases オプションまたは --databases オプションとともに使用されます。これらのオプションのいずれかを指定しないと CREATE DATABASE ステートメントが書き込まれないからです。

- --add-drop-table

  DROP TABLE ステートメントを各 CREATE TABLE ステートメントの前に追加します。

- --add-drop-trigger

  DROP TRIGGER ステートメントを各 CREATE TRIGGER ステートメントの前に追加します。

- --all-tablespaces, -Y

  NDB テーブルが使用するテーブルスペースを作成するために必要なすべての SQL ステートメントをテーブルダンプに追加します。そうしないと、この情報は mysqldump の出力には含まれません。このオプションは、現在 MySQL Cluster テーブルに対してのみ有効です。

- --no-create-db, -n

  このオプションは、--databases オプションまたは --all-databases オプションが指定されていた場合、出力に含まれる CREATE DATABASE ステートメントを抑制します。

- --no-create-info, -t

  ダンプされた各テーブルを再作成する CREATE TABLE ステートメントを書き込みません。

  > 注記
  >
  > このオプションは、ログファイルグループまたはテーブルスペースを作成するステートメントを、mysqldump 出力から除外しません。ただし、このためには --no-tablespaces オプションを使用できます。

- --no-tablespaces, -y

  このオプションは、mysqldump の出力内のすべての CREATE LOGFILE GROUP ステートメントおよびCREATE TABLESPACE ステートメントを抑制します。

- --replace

INSERT ステートメントではなく REPLACE ステートメントを書き込みます。

デバッグオプション

次のオプションは、デバッグ情報を出力したり、ダンプファイルにデバッグ情報をエンコードしたり、または潜在的な問題にかかわらずダンプ操作を続行させたりします。

- --allow-keywords

  キーワードであるカラム名の作成を許可します。これは各カラム名にテーブル名のプリフィクスを用いることで機能します。

- --comments, -i

  プログラムバージョン、サーバーバージョン、およびホストなどの追加情報をダンプファイルに書き込みます。このオプションはデフォルトで有効となっています。この追加情報を抑制するには、--skip-comments を使用してください。

- --debug[=debug_options], -# [debug_options]

  デバッグのログを書き込みます。一般的な debug_options 文字列は d:t:o,file_name です。デフォルト値は d:t:o,/tmp/mysqldump.trace です。

- --debug-check

  プログラムの終了時に、デバッグ情報を出力します。

- --debug-info

  プログラムの終了時に、デバッグ情報とメモリーおよび CPU 使用率の統計を出力します。

- --dump-date

  --comments オプションが指定された場合、mysqldump はダンプの最後に次の形式でコメントを生成します。

  ```
  -- Dump completed on DATE
  ```

  ただし、別のときに取られたダンプファイルが、日付以外のデータがまったく同じでも日付のために異なって見えます。--dump-date および --skip-dump-date は、コメントに日付を追加するかどうかを制御します。デフォルトは --dump-date (日付をコメントに含める) です。--skip-dump-date は日付の出力を抑制します。

- --force, -f

  テーブルダンプの最中に SQL エラーが発生しても続行します。

  このオプションの使い方の 1 つとして、削除されたテーブルをビュー定義が参照するために無効になっているビューを検出したときにも、mysqldump が実行を続けるようにすることです。--force を指定しないと、mysqldump はエラーメッセージで終了します。--force を使用すると、mysqldump はエラーメッセージを出力しますが、さらにビュー定義を含む SQL コメントをダンプ出力に書き込み、実行を継続します。

- --log-error=file_name

  警告およびエラーを、指名されたファイルに追加することによってログに記録します。デフォルトでは、ロギングを行いません。

- --skip-comments

  --comments オプションの説明を参照してください。

- --verbose, -v

  冗長モード。プログラムの動作についてより多くの情報を出力します。

ヘルプオプション

次のオプションは、mysqldump コマンド自身に関する情報を表示します。

- --help, -?

ヘルプメッセージを表示して終了します。

- --version, -V

  バージョン情報を表示して終了します。

国際化オプション

次のオプションは、mysqldump コマンドが各国語の設定で文字データを表現する方法を変更します。

- --character-sets-dir=path

  文字セットがインストールされているディレクトリ。セクション10.5「文字セットの構成」を参照してください。

- --default-character-set=charset_name

  charset_name をデフォルト文字セットとして使用します。セクション10.5「文字セットの構成」を参照してください。文字セットが指定されていない場合、mysqldump は utf8 を使用します。

- --no-set-names, -N

  --set-charset 設定をオフにします。--skip-set-charset を指定するのと同様です。

- --set-charset

  出力に SET NAMES default_character_set を追加します。このオプションはデフォルトで有効となっています。SET NAMES ステートメントを抑制するには、--skip-set-charset を使用してください。

レプリケーションオプション

mysqldump コマンドは、レプリケーションの構成において、空のインスタンスまたはデータを含むインスタンスをスレーブサーバーに作成するためによく使用されます。次のオプションは、レプリケーションマスターサーバーおよびスレーブサーバーでデータのダンプおよびリストアに適用されます。

- --apply-slave-statements

  --dump-slave オプションで作成されたスレーブダンプで、STOP SLAVE ステートメントを CHANGE MASTER TO ステートメントの前に、START SLAVE ステートメントを出力の最後に、それぞれ追加します。

- --delete-master-logs

  マスターレプリケーションサーバーで、ダンプ操作の実行後にサーバーに PURGE BINARY LOGS ステートメントを送信することにより、バイナリログを削除します。このオプションは自動的に --master-data を有効にします。

- --dump-slave[=value]

  このオプションは、ダンプされたサーバーと同じマスターを持つスレーブとして別のサーバーをセットアップするために使用できるダンプファイルを生成するために、レプリケーションスレーブサーバーをダンプするために使用することを除き、--master-data と同様です。これにより、ダンプされたスレーブのマスターのバイナリログ座標 (ファイル名とファイル位置) を示す CHANGE MASTER TO ステートメントがダンプ出力に追加されます。これらは、スレーブがレプリケーションを開始するマスターサーバーの座標です。

  --dump-slave を使用すると、座標は --master-data オプションのようにダンプされたサーバーのものではなく使用されるマスターのものになります。さらに、このオプションを指定すると --master-data オプションが使用されている場合はオーバーライドされ、実質上無視されるようになります。

  オプション値は --master-data と同様に処理され (値を設定しないか、1 を設定すると CHANGE MASTER TO ステートメントがダンプに書き込まれ、2 を設定するとステートメントは書き込まれますが SQL コメントで囲まれます)、ほかのオプションを有効または無効にする点および口ッキングの処理において、--master-data と同様の効果があります。

  このオプションを使用すると、mysqldump はダンプ前にスレーブ SQL スレッドを終了し、ダンプ後に再起動します。

  --dump-slave とともに、--apply-slave-statements オプションおよび --include-master-host-port オプションも使用できます。

- --include-master-host-port

  --dump-slave を使用して生成されたスレーブダンプ内の CHANGE MASTER TO ステートメントに、スレーブのマスターのホスト名および TCP/IP ポート番号の MASTER_HOST オプションおよび MASTER_PORT オプションを追加します。

- --master-data[=value]

  マスターレプリケーションサーバーをダンプして、別のサーバーをマスターのスレーブとして設定するために使用できるダンプファイルを生成する場合に、このオプションを使用します。これにより、ダンプされたサーバーのバイナリログ座標 (ファイル名とファイル位置) を示す CHANGE MASTER TO ステートメントがダンプ出力に追加されます。これらは、ダンプファイルをスレーブにロードしたあとで、スレーブがレプリケーションを開始するマスターサーバーの座標です。

  オプションの値が 2 の場合、CHANGE MASTER TO ステートメントは SQL コメントとして書き込まれるため、情報提供の意味しかなく、ダンプファイルがリロードされるときには何の効果もありません。オプション値が 1 の場合、ステートメントはコメントとしては書き込まれず、ダンプファイルがリロードされるときに実行されます。オプション値が指定されていない場合は、デフォルト値は 1 です。

  このオプションには RELOAD 権限が必要で、バイナリログが有効にされていなければいけません。

  --master-data オプションは自動的に --lock-tables をオフにします。また、--single-transaction も指定されていない場合は、--lock-all-tables をオンにします。その場合、ダンプの最初のわずかな時間のみグローバル読み取りロックが取得されます (--single-transaction の説明を参照してください)。どの場合でも、ログに対するアクションはすべてダンプと同時に発生します。

  --dump-slave オプションを使用してマスターの既存のスレーブをダンプすることによって、スレーブをセットアップすることも可能です。--master-data は、これによってオーバーライドされ、両方のオプションが使用された場合には無視されます。

  MySQL 5.6.4 より前では、レプリケーションログテーブルのダンプにはこのオプションが必要でした (セクション17.2.2「レプリケーションリレーおよびステータスログ」を参照してください)。

- --set-gtid-purged=value

  このオプションを使用すると、SET @@GLOBAL.gtid_purged ステートメントを出力に追加するかどうかを指定することによって、ダンプファイルに書き込まれるグローバルトランザクション ID (GTID) 情報を制御できます。

  次の表は、許可されるオプション値を示しています。デフォルト値は AUTO です。

値	意味
OFF	出力に SET ステートメントを追加しません。
ON	出力に SET ステートメントを追加します。サーバーで GTID が有効になっていない場合は、エラーが発生します。
AUTO	サーバーで GTID が有効になっている場合に、出力に SET ステートメントを追加します。

  このオプションは MySQL 5.6.9 で追加されました。

形式オプション

次のオプションは、ダンプファイル全体またはダンプファイル内のある種のデータの提示方法を指定します。また、ある種のオプションの情報をダンプファイルに書き込むかどうかも制御します。

- --compact

  よりコンパクトな出力を生成します。このオプションは、--skip-add-drop-table、--skip-add-locks、--skip-comments、--skip-disable-keys、および --skip-set-charset オプションを有効にします。

- --compatible=name

  古い MySQL サーバーやほかのデータベースシステムとの互換性がより高い出力を生成します。name の値は ansi、mysql323、mysql40、postgresql、oracle、mssql、db2、maxdb、no_key_options、no_table_options、または no_field_options となります。複数の値を使用する場合は、カンマで区切ります。これらの値は、サーバー SQL モード設定用の対応するオプションと同じ意味を持っています。セクション5.1.7「サーバー SQL モード」を参照してください。

このオプションはほかのサーバーとの互換性を保証するものではありません。現在提供されている、ダンプ出力の互換性を向上させるための SQL モード値を有効にするだけです。たとえば、--compatible=oracle はデータ型を Oracle 型に対応付けたり、Oracle コメント構文を使用したりしません。

このオプションには、バージョン 4.1.0 以降のサーバーが必要です。それよりも古いサーバーでは、何も行われません。

- --complete-insert, -c

  カラム名を含む、完全な INSERT ステートメントを使用します。

- --create-options

  MySQL 固有のテーブルオプションを CREATE TABLE ステートメントに含めます。

- --fields-terminated-by=..., --fields-enclosed-by=..., --fields-optionally-enclosed-by=..., --fields-escaped-by=...

  これらのオプションは --tab オプションとともに使用され、LOAD DATA INFILE の対応する FIELDS 句と同じ意味を持ちます。セクション13.2.6「LOAD DATA INFILE 構文」を参照してください。

- --hex-blob

  16 進表記を使用してバイナリカラムをダンプします (たとえば、'abc'は 0x616263 となります)。影響を受けるデータ型は、BINARY 型、VARBINARY 型、BLOB 型および BIT です。

- --lines-terminated-by=...

  このオプションは --tab とともに使用され、LOAD DATA INFILE の対応する LINES 句と同じ意味を持ちます。セクション13.2.6「LOAD DATA INFILE 構文」を参照してください。

- --quote-names, -Q

  識別子 (データベース、テーブル、およびカラム名など) を 「`」 文字で囲みます。ANSI_QUOTES SQL モードが有効な場合、識別子は 「"」 文字で囲まれます。このオプションはデフォルトで有効となっています。--skip-quote-names で無効にできますが、このオプションは --compatible のような --quote-names を有効にする可能性のあるオプションのあとに指定するようにしてください。

- --result-file=file_name, -r file_name

  指定されたファイルに出力します。このオプションは、Windows 上で改行文字 「\n」 が復帰/改行シーケンス 「\r\n」 に変換されるのを防ぐために使用します。ダンプの生成中にエラーが発生しても、結果ファイルが作成され以前の内容は上書きされます。

- --tab=path, -T path

  タブ区切りのテキスト形式データファイルを生成します。mysqldump は、各ダンプテーブルに対して、テーブルを作成する CREATE TABLE ステートメントを含む tbl_name.sql ファイルを作成し、サーバーはそのデータを含む tbl_name.txt ファイルに書き込みます。オプション値はファイルを書き込むディレクトリです。

  > **注記**
  >
  > このオプションは、mysqldump が mysqld サーバーと同じマシンで動作している場合にのみ使用するようにしてください。ユーザーは FILE 権限を持っている必要があり、サーバーは指定したディレクトリ内にファイルを書き込む許可を持っていなければいけません。

  デフォルトでは、.txt データファイルはカラム値の間にタブ文字、各行の最後に改行を使用する形式になります。この形式は、--fields-xxx オプションおよび --lines-terminated-by オプションを使用して明示的に指定できます。

  カラム値は、--default-character-set オプションで指定された文字セットに変換されます。

- --tz-utc

  このオプションにより、異なるタイムゾーンのサーバー間で TIMESTAMP カラムをダンプしてリロードできるようになります。mysqldump はその接続タイムゾーンを UTC に設定し、SET TIME_ZONE='+00:00' をダンプファイルに追加します。このオプションを使用しないと、TIMESTAMP カラムはダンプ元およびリロード先のサーバーのローカルタイムゾーンでダンプおよびリロードが実行され、サーバーが異なるタイムゾーンにある

場合、値が変更されます。--tz-utc は、サマータイムによる変更からも保護します。--tz-utc はデフォルトで有効です。無効にするには、--skip-tz-utc を使用します。

- --xml, -X

  ダンプ出力および整形式 XML を書き出します。

  NULL、'NULL'、および空の値: このオプションで生成される出力では、column_name という名前のカラムに関して、NULL 値、空の文字列、および文字列値 'NULL' は次のように互いに区別されます。

値:	XML 表現:
NULL (不明な値)	<field name="column_name" xsi:nil="true" />
'' (空の文字列)	<field name="column_name"></field>
'NULL' (文字列値)	<field name="column_name">NULL</field>

  mysql クライアントを --xml オプションを使用して実行した場合の出力も、前記のルールに従います。(セクション4.5.1.1「mysql のオプション」を参照してください。)

  mysqldump からの XML 出力には、次に示すように XML 名前空間が含まれます。

```
shell> mysqldump --xml -u root world City
<?xml version="1.0"?>
<mysqldump xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
<database name="world">
<table_structure name="City">
<field Field="ID" Type="int(11)" Null="NO" Key="PRI" Extra="auto_increment" />
<field Field="Name" Type="char(35)" Null="NO" Key="" Default="" Extra="" />
<field Field="CountryCode" Type="char(3)" Null="NO" Key="" Default="" Extra="" />
<field Field="District" Type="char(20)" Null="NO" Key="" Default="" Extra="" />
<field Field="Population" Type="int(11)" Null="NO" Key="" Default="0" Extra="" />
<key Table="City" Non_unique="0" Key_name="PRIMARY" Seq_in_index="1" Column_name="ID"
Collation="A" Cardinality="4079" Null="" Index_type="BTREE" Comment="" />
<options Name="City" Engine="MyISAM" Version="10" Row_format="Fixed" Rows="4079"
Avg_row_length="67" Data_length="273293" Max_data_length="18858823439613951"
Index_length="43008" Data_free="0" Auto_increment="4080"
Create_time="2007-03-31 01:47:01" Update_time="2007-03-31 01:47:02"
Collation="latin1_swedish_ci" Create_options="" Comment="" />
</table_structure>
<table_data name="City">
<row>
<field name="ID">1</field>
<field name="Name">Kabul</field>
<field name="CountryCode">AFG</field>
<field name="District">Kabol</field>
<field name="Population">1780000</field>
</row>

...

<row>
<field name="ID">4079</field>
<field name="Name">Rafah</field>
<field name="CountryCode">PSE</field>
<field name="District">Rafah</field>
<field name="Population">92020</field>
</row>
</table_data>
</database>
</mysqldump>
```

  MySQL 5.6.5 より前では、このオプションを使用すると --routines オプションが正しく機能しませんでした。つまり、ストアドルーチン、トリガー、およびイベントは XML 形式でダンプできませんでした。(Bug #11760384、Bug #52792)

フィルタリングオプション

次のオプションは、どのような種類のスキーマオブジェクトがダンプファイルに書き出されるかを、トリガーまたはイベントなどのカテゴリによって制御したり、たとえばダンプするデータベースおよびテーブルを選択して名前によって制御したり、または WHERE 句を使用してテーブルデータから行をフィルタリングして制御したりできます。

- --all-databases, -A

すべてのデータベース内のすべてのテーブルをダンプします。これは、コマンド行で --databases オプションを使用してすべてのデータベース名を指定するのと同じです。

MySQL 5.6.4 より前では、このオプションでは slave_master_info テーブルおよび slave_relay_log_info テーブル (セクション17.2.2「レプリケーションリレーおよびステータスログ」を参照してください) は含まれませんでした。

- --databases, -B

  複数のデータベースをダンプします。通常、mysqldump は、コマンド行の最初の名前引数をデータベース名として、それに続く名前をテーブル名として処理します。このオプションを使用すると、名前引数をすべてデータベース名として処理します。出力には、各新しいデータベースの前に CREATE DATABASE ステートメントおよび USE ステートメントが含まれます。

- --events, -E

  ダンプされるデータベースのイベントスケジューライベントを出力に含めます。

- --ignore-table=db_name.tbl_name

  指定されたテーブルをダンプしません。これはデータベース名とテーブル名を両方指定する必要があります。複数のテーブルを無視するには、このオプションを複数回使用してください。このオプションを使用してビューを無視することもできます。

- --no-data, -d

  テーブルの行情報を書き出しません (つまり、テーブルの内容をダンプしません)。これは、テーブルの CREATE TABLE ステートメントのみをダンプする場合に便利です (たとえば、ダンプファイルをロードしてテーブルの空のコピーを作成する場合など)。

- --routines, -R

  ダンプされるデータベースのストアドルーチン (プロシージャーおよび関数) を出力に含めます。このオプションを使用するには、mysql.proc テーブルに対する SELECT 権限が必要です。--routines を使用して生成された出力は、ルーチンの再作成のため、CREATE PROCEDURE ステートメントおよび CREATE FUNCTION ステートメントを含んでいます。ただし、これらのステートメントはルーチンの作成および変更のタイムスタンプなどの属性を含みません。つまり、ルーチンがリロードされたとき、リロード時間と等しいタイムスタンプで作成されます。

  ルーチンを元のタイムスタンプ属性で再作成しなければならない場合は、--routines を使用しないでください。代わりに、mysql データベースの適切な権限を持っている MySQL アカウントを使用して、mysql.proc テーブルの内容を直接ダンプしてリロードしてください。

  MySQL 5.6.5 より前では、このオプションは、--xml オプションとともに使用した場合には何の効果もありません。(Bug #11760384、Bug #52792)

- --tables

  --databases オプションまたは -B オプションをオーバーライドします。mysqldump は、このオプションに続く名前の引数をすべてテーブル名とみなします。

- --triggers

  ダンプされる各テーブルのトリガーを出力に含めます。このオプションはデフォルトで有効です。--skip-triggers を使用して無効にします。

- --where='where_condition', -w 'where_condition'

  指定された WHERE 条件で選択される行のみダンプします。条件が、スペースまたはユーザーのコマンドインタプリタにとって特別なその他の文字を含んでいる場合、条件を引用符で囲まなければなりません。

  例:

```
--where="user='jimf'"
-w"userid>1"
-w"userid<1"
```

パフォーマンスオプション

次のオプションは、特にリストア操作のパフォーマンスにもっとも重要です。大規模なデータセットでは、リストア操作 (ダンプファイル内の INSERT ステートメントの処理) がもっとも時間のかかる部分です。データを迅速にリストアすることが緊急である場合、事前にステージを計画してパフォーマンスをテストします。時間単位で測定されるリストア時間に対して、InnoDB のみ、または混在するデータベースでは MySQL Enterprise Backup、または MyISAM のみのデータベースでは mysqlhotcopy など、代替のバックアップおよびリストアソリューションが望ましい場合もあります。

パフォーマンスは、主にダンプ操作に関して、トランザクションオプションにも影響されます。

- --delayed-insert

  INSERT DELAYED 構文をサポートする非トランザクションテーブルでは、通常の INSERT ステートメントではなくそのステートメントを使用します。

  MySQL 5.6.6 では、DELAYED 挿入は非推奨であるため、このオプションは将来のリリースで削除されます。

- --disable-keys, -K

  テーブルごとに、INSERT ステートメントを /*!40000 ALTER TABLE tbl_name DISABLE KEYS */; ステートメントと /*!40000 ALTER TABLE tbl_name ENABLE KEYS */; ステートメントで囲みます。これにより、行がすべて挿入されたあとにインデックスが作成されるため、ダンプファイルのロードが高速になります。このオプションは、MyISAM テーブルの一意でないインデックスにのみ効果があります。

- --extended-insert, -e

  複数の VALUES リストを含む、複数行の INSERT 構文を使用します。これにより、ダンプファイルのサイズが小さくなり、ファイルがリロードされる際の挿入が高速化されます。

- --insert-ignore

  INSERT ステートメントではなく、INSERT IGNORE ステートメントを書き出します。

- --opt

  このオプションはデフォルトで有効で、--add-drop-table --add-locks --create-options --disable-keys --extended-insert --lock-tables --quick --set-charset の組み合わせの短縮形です。高速ダンプ操作が可能になり、MySQL サーバーに迅速にリロードできるダンプファイルを生成します。

  --opt オプションはデフォルトで有効であるため、いくつかのデフォルト設定をオフにする場合のみ、この逆の --skip-opt を指定します。--opt に影響されるオプションのサブセットを選択的に有効または無効にする方法は、mysqldump オプショングループ の説明を参照してください。

- --quick, -q

  このオプションは大規模なテーブルのダンプに便利です。これは mysqldump に対して、テーブルのすべての行のセットを取得して、書き出す前にメモリーにバッファリングするのではなく、サーバーから 1 行ずつ行を取得することを強制します。

- --skip-opt

  --opt オプションの説明を参照してください。

トランザクションオプション

次のオプションは、エクスポートされるデータの信頼性と一貫性のために、ダンプ操作のパフォーマンスを犠牲にします。

- --add-locks

  LOCK TABLES ステートメントと UNLOCK TABLES ステートメントで各テーブルダンプを囲みます。これにより、ダンプファイルをリロードする際の挿入の速度が向上します。セクション8.2.2.1「INSERT ステートメントの速度」を参照してください。

- --flush-logs, -F

  ダンプを始める前に MySQL サーバーログファイルをフラッシュします。このオプションには RELOAD 権限が必要です。このオプションを --all-databases オプションと組み合わせて使用すると、ログはダンプされるデー

タベースごとにフラッシュされます。例外は、--lock-all-tables、--master-data、または --single-transaction を使用する場合です。この場合、ログはすべてのテーブルがロックされた瞬間に対応して一度のみフラッシュされます。ダンプとログのフラッシュを正確に同時に実行するには、--flush-logs を --lock-all-tables、--master-data、または --single-transaction とともに使用するようにしてください。

- --flush-privileges

  mysql データベースのダンプ後に、ダンプ出力に FLUSH PRIVILEGES ステートメントを追加します。ダンプに mysql データベースおよび mysql データベース内のデータに依存するその他のすべてのデータベースが含まれている場合には、正しいリストアのために必ずこのオプションを使用するようにしてください。

- --lock-all-tables, -x

  データベース内のテーブルをすべてロックします。これは全ダンプの期間、グローバル読み取りロックを取得することで達成されます。このオプションにより、--single-transaction および --lock-tables は自動的にオフになります。

- --lock-tables, -l

  ダンプされる各データベースに対して、ダンプするすべてのテーブルをダンプ前にロックします。MyISAM テーブルの場合には、並列挿入を許可するため、テーブルは READ LOCAL でロックされます。InnoDB などのトランザクションテーブルの場合は、--single-transaction はテーブルをロックする必要がまったくないため、--lock-tables よりはるかに適したオプションです。

  --lock-tables は各データベースに対して個別にテーブルをロックするため、このオプションではダンプファイル内のテーブルがデータベース間で論理的に一貫していることは保証されません。異なるデータベース内のテーブルは完全に異なる状態でダンプされることがあります。

  --opt など、一部のオプションは --lock-tables を自動的に有効にします。これをオーバーライドするには、--skip-lock-tables をオプションリストの最後に使用します。

- --no-autocommit

  ダンプされるテーブルごとに、INSERT ステートメントを SET autocommit = 0 ステートメントと COMMIT ステートメントで囲みます。

- --order-by-primary

  各テーブルの行を、主キーまたは最初の一意のインデックス (このようなインデックスが存在する場合) でソートしてダンプします。これは、InnoDB テーブルにロードされる MyISAM テーブルをダンプする場合に便利ですが、ダンプ操作にかかる時間がかなり長くなります。

- --shared-memory-base-name=name

  Windows で、共有メモリーを使用して作成されるローカルサーバーへの接続の共有メモリー名。デフォルト値は MYSQL です。共有メモリー名では大文字と小文字を区別します。

  共有メモリー接続を可能にするには、サーバーは --shared-memory オプションで起動する必要があります。

- --single-transaction

  このオプションは、データのダンプ前に、トランザクション分離モードを REPEATABLE READ に設定し、START TRANSACTION SQL ステートメントをサーバーに送信します。これは、InnoDB などのトランザクションテーブルの場合にかぎって便利です。その場合、アプリケーションをブロックすることなく、START TRANSACTION が発行された時点のデータベースの一貫した状態をダンプするからです。

  このオプションを使用する場合、一貫した状態でダンプされるのは InnoDB テーブルのみだということに留意してください。たとえば、このオプションの使用中にダンプされた MyISAM テーブルまたは MEMORY テーブルは状態が変化する可能性があります。

  --single-transaction ダンプの処理中、ダンプファイルが正当である (テーブルの内容とバイナリログ座標が正しい) ことを保証するために、ほかの接続で ALTER TABLE、CREATE TABLE、DROP TABLE、RENAME TABLE、TRUNCATE TABLE ステートメントを使用しないようにしてください。一貫性読み取りはこれらのステートメントから分離されないため、ダンプされるテーブルでこれらを使用すると、mysqldump によって実行され、テーブルの内容を取得する SELECT が、正しくない内容を取得したり失敗したりすることがあります。

  --single-transaction オプションおよび --lock-tables オプションは相互に排他的です。これは、保留中のトランザクションが LOCK TABLES により暗黙的にコミットされるためです。

大規模なテーブルをダンプするには、--single-transaction オプションを --quick オプションと組み合わせてください。

オプショングループ

- --opt オプションは、高速なダンプ操作を実行するために協働するいくつかの設定をオンにします。--opt はデフォルトでオンであるため、これらの設定はすべてデフォルトでオンです。したがって、--opt を指定することは、あるとしてもまれです。代わりに、--skip-opt を指定してこれらの設定をグループとしてオフにし、そのあと、コマンド行で関連するオプションを指定して特定の設定を再度有効にできます。
- --compact オプションは、オプションのステートメントおよびコメントが出力に現れるかどうかを制御するいくつかの設定をオフにします。この場合も、このオプションに、特定の設定を再度有効にするその他のオプションを続けたり、--skip-compact の形式を使用してすべての設定をオンにしたりできます。

グループオプションの一部を選択的に効果を有効または無効にする場合、オプションは前から後ろへの順で処理されるため、順序が重要です。たとえば、--disable-keys --lock-tables --skip-opt では意図している効果を得られません。--skip-opt だけの場合と同じになります。

例

データベース全体のバックアップを作成するには:

```
shell> mysqldump db_name > backup-file.sql
```

ダンプファイルをサーバーにロードするには:

```
shell> mysql db_name < backup-file.sql
```

ダンプファイルをリロードする別の方法:

```
shell> mysql -e "source /path-to-backup/backup-file.sql" db_name
```

mysqldump は、1 つの MySQL サーバーから別のサーバーにデータをコピーすることでデータベースを移入するのに非常に便利です。

```
shell> mysqldump --opt db_name | mysql --host=remote_host -C db_name
```

複数のデータベースを 1 つのコマンドでダンプできます。

```
shell> mysqldump --databases db_name1 [db_name2 ...] > my_databases.sql
```

すべてのデータベースをダンプするには、--all-databases オプションを使用します。

```
shell> mysqldump --all-databases > all_databases.sql
```

InnoDB テーブルに関して、mysqldump はオンラインバックアップの作成方法を提供します。

```
shell> mysqldump --all-databases --master-data --single-transaction > all_databases.sql
```

このバックアップでは、ダンプの最初で (FLUSH TABLES WITH READ LOCK を使用して) すべてのテーブルに対するグローバル読み取りロックが取得されます。このロックが取得されるとすぐに、バイナリログの座標が読み取られ、ロックが解除されます。FLUSH ステートメントが発行されたときに長い更新ステートメントが実行中の場合、MySQL サーバーはそれらのステートメントが終わるまで停止する可能性があります。そのあと、ダンプはロックがなくなり、テーブルの読み取りと書き込みを妨げることはなくなります。MySQL サーバーが受信する更新ステートメントが (実行時間の点で) 短い場合、更新の数が多くても最初のロック時間はさほど気にならないはずです。

ポイントインタイムリカバリ (または 「ロールフォワード」、これは古いバックアップをリストアし、そのバックアップ後に発生した変更を再現する必要がある場合) は、バイナリログを交替させる (セクション5.2.4「バイナリログ」を参照してください) か、または少なくともダンプが対応しているバイナリログ座標を知っていると便利な場合があります。

```
shell> mysqldump --all-databases --master-data=2 > all_databases.sql
```

または:

```
shell> mysqldump --all-databases --flush-logs --master-data=2
          > all_databases.sql
```

--master-data オプションおよび --single-transaction オプションは同時に使用できます。これは、テーブルが InnoDB ストレージエンジンを使用して保存されている場合に、ポイントインタイムリカバリの前に使用するのに適したオンラインバックアップを作成する便利な方法を提供します。

バックアップ作成の詳細は、セクション7.2「データベースバックアップ方法」とセクション7.3「バックアップおよびリカバリ戦略の例」を参照してください。

- いくつかの機能を除いて --opt の効果を選択するには、除く各機能に対して --skip オプションを選択します。拡張挿入およびメモリーバッファリングを無効にするには、--opt --skip-extended-insert --skip-quick を使用します。(--opt はデフォルトでオンであるため、実際には --skip-extended-insert --skip-quick で十分です。)
- インデックスの無効化とテーブルのロックを除くすべての機能に関して --opt を反転するには、--skip-opt --disable-keys --lock-tables を使用します。

制約

mysqldump は、デフォルトでは INFORMATION_SCHEMA データベースおよび performance_schema データベースをダンプしません。これらのいずれかをダンプするには、コマンド行で明示的に指定し、--skip-lock-tables オプションも使用します。--databases オプションでも指定できます。

mysqldump は、MySQL Cluster ndbinfo 情報データベースをダンプしません。

MySQL 5.6.6 より前では、mysqldump は mysql データベースのダンプとして general_log テーブルも slow_query_log テーブルもダンプしません。5.6.6 以降では、ダンプにはそれらのテーブルを再作成するためのステートメントが含まれているため、ダンプファイルを再ロードしたあとにそれらのテーブルが失われません。ログテーブルの内容はダンプされません。

権限が不十分なためビューのバックアップに問題が生じる場合は、セクションD.5「ビューの制約」の回避策を参照してください。

4.5.5 mysqlimport — データインポートプログラム

mysqlimport クライアントは、LOAD DATA INFILE SQL ステートメントにコマンド行インタフェースを提供します。mysqlimport に対するほとんどのオプションは、LOAD DATA INFILE 構文の句に直接対応しています。セクション13.2.6「LOAD DATA INFILE 構文」を参照してください。

mysqlimport は次のように起動します。

```
shell> mysqlimport [options] db_name textfile1 [textfile2 ...]
```

mysqlimport は、コマンド行で指定されたテキストファイルごとに、ファイル名の拡張子があればそれを取り除き、その結果をもとに、ファイル内容をインポートするテーブルの名前を決定します。たとえば、patient.txt、patient.text、および patient と名付けられたファイルはすべて patient と名付けられたテーブルにインポートされます。

mysqlimport は次のオプションをサポートします。これらはコマンド行またはオプションファイルの [mysqlimport] グループおよび [client] グループで指定できます。MySQL プログラムによって使用されるオプションファイルの詳細については、セクション4.2.6「オプションファイルの使用」を参照してください。

表 4.9 mysqlimport のオプション

オプション名	説明	導入
--bind-address	指定されたネットワークインタフェースを使用して MySQL サーバーに接続	5.6.1
--columns	このオプションはカンマによって分けられたカラム名のリストを値とします	
--compress	クライアントとサーバー間で送信される情報をすべて圧縮	
--debug	デバッグのログを書き込み	
--debug-check	プログラムの終了時にデバッグ情報を出力	
--debug-info	プログラムの終了時に、デバッグ情報、メモリー、および CPU の統計を出力	
--default-auth	使用する認証プラグイン	5.6.2
--default-character-set	デフォルト文字セットを指定	
--defaults-extra-file	通常のオプションファイルに加えてオプションファイルを読み取る	
--defaults-file	指名されたオプションファイルのみを読み取る	
--defaults-group-suffix	オプショングループのサフィクス値	

オプション名	説明	導入
--delete	テキストファイルをインポートする前にテーブルを空にする	
--fields-enclosed-by	このオプションは、LOAD DATA INFILE の対応する句と同じ意味を持っています	
--fields-escaped-by	このオプションは、LOAD DATA INFILE の対応する句と同じ意味を持っています	
--fields-optionally-enclosed-by	このオプションは、LOAD DATA INFILE の対応する句と同じ意味を持っています	
--fields-terminated-by	-- このオプションは、LOAD DATA INFILE の対応する句と同じ意味を持っています	
--force	SQL エラーが発生しても続行	
--help	ヘルプメッセージを表示して終了	
--host	指定されたホスト上で MySQL サーバーに接続	
--ignore	--replace オプションの説明を参照	
--ignore-lines	データファイルの最初の N 行を無視	
--lines-terminated-by	このオプションは、LOAD DATA INFILE の対応する句と同じ意味を持っています	
--local	クライアントホストから入力ファイルをローカルで読み込む	
--lock-tables	テキストファイルを処理する前に、すべてのテーブルを書き込みロック	
--login-path	ログインパスオプションを .mylogin.cnf から読み取り	5.6.6
--low-priority	テーブルロード時に LOW_PRIORITY を使用。	
--no-defaults	オプションファイルを読み取らない	
--password	サーバーに接続する際に使用するパスワード	
--pipe	Windows で、名前付きパイプを使用してサーバーに接続	
--plugin-dir	プラグインがインストールされているディレクトリ	5.6.2
--port	接続に使用する TCP/IP ポート番号	
--print-defaults	デフォルトを出力	
--protocol	使用する接続プロトコル	
--replace	--replace および --ignore オプションは、一意のキー値に関して既存の行と重複する入力行の処理を制御	
--secure-auth	古い (4.1.1 より前の) 形式でサーバーにパスワードを送信しない	5.6.17
--shared-memory-base-name	共有メモリー接続に使用する共有メモリーの名前	
--silent	エラーが発生したときのみ出力を生成	
--socket	ローカルホストへの接続で、使用する Unix ソケットファイル	
--ssl	接続に SSL を有効化	
--ssl-ca	信頼された SSL CA のリストを含むファイルのパス	
--ssl-capath	信頼された SSL CA の PEM 形式の証明書を含むディレクトリのパス	
--ssl-cert	PEM 形式の X509 証明書を含むファイルのパス	
--ssl-cipher	SSL の暗号化に使用される、許可された暗号のリスト	
--ssl-crl	証明書失効リストを含むファイルのパス	5.6.3
--ssl-crlpath	証明書失効リストファイルを含むディレクトリのパス	5.6.3
--ssl-key	PEM 形式の X509 鍵を含むファイルのパス	
--ssl-verify-server-cert	サーバーへの接続時に、サーバーの証明書内のコモンネーム値をホスト名に対して検証	
--use-threads	並列ファイルロードのスレッド数	

オプション名	説明	導入
--user	サーバーへの接続時に使用する MySQL ユーザー名	
--verbose	冗長モード	
--version	バージョン情報を表示して終了	

- --help, -?

  ヘルプメッセージを表示して終了します。

- --bind-address=ip_address

  複数のネットワークインタフェースを持つコンピュータで、このオプションを使用して、MySQL サーバーへの接続に使用するインタフェースを選択します。

  このオプションは MySQL 5.6.1 からサポートされています。

- --character-sets-dir=path

  文字セットがインストールされているディレクトリ。セクション10.5「文字セットの構成」を参照してください。

- --columns=column_list, -c column_list

  このオプションは、カンマで区切られたカラム名のリストを値として取ります。カラム名の順序は、データファイルのカラムとテーブルのカラムを対応付ける方法を示しています。

- --compress, -C

  クライアントとサーバーの両方が圧縮をサポートしている場合、その間で送受信される情報をすべて圧縮します。

- --debug[=debug_options], -# [debug_options]

  デバッグのログを書き込みます。一般的な debug_options 文字列は d:t:o,file_name です。デフォルトは d:t:o です。

- --debug-check

  プログラムの終了時に、デバッグ情報を出力します。

- --debug-info

  プログラムの終了時に、デバッグ情報とメモリーおよび CPU 使用率の統計を出力します。

- --default-character-set=charset_name

  charset_name をデフォルト文字セットとして使用します。セクション10.5「文字セットの構成」を参照してください。

- --default-auth=plugin

  使用するクライアント側の認証プラグイン。セクション6.3.7「プラガブル認証」を参照してください。

  このオプションは MySQL 5.6.2 で追加されました。

- --defaults-extra-file=file_name

  このオプションファイルは、グローバルオプションファイルのあとに読み取りますが、(UNIX では) ユーザーオプションファイルの前に読み取るようにしてください。ファイルが存在しないかアクセスできない場合、エラーが発生します。file_name は、フルパス名でなく相対パス名として指定された場合、現行ディレクトリを基準にして解釈されます。

- --defaults-file=file_name

  指定されたオプションファイルのみ使用します。ファイルが存在しないかアクセスできない場合、エラーが発生します。file_name は、フルパス名でなく相対パス名として指定された場合、現行ディレクトリを基準にして解釈されます。

- --defaults-group-suffix=str

通常のオプショングループだけでなく、通常の名前に str のサフィクスが付いたグループも読み取ります。たとえば、mysqlimport は通常 [client] グループおよび [mysqlimport] グループを読み取ります。--defaults-group-suffix=_other オプションを指定した場合、mysqlimport は [client_other] グループおよび [mysqlimport_other] グループも読み取ります。

- --delete, -D

  テキストファイルをインポートする前にテーブルを空にします。

- --fields-terminated-by=..., --fields-enclosed-by=..., --fields-optionally-enclosed-by=..., --fields-escaped-by=...

  これらのオプションは LOAD DATA INFILE の対応する句と同じ意味を持っています。セクション 13.2.6「LOAD DATA INFILE 構文」を参照してください。

- --force, -f

  エラーを無視します。たとえば、テキストファイルのテーブルが存在しない場合、残ったファイルの処理を続行します。--force がない場合は、テーブルが存在しないと mysqlimport は終了します。

- --host=host_name, -h host_name

  指定されたホスト上の MySQL サーバーにデータをインポートします。デフォルトホストは localhost です。

- --ignore, -i

  --replace オプションの説明を参照してください。

- --ignore-lines=N

  データファイルの最初の N 行を無視します。

- --lines-terminated-by=...

  このオプションは LOAD DATA INFILE の対応する句と同じ意味を持っています。たとえば、復帰改行と改行のペアで終了する行を持つ Windows ファイルをインポートする場合、--lines-terminated-by="\r\n " を使用してください。(コマンドインタプリタのエスケープ処理の規則によってはバックスラッシュを 2 つ使用しなければならない場合があります。)セクション13.2.6「LOAD DATA INFILE 構文」を参照してください。

- --local, -L

  クライアントホストから入力ファイルをローカルで読み取ります。

- --lock-tables, -l

  テキストファイルを処理する前に、すべてのテーブルを書き込み用にロックします。これにより、すべてのテーブルがサーバー上で同期していることが保証されます。

- --login-path=name

  指名されたログインパスから .mylogin.cnf ログインファイルのオプションを読み取ります。「ログインパス」は、host、user、および password という限定されたオプションのセットのみを許可するオプショングループです。ログインパスは、サーバーホストおよびそのサーバーで認証するための認証情報を示す値のセットであると考えてください。ログインパスファイルを作成するには、mysql_config_editor ユーティリティーを使用します。セクション4.6.6「mysql_config_editor — MySQL 構成ユーティリティー」を参照してください。このオプションは MySQL 5.6.6 で追加されました。

- --low-priority

  テーブルのロード時に LOW_PRIORITY を使用します。これは、テーブルレベルロックのみを使用するストレージエンジン (MyISAM、MEMORY、および MERGE) にのみ影響を与えます。

- --no-defaults

  オプションファイルを読み取りません。オプションファイルから不明のオプションを読み取ることが原因でプログラムの起動に失敗する場合、--no-defaults を使用して、オプションを読み取らないようにできます。

  例外として、.mylogin.cnf ファイルは、存在する場合はすべての場合に読み取られます。これにより、--no-defaults が使用された場合にも、コマンド行よりも安全な方法でパスワードを指定できます。(.mylogin.cnf

は mysql_config_editor ユーティリティーによって作成されます。セクション4.6.6「mysql_config_editor — MySQL 構成ユーティリティー」を参照してください)。

- --password[=password], -p[password]

  サーバーに接続する際に使用するパスワードです。短いオプション形式 (-p) を使用した場合は、オプションとパスワードの間にスペースを置くことはできません。コマンド行で、--password オプションまたは -p オプションに続けて password の値を指定しなかった場合、mysqlimport はそれを要求します。

  コマンド行でのパスワード指定は、セキュアでないと考えるべきです。セクション6.1.2.1「パスワードセキュリティーのためのエンドユーザーガイドライン」を参照してください。オプションファイルを使用すれば、コマンド行でパスワードを指定することを回避できます。

- --pipe, -W

  Windows で、名前付きパイプを使用してサーバーに接続します。このオプションは、サーバーが名前付きパイプ接続をサポートしている場合にのみ適用されます。

- --plugin-dir=path

  プラグインを検索するディレクトリ。--default-auth オプションを使用して認証プラグインを指定したが、mysqlimport がそれを検出できない場合は、このオプションを指定しなければならない可能性があります。セクション6.3.7「プラガブル認証」を参照してください。

  このオプションは MySQL 5.6.2 で追加されました。

- --port=port_num, -P port_num

  接続に使用する TCP/IP ポート番号。

- --print-defaults

  プログラム名と、オプションファイルから受け取るすべてのオプションを出力します。

- --protocol={TCP|SOCKET|PIPE|MEMORY}

  サーバーへの接続に使用する接続プロトコル。このオプションは、ほかの接続パラメータによって、必要なプロトコル以外のものが通常使用される場合に役立ちます。許可される値の詳細は、セクション4.2.2「MySQL サーバーへの接続」を参照してください。

- --replace, -r

  --replace オプションおよび --ignore オプションは、既存の行の一意のキー値を重複させるような入力行の処理を制御します。--replace を指定した場合、同じ一意のキー値を持つ既存の行は新しい行で置換されます。--ignore を指定した場合、既存の行の一意のキー値を重複させる入力行はスキップされます。どちらのオプションも指定しなかった場合、重複するキー値が検出されるとエラーが発生し、残りのテキストファイルは無視されます。

- --secure-auth

  古い (4.1 より前の) 形式でサーバーにパスワードを送信しません。これにより、新しいパスワード形式を使用するサーバー以外への接続を防ぎます。このオプションはデフォルトで有効です。無効にするには --skip-secure-auth を使用します。このオプションは MySQL 5.6.17 で追加されました。

  > **注記**
  >
  > 4.1 より前のハッシュ方式を使用するパスワードはネイティブのパスワードハッシュ方式を使用するパスワードよりもセキュアでないため、使用しないようにしてください。4.1 よりも前のパスワードは非推奨であり、これらのサポートは今後の MySQL リリースで削除される予定です。アカウントのアップグレード手順については、セクション6.3.8.3「4.1 よりも前のパスワードハッシュ方式と mysql_old_password プラグインからの移行」を参照してください。

- --shared-memory-base-name=name

  Windows で、共有メモリーを使用して作成されるローカルサーバーへの接続の共有メモリー名。デフォルト値は MYSQL です。共有メモリー名では大文字と小文字を区別します。

  共有メモリー接続を可能にするには、サーバーは --shared-memory オプションで起動する必要があります。

- --silent, -s

  サイレントモード。エラーが発生したときのみ出力を生成します。

- --socket=path, -S path

  localhost への接続用に使用する、Unix ソケットファイル、または Windows では使用する名前付きパイプの名前。

- --ssl*

  --ssl で始まるオプションは、SSL を使用してサーバーに接続することを許可するかどうかを指定し、SSL 鍵および証明書を検索する場所を指定します。セクション6.3.10.4「SSL コマンドのオプション」を参照してください。

- --user=user_name, -u user_name

  サーバーへの接続時に使用する MySQL ユーザー名。

- --use-threads=N

  N 個のスレッドを使用して複数のファイルを並行してロードします。

- --verbose, -v

  冗長モード。プログラムの動作についてより多くの情報を出力します。

- --version, -V

  バージョン情報を表示して終了します。

mysqlimport の使用方法を表すサンプルのセッションを次に示します。

```
shell> mysql -e 'CREATE TABLE imptest(id INT, n VARCHAR(30))' test
shell> ed
a
100     Max Sydow
101     Count Dracula
.
w imptest.txt
32
q
shell> od -c imptest.txt
0000000   1   0   0  \t   M   a   x       S   y   d   o   w  \n   1   0
0000020   1  \t   C   o   u   n   t       D   r   a   c   u   l   a  \n
0000040
shell> mysqlimport --local test imptest.txt
test.imptest: Records: 2  Deleted: 0  Skipped: 0  Warnings: 0
shell> mysql -e 'SELECT * FROM imptest' test
+------+---------------+
| id   | n             |
+------+---------------+
|  100 | Max Sydow     |
|  101 | Count Dracula |
+------+---------------+
```

4.5.6 mysqlshow — データベース、テーブル、およびカラム情報の表示

mysqlshow クライアントは、どのデータベース、そのテーブル、またはテーブルのカラムまたはインデックスが存在するかを迅速に確認するために使用できます。

mysqlshow は複数の SQL SHOW ステートメントに対してコマンド行インタフェースを提供します。セクション13.7.5「SHOW 構文」を参照してください。それらステートメントを直接使用することで同じ情報を得られます。たとえば、mysql クライアントプログラムからそれらを発行できます。

mysqlshow は次のように起動します。

```
shell> mysqlshow [options] [db_name [tbl_name [col_name]]]
```

- データベースを指定しないと、データベース名のリストが表示されます。
- テーブルを指定しないと、データベース内のすべての一致するテーブルが表示されます。
- カラムを指定しないと、テーブル内のすべての一致するカラムとカラムの型が表示されます。

出力は、ユーザーが何らかの権限を所持しているデータベース、テーブル、またはカラムの名前のみを表示します。

最後の引数にシェルまたは SQL のワイルドカード文字 (「*」、「?」、「%」、または 「_」) が含まれている場合、そのワイルドカードに一致する名前だけが表示されます。データベース名に下線が含まれている場合は、正しいテーブルまたはカラムのリストを取得できるように、1 つのバックスラッシュ (一部の Unix シェルでは 2 つ必要) でエスケープ処理するようにしてください。「*」 文字および 「?」 文字は、SQL の 「%」 および 「_」 ワイルドカード文字に変換されます。これは名前に 「_」 を含むテーブルのカラムを表示しようとした際に問題を引き起こす場合があります。なぜなら、この場合 mysqlshow はパターンに一致するテーブル名のみを表示するからです。これは、コマンド行上で最後に 「%」 を別個の引数として追加することで簡単に修正できます。

mysqlshow は次のオプションをサポートします。これらはコマンド行またはオプションファイルの [mysqlshow] グループおよび [client] グループで指定できます。MySQL プログラムによって使用されるオプションファイルの詳細については、セクション4.2.6「オプションファイルの使用」を参照してください。

表 4.10 mysqlshow のオプション

オプション名	説明	導入
--bind-address	指定されたネットワークインタフェースを使用して MySQL サーバーに接続	5.6.1
--compress	クライアントとサーバー間で送信される情報をすべて圧縮	
--count	テーブルごとの行の数を表示	
--debug	デバッグのログを書き込み	
--debug-check	プログラムの終了時にデバッグ情報を出力	
--debug-info	プログラムの終了時に、デバッグ情報、メモリー、および CPU の統計を出力	
--default-auth	使用する認証プラグイン	5.6.2
--default-character-set	デフォルト文字セットを指定	
--defaults-extra-file	通常のオプションファイルに加えてオプションファイルを読み取る	
--defaults-file	指名されたオプションファイルのみを読み取る	
--defaults-group-suffix	オプショングループのサフィクス値	
--help	ヘルプメッセージを表示して終了	
--host	指定されたホスト上で MySQL サーバーに接続	
--keys	テーブルインデックスを表示	
--login-path	ログインパスオプションを .mylogin.cnf から読み取り	5.6.6
--no-defaults	オプションファイルを読み取らない	
--password	サーバーに接続する際に使用するパスワード	
--pipe	Windows で、名前付きパイプを使用してサーバーに接続	
--plugin-dir	プラグインがインストールされているディレクトリ	5.6.2
--port	接続に使用する TCP/IP ポート番号	
--print-defaults	デフォルトを出力	
--protocol	使用する接続プロトコル	
--secure-auth	古い (4.1.1 より前の) 形式でサーバーにパスワードを送信しない	5.6.17
--shared-memory-base-name	共有メモリー接続に使用する共有メモリーの名前	
--show-table-type	テーブルのタイプを示すカラムを表示	
--socket	ローカルホストへの接続で、使用する Unix ソケットファイル	
--ssl	接続に SSL を有効化	
--ssl-ca	信頼された SSL CA のリストを含むファイルのパス	
--ssl-capath	信頼された SSL CA の PEM 形式の証明書を含むディレクトリのパス	
--ssl-cert	PEM 形式の X509 証明書を含むファイルのパス	

オプション名	説明	導入
--ssl-cipher	SSL の暗号化に使用される、許可された暗号のリスト	
--ssl-crl	証明書失効リストを含むファイルのパス	5.6.3
--ssl-crlpath	証明書失効リストファイルを含むディレクトリのパス	5.6.3
--ssl-key	PEM 形式の X509 鍵を含むファイルのパス	
--ssl-verify-server-cert	サーバーへの接続時に、サーバーの証明書内のコモンネーム値をホスト名に対して検証	
--status	各テーブルの追加情報を表示	
--user	サーバーへの接続時に使用する MySQL ユーザー名	
--verbose	冗長モード	
--version	バージョン情報を表示して終了	

- --help, -?

  ヘルプメッセージを表示して終了します。

- --bind-address=ip_address

  複数のネットワークインタフェースを持つコンピュータで、このオプションを使用して、MySQL サーバーへの接続に使用するインタフェースを選択します。

  このオプションは MySQL 5.6.1 からサポートされています。

- --character-sets-dir=path

  文字セットがインストールされているディレクトリ。セクション10.5「文字セットの構成」を参照してください。

- --compress, -C

  クライアントとサーバーの両方が圧縮をサポートしている場合、その間で送受信される情報をすべて圧縮します。

- --count

  テーブルごとの行の数を表示します。これは MyISAM でないテーブルでは、遅い場合があります。

- --debug[=debug_options], -# [debug_options]

  デバッグのログを書き込みます。一般的な debug_options 文字列は d:t:o,file_name です。デフォルトは d:t:o です。

- --debug-check

  プログラムの終了時に、デバッグ情報を出力します。

- --debug-info

  プログラムの終了時に、デバッグ情報とメモリーおよび CPU 使用率の統計を出力します。

- --default-character-set=charset_name

  charset_name をデフォルト文字セットとして使用します。セクション10.5「文字セットの構成」を参照してください。

- --default-auth=plugin

  使用するクライアント側の認証プラグイン。セクション6.3.7「プラガブル認証」を参照してください。

  このオプションは MySQL 5.6.2 で追加されました。

- --defaults-extra-file=file_name

  このオプションファイルは、グローバルオプションファイルのあとに読み取りますが、(UNIX では) ユーザーオプションファイルの前に読み取るようにしてください。ファイルが存在しないかアクセスできない場合、エ

ラーが発生します。file_name は、フルパス名でなく相対パス名として指定された場合、現行ディレクトリを基準にして解釈されます。

- --defaults-file=file_name

  指定されたオプションファイルのみ使用します。ファイルが存在しないかアクセスできない場合、エラーが発生します。file_name は、フルパス名でなく相対パス名として指定された場合、現行ディレクトリを基準にして解釈されます。

- --defaults-group-suffix=str

  通常のオプショングループだけでなく、通常の名前に str のサフィクスが付いたグループも読み取ります。たとえば、mysqlshow は通常 [client] グループおよび [mysqlshow] グループを読み取ります。--defaults-group-suffix=_other オプションを指定した場合、mysqlshow は [client_other] グループおよび [mysqlshow_other] グループも読み取ります。

- --host=host_name, -h host_name

  指定されたホストの MySQL サーバーに接続します。

- --keys, -k

  テーブルインデックスを表示します。

- --login-path=name

  指名されたログインパスから .mylogin.cnf ログインファイルのオプションを読み取ります。「ログインパス」は、host、user、および password という限定されたオプションのセットのみを許可するオプショングループです。ログインパスは、サーバーホストおよびそのサーバーで認証するための認証情報を示す値のセットであると考えてください。ログインパスファイルを作成するには、mysql_config_editor ユーティリティーを使用します。セクション4.6.6「mysql_config_editor — MySQL 構成ユーティリティー」を参照してください。このオプションは MySQL 5.6.6 で追加されました。

- --no-defaults

  オプションファイルを読み取りません。オプションファイルから不明のオプションを読み取ることが原因でプログラムの起動に失敗する場合、--no-defaults を使用して、オプションを読み取らないようにできます。

  例外として、.mylogin.cnf ファイルは、存在する場合はすべての場合に読み取られます。これにより、--no-defaults が使用された場合にも、コマンド行よりも安全な方法でパスワードを指定できます。(.mylogin.cnf は mysql_config_editor ユーティリティーによって作成されます。セクション4.6.6「mysql_config_editor — MySQL 構成ユーティリティー」を参照してください)。

- --password[=password], -p[password]

  サーバーに接続する際に使用するパスワードです。短いオプション形式 (-p) を使用した場合は、オプションとパスワードの間にスペースを置くことはできません。コマンド行で、--password オプションまたは -p オプションに続けて password の値を指定しなかった場合、mysqlshow はそれを要求します。

  コマンド行でのパスワード指定は、セキュアでないと考えるべきです。セクション6.1.2.1「パスワードセキュリティーのためのエンドユーザーガイドライン」を参照してください。オプションファイルを使用すれば、コマンド行でパスワードを指定することを回避できます。

- --pipe, -W

  Windows で、名前付きパイプを使用してサーバーに接続します。このオプションは、サーバーが名前付きパイプ接続をサポートしている場合にのみ適用されます。

- --plugin-dir=path

  プラグインを検索するディレクトリ。--default-auth オプションを使用して認証プラグインを指定したが、mysqlshow がそれを検出できない場合は、このオプションを指定しなければならない可能性があります。セクション6.3.7「プラガブル認証」を参照してください。

  このオプションは MySQL 5.6.2 で追加されました。

- --port=port_num, -P port_num

  接続に使用する TCP/IP ポート番号。

- --print-defaults

  プログラム名と、オプションファイルから受け取るすべてのオプションを出力します。

- --protocol={TCP|SOCKET|PIPE|MEMORY}

  サーバーへの接続に使用する接続プロトコル。このオプションは、ほかの接続パラメータによって、必要なプロトコル以外のものが通常使用される場合に役立ちます。許可される値の詳細は、セクション4.2.2「MySQL サーバーへの接続」を参照してください。

- --secure-auth

  古い (4.1 より前の) 形式でサーバーにパスワードを送信しません。これにより、新しいパスワード形式を使用するサーバー以外への接続を防ぎます。このオプションはデフォルトで有効です。無効にするには --skip-secure-auth を使用します。このオプションは MySQL 5.6.17 で追加されました。

  > **注記**
  >
  > 4.1 より前のハッシュ方式を使用するパスワードはネイティブのパスワードハッシュ方式を使用するパスワードよりもセキュアでないため、使用しないようにしてください。4.1 よりも前のパスワードは非推奨であり、これらのサポートは今後の MySQL リリースで削除される予定です。アカウントのアップグレード手順については、セクション6.3.8.3「4.1 よりも前のパスワードハッシュ方式と mysql_old_password プラグインからの移行」を参照してください。

- --shared-memory-base-name=name

  Windows で、共有メモリーを使用して作成されるローカルサーバーへの接続の共有メモリー名。デフォルト値は MYSQL です。共有メモリー名では大文字と小文字を区別します。

  共有メモリー接続を可能にするには、サーバーは --shared-memory オプションで起動する必要があります。

- --show-table-type, -t

  SHOW FULL TABLES と同様に、テーブルの型を示すカラムを表示します。型は BASE TABLE または VIEW です。

- --socket=path, -S path

  localhost への接続用に使用する、Unix ソケットファイル、または Windows では使用する名前付きパイプの名前。

- --ssl*

  --ssl で始まるオプションは、SSL を使用してサーバーに接続することを許可するかどうかを指定し、SSL 鍵および証明書を検索する場所を指定します。セクション6.3.10.4「SSL コマンドのオプション」を参照してください。

- --status, -i

  各テーブルの追加情報を表示します。

- --user=user_name, -u user_name

  サーバーへの接続時に使用する MySQL ユーザー名。

- --verbose, -v

  冗長モード。プログラムの動作についてより多くの情報を出力します。このオプションは情報量を増加させるために複数回使用できます。

- --version, -V

  バージョン情報を表示して終了します。

4.5.7 mysqlslap — 負荷エミュレーションクライアント

mysqlslap は MySQL サーバーのクライアント負荷をエミュレートし、各段階のタイミングをレポートする診断プログラムです。複数のクライアントがサーバーにアクセスしているかのように作動します。

mysqlslap は次のように起動します。

```
shell> mysqlslap [options]
```

--create または --query などのオプションを使用すると、SQL ステートメントを含む文字列やステートメントを含むファイルを指定できます。ファイルを指定した場合、デフォルトでは各行に 1 つのステートメントを含んでいなければなりません。(つまり、暗黙的なステートメント区切り文字は改行文字です。)異なる区切り文字を指定するには、--delimiter オプションを使用します。これにより、複数行にわたるステートメントを指定したり、1 行に複数のステートメントを配置したりできます。ファイルにコメントを含めることはできません。mysqlslap はそれらを理解しません。

mysqlslap は次の 3 段階で実行されます。

1. テストに使用するスキーマ、テーブル、およびオプションでストアドプログラムまたはデータを作成します。この段階では、1 つのクライアント接続を使用します。
2. 負荷テストを実行します。この段階では、多数のクライアント接続を使用できます。
3. クリーンアップ (接続の解除、指定した場合はテーブルの削除) を実行します。この段階では、1 つのクライアント接続を使用します。

例:

50 台のクライアントがクエリーを実行し、それぞれ 200 の選択を行うような、create および query SQL ステートメントを提供します (コマンドは単一行に入力します)。

```
mysqlslap --delimiter=";"
  --create="CREATE TABLE a (b int);INSERT INTO a VALUES (23)"
  --query="SELECT * FROM a" --concurrency=50 --iterations=200
```

mysqlslap に、2 つの INT カラムと 3 つの VARCHAR カラムから成るテーブルを含む query SQL ステートメントを構築させます。5 台のクライアントを使ってそれぞれ 20 回ずつクエリーを実行します。テーブルを作成したり、データを挿入したりしないでください (直前のテストのスキーマとデータを使用します)。

```
mysqlslap --concurrency=5 --iterations=20
  --number-int-cols=2 --number-char-cols=3
  --auto-generate-sql
```

プログラムに、指定のファイルから create、insert、および query SQL ステートメントをロードするように指示します。この場合の create.sql ファイルには ';' で区切られた複数のテーブル作成ステートメントと ';' で区切られた複数の挿入ステートメントが含まれています。--query ファイルには、';' で区切られた複数のクエリーが格納されます。5 台のクライアントを使用して、すべての load ステートメントを実行してから、query ファイル内のすべてのクエリーを実行します (それぞれ 5 回ずつ)。

```
mysqlslap --concurrency=5
  --iterations=5 --query=query.sql --create=create.sql
  --delimiter=";"
```

mysqlslap は次のオプションをサポートします。これらはコマンド行またはオプションファイルの [mysqlslap] グループおよび [client] グループで指定できます。MySQL プログラムによって使用されるオプションファイルの詳細については、セクション4.2.6「オプションファイルの使用」を参照してください。

表 4.11 mysqlslap のオプション

オプション名	説明	導入
--auto-generate-sql	SQL ステートメントがファイルおよびコマンドオプションを使用して指定されない場合、自動的に生成	
--auto-generate-sql-add-autoincrement	AUTO_INCREMENT カラムを自動生成されたテーブルに追加	
--auto-generate-sql-execute-number	自動的に生成するクエリーの数を指定	
--auto-generate-sql-guid-primary	自動生成されたテーブルに GUID ベースの主キーを追加	
--auto-generate-sql-load-type	自動的に生成するクエリーの数を指定	
--auto-generate-sql-secondary-indexes	自動生成されたテーブルに追加するセカンダリインデックスの数を指定	
--auto-generate-sql-unique-query-number	自動テスト用に生成する異なるクエリーの数。	

オプション名	説明	導入
--auto-generate-sql-unique-write-number	--auto-generate-sql-write-number 用に生成する異なるクエリーの数	
--auto-generate-sql-write-number	各スレッドで実行する行挿入の回数	
--commit	コミットの前に実行するステートメントの数。	
--compress	クライアントとサーバー間で送信される情報をすべて圧縮	
--concurrency	SELECT ステートメントを発行する際、シミュレートするクライアントの数	
--create	テーブルの作成に使用するステートメントを含むファイルまたは文字列	
--create-schema	テストを実行するスキーマ	
--csv	カンマ区切りの値の形式で出力を生成	
--debug	デバッグのログを書き込み	
--debug-check	プログラムの終了時にデバッグ情報を出力	
--debug-info	プログラムの終了時に、デバッグ情報、メモリー、および CPU の統計を出力	
--default-auth	使用する認証プラグイン	5.6.2
--defaults-extra-file	通常のオプションファイルに加えてオプションファイルを読み取る	
--defaults-file	指名されたオプションファイルのみを読み取る	
--defaults-group-suffix	オプショングループのサフィクス値	
--delimiter	SQL ステートメントで使用する区切り文字	
--detach	N 個のステートメントが終わるごとに各接続を切り離す (閉じてからふたたび開く)	
--enable-cleartext-plugin	平文の認証プラグインを有効化	5.6.7
--engine	テーブルの作成に使用するストレージエンジン	
--help	ヘルプメッセージを表示して終了	
--host	指定されたホスト上で MySQL サーバーに接続	
--iterations	実行するテストの回数	
--login-path	ログインパスオプションを .mylogin.cnf から読み取り	5.6.6
--no-defaults	オプションファイルを読み取らない	
--no-drop	テスト実行中に作成されたスキーマを削除しない	5.6.3
--number-char-cols	--auto-generate-sql が指定された場合に使用する VARCHAR カラムの数	
--number-int-cols	--auto-generate-sql が指定された場合に使用する INT カラムの数	
--number-of-queries	各クライアントのクエリー数をおよそこの数に制限	
--only-print	データベースに接続しない。mysqlslap が実行したであろう内容を出力するのみ	
--password	サーバーに接続する際に使用するパスワード	
--pipe	Windows で、名前付きパイプを使用してサーバーに接続	
--plugin-dir	プラグインがインストールされているディレクトリ	5.6.2
--port	接続に使用する TCP/IP ポート番号	
--post-query	テスト完了後に実行するステートメントを含むファイルまたは文字列	
--post-system	テスト完了後に system() を使用して実行する文字列	
--pre-query	テストの実施前に実行するステートメントを含むファイルまたは文字列	

オプション名	説明	導入
--pre-system	テストの実施前に system() を使用して実行する文字列	
--print-defaults	デフォルトを出力	
--protocol	使用する接続プロトコル	
--query	データ取得のために使用する SELECT ステートメントを含むファイルまたは文字列	
--secure-auth	古い (4.1.1 より前の) 形式でサーバーにパスワードを送信しない	5.6.17
--shared-memory-base-name	共有メモリー接続に使用する共有メモリーの名前	
--silent	サイレントモード	
--socket	ローカルホストへの接続で、使用する Unix ソケットファイル	
--ssl	接続に SSL を有効化	
--ssl-ca	信頼された SSL CA のリストを含むファイルのパス	
--ssl-capath	信頼された SSL CA の PEM 形式の証明書を含むディレクトリのパス	
--ssl-cert	PEM 形式の X509 証明書を含むファイルのパス	
--ssl-cipher	SSL の暗号化に使用される、許可された暗号のリスト	
--ssl-crl	証明書失効リストを含むファイルのパス	5.6.3
--ssl-crlpath	証明書失効リストファイルを含むディレクトリのパス	5.6.3
--ssl-key	PEM 形式の X509 鍵を含むファイルのパス	
--ssl-verify-server-cert	サーバーへの接続時に、サーバーの証明書内のコモンネーム値をホスト名に対して検証	
--user	サーバーへの接続時に使用する MySQL ユーザー名	
--verbose	冗長モード	
--version	バージョン情報を表示して終了	

- --help, -?

  ヘルプメッセージを表示して終了します。

- --auto-generate-sql, -a

  SQL ステートメントがファイルおよびコマンドオプションを使用して指定されない場合、自動的に生成します。

- --auto-generate-sql-add-autoincrement

  AUTO_INCREMENT カラムを自動生成されたテーブルに追加します。

- --auto-generate-sql-execute-number=N

  自動的に生成するクエリーの数を指定します。

- --auto-generate-sql-guid-primary

  自動生成されたテーブルに GUID ベースの主キーを追加します。

- --auto-generate-sql-load-type=type

  テストの負荷タイプを指定します。許可される値は、read (テーブルのスキャン)、write (テーブルに挿入)、key (主キーの読み取り)、update (主キーの更新)、または mixed (挿入とスキャンして選択が半分ずつ) です。デフォルトは mixed です。

- --auto-generate-sql-secondary-indexes=N

  自動生成されたテーブルに追加するセカンダリインデックスの数を指定します。デフォルトでは、何も追加されません。

- --auto-generate-sql-unique-query-number=N

自動テスト用に生成する異なるクエリーの数。たとえば、1000 回の選択を行う key テストを実施する場合、このオプションの値を 1000 にして一意のクエリーを 1000 個実行することも、値を 50 にして異なるクエリーを 50 回行うこともできます。デフォルトは 10 です。

- --auto-generate-sql-unique-write-number=N

  --auto-generate-sql-write-number 用に生成する異なるクエリーの数を指定します。デフォルトは 10 です。

- --auto-generate-sql-write-number=N

  各スレッドで実行する行挿入の回数を指定します。デフォルトは 100 です。

- --commit=N

  コミットの前に実行するステートメントの数。デフォルトは 0 (コミットは行われない) です。

- --compress, -C

  クライアントとサーバーの両方が圧縮をサポートしている場合、その間で送受信される情報をすべて圧縮します。

- --concurrency=N, -c N

  SELECT ステートメントを発行する場合に、シミュレートするクライアントの数。

- --create=value

  テーブルの作成に使用するステートメントを含むファイルまたは文字列。

- --create-schema=value

  テストを実行するスキーマ。

  > **注記**
  >
  > --auto-generate-sql オプションも指定されている場合、mysqlslap はテスト実行の最後にスキーマを削除します。これを避けるには、--no-drop オプションも使用します。

- --csv[=file_name]

  カンマ区切りの値の形式で出力を生成します。出力は指定されたファイルか、ファイルが指定されていない場合標準出力に送られます。

- --debug[=debug_options], -# [debug_options]

  デバッグのログを書き込みます。一般的な debug_options 文字列は d:t:o,file_name です。デフォルトは d:t:o,/tmp/mysqlslap.trace です。

- --debug-check

  プログラムの終了時に、デバッグ情報を出力します。

- --debug-info, -T

  プログラムの終了時に、デバッグ情報とメモリーおよび CPU 使用率の統計を出力します。

- --default-auth =plugin

  使用するクライアント側の認証プラグイン。セクション6.3.7「プラガブル認証」を参照してください。

  このオプションは MySQL 5.6.2 で追加されました。

- --defaults-extra-file=file_name

  このオプションファイルは、グローバルオプションファイルのあとに読み取りますが、(UNIX では) ユーザーオプションファイルの前に読み取るようにしてください。ファイルが存在しないかアクセスできない場合、エラーが発生します。file_name は、フルパス名でなく相対パス名として指定された場合、現行ディレクトリを基準にして解釈されます。

- --defaults-file=file_name

指定されたオプションファイルのみ使用します。ファイルが存在しないかアクセスできない場合、エラーが発生します。file_name は、フルパス名でなく相対パス名として指定された場合、現行ディレクトリを基準にして解釈されます。

- --defaults-group-suffix=str

  通常のオプショングループだけでなく、通常の名前に str のサフィクスが付いたグループも読み取ります。たとえば、mysqlslap は通常 [client] グループおよび [mysqlslap] グループを読み取ります。--defaults-group-suffix=_other オプションを指定した場合、mysqlslap は [client_other] グループおよび [mysqlslap_other] グループも読み取ります。

- --delimiter=str, -F str

  ファイルまたはコマンドオプションを使用して提供される SQL ステートメントで使用される区切り文字。

- --detach=N

  N 個のステートメントごとに各接続を切り離します (閉じてからふたたび開きます)。デフォルトは 0 (接続は切り離されない) です。

- --enable-cleartext-plugin

  mysql_clear_password 平文認証プラグインを有効にします。(セクション6.3.8.7「クライアント側のクリアテキスト認証プラグイン」を参照してください。)このオプションは MySQL 5.6.7 で追加されました。

- --engine=engine_name, -e engine_name

  テーブルの作成に使用するストレージエンジンを指定します。

- --host=host_name, -h host_name

  指定されたホストの MySQL サーバーに接続します。

- --iterations=N, -i N

  実行するテストの回数。

- --login-path=name

  指名されたログインパスから .mylogin.cnf ログインファイルのオプションを読み取ります。「ログインパス」は、host、user、および password という限定されたオプションのセットのみを許可するオプショングループです。ログインパスは、サーバーホストおよびそのサーバーで認証するための認証情報を示す値のセットであると考えてください。ログインパスファイルを作成するには、mysql_config_editor ユーティリティーを使用します。セクション4.6.6「mysql_config_editor — MySQL 構成ユーティリティー」を参照してください。このオプションは MySQL 5.6.6 で追加されました。

- --no-drop

  mysqlslap がテスト実行中に作成するスキーマをドロップしないようにします。このオプションは MySQL 5.6.3 で追加されました。

- --no-defaults

  オプションファイルを読み取りません。オプションファイルから不明のオプションを読み取ることが原因でプログラムの起動に失敗する場合、--no-defaults を使用して、オプションを読み取らないようにできます。

  例外として、.mylogin.cnf ファイルは、存在する場合はすべての場合に読み取られます。これにより、--no-defaults が使用された場合にも、コマンド行よりも安全な方法でパスワードを指定できます。(.mylogin.cnf は mysql_config_editor ユーティリティーによって作成されます。セクション4.6.6「mysql_config_editor — MySQL 構成ユーティリティー」を参照してください)。

- --number-char-cols=N, -x N

  --auto-generate-sql が指定されている場合に使用する VARCHAR カラムの数。

- --number-int-cols=N, -y N

  --auto-generate-sql が指定されている場合に使用する INT カラムの数。

- --number-of-queries=N

  各クライアントのクエリー数をおよそこの数に制限します。クエリーのカウントには、ステートメント区切り文字が考慮されます。たとえば、mysqlslap を次のように起動する場合、; 区切り文字が認識され、クエリー文字列の各インスタンスは 2 つのクエリーとカウントされます。その結果、(10 ではなく) 5 つの行が挿入されます。

```
shell> mysqlslap --delimiter=";" --number-of-queries=10
       --query="use test;insert into t values(null)"
```

- --only-print

  データベースには接続しません。mysqlslap は、実行したであろう内容を出力するだけです。

- --password[=password], -p[password]

  サーバーに接続する際に使用するパスワードです。短いオプション形式 (-p) を使用した場合は、オプションとパスワードの間にスペースを置くことはできません。コマンド行で、--password オプションまたは -p オプションに続けて password の値を指定しなかった場合、mysqlslap はそれを要求します。

  コマンド行でのパスワード指定は、セキュアでないと考えるべきです。セクション6.1.2.1「パスワードセキュリティーのためのエンドユーザーガイドライン」を参照してください。オプションファイルを使用すれば、コマンド行でパスワードを指定することを回避できます。

- --pipe, -W

  Windows で、名前付きパイプを使用してサーバーに接続します。このオプションは、サーバーが名前付きパイプ接続をサポートしている場合にのみ適用されます。

- --plugin-dir=path

  プラグインを検索するディレクトリ。--default-auth オプションを使用して認証プラグインを指定したが、mysqlslap がそれを検出できない場合は、このオプションを指定しなければならない可能性があります。セクション6.3.7「プラガブル認証」を参照してください。

  このオプションは MySQL 5.6.2 で追加されました。

- --port=port_num, -P port_num

  接続に使用する TCP/IP ポート番号。

- --post-query=value

  テスト完了後に実行するステートメントを含むファイルまたは文字列。この実行は、タイミングの目的ではカウントされません。

- --post-system=str

  テスト完了後に system() を使用して実行する文字列。この実行は、タイミングの目的ではカウントされません。

- --pre-query=value

  テストの実行前に実行するステートメントを含むファイルまたは文字列。この実行は、タイミングの目的ではカウントされません。

- --pre-system=str

  テストの実行前に system() を使用して実行する文字列。この実行は、タイミングの目的ではカウントされません。

- --print-defaults

  プログラム名と、オプションファイルから受け取るすべてのオプションを出力します。

- --protocol={TCP|SOCKET|PIPE|MEMORY}

  サーバーへの接続に使用する接続プロトコル。このオプションは、ほかの接続パラメータによって、必要なプロトコル以外のものが通常使用される場合に役立ちます。許可される値の詳細は、セクション4.2.2「MySQL サーバーへの接続」を参照してください。

- --query=value, -q value

  データ取得のため使用する SELECT ステートメントを含むファイルまたは文字列。

- --secure-auth

  古い (4.1 より前の) 形式でサーバーにパスワードを送信しません。これにより、新しいパスワード形式を使用するサーバー以外への接続を防ぎます。このオプションはデフォルトで有効です。無効にするには --skip-secure-auth を使用します。このオプションは MySQL 5.6.17 で追加されました。

  > **注記**
  >
  > 4.1 より前のハッシュ方式を使用するパスワードはネイティブのパスワードハッシュ方式を使用するパスワードよりもセキュアでないため、使用しないようにしてください。4.1 よりも前のパスワードは非推奨であり、これらのサポートは今後の MySQL リリースで削除される予定です。アカウントのアップグレード手順については、セクション6.3.8.3「4.1 よりも前のパスワードハッシュ方式と mysql_old_password プラグインからの移行」を参照してください。

- --shared-memory-base-name=name

  Windows で、共有メモリーを使用して作成されるローカルサーバーへの接続の共有メモリー名。このオプションは、サーバーが共有メモリー接続をサポートしている場合にのみ適用されます。

- --silent, -s

  サイレントモード。出力はありません。

- --socket=path, -S path

  localhost への接続用に使用する、Unix ソケットファイル、または Windows では使用する名前付きパイプの名前。

- --ssl*

  --ssl で始まるオプションは、SSL を使用してサーバーに接続することを許可するかどうかを指定し、SSL 鍵および証明書を検索する場所を指定します。セクション6.3.10.4「SSL コマンドのオプション」を参照してください。

- --user=user_name, -u user_name

  サーバーへの接続時に使用する MySQL ユーザー名。

- --verbose, -v

  冗長モード。プログラムの動作についてより多くの情報を出力します。このオプションは情報量を増加させるために複数回使用できます。

- --version, -V

  バージョン情報を表示して終了します。

4.6 MySQL 管理プログラムおよびユーティリティープログラム

このセクションでは、管理プログラムおよびその他のユーティリティー操作を実行するプログラムについて説明します。

4.6.1 innochecksum — オフライン InnoDB ファイルチェックサムユーティリティー

innochecksum は、InnoDB ファイルのチェックサムを出力します。このツールは InnoDB テーブルスペースファイルを読み取って各ページのチェックサムを計算し、計算されたチェックサムを保存されているチェックサムと比較して不一致をレポートします。不一致はページが破損していることを示します。元は、停電後にテーブルスペースの完全性の検証を迅速化するために開発されましたが、ファイルコピーのあとにも使用できます。チェックサムの不一致があると、InnoDB は稼働中のサーバーを意図的にシャットダウンするため、本番稼働しているサーバーが破損したページに遭遇するのを待つのではなく、このツールを使用することが推奨されま

す。MySQL 5.6.16 では、innochecksum は 2G バイトを超えるサイズのファイルをサポートしています。以前は、innochecksum は 2G バイトまでのサイズのファイルだけをサポートしていました。

innochecksum は、サーバーがすでにオープンしているテーブルスペースファイルには使用できません。このようなファイルに関しては、CHECK TABLE を使用してテーブルスペース内のテーブルをチェックするとよいでしょう。

チェックサムの不一致が検出された場合、通常はバックアップからテーブルスペースをリストアするか、またはサーバーを起動して、mysqldump を使用してテーブルスペース内のテーブルのバックアップを作成します。

innochecksum は次のように起動します。

```
shell> innochecksum [options] file_name
```

innochecksum は次のオプションをサポートします。ページ番号を参照するオプションについては、数字はゼロベースです。

- -c

  ファイル内のページ数のカウントを出力します。

- -d

  デバッグモード。各ページのチェックサムを出力します。

- -e num

  このページ番号で終了します。

- -p num

  このページ番号のみをチェックします。

- -s num

  このページ番号から開始します。

- -v

  詳細モード。5 秒ごとに進行状況インジケータを出力します。

4.6.2 myisam_ftdump — 全文インデックス情報の表示

myisam_ftdump は MyISAM テーブル内の FULLTEXT インデックスに関する情報を表示します。MyISAM インデックスファイルを直接読み取るため、テーブルのあるサーバーホスト上で実行する必要があります。サーバーが稼働中の場合は、myisam_ftdump を使用する前に、必ず最初に FLUSH TABLES ステートメントを発行してください。

myisam_ftdump はインデックス全体をスキャンしてダンプします。これは特に高速ではありません。一方、単語の分布の変更頻度は高くないため、頻繁に実行する必要はありません。

myisam_ftdump は次のように起動します。

```
shell> myisam_ftdump [options] tbl_name index_num
```

tbl_name 引数は MyISAM テーブルの名前であるべきです。インデックスファイル (.MYI サフィクスの付いたファイル) を指名することでテーブルを指定することもできます。テーブルファイルがあるディレクトリで myisam_ftdump を起動しない場合は、テーブルまたはインデックスのファイル名の前に、テーブルのデータベースディレクトリのパス名を指定する必要があります。インデックス番号は 0 から始まります。

例: test データベースに、次の定義を持つ mytexttable という名前のテーブルが含まれるとします。

```
CREATE TABLE mytexttable
(
 id   INT NOT NULL,
 txt  TEXT NOT NULL,
 PRIMARY KEY (id),
 FULLTEXT (txt)
```

```
) ENGINE=MyISAM;
```

id のインデックスはインデックス 0 で、txt の FULLTEXT インデックスはインデックス 1 です。作業ディレクトリが test データベースディレクトリの場合は、myisam_ftdump を次のように起動します。

```
shell> myisam_ftdump mytexttable 1
```

test データベースディレクトリへのパス名が /usr/local/mysql/data/test の場合、そのパス名を指定してテーブル名引数を指定することもできます。これは、データベースディレクトリ内で myisam_ftdump を起動しない場合役に立ちます。

```
shell> myisam_ftdump /usr/local/mysql/data/test/mytexttable 1
```

次のように、myisam_ftdump を使用して、インデックスエントリのリストを出現頻度順に生成できます。

```
shell> myisam_ftdump -c mytexttable 1 | sort -r
```

myisam_ftdump は次のオプションをサポートします。

- --help, -h -?

  ヘルプメッセージを表示して終了します。

- --count, -c

  1 語当たりの統計 (カウントとグローバルな重み) を計算します。

- --dump, -d

  データオフセットや語の重みを含むインデックスをダンプします。

- --length, -l

  長さの分布をレポートします。

- --stats, -s

  グローバルインデックス統計をレポートします。ほかの操作が指定されていない場合、これがデフォルトの操作です。

- --verbose, -v

  冗長モード。プログラムの動作についてより多くの情報を出力します。

4.6.3 myisamchk — MyISAM テーブルメンテナンスユーティリティー

myisamchk ユーティリティーは、データベーステーブルに関する情報を取得したり、データベーステーブルのチェック、修復、または最適化を実行したりします。myisamchk は MyISAM テーブル (データおよびインデックスの保存のための .MYD および .MYI ファイルを持つテーブル) に機能します。

さらに、CHECK TABLE および REPAIR TABLE ステートメントを使用して、MyISAM テーブルをチェックして修復することもできます。セクション13.7.2.2「CHECK TABLE 構文」およびセクション13.7.2.5「REPAIR TABLE 構文」を参照してください。

myisamchk をパーティション化されたテーブルに対して使用することはサポートされていません。

> **注意**
>
> テーブルの修復操作を実行する前に、テーブルのバックアップを作成することをお勧めします。状況によっては、この操作のためにデータ損失が発生することがあります。考えられる原因としては、ファイルシステムのエラーなどがありますがこれに限りません。

myisamchk は次のように起動します。

```
shell> myisamchk [options] tbl_name ...
```

options は myisamchk に実行させる内容を指定します。次のセクションで、これらについて説明します。また、myisamchk --help を起動することでオプションのリストを取得できます。

オプションを指定しないと、myisamchk はデフォルトの操作としてユーザーのテーブルをチェックします。詳細な情報を取得したり、myisamchk に修正アクションを取らせたりするには、次の議論で説明されているようにオプションを指定してください。

tbl_name は、チェックまたは修復するデータベーステーブルです。データベースディレクトリ以外で myisamchk を起動する場合は、データベースディレクトリへのパスを指定する必要があります。これは、myisamchk にはデータベースディレクトリの場所がまったくわからないからです。実際には、myisamchk は作業対象のファイルがデータベースディレクトリにあるかどうかは考慮しません。データベーステーブルに応対するファイルをほかの場所へコピーして、そこでそれらのファイルに対してリカバリ操作を行うことができます。

必要に応じて、myisamchk コマンド行で複数のテーブルを指定できます。インデックスファイル (.MYI サフィクスの付いたファイル) を指名することでテーブルを指定することもできます。これにより、パターン *.MYI を使用して。ディレクトリ内のすべてのテーブルを指定することも可能になります。たとえば、データベースディレクトリ内に居る場合、次のようにそのディレクトリ内のすべての MyISAM テーブルをチェックできます。

```
shell> myisamchk *.MYI
```

データベースディレクトリ以外の場所からは、ディレクトリへのパスを指定することですべてのテーブルをチェックできます。

```
shell> myisamchk /path/to/database_dir/*.MYI
```

MySQL データディレクトリへのパスにワイルドカードを指定することで、すべてのデータベースのすべてのテーブルをチェックすることもできます。

```
shell> myisamchk /path/to/datadir/*/*.MYI
```

すべての MyISAM テーブルをチェックするお勧めの方法は:

```
shell> myisamchk --silent --fast /path/to/datadir/*/*.MYI
```

すべての MyISAM テーブルをチェックし、破損しているものを修復する場合は、次のコマンドを使用できます。

```
shell> myisamchk --silent --force --fast --update-state \
       --key_buffer_size=64M --myisam_sort_buffer_size=64M \
       --read_buffer_size=1M --write_buffer_size=1M \
       /path/to/datadir/*/*.MYI
```

このコマンドは 64M バイト以上の空きがあることが前提です。myisamchk とメモリーの割り当ての詳細は、セクション4.6.3.6「myisamchk メモリー使用量」を参照してください。

myisamchk の使用に関する詳細は、セクション7.6「MyISAM テーブルの保守とクラッシュリカバリ」を参照してください。

> **重要**
>
> myisamchk の実行中にほかのプログラムがテーブルを使用しないことを、保証する必要があります。そのためのもっとも効果的な方法は、myisamchk の実行中に MySQL サーバーをシャットダウンするか、または myisamchk が対象とするすべてのテーブルをロックする方法です。
>
> そうしないと、myisamchk を起動したとき、次のエラーが表示されることがあります。
>
> ```
> warning: clients are using or haven't closed the table properly
> ```
>
> これは、別のプログラム (mysqld サーバーなど) がテーブルを更新し、そのファイルをまだ閉じていないか、ファイルを適切に閉じずに異常終了したテーブルをチェックしようとしていることを意味します。この場合、1 つまたは複数の MyISAM テーブルが破損することがあります。
>
> mysqld が稼働している場合、FLUSH TABLES を使用して、メモリーにバッファリングされているテーブルに加えられた変更があれば、それをフラッシュするように命令する必要があります。そのあと、myisamchk の実行中にほかのプログラムがテーブルを使用しないことを、保証する必要があります。
>
> ただし、この問題を回避するもっとも簡単な方法は、myisamchk ではなく CHECK TABLE を使用してテーブルをチェックする方法です。セクション13.7.2.2「CHECK TABLE 構文」を参照してください。

myisamchk は次のオプションをサポートします。これらはコマンド行または任意のオプションファイルの [myisamchk] グループで指定できます。MySQL プログラムによって使用されるオプションファイルの詳細については、セクション4.2.6「オプションファイルの使用」を参照してください。

表 4.12 myisamchk のオプション

オプション名	説明	導入	非推奨
--analyze	キー値の分布を分析		
--backup	.MYD ファイルのバックアップを file_name-time.BAK として作成		
--block-search	指定されたオフセットのブロックが属するレコードを検索		
--check	テーブルにエラーがないか確認		
--check-only-changed	最後に行われた検査以降に変更されたテーブルのみをチェック		
--correct-checksum	テーブルのチェックサム情報を修正		
--data-file-length	データファイルの最大長 (データファイルがいっぱいになったとき再作成する場合)		
--debug	デバッグのログを書き込み		
--decode_bits	Decode_bits		
--defaults-extra-file	通常のオプションファイルに加えてオプションファイルを読み取る		
--defaults-file	指名されたオプションファイルのみを読み取る		
--defaults-group-suffix	オプショングループのサフィクス値		
--description	テーブルの説明情報を出力		
--extend-check	データファイルからすべての行をリカバリする修復を試みる、非常に徹底したテーブルチェックを実行または修復を実行		
--fast	正しく閉じられていないテーブルのみをチェック		
--force	myisamchk がエラーをテーブル内で発見した場合、自動的に修復オペレーションを実行する。		
--force	古い一時ファイルを上書き。-r オプションまたは -o オプションとともに使用		
--ft_max_word_len	FULLTEXT インデックスの単語の最大長		
--ft_min_word_len	FULLTEXT インデックスの単語の最小長		
--ft_stopword_file	組み込みのリストではなくこのファイルからのストップワードを使用		
--HELP	ヘルプメッセージを表示して終了		
--help	ヘルプメッセージを表示して終了		
--information	チェックされたテーブルの統計を出力		
--key_buffer_size	MyISAM テーブルのインデックスブロックに使用するバッファーのサイズ		
--keys-used	どのインデックスを更新するかを示すビット値		
--max-record-length	myisamchk が記憶するためのメモリーを確保できない場合、指定された長さを超える行をスキップ		
--medium-check	--extend-check 操作よりも速いチェックを実行		
--myisam_block_size	MyISAM インデックスページに使用するブロックサイズ。		
--myisam_sort_buffer_size	REPAIR 実行時のインデックスのソート、または CREATE INDEX か ALTER TABLE によるインデックスの作成の際に割り当てられるバッファー	5.6.9	
--no-defaults	オプションファイルを読み取らない		

オプション名	説明	導入	非推奨
--parallel-recover	-r および -n と同じテクニックを使用するが、異なるスレッドを使用してすべてのキーを並行して作成 (β)		
--print-defaults	デフォルトを出力		
--quick	データファイルを変更しないことで、修復のスピードを向上。		
--read_buffer_size	順次スキャンを実行する各スレッドは、スキャンする各テーブルについてこのサイズのバッファーを割り当て		
--read-only	テーブルを検査済みとマークしない		
--recover	一意ではない一意なキー以外のすべてを修復できる修復を実行		
--safe-recover	すべての行を順に読み取り、検出された行に基づいてすべてのインデックスツリーを更新する古いリカバリ方法を使用して修復		
--set-auto-increment	新しいレコードが指定された値で開始するように AUTO_INCREMENT ナンバリングを強制		
--set-collation	テーブルインデックスのソートに使用する照合順序を指定		
--silent	サイレントモード		
--sort_buffer_size	REPAIR 実行時のインデックスのソート、または CREATE INDEX か ALTER TABLE によるインデックスの作成の際に割り当てられるバッファー		5.6.9
--sort-index	インデックスツリーブロックを高いものから低いものへの順にソート		
--sort_key_blocks	sort_key_blocks		
--sort-records	特定のインデックスに基づいてレコードをソート		
--sort-recover	一時ファイルのサイズが非常に大きくなっても、キーの解決にソートを使用することを myisamchk に強制		
--stats_method	MyISAM インデックス統計コレクションコードでの NULL の取り扱い方法を指定		
--tmpdir	一時ファイルのソートに使用するディレクトリのパス		
--unpack	myisampack でパックされたテーブルをアンパック		
--update-state	情報を .MYI ファイルに保存し、いつテーブルがチェックされたか、およびテーブルがクラッシュしたかどうかをチェックします。		
--verbose	冗長モード		
--version	バージョン情報を表示して終了		
--write_buffer_size	書き込みバッファーサイズ		

4.6.3.1 myisamchk の一般オプション

このセクションで紹介されているオプションは myisamchk によって実行されるすべてのテーブルメンテナンス操作に使用できます。このセクション以降のセクションは、テーブルのチェックまたは修復などの特定の操作のみに関するオプションを説明します。

- --help, -?

  ヘルプメッセージを表示して終了します。オプションは操作の種類によってグループ化されています。

- --HELP, -H

  ヘルプメッセージを表示して終了します。オプションは単独のリストに提示されます。

- --debug=debug_options, -# debug_options

デバッグのログを書き込みます。一般的な debug_options 文字列は d:t:o,file_name です。デフォルトは d:t:o,/tmp/myisamchk.trace です。

- --defaults-extra-file=file_name

  このオプションファイルは、グローバルオプションファイルのあとに読み取りますが、(UNIX では) ユーザーオプションファイルの前に読み取るようにしてください。ファイルが存在しないかアクセスできない場合、エラーが発生します。file_name は、フルパス名でなく相対パス名として指定された場合、現行ディレクトリを基準にして解釈されます。

- --defaults-file=file_name

  指定されたオプションファイルのみ使用します。ファイルが存在しないかアクセスできない場合、エラーが発生します。file_name は、フルパス名でなく相対パス名として指定された場合、現行ディレクトリを基準にして解釈されます。

- --defaults-group-suffix=str

  通常のオプショングループだけでなく、通常の名前に str のサフィクスが付いたグループも読み取ります。たとえば、myisamchk は通常 [myisamchk] グループを読み取ります。--defaults-group-suffix=_other オプションを指定した場合、myisamchk は [myisamchk_other] グループも読み取ります。

- --no-defaults

  オプションファイルを読み取りません。オプションファイルから不明のオプションを読み取ることが原因でプログラムの起動に失敗する場合、--no-defaults を使用して、オプションを読み取らないようにできます。

  例外として、.mylogin.cnf ファイルは、存在する場合はすべての場合に読み取られます。これにより、--no-defaults が使用された場合にも、コマンド行よりも安全な方法でパスワードを指定できます。(.mylogin.cnf は mysql_config_editor ユーティリティーによって作成されます。セクション4.6.6「mysql_config_editor — MySQL 構成ユーティリティー」を参照してください)。

- --print-defaults

  プログラム名と、オプションファイルから受け取るすべてのオプションを出力します。

- --silent, -s

  サイレントモード。エラーが発生したときのみ出力を書き出します。-s は myisamchk を非常にサイレントにするために 2 回 (-ss) 使用できます。

- --verbose, -v

  冗長モード。プログラムの動作についてより多くの情報を出力します。これは -d と -e とともに使用できます。さらに多くの出力を得るためには、-v を複数回 (-vv、-vvv) 使用します。

- --version, -V

  バージョン情報を表示して終了します。

- --wait, -w

  テーブルがロックされている場合に、エラーで終了する代わりに、テーブルがロック解除されるまで待機してから続行します。外部ロックを無効にした状態で mysqld を実行している場合、テーブルをロックできるのはもう 1 つの myisamchk コマンドのみです。

--var_name=value 構文を使用すれば、次の変数も設定できます。

変数	デフォルト値
decode_bits	9
ft_max_word_len	バージョンに依存
ft_min_word_len	4
ft_stopword_file	組み込みのリスト
key_buffer_size	523264
myisam_block_size	1024

変数	デフォルト値
myisam_sort_key_blocks	16
read_buffer_size	262136
sort_buffer_size	2097144
sort_key_blocks	16
stats_method	nulls_unequal
write_buffer_size	262136

可能な myisamchk 変数とデフォルト値は myisamchk --help で確認できます。

sort_buffer_size はキーをソートすることでキーが修復される場合に使用されます。--recover を使用した場合はこれが通常です。MySQL 5.6.9 では、myisam_sort_buffer_size は sort_buffer_size の別名として使用できます。myisam_sort_buffer_size は、その名前が同様の意味を持つ myisam_sort_buffer_size サーバーシステム変数に対応するため、sort_buffer_size より推奨されます。sort_buffer_size は非推奨とみなすようにしてください。

key_buffer_size は --extend-check でテーブルをチェックするとき、またはテーブルに (通常の挿入を実行する場合のように) 1 行ずつキーを入力することでキーを修復する際に使用されます。キーバッファーを通しての修復は次の場合に使用されます。

- --safe-recover を使用する。
- キーのソートに必要な一時ファイルが、直接キーファイルを作成する場合と比較して倍以上の大きさとなる。これは CHAR、VARCHAR、または TEXT カラムに大きなキー値が与えられている場合によくあります。これは、ソート操作では処理中に完全なキー値を保存する必要があるからです。一時スペースに余裕があり、ソートによって修復することを myisamchk に強制できる場合、--sort-recover オプションを使用できます。

キーバッファーを使用する修復はソートを使用するよりもはるかにディスクの使用量が少ないですが、速度も落ちます。

より高速に修復を行いたい場合は、key_buffer_size 変数および myisam_sort_buffer_size 変数を、使用可能なメモリーの約 25% に設定します。両方の変数が同時に使用されることはないので、両方の変数に大きい値を設定できます。

myisam_block_size はインデックスブロックに使用されるサイズです。

stats_method は、--analyze オプションを指定した場合に、インデックス統計の集計で NULL 値がどう扱われるかに影響します。myisam_stats_method システム変数のような働きをします。詳細は、セクション5.1.4「サーバーシステム変数」とセクション8.3.7「InnoDB および MyISAM インデックス統計コレクション」に含まれる myisam_stats_method の説明を参照してください。

ft_min_word_len および ft_max_word_len は、MyISAM テーブルの FULLTEXT インデックスの語長を示します。ft_stopword_file はストップワードファイルを指名します。次の状況ではこれらをセットする必要があります。

myisamchk を使用してテーブルインデックスを変更する操作を実行する場合 (たとえば修復や分析)、特に指定しなければ、FULLTEXT インデックスは、最小および最大の語長とストップワードファイルにデフォルトの全文パラメータ値を使用して再構築されます。これにより、クエリーに失敗する可能性があります。

この問題は、これらのパラメータがサーバーでのみ認識されていることが原因で発生します。MyISAM インデックスファイルには格納されていません。サーバー内の最小および最大の語長またはストップワードファイルを変更した場合に問題を回避するには、mysqld で使用するのと同じ ft_min_word_len、ft_max_word_len と ft_stopword_file の値を myisamchk に指定してください。たとえば、単語の最小長を 3 に設定した場合は、次のように myisamchk を使用してテーブルを修復できます。

```
shell> myisamchk --recover --ft_min_word_len=3 tbl_name.MYI
```

myisamchk とサーバーが全文パラメータに同じ値を確実に使用するには、オプションファイルの [mysqld] と [myisamchk] セクションの両方にそれぞれを置いてください。

```
[mysqld]
ft_min_word_len=3

[myisamchk]
ft_min_word_len=3
```

myisamchk を使用する代わりに、REPAIR TABLE、ANALYZE TABLE、OPTIMIZE TABLE、または ALTER TABLE を使用できます。これらのステートメントは、適切に使用される全文パラメータ値が認識されているサーバーで実行されます。

4.6.3.2 myisamchk のチェックオプション

myisamchk はテーブルチェック操作用に次のオプションをサポートします。

- --check, -c

  テーブルにエラーがないか確認します。明示的に操作のタイプを選択しない場合、これがデフォルトの操作です。

- --check-only-changed, -C

  最後に行われた検査以降に変更されたテーブルのみをチェックします。

- --extend-check, -e

  テーブルチェックを非常に詳細に行います。テーブルにインデックスが多数ある場合、これには非常に時間がかかります。このオプションは極端な場合のみに使用するべきです。通常、myisamchk または myisamchk --medium-check を使用してテーブル内のエラーの有無を確認できます。

  --extend-check を使用し、メモリー容量が十分な場合、key_buffer_size 値を大きくセットすれば、修復操作のスピードを上げることができます。

  このオプションの説明は、テーブル修復オプションも参照してください。

  出力形式の説明は、セクション4.6.3.5「myisamchk によるテーブル情報の取得」を参照してください。

- --fast, -F

  正しく閉じられていないテーブルのみを検査します。

- --force, -f

  myisamchk がテーブル内にエラーを発見した場合、自動的に修復操作を実行します。修復タイプは --recover または -r オプションで指定されるものと同じです。

- --information, -i

  チェックされたテーブルの統計を出力します。

- --medium-check, -m

  --extend-check 操作よりも高速なチェックを実行します。これはすべてのエラーの 99.99% のみを確認し、ほとんどの場合はこれで十分でしょう。

- --read-only, -T

  テーブルを検査済みとマークしません。これは、mysqld が外部ロックが無効の状態で稼働している場合など、ロックを使用しないほかのアプリケーションが使用しているテーブルをチェックするために myisamchk を使用する場合に便利です。

- --update-state, -U

  情報を .MYI ファイルに保存して、テーブルがチェックされた時期およびテーブルがクラッシュしたかどうかを示します。--check-only-changed オプションの利便性を最大限に引き出すためにはこれを使用するべきですが、mysqld サーバーがテーブルを使用し、かつ外部ロックを無効にした状態で起動している場合は、このオプションを使用しないでください。

4.6.3.3 myisamchk の修復オプション

myisamchk は、テーブルの修復操作 (--recover オプションまたは --safe-recover オプションなどのオプションが指定された場合に実行される操作) のために次のオプションをサポートします。

- --backup, -B

  .MYD ファイルのバックアップを file_name-time.BAK として作成します。

- --character-sets-dir=path

  文字セットがインストールされているディレクトリ。セクション10.5「文字セットの構成」を参照してください。

- --correct-checksum

  テーブルのチェックサム情報を修正します。

- --data-file-length=len, -D len

  データファイルの最大長 (データファイルが「いっぱい」になったとき再作成する場合)。

- --extend-check, -e

  データファイルからできるかぎりの行をリカバリしようとする修復を実行します。これは通常、ガベージ行も大量に検出します。このオプションは、切羽詰まった状況でなければ使用しないでください。

  このオプションの説明は、テーブルチェックオプションも参照してください。

  出力形式の説明は、セクション4.6.3.5「myisamchk によるテーブル情報の取得」を参照してください。

- --force, -f

  中止せず、古い中間ファイル (tbl_name.TMD のような名前のファイル) を上書きします。

- --keys-used=val, -k val

  myisamchk では、オプション値はどのインデックスを更新するかを示すビット値です。オプション値の各バイナリビットがテーブルインデックスに対応します。最初のインデックスはビット 0 です。オプション値が 0 の場合、すべてのインデックスの更新が無効になります。より高速な挿入を行うために使用できます。無効化されたインデックスは myisamchk -r を使用して再度有効化できます。

- --no-symlinks, -l

  シンボリックリンクをたどりません。通常、myisamchk はシンボリックリンクが示すテーブルを修復します。MySQL 4.0 以降のバージョンでは修復操作中にシンボリックリンクを削除しないため、このオプションは 4.0 には存在しません。

- --max-record-length=len

  myisamchk が指定された長さより大きい行を保存するためにメモリーを割り当てられない場合、行をスキップします。

- --parallel-recover, -p

  -r および -n と同じテクニックを使用しますが、異なるスレッドを使用してすべてのキーを並行して作成します。これはベータ品質のコードです。自己責任でご使用ください。

- --quick, -q

  データファイルではなくインデックスファイルのみを変更することで、より高速な修復を実現します。このオプションを 2 回指定することで、重複キーの場合に myisamchk を使用して強制的に元のデータファイルを変更させることができます。

- --recover, -r

  一意ではない一意なキー以外のすべてを修復できる修復を実行します (これは MyISAM テーブルではきわめてまれなエラーです)。テーブルをリカバリするには、このオプションをまず試してください。--safe-recover は、myisamchk が、--recover を使用してテーブルをリカバリできないとレポートする場合のみ使用するようにしてください。(非常にまれではありますが、--recover が失敗した場合、データファイルはそのままです。)

  メモリー容量に余裕がある場合、myisam_sort_buffer_size の値を増やすようにしてください。

- --safe-recover, -o

  すべての行を順に読み取り、検出された行に基づいてすべてのインデックスツリーを更新する古いリカバリ方法を使用して修復します。この方法は --recover よりも格段に遅くなりますが、--recover では対応できないいくつかのまれなケースに対応できます。また、このリカバリ方法は --recover よりもはるかに少ないディスクスペースを使用します。通常、まず --recover を使用して修復し、--recover が失敗した場合にかぎって --safe-recover を使用するようにしてください。

  メモリー容量に余裕がある場合、key_buffer_size の値を増やすとよいでしょう。

- --set-character-set=name

テーブルインデックスで使用する文字セットを変更します。このオプションは、MySQL 5.0.3 で --set-collation に置換されました。

- --set-collation=name

  テーブルインデックスのソートに使用する照合順序を指定します。照合順序名の初めの部位が文字セット名を示しています。

- --sort-recover, -n

  一時ファイルのサイズが非常に大きくなっても、キーの解決にソートを使用することを myisamchk に強制します。

- --tmpdir=path, -t path

  一時ファイルの保存に使用するディレクトリのパス。設定しない場合、myisamchk は TMPDIR 環境変数の値を使用します。--tmpdir に、一時ファイルの作成にラウンドロビン方式で順に使用する、ディレクトリパスのリストを設定できます。ディレクトリ名の間の区切り文字は、Unix ではコロン (「:」)、Windows ではセミコロン (「;」) です。

- --unpack, -u

  myisampack でパックされたテーブルをアンパックします。

4.6.3.4 その他の myisamchk オプション

myisamchk はテーブルチェックおよび修復以外のアクションのために、次のオプションをサポートします。

- --analyze, -a

  キー値の分布を分析します。これは、結合オプティマイザが、テーブルを結合する順番と、それが使用するインデックスをより適切に選択できるようにすることで、結合パフォーマンスを向上させます。キー分布に関する情報を取得するには、myisamchk --description --verbose tbl_name コマンドまたは SHOW INDEX FROM tbl_name ステートメントを使用します。

- --block-search=offset, -b offset

  指定されたオフセットのブロックが属するレコードを検索します。

- --description, -d

  テーブルの説明情報を出力します。--verbose オプションを一回または二回使用すると、追加情報が生成されます。セクション4.6.3.5「myisamchk によるテーブル情報の取得」を参照してください。

- --set-auto-increment[=value], -A[value]

  新しい行に対する AUTO_INCREMENT の番号付けを、指定した値 (または、AUTO_INCREMENT 値がこの値と同じであるレコードが存在する場合は、それより大きい値) で開始するように強制します。value が指定されていない場合は、新しいレコードの AUTO_INCREMENT の数字は現在テーブル内にあるもっとも大きい値 +1 で開始します。

- --sort-index, -S

  インデックスツリーブロックを降順でソートします。これはシークを最適化し、インデックスを使用するテーブルスキャンを高速化します。

- --sort-records=N, -R N

  特定のインデックスに基づいてレコードをソートします。これにより、データが大幅に局所に集中化されるため、このインデックスを使用する、範囲に基づいた SELECT または ORDER BY 操作が高速化する可能性があります。(テーブルのソートにはじめてこのオプションを使用する場合、かなり遅い場合があります。)テーブルのインデックス番号を決定するには、myisamchk が認識するのと同じ順序でテーブルのインデックスを表示する SHOW INDEX を使用してください。インデックスの番号は 1 から始まります。

  キーがパックされていない場合 (PACK_KEYS=0)、長さが同じであるため、myisamchk がレコードをソートして移動する際、インデックスのレコードオフセットを上書きするだけです。キーがパックされている場合 (PACK_KEYS=1)、myisamchk はまずキーブロックをアンパックし、次にインデックスを再作成してキーブロックをパックする必要があります。(この場合、各インデックスのオフセットを更新するよりも、インデックスを再作成する方が早くなります。)

4.6.3.5 myisamchk によるテーブル情報の取得

MyISAM テーブル情報またはそれに関する統計を取得するには、次に示すコマンドを使用します。これらのコマンドの出力は、このセクションのあとの方で説明します。

- myisamchk -d tbl_name

  myisamchk を 「describe モード」 で実行し、テーブル情報を生成します。外部ロックが無効になっている MySQL サーバーを起動した場合には、myisamchk は、実行中に更新があったテーブルに対してエラーをレポートすることがあります。ただし、describe モードでは myisamchk はテーブルを変更しないため、データが破損される危険はありません。

- myisamchk -dv tbl_name

  -v を追加すると、myisamchk は詳細モードで実行され、テーブルについてより多くの情報を生成します。-v をもう一度使用するとさらに多くの情報が生成されます。

- myisamchk -eis tbl_name

  テーブルからのもっとも重要な情報のみを表示します。この操作はテーブル全体を読み取る必要があるため、時間がかかります。

- myisamchk -eiv tbl_name

  これは -eis と同様ですが、進行中の処理が表示されます。

tbl_name 引数は、セクション4.6.3「myisamchk — MyISAM テーブルメンテナンスユーティリティー」で説明するように、MyISAM テーブルの名前か、またはそのインデックスファイル名のいずれかです。複数の tbl_name 引数を指定できます。

person という名前のテーブルの構造が、次のようになっているとします。(あとに示す myisamchk からの出力例で、一部の値がより小さく、出力形式に適合しやすいように、MAX_ROWS テーブルオプションが含まれます。)

```
CREATE TABLE person
(
  id         INT NOT NULL AUTO_INCREMENT,
  last_name  VARCHAR(20) NOT NULL,
  first_name VARCHAR(20) NOT NULL,
  birth      DATE,
  death      DATE,
  PRIMARY KEY (id),
  INDEX (last_name, first_name),
  INDEX (birth)
) MAX_ROWS = 1000000;
```

また、テーブルのデータファイルおよびインデックスファイルのサイズは次のようになっているとします。

```
-rw-rw----  1 mysql  mysql  9347072 Aug 19 11:47 person.MYD
-rw-rw----  1 mysql  mysql  6066176 Aug 19 11:47 person.MYI
```

myisamchk -dvv の出力例:

```
MyISAM file:         person
Record format:       Packed
Character set:       latin1_swedish_ci (8)
File-version:        1
Creation time:       2009-08-19 16:47:41
Recover time:        2009-08-19 16:47:56
Status:              checked,analyzed,optimized keys
Auto increment key:             1  Last value:              306688
Data records:              306688  Deleted blocks:               0
Datafile parts:            306688  Deleted data:                 0
Datafile pointer (bytes):       4  Keyfile pointer (bytes):      3
Datafile length:          9347072  Keyfile length:         6066176
Max datafile length:   4294967294  Max keyfile length:   17179868159
Recordlength:                  54

table description:
Key Start Len Index   Type              Rec/key        Root  Blocksize
1   2     4   unique  long              1       99328       1024
2   6     20  multip. varchar prefix          512     3563520       1024
    27    20          varchar             512
3   48    3   multip. uint24 NULL         306688      6065152       1024

Field Start Length Nullpos Nullbit Type
```

```
1   1   1
2   2   4                 no zeros
3   6   21                varchar
4   27  21                varchar
5   48  3   1   1         no zeros
6   51  3   1   2         no zeros
```

myisamchk が生成する情報のタイプについて次に説明します。「Keyfile」 とはインデックスファイルのことです。「レコード」と「行」、「フィールド」と 「カラム」は、それぞれシノニムです。

テーブル情報の最初の部分には次の値が含まれます。

- MyISAM file

  MyISAM (インデックス) ファイルの名前。

- Record format

  テーブルの行を保存するために使用される形式。前述の例では Fixed length を使用しています。ほかの値には、Compressed および Packed があります。(Packed は SHOW TABLE STATUS レポートで Dynamic とレポートされるものに対応します。)

- Chararacter set

  テーブルのデフォルト文字セット。

- File-version

  MyISAM 形式のバージョン。現在、常に 1 です。

- Creation time

  いつデータファイルが作成されたか。

- Recover time

  インデックスファイル/データファイルが最後にいつ再構成されたか。

- Status

  テーブルのステータスフラグ。可能な値は crashed、open、changed、analyzed、optimized keys、および sorted index pages です。

- Auto increment key、Last value

  テーブルの AUTO_INCREMENT カラムに関連付けられたキー番号、およびこのカラムに対して直近に生成された値。そのようなカラムがない場合はこのフィールドは表示されません。

- Data records

  テーブル内の行数。

- Deleted blocks

  削除されたブロックで、スペースがまだ予約されているものの数。テーブルを最適化してこのスペースを最小化できます。セクション7.6.4「MyISAM テーブルの最適化」を参照してください。

- Datafile parts

  動的行フォーマットでは、これはデータブロックの数を示します。断片化レコードがない最適化されたテーブルでは、これはデータレコードと同じです。

- Deleted data

  未使用の削除されたデータのバイト数。テーブルを最適化してこのスペースを最小化できます。セクション7.6.4「MyISAM テーブルの最適化」を参照してください。

- Datafile pointer

  データファイルポインタのサイズ (バイト単位)。通常は、2、3、4、または 5 バイトです。ほとんどのテーブルは 2 バイトで対応できますが、これはまだ MySQL では制御できません。固定テーブルでは、これは行のアドレスです。動的テーブルでは、これはバイトアドレスです。

- Keyfile pointer

  インデックスファイルポインタのサイズ (バイト単位)。通常は、1、2、または 3 バイトです。ほとんどのテーブルは 2 バイトで対応できますが、これは MySQL によって自動的に計算されます。常にブロックアドレスです。

- Max datafile length

  テーブルデータファイルがどこまで長くなれるか (バイト単位)。

- Max keyfile length

  テーブルインデックスファイルがどこまで長くなれるか (バイト単位)。

- Recordlength

  各行が使用するスペース (バイト単位)。

出力の table description の部分にはテーブルのすべてのキーのリストが含まれます。myisamchk は、それぞれのキーについて低レベル情報を表示します。

- Key

  このキーの番号。この値は、キーの最初のカラムに関してのみ表示されます。この値が欠落している場合、行は複数カラムキーの 2 番目以降のカラムに対応します。例に示したテーブルでは、2 番目のインデックスについて 2 つの table description 行があります。これは、2 つの部分で構成されるマルチパートインデックスであることを示しています。

- Start

  インデックスのこの部分が行の中のどこで始まるか。

- Len

  インデックスのこの部分の長さ。パックされた番号では、これは常にカラム全体の長さであるはずです。文字列では、文字列カラムのプリフィクスにインデックスを付けることができるため、インデックスされるカラム全体の長さより短い場合があります。マルチパートキーの合計の長さは、すべてのキーパートの Len 値の合計です。

- Index

  キー値がインデックス内に複数回存在できるかどうか。可能な値は unique または multip (multiple) です。

- Type

  インデックスのこの部分のデータ型。packed、stripped、または empty のいずれかの値の MyISAM データ型です。

- Root

  ルートインデックスブロックのアドレス。

- Blocksize

  各インデックスブロックのサイズ。デフォルトでは 1024 ですが、この値は MySQL がソースからビルドされる場合はコンパイル時に変更できます。

- Rec/key

  これはオプティマイザによって使用される統計値です。このインデックスで、値当たりにいくつの行があるかを示します。一意のインデックスでは値は常に 1 です。これは、テーブルのロード (または大きな変更) のあとに myisamchk -a で更新される場合があります。まったく更新されない場合はデフォルト値の 30 が指定されます。

出力の最後の部分は、各カラムの情報を示します。

- Field

  カラム番号。

- Start

テーブルの行の中でのカラムのバイト位置。

- Length

  カラムの長さ (バイト単位)。

- Nullpos、Nullbit

  NULL を取れるカラムでは、MyISAM は NULL 値をバイト内のフラグとして保存します。Null にできるカラムがいくつあるかによって、このために使用されるバイトが 1 またはそれ以上ある場合があります。Nullpos 値および Nullbit 値が空でない場合は、カラムが NULL かどうかを示すフラグが、どのバイトおよびビットに含まれるかを示します。

  NULL フラグを保存するために使用される位置とバイト数は、フィールド 1 の行に示されます。person テーブルには 5 つのカラムしかないのに Field 行が 6 個あるのはこのためです。

- Type

  データ型。この値は次のいずれかのディスクリプタを含むことがあります。

  - constant

    すべての行は同じ値を持っています。

  - no endspace

    エンドスペースを保存しません。

  - no endspace, not_always

    エンドスペースを保存せず、またすべての値にエンドスペース圧縮を行いません。

  - no endspace, no empty

    エンドスペースを保存しません。空の値を保存しません。

  - table-lookup

    カラムは ENUM に変換されました。

  - zerofill(N)

    値の中の最上位の N バイトは常に 0 であり、保存されません。

  - no zeros

    ゼロを保存しません。

  - always zero

    ゼロ値は 1 ビットを使用して保存されます。

- Huff tree

  カラムに関連しているハフマンツリーの数。

- Bits

  ハフマンツリーで使用されているビット数。

Huff tree フィールドおよび Bits フィールドは、テーブルが myisampack で圧縮されている場合に表示されます。この情報の例は、セクション4.6.5「myisampack — 圧縮された読み取り専用の MyISAM テーブルの生成」を参照してください。

myisamchk -eiv の出力例

```
Checking MyISAM file: person
Data records:  306688   Deleted blocks:       0
- check file-size
- check record delete-chain
```

```
No recordlinks
- check key delete-chain
block_size 1024:
- check index reference
- check data record references index: 1
Key:  1:  Keyblocks used:  98%  Packed:    0%  Max levels:  3
- check data record references index: 2
Key:  2:  Keyblocks used:  99%  Packed:   97%  Max levels:  3
- check data record references index: 3
Key:  3:  Keyblocks used:  98%  Packed:  -14%  Max levels:  3
Total:    Keyblocks used:  98%  Packed:   89%

- check records and index references
*** LOTS OF ROW NUMBERS DELETED ***

Records:            306688  M.recordlength:       25  Packed:           83%
Recordspace used:       97% Empty space:          2% Blocks/Record:   1.00
Record blocks:      306688  Delete blocks:         0
Record data:       7934464  Deleted data:          0
Lost space:         256512  Linkdata:        1156096

User time 43.08, System time 1.68
Maximum resident set size 0, Integral resident set size 0
Non-physical pagefaults 0, Physical pagefaults 0, Swaps 0
Blocks in 0 out 7, Messages in 0 out 0, Signals 0
Voluntary context switches 0, Involuntary context switches 0
Maximum memory usage: 1046926 bytes (1023k)
```

myisamchk -eiv の出力には、次の情報が含まれます。

- Data records

  テーブル内の行数。

- Deleted blocks

  削除されたブロックで、スペースがまだ予約されているものの数。テーブルを最適化してこのスペースを最小化できます。セクション7.6.4「MyISAM テーブルの最適化」を参照してください。

- Key

  キー番号。

- Keyblocks used

  キーブロックの何パーセントが使用されているか。テーブルが myisamchk で再構成されたばかりの場合は、値が非常に高く (理論上の最大値に非常に近く) なります。

- Packed

  MySQL は、共通のサフィクスを持つキー値のパックを試行します。これは、CHAR カラムおよび VARCHAR カラムのインデックスにのみ使用できます。左端に同様の部分を持つインデックス付きの長い文字列では、使用されるスペースをこれによって大幅に削減できる場合があります。前記の例では 2 番目のキーは長さが 40 バイトで、97% のスペース削減が実現されています。

- Max levels

  このキーの B ツリーの深さ。キー値の長い大規模なテーブルでは値が大きくなります。

- Records

  テーブル内の行数。

- M.recordlength

  平均行長。固定長の行を持つテーブルではすべての行が同じ長さであるため、これは正確な行の長さです。

- Packed

  MySQL は文字列の最後からスペースを削除します。Packed 値は、これを行うことによって達成された削減のパーセンテージを示します。

- Recordspace used

  データファイルの何パーセントが使用されているか。

- Empty space

  データファイルの何パーセントが未使用か。

- Blocks/Record

  行当たりの平均ブロック数 (すなわち、断片化された行がいくつのリンクで構成されるか)。固定形式のテーブルではこの値は常に 1.0 です。この値はできるだけ 1.0 に近くなるようにしてください。大きくなりすぎた場合は、テーブルを再構成できます。セクション7.6.4「MyISAM テーブルの最適化」を参照してください。

- Recordblocks

  使用されているブロック (リンク) の数。固定形式のテーブルでは、これは行数と同じです。

- Deleteblocks

  削除されたブロック (リンク) の数。

- Recorddata

  データファイル内の使用されているバイト数。

- Deleted data

  データファイル内の削除された (未使用の) バイト数。

- Lost space

  行がより短い長さに更新されると、一部のスペースが失われます。これは、そのような損失の合計 (バイト単位) です。

- Linkdata

  動的テーブル形式が使用される場合、行の断片はポインタ (それぞれ 4 から 7 バイト) でリンクされます。Linkdata は、このようなポインタすべてが使用しているストレージ量の合計です。

4.6.3.6 myisamchk メモリー使用量

myisamchk を実行する際、メモリー割り当ては重要です。myisamchk はメモリー関係の変数の設定を超えてメモリーを使用することはありません。myisamchk を非常に大きなテーブルで使用する場合、まずどのくらいのメモリーを使用するか決定する必要があります。デフォルトでは、修復に 3M バイト程度しか使用しないように設定されています。より大きな値を使用することで、myisamchk の動作速度を上げることができます。たとえば、512M バイトを超える RAM が使用可能な場合は、(ほかに指定するオプションに加えて) 次のようなオプションを使用できます。

```
shell> myisamchk --myisam_sort_buffer_size=256M \
         --key_buffer_size=512M \
         --read_buffer_size=64M \
         --write_buffer_size=64M ...
```

おそらくほとんどの場合には --myisam_sort_buffer_size=16M を使用すれば十分です。

myisamchk は TMPDIR 内の一時ファイルを使用することに注意してください。TMPDIR がメモリーファイルシステムを指している場合、メモリー不足エラーが容易におきる可能性があります。この場合は、--tmpdir=path オプションを指定してより多くのスペースを持つファイルシステムにあるディレクトリを指定して、myisamchk を実行します。

修復操作を実行する場合、myisamchk はディスクスペースも大量に必要とします。

- データファイルのサイズの 2 倍 (元のファイルとコピー)。--quick で修復を行う場合、スペースは必要ありません。この場合、再作成されるのはインデックスファイルのみです。コピーはオリジナルと同じディレクトリ内に作成されるため、このスペースはオリジナルのデータファイルと同じファイルシステム上で使用可能でなければなりません。

- 古いインデックスファイルを置換する新しいもの用のスペース。古いインデックスファイルは修復操作の最初に切り捨てられるため、通常このスペースは無視します。このスペースは、オリジナルのデータファイルと同じファイルシステム上で使用可能でなければなりません。

- --recover または --sort-recover を使用する場合 (ただし --safe-recover を使用する場合を除く)、ソートのためのスペースがディスク上に必要です。このスペースは一時ディレクトリ (TMPDIR または --tmpdir=path で指定されます) 上に割り当てられます。次の式は必要なスペースの量を求めます。

```
(largest_key + row_pointer_length) * number_of_rows * 2
```

キー長および row_pointer_length は myisamchk -dv tbl_name で確認できます (セクション4.6.3.5「myisamchk によるテーブル情報の取得」を参照してください)。row_pointer_length 値および number_of_rows 値は、テーブル情報内の Datafile pointer 値および Data records 値です。largest_key 値を判断するには、テーブル情報内の Key 行を確認します。Len カラムは各キーパートのバイト数を示します。マルチカラムインデックスでは、キーサイズはすべてのキーパートの Len 値の合計です。

修復中にディスクスペースの問題がある場合は、--recover の代わりに --safe-recover を使用してみてください。

4.6.4 myisamlog — MyISAM ログファイルの内容の表示

myisamlog は MyISAM ログファイルの内容を処理します。このようなファイルを作成するには、サーバーを --log-isam=log_file オプションで起動します。

myisamlog は次のように起動します。

```
shell> myisamlog [options] [file_name [tbl_name] ...]
```

デフォルトの操作は更新 (-u) です。リカバリが実行された場合 (-r)、すべての書き込み、および更新や削除が実行され、エラーは数えられるだけです。log_file 引数が指定されない場合は、デフォルトのログファイル名は myisam.log です。コマンド行でテーブルが指名された場合は、そのテーブルのみが更新されます。

myisamlog は次のオプションをサポートします。

- -?、-I

  ヘルプメッセージを表示して終了します。

- -c N

  N 個のコマンドのみ実行します。

- -f N

  開かれているファイルの最大数を指定します。

- -i

  終了する前に追加情報を表示します。

- -o offset

  開始オフセットを指定します。

- -p N

  パスから N 個のコンポーネントを取り除きます。

- -r

  リカバリ操作を実行します。

- -R record_pos_file record_pos

  レコード位置ファイルとレコード位置を指定します。

- -u

  更新操作を実行します。

- -v

  冗長モード。プログラムの動作についてより多くの情報を出力します。このオプションを複数回指定して、さらに多くの出力を生成できます。

- -w write_file

  書き込みファイルを指定します。

- -V

  バージョン情報を表示します。

4.6.5 myisampack — 圧縮された読み取り専用の MyISAM テーブルの生成

myisampack ユーティリティーは MyISAM テーブルを圧縮します。myisampack は、テーブルの各カラムを独立して圧縮することによって機能します。通常、myisampack はデータファイルを 40% から 70% パックします。

テーブルがあとで使用される場合、カラムの解凍に必要な情報をサーバーがメモリー内に読み取ります。これにより、個々の行をアクセスする際のパフォーマンスが大幅に向上します。これは、圧縮解除しなければならないのは 1 つの行のみであるためです。

MySQL は、圧縮されたテーブルでメモリーのマッピングを行う場合に、可能であれば mmap() を使用します。mmap() が機能しない場合、MySQL は普通のファイル読み取り/書き込み操作に戻ります。

次の点に注意してください。

- mysqld サーバーが外部ロックが無効化された状態で起動された場合、パック処理の最中にサーバーによってテーブルが更新される可能性がある場合は、myisampack の起動は推奨されません。サーバーが停止している状態でテーブルを圧縮するのがもっとも安全です。
- テーブルは、パック後に読み取り専用になります。通常これは意図されたものです (CD 内のパックされたテーブルにアクセスする場合など)。
- myisampack はパーティション化されたテーブルをサポートしません。

myisampack は次のように起動します。

```
shell> myisampack [options] file_name ...
```

各ファイル名引数はインデックス (.MYI) ファイルの名前にしてください。データベースディレクトリ内にいない場合、ファイルへのパスを指定するようにしてください。.MYI 拡張子は省略可能です。

myisampack でテーブルを圧縮したら、myisamchk -rq を使用してインデックスを再構築するようにしてください。セクション4.6.3「myisamchk — MyISAM テーブルメンテナンスユーティリティー」。

myisampack は次のオプションをサポートします。また、オプションファイルを読み取り、セクション4.2.7「オプションファイルの処理に影響するコマンド行オプション」に説明されている、それらを処理するためのオプションもサポートします。

- --help, -?

  ヘルプメッセージを表示して終了します。

- --backup, -b

  tbl_name.OLD という名前を使用して、各テーブルのデータファイルのバックアップを作成します。

- --character-sets-dir=path

  文字セットがインストールされているディレクトリ。セクション10.5「文字セットの構成」を参照してください。

- --debug[=debug_options], -# [debug_options]

  デバッグのログを書き込みます。一般的な debug_options 文字列は d:t:o,file_name です。デフォルトは d:t:o です。

- --force, -f

  パックされたテーブルが元のテーブルより大きくなる場合や、以前に myisampack を呼び出した際の中間ファイルが存在する場合でも、パックされたテーブルを生成します (myisampack はテーブルの圧縮中に、tbl_name.TMD という名前の中間ファイルをデータベースディレクトリに作成します。myisampack を強制終了した場合、.TMD ファイルは削除されないことがあります。)通常、myisampack は tbl_name.TMD が存在することを検出すると、エラーで終了します。--force を使用すると、myisampack は必ずテーブルをパックします。

- --join=big_tbl_name, -j big_tbl_name

コマンド行で指名されたすべてのテーブルを、単一のパックされたテーブル big_tbl_name に結合します。結合されるすべてのテーブルは、必ずまったく同一の構造でなければなりません (同一カラム名、型、同一インデックスなど)。

結合操作の前に、big_tbl_name が存在していなければなりません。コマンド行で指名され big_tbl_name にマージされるすべてのソーステーブルが存在していなければなりません。ソースのテーブルは結合のために読み取られますが、変更はされません。この結合操作では、big_tbl_name の .frm ファイルは作成されないので、結合操作の完了後、.frm ファイルをソーステーブルの 1 つからコピーして big_tbl_name.frm と名付けます。

- --silent, -s

  サイレントモード。エラーが発生したときのみ出力を書き出します。

- --test, -t

  実際にテーブルをパックせず、パックのテストのみを実行します。

- --tmpdir=path, -T path

  指名されたディレクトリを、myisampack が一時ファイルを作成する場所として使用します。

- --verbose, -v

  冗長モード。パック操作の進行状況とその結果に関する情報を書き込みます。

- --version, -V

  バージョン情報を表示して終了します。

- --wait, -w

  テーブルが使用中の場合、待機してから再度試みます。mysqld サーバーが外部ロックが無効化された状態で起動された場合、パック処理の最中にサーバーによってテーブルが更新される可能性がある場合は、myisampack の起動は推奨されません。

次のコマンドシーケンスは典型的なテーブル圧縮セッションを表しています。

```
shell> ls -l station.*
-rw-rw-r--   1 monty    my        994128 Apr 17 19:00 station.MYD
-rw-rw-r--   1 monty    my         53248 Apr 17 19:00 station.MYI
-rw-rw-r--   1 monty    my          5767 Apr 17 19:00 station.frm

shell> myisamchk -dvv station

MyISAM file:     station
Isam-version:  2
Creation time: 1996-03-13 10:08:58
Recover time:  1997-02-02  3:06:43
Data records:             1192  Deleted blocks:            0
Datafile parts:           1192  Deleted data:              0
Datafile pointer (bytes):    2  Keyfile pointer (bytes):    2
Max datafile length:   54657023  Max keyfile length:   33554431
Recordlength:              834
Record format: Fixed length

table description:
Key Start Len Index   Type                 Root  Blocksize    Rec/key
1   2     4   unique  unsigned long        1024       1024          1
2   32    30  multip. text               10240       1024          1

Field Start Length Type
1     1     1
2     2     4
3     6     4
4     10    1
5     11    20
6     31    1
7     32    30
8     62    35
9     97    35
10    132   35
11    167   4
12    171   16
13    187   35
```

```
14   222  4
15   226  16
16   242  20
17   262  20
18   282  20
19   302  30
20   332  4
21   336  4
22   340  1
23   341  8
24   349  8
25   357  8
26   365  2
27   367  2
28   369  4
29   373  4
30   377  1
31   378  2
32   380  8
33   388  4
34   392  4
35   396  4
36   400  4
37   404  1
38   405  4
39   409  4
40   413  4
41   417  4
42   421  4
43   425  4
44   429  20
45   449  30
46   479  1
47   480  1
48   481  79
49   560  79
50   639  79
51   718  79
52   797  8
53   805  1
54   806  1
55   807  20
56   827  4
57   831  4

shell> myisampack station.MYI
Compressing station.MYI: (1192 records)
- Calculating statistics

normal:    20  empty-space:   16  empty-zero:    12  empty-fill:  11
pre-space:   0  end-space:    12  table-lookups:   5  zero:        7
Original trees:  57  After join: 17
- Compressing file
87.14%
Remember to run myisamchk -rq on compressed tables

shell> ls -l station.*
-rw-rw-r--   1 monty    my        127874 Apr 17 19:00 station.MYD
-rw-rw-r--   1 monty    my         55296 Apr 17 19:04 station.MYI
-rw-rw-r--   1 monty    my          5767 Apr 17 19:00 station.frm

shell> myisamchk -dvv station

MyISAM file:     station
Isam-version:  2
Creation time: 1996-03-13 10:08:58
Recover time:  1997-04-17 19:04:26
Data records:              1192  Deleted blocks:            0
Datafile parts:            1192  Deleted data:              0
Datafile pointer (bytes):     3  Keyfile pointer (bytes):    1
Max datafile length:    16777215  Max keyfile length:    131071
Recordlength:               834
Record format: Compressed

table description:
Key Start Len Index   Type              Root  Blocksize    Rec/key
1   2     4   unique  unsigned long     10240       1024          1
2   32    30  multip. text              54272       1024          1

Field Start Length Type                     Huff tree  Bits
```

```
1   1   1    constant                        1   0
2   2   4    zerofill(1)                     2   9
3   6   4    no zeros, zerofill(1)           2   9
4   10  1                                    3   9
5   11  20   table-lookup                     4   0
6   31  1                                    3   9
7   32  30   no endspace, not_always          5   9
8   62  35   no endspace, not_always, no empty   6   9
9   97  35   no empty                         7   9
10  132 35   no endspace, not_always, no empty   6   9
11  167 4    zerofill(1)                     2   9
12  171 16   no endspace, not_always, no empty   5   9
13  187 35   no endspace, not_always, no empty   6   9
14  222 4    zerofill(1)                     2   9
15  226 16   no endspace, not_always, no empty   5   9
16  242 20   no endspace, not_always         8   9
17  262 20   no endspace, no empty           8   9
18  282 20   no endspace, no empty           5   9
19  302 30   no endspace, no empty           6   9
20  332 4    always zero                     2   9
21  336 4    always zero                     2   9
22  340 1                                    3   9
23  341 8    table-lookup                    9   0
24  349 8    table-lookup                    10  0
25  357 8    always zero                     2   9
26  365 2                                    2   9
27  367 2    no zeros, zerofill(1)           2   9
28  369 4    no zeros, zerofill(1)           2   9
29  373 4    table-lookup                    11  0
30  377 1                                    3   9
31  378 2    no zeros, zerofill(1)           2   9
32  380 8    no zeros                        2   9
33  388 4    always zero                     2   9
34  392 4    table-lookup                    12  0
35  396 4    no zeros, zerofill(1)           13  9
36  400 4    no zeros, zerofill(1)           2   9
37  404 1                                    2   9
38  405 4    no zeros                        2   9
39  409 4    always zero                     2   9
40  413 4    no zeros                        2   9
41  417 4    always zero                     2   9
42  421 4    no zeros                        2   9
43  425 4    always zero                     2   9
44  429 20   no empty                        3   9
45  449 30   no empty                        3   9
46  479 1                                    14  4
47  480 1                                    14  4
48  481 79   no endspace, no empty           15  9
49  560 79   no empty                        2   9
50  639 79   no empty                        2   9
51  718 79   no endspace                     16  9
52  797 8    no empty                        2   9
53  805 1                                    17  1
54  806 1                                    3   9
55  807 20   no empty                        3   9
56  827 4    no zeros, zerofill(2)           2   9
57  831 4    no zeros, zerofill(1)           2   9
```

myisampack は次の種類の情報を表示します。

- normal

  追加のパックが使用されていないカラムの数。

- empty-space

  スペースのみの値を含むカラムの数。これらは 1 ビットを占めます。

- empty-zero

  バイナリのゼロのみの値を含むカラムの数。これらは 1 ビットを占めます。

- empty-fill

  それぞれの型のバイト範囲をすべて占領しない整数カラムの数。これらはより小さい型に変更されます。たとえば、BIGINT カラム (8 バイト) のすべての値が -128 から 127 の範囲内にある場合は、このカラムを TINYINT カラム (1 バイト) として格納できます。

- pre-space

  先頭にスペースが付いて保存されている 10 進数カラムの数。この場合、各値は先頭のスペースの数のカウントを含んでいます。

- end-space

  後続のスペースを多く含むカラムの数。この場合、各値は後続のスペースの数のカウントを含んでいます。

- table-lookup

  カラムには少数の異なる値のみがあり、ハフマン圧縮の前に ENUM に変換されました。

- zero

  すべての値がゼロのカラムの数。

- Original trees

  もともとのハフマンツリーの数です。

- After join

  ヘッダースペースを節約するため、ツリーの結合のあとに残った異なるハフマンツリーの数。

テーブルの圧縮後、myisamchk -dvv が表示する Field 行には、各カラムに関する追加情報が含まれます。

- Type

  データ型。この値は次のいずれかのディスクリプタを含むことがあります。

  - constant

    すべての行は同じ値を持っています。

  - no endspace

    エンドスペースを保存しません。

  - no endspace, not_always

    エンドスペースを保存せず、またすべての値にエンドスペース圧縮を行いません。

  - no endspace, no empty

    エンドスペースを保存しません。空の値を保存しません。

  - table-lookup

    カラムは ENUM に変換されました。

  - zerofill(N)

    値の中の最上位の N バイトは常に 0 であり、保存されません。

  - no zeros

    ゼロを保存しません。

  - always zero

    ゼロ値は 1 ビットを使用して保存されます。

- Huff tree

  カラムに関連しているハフマンツリーの数。

- Bits

  ハフマンツリーで使用されているビット数。

myisampack の起動後、インデックスを再作成するにはmyisamchk を起動しなければいけません。このとき、MySQL オプティマイザの動作効率化を図るために、インデックスブロックのソートと統計の作成を行うことができます。

```
shell> myisamchk -rq --sort-index --analyze tbl_name.MYI
```

パックされたテーブルを MySQL データベースディレクトリにインストールしたあと、mysqld が新しいテーブルを使用することを強制するため、mysqladmin flush-tables を実行するようにしてください。

パックされたテーブルをアンパックするには、myisamchk に対して --unpack オプションを使用してください。

4.6.6 mysql_config_editor — MySQL 構成ユーティリティー

mysql_config_editor ユーティリティー (MySQL 5.6.6 で使用可能です) を使用すると、.mylogin.cnf という名前の暗号化されたログインファイルに認証情報を保存できます。ファイルの場所は、Windows では %APPDATA%\MySQL ディレクトリ、非 Windows システムでは現在のユーザーのホームディレクトリです。このファイルは、MySQL Server に接続するための認証情報を取得するために、MySQL クライアントプログラムによってあとで読み取ることができます。

別のファイル名を指定するには、MYSQL_TEST_LOGIN_FILE 環境変数を設定します。この変数は mysql-test-run.pl テストユーティリティーが使用しますが、mysql_config_editor および mysql、mysqladmin、などの MySQL クライアントによっても認識されます。

mysql_config_editor は .mylogin.cnf ファイルを暗号化するため平文として読み取ることはできず、クライアントプログラムによって復号化された場合、その内容はメモリー内でのみ使用されます。これにより、パスワードを平文ではない形式のファイルに保存して、コマンド行または環境変数にさらされる必要もまったくなく、あとで使用できます。mysql_config_editor は、ユーザーがファイルの内容を表示できるようにする print コマンドを提供しますが、この場合でもパスワード値はマスクされるため、ほかのユーザーが見られるような方法では決して表示されません。

mysql_config_editor が使用する暗号化は、パスワードが .mylogin.cnf に平文として現れることを防ぎ、パスワードが意図せずにさらされることを防ぐセキュリティー手段を提供します。たとえば、通常の暗号化されていない my.cnf オプションファイルを画面に表示すると、そこに含まれるパスワードはだれでも見ることができます。.mylogin.cnf では、そうではありません。しかし、使用される暗号化は、強力な攻撃者を阻止することはなく、解読されることがないとは考えないようにしてください。マシン上であなたのファイルにアクセスするためにシステム管理者権限を取得できるユーザーなら、少しの作業で .mylogin.cnf ファイルを復号化できます。

ログインファイルは現在のユーザーが読み取りおよび書き込み可能で、ほかのユーザーからはアクセス不能でなければなりません。そうでないと、mysql_config_editor はそれを無視し、ファイルはクライアントプログラムでも使用されません。Windows では、この制約は適用されません。代わりに、ユーザーは %APPDATA%\MySQL ディレクトリにアクセスできなければなりません。

暗号化されていない形式の .mylogin.cnf ログインファイルは、ほかのオプションファイルと同様にオプショングループで構成されます。.mylogin.cnf 内の各オプショングループは「ログインパス」と呼ばれ、host、user、および password という限定されたオプションのセットのみを許可するグループです。ログインパスは、サーバーホストおよびそのサーバーで認証するための認証情報を示す値のセットであると考えてください。次に例を示します。

```
[myloginpath]
user = myname
password = mypass
host = 127.0.0.1
```

サーバーに接続するためにクライアントプログラムを起動すると、ほかのオプションファイルとともに .mylogin.cnf が使用されます。その優先順位はほかのオプションファイルより高くなりますが、クライアントのコマンド行で明示的に指定されたオプションよりは低くなります。オプションファイルが使用される順序の詳細は、セクション4.2.6「オプションファイルの使用」を参照してください。

mysql_config_editor は次のように起動します。

```
shell> mysql_config_editor [program_options] command [command_options]
```

program_options は一般的な mysql_config_editor オプションで構成されます。command は実行するコマンドを示し、command_options はコマンドが必要とする追加のオプションを示します。

コマンドは、.mylogin.cnf login ファイルに対して実行するアクションを示します。たとえば、set はログインパスをファイルに書き出し、remove はログインパスを削除し、print はログインパスの内容を表示します。オプショ

ンが指定された場合は、ログインパス名およびログインパスで使用する値など、コマンドに対して情報を提供します。

プログラム引数のセット内でのコマンド名の位置は重要です。たとえば、次のコマンド行は同じ引数を持ちますが、結果は異なります。

```
mysql_config_editor --help set
mysql_config_editor set --help
```

最初のコマンド行は一般的な mysql_config_editor ヘルプを表示し、set コマンドは無視されます。2 番目のコマンド行は set コマンドのヘルプを表示します。

ローカル MySQL サーバーおよびホスト remote.example.com に接続するためにlocal および remote という名前の 2 つのログインパスを確立するとします。ローカルサーバーには localuser および localpass というユーザー名とパスワードで認証を行い、リモートサーバーには remoteuser および remotepass というユーザー名とパスワードで認証を実行します。.mylogin.cnf ファイルにログインパスを設定するには、次の set コマンドを使用します。コマンドは 1 行に 1 つずつ入力し、次に要求されたら適切なパスワードを入力します。

```
shell> mysql_config_editor set --login-path=local
       --host=localhost --user=localuser --password
Enter password: enter password "localpass" here
shell> mysql_config_editor set --login-path=remote
       --host=remote.example.com --user=remoteuser --password
Enter password: enter password "remotepass" here
```

mysql_config_editor が .mylogin.cnf ファイルに書き出す内容を表示するには、print コマンドを使用します。

```
shell> mysql_config_editor print --all
[local]
user = localuser
password = *****
host = localhost
[remote]
user = remoteuser
password = *****
host = remote.example.com
```

print コマンドは、各ログインパスを行のセットとして表示します。各セットには最初にグループヘッダー (角かっこ内にログインパス名を示します) があり、ログインパスのオプション値がそれに続きます。パスワード値はマスクされ、平文としては表示されません。

前の例に示されるように、.mylogin.cnf ファイルには複数のログインパスを含めることができます。これにより、mysql_config_editor で、異なる MySQL サーバーに接続するために複数の「人物」を設定するのが容易になります。これらはすべて、あとでクライアントプログラムを起動するときに --login-path オプションを使用して名前で選択できます。たとえば、ローカルサーバーに接続するには次のコマンドを使用します。

```
shell> mysql --login-path=local
```

リモートサーバーに接続するには次のコマンドを使用します。

```
shell> mysql --login-path=remote
```

set コマンドを mysql_config_editor で使用してログインパスを作成する場合、可能な 3 つのオプション値 (ホスト名、ユーザー名、およびパスワード) をすべて指定する必要はありません。指定した値のみがパスに書き出されます。欠落している値があとで必要になった場合は、クライアントパスを起動して MySQL サーバーに接続するときに、オプションファイルまたはコマンド行で指定できます。また、コマンド行で指定されたオプションは、.mylogin.cnf ファイルを含めたオプションファイルのオプションをオーバーライドします。たとえば、remote ログインパスの認証情報がホスト remote2.example.com にも適用される場合、そのホストのサーバーに次のように接続できます。

```
shell> mysql --login-path=remote --host=remote2.example.com
```

.mylogin.cnf ファイルが存在する場合は、--no-defaults オプションが使用された場合でも、必ず読み取られます。これにより、--no-defaults が存在する場合でも、コマンド行よりも安全な方法でパスワードを指定できます。

mysql_config_editor のコマンド

このセクションでは、許可される mysql_config_editor コマンド、およびコマンド固有の意味を持つオプションの解釈について説明します。また、mysql_config_editor は、mysql_config_editor の実行中により多くの情報を生成

するための --verbose のように、任意のコマンドに使用できるその他のオプションも取ります。このオプションは、操作が期待する効果を生じない場合に、問題を診断するのに役立ちます。サポートされるオプションのリストは、mysql_config_editor のオプションを参照してください。

mysql_config_editor は次のコマンドをサポートします。

- help

  ヘルプメッセージを表示して終了します。

- print [options]

  .mylogin.cnf の内容を暗号化されていない形式で出力します。パスワードは ***** として表示されます。

  print コマンドは次のオプションを取ります。

  - --all

    すべてのログインパスを出力します。

  - --login-path=name

    指名されたログインパスを出力します。

  ログインパスが指定されていない場合のデフォルトパス名は client です。--all および --login-path の両方が指定されている場合、--all が優先されます。

- remove [options]

  ログインパスを .mylogin.cnf ファイルから削除します。

  remove コマンドは次のオプションを取ります。

  - --host

    ホスト名をログインパスから削除します。

  - --login-path=name

    削除するログインパス。このオプションが指定されていない場合のデフォルトパス名は client です。

  - --password

    パスワードをログインパスから削除します。

  - --port

    TCP/IP ポート番号をログインパスから削除します。

  - --socket

    Unix ソケットファイル名をログインパスから削除します。

  - --user

    ユーザー名をログインパスから削除します。

  remove コマンドに対する --host、--user、および --password オプションは、MySQL 5.6.9 でサポートされます。remove コマンドに対する --port および --socket オプションは、MySQL 5.6.11 でサポートされます。

  remove コマンドは、--host、--password、--port、--socket、および --user の各オプションで指定された値のみをログインパスから削除します。これらがどれも指定されていない場合、remove はログインパス全体を削除します。たとえば次のコマンドは、client ログインパス全体ではなく、client ログインパスから user の値のみを削除します。

  ```
  mysql_config_editor remove --login-path=client --user
  ```

- reset

  .mylogin.cnf ファイルの内容を空にします。ファイルが存在しない場合は作成されます。

- set [options]

  ログインパスを .mylogin.cnf ファイルに書き出します。

  set コマンドは次のオプションを取ります。

  - --host=host_name

    ログインパスに書き出すホスト名。

  - --login-path=name

    作成するログインパス。このオプションが指定されていない場合のデフォルトパス名は client です。

  - --password

    ログインパスに書き出すパスワードを要求します。

  - --port=port_num

    ログインパスに書き出す TCP/IP ポート番号。

  - --socket=file_name

    ログインパスに書き出す Unix ソケットファイル名。

  - --user=user_name

    ログインパスに書き出すユーザー名。

  set コマンドに対する --port および --socket オプションは、MySQL 5.6.11 でサポートされます。

  set コマンドは、--host、--password、--port、--socket、および --user の各オプションで指定された値のみをログインパスに書き出します。これらのオプションがいずれも指定されない場合は、mysql_config_editor はログインパスを空のグループとして書き出します。

  空のパスワードを指定するには、set コマンドを --password オプションとともに使用し、次にパスワードプロンプトで Enter を押します。その結果 .mylogin.cnf に書き出されるログインパスは、次のような行を含みます。

```
password =
```

  .mylogin.cnf にログインパスがすでに存在する場合は、set コマンドはそれを置換します。これがユーザーが意図したものであることを確認するために、mysql_config_editor は警告を出力して確認を要求します。警告とプロンプトを抑制するには、--skip-warn オプションを使用します。

mysql_config_editor のオプション

mysql_config_editor は次のオプションをサポートします。

表 4.13 mysql_config_editor のオプション

オプション名	説明	導入
--all	すべてのログインパスを出力	
--debug	デバッグのログを書き込み	
--help	ヘルプメッセージを表示して終了	
--host	ログインファイルを書き出すホスト	
--login-path	ログインパス名	
--password	ログインファイルに書き出すパスワードを要求	
--port	ログインファイルに書き出す TCP/IP ポート番号	5.6.11
--socket	ログインファイルに書き出す Unix ソケットファイル名	5.6.11
--user	ログインファイルを書き出すユーザー名	
--verbose	冗長モード	
--version	バージョン情報を表示して終了	

オプション名	説明	導入
--warn	ログインパスの上書きについて警告し、確認を要求	

- --help, -?

  ヘルプメッセージを表示して終了します。set または remove などのコマンド名が前にある場合は、そのコマンドに関する情報を表示します。

- --all

  print コマンドに対して、ログインファイル内のすべてのログインパスを出力します。

- --debug[=debug_options], -# debug_options

  デバッグのログを書き込みます。一般的な debug_options 文字列は d:t:o,file_name です。デフォルトは d:t:o です。

- --host=host_name, -h host_name

  set コマンドに対しては、ログインパスに書き出すホスト名。remove コマンドに対しては、ログインパスからホスト名を削除します。

- --login-path=name, -G name

  print、remove、および set コマンドに対して、.mylogin.cnf ログインファイル内で使用するログインパス。

  ユーザーが MySQL サーバーへの接続に使用するログインパスを指定できるように、クライアントプログラムも --login-path オプションをサポートします。クライアントプログラムでは、--login-path は指定する最初のオプションでなければなりません。これは mysql_config_editor には該当しません。セクション4.2.7「オプションファイルの処理に影響するコマンド行オプション」を参照してください。

- --password, -p

  set コマンドに対して、mysql_config_editor がパスワードを要求し、ユーザーが入力した値をログインパスに書き出すようになります。mysql_config_editor が起動してプロンプトを表示したあと、ユーザーはパスワードを入力して Enter を押すようにしてください。ほかのユーザーがパスワードを見るのを防ぐため、mysql_config_editor はエコーしません。

  このオプションでは、オプション名のあとにパスワード値を指定することはできません。つまり mysql_config_editor では、ほかのユーザーに見られる可能性のあるコマンド行にパスワードを入力することは決してありません。これは、ほかのほとんどの MySQL プログラムとは異なります。ほかのプログラムでは --password=pass_val または -ppass_val としてコマンド行でパスワードを指定できます。(ただし、この方法はセキュアではなく、避けるべきです。)

  remove コマンドに対しては、ログインパスからパスワードを削除します。

- --port=port_num, -P port_num

  set コマンドに対しては、ログインパスに書き出す TCP/IP ポート番号。remove コマンドに対しては、ログインパスからポート番号を削除します。

- --socket=file_name, -S file_name

  set コマンドに対しては、ログインパスに書き出す Unix ソケットファイル名。remove コマンドに対しては、ログインパスからソケットファイルを削除します。

- --user=user_name, -u user_name

  set コマンドに対しては、ログインパスに書き出すユーザー名。remove コマンドに対しては、ログインパスからユーザー名を削除します。

- --verbose, -v

  冗長モード。プログラムの動作についてより多くの情報を出力します。

- --version, -V

  バージョン情報を表示して終了します。

- --warn, -w

  set コマンドに対して、コマンドが既存のログインパスを上書きしようとする場合に、ユーザーに警告して確認を求めます。このオプションはデフォルトで有効です。無効にするには、--skip-warn を使用します。

4.6.7 mysqlaccess — アクセス権限をチェックするクライアント

> **注記**
>
> このユーティリティーは MySQL 5.6.17 で非推奨で、MySQL 5.7 で削除されます。

mysqlaccess は MySQL 配布のために Yves Carlier が提供した診断ツールです。ホスト名、ユーザー名、およびデータベースの組み合わせに対してアクセス権限をチェックします。mysqlaccess は user テーブルおよび db テーブルのみを使用してアクセスをチェックすることに注意してください。tables_priv、columns_priv、および procs_priv の各テーブルで指定されるテーブル、カラム、そしてルーチンの権限はチェックしません。

mysqlaccess は次のように起動します。

```
shell> mysqlaccess [host_name [user_name [db_name]]] [options]
```

mysqlaccess は次のオプションをサポートします。

表 4.14 mysqlaccess のオプション

オプション名	説明
--brief	1 行の表形式でレポートを生成
--commit	一時テーブルから元の付与テーブルへ新しいアクセス権限をコピー
--copy	元の付与テーブルから一時付与テーブルをリロード
--db	データベース名を指定
--debug	デバッグレベルを指定
--help	ヘルプメッセージを表示して終了
--host	指定されたホスト上で MySQL サーバーに接続
--howto	mysqlaccess の使用方法を示す例を表示
--old_server	サーバーが古い MySQL サーバーであるとみなす (MySQL 3.21 より前)
--password	サーバーに接続する際に使用するパスワード
--plan	将来のリリースのためのアイデアや提案を表示
--preview	一時付与テーブルに変更を加えたあと、権限の差異を表示
--relnotes	リリースノートを表示
--rhost	指定されたホスト上で MySQL サーバーに接続
--rollback	一時付与テーブルへの直近の変更を取り消し。
--spassword	スーパーユーザーとしてサーバーに接続する際に使用するパスワード
--superuser	スーパーユーザーとして接続するためのユーザー名を指定
--table	表形式でレポートを生成
--user	サーバーへの接続時に使用する MySQL ユーザー名
--version	バージョン情報を表示して終了

- --help, -?

  ヘルプメッセージを表示して終了します。

- --brief, -b

  1 行の表形式でレポートを生成します。

- --commit

  一時テーブルから元の付与テーブルへ新しいアクセス権限をコピーします。新しい権限が有効になるには、付与テーブルをフラッシュする必要があります。(たとえば、mysqladmin reload コマンドを実行します。)

- --copy

元の付与テーブルから一時付与テーブルをリロードします。

- --db=db_name, -d db_name

  データベース名を指定します。

- --debug=N

  デバッグレベルを指定します。N は 0 から 3 までの整数です。

- --host=host_name, -h host_name

  アクセス権限で使用するホスト名。

- --howto

  mysqlaccess の使用方法を示す例を表示します。

- --old_server

  サーバーが、完全な WHERE 句の処理方法を理解していない古い (MySQL 3.21 より前の) MySQL サーバーであるとみなします。

- --password[=password], -p[password]

  サーバーに接続する際に使用するパスワードです。コマンド行で、--password オプションまたは -p オプションに続けて password の値を指定しなかった場合、mysqlaccess はそれを要求します。

  コマンド行でパスワードを指定することはセキュアでないとみなすべきです。セクション6.1.2.1「パスワードセキュリティーのためのエンドユーザーガイドライン」を参照してください。

- --plan

  将来のリリースのためのアイデアや提案を表示します。

- --preview

  一時付与テーブルに変更を加えたあと、権限の差異を表示します。

- --relnotes

  リリースノートを表示します。

- --rhost=host_name, -H host_name

  指定されたホストの MySQL サーバーに接続します。

- --rollback

  一時付与テーブルへの直近の変更を取り消し。

- --spassword[=password], -P[password]

  スーパーユーザーとしてサーバーに接続する際に使用するパスワード。コマンド行で、--spassword オプションまたは -p オプションに続けて password の値を指定しなかった場合、mysqlaccess はそれを要求します。

  コマンド行でパスワードを指定することはセキュアでないとみなすべきです。セクション6.1.2.1「パスワードセキュリティーのためのエンドユーザーガイドライン」を参照してください。

- --superuser=user_name, -U user_name

  スーパーユーザーとして接続するためのユーザー名を指定します。

- --table, -t

  表形式でレポートを生成します。

- --user=user_name, -u user_name

  アクセス権限で使用するユーザー名。

- --version, -v

  バージョン情報を表示して終了します。

MySQL 配布が標準以外の場所にインストールされている場合は、mysqlaccess が mysql クライアントを検出することを想定する場所を変更する必要があります。mysqlaccess スクリプトの約 18 行目を編集します。次のような行を検索します。

```
$MYSQL     = '/usr/local/bin/mysql';    # path to mysql executable
```

ユーザーのシステムで mysql が実際に保存されている場所を反映するように、パスを変更します。これを実行しないと、mysqlaccess を起動したときに Broken pipe エラーが発生します。

4.6.8 mysqlbinlog — バイナリログファイルを処理するためのユーティリティー

サーバーのバイナリログは、データベースの内容に対する変更を記述する「イベント」を含むファイルで構成されます。サーバーはこれらのファイルをバイナリ形式で書き出します。内容をテキスト形式で表示するには、mysqlbinlog ユーティリティーを使用します。レプリケーションセットアップで、mysqlbinlog を使用して、スレーブサーバーが書き出したリレーログファイルの内容を表示することもできます。リレーログはバイナリログと同じ形式を持つからです。バイナリログおよびリレーログは、セクション5.2.4「バイナリログ」およびセクション17.2.2「レプリケーションリレーおよびステータスログ」でさらに説明します。

mysqlbinlog は次のように起動します。

```
shell> mysqlbinlog [options] log_file ...
```

たとえば binlog.000003 という名前のバイナリログファイルの内容を表示するには、このコマンドを使用してください。

```
shell> mysqlbinlog binlog.0000003
```

出力には、binlog.000003 に含まれるイベントが含まれます。ステートメントベースのロギングでは、イベント情報には SQL ステートメント、それが実行されたサーバーの ID、ステートメントが実行されたタイムスタンプ、かかった時間などが含まれます。行ベースのロギングでは、イベントは SQL ステートメントではなく行の変更を示します。ロギングモードの詳細はセクション17.1.2「レプリケーション形式」を参照してください。

イベントには、その前に追加情報を提供するヘッダーコメントがあります。例:

```
# at 141
#100309  9:28:36 server id 123  end_log_pos 245
  Query thread_id=3350  exec_time=11  error_code=0
```

最初の行で、at に続く数字はバイナリログファイル内でのイベントのファイルオフセット、つまり開始位置を示します。

2 行目は、イベントが発生したサーバー上でステートメントがいつ開始されたかを示す日付と時間で始まります。レプリケーションに関しては、このタイムスタンプがスレーブサーバーに伝播されます。server id はイベントが発生したサーバーの server_id 値です。end_log_pos は次のイベントが始まる場所 (すなわち、現在のイベントの終了位置 + 1) を示します。thread_id はイベントを実行したスレッドを示します。exec_time は、マスターサーバーではイベントの実行にかかった時間です。スレーブでは、これはスレーブでの実行終了時間からマスターでの実行開始時間を引いた差分です。この差は、レプリケーションがマスターからどの程度遅れているかを示します。error_code はイベントの実行結果を示します。ゼロはエラーが発生しなかったことを意味します。

> **注記**
>
> イベントグループを使用する場合は、イベントのファイルオフセットのグループ化およびイベントのコメントのグループ化ができます。これらのグループ化イベントをブランクファイルオフセットと間違えないでください。

mysqlbinlog からの出力は、(たとえばそれを mysql への入力として使用することによって) 再実行し、ログ内のステートメントをやり直せます。これはサーバーがクラッシュした際のリカバリ操作として便利です。ほかの使用例は、このセクションのあとの方の説明およびセクション7.5「バイナリログを使用したポイントインタイム (増分) リカバリ」を参照してください。

通常、mysqlbinlog を使用してバイナリログファイルを直接読み取り、ローカル MySQL サーバーに適用します。--read-from-remote-server オプションを使用してリモートサーバーからバイナリログを読み取ることも可能です。リモートバイナリログを読み取るために、接続パラメータオプションを指定してサーバーへの接続方法を示すことができます。これらのオプションは、--host、--password、--port、--protocol, --socket、および --user です。--read-from-remote-server オプションも使用した場合以外は無視されます。

mysqlbinlog を大規模なバイナリログに対して実行する場合は、結果のファイルのためにファイルシステムに十分なスペースがあるように注意してください。mysqlbinlog が一時ファイル用に使用するディレクトリを構成するには、TMPDIR 環境変数を使用します。

mysqlbinlog は次のオプションをサポートします。これらはコマンド行またはオプションファイルの [mysqlbinlog] グループおよび [client] グループで指定できます。MySQL プログラムによって使用されるオプションファイルの詳細については、セクション4.2.6「オプションファイルの使用」を参照してください。

表 4.15 mysqlbinlog のオプション

オプション名	説明	導入
--base64-output	バイナリログのエントリを base-64 エンコードで出力	
--bind-address	指定されたネットワークインタフェースを使用して MySQL サーバーに接続	5.6.1
--binlog-row-event-max-size	バイナリログの最大イベントサイズ	
--character-sets-dir	文字セットがインストールされているディレクトリ	
--connection-server-id	テストとデバッグに使用。適用されるデフォルト値およびその他の事項については、テキストを参照してください。	5.6.20
--database	このデータベースのみのエントリをリスト	
--debug	デバッグのログを書き込み	
--debug-check	プログラムの終了時にデバッグ情報を出力	
--debug-info	プログラムの終了時に、デバッグ情報、メモリー、および CPU の統計を出力	
--default-auth	使用する認証プラグイン	5.6.2
--defaults-extra-file	通常のオプションファイルに加えてオプションファイルを読み取る	
--defaults-file	指名されたオプションファイルのみを読み取る	
--defaults-group-suffix	オプショングループのサフィクス値	
--disable-log-bin	バイナリロギングを無効化	
--exclude-gtids	提供された GTID セットのグループを表示しない	5.6.5
--force-if-open	バイナリログファイルが開いているか適切にクローズしていない場合でも読み取る	
--force-read	mysqlbinlog が認識しないバイナリログイベントを読み取った場合、警告を出力	
--help	ヘルプメッセージを表示して終了	
--hexdump	コメント内にログの 16 進ダンプを表示	
--host	指定されたホスト上で MySQL サーバーに接続	
--include-gtids	提供された GTID セットのグループのみを表示	5.6.5
--local-load	指定されたディレクトリ内に LOAD DATA INFILE のローカル一時ファイルを準備	
--login-path	ログインパスオプションを .mylogin.cnf から読み取り	5.6.6
--no-defaults	オプションファイルを読み取らない	
--offset	ログの最初の N エントリをスキップ	
--password	サーバーに接続する際に使用するパスワード	
--plugin-dir	プラグインがインストールされているディレクトリ	5.6.2
--port	接続に使用する TCP/IP ポート番号	
--print-defaults	デフォルトを出力	
--protocol	使用する接続プロトコル	
--raw	イベントを生の (バイナリ) 形式で出力ファイルに書き込み	
--read-from-remote-master	バイナリログをローカルログファイルからではなく MySQL マスターから読み取り	5.6.5

オプション名	説明	導入
--read-from-remote-server	バイナリログをローカルログファイルからではなく MySQL サーバーから読み取り	
--result-file	指定されたファイルに出力を送信	
--secure-auth	古い (4.1.1 より前の) 形式でサーバーにパスワードを送信しない	5.6.17
--server-id	指定されたサーバー ID を持つサーバーによって作成されたイベントのみを抽出	
--server-id-bits	サーバー ID ビットが最大値未満に設定されている mysqld によってログが書き込まれた場合に、バイナリログのサーバー ID を解釈する方法を、mysqlbinlog に対して指定。MySQL Cluster バージョンの mysqlbinlog でのみサポート	
--set-charset	SET NAMES charset_name ステートメントを出力に追加	
--shared-memory-base-name	共有メモリー接続に使用する共有メモリーの名前	
--short-form	ログに含まれるステートメントのみを表示	
--skip-gtids	GTID を出力しない。これは、GTID を含むバイナリログからのダンプファイルを書き込む場合に使用します。	5.6.5
--socket	ローカルホストへの接続で、使用する Unix ソケットファイル	
--ssl-crl	証明書失効リストを含むファイルのパス	5.6.3
--ssl-crlpath	証明書失効リストファイルを含むディレクトリのパス	5.6.3
--start-datetime	タイムスタンプが datetime 引数と同じかそれよりあとの最初のイベントからバイナリログを読み取り	
--start-position	位置が引数と同じかそれより大きい最初のイベントからバイナリログを読み取り	
--stop-datetime	タイムスタンプが datetime 引数と同じかそれより大きい最初のイベントでバイナリログの読み取りを停止	
--stop-never	最後のバイナリログファイルの読み取り後、サーバーとの接続を維持	
--stop-never-slave-server-id	サーバーへの接続時にレポートするスレーブサーバー ID	
--stop-position	位置が引数と同じかそれより大きい最初のイベントでバイナリログの読み取りを停止	
--to-last-log	MySQL サーバーから要求されたバイナリログの最後で停止せず、最後のバイナリログまで続けて出力	
--user	サーバーへの接続時に使用する MySQL ユーザー名	
--verbose	行イベントを SQL ステートメントとして再構築	
--verify-binlog-checksum	バイナリログのチェックサムを検証	5.6.1
--version	バージョン情報を表示して終了	

- --help, -?

  ヘルプメッセージを表示して終了します。

- --base64-output=value

  このオプションは、イベントをいつ BINLOG ステートメントを使用して、base-64 文字列としてエンコードして表示するべきかを決定します。このオプションには次の可能な値があります (大文字小文字は区別されません)。

  - AUTO (「自動」) または UNSPEC (「未指定」) では、必要なときに (すなわち、形式記述イベントおよび行イベント) 自動的に BINLOG ステートメントを表示します。--base64-output オプションが指定されない場合は、効果は --base64-output=AUTO と同じです。

    > **注記**
    >
    > 自動的な BINLOG 表示は、mysqlbinlog の出力を使用してバイナリログファイルの内容を再実行する場合には、唯一の安全な動作です。その他のオプション値はデ

> バッグまたはテスト専用です。これらのオプションで生成される出力には、すべてのイベントが実行可能な形式で含まれるわけではないからです。

- NEVER を使用すると BINLOG ステートメントは表示されなくなります。mysqlbinlog は、BINLOG を使用して表示しなければならない行イベントが検出された場合にはエラーで終了します。
- DECODE-ROWS は、--verbose オプションも指定することによって、行イベントをコメント付きの SQL ステートメントとしてデコードおよび表示することをユーザーが意図していることを、mysqlbinlog に指定します。NEVER と同様に、DECODE-ROWS は BINLOG ステートメントの表示を抑制しますが、NEVER とは異なり、行イベントが検出されてもエラーで終了しません。

--base64-output および --verbose の行イベント出力への影響を示す例は、セクション4.6.8.2「mysqlbinlog 行イベントの表示」を参照してください。

- --bind-address=ip_address

  複数のネットワークインタフェースを持つコンピュータで、このオプションを使用して、MySQL サーバーへの接続に使用するインタフェースを選択します。

  このオプションは MySQL 5.6.1 からサポートされています。

- --binlog-row-event-max-size=N

コマンド行形式	--binlog-row-event-max-size=#
型	数値
デフォルト	4294967040
最小値	256
最大値	18446744073709547520

  行ベースのバイナリログイベントの最大サイズをバイト単位で指定します。可能であれば、行はこのサイズより小さいイベントにグループ化されます。値は 256 の倍数であるべきです。デフォルトは 4G バイトです。

- --character-sets-dir=path

  文字セットがインストールされているディレクトリ。セクション10.5「文字セットの構成」を参照してください。

- --connection-server-id=server_id

  このオプションは、MySQL サーバーが BINLOG_DUMP_NON_BLOCK 接続フラグをサポートするかどうかをテストするために使用されます。このフラグは何らかの都合で MySQL 5.6.5 で削除され、MySQL 5.6.20 でリストアされました (Bug #18000079、Bug #71178)。通常の操作では不要です。

  このオプションの実質的なデフォルトと最小値は、mysqlbinlog がブロッキングモードで実行されているか非ブロッキングモードで実行されているかによって異なります。mysqlbinlog がブロッキングモードで実行されている場合は、デフォルト (および最小) 値は 1 です。非ブロッキングモードで実行されている場合は、デフォルト (および最小) 値は 0 です。

  このオプションは MySQL 5.6.20 で追加されました。

- --database=db_name, -d db_name

  このオプションを使用すると、mysqlbinlog は、USE によって db_name がデフォルトデータベースとして選択されている間に発生するバイナリログ (ローカルログのみ) からのエントリを出力するようになります。

  mysqlbinlog の --database オプションは、mysqld の --binlog-do-db オプションと同様ですが、指定できるのは 1 つのデータベースのみです。--database を複数回指定すると、最後のインスタンスのみが使用されます。

  このオプションの影響は、ステートメントベースまたは行ベースの口ギング形式のどちらが使用されているかによって異なります。これは、--binlog-do-db の影響がステートメントベースまたは行ベースのいずれの口ギングが使用されているかによって異なるのと同じです。

  ステートメントベースの口ギング　--database オプションは次のように機能します。

  - db_name がデフォルトデータベースである間、ステートメントは db_name または別のデータベースのテーブルを修正する場合でも出力されます。
  - db_name がデフォルトデータベースとして選択されていない場合は、db_name のテーブルを修正する場合でもステートメントは出力されません。
  - CREATE DATABASE、ALTER DATABASE、および DROP DATABASE は例外です。ステートメントを出力するかどうかを判断するときには、作成、変更、または削除されたデータベースがデフォルトのデータベースであるとみなされます。

  これらのステートメントを実行することによって、ステートメントベースの口ギングを使用してバイナリログが作成されたとします。

```
INSERT INTO test.t1 (i) VALUES(100);
INSERT INTO db2.t2 (j)  VALUES(200);
USE test;
INSERT INTO test.t1 (i) VALUES(101);
INSERT INTO t1 (i)      VALUES(102);
INSERT INTO db2.t2 (j)  VALUES(201);
USE db2;
INSERT INTO test.t1 (i) VALUES(103);
INSERT INTO db2.t2 (j)  VALUES(202);
INSERT INTO t2 (j)      VALUES(203);
```

  デフォルトデータベースがないため、mysqlbinlog --database=test は最初の 2 つの INSERT ステートメントを出力しません。USE test に続く 3 つの INSERT ステートメントは出力しますが、USE db2 に続く 3 つの INSERT ステートメントは出力しません。

  デフォルトデータベースがないため、mysqlbinlog --database=db2 は最初の 2 つの INSERT ステートメントを出力しません。USE test に続く 3 つの INSERT ステートメントは出力しませんが、USE db2 に続く 3 つの INSERT ステートメントは出力します。

  行ベースの口ギング　mysqlbinlog は、db_name に属するテーブルを変更するエントリのみを出力します。これにはデフォルトのデータベースには影響しません。今説明したバイナリログが、ステートメントベースの口ギングではなく行ベースの口ギングを使用して作成されたとします。mysqlbinlog --database=test は、USE が発行されたか、またはデフォルトのデータベースが何かにかかわらず、テストデータベースの t1 を変更するエントリのみを出力します。

  サーバーが、binlog_format が MIXED に設定された状態で稼働していて、mysqlbinlog を --database オプションで使用できるようにする場合、変更されるテーブルが USE で選択されたデータベースであることを保証する必要があります。(特に、クロスデータベースの更新は使用しないようにしてください。)

  MySQL 5.6.10 より前では、GTID が有効な MySQL サーバーによって書き出されたログでは、--database オプションは正しく機能しませんでした。(Bug#15912728)

- --debug[=debug_options], -# [debug_options]

  デバッグのログを書き込みます。一般的な debug_options 文字列は d:t:o,file_name です。デフォルトは d:t:o,/tmp/mysqlbinlog.trace です。

- --debug-check

  プログラムの終了時に、デバッグ情報を出力します。

- --debug-info

  プログラムの終了時に、デバッグ情報とメモリーおよび CPU 使用率の統計を出力します。

- --default-auth=plugin

  使用するクライアント側の認証プラグイン。セクション6.3.7「プラガブル認証」を参照してください。

  このオプションは MySQL 5.6.2 で追加されました。

- --defaults-extra-file=file_name

  このオプションファイルは、グローバルオプションファイルのあとに読み取りますが、(UNIX では) ユーザーオプションファイルの前に読み取るようにしてください。ファイルが存在しないかアクセスできない場合、エラーが発生します。file_name は、フルパス名でなく相対パス名として指定された場合、現行ディレクトリを基準にして解釈されます。

- --defaults-file=file_name

  指定されたオプションファイルのみ使用します。ファイルが存在しないかアクセスできない場合、エラーが発生します。file_name は、フルパス名でなく相対パス名として指定された場合、現行ディレクトリを基準にして解釈されます。

- --defaults-group-suffix=str

  通常のオプショングループだけでなく、通常の名前に str のサフィクスが付いたグループも読み取ります。たとえば、mysqlbinlog は通常 [client] グループおよび [mysqlbinlog] グループを読み取ります。--defaults-group-suffix=_other オプションを指定した場合、mysqlbinlog は [client_other] グループおよび [mysqlbinlog_other] グループも読み取ります。

- --disable-log-bin, -D

  バイナリロギングを無効化します。これは、--to-last-log オプションを使用して同じ MySQL サーバーに対して出力を送信している場合、無限ループを回避するのに便利です。このオプションは、クラッシュ後、すでに口グに記録したステートメントの重複を回避するのにも便利です。

  このオプションには SUPER 権限を保持していることが必要です。mysqlbinlog が、出力に SET sql_log_bin = 0 ステートメントを含め、そのあとの出力のバイナリロギングを無効にするようになります。SET ステートメントは、SUPER 権限がない場合は効果がありません。

- --exclude-gtids=gtid_set

  gtid_set にリストされたグループを表示しません。MySQL 5.6.5 で追加されました。

- --force-if-open, -F

  バイナリログファイルが開いているか適切に閉じられていない場合でも読み取ります。

- --force-read, -f

  このオプションでは、mysqlbinlog が認識できないバイナリログイベントを読み取った場合に、警告を出力し、イベントを無視して続行します。このオプションを使用しない場合は、mysqlbinlog はそのようなイベントを読み取ると停止します。

- --hexdump, -H

  セクション4.6.8.1「mysqlbinlog 16 進ダンプ形式」に説明されているように、ログの 16 進ダンプをコメントに表示します。16 進出力はレプリケーションのデバッグに便利な場合があります。

- --host=host_name, -h host_name

  指定されたホストの MySQL サーバーからバイナリログを取得します。

- --include-gtids=gtid_set

  gtid_set にリストされたグループのみを表示します。MySQL 5.6.5 で追加されました。

- --local-load=path, -l path

  指定されたディレクトリに LOAD DATA INFILE 用のローカル一時ファイルを準備します。

> **重要**
>
> これらの一時ファイルは、mysqlbinlog およびその他のどの MySQL プログラムによっても自動的に削除されません。

- --login-path=name

  指名されたログインパスから .mylogin.cnf ログインファイルのオプションを読み取ります。「ログインパス」は、host、user、および password という限定されたオプションのセットのみを許可するオプショングループです。ログインパスは、サーバーホストおよびそのサーバーで認証するための認証情報を示す値のセットであると考えてください。ログインパスファイルを作成するには、mysql_config_editor ユーティリティーを使用します。セクション4.6.6「mysql_config_editor — MySQL 構成ユーティリティー」を参照してください。このオプションは MySQL 5.6.6 で追加されました。

- --no-defaults

  オプションファイルを読み取りません。オプションファイルから不明のオプションを読み取ることが原因でプログラムの起動に失敗する場合、--no-defaults を使用して、オプションを読み取らないようにできます。

  例外として、.mylogin.cnf ファイルは、存在する場合はすべての場合に読み取られます。これにより、--no-defaults が使用された場合にも、コマンド行よりも安全な方法でパスワードを指定できます。(.mylogin.cnf は mysql_config_editor ユーティリティーによって作成されます。セクション4.6.6「mysql_config_editor — MySQL 構成ユーティリティー」を参照してください)。

- --offset=N, -o N

  ログの最初の N 個のエントリをスキップします。

- --password[=password], -p[password]

  サーバーに接続する際に使用するパスワードです。短いオプション形式 (-p) を使用した場合は、オプションとパスワードの間にスペースを置くことはできません。コマンド行で、--password オプションまたは -p オプションに続けて password の値を指定しなかった場合、mysqlbinlog はそれを要求します。

  コマンド行でパスワードを指定することはセキュアでないとみなすべきです。セクション6.1.2.1「パスワードセキュリティーのためのエンドユーザーガイドライン」を参照してください。オプションファイルを使用すれば、コマンド行でパスワードを指定することを回避できます。

- --plugin-dir=path

  プラグインを検索するディレクトリ。--default-auth オプションを使用して認証プラグインを指定したが、mysqlbinlog がそれを検出できない場合は、このオプションを指定しなければならない可能性があります。セクション6.3.7「プラガブル認証」を参照してください。

  このオプションは MySQL 5.6.2 で追加されました。

- --port=port_num, -P port_num

  リモートサーバーへの接続に使用する TCP/IP ポート番号。

- --print-defaults

  プログラム名と、オプションファイルから受け取るすべてのオプションを出力します。

- --protocol={TCP|SOCKET|PIPE|MEMORY}

  サーバーへの接続に使用する接続プロトコル。このオプションは、ほかの接続パラメータによって、必要なプロトコル以外のものが通常使用される場合に役立ちます。許可される値の詳細は、セクション4.2.2「MySQL サーバーへの接続」を参照してください。

- --raw

  デフォルトでは、mysqlbinlog はバイナリログファイルを読み取り、イベントをテキスト形式で書き出します。--raw オプションは mysqlbinlog に対して、元のバイナリ形式で書き出すことを指示します。ファイルはサーバーから要求されるため、これを使用するには、--read-from-remote-server も使用する必要があります。mysqlbinlog はサーバーから読み取った各ファイルに対して 1 つの出力ファイルを書き出します。--raw オプションは、サーバーのバイナリログのバックアップを作成するために使用できます。--stop-never オプションを使用すると、mysqlbinlog はサーバーに接続されたままになるため、バックアップは「ライブ」です。デフォ

ルトでは、出力ファイルは現在のディレクトリに元のログファイルと同じ名前で書き出されます。出力ファイル名は --result-file オプションを使用して変更できます。詳細は、セクション4.6.8.3「バイナリログファイルのバックアップのための mysqlbinlog の使用」を参照してください。

このオプションは MySQL 5.6.0 で追加されました。

- --read-from-remote-master=type

  COM_BINLOG_DUMP コマンドまたは COM_BINLOG_DUMP_GTID コマンドで、オプション値をそれぞれ BINLOG-DUMP-NON-GTIDS または BINLOG-DUMP-GTIDS に設定して、MySQL サーバーからバイナリログを読み取ります。--read-from-remote-master=BINLOG-DUMP-GTIDS が --exclude-gtids と組み合わされると、トランザクションをマスターでフィルタでき、不要なネットワークトラフィックを防ぐことができます。

  --read-from-remote-server の説明も参照してください。

  このオプションは MySQL 5.6.5 で追加されました。

- --read-from-remote-server, -R

  バイナリログをローカルログファイルからではなく MySQL サーバーから読み取ります。次のオプションも指定されていないかぎり、接続パラメータオプションはすべて無視されます。これらのオプションは --host、--password、--port、--protocol、--socket、および --user です。

  このオプションでは、リモートサーバーが稼働していることが必要です。リモートサーバー上のバイナリログファイルに対してのみ機能します。リレーログファイルには機能しません。

  MySQL 5.6.5 では、このオプションは --read-from-remote-master=BINLOG-DUMP-NON-GTIDS と同様です。

- --result-file=name, -r name

  --raw オプションがない場合は、このオプションは mysqlbinlog がテキスト出力を書き出すファイルを示します。--raw がある場合は、mysqlbinlog はサーバーから転送される各ログファイルに対して 1 つのバイナリ出力ファイルを、デフォルトでは現在のディレクトリに元のログファイルと同じ名前で書き出します。この場合、--result-file オプションの値は出力ファイル名を変更するプリフィクスとして処理されます。

- --secure-auth

  古い (4.1 より前の) 形式でサーバーにパスワードを送信しません。これにより、新しいパスワード形式を使用するサーバー以外への接続を防ぎます。このオプションはデフォルトで有効です。無効にするには --skip-secure-auth を使用します。このオプションは MySQL 5.6.17 で追加されました。

  > **注記**
  >
  > 4.1 以前のハッシュ方式を使用するパスワードはネイティブのパスワードハッシュ方式を使用するパスワードよりもセキュアでないため、使用しないようにしてください。4.1 よりも前のパスワードは非推奨であり、これらのサポートは今後の MySQL リリースで削除される予定です。アカウントのアップグレード手順については、セクション6.3.8.3「4.1 よりも前のパスワードハッシュ方式と mysql_old_password プラグインからの移行」を参照してください。

- --server-id=id

  指定されたサーバー ID を持つサーバーによって作成されたイベントのみを表示します。

- --server-id-bits=N

  サーバーを特定するために、server_id の最初の N ビットのみを使用します。server-id-bits が 32 未満にセットされ、ユーザーデータが最上位ビットに保存される mysqld によってバイナリログが書き出された場合、--server-id-bits を 32 にセットして mysqlbinlog を実行するとこのデータを表示できます。

  このオプションは、MySQL Cluster 配布で提供されるバージョン、または MySQL Cluster ソースからビルドされた mysqlbinlog によってのみサポートされます。

- --set-charset=charset_name

  SET NAMES charset_name ステートメントを出力に追加して、ログファイルの処理に使用される文字セットを指定します。

- --shared-memory-base-name=name

Windows で、共有メモリーを使用して作成されるローカルサーバーへの接続の共有メモリー名。デフォルト値は MYSQL です。共有メモリー名では大文字と小文字を区別します。

共有メモリー接続を可能にするには、サーバーは --shared-memory オプションで起動する必要があります。

- --short-form, -s

  追加情報および行ベースのイベントなしで、ログに含まれるステートメントのみを表示します。これはテスト専用で、本番環境のシステムで使用するべきではありません。

- --skip-gtids[=(true|false)]

  出力に GTID を表示しません。これは、次の例に示すように GTID を含む 1 つまたは複数のバイナリログからダンプファイルに書き出す場合に必要です。

```
shell> mysqlbinlog --skip-gtids binlog.000001 >  /tmp/dump.sql
shell> mysqlbinlog --skip-gtids binlog.000002 >> /tmp/dump.sql
shell> mysql -u root -p -e "source /tmp/dump.sql"
```

  そうでない場合には、このオプションを本番環境で使用することは、通常推奨されません。

  このオプションは MySQL 5.6.5 で追加されました。

- --socket=path, -S path

  localhost への接続用に使用する、Unix ソケットファイル、または Windows では使用する名前付きパイプの名前。

- --start-datetime=datetime

  datetime 引数と同じかそれより遅いタイムスタンプを持つ最初のイベントから、バイナリログの読み取りを始めます。datetime 値は、mysqlbinlog を実行するマシンのローカルタイムゾーンに相対的です。値は DATETIME または TIMESTAMP データ型に受け付けられる形式にしてください。例:

```
shell> mysqlbinlog --start-datetime="2005-12-25 11:25:56" binlog.000003
```

  このオプションはポイントインタイムリカバリに便利です。セクション7.3「バックアップおよびリカバリ戦略の例」を参照してください。

- --start-position=N, -j N

  N 以上の位置を持つ最初のイベントから、バイナリログの読み取りを始めます。このオプションはコマンド行で最初に指名された口グファイルに適用されます。

  このオプションはポイントインタイムリカバリに便利です。セクション7.3「バックアップおよびリカバリ戦略の例」を参照してください。

- --stop-datetime=datetime

  datetime 引数と同じかそれより遅いタイムスタンプを持つ最初のイベントで、バイナリログの読み取りを終了します。このオプションはポイントインタイムリカバリに便利です。datetime 値の詳細については、--start-datetime オプションの説明を参照してください。

  このオプションはポイントインタイムリカバリに便利です。セクション7.3「バックアップおよびリカバリ戦略の例」を参照してください。

- --stop-never

  このオプションは --read-from-remote-server とともに使用されます。mysqlbinlog に対して、サーバーに接続したままの状態を保つことを指示します。そうしないと、mysqlbinlog は最後のログファイルがサーバーから転送された時点で終了します。--stop-never は暗黙的に --to-last-log を指定するため、コマンド行で指名する必要があるのは転送される最初の口グファイルのみです。

  --stop-never は一般的に、ライブバイナリログバックアップを作成するために --raw とともに使用されますが、--raw なしで使用して、サーバーがログイベントを生成するに従ってそれらを継続的にテキスト表示することも可能です。

  このオプションは MySQL 5.6.0 で追加されました。

- --stop-never-slave-server-id=id

--stop-never を使用すると、mysqlbinlog はサーバーに接続する際にサーバー ID の 65535 をレポートします。--stop-never-slave-server-id はレポートするサーバー ID を明示的に指定します。スレーブサーバーまたは別の mysqlbinlog プロセスの ID との競合を避けるために使用できます。セクション4.6.8.4「mysqlbinlog サーバー ID の指定」を参照してください。

このオプションは MySQL 5.6.0 で追加されました。

- --stop-position=N

  N 以上の位置を持つ最初のイベントで、バイナリログの読み取りを終了します。このオプションは、コマンド行で最後に指名されたログファイルに適用されます。

  このオプションはポイントインタイムリカバリに便利です。セクション7.3「バックアップおよびリカバリ戦略の例」を参照してください。

- --to-last-log, -t

  MySQL サーバーから要求されたバイナリログの最後で終了せず、最後のバイナリログの最後まで続けて出力します。同じ MySQL サーバーに出力を送信した場合、無限ループになる場合があります。このオプションには --read-from-remote-server が必要です。

- --user=user_name, -u user_name

  リモートサーバーへの接続時に使用する MySQL ユーザー名。

- --verbose, -v

  行イベントを再構築し、コメント化された SQL ステートメントとして表示します。このオプションを 2 回指定すると、出力にはカラムデータ型と一部のメタデータを示すコメントが含まれます。

  --base64-output および --verbose の行イベント出力への影響を示す例は、セクション4.6.8.2「mysqlbinlog 行イベントの表示」を参照してください。

- --verify-binlog-checksum, -c

  バイナリログファイルのチェックサムを検証します。このオプションは MySQL 5.6.1 で追加されました。

- --version, -V

  バージョン情報を表示して終了します。

  MySQL 5.6.11 より前では、表示される mysqlbinlog のバージョン番号は 3.3 でした。MySQL 5.6.11 以降では 3.4 です。(Bug #15894381、Bug #67643)

--var_name=value 構文を使用して、次の変数を設定することもできます。

- open_files_limit

  予約するオープンファイルディスクリプタの数を指定します。

mysqlbinlog の出力を mysql クライアントにパイプして、バイナリログに含まれるイベントを実行できます。このテクニックは、古いバックアップがある場合にクラッシュからリカバリするために使用されます (セクション7.5「バイナリログを使用したポイントインタイム (増分) リカバリ」を参照してください)。例:

```
shell> mysqlbinlog binlog.000001 | mysql -u root -p
```

または:

```
shell> mysqlbinlog binlog.[0-9]* | mysql -u root -p
```

mysqlbinlog が生成したステートメントに BLOB 値が含まれる可能性がある場合、mysql がそれらを処理するときに問題が生じることがあります。この場合は、mysql を --binary-mode オプションで起動します。

ステートメントログをまず変更する必要がある場合 (たとえば、何らかの理由で実行しないステートメントを削除するなど) は、代わりに mysqlbinlog の出力をテキストファイルにリダイレクトすることもできます。ファイルの編集後、mysql プログラムへの入力として使用することによって、そこに含まれるステートメントを実行します。

```
shell> mysqlbinlog binlog.000001 > tmpfile
```

```
shell> ... edit tmpfile ...
shell> mysql -u root -p < tmpfile
```

mysqlbinlog は、--start-position オプションで起動された場合、バイナリログ内のオフセットが指定された位置以上のイベントのみを表示します (指定された位置は 1 つのイベントの開始位置に一致していなければなりません)。指定された日付と時間を持つイベントを検出したときに停止および起動するオプションもあります。これにより、--stop-datetime オプションを使用してポイントインタイムリカバリが実行できます (たとえば、「データベースを今日の 10:30 am 現在の状態までロールバックする」といったことが可能になります)。

MySQL サーバーに実行する複数のバイナリログがある場合、安全な方法は、サーバーへの 1 つの接続を使用して、それらすべてを処理することです。これは、安全でない可能性があることを示す例です。

```
shell> mysqlbinlog binlog.000001 | mysql -u root -p # DANGER!!
shell> mysqlbinlog binlog.000002 | mysql -u root -p # DANGER!!
```

サーバーに対してこのように複数の接続を使用してバイナリログを処理する場合、最初のログファイルに CREATE TEMPORARY TABLE ステートメントが含まれており、2 番目のログには一時テーブルを使用するステートメントが含まれていると、問題が発生します。最初の mysql プロセスが終了すると、サーバーは一時テーブルを削除します。2 番目の mysql プロセスでテーブルの使用を試みると、サーバーは「不明なテーブル」と報告します。

このような問題を回避するには、1 つの mysql プロセスを使用して、処理するすべてのバイナリログの内容を実行します。これはそれを実行する 1 つの方法です。

```
shell> mysqlbinlog binlog.000001 binlog.000002 | mysql -u root -p
```

もう 1 つのアプローチは、すべてのログを 1 つのファイルに書き込み、次にそのファイルを処理することです。

```
shell> mysqlbinlog binlog.000001 >  /tmp/statements.sql
shell> mysqlbinlog binlog.000002 >> /tmp/statements.sql
shell> mysql -u root -p -e "source /tmp/statements.sql"
```

mysqlbinlog は、元のデータファイルなしで LOAD DATA INFILE 操作を再現する出力を生成できます。mysqlbinlog はデータを一時ファイルにコピーし、そのファイルを参照する LOAD DATA LOCAL INFILE ステートメントを書き出します。これらのファイルが書き出されるディレクトリのデフォルトの場所はシステムごとに異なります。ディレクトリを明示的に指定するには、--local-load オプションを使用します。

mysqlbinlog は LOAD DATA INFILE ステートメントを LOAD DATA LOCAL INFILE ステートメントに変換するため (つまり、LOCAL を追加します)、ステートメントの処理に使用するクライアントとサーバーの両方が、LOCAL 機能を有効にして構成されていなければなりません。セクション6.1.6「LOAD DATA LOCAL のセキュリティーの問題」を参照してください。

> **警告**
>
> LOAD DATA LOCAL ステートメント用に作成された一時ファイルは、これらのステートメントをユーザーが実際に実行するまで必要なため、自動的には削除されません。ステートメントログが必要なくなった時点で、ユーザーが一時ファイルを削除するようにしてください。これらのファイルは一時ファイルディレクトリに存在し、original_file_name-#-# といった名前が付いています。

4.6.8.1 mysqlbinlog 16 進ダンプ形式

--hexdump オプションを使用すると、mysqlbinlog はバイナリログの内容の 16 進ダンプを生成するようになります。

```
shell> mysqlbinlog --hexdump master-bin.000001
```

16 進出力は、# で始まるコメント行で構成され、前のコマンドの出力は次のようになります。

```
/*!40019 SET @@SESSION.max_insert_delayed_threads=0*/;
/*!50003 SET @OLD_COMPLETION_TYPE=@@COMPLETION_TYPE,COMPLETION_TYPE=0*/;
# at 4
#051024 17:24:13 server id 1  end_log_pos 98
# Position  Timestamp   Type   Master ID        Size      Master Pos    Flags
# 00000004 9d fc 5c 43   0f   01 00 00 00   5e 00 00 00   62 00 00 00   00 00
# 00000017 04 00 35 2e 30 2e 31 35  2d 64 65 62 75 67 2d 6c |..5.0.15.debug.l|
# 00000027 6f 67 00 00 00 00 00 00  00 00 00 00 00 00 00 00 |og..............|
# 00000037 00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00 |................|
# 00000047 00 00 00 00 9d fc 5c 43  13 38 0d 00 08 00 12 00 |.......C.8......|
# 00000057 04 04 04 04 12 00 00 4b  00 04 1a               |.......K...|
#       Start: binlog v 4, server v 5.0.15-debug-log created 051024 17:24:13
```

```
#       at startup
ROLLBACK;
```

現在、16 進出力には次のリストの要素が含まれます。この形式は変更される場合があります。(バイナリログの形式の詳細は、「MySQL Internals: The Binary Log」を参照してください。

- Position: ログファイル内のバイト位置。
- Timestamp: イベントのタイムスタンプ。示した例では、'9d fc 5c 43' は '051024 17:24:13' の 16 進表現です。
- Type: イベントタイプコード。示した例では、'0f' は FORMAT_DESCRIPTION_EVENT を示します。次のテーブルは可能なタイプコードのリストです。

型	名前	意味
00	UNKNOWN_EVENT	このイベントはログ内に存在するべきではありません。
01	START_EVENT_V3	MySQL 4 より前のバージョンによって書き出されたログファイルの始めを示しています。
02	QUERY_EVENT	もっとも一般的なイベントのタイプ。これらはマスターで実行されるステートメントを含んでいます。
03	STOP_EVENT	マスターが停止したことを示します。
04	ROTATE_EVENT	マスターが新しいログファイルにスイッチした際に書き出されます。
05	INTVAR_EVENT	AUTO_INCREMENT 値、またはステートメントで LAST_INSERT_ID() 関数が使用される場合に使用されます。
06	LOAD_EVENT	MySQL 3.23 で LOAD DATA INFILE 用に使用されます。
07	SLAVE_EVENT	将来使用するために予約されています。
08	CREATE_FILE_EVENT	LOAD DATA INFILE ステートメントに使用されます。これはそのようなステートメントの実行の始まりを示しています。スレーブ上で一時ファイルが作成されます。MySQL 4 でのみ使用されます。
09	APPEND_BLOCK_EVENT	LOAD DATA INFILE ステートメントで使用されるデータを含んでいます。データはスレーブ上の一時ファイル内に保存されます。
0a	EXEC_LOAD_EVENT	LOAD DATA INFILE ステートメントに使用されます。一時ファイルの内容はスレーブ上のテーブルに保存されます。MySQL 4 でのみ使用されます。
0b	DELETE_FILE_EVENT	LOAD DATA INFILE ステートメントのロールバック。一時ファイルはスレーブ上で削除されるはずです。
0c	NEW_LOAD_EVENT	MySQL 4 以前のバージョンで LOAD DATA INFILE に使用されます。
0d	RAND_EVENT	RAND() 関数がステートメントで使用されている場合、ランダム値の情報を送信するのに使用されます。
0e	USER_VAR_EVENT	ユーザー変数のレプリケーションに使用されます。
0f	FORMAT_DESCRIPTION_EVENT	MySQL 5 以降のバージョンで書き出されたログファイルの始まりを示します。
10	XID_EVENT	XA トランザクションのコミットを示すイベント。
11	BEGIN_LOAD_QUERY_EVENT	MySQL 5 以降で LOAD DATA INFILE ステートメントに使用されます。
12	EXECUTE_LOAD_QUERY_EVENT	MySQL 5 以降で LOAD DATA INFILE ステートメントに使用されます。
13	TABLE_MAP_EVENT	テーブル定義の情報。MySQL 5.1.5 以降で使用されます。
14	PRE_GA_WRITE_ROWS_EVENT	作成されるべき単一のテーブルの行データ。MySQL 5.1.5 から 5.1.17 で使用されます。
15	PRE_GA_UPDATE_ROWS_EVENT	更新が必要な単一テーブルの行データ。MySQL 5.1.5 から 5.1.17 で使用されます。
16	PRE_GA_DELETE_ROWS_EVENT	削除されるべき単一テーブルの行データ。MySQL 5.1.5 から 5.1.17 で使用されます。

型	名前	意味
17	WRITE_ROWS_EVENT	作成されるべき単一のテーブルの行データ。MySQL 5.1.18 以降で使用されます。
18	UPDATE_ROWS_EVENT	更新が必要な単一テーブルの行データ。MySQL 5.1.18 以降で使用されます。
19	DELETE_ROWS_EVENT	削除されるべき単一テーブルの行データ。MySQL 5.1.18 以降で使用されます。
1a	INCIDENT_EVENT	通常ではないことが何か起こりました。MySQL 5.1.18 で追加されました。

- Master ID: イベントを作成したマスターのサーバー ID。
- Size:イベントのサイズをバイトで表しています。
- Master Pos: 元のマスターログファイル内での次のイベントの位置。
- Flags: 16 フラグ。現在、次のフラグが使用されています。ほかのものは将来のために予約されています。

フラグ	名前	意味
01	LOG_EVENT_BINLOG_IN_USE_F	ログファイルは正しく閉じられました。(FORMAT_DESCRIPTION_EVENT でのみ使用されます。)FORMAT_DESCRIPTION_EVENT 内でこのフラグがセットされている場合 (フラグがたとえば '01 00' の場合)、ログファイルは適切に閉じられていません。マスターが (たとえば、停電などにより) クラッシュしたことによる場合がもっとも可能性が高くなります。
02		将来使用するために予約されています。
04	LOG_EVENT_THREAD_SPECIFIC_F	たとえばイベントが一時テーブルを使用する場合のように、イベントが実行される接続に依存する場合にセットします (たとえば '04 00')。
08	LOG_EVENT_SUPPRESS_USE_F	イベントがデフォルトデータベースに依存していない一部の状況でセットします。

4.6.8.2 mysqlbinlog 行イベントの表示

次の例は、データの変更を指定する行イベントを mysqlbinlog が表示する方法を説明しています。これらは WRITE_ROWS_EVENT、UPDATE_ROWS_EVENT、および DELETE_ROWS_EVENT タイプコードを持つイベントに対応します。--base64-output=DECODE-ROWS オプションおよび --verbose オプションを、行イベント出力に影響を与えるために使用できます。

サーバーが行ベースのバイナリロギングを使用しており、次のステートメントのシーケンスを実行するとします。

```
CREATE TABLE t
(
  id   INT NOT NULL,
  name VARCHAR(20) NOT NULL,
  date DATE NULL
) ENGINE = InnoDB;

START TRANSACTION;
INSERT INTO t VALUES(1, 'apple', NULL);
UPDATE t SET name = 'pear', date = '2009-01-01' WHERE id = 1;
DELETE FROM t WHERE id = 1;
COMMIT;
```

デフォルトでは、mysqlbinlog は行イベントを、BINLOG ステートメントを使用して base-64 文字列としてエンコードして表示します。無関係な行を省略すると、前のステートメントシーケンスによって生成される行イベントの出力は次のようになります。

```
shell> mysqlbinlog log_file
...
# at 218
```

```
#080828 15:03:08 server id 1  end_log_pos 258   Write_rows: table id 17 flags: STMT_END_F

BINLOG '
fAS3SBMBAAAALAAAANoAAAAAABEAAAAAAAAAABHRlc3QAAXQAAwMPCgIUAAQ=
fAS3SBcBAAAAKAAAAAIBAAAQABEAAAAAAAEAA//8AQAAAAVhcHBsZQ==
'/*!*/;
...
# at 302
#080828 15:03:08 server id 1  end_log_pos 356   Update_rows: table id 17 flags: STMT_END_F

BINLOG '
fAS3SBMBAAAALAAAAC4BAAAAABEAAAAAAAAAABHRlc3QAAXQAAwMPCgIUAAQ=
fAS3SBgBAAAANgAAAGQBAAAQABEAAAAAAAEAA////AEAAAAFYXBwbGX4AQAAAARwZWFyIbIP
'/*!*/;
...
# at 400
#080828 15:03:08 server id 1  end_log_pos 442   Delete_rows: table id 17 flags: STMT_END_F

BINLOG '
fAS3SBMBAAAALAAAAJABAAAAABEAAAAAAAAAABHRlc3QAAXQAAwMPCgIUAAQ=
fAS3SBkBAAAAKgAAALoBAAAQABEAAAAAAAEAA//4AQAAAARwZWFyIbIP
'/*!*/;
```

行イベントを「擬似 SQL」 ステートメントの形式のコメントして表示するには、mysqlbinlog を --verbose オプションまたは -v オプションで実行します。出力には ### で始まる行が含まれます。

```
shell> mysqlbinlog -v log_file
...
# at 218
#080828 15:03:08 server id 1  end_log_pos 258   Write_rows: table id 17 flags: STMT_END_F

BINLOG '
fAS3SBMBAAAALAAAANoAAAAAABEAAAAAAAAAABHRlc3QAAXQAAwMPCgIUAAQ=
fAS3SBcBAAAAKAAAAAIBAAAQABEAAAAAAAEAA//8AQAAAAVhcHBsZQ==
'/*!*/;
### INSERT INTO test.t
### SET
###   @1=1
###   @2='apple'
###   @3=NULL
...
# at 302
#080828 15:03:08 server id 1  end_log_pos 356   Update_rows: table id 17 flags: STMT_END_F

BINLOG '
fAS3SBMBAAAALAAAAC4BAAAAABEAAAAAAAAAABHRlc3QAAXQAAwMPCgIUAAQ=
fAS3SBgBAAAANgAAAGQBAAAQABEAAAAAAAEAA////AEAAAAFYXBwbGX4AQAAAARwZWFyIbIP
'/*!*/;
### UPDATE test.t
### WHERE
###   @1=1
###   @2='apple'
###   @3=NULL
### SET
###   @1=1
###   @2='pear'
###   @3='2009:01:01'
...
# at 400
#080828 15:03:08 server id 1  end_log_pos 442   Delete_rows: table id 17 flags: STMT_END_F

BINLOG '
fAS3SBMBAAAALAAAAJABAAAAABEAAAAAAAAAABHRlc3QAAXQAAwMPCgIUAAQ=
fAS3SBkBAAAAKgAAALoBAAAQABEAAAAAAAEAA//4AQAAAARwZWFyIbIP
'/*!*/;
### DELETE FROM test.t
### WHERE
###   @1=1
###   @2='pear'
###   @3='2009:01:01'
```

--verbose または -v を 2 回指定すると、各カラムのデータ型および一部のメタデータも表示されます。出力には、各カラムの変更に続いて追加のコメントが含まれます。

```
shell> mysqlbinlog -vv log_file
...
# at 218
#080828 15:03:08 server id 1  end_log_pos 258   Write_rows: table id 17 flags: STMT_END_F
```

```
BINLOG '
fAS3SBMBAAAALAAAANoAAAAAABEAAAAAAAAAABHRlc3QAAXQAAwMPCgIUAAQ=
fAS3SBcBAAAAKAAAAAIBAAAQABEAAAAAAAEAA//8AQAAAAVhcHBsZQ==
'/*!*/;
### INSERT INTO test.t
### SET
###   @1=1 /* INT meta=0 nullable=0 is_null=0 */
###   @2='apple' /* VARSTRING(20) meta=20 nullable=0 is_null=0 */
###   @3=NULL /* VARSTRING(20) meta=0 nullable=1 is_null=1 */
...
# at 302
#080828 15:03:08 server id 1  end_log_pos 356   Update_rows: table id 17 flags: STMT_END_F

BINLOG '
fAS3SBMBAAAALAAAAC4BAAAAABEAAAAAAAAAABHRlc3QAAXQAAwMPCgIUAAQ=
fAS3SBgBAAAANgAAAGQBAAAQABEAAAAAAAEAA////AEAAAAFYXBwbGX4AQAAAARwZWFyIbIP
'/*!*/;
### UPDATE test.t
### WHERE
###   @1=1 /* INT meta=0 nullable=0 is_null=0 */
###   @2='apple' /* VARSTRING(20) meta=20 nullable=0 is_null=0 */
###   @3=NULL /* VARSTRING(20) meta=0 nullable=1 is_null=1 */
### SET
###   @1=1 /* INT meta=0 nullable=0 is_null=0 */
###   @2='pear' /* VARSTRING(20) meta=20 nullable=0 is_null=0 */
###   @3='2009:01:01' /* DATE meta=0 nullable=1 is_null=0 */
...
# at 400
#080828 15:03:08 server id 1  end_log_pos 442   Delete_rows: table id 17 flags: STMT_END_F

BINLOG '
fAS3SBMBAAAALAAAAJABAAAAABEAAAAAAAAAABHRlc3QAAXQAAwMPCgIUAAQ=
fAS3SBkBAAAAKgAAALoBAAAQABEAAAAAAAEAA//4AQAAAARwZWFyIbIP
'/*!*/;
### DELETE FROM test.t
### WHERE
###   @1=1 /* INT meta=0 nullable=0 is_null=0 */
###   @2='pear' /* VARSTRING(20) meta=20 nullable=0 is_null=0 */
###   @3='2009:01:01' /* DATE meta=0 nullable=1 is_null=0 */
```

--base64-output=DECODE-ROWS オプションを使用して、行イベントに対する BINLOG ステートメントを抑制するように mysqlbinlog に指示できます。これは --base64-output=NEVER と同様ですが、行イベントが検出された場合にエラーで終了しません。--base64-output=DECODE-ROWS および --verbose の組み合わせにより、行イベントを SQL ステートメントとしてのみ表示する便利な方法が提供されます。

```
shell> mysqlbinlog -v --base64-output=DECODE-ROWS log_file
...
# at 218
#080828 15:03:08 server id 1  end_log_pos 258   Write_rows: table id 17 flags: STMT_END_F
### INSERT INTO test.t
### SET
###   @1=1
###   @2='apple'
###   @3=NULL
...
# at 302
#080828 15:03:08 server id 1  end_log_pos 356   Update_rows: table id 17 flags: STMT_END_F
### UPDATE test.t
### WHERE
###   @1=1
###   @2='apple'
###   @3=NULL
### SET
###   @1=1
###   @2='pear'
###   @3='2009:01:01'
...
# at 400
#080828 15:03:08 server id 1  end_log_pos 442   Delete_rows: table id 17 flags: STMT_END_F
### DELETE FROM test.t
### WHERE
###   @1=1
###   @2='pear'
###   @3='2009:01:01'
```

> **注記**
>
> mysqlbinlog の出力を再実行する予定である場合は、BINLOG ステートメントを抑制するべきではありません。

--verbose が行イベントに関して生成した SQL ステートメントは、対応する BINLOG ステートメントよりもはるかに読みやすくなります。ただし、イベントを生成した元の SQL ステートメントと正確には対応しません。次の制約が適用されます。

- 元のカラム名は失われて @N に置換されます。N はカラム番号です。
- バイナリログでは文字セット情報は利用できません。これは文字列カラムの表示に影響します。
  - 対応するバイナリ文字列型と非バイナリ文字列型の間に区別はありません (BINARY と CHAR、VARBINARY と VARCHAR、BLOB と TEXT)。出力では固定長文字列に対して STRING、可変長文字列に対して VARSTRING データ型が使用されます。
  - マルチバイト文字セットでは、文字当たりの最大のバイト数はバイナリログには現れないため、文字列型の長さは文字数ではなくバイト数で表示されます。たとえば、STRING(4) は次のカラム型のいずれかからの値のデータ型として使用されます。

    ```
    CHAR(4) CHARACTER SET latin1
    CHAR(2) CHARACTER SET ucs2
    ```

  - UPDATE_ROWS_EVENT 型のイベントのストレージフォーマットのため、UPDATE ステートメントは WHERE 句が SET 句の前に表示されます。

行イベントを適切に解釈するには、バイナリログの最初にある形式の記述の情報が必要です。mysqlbinlog はログの残りの部分に行イベントが含まれるかどうかは前もってわからないため、デフォルトでは出力の最初の部分に BINLOG ステートメントを使用して形式記述イベントを表示します。

バイナリログに BINLOG ステートメントを必要とするイベントが含まれないことがわかっている (つまり行イベントがない) 場合は、--base64-output=NEVER オプションを使用してこのヘッダーが書き込まれるのを回避できます。

4.6.8.3 バイナリログファイルのバックアップのための mysqlbinlog の使用

デフォルトでは、mysqlbinlog はバイナリログファイルを読み取り、その内容をテキスト形式で表示します。これにより、ファイル内のイベントが調べやすくなり、(たとえば出力を mysql への入力として使用して) それらを再実行できます。mysqlbinlog はローカルファイルシステムから直接ログファイルを読み取るか、または --read-from-remote-server オプションで、サーバーに接続してそのサーバーからバイナリログの内容を要求できます。mysqlbinlog はテキスト出力を標準出力、または --result-file=file_name オプションが指定された場合はその値で指名されるファイルに書き出すことができます。

MySQL 5.6 では、mysqlbinlog はバイナリログファイルを読み取って、同じ内容、つまりテキスト形式ではなくバイナリ形式の内容を含む新しいファイルに書き出すことができます。この機能により、バイナリログを容易に元の形式でバックアップできます。mysqlbinlog は、ログファイルのセットのバックアップを実行して最後のファイルの最後に到達したときに停止して、静的バックアップを作成できます。また、最後のログファイルの最後に到達したときにサーバーとの接続を保ち、新しいイベントが生成されるたびにコピーを継続して、連続的な (「ライブ」) バックアップを作成することもできます。連続的なバックアップ操作では、mysqlbinlog は接続が終了するまで (たとえばサーバーが終了するときなど)、または mysqlbinlog が強制的に終了させられるまで実行されます。接続が終了する際、スレーブレプリケーションサーバーとは異なり、mysqlbinlog は待機して再接続を試みません。サーバーの再起動後にライブバックアップを続行するには、mysqlbinlog も再起動する必要があります。

バイナリログバックアップには、mysqlbinlog を少なくとも 2 つのオプションを使用して起動する必要があります。

- --read-from-remote-server (または -R) オプションは mysqlbinlog に、サーバーに接続してバイナリログを要求するよう指示します。(これは、マスターサーバーに接続するスレーブレプリケーションサーバーと同様です。)
- --raw オプションは mysqlbinlog に、テキスト出力ではなく生の (バイナリ) 出力を書き出すように指示します。

--read-from-remote-server オプションに加えて、次のオプションを指定するのが一般的です。--host はサーバーが稼働している場所を示し、--user および --password などの接続オプションも必要な場合があります。

--raw とともに使用すると便利なオプションがほかにいくつかあります。

- --stop-never: 最後のログファイルの最後に到達したあと、サーバーとの接続を維持して新しいイベントの読み取りを続行します。
- --stop-never-slave-server-id=id: --stop-never が使用される場合に mysqlbinlog がレポートするサーバー ID。デフォルトは 65535 です。これは、スレーブサーバーまたは別の mysqlbinlog プロセスの ID との競合を避けるために使用できます。セクション4.6.8.4「mysqlbinlog サーバー ID の指定」を参照してください。

- --result-file: あとで説明するように、出力ファイル名のプリフィクス。

mysqlbinlog でサーバーのバイナリログファイルのバックアップを行うには、サーバーに実際に存在するファイル名を指定する必要があります。名前がわからない場合は、サーバーに接続して SHOW BINARY LOGS ステートメントを使用して現在の名前を表示します。このステートメントによって次の出力が生成されるとします。

```
mysql> SHOW BINARY LOGS;
+---------------+-----------+
| Log_name      | File_size |
+---------------+-----------+
| binlog.000130 |     27459 |
| binlog.000131 |     13719 |
| binlog.000132 |     43268 |
+---------------+-----------+
```

この情報により、mysqlbinlog を使用して次のようにバイナリログを現在のディレクトリにバックアップできます (コマンドは 1 行に 1 つずつ入力します)。

- binlog.000130 から binlog.000132 までの静的バックアップを行うには、次のいずれかのコマンドを使用します。

```
mysqlbinlog --read-from-remote-server --host=host_name --raw
  binlog.000130 binlog.000131 binlog.000132

mysqlbinlog --read-from-remote-server --host=host_name --raw
  --to-last-log binlog.000130
```

最初のコマンドはすべてのファイル名を明示的に指定します。2 番目は最初のファイルのみを指名し、--to-last-log を使用して最後まで読み取ります。これらのコマンド間の違いは、mysqlbinlog が binlog.000132 の最後に到達する前にサーバーが binlog.000133 を開いた場合に、最初のコマンドはそれを読み取りませんが、2 番目のコマンドは読み取るという点です。

- mysqlbinlog が binlog.000130 から既存のログファイルのコピーを開始し、その後接続を維持してサーバーが新しいイベントを生成するにつれてそれらをコピーするライブバックアップを行うには:

```
mysqlbinlog --read-from-remote-server --host=host_name --raw
  --stop-never binlog.000130
```

--stop-never を使用すると、--to-last-log オプションは暗示されているため、最後のログファイルまで読み取るようにこのオプションを指定する必要はありません。

出力ファイルの命名

--raw を使用しないと、mysqlbinlog はテキスト出力を生成し、--result-file オプションが与えられた場合は、すべての出力が書き出される単一のファイルの名前を指定します。--raw を使用すると、mysqlbinlog はサーバーから転送される各ログファイルに対して 1 つのバイナリ出力ファイルを書き出します。デフォルトでは、mysqlbinlog はファイルを現在のディレクトリに元のログファイルと同じ名前で書き出します。出力ファイル名を変更するには、--result-file オプションを使用します。--raw も指定されている場合、--result-file オプション値は出力ファイル名を変更するプリフィクスとして処理されます。

サーバーに現在 binlog.000999 以上の名前のバイナリログファイルがあるとします。mysqlbinlog --raw を使用してファイルのバックアップを行う場合、--result-file オプションは次の表に示すように出力ファイル名を生成します。--result-file の値をディレクトリパスで始まるようにすることで、ファイルを特定のディレクトリに書き出すことができます。--result-file の値がディレクトリ名のみで構成されている場合は、その値はパス名区切り文字で終わっていなければなりません。出力ファイルが存在する場合は上書きされます。

--result-file オプション	出力ファイル名
--result-file=x	xbinlog.000999 以上
--result-file=/tmp/	/tmp/binlog.000999 以上
--result-file=/tmp/x	/tmp/xbinlog.000999 以上

例: mysqldump と mysqlbinlog を合わせてバックアップとリストアを行う

次の例は、mysqldump と mysqlbinlog を一緒に使用してサーバーのデータおよびバイナリログのバックアップを取る方法、およびデータの損失が生じた場合にバックアップを使用してサーバーのリストアを行う方法を示す簡単なシナリオを説明しています。例では、サーバーはホスト host_name 上で稼働し、最初のバイナリログファイルの名前は binlog.000999 であるとします。コマンドは 1 行に 1 つずつ入力します。

mysqlbinlog を使用してバイナリログのバックアップを継続的に作成します。

```
mysqlbinlog --read-from-remote-server --host=host_name --raw
 --stop-never binlog.000999
```

mysqldump を使用してサーバーのデータのスナップショットとしてダンプファイルを作成します。--all-databases、--events、および --routines を使用してすべてのデータのバックアップを行い、--master-data=2 を使用して現在のバイナリログ座標をダンプファイルに含めます。

```
mysqldump --host=host_name --all-databases --events --routines --master-data=2> dump_file
```

必要に応じて mysqldump コマンドを定期的に実行して、新しいスナップショットを作成します。

データ損失が生じた場合 (たとえばサーバーのクラッシュなど)、直近のダンプファイルを使用してデータをリストアします。

```
mysql --host=host_name -u root -p < dump_file
```

次にバイナリログバックアップを使用して、ダンプファイル内にリストされている座標よりあとで書き出されたイベントを再実行します。ファイル内の座標が次のようになっているとします。

```
-- CHANGE MASTER TO MASTER_LOG_FILE='binlog.001002', MASTER_LOG_POS=27284;
```

直近にバックアップされたログファイルが binlog.001004 という名前の場合、次のようにログイベントを再実行します。

```
mysqlbinlog --start-position=27284 binlog.001002 binlog.001003 binlog.001004
 | mysql --host=host_name -u root -p
```

リストア操作をより容易に実行するために、または MySQL でリモート root アクセスが許可されない場合は、バックアップファイル (ダンプファイルおよびバイナリログファイル) をサーバーホストにコピーする方が簡単な場合もあります。

4.6.8.4 mysqlbinlog サーバー ID の指定

mysqlbinlog は、--read-from-remote-server オプションで呼び出された場合、MySQL サーバーに接続し、自分を証明するためにサーバー ID を指定し、サーバーからバイナリログファイルを要求します。mysqlbinlog を使用してサーバーからログファイルを要求するには、いくつかの方法があります。

- 一連のファイルを名前を明示的に示して指定します。mysqlbinlog は、各ファイルに接続して Binlog dump コマンドを発行します。サーバーはファイルを送信して切断します。ファイルごとに 1 つの接続があります。
- 開始ファイルと --to-last-log を指定します。mysqlbinlog はすべてのファイルに接続して Binlog dump コマンドを発行します。サーバーはすべてのファイルを送信して切断します。
- 開始ファイルと --stop-never を指定します (これは暗黙的に --to-last-log を示します)。mysqlbinlog はすべてのファイルに対して接続して Binlog dump コマンドを発行します。サーバーはすべてのファイルを送信しますが、最後のファイルの送信後に切断しません。

--read-from-remote-server のみを使用すると、mysqlbinlog は 0 をサーバー ID として使用します。これは、要求された最後のログファイルの送信後に切断することをサーバーに指示します。

--read-from-remote-server および --stop-never を使用すると、mysqlbinlog はゼロ以外のサーバー ID を使用して接続するため、サーバーは最後のログファイルの送信後に切断しません。デフォルトではサーバー ID は 65535 ですが、--stop-never-slave-server-id で変更できます。

したがって、ファイルを要求する最初の 2 つの方法では、mysqlbinlog がサーバー ID の 0 を指定するためサーバーは切断します。--stop-never が指定された場合は、mysqlbinlog はゼロ以外のサーバー ID を指定するため切断しません。

4.6.9 mysqldumpslow — スロークエリーログファイルの要約

MySQL スロークエリーログは、実行に長い時間のかかるクエリーに関する情報を含みます (セクション5.2.5「スロークエリーログ」を参照してください)。mysqldumpslow は MySQL スロークエリーログファイルを解析して内容のサマリーを出力します。

通常、mysqldumpslow は数字の特定の値および文字列データ値以外が同様のクエリーをグループ化します。サマリーの出力を表示する際、これらの値を N および 'S' に「抽象化」します。-a オプションおよび -n オプションを使用して、値の抽象化の動作を変更できます。

mysqldumpslow は次のように起動します。

```
shell> mysqldumpslow [options] [log_file ...]
```

mysqldumpslow は次のオプションをサポートします。

表 4.16 mysqldumpslow のオプション

オプション名	説明
-a	すべての数字を N に、文字列を S に抽象化しない
-n	少なくとも指定された桁数の数字を抽象化
--debug	デバッグ情報を書き込み
-g	パターンに一致するステートメントのみを考慮
--help	ヘルプメッセージを表示して終了
-h	ログファイル名内のサーバーのホスト名
-i	サーバーインスタンスの名前
-l	合計時間からロック時間を減算しない
-r	ソート順序を逆転
-s	出力のソート方法
-t	最初から指定された数だけのクエリーのみ表示
--verbose	冗長モード

- --help

  ヘルプメッセージを表示して終了します。

- -a

  すべての数字を N に、文字列を 'S' に抽象化しません。

- --debug, -d

  デバッグモードで実行します。

- -g pattern

  (grep 形式の) パターンに一致するクエリーのみを考慮します。

- -h host_name

  *-slow.log ファイル名の MySQL サーバーのホスト名。値にはワイルドカードを含めることができます。デフォルトは * (すべて一致) です。

- -i name

  サーバーインスタンス名 (mysql.server 起動スクリプトを使用している場合)。

- -l

  合計時間からロック時間を減算しません。

- -n N

  少なくとも N 桁の数字を名前に抽象化します。

- -r

  ソート順序を逆転します。

- -s sort_type

  出力のソート方法。sort_type の値は次のリストから選択するようにしてください。

  - t、at: クエリー時間または平均クエリー時間でソート
  - l、al: ロック時間または平均ロック時間でソート

  - r、ar: 送信行数または平均送信行数でソート
  - c: カウントでソート

  デフォルトでは、mysqldumpslow は平均クエリー時間でソートします (-s at と同等)。

- -t N

  出力内の最初の N 個のクエリーのみを表示します。

- --verbose, -v

  冗長モード。プログラムの動作についてより多くの情報を出力します。

使用例:

```
shell> mysqldumpslow

Reading mysql slow query log from /usr/local/mysql/data/mysqld51-apple-slow.log
Count: 1  Time=4.32s (4s)  Lock=0.00s (0s)  Rows=0.0 (0), root[root]@localhost
 insert into t2 select * from t1

Count: 3  Time=2.53s (7s)  Lock=0.00s (0s)  Rows=0.0 (0), root[root]@localhost
 insert into t2 select * from t1 limit N

Count: 3  Time=2.13s (6s)  Lock=0.00s (0s)  Rows=0.0 (0), root[root]@localhost
 insert into t1 select * from t1
```

4.6.10 mysqlhotcopy — データベースバックアッププログラム

> **注記**
>
> このユーティリティーは MySQL 5.6.20 で非推奨で、MySQL 5.7 で削除されます。

mysqlhotcopy はもともと Tim Bunce によって書かれ、提供された Perl スクリプトです。FLUSH TABLES、LOCK TABLES、および cp または scp を使用してデータベースのバックアップを作成します。データベースまたは単一のテーブルのバックアップを作成するための高速な方法ですが、データベースディレクトリが置かれているのと同じマシンでしか実行できません。mysqlhotcopy は MyISAM テーブルおよびARCHIVE テーブルのみに機能します。Unix で実行されます。

mysqlhotcopy を使用するには、バックアップを行うテーブルのファイルへの読み取りアクセス、これらのテーブルの SELECT 権限、RELOAD 権限 (FLUSH TABLES を実行できるように)、および LOCK TABLES 権限 (テーブルをロックできるように) を持っていなければなりません。

```
shell> mysqlhotcopy db_name [/path/to/new_directory]
```

```
shell> mysqlhotcopy db_name_1 ... db_name_n /path/to/new_directory
```

指定されたデータベース内で正規表現と一致するテーブルをバックアップします。

```
shell> mysqlhotcopy db_name./regex/
```

テーブル名の正規表現は、チルダ (「~」) をプリフィクスとして使用することで否定できます。

```
shell> mysqlhotcopy db_name./~regex/
```

mysqlhotcopy は次のオプションをサポートします。これらはコマンド行またはオプションファイルの [mysqlhotcopy] グループおよび [client] グループで指定できます。MySQL プログラムによって使用されるオプションファイルの詳細については、セクション4.2.6「オプションファイルの使用」を参照してください。

表 4.17 mysqlhotcopy のオプション

オプション名	説明
--addtodest	ターゲットディレクトリの名前変更をせず (存在する場合)、単にファイルを追加
--allowold	ターゲットが存在する場合中止せず、_old サフィクスを追加することで名前変更
--checkpoint	チェックポイントエントリを挿入
--chroot	mysqld が動作する chroot ジェイルのベースディレクトリ

オプション名	説明
--debug	デバッグのログを書き込み
--dryrun	アクションを実行せずにレポート
--flushlog	すべてのテーブルがロックされたあとにログをフラッシュ
--help	ヘルプメッセージを表示して終了
--host	指定されたホスト上で MySQL サーバーに接続
--keepold	終了後に以前の (名前変更された) ターゲットを消去しない
--method	ファイルをコピーする方法
--noindices	フルインデックスファイルをバックアップに含めない
--old_server	FLUSH TABLES tbl_list WITH READ LOCK をサポートしないサーバーに接続しない
--password	サーバーに接続する際に使用するパスワード
--port	接続に使用する TCP/IP ポート番号
--quiet	エラー発生時以外サイレント
--regexp	与えられた正規表現と一致する名前を持つデータベースをすべてコピー
--resetmaster	テーブルをすべてロックしたあとにバイナリログをリセット
--resetslave	テーブルをすべてロックしたあとに master.info ファイルをリセット
--socket	ローカルホストへの接続で、使用する Unix ソケットファイル
--tmpdir	一時ディレクトリ
--user	サーバーへの接続時に使用する MySQL ユーザー名

- --help, -?

  ヘルプメッセージを表示して終了します。

- --addtodest

  ターゲットディレクトリの名前変更をせず (存在する場合)、単にファイルを追加します。

- --allowold

  ターゲットが存在する場合、中止せず _old サフィクスを追加することで名前変更します。

- --checkpoint=db_name.tbl_name

  指定されたデータベース db_name とテーブル tbl_name にチェックポイントエントリを挿入します。

- --chroot=path

  mysqld が稼働している chroot jail のベースディレクトリ。path 値は mysqld に与えられる --chroot オプションと一致するようにしてください。

- --debug

  デバッグ出力を有効化します。

- --dryrun, -n

  アクションを実行せずにレポートします。

- --flushlog

  すべてのテーブルがロックされたあとにログをフラッシュします。

- --host=host_name, -h host_name

  ローカルサーバーへの TCP/IP 接続を行うために使用するローカルホストのホスト名。デフォルトでは、Unix ソケットファイルを使用して localhost に接続します。

- --keepold

終了後に以前の (名前変更された) ターゲットを削除しません。

- --method=command

  ファイルのコピー方法 (cp または scp)。デフォルトは cp です。

- --noindices

  MyISAM テーブルのフルインデックスファイルをバックアップに含めません。これによりバックアップを小さく、高速にできます。リロードされたテーブルのインデックスはあとで myisamchk -rq を使用して再構築できます。

- --password=password, -ppassword

  サーバーに接続する際に使用するパスワードです。ほかの MySQL プログラムとは異なり、このオプションではパスワード値はオプションではありません。

  コマンド行でパスワードを指定することはセキュアでないとみなすべきです。セクション6.1.2.1「パスワードセキュリティーのためのエンドユーザーガイドライン」を参照してください。オプションファイルを使用すれば、コマンド行でパスワードを指定することを回避できます。

- --port=port_num, -P port_num

  ローカルサーバーへの接続時に使用する TCP/IP ポート番号。

- --old_server

  MySQL 5.6 では、mysqlhotcopy は FLUSH TABLES tbl_list WITH READ LOCK を使用してテーブルのフラッシュおよびロックを実行します。サーバーが、このステートメントが導入された 5.5.3 より古い場合は、--old_server オプションを使用します。

- --quiet, -q

  エラー発生時以外サイレントにします。

- --record_log_pos=db_name.tbl_name

  指定されたデータベース db_name およびテーブル tbl_name に、マスターとスレーブのステータスを記録します。

- --regexp=expr

  与えられた正規表現と一致する名前を持つデータベースをすべてコピーします。

- --resetmaster

  テーブルをすべてロックしたあとにバイナリログをリセットします。

- --resetslave

  テーブルをすべてロックしたあとに、マスター情報リポジトリファイルまたはテーブルをリセットします。

- --socket=path, -S path

  localhost への接続に使用される Unix ソケットファイル。

- --suffix=str

  コピーされたデータベースの名前に使用するサフィクス。

- --tmpdir=path

  一時ディレクトリ。デフォルトは /tmp です。

- --user=user_name, -u user_name

  サーバーへの接続時に使用する MySQL ユーザー名。

--checkpoint オプションおよび --record_log_pos オプションに必要なテーブルの構造に関する情報を含めて、追加の mysqlhotcopy ドキュメントに関しては、perldoc を使用してください。

```
shell> perldoc mysqlhotcopy
```

4.6.11 mysql_convert_table_format — 指定されたストレージエンジンを使用するためのテーブルの変換

> **注記**
>
> このユーティリティーは MySQL 5.6.17 で非推奨で、MySQL 5.7 で削除されます。

mysql_convert_table_format は、データベース内のテーブルを、特定のストレージエンジン (デフォルトでは MyISAM) を使用するように変換します。mysql_convert_table_format は Perl で作成されており、DBI および DBD::mysql Perl モジュールがインストールされている必要があります (セクション2.13「Perl のインストールに関する注釈」を参照してください)。

mysql_convert_table_format は次のように起動します。

```
shell> mysql_convert_table_format [options]db_name
```

db_name 引数は変換されるテーブルを含むデータベースを示します。

mysql_convert_table_format は次のリストで説明するオプションをサポートします。

- --help

  ヘルプメッセージを表示して終了します。

- --force

  エラーが発生しても続行します。

- --host=host_name

  指定されたホストの MySQL サーバーに接続します。

- --password=password

  サーバーに接続する際に使用するパスワードです。ほかの MySQL プログラムとは異なり、このオプションではパスワード値はオプションではありません。

  コマンド行でパスワードを指定することはセキュアでないとみなすべきです。セクション6.1.2.1「パスワードセキュリティーのためのエンドユーザーガイドライン」を参照してください。オプションファイルを使用すれば、コマンド行でパスワードを指定することを回避できます。

- --port=port_num

  接続に使用する TCP/IP ポート番号。

- --socket=path

  localhost への接続で使用する Unix ソケットファイル。

- --type=engine_name

  テーブルを変換して使用するようにするべきストレージエンジンを指定します。このオプションが指定されていない場合、デフォルトは MyISAM です。

- --user=user_name

  サーバーへの接続時に使用する MySQL ユーザー名。

- --verbose

  冗長モード。プログラムの動作についてより多くの情報を出力します。

- --version

  バージョン情報を表示して終了します。

4.6.12 mysql_find_rows — ファイルからの SQL ステートメントの抽出

> **注記**
>
> このユーティリティーは MySQL 5.6.17 で非推奨で、MySQL 5.7 で削除されます。

mysql_find_rows は、SQL ステートメントを含むファイルを読み取り、指定された正規表現に一致するステートメント、または USE db_name か SET ステートメントを含むステートメントを抽出します。このユーティリティーは、ステートメントがセミコロン (;) 文字で終了することを想定しています。

mysql_find_rows は次のように起動します。

```
shell> mysql_find_rows [options] [file_name ...]
```

各file_name 引数は SQL ステートメントを含むファイル名であるべきです。ファイル名が指定されない場合、mysql_find_rows は標準入力を読み取ります。

例:

```
mysql_find_rows --regexp=problem_table --rows=20 < update.log
mysql_find_rows --regexp=problem_table  update-log.1 update-log.2
```

mysql_find_rows は次のオプションをサポートします。

- --help, --Information

  ヘルプメッセージを表示して終了します。

- --regexp=pattern

  パターンに一致するクエリーを表示します。

- --rows=N

  N 個のクエリーを表示したあと終了します。

- --skip-use-db

  出力に USE db_name ステートメントを含めません。

- --start_row=N

  この行から出力を開始します。

4.6.13 mysql_fix_extensions — テーブルファイル名の拡張子の正規化

> **注記**
>
> このユーティリティーは MySQL 5.6.17 で非推奨で、MySQL 5.7 で削除されます。

mysql_fix_extensions は、MyISAM (または ISAM) テーブルファイルの拡張子を標準形式に変換します。.frm、.myd、.myi、.isd、および .ism (大文字小文字を区別しません) に一致する拡張子を持つファイルを検索し、それぞれ .frm、.MYD、.MYI、.ISD、および .ISM に名前変更します。これは、(Windows などの) ファイル名の大文字と小文字を区別しないシステムから、ファイル名の大文字と小文字を区別するシステムにファイルを転送したあとで便利です。

mysql_fix_extensions は次のように起動します。ここで data_dir は MySQL データディレクトリへのパス名です。

```
shell> mysql_fix_extensions data_dir
```

4.6.14 mysql_setpermission — 付与テーブルに許可をインタラクティブに設定

> **注記**
>
> このユーティリティーは MySQL 5.6.17 で非推奨で、MySQL 5.7 で削除されます。

mysql_setpermission はもともと Luuk de Boer によって作成および提供された Perl スクリプトです。それは MySQL 付与テーブルに許可をインタラクティブに設定します。mysql_setpermission は Perl で記述され、DBI および DBD::mysql Perl モジュールがインストールされていることを必要とします (セクション2.13「Perl のインストールに関する注釈」を参照してください)。

mysql_setpermission は次のように起動します。

```
shell> mysql_setpermission [options]
```

options は、ヘルプメッセージを表示するための --help、または MySQL サーバーへの接続方法を示すオプションにしてください。付与テーブル内の既存の許可を変更する場合は、接続するときに使用されるアカウントによって、保有している許可が決定されます。

mysql_setpermissions は、ユーザーのホームディレクトリに .my.cnf ファイルが存在する場合は、そこから [client] グループおよび [perl] グループのオプションも読み取ります。

mysql_setpermission は次のオプションをサポートします。

- --help

  ヘルプメッセージを表示して終了します。

- --host=host_name

  指定されたホストの MySQL サーバーに接続します。

- --password=password

  サーバーに接続する際に使用するパスワードです。ほかの MySQL プログラムとは異なり、このオプションではパスワード値はオプションではありません。

  コマンド行でパスワードを指定することはセキュアでないとみなすべきです。セクション6.1.2.1「パスワードセキュリティーのためのエンドユーザーガイドライン」を参照してください。オプションファイルを使用すれば、コマンド行でパスワードを指定することを回避できます。

- --port=port_num

  接続に使用する TCP/IP ポート番号。

- --socket=path

  localhost への接続で使用する Unix ソケットファイル。

- --user=user_name

  サーバーへの接続時に使用する MySQL ユーザー名。

4.6.15 mysql_waitpid — プロセスを強制終了して終了を待機

mysql_waitpid はプロセスに対して終了するように信号を送信して、そのプロセスの終了を待機します。kill() システムコールおよび Unix シグナルを使用するため、Unix および Unix 類似のシステムで動作します。

このプログラムは MySQL 5.6.19 で非推奨となり、MySQL 5.7 で削除されています。

mysql_waitpid は次のように起動します。

```
shell> mysql_waitpid [options] pid wait_time
```

mysql_waitpid は pid で特定されるプロセスにシグナル 0 を送信し、プロセスの終了を最大 wait_time 秒まで待機します。pid および wait_time は正の整数でなければなりません。

プロセスが待機時間内に終了するか、プロセスが存在しない場合は、mysql_waitpid は 0 を返します。そうでない場合は 1 を返します。

kill() システムコールでシグナル 0 を処理できない場合、mysql_waitpid() は代わりにシグナル 1 を使用します。

mysql_waitpid は次のオプションをサポートします。

- --help, -?, -I

  ヘルプメッセージを表示して終了します。

- --verbose, -v

  冗長モード。シグナル 0 を使用できず、代わりにシグナル 1 を使用した場合に警告を表示します。

- --version, -V

バージョン情報を表示して終了します。

4.6.16 mysql_zap — パターンに一致するプロセスを強制終了

mysql_zap は、パターンに一致するプロセスを強制終了します。ps コマンドおよび Unix シグナルを使用するため、Unix および Unix 類似のシステムで動作します。

このプログラムは MySQL 5.6.19 で非推奨となり、MySQL 5.7 で削除されています。

mysql_zap は次のように起動します。

```
shell> mysql_zap [-signal] [-?Ift] pattern
```

プロセスは ps コマンドからの出力行にパターンが含まれている場合に一致します。デフォルトでは、mysql_zap は各プロセスに対して確認を要求します。プロセスを強制終了するには y、mysql_zap を終了するには q で応答します。その他の応答に対しては、mysql_zap はプロセスの強制終了を試みません。

-signal オプションが指定された場合は、各プロセスに送信するシグナルの名前または数字を指定します。そうでない場合は、mysql_zap はまず TERM (シグナル 15)、次に KILL (シグナル 9)を試します。

mysql_zap は次の追加オプションをサポートします。

- --help, -?, -I

  ヘルプメッセージを表示して終了します。

- -f

  強制モード。mysql_zap は各プロセスを確認なしで強制終了しようとします。

- -t

  テストモード。各プロセスの情報を表示しますが強制終了はしません。

4.7 MySQL プログラム開発ユーティリティー

このセクションでは、MySQL プログラムを開発する際に有用であると思われるいくつかのユーティリティーについて説明します。

シェルスクリプトで my_print_defaults プログラムを使用して、オプションファイルを解析し、所定のプログラムがどのオプションを使用するかを表示できます。次の例は、[client] グループおよび [mysql] グループで検出されたオプションを表示するように指示された場合に、my_print_defaults が生成するであろう出力を示しています。

```
shell> my_print_defaults client mysql
--port=3306
--socket=/tmp/mysql.sock
--no-auto-rehash
```

開発者へのメモ: オプションファイルの処理は、単にコマンド行引数の前に適切なグループ内のすべてのオプションを処理することによって、C クライアントライブラリに実装されています。これは、複数回指定されたオプションの最後のインスタンスを使用するプログラムではうまく機能します。複数指定されたオプションをこの方法で処理するが、オプションファイルを読み取らない C プログラムまたは C++ プログラムがある場合は、その機能を与えるために 2 行のみを追加する必要があります。標準 MySQL クライアントの任意のソースコードをチェックして、その方法を確認します。

MySQL へのほかのいくつかの言語インタフェースは C クライアントライブラリに基づいており、そのうちのいくつかはオプションファイルの内容にアクセスする方法を提供します。これらには、Perl および Python が含まれます。詳細は、使用するインタフェースのドキュメントを参照してください。

4.7.1 msql2mysql — mSQL プログラムを MySQL で使用するために変換

> **注記**
>
> このユーティリティーは MySQL 5.6.17 で非推奨で、MySQL 5.7 で削除されます。

MySQL C API は最初、mSQL データベースシステムのものに非常に類似するように開発されました。このため mSQL プログラムは、C API 関数の名前を変えることによって MySQL と一緒に使用するために比較的容易に変換できることが多くあります。

msql2mysql ユーティリティーは、mSQL C API 関数呼び出しを MySQL の同等の関数呼び出しに変換します。msql2mysql は入力ファイルを直接変換するため、変換する前にオリジナルのコピーを作成してください。たとえば、msql2mysql は次のように使用します。

```
shell> cp client-prog.c client-prog.c.orig
shell> msql2mysql client-prog.c
client-prog.c converted
```

そのあと client-prog.c を調べ、必要に応じて変換後の修正を行なってください。

msql2mysql は関数名の置換を実行するために replace ユーティリティーを使います。セクション4.8.2「replace — 文字列置換ユーティリティー」を参照してください。

4.7.2 mysql_config — クライアントのコンパイル用オプションの表示

mysql_config は、MySQ クライアントをコンパイルして MySQL に接続するのに有用な情報を提供します。シェルスクリプトであるため、Unix および Unix 類似システムでのみ使用可能です。

mysql_config は次のオプションをサポートします。

- --cflags

  libmysqlclient ライブラリのコンパイルに使用される、インクルードファイルを検索するための C コンパイラフラグおよび重要なコンパイラフラグおよび定義。返されるオプションは、ライブラリが作成されたときに使用された特定のコンパイラに結びついており、ユーザー自身のコンパイラ設定ではクラッシュする場合があります。インクルードパスのみを含むより移植性の高いオプションには、--include を使用します。

- --cxxflags

  --cflags と同様ですが、C++ コンパイラフラグ用です。このオプションは MySQL 5.6.4 で追加されました。

- --include

  MySQL インクルードファイルを検出するためのコンパイラオプション。

- --libmysqld-libs, --embedded

  MySQL 組み込みサーバーにリンクするために必要なライブラリおよびオプション。

- --libs

  MySQL クライアントライブラリにリンクするために必要なライブラリおよびオプション。

- --libs_r

  スレッドセーフな MySQL クライアントライブラリにリンクするために必要なライブラリおよびオプション。In MySQL 5.6 ではすべてのクライアントライブラリはスレッドセーフであるため、このオプションを使用する必要はありません。すべての場合に --libs オプションを使用できます。

- --plugindir

  MySQL の構成時に定義される、デフォルトのプラグインディレクトリパス名。

- --port

  MySQL の構成時に定義される、デフォルトの TCP/IP ポート番号。

- --socket

  MySQL の構成時に定義される、デフォルトの Unix ソケットファイル。

- --version

  MySQL 配布のバージョン番号。

mysql_config をオプションなしで呼び出すと、サポートされるすべてのオプションおよびそれらの値のリストが表示されます。

```
shell> mysql_config
Usage: /usr/local/mysql/bin/mysql_config [options]
```

```
Options:
 --cflags         [-I/usr/local/mysql/include/mysql -mcpu=pentiumpro]
 --include        [-I/usr/local/mysql/include/mysql]
 --libs          [-L/usr/local/mysql/lib/mysql -lmysqlclient
                 -lpthread -lm -lrt -lssl -lcrypto -ldl]
 --libs_r         [-L/usr/local/mysql/lib/mysql -lmysqlclient_r
                 -lpthread -lm -lrt -lssl -lcrypto -ldl]
 --socket         [/tmp/mysql.sock]
 --port          [3306]
 --version        [5.6.11]
 --libmysqld-libs [-L/usr/local/mysql/lib/mysql -lmysqld
                 -lpthread -lm -lrt -lssl -lcrypto -ldl -lcrypt]
```

コマンド行で、逆引用符を使用して mysql_config を使用して、特定のオプションに対して生成される出力を含めることができます。たとえば。MySQL クライアントプログラムのコンパイルおよびリンクを行うには、mysql_config を次のように使用します。

```
shell> gcc -c `mysql_config --cflags` progname.c
shell> gcc -o progname progname.o `mysql_config --libs`
```

4.7.3 my_print_defaults — オプションファイルからのオプションの表示

my_print_defaults はオプションファイルのオプショングループ内にあるオプションを表示します。出力は指定されたオプショングループを読み取るプログラムによって使用されるオプションを示します。たとえば、mysqlcheck プログラムは [mysqlcheck] および [client] のオプショングループを読み取ります。標準オプションファイル内のこれらのグループに存在するオプションを確認するには、my_print_defaults を次のように起動します。

```
shell> my_print_defaults mysqlcheck client
--user=myusername
--password=secret
--host=localhost
```

出力には、コマンド行で指定される形式のオプションが 1 行につき 1 つ含まれます。

my_print_defaults は次のオプションをサポートします。

- --help, -?

  ヘルプメッセージを表示して終了します。

- --config-file=file_name, --defaults-file=file_name, -c file_name

  指定されたオプションファイルのみを読み取ります。

- --debug=debug_options, -# debug_options

  デバッグのログを書き込みます。一般的な debug_options 文字列は d:t:o,file_name です。デフォルトは d:t:o,/tmp/my_print_defaults.trace です。

- --defaults-extra-file=file_name, --extra-file=file_name, -e file_name

  このオプションファイルは、グローバルオプションファイルのあとに読み取りますが、(UNIX では) ユーザーオプションファイルの前に読み取るようにしてください。

- --defaults-group-suffix=suffix, -g suffix

  コマンド行で指名されたグループのほかに、指定されたサフィクスのあるグループを読み取ります。

- --login-path=name, -l name

  指名されたログインパスから .mylogin.cnf ログインファイルのオプションを読み取ります。「ログインパス」は、host、user、および password という限定されたオプションのセットのみを許可するオプショングループです。ログインパスは、サーバーホストおよびそのサーバーで認証するための認証情報を示す値のセットであると考えてください。ログインパスファイルを作成するには、mysql_config_editor ユーティリティーを使用します。セクション4.6.6「mysql_config_editor — MySQL 構成ユーティリティー」を参照してください。このオプションは MySQL 5.6.6 で追加されました。

- --no-defaults, -n

  空の文字列を返します。

- --verbose, -v

  冗長モード。プログラムの動作についてより多くの情報を出力します。

- --version, -V

  バージョン情報を表示して終了します。

4.7.4 resolve_stack_dump — 数値スタックトレースダンプをシンボルに解決

resolve_stack_dump は、数値スタックトレースダンプをシンボルに解決します。

resolve_stack_dump は次のように起動します。

```
shell> resolve_stack_dump [options] symbols_file [numeric_dump_file]
```

シンボルファイルは nm --numeric-sort mysqld コマンドからの出力を含むようにしてください。数値ダンプファイルは mysqld からの数値スタックトラックを含むようにしてください。コマンド行で数値ダンプファイルが指名されていない場合は、スタックトレースは標準入力から読み取られます。

resolve_stack_dump は次のオプションをサポートします。

- --help, -h

  ヘルプメッセージを表示して終了します。

- --numeric-dump-file=file_name, -n file_name

  指定されたファイルからスタックトレースを読み取ります。

- --symbols-file=file_name, -s file_name

  指定されたシンボルファイルを使用します。

- --version, -V

  バージョン情報を表示して終了します。

詳細はセクション24.4.1.5「スタックトレースの使用」を参照してください。

4.8 その他のプログラム

4.8.1 perror — エラーコードの説明

ほとんどのシステムエラーでは、MySQL は内部テキストメッセージに加えて、次のスタイルのいずれかでシステムエラーコードを表示します。

```
message ... (errno: #)
message ... (Errcode: #)
```

システムのドキュメントを確認するか perror ユーティリティーを使用することで、エラーコードの意味がわかります。

perror は、システムエラーコードまたはストレージエンジン (テーブルハンドラ) エラーコードの説明を出力します。

perror は次のように起動します。

```
shell> perror [options] errorcode ...
```

例:

```
shell> perror 13 64
OS error code  13:  Permission denied
OS error code  64:  Machine is not on the network
```

MySQL Cluster エラーコードのエラーメッセージを取得するには、perror を --ndb オプションで呼び出します。

```
shell> perror --ndb errorcode
```

システムエラーメッセージの意味はユーザーのオペレーティングシステムによって異なる場合があります。異なるオペレーティングシステムでは、所定のエラーコードの意味が異なる場合があります。

perror は次のオプションをサポートします。

- --help, --info, -I, -?

  ヘルプメッセージを表示して終了します。

- --ndb

  MySQL Cluster エラーコードのエラーメッセージを出力します。

- --silent, -s

  サイレントモード。エラーメッセージのみ出力します。

- --verbose, -v

  冗長モード。エラーコードおよびメッセージを出力します。これはデフォルトの動作です。

- --version, -V

  バージョン情報を表示して終了します。

4.8.2 replace — 文字列置換ユーティリティー

replace ユーティリティープログラムは、ファイルまたは標準入力の文字列を変更します。

replace は次のいずれかの方法で起動します。

```
shell> replace from to [from to] ... -- file_name [file_name] ...
shell> replace from to [from to] ... < file_name
```

from は検索する文字列を表し、to は変更後の文字列を表しています。文字列のペアは 1 つまたは複数指定できます。

-- オプションを使用して、文字列置換リストが終了しファイル名が開始する位置を示します。この場合、コマンド行で指定されたファイルは直接変更されるため、変換する前にオリジナルのコピーを作成するとよいでしょう。replace は、どの入力ファイルが実際に変更されたを示すメッセージを出力します。

-- オプションが与えられていない場合、replace は標準入力を読み取って標準出力に書き出します。

replace は有限状態マシンを使用して長い文字列から先に一致します。これは文字列の交換に使用できます。たとえば次のコマンドは、指定されたファイル file1 および file2 内の a と b を交換します。

```
shell> replace a b b a -- file1 file2 ...
```

replace プログラムは msql2mysql に使用されています。セクション4.7.1「msql2mysql — mSQL プログラムを MySQL で使用するために変換」を参照してください。

replace は次のオプションをサポートします。

- -?、-I

  ヘルプメッセージを表示して終了します。

- -#debug_options

  デバッグを有効にします。

- -s

  サイレントモード。プログラムの動作についてより少ない情報を出力します。

- -v

  冗長モード。プログラムの動作についてより多くの情報を出力します。

- -V

バージョン情報を表示して終了します。

4.8.3 resolveip — ホスト名と IP アドレスの解決

resolveip ユーティリティーはホスト名を IP アドレスに、IP アドレスをホスト名に解決します。

resolveip は次のように起動します。

```
shell> resolveip [options] {host_name|ip-addr} ...
```

resolveip は次のオプションをサポートします。

- --help, --info, -?, -I

  ヘルプメッセージを表示して終了します。

- --silent, -s

  サイレントモード。出力の生成を少なくします。

- --version, -V

  バージョン情報を表示して終了します。

第 5 章 MySQL サーバーの管理

目次

5.1 MySQL Server 393
5.1.1 サーバーオプションおよび変数リファレンス 394
5.1.2 サーバー構成のデフォルト値 422
5.1.3 サーバーコマンドオプション 424
5.1.4 サーバーシステム変数 452
5.1.5 システム変数の使用 548
5.1.6 サーバーステータス変数 560
5.1.7 サーバー SQL モード 585
5.1.8 サーバープラグイン 593
5.1.9 IPv6 サポート 596
5.1.10 サーバー側のヘルプ 600
5.1.11 シグナルへのサーバー応答 600
5.1.12 シャットダウンプロセス 601
5.2 MySQL Server ログ 602
5.2.1 一般クエリーログおよびスロークエリーログの出力先の選択 603
5.2.2 エラーログ 605
5.2.3 一般クエリーログ 606
5.2.4 バイナリログ 607
5.2.5 スロークエリーログ 617
5.2.6 DDL ログ 618
5.2.7 サーバーログの保守 619
5.3 1 つのマシン上での複数の MySQL インスタンスの実行 620
5.3.1 複数のデータディレクトリのセットアップ 621
5.3.2 Windows 上での複数の MySQL インスタンスの実行 622
5.3.3 Unix 上での複数の MySQL インスタンスの実行 624
5.3.4 複数サーバー環境でのクライアントプログラムの使用 625
5.4 DTrace を使用した mysqld のトレース 626
5.4.1 mysqld DTrace プローブリファレンス 627

MySQL サーバー (mysqld) は、MySQL インストールの中核を担うメインプログラムです。この章では、MySQL サーバーの概要を説明し、一般的なサーバー管理を取り上げます。

- サーバーの構成。
- サーバーログファイル。
- 単一マシン上の複数のサーバーの管理。

管理に関するトピックの追加情報は、次も参照してください。

- 第6章「セキュリティー」
- セクション14.3「InnoDB の構成」およびセクション14.4「InnoDB の管理」
- 第7章「バックアップとリカバリ」
- 第17章「レプリケーション」
- 第19章「パーティション化」

5.1 MySQL Server

mysqld は MySQL Server です。次の説明では、これらの MySQL Server の構成トピックについて扱います。

- サーバーがサポートする起動オプション。コマンド行、構成ファイル、あるいは両方でこれらのオプションを指定できます。
- サーバーシステム変数。これらの変数は、起動オプションの現在の状態および値を反映したもので、変数の一部はサーバーの実行中に変更できます。
- サーバーステータス変数。これらの変数には、ランタイム動作についてのカウンタおよび統計が含まれています。

- サーバー SQL モードの設定方法。この設定は、たとえば別のデータベースシステムからのコードとの互換性を保ったり、特定の状況についてのエラー処理を制御したりするために、SQL の構文およびセマンティクスの特定の側面を変更します。
- サーバーのシャットダウンプロセス。テーブルのタイプ (トランザクションまたは非トランザクション) やレプリケーションを使用するかどうかに応じて、パフォーマンスおよび信頼性についての考慮事項があります。

> **注記**
>
> すべてのストレージエンジンが、すべての MySQL Server のバイナリおよび構成によってサポートされているわけではありません。MySQL Server のインストール環境がサポートしているストレージエンジンを判別するための方法を見つけるには、セクション 13.7.5.17「SHOW ENGINES 構文」を参照してください。

5.1.1 サーバーオプションおよび変数リファレンス

次の表には、mysqld 内で適用可能なすべてのコマンド行オプション、サーバー変数、およびステータス変数のリストが提供されています。

この表には、コマンド行オプション (コマンド行)、構成ファイル内で有効なオプション (オプションファイル)、サーバーシステム変数 (システム変数)、およびステータス変数 (ステータス変数) が 1 つの統一リストに記載され、さらに各オプションまたは変数が有効かどうかが示されています。コマンド行またはオプションファイルに設定されるサーバーオプションが、対応するサーバーシステム変数またはステータス変数の名前と異なる場合、対応するオプションのすぐ下に変数名が記載されています。ステータス変数の場合、変数のスコープ (スコープ) は、グローバル、セッション、または両方として示されています。オプションおよび変数の設定と使用の詳細については、対応するセクションを参照してください。項目に関する追加情報への直接リンクが利用できる場合もあります。

MySQL クラスタに固有となるこの表のバージョンについては、セクション18.3.4.1「MySQL Cluster の mysqld オプションおよび変数のリファレンス」を参照してください。

表 5.1 Option/Variable Summary

名前	コマンド行	オプションファイル	システム変数	ステータス変数	変数スコープ	動的
abort-slave-event-count	はい	はい				
Aborted_clients				はい	グローバル	いいえ
Aborted_connects				はい	グローバル	いいえ
allow-suspicious-udfs	はい	はい				
ansi	はい	はい				
audit-log	はい	はい				
audit_log_buffer_size	はい	はい	はい		グローバル	いいえ
audit_log_connection_policy	はい	はい	はい		グローバル	はい
audit_log_current_session			はい		両方	いいえ
Audit_log_current_size				はい	グローバル	いいえ
Audit_log_event_max_drop_size				はい	グローバル	いいえ
Audit_log_events				はい	グローバル	いいえ
Audit_log_events_filtered				はい	グローバル	いいえ
Audit_log_events_lost				はい	グローバル	いいえ
Audit_log_events_written				はい	グローバル	いいえ
audit_log_exclude_accounts	はい	はい	はい		グローバル	はい
audit_log_file	はい	はい	はい		グローバル	いいえ
audit_log_flush			はい		グローバル	はい
audit_log_format	はい	はい	はい		グローバル	いいえ
audit_log_include_accounts	はい	はい	はい		グローバル	はい
audit_log_policy	はい	はい	はい		グローバル	異なる
audit_log_rotate_on_size	はい	はい	はい		グローバル	はい

名前	コマンド行	オプションファイル	システム変数	ステータス変数	変数スコープ	動的
audit_log_statement_policy	はい	はい	はい		グローバル	はい
audit_log_strategy	はい	はい	はい		グローバル	いい
Audit_log_total_size				はい	グローバル	いい
Audit_log_write_waits				はい	グローバル	いい
authentication_windows_log_level	はい	はい				
authentication_windows_use_principal_name	はい	はい				
auto_increment_increment			はい		両方	はい
auto_increment_offset			はい		両方	はい
autocommit	はい	はい	はい		両方	はい
automatic_sp_privileges			はい		グローバル	はい
back_log			はい		グローバル	いい
basedir	はい	はい	はい		グローバル	いい
big_tables	はい	はい	はい		両方	はい
bind_address	はい	はい	はい		グローバル	いい
Binlog_cache_disk_use				はい	グローバル	いい
binlog_cache_size	はい	はい	はい		グローバル	はい
Binlog_cache_use				はい	グローバル	いい
binlog-checksum	はい	はい				
binlog_checksum			はい		グローバル	はい
binlog_direct_non_transactional_updates	はい	はい	はい		両方	はい
binlog-do-db	はい	はい				
binlog_error_action	はい	はい	はい		両方	はい
binlog_format	はい	はい	はい		両方	はい
binlog_gtid_recovery_simplified	はい	はい	はい		グローバル	いい
binlog-ignore-db	はい	はい				
binlog_max_flush_queue_time			はい		グローバル	はい
binlog_order_commits			はい		グローバル	はい
binlog-row-event-max-size	はい	はい				
binlog_row_image	はい	はい	はい		両方	はい
binlog-rows-query-log-events	はい	はい				
binlog_rows_query_log_events			はい		両方	はい
Binlog_stmt_cache_disk_use				はい	グローバル	いい
binlog_stmt_cache_size	はい	はい	はい		グローバル	はい
Binlog_stmt_cache_use				はい	グローバル	いい
binlogging_impossible_mode	はい	はい	はい		両方	はい
block_encryption_mode	はい	はい	はい		両方	はい
bootstrap	はい	はい				
bulk_insert_buffer_size	はい	はい	はい		両方	はい
Bytes_received				はい	両方	いい
Bytes_sent				はい	両方	いい
character_set_client			はい		両方	はい

名前	コマンド行	オプションファイル	システム変数	ステータス変数	変数スコープ	動的
character-set-client-handshake	はい	はい				
character_set_connection			はい		両方	はい
character_set_database (note 1)			はい		両方	はい
character_set_filesystem	はい	はい	はい		両方	はい
character_set_results			はい		両方	はい
character_set_server	はい	はい	はい		両方	はい
character_set_system			はい		グローバル	いいえ
character_sets_dir	はい	はい	はい		グローバル	いいえ
chroot	はい	はい				
collation_connection			はい		両方	はい
collation_database (note 1)			はい		両方	はい
collation_server	はい	はい	はい		両方	はい
Com_admin_commands				はい	両方	いいえ
Com_alter_db				はい	両方	いいえ
Com_alter_db_upgrade				はい	両方	いいえ
Com_alter_event				はい	両方	いいえ
Com_alter_function				はい	両方	いいえ
Com_alter_procedure				はい	両方	いいえ
Com_alter_server				はい	両方	いいえ
Com_alter_table				はい	両方	いいえ
Com_alter_tablespace				はい	両方	いいえ
Com_alter_user				はい	両方	いいえ
Com_analyze				はい	両方	いいえ
Com_assign_to_keycache				はい	両方	いいえ
Com_begin				はい	両方	いいえ
Com_binlog				はい	両方	いいえ
Com_call_procedure				はい	両方	いいえ
Com_change_db				はい	両方	いいえ
Com_change_master				はい	両方	いいえ
Com_check				はい	両方	いいえ
Com_checksum				はい	両方	いいえ
Com_commit				はい	両方	いいえ
Com_create_db				はい	両方	いいえ
Com_create_event				はい	両方	いいえ
Com_create_function				はい	両方	いいえ
Com_create_index				はい	両方	いいえ
Com_create_procedure				はい	両方	いいえ
Com_create_server				はい	両方	いいえ
Com_create_table				はい	両方	いいえ
Com_create_trigger				はい	両方	いいえ
Com_create_udf				はい	両方	いいえ

名前	コマンド行	オプションファイル	システム変数	ステータス変数	変数スコープ	動的
Com_create_user				はい	両方	いい
Com_create_view				はい	両方	いい
Com_dealloc_sql				はい	両方	いい
Com_delete				はい	両方	いい
Com_delete_multi				はい	両方	いい
Com_do				はい	両方	いい
Com_drop_db				はい	両方	いい
Com_drop_event				はい	両方	いい
Com_drop_function				はい	両方	いい
Com_drop_index				はい	両方	いい
Com_drop_procedure				はい	両方	いい
Com_drop_server				はい	両方	いい
Com_drop_table				はい	両方	いい
Com_drop_trigger				はい	両方	いい
Com_drop_user				はい	両方	いい
Com_drop_view				はい	両方	いい
Com_empty_query				はい	両方	いい
Com_execute_sql				はい	両方	いい
Com_flush				はい	両方	いい
Com_get_diagnostics				はい	両方	いい
Com_grant				はい	両方	いい
Com_ha_close				はい	両方	いい
Com_ha_open				はい	両方	いい
Com_ha_read				はい	両方	いい
Com_help				はい	両方	いい
Com_insert				はい	両方	いい
Com_insert_select				はい	両方	いい
Com_install_plugin				はい	両方	いい
Com_kill				はい	両方	いい
Com_load				はい	両方	いい
Com_lock_tables				はい	両方	いい
Com_optimize				はい	両方	いい
Com_preload_keys				はい	両方	いい
Com_prepare_sql				はい	両方	いい
Com_purge				はい	両方	いい
Com_purge_before_date				はい	両方	いい
Com_release_savepoint				はい	両方	いい
Com_rename_table				はい	両方	いい
Com_rename_user				はい	両方	いい
Com_repair				はい	両方	いい
Com_replace				はい	両方	いい
Com_replace_select				はい	両方	いい
Com_reset				はい	両方	いい

名前	コマンド行	オプションファイル	システム変数	ステータス変数	変数スコープ	動的
Com_resignal				はい	両方	いいえ
Com_revoke				はい	両方	いいえ
Com_revoke_all				はい	両方	いいえ
Com_rollback				はい	両方	いいえ
Com_rollback_to_savepoint				はい	両方	いいえ
Com_savepoint				はい	両方	いいえ
Com_select				はい	両方	いいえ
Com_set_option				はい	両方	いいえ
Com_show_authors				はい	両方	いいえ
Com_show_binlog_events				はい	両方	いいえ
Com_show_binlogs				はい	両方	いいえ
Com_show_charsets				はい	両方	いいえ
Com_show_collations				はい	両方	いいえ
Com_show_contributors				はい	両方	いいえ
Com_show_create_db				はい	両方	いいえ
Com_show_create_event				はい	両方	いいえ
Com_show_create_func				はい	両方	いいえ
Com_show_create_proc				はい	両方	いいえ
Com_show_create_table				はい	両方	いいえ
Com_show_create_trigger				はい	両方	いいえ
Com_show_databases				はい	両方	いいえ
Com_show_engine_logs				はい	両方	いいえ
Com_show_engine_mutex				はい	両方	いいえ
Com_show_engine_status				はい	両方	いいえ
Com_show_errors				はい	両方	いいえ
Com_show_events				はい	両方	いいえ
Com_show_fields				はい	両方	いいえ
Com_show_function_code				はい	両方	いいえ
Com_show_function_status				はい	両方	いいえ
Com_show_grants				はい	両方	いいえ
Com_show_keys				はい	両方	いいえ
Com_show_master_status				はい	両方	いいえ
Com_show_ndb_status				はい	両方	いいえ
Com_show_new_master				はい	両方	いいえ
Com_show_open_tables				はい	両方	いいえ
Com_show_plugins				はい	両方	いいえ
Com_show_privileges				はい	両方	いいえ
Com_show_procedure_code				はい	両方	いいえ
Com_show_procedure_status				はい	両方	いいえ
Com_show_processlist				はい	両方	いいえ
Com_show_profile				はい	両方	いいえ
Com_show_profiles				はい	両方	いいえ
Com_show_relaylog_events				はい	両方	いいえ

名前	コマンド行	オプションファイル	システム変数	ステータス変数	変数スコープ	動的
Com_show_slave_hosts				はい	両方	いい
Com_show_slave_status				はい	両方	いい
Com_show_status				はい	両方	いい
Com_show_storage_engines				はい	両方	いい
Com_show_table_status				はい	両方	いい
Com_show_tables				はい	両方	いい
Com_show_triggers				はい	両方	いい
Com_show_variables				はい	両方	いい
Com_show_warnings				はい	両方	いい
Com_signal				はい	両方	いい
Com_slave_start				はい	両方	いい
Com_slave_stop				はい	両方	いい
Com_stmt_close				はい	両方	いい
Com_stmt_execute				はい	両方	いい
Com_stmt_fetch				はい	両方	いい
Com_stmt_prepare				はい	両方	いい
Com_stmt_reprepare				はい	両方	いい
Com_stmt_reset				はい	両方	いい
Com_stmt_send_long_data				はい	両方	いい
Com_truncate				はい	両方	いい
Com_uninstall_plugin				はい	両方	いい
Com_unlock_tables				はい	両方	いい
Com_update				はい	両方	いい
Com_update_multi				はい	両方	いい
Com_xa_commit				はい	両方	いい
Com_xa_end				はい	両方	いい
Com_xa_prepare				はい	両方	いい
Com_xa_recover				はい	両方	いい
Com_xa_rollback				はい	両方	いい
Com_xa_start				はい	両方	いい
completion_type	はい	はい	はい		両方	はい
Compression				はい	セッション	いい
concurrent_insert	はい	はい	はい		グローバル	はい
connect_timeout	はい	はい	はい		グローバル	はい
Connection_errors_accept				はい	グローバル	いい
Connection_errors_internal				はい	グローバル	いい
Connection_errors_max_connections				はい	グローバル	いい
Connection_errors_peer_addr				はい	グローバル	いい
Connection_errors_select				はい	グローバル	いい
Connection_errors_tcpwrap				はい	グローバル	いい
Connections				はい	グローバル	いい
console	はい	はい				
core-file	はい	はい				

名前	コマンド行	オプションファイル	システム変数	ステータス変数	変数スコープ	動的
core_file			はい		グローバル	いいえ
Created_tmp_disk_tables				はい	両方	いいえ
Created_tmp_files				はい	グローバル	いいえ
Created_tmp_tables				はい	両方	いいえ
daemon_memcached_enable_binlog	はい	はい	はい		グローバル	いいえ
daemon_memcached_engine_lib_name	はい	はい	はい		グローバル	いいえ
daemon_memcached_engine_lib_path	はい	はい	はい		グローバル	いいえ
daemon_memcached_option	はい	はい	はい		グローバル	いいえ
daemon_memcached_r_batch_size	はい	はい	はい		グローバル	いいえ
daemon_memcached_w_batch_size	はい	はい	はい		グローバル	いいえ
datadir	はい	はい	はい		グローバル	いいえ
date_format			はい		グローバル	いいえ
datetime_format			はい		グローバル	いいえ
debug	はい	はい	はい		両方	はい
debug_sync			はい		セッション	はい
debug-sync-timeout	はい	はい				
default-authentication-plugin	はい	はい				
default_storage_engine	はい	はい	はい		両方	はい
default-time-zone	はい	はい				
default_tmp_storage_engine	はい	はい	はい		両方	はい
default_week_format	はい	はい	はい		両方	はい
defaults-extra-file	はい					
defaults-file	はい					
defaults-group-suffix	はい					
delay_key_write	はい	はい	はい		グローバル	はい
Delayed_errors				はい	グローバル	いいえ
delayed_insert_limit	はい	はい	はい		グローバル	はい
Delayed_insert_threads				はい	グローバル	いいえ
delayed_insert_timeout	はい	はい	はい		グローバル	はい
delayed_queue_size	はい	はい	はい		グローバル	はい
Delayed_writes				はい	グローバル	いいえ
des-key-file	はい	はい				
disable_gtid_unsafe_statements	はい	はい	はい		グローバル	いいえ
disable_gtid_unsafe_statements	はい	はい	はい		グローバル	いいえ
disconnect_on_expired_password	はい	はい	はい		セッション	いいえ
disconnect-slave-event-count	はい	はい				
div_precision_increment	はい	はい	はい		両方	はい
enable-named-pipe	はい	はい				
- 変数: named_pipe						
end_markers_in_json			はい		両方	はい
enforce_gtid_consistency	はい	はい	はい		グローバル	いいえ
enforce_gtid_consistency	はい	はい	はい		グローバル	いいえ

名前	コマンド行	オプションファイル	システム変数	ステータス変数	変数スコープ	動的
engine_condition_pushdown	はい	はい	はい		両方	はい
eq_range_index_dive_limit			はい		両方	はい
error_count			はい		セッション	いい
event_scheduler	はい	はい	はい		グローバル	はい
exit-info	はい	はい				
expire_logs_days	はい	はい	はい		グローバル	はい
explicit_defaults_for_timestamp	はい	はい	はい		セッション	いい
external-locking	はい	はい				
- 変数: skip_external_locking						
external_user			はい		セッション	いい
federated	はい	はい				
flush	はい	はい	はい		グローバル	はい
Flush_commands				はい	グローバル	いい
flush_time	はい	はい	はい		グローバル	はい
foreign_key_checks			はい		両方	はい
ft_boolean_syntax	はい	はい	はい		グローバル	はい
ft_max_word_len	はい	はい	はい		グローバル	いい
ft_min_word_len	はい	はい	はい		グローバル	いい
ft_query_expansion_limit	はい	はい	はい		グローバル	いい
ft_stopword_file	はい	はい	はい		グローバル	いい
gdb	はい	はい				
general_log	はい	はい	はい		グローバル	はい
general_log_file	はい	はい	はい		グローバル	はい
group_concat_max_len	はい	はい	はい		両方	はい
gtid_done			はい		両方	いい
gtid_executed			はい		両方	いい
gtid_lost			はい		グローバル	いい
gtid_mode	はい	はい	はい		グローバル	いい
gtid_mode			はい		グローバル	いい
gtid_next			はい		セッション	はい
gtid_owned			はい		両方	いい
gtid_purged			はい		グローバル	はい
Handler_commit				はい	両方	いい
Handler_delete				はい	両方	いい
Handler_discover				はい	両方	いい
Handler_external_lock				はい	両方	いい
Handler_mrr_init				はい	両方	いい
Handler_prepare				はい	両方	いい
Handler_read_first				はい	両方	いい
Handler_read_key				はい	両方	いい
Handler_read_last				はい	両方	いい
Handler_read_next				はい	両方	いい
Handler_read_prev				はい	両方	いい

名前	コマンド行	オプションファイル	システム変数	ステータス変数	変数スコープ	動的
Handler_read_rnd				はい	両方	いいえ
Handler_read_rnd_next				はい	両方	いいえ
Handler_rollback				はい	両方	いいえ
Handler_savepoint				はい	両方	いいえ
Handler_savepoint_rollback				はい	両方	いいえ
Handler_update				はい	両方	いいえ
Handler_write				はい	両方	いいえ
have_compress			はい		グローバル	いいえ
have_crypt			はい		グローバル	いいえ
have_csv			はい		グローバル	いいえ
have_dynamic_loading			はい		グローバル	いいえ
have_geometry			はい		グローバル	いいえ
have_innodb			はい		グローバル	いいえ
have_ndbcluster			はい		グローバル	いいえ
have_openssl			はい		グローバル	いいえ
have_partitioning			はい		グローバル	いいえ
have_profiling			はい		グローバル	いいえ
have_query_cache			はい		グローバル	いいえ
have_rtree_keys			はい		グローバル	いいえ
have_ssl			はい		グローバル	いいえ
have_symlink			はい		グローバル	いいえ
help	はい	はい				
host_cache_size			はい		グローバル	はい
hostname			はい		グローバル	いいえ
identity			はい		セッション	はい
ignore_builtin_innodb	はい	はい	はい		グローバル	いいえ
ignore-db-dir	はい	はい				
ignore_db_dirs			はい		グローバル	いいえ
init_connect	はい	はい	はい		グローバル	はい
init_file	はい	はい	はい		グローバル	いいえ
init_slave	はい	はい	はい		グローバル	はい
innodb	はい	はい				
innodb_adaptive_flushing	はい	はい	はい		グローバル	はい
innodb_adaptive_flushing_lwm	はい	はい	はい		グローバル	はい
innodb_adaptive_hash_index	はい	はい	はい		グローバル	はい
innodb_adaptive_max_sleep_delay	はい	はい	はい		グローバル	はい
innodb_additional_mem_pool_size	はい	はい	はい		グローバル	いいえ
innodb_api_bk_commit_interval	はい	はい	はい		グローバル	はい
innodb_api_disable_rowlock	はい	はい	はい		グローバル	いいえ
innodb_api_enable_binlog	はい	はい	はい		グローバル	いいえ
innodb_api_enable_mdl	はい	はい	はい		グローバル	いいえ
innodb_api_trx_level	はい	はい	はい		グローバル	はい
innodb_autoextend_increment	はい	はい	はい		グローバル	はい

名前	コマンド行	オプションファイル	システム変数	ステータス変数	変数スコープ	動的
innodb_autoinc_lock_mode	はい	はい	はい		グローバル	いい
Innodb_available_undo_logs				はい	グローバル	いい
Innodb_buffer_pool_bytes_data				はい	グローバル	いい
Innodb_buffer_pool_bytes_dirty				はい	グローバル	いい
innodb_buffer_pool_dump_at_shutdown	はい	はい	はい		グローバル	はい
innodb_buffer_pool_dump_now	はい	はい	はい		グローバル	はい
Innodb_buffer_pool_dump_status				はい	グローバル	いい
innodb_buffer_pool_filename	はい	はい	はい		グローバル	はい
innodb_buffer_pool_instances	はい	はい	はい		グローバル	いい
innodb_buffer_pool_load_abort	はい	はい	はい		グローバル	はい
innodb_buffer_pool_load_at_startup	はい	はい	はい		グローバル	いい
innodb_buffer_pool_load_now	はい	はい	はい		グローバル	はい
Innodb_buffer_pool_load_status				はい	グローバル	いい
Innodb_buffer_pool_pages_data				はい	グローバル	いい
Innodb_buffer_pool_pages_dirty				はい	グローバル	いい
Innodb_buffer_pool_pages_flushed				はい	グローバル	いい
Innodb_buffer_pool_pages_free				はい	グローバル	いい
Innodb_buffer_pool_pages_latched				はい	グローバル	いい
Innodb_buffer_pool_pages_misc				はい	グローバル	いい
Innodb_buffer_pool_pages_total				はい	グローバル	いい
Innodb_buffer_pool_read_ahead				はい	グローバル	いい
Innodb_buffer_pool_read_ahead_evicted				はい	グローバル	いい
Innodb_buffer_pool_read_requests				はい	グローバル	いい
Innodb_buffer_pool_reads				はい	グローバル	いい
innodb_buffer_pool_size	はい	はい	はい		グローバル	いい
Innodb_buffer_pool_wait_free				はい	グローバル	いい
Innodb_buffer_pool_write_requests				はい	グローバル	いい
innodb_change_buffer_max_size	はい	はい	はい		グローバル	はい
innodb_change_buffering	はい	はい	はい		グローバル	はい
innodb_checksum_algorithm	はい	はい	はい		グローバル	はい
innodb_checksums	はい	はい	はい		グローバル	いい
innodb_cmp_per_index_enabled	はい	はい	はい		グローバル	はい
innodb_commit_concurrency	はい	はい	はい		グローバル	はい
innodb_compression_failure_threshold_pct	はい	はい	はい		グローバル	はい
innodb_compression_level	はい	はい	はい		グローバル	はい
innodb_compression_pad_pct_max	はい	はい	はい		グローバル	はい
innodb_concurrency_tickets	はい	はい	はい		グローバル	はい
innodb_data_file_path	はい	はい	はい		グローバル	いい
Innodb_data_fsyncs				はい	グローバル	いい
innodb_data_home_dir	はい	はい	はい		グローバル	いい
Innodb_data_pending_fsyncs				はい	グローバル	いい
Innodb_data_pending_reads				はい	グローバル	いい
Innodb_data_pending_writes				はい	グローバル	いい

名前	コマンド行	オプションファイル	システム変数	ステータス変数	変数スコープ	動的
Innodb_data_read				はい	グローバル	いいえ
Innodb_data_reads				はい	グローバル	いいえ
Innodb_data_writes				はい	グローバル	いいえ
Innodb_data_written				はい	グローバル	いいえ
Innodb_dblwr_pages_written				はい	グローバル	いいえ
Innodb_dblwr_writes				はい	グローバル	いいえ
innodb_disable_sort_file_cache	はい	はい	はい		グローバル	はい
innodb_doublewrite	はい	はい	はい		グローバル	いいえ
innodb_fast_shutdown	はい	はい	はい		グローバル	はい
innodb_file_format	はい	はい	はい		グローバル	はい
innodb_file_format_check	はい	はい	はい		グローバル	いいえ
innodb_file_format_max	はい	はい	はい		グローバル	はい
innodb_file_per_table	はい	はい	はい		グローバル	はい
innodb_flush_log_at_timeout			はい		グローバル	はい
innodb_flush_log_at_trx_commit	はい	はい	はい		グローバル	はい
innodb_flush_method	はい	はい	はい		グローバル	いいえ
innodb_flush_neighbors	はい	はい	はい		グローバル	はい
innodb_flushing_avg_loops	はい	はい	はい		グローバル	はい
innodb_force_load_corrupted	はい	はい	はい		グローバル	いいえ
innodb_force_recovery	はい	はい	はい		グローバル	いいえ
innodb_ft_aux_table			はい		グローバル	はい
innodb_ft_cache_size	はい	はい	はい		グローバル	いいえ
innodb_ft_enable_diag_print	はい	はい	はい		グローバル	はい
innodb_ft_enable_stopword	はい	はい	はい		グローバル	はい
innodb_ft_max_token_size	はい	はい	はい		グローバル	いいえ
innodb_ft_min_token_size	はい	はい	はい		グローバル	いいえ
innodb_ft_num_word_optimize	はい	はい	はい		グローバル	はい
innodb_ft_result_cache_limit	はい	はい	はい		グローバル	はい
innodb_ft_server_stopword_table	はい	はい	はい		グローバル	はい
innodb_ft_sort_pll_degree	はい	はい	はい		グローバル	いいえ
innodb_ft_total_cache_size	はい	はい	はい		グローバル	いいえ
innodb_ft_user_stopword_table	はい	はい	はい		両方	はい
Innodb_have_atomic_builtins				はい	グローバル	いいえ
innodb_io_capacity	はい	はい	はい		グローバル	はい
innodb_io_capacity_max	はい	はい	はい		グローバル	はい
innodb_large_prefix	はい	はい	はい		グローバル	はい
innodb_lock_wait_timeout	はい	はい	はい		両方	はい
innodb_locks_unsafe_for_binlog	はい	はい	はい		グローバル	いいえ
innodb_log_buffer_size	はい	はい	はい		グローバル	いいえ
innodb_log_compressed_pages	はい	はい	はい		グローバル	はい
innodb_log_file_size	はい	はい	はい		グローバル	いいえ
innodb_log_files_in_group	はい	はい	はい		グローバル	いいえ
innodb_log_group_home_dir	はい	はい	はい		グローバル	いいえ

名前	コマンド行	オプションファイル	システム変数	ステータス変数	変数スコープ	動的
Innodb_log_waits				はい	グローバル	いい
Innodb_log_write_requests				はい	グローバル	いい
Innodb_log_writes				はい	グローバル	いい
innodb_lru_scan_depth	はい	はい	はい		グローバル	はい
innodb_max_dirty_pages_pct	はい	はい	はい		グローバル	はい
innodb_max_dirty_pages_pct_lwm	はい	はい	はい		グローバル	はい
innodb_max_purge_lag	はい	はい	はい		グローバル	はい
innodb_max_purge_lag_delay	はい	はい	はい		グローバル	はい
innodb_mirrored_log_groups	はい	はい	はい		グローバル	いい
innodb_monitor_disable	はい	はい	はい		グローバル	はい
innodb_monitor_enable	はい	はい	はい		グローバル	はい
innodb_monitor_reset	はい	はい	はい		グローバル	はい
innodb_monitor_reset_all	はい	はい	はい		グローバル	はい
Innodb_num_open_files				はい	グローバル	いい
innodb_old_blocks_pct	はい	はい	はい		グローバル	はい
innodb_old_blocks_time	はい	はい	はい		グローバル	はい
innodb_online_alter_log_max_size	はい	はい	はい		グローバル	はい
innodb_open_files	はい	はい	はい		グローバル	いい
innodb_optimize_fulltext_only	はい	はい	はい		グローバル	はい
Innodb_os_log_fsyncs				はい	グローバル	いい
Innodb_os_log_pending_fsyncs				はい	グローバル	いい
Innodb_os_log_pending_writes				はい	グローバル	いい
Innodb_os_log_written				はい	グローバル	いい
Innodb_page_size				はい	グローバル	いい
innodb_page_size	はい	はい	はい		グローバル	いい
Innodb_pages_created				はい	グローバル	いい
Innodb_pages_read				はい	グローバル	いい
Innodb_pages_written				はい	グローバル	いい
innodb_print_all_deadlocks	はい	はい	はい		グローバル	はい
innodb_purge_batch_size	はい	はい	はい		グローバル	はい
innodb_purge_threads	はい	はい	はい		グローバル	いい
innodb_random_read_ahead	はい	はい	はい		グローバル	はい
innodb_read_ahead_threshold	はい	はい	はい		グローバル	はい
innodb_read_io_threads	はい	はい	はい		グローバル	いい
innodb_read_only	はい	はい	はい		グローバル	いい
innodb_replication_delay	はい	はい	はい		グローバル	はい
innodb_rollback_on_timeout	はい	はい	はい		グローバル	いい
innodb_rollback_segments	はい	はい	はい		グローバル	はい
Innodb_row_lock_current_waits				はい	グローバル	いい
Innodb_row_lock_time				はい	グローバル	いい
Innodb_row_lock_time_avg				はい	グローバル	いい
Innodb_row_lock_time_max				はい	グローバル	いい
Innodb_row_lock_waits				はい	グローバル	いい

名前	コマンド行	オプションファイル	システム変数	ステータス変数	変数スコープ	動的
Innodb_rows_deleted				はい	グローバル	いいえ
Innodb_rows_inserted				はい	グローバル	いいえ
Innodb_rows_read				はい	グローバル	いいえ
Innodb_rows_updated				はい	グローバル	いいえ
innodb_sort_buffer_size	はい	はい	はい		グローバル	いいえ
innodb_spin_wait_delay	はい	はい	はい		グローバル	はい
innodb_stats_auto_recalc	はい	はい	はい		グローバル	はい
innodb_stats_method	はい	はい	はい		グローバル	はい
innodb_stats_on_metadata	はい	はい	はい		グローバル	はい
innodb_stats_persistent	はい	はい	はい		グローバル	はい
innodb_stats_persistent_sample_pages	はい	はい	はい		グローバル	はい
innodb_stats_sample_pages	はい	はい	はい		グローバル	はい
innodb_stats_transient_sample_pages	はい	はい	はい		グローバル	はい
innodb-status-file	はい	はい				
innodb_status_output	はい	はい	はい		グローバル	はい
innodb_status_output_locks	はい	はい	はい		グローバル	はい
innodb_strict_mode	はい	はい	はい		両方	はい
innodb_support_xa	はい	はい	はい		両方	はい
innodb_sync_array_size	はい	はい	はい		グローバル	いいえ
innodb_sync_spin_loops	はい	はい	はい		グローバル	はい
innodb_table_locks	はい	はい	はい		両方	はい
innodb_thread_concurrency	はい	はい	はい		グローバル	はい
innodb_thread_sleep_delay	はい	はい	はい		グローバル	はい
Innodb_truncated_status_writes				はい	グローバル	いいえ
innodb_undo_directory	はい	はい	はい		グローバル	いいえ
innodb_undo_logs	はい	はい	はい		グローバル	はい
innodb_undo_tablespaces	はい	はい	はい		グローバル	いいえ
innodb_use_native_aio	はい	はい	はい		グローバル	いいえ
innodb_use_sys_malloc	はい	はい	はい		グローバル	いいえ
innodb_version			はい		グローバル	いいえ
innodb_write_io_threads	はい	はい	はい		グローバル	いいえ
insert_id			はい		セッション	はい
install	はい					
install-manual	はい					
interactive_timeout	はい	はい	はい		両方	はい
join_buffer_size	はい	はい	はい		両方	はい
keep_files_on_create	はい	はい	はい		両方	はい
Key_blocks_not_flushed				はい	グローバル	いいえ
Key_blocks_unused				はい	グローバル	いいえ
Key_blocks_used				はい	グローバル	いいえ
key_buffer_size	はい	はい	はい		グローバル	はい
key_cache_age_threshold	はい	はい	はい		グローバル	はい
key_cache_block_size	はい	はい	はい		グローバル	はい

名前	コマンド行	オプションファイル	システム変数	ステータス変数	変数スコープ	動的
key_cache_division_limit	はい	はい	はい		グローバル	はい
Key_read_requests				はい	グローバル	いい
Key_reads				はい	グローバル	いい
Key_write_requests				はい	グローバル	いい
Key_writes				はい	グローバル	いい
language	はい	はい	はい		グローバル	いい
large_files_support			はい		グローバル	いい
large_page_size			はい		グローバル	いい
large_pages	はい	はい	はい		グローバル	いい
last_insert_id			はい		セッション	はい
Last_query_cost				はい	セッション	いい
Last_query_partial_plans				はい	セッション	いい
lc_messages	はい	はい	はい		両方	はい
lc_messages_dir	はい	はい	はい		グローバル	いい
lc_time_names			はい		両方	はい
license			はい		グローバル	いい
local_infile			はい		グローバル	はい
local-service	はい					
lock_wait_timeout	はい	はい	はい		両方	はい
locked_in_memory			はい		グローバル	いい
log	はい	はい	はい		グローバル	はい
log_bin	はい	はい	はい		グローバル	いい
log_bin			はい		グローバル	いい
log_bin_basename			はい		グローバル	いい
log-bin-index	はい	はい				
log_bin_index			はい		グローバル	いい
log_bin_trust_function_creators	はい	はい	はい		グローバル	はい
log_bin_use_v1_row_events	はい	はい	はい		グローバル	いい
log_bin_use_v1_row_events	はい	はい	はい		グローバル	いい
log_error	はい	はい	はい		グローバル	いい
log-isam	はい	はい				
log_output	はい	はい	はい		グローバル	はい
log_queries_not_using_indexes	はい	はい	はい		グローバル	はい
log-raw	はい	はい				
log-short-format	はい	はい				
log_slave_updates	はい	はい	はい		グローバル	いい
log_slave_updates	はい	はい	はい		グローバル	いい
log-slow-admin-statements	はい	はい				
log_slow_admin_statements			はい		グローバル	はい
log_slow_queries	はい	はい	はい		グローバル	はい
log-slow-slave-statements	はい	はい				
log_slow_slave_statements			はい		グローバル	はい

名前	コマンド行	オプションファイル	システム変数	ステータス変数	変数スコープ	動的
log-tc	はい	はい				
log-tc-size	はい	はい				
log_throttle_queries_not_using_indexes			はい		グローバル	はい
log_warnings	はい	はい	はい		異なる	はい
long_query_time	はい	はい	はい		両方	はい
low_priority_updates	はい	はい	はい		両方	はい
lower_case_file_system			はい		グローバル	いいえ
lower_case_table_names	はい	はい	はい		グローバル	いいえ
master-info-file	はい	はい				
master-info-repository	はい	はい				
master_info_repository	はい	はい	はい		グローバル	はい
master-retry-count	はい	はい				
master-verify-checksum	はい	はい				
master_verify_checksum			はい		グローバル	はい
max_allowed_packet	はい	はい	はい		グローバル	はい
max_binlog_cache_size	はい	はい	はい		グローバル	はい
max-binlog-dump-events	はい	はい				
max_binlog_size	はい	はい	はい		グローバル	はい
max_binlog_stmt_cache_size	はい	はい	はい		グローバル	はい
max_connect_errors	はい	はい	はい		グローバル	はい
max_connections	はい	はい	はい		グローバル	はい
max_delayed_threads	はい	はい	はい		両方	はい
max_error_count	はい	はい	はい		両方	はい
max_heap_table_size	はい	はい	はい		両方	はい
max_insert_delayed_threads			はい		両方	はい
max_join_size	はい	はい	はい		両方	はい
max_length_for_sort_data	はい	はい	はい		両方	はい
max_prepared_stmt_count	はい	はい	はい		グローバル	はい
max_relay_log_size	はい	はい	はい		グローバル	はい
max_seeks_for_key	はい	はい	はい		両方	はい
max_sort_length	はい	はい	はい		両方	はい
max_sp_recursion_depth	はい	はい	はい		両方	はい
Max_used_connections				はい	グローバル	いいえ
max_user_connections	はい	はい	はい		両方	はい
max_write_lock_count	はい	はい	はい		グローバル	はい
memlock	はい	はい				
- 変数: locked_in_memory						
metadata_locks_cache_size			はい		グローバル	いいえ
metadata_locks_hash_instances			はい		グローバル	いいえ
min_examined_row_limit	はい	はい	はい		両方	はい
myisam-block-size	はい	はい				
myisam_data_pointer_size	はい	はい	はい		グローバル	はい

名前	コマンド行	オプションファイル	システム変数	ステータス変数	変数スコープ	動的
myisam_max_sort_file_size	はい	はい	はい		グローバル	はい
myisam_mmap_size	はい	はい	はい		グローバル	いい
myisam-recover-options	はい	はい				
myisam_recover_options			はい		グローバル	いい
myisam_repair_threads	はい	はい	はい		両方	はい
myisam_sort_buffer_size	はい	はい	はい		両方	はい
myisam_stats_method	はい	はい	はい		両方	はい
myisam_use_mmap	はい	はい	はい		グローバル	はい
named_pipe			はい		グローバル	いい
Ndb_api_bytes_received_count				はい	グローバル	いい
Ndb_api_bytes_received_count_session				はい	セッション	いい
Ndb_api_bytes_received_count_slave				はい	グローバル	いい
Ndb_api_bytes_sent_count				はい	グローバル	いい
Ndb_api_bytes_sent_count_session				はい	セッション	いい
Ndb_api_bytes_sent_count_slave				はい	グローバル	いい
Ndb_api_event_bytes_count				はい	グローバル	いい
Ndb_api_event_bytes_count_injector				はい	グローバル	いい
Ndb_api_event_data_count				はい	グローバル	いい
Ndb_api_event_data_count_injector				はい	グローバル	いい
Ndb_api_event_nondata_count				はい	グローバル	いい
Ndb_api_event_nondata_count_injector				はい	グローバル	いい
Ndb_api_pk_op_count				はい	グローバル	いい
Ndb_api_pk_op_count_session				はい	セッション	いい
Ndb_api_pk_op_count_slave				はい	グローバル	いい
Ndb_api_pruned_scan_count				はい	グローバル	いい
Ndb_api_pruned_scan_count_session				はい	セッション	いい
Ndb_api_pruned_scan_count_slave				はい	グローバル	いい
Ndb_api_range_scan_count				はい	グローバル	いい
Ndb_api_range_scan_count_session				はい	セッション	いい
Ndb_api_range_scan_count_slave				はい	グローバル	いい
Ndb_api_read_row_count				はい	グローバル	いい
Ndb_api_read_row_count_session				はい	セッション	いい
Ndb_api_read_row_count_slave				はい	グローバル	いい
Ndb_api_scan_batch_count				はい	グローバル	いい
Ndb_api_scan_batch_count_session				はい	セッション	いい
Ndb_api_scan_batch_count_slave				はい	グローバル	いい
Ndb_api_table_scan_count				はい	グローバル	いい
Ndb_api_table_scan_count_session				はい	セッション	いい
Ndb_api_table_scan_count_slave				はい	グローバル	いい
Ndb_api_trans_abort_count				はい	グローバル	いい
Ndb_api_trans_abort_count_session				はい	セッション	いい
Ndb_api_trans_abort_count_slave				はい	グローバル	いい
Ndb_api_trans_close_count				はい	グローバル	いい

名前	コマンド行	オプションファイル	システム変数	ステータス変数	変数スコープ	動的
Ndb_api_trans_close_count_session				はい	セッション	いいえ
Ndb_api_trans_close_count_slave				はい	グローバル	いいえ
Ndb_api_trans_commit_count				はい	グローバル	いいえ
Ndb_api_trans_commit_count_session				はい	セッション	いいえ
Ndb_api_trans_commit_count_slave				はい	グローバル	いいえ
Ndb_api_trans_local_read_row_count				はい	グローバル	いいえ
Ndb_api_trans_local_read_row_count_session				はい	セッション	いいえ
Ndb_api_trans_local_read_row_count_slave				はい	グローバル	いいえ
Ndb_api_trans_start_count				はい	グローバル	いいえ
Ndb_api_trans_start_count_session				はい	セッション	いいえ
Ndb_api_trans_start_count_slave				はい	グローバル	いいえ
Ndb_api_uk_op_count				はい	グローバル	いいえ
Ndb_api_uk_op_count_session				はい	セッション	いいえ
Ndb_api_uk_op_count_slave				はい	グローバル	いいえ
Ndb_api_wait_exec_complete_count				はい	グローバル	いいえ
Ndb_api_wait_exec_complete_count_session				はい	セッション	いいえ
Ndb_api_wait_exec_complete_count_slave				はい	グローバル	いいえ
Ndb_api_wait_meta_request_count				はい	グローバル	いいえ
Ndb_api_wait_meta_request_count_session				はい	セッション	いいえ
Ndb_api_wait_meta_request_count_slave				はい	グローバル	いいえ
Ndb_api_wait_nanos_count				はい	グローバル	いいえ
Ndb_api_wait_nanos_count_session				はい	セッション	いいえ
Ndb_api_wait_nanos_count_slave				はい	グローバル	いいえ
Ndb_api_wait_scan_result_count				はい	グローバル	いいえ
Ndb_api_wait_scan_result_count_session				はい	セッション	いいえ
Ndb_api_wait_scan_result_count_slave				はい	グローバル	いいえ
ndb_autoincrement_prefetch_sz	はい	はい	はい		両方	はい
ndb_batch_size	はい	はい	はい		グローバル	いいえ
ndb_blob_read_batch_bytes	はい	はい	はい		両方	はい
ndb_blob_write_batch_bytes	はい	はい	はい		両方	はい
ndb_cache_check_time	はい	はい	はい		グローバル	はい
ndb_cluster_connection_pool	はい	はい	はい		グローバル	いいえ
Ndb_cluster_node_id				はい	両方	いいえ
Ndb_config_from_host				はい	両方	いいえ
Ndb_config_from_port				はい	両方	いいえ
Ndb_conflict_fn_epoch				はい	グローバル	いいえ
Ndb_conflict_fn_epoch_trans				はい	グローバル	いいえ
Ndb_conflict_fn_max				はい	グローバル	いいえ
Ndb_conflict_fn_old				はい	グローバル	いいえ
Ndb_conflict_trans_conflict_commit_count				はい	グローバル	いいえ
Ndb_conflict_trans_detect_iter_count				はい	グローバル	いいえ
Ndb_conflict_trans_reject_count				はい	グローバル	いいえ
Ndb_conflict_trans_row_conflict_count				はい	グローバル	いいえ

名前	コマンド行	オプションファイル	システム変数	ステータス変数	変数スコープ	動的
Ndb_conflict_trans_row_reject_count				はい	グローバル	いい
ndb-connectstring	はい	はい				
ndb_deferred_constraints	はい	はい	はい		両方	はい
ndb_deferred_constraints	はい	はい	はい		両方	はい
ndb_distribution	はい	はい	はい		グローバル	はい
ndb_distribution	はい	はい	はい		グローバル	はい
ndb_eventbuffer_max_alloc	はい	はい	はい		グローバル	はい
Ndb_execute_count				はい	グローバル	いい
ndb_extra_logging	はい	はい	はい		グローバル	はい
ndb_force_send	はい	はい	はい		両方	はい
ndb_index_stat_cache_entries	はい	はい	はい		両方	はい
ndb_index_stat_enable	はい	はい	はい		両方	はい
ndb_index_stat_option	はい	はい	はい		両方	はい
ndb_index_stat_update_freq	はい	はい	はい		両方	はい
ndb_join_pushdown			はい		両方	はい
Ndb_last_commit_epoch_server				はい	グローバル	いい
Ndb_last_commit_epoch_session				はい	セッション	いい
ndb_log_apply_status	はい	はい	はい		グローバル	いい
ndb_log_apply_status	はい	はい	はい		グローバル	いい
ndb_log_bin	はい		はい		両方	はい
ndb_log_binlog_index	はい		はい		グローバル	はい
ndb_log_empty_epochs	はい	はい	はい		グローバル	はい
ndb_log_empty_epochs	はい	はい	はい		グローバル	はい
ndb_log_exclusive_reads	はい	はい	はい		両方	はい
ndb_log_exclusive_reads	はい	はい	はい		両方	はい
ndb_log_orig	はい	はい	はい		グローバル	いい
ndb_log_orig	はい	はい	はい		グローバル	いい
ndb_log_transaction_id	はい	はい	はい		グローバル	いい
ndb_log_transaction_id			はい		グローバル	いい
ndb_log_update_as_write	はい	はい	はい		グローバル	はい
ndb_log_updated_only	はい	はい	はい		グローバル	はい
ndb-mgmd-host	はい	はい				
ndb_nodeid	はい	はい		はい	グローバル	いい
Ndb_number_of_data_nodes				はい	グローバル	いい
ndb_optimization_delay			はい		グローバル	はい
ndb_optimized_node_selection	はい	はい	はい		グローバル	いい
Ndb_pruned_scan_count				はい	グローバル	いい
Ndb_pushed_queries_defined				はい	グローバル	いい
Ndb_pushed_queries_dropped				はい	グローバル	いい
Ndb_pushed_queries_executed				はい	グローバル	いい
Ndb_pushed_reads				はい	グローバル	いい
ndb-recv-thread-activation-threshold	はい	はい				
ndb_recv_thread_activation_threshold			はい		グローバル	はい

名前	コマンド行	オプションファイル	システム変数	ステータス変数	変数スコープ	動的
ndb-recv-thread-cpu-mask	はい	はい				
ndb_recv_thread_cpu_mask			はい		グローバル	はい
ndb_report_thresh_binlog_epoch_slip	はい	はい				
ndb_report_thresh_binlog_mem_usage	はい	はい				
Ndb_scan_count				はい	グローバル	いいえ
ndb_show_foreign_key_mock_tables	はい	はい	はい		グローバル	はい
ndb_slave_conflict_role	はい	はい	はい		グローバル	はい
Ndb_slave_last_conflict_epoch			はい		グローバル	いいえ
Ndb_slave_max_replicated_epoch			はい		グローバル	いいえ
ndb_table_no_logging			はい		セッション	はい
ndb_table_temporary			はい		セッション	はい
ndb-transid-mysql-connection-map	はい					
ndb_use_copying_alter_table			はい		両方	いいえ
ndb_use_exact_count			はい		両方	はい
ndb_use_transactions	はい	はい	はい		両方	はい
ndb_version			はい		グローバル	いいえ
ndb_version_string			はい		グローバル	いいえ
ndb_wait_connected	はい	はい	はい		グローバル	いいえ
ndb_wait_setup	はい	はい	はい		グローバル	いいえ
ndbcluster	はい	はい				
- 変数: have_ndbcluster						
ndbinfo_database			はい		グローバル	いいえ
ndbinfo_max_bytes	はい		はい		両方	はい
ndbinfo_max_rows	はい		はい		両方	はい
ndbinfo_offline			はい		グローバル	はい
ndbinfo_show_hidden	はい		はい		両方	はい
ndbinfo_table_prefix	はい		はい		両方	はい
ndbinfo_version			はい		グローバル	いいえ
net_buffer_length	はい	はい	はい		両方	はい
net_read_timeout	はい	はい	はい		両方	はい
net_retry_count	はい	はい	はい		両方	はい
net_write_timeout	はい	はい	はい		両方	はい
new	はい	はい	はい		両方	はい
no-defaults	はい					
Not_flushed_delayed_rows				はい	グローバル	いいえ
old	はい	はい	はい		グローバル	いいえ
old_alter_table	はい	はい	はい		両方	はい
old_passwords			はい		両方	はい
old-style-user-limits	はい	はい				
one-thread	はい	はい				
Open_files				はい	グローバル	いいえ
open_files_limit	はい	はい	はい		グローバル	いいえ

名前	コマンド行	オプションファイル	システム変数	ステータス変数	変数スコープ	動的
Open_streams				はい	グローバル	いい
Open_table_definitions				はい	グローバル	いい
Open_tables				はい	両方	いい
Opened_files				はい	グローバル	いい
Opened_table_definitions				はい	両方	いい
Opened_tables				はい	両方	いい
optimizer_join_cache_level	はい	はい	はい		両方	はい
optimizer_prune_level	はい	はい	はい		両方	はい
optimizer_search_depth	はい	はい	はい		両方	はい
optimizer_switch	はい	はい	はい		両方	はい
optimizer_trace			はい		両方	はい
optimizer_trace_features			はい		両方	はい
optimizer_trace_limit			はい		両方	はい
optimizer_trace_max_mem_size			はい		両方	はい
optimizer_trace_offset			はい		両方	はい
partition	はい	はい				
- 変数: have_partitioning						
performance_schema	はい	はい	はい		グローバル	いい
Performance_schema_accounts_lost				はい	グローバル	いい
performance_schema_accounts_size	はい	はい	はい		グローバル	いい
Performance_schema_cond_classes_lost				はい	グローバル	いい
Performance_schema_cond_instances_lost				はい	グローバル	いい
performance-schema-consumer-events-stages-current	はい	はい				
performance-schema-consumer-events-stages-history	はい	はい				
performance-schema-consumer-events-stages-history-long	はい	はい				
performance-schema-consumer-events-statements-current	はい	はい				
performance-schema-consumer-events-statements-history	はい	はい				
performance-schema-consumer-events-statements-history-long	はい	はい				
performance-schema-consumer-events-waits-current	はい	はい				
performance-schema-consumer-events-waits-history	はい	はい				

名前	コマンド行	オプションファイル	システム変数	ステータス変数	変数スコープ	動的
performance-schema-consumer-events-waits-history-long	はい	はい				
performance-schema-consumer-global-instrumentation	はい	はい				
performance-schema-consumer-statements-digest	はい	はい				
performance-schema-consumer-thread-instrumentation	はい	はい				
Performance_schema_digest_lost				はい	グローバル	いいえ
performance_schema_digests_size	はい	はい	はい		グローバル	いいえ
performance_schema_events_stages_history_long_size	はい	はい	はい		グローバル	いいえ
performance_schema_events_stages_history_size	はい	はい	はい		グローバル	いいえ
performance_schema_events_statements_history_long_size	はい	はい	はい		グローバル	いいえ
performance_schema_events_statements_history_size	はい	はい	はい		グローバル	いいえ
performance_schema_events_waits_history_long_size	はい	はい	はい		グローバル	いいえ
performance_schema_events_waits_history_size	はい	はい	はい		グローバル	いいえ
Performance_schema_file_classes_lost				はい	グローバル	いいえ
Performance_schema_file_handles_lost				はい	グローバル	いいえ
Performance_schema_file_instances_lost				はい	グローバル	いいえ
Performance_schema_hosts_lost				はい	グローバル	いいえ
performance_schema_hosts_size	はい	はい	はい		グローバル	いいえ
performance-schema-instrument	はい	はい				
Performance_schema_locker_lost				はい	グローバル	いいえ
performance_schema_max_cond_classes	はい	はい	はい		グローバル	いいえ
performance_schema_max_cond_instances	はい	はい	はい		グローバル	いいえ
performance_schema_max_file_classes	はい	はい	はい		グローバル	いいえ
performance_schema_max_file_handles	はい	はい	はい		グローバル	いいえ
performance_schema_max_file_instances	はい	はい	はい		グローバル	いいえ
performance_schema_max_mutex_classes	はい	はい	はい		グローバル	いいえ
performance_schema_max_mutex_instances	はい	はい	はい		グローバル	いいえ
performance_schema_max_rwlock_classes	はい	はい	はい		グローバル	いいえ
performance_schema_max_rwlock_instances	はい	はい	はい		グローバル	いいえ
performance_schema_max_socket_classes	はい	はい	はい		グローバル	いいえ
performance_schema_max_socket_instances	はい	はい	はい		グローバル	いいえ
performance_schema_max_stage_classes	はい	はい	はい		グローバル	いいえ
performance_schema_max_statement_classes	はい	はい	はい		グローバル	いいえ
performance_schema_max_table_handles	はい	はい	はい		グローバル	いいえ
performance_schema_max_table_instances	はい	はい	はい		グローバル	いいえ
performance_schema_max_thread_classes	はい	はい	はい		グローバル	いいえ
performance_schema_max_thread_instances	はい	はい	はい		グローバル	いいえ
Performance_schema_mutex_classes_lost				はい	グローバル	いいえ

名前	コマンド行	オプションファイル	システム変数	ステータス変数	変数スコープ	動的
Performance_schema_mutex_instances_lost				はい	グローバル	いい
Performance_schema_rwlock_classes_lost				はい	グローバル	いい
Performance_schema_rwlock_instances_lost				はい	グローバル	いい
Performance_schema_session_connect_attrs_lost				はい	グローバル	いい
performance_schema_session_connect_attrs_size	はい	はい	はい		グローバル	いい
performance_schema_setup_actors_size	はい	はい	はい		グローバル	いい
performance_schema_setup_objects_size	はい	はい	はい		グローバル	いい
Performance_schema_socket_classes_lost				はい	グローバル	いい
Performance_schema_socket_instances_lost				はい	グローバル	いい
Performance_schema_stage_classes_lost				はい	グローバル	いい
Performance_schema_statement_classes_lost				はい	グローバル	いい
Performance_schema_table_handles_lost				はい	グローバル	いい
Performance_schema_table_instances_lost				はい	グローバル	いい
Performance_schema_thread_classes_lost				はい	グローバル	いい
Performance_schema_thread_instances_lost				はい	グローバル	いい
Performance_schema_users_lost				はい	グローバル	いい
performance_schema_users_size	はい	はい	はい		グローバル	いい
pid_file	はい	はい	はい		グローバル	いい
plugin	はい	はい				
plugin_dir	はい	はい	はい		グローバル	いい
plugin-load	はい	はい				
plugin-load-add	はい	はい				
port	はい	はい	はい		グローバル	いい
port-open-timeout	はい	はい				
preload_buffer_size	はい	はい	はい		両方	はい
Prepared_stmt_count				はい	グローバル	いい
print-defaults	はい					
profiling			はい		両方	はい
profiling_history_size	はい	はい	はい		両方	はい
protocol_version			はい		グローバル	いい
proxy_user			はい		セッション	いい
pseudo_slave_mode			はい		セッション	はい
pseudo_thread_id			はい		セッション	はい
Qcache_free_blocks				はい	グローバル	いい
Qcache_free_memory				はい	グローバル	いい
Qcache_hits				はい	グローバル	いい
Qcache_inserts				はい	グローバル	いい
Qcache_lowmem_prunes				はい	グローバル	いい
Qcache_not_cached				はい	グローバル	いい
Qcache_queries_in_cache				はい	グローバル	いい
Qcache_total_blocks				はい	グローバル	いい
Queries				はい	両方	いい
query_alloc_block_size	はい	はい	はい		両方	はい

名前	コマンド行	オプションファイル	システム変数	ステータス変数	変数スコープ	動的
query_cache_limit	はい	はい	はい		グローバル	はい
query_cache_min_res_unit	はい	はい	はい		グローバル	はい
query_cache_size	はい	はい	はい		グローバル	はい
query_cache_type	はい	はい	はい		両方	はい
query_cache_wlock_invalidate	はい	はい	はい		両方	はい
query_prealloc_size	はい	はい	はい		両方	はい
Questions				はい	両方	いいえ
rand_seed1			はい		セッション	はい
rand_seed2			はい		セッション	はい
range_alloc_block_size	はい	はい	はい		両方	はい
read_buffer_size	はい	はい	はい		両方	はい
read_only	はい	はい	はい		グローバル	はい
read_rnd_buffer_size	はい	はい	はい		両方	はい
relay_log	はい	はい	はい		グローバル	いいえ
relay_log_basename			はい		グローバル	いいえ
relay_log_index	はい	はい	はい		グローバル	いいえ
relay_log_index	はい	はい	はい		グローバル	いいえ
relay-log-info-file	はい	はい				
relay_log_info_file	はい	はい	はい		グローバル	いいえ
relay-log-info-repository	はい	はい				
relay_log_info_repository			はい		グローバル	はい
relay_log_purge	はい	はい	はい		グローバル	はい
relay-log-recovery	はい	はい				
relay_log_recovery	はい	はい	はい		グローバル	異なる
relay_log_space_limit	はい	はい	はい		グローバル	いいえ
remove	はい					
replicate-do-db	はい	はい				
replicate-do-table	はい	はい				
replicate-ignore-db	はい	はい				
replicate-ignore-table	はい	はい				
replicate-rewrite-db	はい	はい				
replicate-same-server-id	はい	はい				
replicate-wild-do-table	はい	はい				
replicate-wild-ignore-table	はい	はい				
report_host	はい	はい	はい		グローバル	いいえ
report_password	はい	はい	はい		グローバル	いいえ
report_port	はい	はい	はい		グローバル	いいえ
report_user	はい	はい	はい		グローバル	いいえ
Rpl_semi_sync_master_clients				はい	グローバル	いいえ
rpl_semi_sync_master_enabled			はい		グローバル	はい
Rpl_semi_sync_master_net_avg_wait_time				はい	グローバル	いいえ
Rpl_semi_sync_master_net_wait_time				はい	グローバル	いいえ

名前	コマンド行	オプションファイル	システム変数	ステータス変数	変数スコープ	動的
Rpl_semi_sync_master_net_waits				はい	グローバル	いい
Rpl_semi_sync_master_no_times				はい	グローバル	いい
Rpl_semi_sync_master_no_tx				はい	グローバル	いい
Rpl_semi_sync_master_status				はい	グローバル	いい
Rpl_semi_sync_master_timefunc_failures				はい	グローバル	いい
rpl_semi_sync_master_timeout			はい		グローバル	はい
rpl_semi_sync_master_trace_level			はい		グローバル	はい
Rpl_semi_sync_master_tx_avg_wait_time				はい	グローバル	いい
Rpl_semi_sync_master_tx_wait_time				はい	グローバル	いい
Rpl_semi_sync_master_tx_waits				はい	グローバル	いい
rpl_semi_sync_master_wait_no_slave			はい		グローバル	はい
Rpl_semi_sync_master_wait_pos_backtraverse				はい	グローバル	いい
Rpl_semi_sync_master_wait_sessions				はい	グローバル	いい
Rpl_semi_sync_master_yes_tx				はい	グローバル	いい
rpl_semi_sync_slave_enabled			はい		グローバル	はい
Rpl_semi_sync_slave_status				はい	グローバル	いい
rpl_semi_sync_slave_trace_level			はい		グローバル	はい
rpl_stop_slave_timeout	はい	はい	はい		グローバル	はい
Rsa_public_key				はい	グローバル	いい
safe-mode	はい	はい				
safe-user-create	はい	はい				
secure_auth	はい	はい	はい		グローバル	はい
secure_file_priv	はい	はい	はい		グローバル	いい
Select_full_join				はい	両方	いい
Select_full_range_join				はい	両方	いい
Select_range				はい	両方	いい
Select_range_check				はい	両方	いい
Select_scan				はい	両方	いい
server_id	はい	はい	はい		グローバル	はい
server_id_bits	はい	はい	はい		グローバル	いい
server_id_bits	はい	はい	はい		グローバル	いい
server_uuid			はい		グローバル	いい
sha256_password_private_key_path			はい		グローバル	いい
sha256_password_public_key_path			はい		グローバル	いい
shared_memory	はい	はい	はい		グローバル	いい
shared_memory_base_name			はい		グローバル	いい
show-slave-auth-info	はい	はい				
simplified_binlog_gtid_recovery	はい	はい	はい		グローバル	いい
skip-character-set-client-handshake	はい	はい				
skip-concurrent-insert	はい	はい				
- 変数: concurrent_insert						
skip-event-scheduler	はい	はい				

名前	コマンド行	オプションファイル	システム変数	ステータス変数	変数スコープ	動的
skip_external_locking	はい	はい	はい		グローバル	いいえ
skip-grant-tables	はい	はい				
skip-host-cache	はい	はい				
skip_name_resolve	はい	はい	はい		グローバル	いいえ
skip-ndbcluster	はい	はい				
skip_networking	はい	はい	はい		グローバル	いいえ
skip-new	はい	はい				
skip-partition	はい	はい				
skip_show_database	はい	はい	はい		グローバル	いいえ
skip-slave-start	はい	はい				
skip-ssl	はい	はい				
skip-stack-trace	はい	はい				
skip-symbolic-links	はい					
skip-thread-priority	はい	はい				
slave_allow_batching	はい	はい	はい		グローバル	はい
slave-checkpoint-group	はい	はい				
slave_checkpoint_group	はい	はい	はい		グローバル	はい
slave-checkpoint-period	はい	はい				
slave_checkpoint_period	はい	はい	はい		グローバル	はい
slave_compressed_protocol	はい	はい	はい		グローバル	はい
slave_exec_mode	はい	はい	はい		グローバル	はい
Slave_heartbeat_period				はい	グローバル	いいえ
Slave_last_heartbeat				はい	グローバル	いいえ
slave_load_tmpdir	はい	はい	はい		グローバル	いいえ
slave-max-allowed-packet	はい	はい				
slave_max_allowed_packet			はい		グローバル	はい
slave_net_timeout	はい	はい	はい		グローバル	はい
Slave_open_temp_tables				はい	グローバル	いいえ
slave-parallel-workers	はい	はい				
slave_parallel_workers	はい		はい		グローバル	はい
slave-pending-jobs-size-max	はい					
slave_pending_jobs_size_max			はい		グローバル	はい
Slave_received_heartbeats				はい	グローバル	いいえ
Slave_retried_transactions				はい	グローバル	いいえ
slave-rows-search-algorithms	はい	はい				
slave_rows_search_algorithms			はい		グローバル	はい
Slave_running				はい	グローバル	いいえ
slave_skip_errors	はい	はい	はい		グローバル	いいえ
slave-sql-verify-checksum	はい	はい				
slave_sql_verify_checksum			はい		グローバル	はい
slave_transaction_retries	はい	はい	はい		グローバル	はい

名前	コマンド行	オプションファイル	システム変数	ステータス変数	変数スコープ	動的
slave_type_conversions	はい	はい	はい		グローバル	いい
Slow_launch_threads				はい	両方	いい
slow_launch_time	はい	はい	はい		グローバル	はい
Slow_queries				はい	両方	いい
slow_query_log	はい	はい	はい		グローバル	はい
slow_query_log_file	はい	はい	はい		グローバル	はい
slow-start-timeout	はい	はい				
socket	はい	はい	はい		グローバル	いい
sort_buffer_size	はい	はい	はい		両方	はい
Sort_merge_passes				はい	両方	いい
Sort_range				はい	両方	いい
Sort_rows				はい	両方	いい
Sort_scan				はい	両方	いい
sporadic-binlog-dump-fail	はい	はい				
sql_auto_is_null			はい		両方	はい
sql_big_selects			はい		両方	はい
sql_big_tables			はい		両方	はい
sql_buffer_result			はい		両方	はい
sql_log_bin			はい		両方	はい
sql_log_off			はい		両方	はい
sql_low_priority_updates			はい		両方	はい
sql_max_join_size			はい		両方	はい
sql_mode	はい	はい	はい		両方	はい
sql_notes			はい		両方	はい
sql_quote_show_create			はい		両方	はい
sql_safe_updates			はい		両方	はい
sql_select_limit			はい		両方	はい
sql_slave_skip_counter			はい		グローバル	はい
sql_warnings			はい		両方	はい
ssl	はい	はい				
Ssl_accept_renegotiates				はい	グローバル	いい
Ssl_accepts				はい	グローバル	いい
ssl_ca	はい	はい	はい		グローバル	いい
Ssl_callback_cache_hits				はい	グローバル	いい
ssl_capath	はい	はい	はい		グローバル	いい
ssl_cert	はい	はい	はい		グローバル	いい
Ssl_cipher				はい	両方	いい
ssl_cipher	はい	はい	はい		グローバル	いい
Ssl_cipher_list				はい	両方	いい
Ssl_client_connects				はい	グローバル	いい
Ssl_connect_renegotiates				はい	グローバル	いい
ssl_crl	はい	はい	はい		グローバル	いい
ssl_crlpath	はい	はい	はい		グローバル	いい

名前	コマンド行	オプションファイル	システム変数	ステータス変数	変数スコープ	動的
Ssl_ctx_verify_depth				はい	グローバル	いいえ
Ssl_ctx_verify_mode				はい	グローバル	いいえ
Ssl_default_timeout				はい	両方	いいえ
Ssl_finished_accepts				はい	グローバル	いいえ
Ssl_finished_connects				はい	グローバル	いいえ
ssl_key	はい	はい	はい		グローバル	いいえ
Ssl_server_not_after				はい	両方	いいえ
Ssl_server_not_before				はい	両方	いいえ
Ssl_session_cache_hits				はい	グローバル	いいえ
Ssl_session_cache_misses				はい	グローバル	いいえ
Ssl_session_cache_mode				はい	グローバル	いいえ
Ssl_session_cache_overflows				はい	グローバル	いいえ
Ssl_session_cache_size				はい	グローバル	いいえ
Ssl_session_cache_timeouts				はい	グローバル	いいえ
Ssl_sessions_reused				はい	両方	いいえ
Ssl_used_session_cache_entries				はい	グローバル	いいえ
Ssl_verify_depth				はい	両方	いいえ
Ssl_verify_mode				はい	両方	いいえ
Ssl_version				はい	両方	いいえ
standalone	はい	はい				
storage_engine			はい		両方	はい
stored_program_cache	はい	はい	はい		グローバル	はい
super-large-pages	はい	はい				
symbolic-links	はい	はい				
sync_binlog	はい	はい	はい		グローバル	はい
sync_frm	はい	はい	はい		グローバル	はい
sync_master_info	はい	はい	はい		グローバル	はい
sync_relay_log	はい	はい	はい		グローバル	はい
sync_relay_log_info	はい	はい	はい		グローバル	はい
sysdate-is-now	はい	はい				
system_time_zone			はい		グローバル	いいえ
table_definition_cache			はい		グローバル	はい
Table_locks_immediate				はい	グローバル	いいえ
Table_locks_waited				はい	グローバル	いいえ
table_open_cache			はい		グローバル	はい
Table_open_cache_hits				はい	両方	いいえ
table_open_cache_instances			はい		グローバル	いいえ
Table_open_cache_misses				はい	両方	いいえ
Table_open_cache_overflows				はい	両方	いいえ
tc-heuristic-recover	はい	はい				
Tc_log_max_pages_used				はい	グローバル	いいえ
Tc_log_page_size				はい	グローバル	いいえ
Tc_log_page_waits				はい	グローバル	いいえ

名前	コマンド行	オプションファイル	システム変数	ステータス変数	変数スコープ	動的
temp-pool	はい	はい				
thread_cache_size	はい	はい	はい		グローバル	はい
thread_concurrency	はい	はい	はい		グローバル	いい
thread_handling	はい	はい	はい		グローバル	いい
thread_pool_algorithm	はい	はい	はい		グローバル	いい
thread_pool_high_priority_connection	はい	はい	はい		両方	はい
thread_pool_max_unused_threads	はい	はい	はい		グローバル	はい
thread_pool_prio_kickup_timer	はい	はい	はい		両方	はい
thread_pool_size	はい	はい	はい		グローバル	いい
thread_pool_stall_limit	はい	はい	はい		グローバル	はい
thread_stack	はい	はい	はい		グローバル	いい
Threads_cached				はい	グローバル	いい
Threads_connected				はい	グローバル	いい
Threads_created				はい	グローバル	いい
Threads_running				はい	グローバル	いい
time_format			はい		グローバル	いい
time_zone			はい		両方	はい
timed_mutexes	はい	はい	はい		グローバル	はい
timestamp			はい		セッション	はい
tmp_table_size	はい	はい	はい		両方	はい
tmpdir	はい	はい	はい		グローバル	いい
transaction_alloc_block_size	はい	はい	はい		両方	はい
transaction_allow_batching			はい		セッション	はい
transaction-isolation	はい	はい				
- 変数: tx_isolation						
transaction_prealloc_size	はい	はい	はい		両方	はい
transaction-read-only	はい	はい				
- 変数: tx_read_only						
tx_isolation			はい		両方	はい
tx_read_only			はい		両方	はい
unique_checks			はい		両方	はい
updatable_views_with_limit	はい	はい	はい		両方	はい
Uptime				はい	グローバル	いい
Uptime_since_flush_status				はい	グローバル	いい
user	はい	はい				
validate-password	はい	はい				
validate_password_dictionary_file			はい		グローバル	異な
validate_password_length			はい		グローバル	はい
validate_password_mixed_case_count			はい		グローバル	はい
validate_password_number_count			はい		グローバル	はい
validate_password_policy			はい		グローバル	はい
validate_password_special_char_count			はい		グローバル	はい
validate_user_plugins			はい		グローバル	いい

名前	コマンド行	オプションファイル	システム変数	ステータス変数	変数スコープ	動的
verbose	はい	はい				
version			はい		グローバル	いいえ
version_comment			はい		グローバル	いいえ
version_compile_machine			はい		グローバル	いいえ
version_compile_os			はい		グローバル	いいえ
wait_timeout	はい	はい	はい		両方	はい
warning_count			はい		セッション	いいえ

Notes:

1. このオプションは動的ですが、サーバーのみがこの情報を設定するようにしてください。この変数の値は手動で設定しないでください。

5.1.2 サーバー構成のデフォルト値

MySQL Server には多くの操作パラメータがあり、これらはコマンド行オプションまたは構成ファイル (オプションファイル) を使用して、サーバー起動時に変更できます。多くのパラメータを実行時に変更することもできます。起動時または実行時にパラメータを設定することに関する一般的な指示については、セクション5.1.3「サーバーコマンドオプション」およびセクション5.1.4「サーバーシステム変数」を参照してください。

5.1.2.1 サーバーのデフォルト値への変更

MySQL 5.6.6 以降では、MySQL Server のいくつかのデフォルトパラメータが、前のリリースのデフォルト値と異なっています。これらの変更の目的は、初期設定のままで優れたパフォーマンスを提供し、データベース管理者が設定を手動で変更することの必要性を低下させることです。

パラメータが別の固定のデフォルト値を持つ場合もあります。あるいは、固定値を使用するのではなく、ほかの関連するパラメータやサーバーホスト構成に基づく数式を使用して、サーバーが起動時にパラメータを自動サイズ設定する場合もあります。たとえば、back_log の設定は、以前のデフォルトの 50 で、max_connections の値に比例する量に応じて上方に調整されます。自動サイズ設定の背後には、固定値よりも適切だと思われるパラメータ設定について、決定を下すために利用できる情報をサーバーが持つ場合、その決定を下すという考え方があります。

次の表は、デフォルトへの変更を要約したものです。「バージョン」カラムは、それぞれのデフォルトが変更されたバージョンを示します。自動サイズ設定される変数の場合、変数の主な説明にはサイズ設定アルゴリズムに関する追加の詳細情報が提供されています。セクション5.1.4「サーバーシステム変数」、セクション14.12「InnoDB の起動オプションおよびシステム変数」を参照してください。これらすべてのデフォルト設定は、サーバー起動時に明示的な値を指定することによってオーバーライドできます。

表 5.2 MySQL 5.6 でのサーバーデフォルトへの変更

パラメータ	古いデフォルト	新しいデフォルト	バージョン
back_log	50	max_connections を使用した自動サイズ設定	5.6.6
binlog_checksum	NONE	CRC32	5.6.6
--binlog-row-event-max-size	1024	8192	5.6.6
flush_time	1800 (Windows の場合)	0	5.6.6
host_cache_size	128	max_connections を使用した自動サイズ設定	5.6.8
innodb_autoextend_increment	8	64	5.6.6
innodb_buffer_pool_instances	1	8 (プラットフォームに依存)	5.6.6
innodb_concurrency_tickets	500	5000	5.6.6
innodb_data_file_path	ibdata1:10M:autoextend	ibdata1:12M:autoextend	5.6.7
innodb_file_per_table	0	1	5.6.6

パラメータ	古いデフォルト	新しいデフォルト	バージョン
innodb_log_file_size	5MB	48MB	5.6.8
innodb_old_blocks_time	0	1000	5.6.6
innodb_open_files	300	innodb_file_per_table、table_open_cache を使用した自動サイズ設定	5.6.6
innodb_stats_on_metadata	ON	OFF	5.6.6
join_buffer_size	128KB	256KB	5.6.6
max_allowed_packet	1MB	4MB	5.6.6
max_connect_errors	10	100	5.6.6
open_files_limit	0	max_connections を使用した自動サイズ設定	5.6.8
performance_schema	OFF	ON	5.6.6
performance_schema_events_waits_history_long_size	10000	自動サイズ設定	5.6.6
performance_schema_events_waits_history_size	10	自動サイズ設定	5.6.6
performance_schema_max_cond_instances	1000	自動サイズ設定	5.6.6
performance_schema_max_file_instances	10000	自動サイズ設定	5.6.6
performance_schema_max_mutex_instances	1000000	自動サイズ設定	5.6.6
performance_schema_max_rwlock_instances	1000000	自動サイズ設定	5.6.6
performance_schema_max_table_handles	100000	自動サイズ設定	5.6.6
performance_schema_max_table_instances	50000	自動サイズ設定	5.6.6
performance_schema_max_thread_instances	1000	自動サイズ設定	5.6.6
query_cache_size	0	1M	5.6.8
query_cache_type	ON	OFF	5.6.8
secure_auth	OFF	ON	5.6.7
sql_mode	'' (空の文字列)	NO_ENGINE_SUBSTITUTION	5.6.6
sync_master_info	0	10000	5.6.6
sync_relay_log	0	10000	5.6.6
sync_relay_log_info	0	10000	5.6.6
table_definition_cache	400	table_open_cache を使用した自動サイズ設定	5.6.8
table_open_cache	400	2000	5.6.8
thread_cache_size	0	max_connections を使用した自動サイズ設定	5.6.8

MySQL 5.6.6 では、innodb_checksum_algorithm のデフォルトは INNODB から CRC32 に変更されました。互換性の理由から、デフォルトは 5.6.7 で INNODB に戻されました。

5.1.2.2 サンプルのデフォルトサーバー構成ファイルの使用

MySQL 5.6.8 以降では、mysql_install_db は UNIX プラットフォーム上で、my.cnf という名前のデフォルトオプションファイルを基本インストールディレクトリに作成します。このファイルは my-default.cnf という名前の配布パッケージに含まれるテンプレートから作成されます。テンプレートは基本インストールディレクトリ内またはその配下から見つけることができます。mysqld_safe を使用して開始すると、サーバーはデフォルトで my.cnf ファイルを使用します。my.cnf がすでに存在する場合、mysql_install_db はそのファイルが使用中だと認識し、my-new.cnf という名前の新しいファイルを代わりに書き込みます。

1 つの例外を除き、デフォルトのオプションファイル内の設定はコメント化されて無効になっています。例外は、このファイルが sql_mode システム変数をデフォルトの NO_ENGINE_SUBSTITUTION から STRICT_TRANS_TABLES も含むように変更するということです。

```
sql_mode=NO_ENGINE_SUBSTITUTION,STRICT_TRANS_TABLES
```

この設定により、トランザクションテーブルを変更する操作での不良データに対して警告でなくエラーを発生させるサーバー構成が提供されます。セクション5.1.7「サーバー SQL モード」を参照してください。

my-default.cnf テンプレートは、MySQL の以前の配布で提供されていた古いサンプルオプションファイル (my-small.cnf、my-medium.cnf、my-large.cnf、および my-huge.cnf) を置き換えます。MySQL 5.6.8 以降では、これらの古いファイルは配布されなくなりました。

Windows では、MySQL Installer がユーザーと対話し、my.ini という名前のファイルをデフォルトオプションファイルとして基本インストールディレクトリに作成します。Windows 上で Zip アーカイブからインストールする場合、基本インストールディレクトリ内の my-default.ini テンプレートファイルを my.ini にコピーし、後者をデフォルトオプションファイルとして使用できます。

> **注記**
>
> Windows では、.ini または .cnf というオプションのファイル拡張子が表示されない場合もあります。

すべてのプラットフォームで、インストールプロセスの完了後、デフォルトオプションファイルを編集して、サーバーによって使用されるパラメータをいつでも変更できます。たとえば、行の先頭の # 文字によってコメント化されたファイル内のパラメータ設定を使用するには、# を削除し、必要に応じてパラメータ値を変更します。設定を無効にするには、行の先頭に # を追加するか、設定を削除します。

オプションファイルの形式および構文についての追加情報は、セクション4.2.6「オプションファイルの使用」を参照してください。

MySQL 5.6.8 以前では、MySQL の配布には、MySQL Server の調整の基礎として使用できるいくつかのサンプルオプションファイルが含まれていました。my-small.cnf、my-medium.cnf、my-large.cnf、および my-huge.cnf という名前のファイルを探してください。これらは小規模、中規模、大規模、および非常に大規模なシステム用のサンプルファイルです。Windows の場合、拡張子は .cnf でなく .ini です。

バイナリの配布については、インストールディレクトリ内またはその配下のサンプルファイルを探してください。ソースの配布がある場合、support-files ディレクトリ内を探してください。サンプルファイルを基本構成ファイルとして使用するには、ファイルのコピーを名前変更し、コピーを適切な場所に配置します。名前および適切な場所に関しては、セクション4.2.6「オプションファイルの使用」の一般的な情報を参照してください。

5.1.3 サーバーコマンドオプション

mysqld サーバーを起動するときに、セクション4.2.3「プログラムオプションの指定」に記載されているいずれかの方法で、プログラムオプションを指定できます。もっとも一般的な方法は、オプションファイルまたはコマンド行でオプションを提供するやり方です。ただし、ほとんどの場合では、サーバーが毎回実行するときサーバーが必ず同じオプションを使用します。これを確実に行う最適な方法は、オプションファイルにオプションを一覧表示することです。セクション4.2.6「オプションファイルの使用」を参照してください。このセクションではオプションファイルの形式および構文についても説明します。

mysqld は [mysqld] および [server] グループからオプションを読み取ります。mysqld_safe は [mysqld]、[server]、[mysqld_safe]、および [safe_mysqld] グループからオプションを読み取ります。mysql.server は [mysqld] および [mysql.server] グループからオプションを読み取ります。

Embedded MySQL Server は通常、[server]、[embedded]、および [xxxxx_SERVER] グループからオプションを読み取り、ここで xxxxx はこのサーバーを組み込んでいるアプリケーション名です。

mysqld には多くのコマンドオプションがあります。簡単なサマリーについては、mysqld --help を実行します。完全なリストを表示するには、mysqld --verbose --help を使用します。

次に、もっとも一般的なサーバーオプションの一部を示します。その他のオプションは、ほかのセクションに記載されています。

- セキュリティーに影響するオプション。セクション6.1.4「セキュリティー関連の mysqld オプションおよび変数」を参照してください。
- SSL 関連オプション。セクション6.3.10.4「SSL コマンドのオプション」を参照してください。
- バイナリログ制御オプション。セクション5.2.4「バイナリログ」を参照してください。
- レプリケーション関連オプション。セクション17.1.4「レプリケーションおよびバイナリロギングのオプションと変数」を参照してください。
- プラガブルストレージエンジンなどのプラグインをロードするためのオプション。セクション5.1.8.1「プラグインのインストールおよびアンインストール」を参照してください。
- 特定のストレージエンジンに固有のオプション。セクション14.12「InnoDB の起動オプションおよびシステム変数」およびセクション15.2.1「MyISAM 起動オプション」を参照してください。

また、このセクションの最後に記載されているように、サーバーシステム変数の値を設定するとき、変数名をオプションとして使用することもできます。

一部のオプションはバッファーまたはキャッシュのサイズを制御します。所定のバッファーについて、サーバーは内部データ構造を割り当てる必要がある場合もあります。これらの構造は、バッファーに割り当てられた合計メモリーから割り当てられ、必要なスペースの量はプラットフォームに依存することがあります。つまり、バッファーサイズを制御するオプションに値を割り当てたとき、実際に使用可能なスペースの量が、割り当てられた値と異なる場合もあることを意味します。一部の場合では、この量は割り当てられた値より少ないこともあります。また、サーバーが値を上方に調整することも考えられます。たとえば、最小値が 1024 のオプションに値 0 を割り当てた場合、サーバーは値を 1024 に設定します。

バッファーサイズ、長さ、およびスタックサイズの値は、別途指定しないかぎりバイト単位で指定されます。

一部のオプションはファイル名の値を取ります。別途指定しないかぎり、値が相対パス名であれば、デフォルトのファイルの場所はデータディレクトリです。場所を明示的に指定するには、絶対パス名を使用します。たとえばデータディレクトリが /var/mysql/data だとします。ファイル値のオプションが相対パス名として指定された場合、ファイルは /var/mysql/data の下に配置されます。値が絶対パス名である場合、その場所はパス名によって指定されます。

- --help, -?

コマンド行形式	--help

短いヘルプメッセージを表示して終了します。詳細メッセージを表示するには、--verbose および --help の両方のオプションを使用します。

- --allow-suspicious-udfs

コマンド行形式	--allow-suspicious-udfs
型	ブール
デフォルト	FALSE

このオプションは、メイン関数に xxx 記号のみを持つユーザー定義関数をロードできるかどうかを制御します。デフォルトでは、このオプションはオフで、少なくとも 1 つの補助記号を持つ UDF のみをロードできます。これにより、正当な UDF を含むもの以外の共有オブジェクトファイルから関数をロードしないようにします。セクション24.3.2.6「ユーザー定義関数のセキュリティー上の予防措置」を参照してください。

- --ansi

コマンド行形式	--ansi

MySQL 構文の代わりに標準 (ANSI) SQL 構文を使用します。サーバー SQL モードをさらに正確に制御するには、代わりに --sql-mode オプションを使用します。セクション1.7「MySQL の標準への準拠」、セクション5.1.7「サーバー SQL モード」を参照してください。

- --basedir=path, -b path

コマンド行形式	--basedir=path
システム変数	basedir
スコープ	グローバル
動的	いいえ
型	ディレクトリ名

MySQL インストールディレクトリへのパス。すべてのパスは、通常はこのディレクトリを基準として解決されます。

- --big-tables

コマンド行形式	--big-tables
システム変数	big_tables
スコープ	グローバル、セッション
動的	はい

型	ブール
デフォルト	OFF

すべての一時的セットをファイルに保存することによって、大きい結果セットを有効にします。このオプションによって、ほとんどの「table full」エラーを防ぎますが、インメモリーテーブルだけで十分なクエリーの速度が低下します。MySQL 3.23.2 以降では、サーバーは小さい一時テーブルについてはメモリーを使用し、必要な場合にディスクテーブルに切り換えることによって、大きい結果セットを自動的に処理できます。

- --bind-address=addr

コマンド行形式	--bind-address=addr
システム変数 (≥ 5.6.1)	bind_address
スコープ (≥ 5.6.1)	グローバル
動的 (≥ 5.6.1)	いいえ
型	文字列
デフォルト (≥ 5.6.6)	*
デフォルト (≤ 5.6.5)	0.0.0.0

MySQL Server は、TCP/IP 接続について単一ネットワークソケットを listen します。このソケットは単一アドレスにバインドされますが、あるアドレスを複数のネットワークインタフェースにマップできます。アドレスを指定するには、サーバー起動時に --bind-address=addr オプションを使用します。ここで、addr は IPv4 または IPv6 アドレスあるいはホスト名です。addr がホスト名の場合、サーバーはこの名前を IP アドレスに解決し、そのアドレスにバインドします。

サーバーはさまざまなタイプのアドレスを次のように処理します。

- アドレスが * の場合、サーバーホストが IPv6 アドレスをサポートする場合はすべてのサーバーホストの IPv6 および IPv4 インタフェース上の TCP/IP 接続を受け入れ、そうでない場合はすべての IPv4 アドレスの TCP/IP 接続を受け入れます。すべてのサーバーインタフェース上の IPv4 および IPv6 の両方の接続を許可するには、このアドレスを使用します。この値は、MySQL 5.6.6 以降で許可されています (またデフォルトです)。
- アドレスが 0.0.0.0 の場合、サーバーはすべてのサーバーホスト IPv4 インタフェース上の TCP/IP 接続を受け入れます。これは MySQL 5.6.6 以前のデフォルトです。
- アドレスが :: の場合、サーバーはすべてのサーバーホスト IPv4 および IPv6 インタフェース上の TCP/IP 接続を受け入れます。
- アドレスが IPv4 にマップ済みのアドレスの場合、サーバーは IPv4 または IPv6 のいずれかの形式で、そのアドレスの TCP/IP 接続を受け入れます。たとえば、サーバーが ::ffff:127.0.0.1 にバインドされている場合、クライアントは --host=127.0.0.1 または --host=::ffff:127.0.0.1 のいずれかを使用して接続できます。
- アドレスが「通常の」IPv4 または IPv6 アドレスの場合 (127.0.0.1 や ::1 など)、サーバーはその IPv4 または IPv6 アドレスについてのみ TCP/IP 接続を受け入れます。

サーバーを特定のアドレスにバインドする予定の場合、そのアドレスに接続するために使用できる管理者権限を持つアカウントが mysql.user 付与テーブルに含まれていることを確認します。そうでない場合、サーバーをシャットダウンできません。たとえば、サーバーを * にバインドしている場合、すべての既存のアカウントを使用して接続できます。ただし、サーバーを ::1 にバインドしている場合、そのアドレスの接続のみ受け入れます。この場合、'root'@'::1' アカウントが mysql.user テーブルに存在することをまず確認して、サーバーに接続してシャットダウンできることをたしかめます。

- --binlog-format={ROW|STATEMENT|MIXED}

コマンド行形式	--binlog-format=format
システム変数	binlog_format
スコープ	グローバル、セッション
動的	はい
型	列挙
デフォルト (≥ 5.6.10-ndb-7.3.1)	MIXED
デフォルト	STATEMENT

有効な値	ROW STATEMENT MIXED

行ベース、ステートメントベース、または複合型のレプリケーションのいずれを使用するか指定します。ステートメントベースは MySQL 5.6 のデフォルトです。セクション17.1.2「レプリケーション形式」を参照してください。

一部の状況では、この変数を実行時に変更できなかったり、レプリケーションの失敗の原因になったりします。詳細については、セクション5.2.4.2「バイナリログ形式の設定」を参照してください。

バイナリロギングを有効にしないでバイナリロギング形式を設定すると、binlog_format グローバルシステム変数が設定されて、警告がログに記録されます。

- --bootstrap

コマンド行形式	--bootstrap

このオプションは、MySQL Server 全体を開始させることなく MySQL 特権テーブルを作成するために、mysql_install_db プログラムによって使用されます。

MySQL 5.6.6 以降では、このオプションが使用されるとレプリケーションおよびグローバルトランザクション ID が常に自動的に無効化されます (バグ #1332602)。セクション17.1.3「グローバルトランザクション識別子を使用したレプリケーション」を参照してください。

- --character-sets-dir=path

コマンド行形式	--character-sets-dir=path
システム変数	character_sets_dir
スコープ	グローバル
動的	いいえ
型	ディレクトリ名

文字セットがインストールされているディレクトリ。セクション10.5「文字セットの構成」を参照してください。

- --character-set-client-handshake

コマンド行形式	--character-set-client-handshake
型	ブール
デフォルト	TRUE

クライアントによって送信された文字セット情報を無視しません。クライアント情報を無視して、サーバーのデフォルトの文字セットを使用するには、--skip-character-set-client-handshake を使用します。これにより、MySQL は MySQL 4.0. のように動作します。

- --character-set-filesystem=charset_name

コマンド行形式	--character-set-filesystem=name
システム変数	character_set_filesystem
スコープ	グローバル、セッション
動的	はい
型	文字列
デフォルト	binary

ファイルシステムの文字セット。このオプションは、character_set_filesystem のシステム変数を設定します。

- --character-set-server=charset_name, -C charset_name

コマンド行形式	--character-set-server
システム変数	character_set_server
スコープ	グローバル、セッション
動的	はい
型	文字列
デフォルト	latin1

charset_name をデフォルトの文字セットとして使用します。セクション10.5「文字セットの構成」を参照してください。このオプションを使用してデフォルト以外の文字セットを指定する場合、照合順序を指定するために --collation-server も使用します。

- --chroot=path, -r path

コマンド行形式	--chroot=dir_name
型	ディレクトリ名

chroot() のシステムコールを使用して、mysqld サーバーを起動中にクローズ環境にします。これは推奨されるセキュリティー対策です。このオプションを使用すると、LOAD DATA INFILE および SELECT ... INTO OUTFILE がいくらか制限されることに注意してください。

- --collation-server=collation_name

コマンド行形式	--collation-server
システム変数	collation_server
スコープ	グローバル、セッション
動的	はい
型	文字列
デフォルト	latin1_swedish_ci

collation_name をサーバーのデフォルトの照合順序として使用します。セクション10.5「文字セットの構成」を参照してください。

- --console

コマンド行形式	--console
プラットフォーム固有	Windows

(Windows のみ。)このオプションが使用されている場合、エラーログメッセージを stderr に書き込み、stdout. mysqld はコンソールウィンドウを閉じません。

--log-error および --console の両方が指定されている場合、--log-error が優先されます。サーバーはログファイルに書き込みますが、コンソールには書き込みません。

- --core-file

コマンド行形式	--core-file
型	ブール
デフォルト	OFF

mysqld が異常終了した場合にコアファイルを作成します。コアファイルの名前および場所はシステムに依存します。Linux の場合、core.pid という名前のコアファイルがプロセスの現在の作業ディレクトリに書き込まれ、これは mysqld のデータディレクトリです。pid はサーバープロセスのプロセス ID を表します。OS X の場合、core.pid という名前のコアファイルが /cores ディレクトリに書き込まれます。Solaris の場合、coreadm コマンドを使用して、コアファイルの書き込み先と名前を指定する方法を指定します。

一部のシステムでコアファイルを取得するには、mysqld_safe に --core-file-size オプションを指定する必要もあります。セクション4.3.2「mysqld_safe — MySQL サーバー起動スクリプト」を参照してください。Solaris などの一部のシステムでは、--user オプションも使用しないとコアファイルを取得できません。追加の制限また

は制約がある場合もあります。たとえば、サーバーを起動する前に ulimit -c unlimited を実行することが必要な場合もあります。システムのドキュメントを参照してください。

- --datadir=path, -h path

コマンド行形式	--datadir=path
システム変数	datadir
スコープ	グローバル
動的	いいえ
型	ディレクトリ名

データディレクトリへのパス。

- --debug[=debug_options], -# [debug_options]

コマンド行形式	--debug[=debug_options]
システム変数	debug
スコープ	グローバル、セッション
動的	はい
型	文字列
デフォルト (Windows)	d:t:i:O,\mysqld.trace
デフォルト (Unix)	d:t:i:o,/tmp/mysqld.trace

MySQL が -DWITH_DEBUG=1 で構成されている場合、このオプションを使用して、mysqld の動作のトレースファイルを取得できます。一般的な debug_options 文字列は d:t:o,file_name です。デフォルトは、UNIX の場合は d:t:i:o,/tmp/mysqld.trace、Windows の場合は d:t:i:O,\mysqld.trace です。

-DWITH_DEBUG=1 を使用して MySQL にデバッグサポートを構成することにより、サーバーを起動するときに --debug="d,parser_debug" オプションを使用できるようになります。これにより、SQL ステートメントの処理に使用される Bison パーサーが、パーサートレースをサーバーの標準エラー出力にダンプします。一般的に、この出力はエラーログに書き込まれます。

このオプションは複数回指定されることがあります。+ または - で開始される値が以前の値に加算または減算されます。たとえば、--debug=T --debug=+P と指定すると、値は P:T に設定されます。

詳細については、セクション24.4.3「DBUG パッケージ」を参照してください。

- --debug-sync-timeout[=N]

コマンド行形式	--debug-sync-timeout[=#]
型	数値

テストおよびデバッグのための Debug Sync 機能が有効かどうかを制御します。Debug Sync を使用するには MySQL が -DENABLE_DEBUG_SYNC=1 オプションで構成されている必要があります (セクション 2.9.4「MySQL ソース構成オプション」を参照)。Debug Sync がコンパイルされていない場合、このオプションは使用できません。オプション値は秒単位のタイムアウトです。デフォルト値は 0 で、Debug Sync を無効にします。これを有効にするには、0 より大きい値を指定してください。この値は、個々の同期点についてのデフォルトのタイムアウトになります。オプションが値なしで指定された場合、タイムアウトは 300 秒に設定されます。

Debug Sync 機能および同期点の使用方法についての説明は、「MySQL Internals: Test Synchronization」を参照してください。

- --default-authentication-plugin=plugin_name

コマンド行形式	--default-authentication-plugin=plugin_name
導入	5.6.6
型	列挙
デフォルト	mysql_native_password
有効な値	mysql_native_password

	sha256_password

このオプションは、デフォルトの認証プラグインを設定します。許可される値は mysql_native_password (MySQL ネイティブパスワードを使用する) および sha256_password (SHA-256 パスワードを使用する) です。これらのプラグインについての詳細は、セクション6.3.8.1「ネイティブ認証プラグイン」およびセクション6.3.8.4「SHA-256 認証プラグイン」を参照してください。このオプションは MySQL 5.6.6 で追加されました。

> **注記**
>
> MySQL 5.6.17 以前でこのオプションを使用して、デフォルトの認証プラグインを mysql_native_password 以外の値に変更した場合、MySQL 5.5.7 より古いクライアントは認証プロトコルへの変更の結果を認識できないため接続できなくなります。

--default-authentication-plugin 値は、サーバー操作の次の側面に影響します。

- プラグインの名前を IDENTIFIED WITH 句で明示的に指定しない CREATE USER および GRANT ステートメントによって作成された新規アカウントに対して、サーバーが割り当てる認証プラグインを決定します。
- 起動時に old_passwords システム変数を、デフォルトプラグインによって必要となるパスワードハッシュ方式と整合性のある値に設定します。old_passwords 値は、CREATE USER および GRANT の IDENTIFIED BY 句で指定されたパスワードのハッシュと、PASSWORD() 関数の引数として指定されるパスワードに影響します。
- 次のいずれかのステートメントで作成されるアカウントについて、サーバーはアカウントにデフォルト認証プラグインを関連付け、old_passwords の値によってハッシュされた所定のパスワードをアカウントに割り当てます。

```
CREATE USER ... IDENTIFIED BY 'cleartext password';
GRANT ...  IDENTIFIED BY 'cleartext password';
```

- 次のいずれかのステートメントで作成されるアカウントについて、デフォルト認証プラグインで必要なハッシュ形式を使用してパスワードハッシュが暗号化されていない場合、ステートメントは失敗します。そうでない場合、サーバーはアカウントにデフォルト認証プラグインを関連付け、アカウントに所定のパスワードハッシュを割り当てます。

```
CREATE USER ... IDENTIFIED BY PASSWORD 'encrypted password';
GRANT ...  IDENTIFIED BY PASSWORD 'encrypted password';
```

- --default-storage-engine=type

コマンド行形式	--default-storage-engine=name
システム変数	default_storage_engine
スコープ	グローバル、セッション
動的	はい
型	列挙
デフォルト	InnoDB

テーブルのデフォルトストレージエンジンを設定します。第15章「代替ストレージエンジン」を参照してください。MySQL 5.6.3 以降では、このオプションは永続的なテーブルについてのみストレージエンジンを設定します。TEMPORARY テーブルについてストレージエンジンを設定するには、default_tmp_storage_engine システム変数を設定します。

サーバー起動時のデフォルトストレージエンジンを無効にした場合、永続テーブルと TEMPORARY テーブルの両方のデフォルトエンジンを別のエンジンに設定しなければならず、そうしないとサーバーは起動しません。

- --default-time-zone=timezone

コマンド行形式	--default-time-zone=name
型	文字列

デフォルトのサーバータイムゾーンを設定します。このオプションは、グローバルな time_zone システム変数を設定します。このオプションを指定しない場合、デフォルトのタイムゾーンは、(system_time_zone システム変数の値によって指定される) システムのタイムゾーンと同一になります。

- --defaults-extra-file=file_name

  このオプションファイルは、グローバルオプションファイルのあとに読み取りますが、(UNIX では) ユーザーオプションファイルの前に読み取るようにしてください。ファイルが存在しないかアクセスできない場合、エラーが発生します。file_name は、フルパス名でなく相対パス名として指定された場合、現行ディレクトリを基準にして解釈されます。

- --defaults-file=file_name

  指定されたオプションファイルのみ使用します。ファイルが存在しないかアクセスできない場合、エラーが発生します。file_name は、フルパス名でなく相対パス名として指定された場合、現行ディレクトリを基準にして解釈されます。

- --defaults-group-suffix=str

  通常のオプショングループだけでなく、通常の名前に str のサフィクスが付いたグループも読み取ります。たとえば、mysqld は通常 [mysqld] グループを読み取ります。--defaults-group-suffix=_other オプションを指定した場合、mysqld は [mysqld_other] グループも読み取ります。

- --delay-key-write[={OFF|ON|ALL}]

コマンド行形式	--delay-key-write[=name]
システム変数	delay_key_write
スコープ	グローバル
動的	はい
型	列挙
デフォルト	ON
有効な値	ON OFF ALL

  鍵の遅延書き込みを使用する方法を指定します。鍵の遅延書き込みにより、MyISAM テーブルについて、書き込みと書き込みの間に鍵バッファーがフラッシュされなくなります。OFF は、鍵の遅延書き込みを無効にします。ON は、DELAY_KEY_WRITE オプションを付けて作成されたテーブルに対して鍵の遅延書き込みを有効にします。ALL は、すべての MyISAM テーブルに対する鍵の書き込みを遅延させます。セクション8.11.2「サーバーパラメータのチューニング」およびセクション15.2.1「MyISAM 起動オプション」を参照してください。

  > 注記
  >
  > この変数を ALL に設定した場合、MyISAM テーブルの使用中に、このテーブルを別のプログラム内 (別の MySQL Server または myisamchk など) から使用しないでください。これを行うと、インデックスが破損します。

- --des-key-file=file_name

コマンド行形式	--des-key-file=file_name

  このファイルからデフォルトの DES 鍵を読み取ります。これらの鍵は、DES_ENCRYPT() および DES_DECRYPT() 関数によって使用されます。

- --enable-named-pipe

コマンド行形式	--enable-named-pipe
プラットフォーム固有	Windows

  名前付きパイプのサポートを有効にします。このオプションは Windows にのみ適用されます。

- --engine-condition-pushdown={ON|OFF}

コマンド行形式	--engine-condition-pushdown
非推奨	はい (removed in 5.6.1); use optimizer_switch instead

システム変数	engine_condition_pushdown
スコープ	グローバル、セッション
動的	はい
型	ブール
デフォルト	ON

以前は、このオプションは engine_condition_pushdown システム変数を設定しており、このオプションおよび変数は MySQL 5.6.1 で両方とも削除されました。詳細については、セクション8.2.1.5「エンジンコンディションプッシュダウンの最適化」を参照してください。

- --event-scheduler[=value]

コマンド行形式	--event-scheduler[=value]
システム変数	event_scheduler
スコープ	グローバル
動的	はい
型	列挙
デフォルト	OFF
有効な値	ON OFF DISABLED

イベントスケジューラを有効または無効にし、開始または停止します。

詳細情報については、--event-scheduler オプションを参照してください。

- --exit-info[=flags], -T [flags]

コマンド行形式	--exit-info[=flags]
型	数値

mysqld サーバーのデバッグに使用できる、複数の異なるフラグのビットマスクです。このオプションを使用するには、完全にこのオプションを理解していることが必要です。

- --external-locking

コマンド行形式	--external-locking
型	ブール
デフォルト	FALSE

MySQL 4.0 以降ではデフォルトで無効になった、外部ロック (システムロック) を有効にします。lockd が完全には機能しないシステム (Linux など) でこのオプションを使用すると、mysqld がデッドロックしやすくなることに注意してください。

外部ロックを明示的に無効にするには、--skip-external-locking を使用します。

外部ロックは MyISAM テーブルアクセスにのみ影響します。使用できるまたはできない状況も含めた詳細情報については、セクション8.10.5「外部ロック」を参照してください。

- --flush

コマンド行形式	--flush
システム変数	flush
スコープ	グローバル
動的	はい
型	ブール

デフォルト	OFF

各 SQL ステートメント後にすべての変更内容をディスクにフラッシュ (同期) します。通常、MySQL では各 SQL ステートメントの終了後にのみすべての変更内容をディスクに書き込み、ディスクへの同期はオペレーティングシステムが処理します。セクションB.5.4.2「MySQL が繰り返しクラッシュする場合の対処方法」を参照してください。

- --gdb

コマンド行形式	--gdb
型	ブール
デフォルト	FALSE

SIGINT 用の割り込みハンドラ (ブレークポイントを設定するための ^C を使用して mysqld を停止するために必要) をインストールし、スタックトレースおよびコアファイルの処理を無効にします。セクション24.4「MySQL のデバッグおよび移植」 を参照してください。

- --general-log[={0|1}]

コマンド行形式	--general-log
システム変数	general_log
スコープ	グローバル
動的	はい
型	ブール
デフォルト	OFF

一般的なクエリーログの初期状態を指定します。引数がないか引数が 1 の場合、--general-log オプションはログを使用可能にします。省略されるか引数が 0 の場合、オプションはログを使用不可にします。

- --ignore-db-dir=dir_name

コマンド行形式	--ignore-db-dir
導入	5.6.3
型	ディレクトリ名

このオプションは、SHOW DATABASES ステートメントまたは INFORMATION_SCHEMA テーブルの特定のディレクトリ名を無視するようサーバーに指示します。たとえば、MySQL 構成によって、UNIX のファイルシステムのルートにデータディレクトリが指定された場合、システムはその場所に、サーバーが無視する lost+found ディレクトリを作成することがあります。--ignore-db-dir=lost+found を指定してサーバーを起動すると、その名前はデータベースとしてリストされません。

複数の名前を指定するには、このオプションをそれぞれの名前に 1 回ずつ複数回使用します。空の値を使用してオプションを指定すると (つまり、--ignore-db-dir=) ディレクトリリストが空のリストにリセットされます。

サーバー起動時に指定されるこのオプションのインスタンスは、ignore_db_dirs システム変数を設定するために使用されます。

このオプションは MySQL 5.6.3 で追加されました。

- --init-file=file_name

コマンド行形式	--init-file=file_name
システム変数	init_file
スコープ	グローバル
動的	いいえ
型	ファイル名

起動時にこのファイルから SQL ステートメントを読み取ります。各ステートメントは単一の行にして、コメントを含めなでください。

- --innodb-xxx

  InnoDB ストレージエンジンのオプションを設定します。InnoDB オプションは、セクション14.12「InnoDB の起動オプションおよびシステム変数」にリストされています。

- --install [service_name]

コマンド行形式	--install [service_name]
プラットフォーム固有	Windows

(Windows のみ) Windows の起動時に自動的に開始する Windows サービスとしてサーバーをインストールします。service_name の値が指定されない場合、デフォルトサービス名は MySQL です。詳細については、セクション2.3.5.7「Windows のサービスとして MySQL を起動する」を参照してください。

> 注記
>
> --defaults-file オプションおよび --install オプションを使用してサーバーを起動する場合、--install を先にする必要があります。

- --install-manual [service_name]

コマンド行形式	--install-manual [service_name]
プラットフォーム固有	Windows

(Windows のみ) 手動で開始する必要がある Windows サービスとしてサーバーをインストールします。Windows の起動中に自動的に開始されません。service_name の値が指定されない場合、デフォルトサービス名は MySQL です。詳細については、セクション2.3.5.7「Windows のサービスとして MySQL を起動する」を参照してください。

> 注記
>
> --defaults-file オプションおよび --install-manual オプションを使用してサーバーを起動する場合、--install-manual を先にする必要があります。

- --language=lang_name, -L lang_name

コマンド行形式	--language=name
非推奨	5.6.1; use lc-messages-dir instead
システム変数	language
スコープ	グローバル
動的	いいえ
型	ディレクトリ名
デフォルト	/usr/local/mysql/share/mysql/english/

エラーメッセージに使用する言語。lang_name は、言語名として指定するか、言語ファイルがインストールされているディレクトリへのフルパス名として指定できます。セクション10.2「エラーメッセージ言語の設定」を参照してください。

MySQL 5.6 では、--lc-messages-dir および --lc-messages を使用し、--language は使用しないでください。こちらは MySQL 5.6.1 で非推奨となり、--lc-messages-dir のエイリアスとして処理されます。--language オプションは今後の MySQL リリースで削除されます。

- --large-pages

コマンド行形式	--large-pages
システム変数	large_pages
スコープ	グローバル
動的	いいえ
プラットフォーム固有	Linux
型	ブール

デフォルト	FALSE

ハードウェアまたはオペレーティングシステムのアーキテクチャーによっては、デフォルト (通常は 4K バイト) よりも大きいメモリーページをサポートしています。このサポートの実際の実装は、ベースとなるハードウェアやオペレーティングシステムに依存します。大量のメモリーアクセスがあるアプリケーションの場合、大きいページを使用して、トランスレーションルックアサイドバッファー (TLB; Translation Lookaside Buffer) のミスが減ることによってパフォーマンスが改善される可能性があります。

MySQL 5.6 は、大きいページのサポートの Linux 実装 (Linux では HugeTLB と呼ばれる) をサポートします。セクション8.11.4.2「ラージページのサポートの有効化」を参照してください。大きいページの Solaris サポートについては、--super-large-pages オプションの説明を参照してください。

--large-pages はデフォルトで無効になっています。

- --lc-messages=locale_name

コマンド行形式	--lc-messages=name
システム変数	lc_messages
スコープ	グローバル、セッション
動的	はい
型	文字列
デフォルト	en_US

エラーメッセージに使用するロケール。デフォルトは en_US です。サーバーは引数を言語名に変換し、これを --lc-messages-dir の値と組み合わせてエラーメッセージファイルの場所を生成します。セクション10.2「エラーメッセージ言語の設定」を参照してください。

- --lc-messages-dir=path

コマンド行形式	--lc-messages-dir=dir_name
システム変数	lc_messages_dir
スコープ	グローバル
動的	いいえ
型	ディレクトリ名

エラーメッセージが配置されているディレクトリ。サーバーはこの値を --lc-messages の値と一緒に使用して、エラーメッセージファイルの場所を生成します。セクション10.2「エラーメッセージ言語の設定」を参照してください。

- --local-service

コマンド行形式	--local-service

(Windows のみ) サービス名のあとに --local-service オプションが指定されると、システム権限が制限された LocalService の Windows アカウントを使用してサーバーが実行されます。このアカウントは Windows XP またはそれ以降でのみ利用できます。--defaults-file および --local-service の両方がサービス名のあとに指定される場合、どのような順序でもかまいません。セクション2.3.5.7「Windows のサービスとして MySQL を起動する」を参照してください。

- --log[=file_name], -l [file_name]

コマンド行形式	--log[=file_name]
非推奨	はい (removed in 5.6.1); use general-log instead
システム変数	log
スコープ	グローバル
動的	はい

型	ファイル名

--log オプションは MySQL 5.6.1 で (log システム変数と一緒に) 削除されました。代わりに、一般クエリーログを有効にするには --general_log オプションを使用し、一般クエリーログファイル名を設定するには --general_log_file=file_name オプションを使用してください。

- --log-error[=file_name]

コマンド行形式	--log-error[=file_name]
システム変数	log_error
スコープ	グローバル
動的	いいえ
型	ファイル名

エラーおよび起動メッセージのログをこのファイルに記録します。セクション5.2.2「エラーログ」 を参照してください。ファイル名を省略すると、MySQL は host_name.err を使用します。ファイル名に拡張子がない場合、サーバーは拡張子 .err を追加します。

- --log-isam[=file_name]

コマンド行形式	--log-isam[=file_name]
型	ファイル名

MyISAM のすべての変更内容をこのファイルに記録します (MyISAM をデバッグするときだけ使用します)。

- --log-output=value,...

コマンド行形式	--log-output=name
システム変数	log_output
スコープ	グローバル
動的	はい
型	セット
デフォルト	FILE
有効な値	TABLE FILE NONE

このオプションは、一般クエリーログおよびスロークエリーログの出力先を決定します。オプション値は、TABLE、FILE、NONE の 1 つ以上の文字で指定できます。TABLE は、宛先として mysql データベース内の general_log および slow_log テーブルへのロギングを選択します。FILE は、ログファイルを宛先としたロギングを選択します。NONE はロギングを無効にします。NONE がオプション値にある場合は、存在するすべての文字よりも優先されます。TABLE および FILE を両方指定して、両方のログ出力の宛先を選択できます。

このオプションはログ出力の宛先を選択しますが、ログ出力を有効化しません。これを行うには、--general_log および --slow_query_log オプションを使用します。FILE でのロギングの場合、--general_log_file オプションおよび -slow_query_log_file オプションによってログファイルの場所が決定されます。詳細については、セクション5.2.1「一般クエリーログおよびスロークエリーログの出力先の選択」を参照してください。

- --log-queries-not-using-indexes

コマンド行形式	--log-queries-not-using-indexes
システム変数	log_queries_not_using_indexes
スコープ	グローバル
動的	はい
型	ブール

デフォルト	OFF

スロークエリーログを有効にしてこのオプションを使用した場合、すべての行を取得することが予想されるクエリーがログに記録されます。セクション5.2.5「スロークエリーログ」を参照してください。このオプションは、インデックスが使用されないことを必ずしも意味するわけではありません。たとえば、フルインデックススキャンを使用するクエリーはインデックスを使用しますが、インデックスは行数を制限しないため、クエリーはログに記録されます。

- --log-raw

コマンド行形式	--log-raw[=value]
導入	5.6.3
型	ブール
デフォルト	OFF

MySQL 5.6.3 以降では、一般クエリーログ、スロークエリーログ、およびバイナリログに書き込まれる特定のステートメントのパスワードは、文字どおり平文で出現しないようにサーバーによって書き直されます。一般クエリーログについてのパスワードの書き換えは、--log-raw オプションでサーバーを起動することによって抑制できます。このオプションは、サーバーによって受け取られるステートメントの正確なテキストを表示する際の診断目的で役立つ場合がありますが、セキュリティー上の理由で本番用途では推奨されません。

MySQL 5.6.3 より前では、ステートメント内のパスワードは書き換えられないため、一般クエリーログを保護するようにしてください。セクション6.1.2.3「パスワードおよびロギング」を参照してください。

このオプションは MySQL 5.6.3 で追加されました。

- --log-short-format

コマンド行形式	--log-short-format
型	ブール
デフォルト	FALSE

バイナリログおよびスロークエリーログがアクティブ化されている場合、これらのログに記録する情報を少なくします。

- --log-slow-admin-statements

コマンド行形式	--log-slow-admin-statements (≤ 5.6.10)
削除	5.6.11
型	ブール
デフォルト	OFF

スロークエリーログに書き込まれるステートメントにスロー管理ステートメントを含めます。管理ステートメントには、ALTER TABLE、ANALYZE TABLE、CHECK TABLE、CREATE INDEX、DROP INDEX、OPTIMIZE TABLE、および REPAIR TABLE が含まれます。

このコマンド行オプションは MySQL 5.6.11 で削除され、log_slow_admin_statements システム変数によって置き換えられました。システム変数はオプションと同じ方法でコマンド行またはオプションファイルに設定できるため、サーバー起動時に何らかの変更を行う必要はありませんが、システム変数は実行時に値を検査または設定することも可能です。

- --log-slow-queries[=file_name]

コマンド行形式	--log-slow-queries[=name]
非推奨	はい (removed in 5.6.1); use slow-query-log instead
システム変数	log_slow_queries
スコープ	グローバル
動的	はい

型	ブール

--log-slow-queries オプションは MySQL 5.6.1 で (log_slow_queries システム変数と一緒に) 削除されました。代わりに、スロークエリーログを有効にするには --slow_query_log オプションを使用し、スロークエリーログファイル名を設定するには --slow_query_log_file=file_name オプションを使用してください。

- --log-tc=file_name

コマンド行形式	--log-tc=file_name
型	ファイル名
デフォルト	tc.log

メモリーマップ済みのトランザクションコーディネータログファイルの名前 (バイナリログが無効のときに複数のストレージエンジンに影響する XA トランザクション用)。デフォルト名は tc.log です。フルパス名が指定されない場合、ファイルはデータディレクトリの下に作成されます。現在このオプションは使用されていません。

- --log-tc-size=size

コマンド行形式	--log-tc-size=#
型	数値
デフォルト	24576
最大値 (64 ビットプラットフォーム)	18446744073709551615
最大値 (32 ビットプラットフォーム)	4294967295

メモリーマップ済みのトランザクションコーディネータログのバイト単位のサイズ。デフォルトサイズは 24K バイトです。

- --log-warnings[=level], -W [level]

コマンド行形式	--log-warnings[=#]
システム変数	log_warnings
スコープ (≥ 5.6.4)	グローバル
スコープ (≤ 5.6.3)	グローバル、セッション
動的	はい
型	数値
デフォルト	1
最小値	0
最大値 (64 ビットプラットフォーム)	18446744073709551615
最大値 (32 ビットプラットフォーム)	4294967295

「Aborted connection...」などの警告をエラーログに出力します。このオプションは、デフォルトで有効 (1) になっています。これを無効にするには、--log-warnings=0 を使用します。level 値を付けずにオプションを指定すると、現在の値が 1 増加します。レプリケーションを使用するなどの場合は、このオプションを 0 より大きく設定して有効にすることを推奨します (ネットワーク障害や再接続についてのメッセージなど、発生中の詳細情報を受け取ります)。値が 1 より大きい場合、中止された接続がエラーログに書き込まれ、新しい接続の試行についてのアクセス拒否エラーが書き込まれます。セクションB.5.2.11「通信エラーおよび中止された接続」を参照してください。

スレーブサーバーが --log-warnings を有効にして起動された場合、スレーブは、スレーブのステータスに関する情報を提供するためのメッセージをエラーログに出力し、この情報には、スレーブがジョブを開始したときのバイナリログおよびリレーログの座標、別のリレーログに切り替える時期、切断後に再接続する時期などがあります。--log-warnings が 0 より大きい場合、サーバーはステートメントベースのロギングについて、安全ではないステートメントに関するメッセージをログに記録します。

- --low-priority-updates

コマンド行形式	--low-priority-updates
システム変数	low_priority_updates
スコープ	グローバル、セッション
動的	はい
型	ブール
デフォルト	FALSE

テーブル変更操作 (INSERT、REPLACE、DELETE、UPDATE) に、選択よりも低い優先順位を与えます。これは {INSERT | REPLACE | DELETE | UPDATE} LOW_PRIORITY ... を使用して 1 つのクエリーのみ優先順位を下げたり、SET LOW_PRIORITY_UPDATES=1 によって 1 つのスレッドの優先順位を変更したりして実行することもできます。これは、テーブルレベルロックのみを使用するストレージエンジン (MyISAM、MEMORY、MERGE) にのみ影響を与えます。セクション8.10.2「テーブルロックの問題」を参照してください。

- --min-examined-row-limit=number

コマンド行形式	--min-examined-row-limit=#
システム変数	min_examined_row_limit
スコープ	グローバル、セッション
動的	はい
型	数値
デフォルト	0
最小値	0
最大値 (64 ビットプラットフォーム)	18446744073709551615
最大値 (32 ビットプラットフォーム)	4294967295

このオプションを設定すると、number 行より少ない行を検査するクエリーがスロークエリーログに書き込まれません。デフォルトは 0 です。

- --memlock

コマンド行形式	--memlock
型	ブール
デフォルト	FALSE

メモリー内の mysqld プロセスをロックします。このオプションは、オペレーティングシステムによって mysqld がディスクへのスワップを実行するという問題がある場合に役立つことがあります。

--memlock は、mlockall() システムコールをサポートするシステム上で機能し、システムには、Solaris、2.4 またはそれより新しいカーネルを使用するほとんどの Linux 配布、およびその他の UNIX システムが含まれま

す。Linux システムの場合、mlockall() (およびこのオプション) が、システムの mman.h ファイルで定義されているかどうかを次のようにして確認することによって、これがサポートされているかどうかを識別できます。

```
shell> grep mlockall /usr/include/sys/mman.h
```

mlockall() がサポートされている場合、前のコマンドの出力に、次のように表示されます。

```
extern int mlockall (int __flags) __THROW;
```

> **重要**
>
> このオプションを使用する場合、サーバーを root として実行することが必要な場合もあり、これはセキュリティー上の理由から通常はよい考えではありません。セクション6.1.5「MySQL を通常ユーザーとして実行する方法」を参照してください。
>
> Linux およびおそらくその他のシステムでは、limits.conf ファイルを変更することによって、サーバーを root として実行しないで済みます。セクション8.11.4.2「ラージページのサポートの有効化」の memlock 制限に関するメモを参照してください。
>
> mlockall() システムコールをサポートしないシステムでは、このオプションを使用しないでください。これを行うと、mysqld は開始してすぐに高い確率でクラッシュします。

- --myisam-block-size=N

コマンド行形式	--myisam-block-size=#
型	数値
デフォルト	1024
最小値	1024
最大値	16384

MyISAM インデックスページに使用するブロックサイズ。

- --myisam-recover-options[=option[,option]...]]

コマンド行形式	--myisam-recover-options[=name]
型	列挙
デフォルト	OFF
有効な値	OFF DEFAULT BACKUP FORCE QUICK

MyISAM のストレージエンジンのリカバリモードを設定します。オプション値は、OFF、DEFAULT、BACKUP、FORCE、または QUICK の任意の組み合わせです。複数の値を指定する場合、値をカンマで区切ります。オプションに引数を指定しないことは DEFAULT を指定するのと同じで、明示的な値 "" を指定するとリカバリが無効になります (値 OFF と同じ)。リカバリが有効な場合、mysqld は MyISAM テーブルをオープンするたび、テーブルがクラッシュしたというマークが付いているか、テーブルが正しくクローズしなかったかどうかをチェックします。(最後のオプションは外部ロックを無効にして実行している場合のみ機能します。)このような場合、mysqld はテーブル上でチェックを実行します。テーブルが破損していた場合、mysqld は修復を試みます。

次のオプションは修復の動作方法に影響します。

オプション	説明
OFF	リカバリなし。
DEFAULT	バックアップ、強制、クイックチェックを行わないリカバリ。

オプション	説明
BACKUP	データファイルがリカバリ中に変更された場合、tbl_name.MYD ファイルのバックアップを tbl_name-datetime.BAK として保存します。
FORCE	.MYD ファイルから複数のレコードがなくなる場合でもリカバリを実行します。
QUICK	削除ブロックがない場合、テーブルの行をチェックしません。

サーバーがテーブルを自動的に修復する前に、サーバーは修復に関するメモをエラーログに書き込みます。ユーザーが介入せずにほとんどの問題をリカバリできるようにするには、BACKUP,FORCE オプションを使用します。これにより、一部の行が削除される場合でもテーブルの修復を強制しますが、古いデータファイルをバックアップとして保持しているため、何が発生したかをあとで検査できます。

セクション15.2.1「MyISAM 起動オプション」を参照してください。

- --no-defaults

  オプションファイルを読み取りません。オプションファイルから不明のオプションを読み取ることが原因でプログラムの起動に失敗する場合、--no-defaults を使用して、オプションを読み取らないようにできます。

  例外として、.mylogin.cnf ファイルは、存在する場合はすべての場合に読み取られます。これにより、--no-defaults が使用されたとしても、コマンド行よりも安全な方法でパスワードを指定できます (.mylogin.cnf は mysql_config_editor ユーティリティーによって作成されます。セクション4.6.6「mysql_config_editor — MySQL 構成ユーティリティー」を参照してください)。

- --old-alter-table

コマンド行形式	--old-alter-table
システム変数	old_alter_table
スコープ	グローバル、セッション
動的	はい
型	ブール
デフォルト	OFF

  このオプションが指定された場合、サーバーは ALTER TABLE 操作の処理の最適化された方式を使用しません。一時テーブルの使用に戻り、データのコピー後、MySQL 5.0 以前で使用されていたように、一時テーブルを元のテーブルの名前に変更します。ALTER TABLE の操作について詳しくは、セクション13.1.7「ALTER TABLE 構文」を参照してください。

- --old-style-user-limits

コマンド行形式	--old-style-user-limits
型	ブール
デフォルト	FALSE

  古いスタイルのユーザー制限を有効にします。(MySQL 5.0.3 以前では、アカウントリソースは、user テーブルのアカウント行単位ではなく、ユーザーが接続したホストごとに別々にカウントされていました。)セクション6.3.4「アカウントリソース制限の設定」を参照してください。

- --one-thread

コマンド行形式	--one-thread
削除	5.6.1

  このオプションは MySQL 5.6.1 で削除されました。代わりに --thread_handling=no-threads を使用してください。

- --open-files-limit=count

コマンド行形式	--open-files-limit=#
システム変数	open_files_limit
スコープ	グローバル

動的	いいえ
型	数値
デフォルト (≥ 5.6.8)	5000, with possible adjustment
デフォルト (≤ 5.6.7)	0
最小値	0
最大値	platform dependent

mysqld で使用可能なファイルディスクリプタの数を変更します。mysqld が「Too many open files」のエラーを出す場合、このオプションの値を増やしてみてください。mysqld はこのオプション値を使用して、setrlimit() でディスクリプタを予約します。内部的には、このオプションの最大値は符号なし整数値の最大値ですが、実際の最大値はプラットフォームに依存します。リクエストされた数のファイルディスクリプタを割り当てできない場合、mysqld はエラーログに警告を書き込みます。

mysqld は、max_connections および table_open_cache の値を使用して、さらに多くのディスクリプタが必要かどうかを推定し、(利用可能な場合は) リクエストされた数のディスクリプタよりも多くを割り当てようとする場合もあります。

UNIX の場合、値を ulimit -n より少なく設定できません。

- --partition[=value]

コマンド行形式	--partition
無効化	skip-partition
型	ブール
デフォルト	ON

MySQL Server サーバーのユーザー定義のパーティショニングサポートを有効または無効にします。

- --performance-schema-xxx

パフォーマンススキーマオプションを構成します。詳細については、セクション22.11「パフォーマンススキーマコマンドオプション」を参照してください。

- --pid-file=path

コマンド行形式	--pid-file=file_name
システム変数	pid_file
スコープ	グローバル
動的	いいえ
型	ファイル名

プロセス ID ファイルのパス名。サーバーは、別のディレクトリを指定する絶対パス名が指定されないかぎり、データディレクトリ内にファイルを作成します。このファイルは、mysqld_safe などの別のプログラムによってサーバーのプロセス ID を判別するときに使用されます。

- --plugin-xxx

サーバープラグインに関するオプションを指定します。たとえば、多くのストレージエンジンはプラグインとして構築でき、そのようなエンジンに対してそれらのオプションを --plugin プリフィクスで指定できます。したがって、InnoDB の --innodb_file_per_table オプションを --plugin-innodb_file_per_table のように指定できます。

有効または無効にできるブールオプションの場合、--skip プリフィクスおよびその他の代替形式もサポートされます (セクション4.2.5「プログラムオプション修飾子」を参照してください)。たとえば、--skip-plugin-innodb_file_per_table は innodb_file_per_table を無効にします。

--plugin プリフィクスの理由として、組み込みサーバーオプションとの名前の競合がある場合に、あいまいさを排除してプラグインオプションを指定できるということがあります。たとえば、プラグイン 「sql」 に名前を指定し 「mode」 オプションを実装するプラグインライターは、オプション名が --sql-mode となることがあり、同じ名前の組み込みオプションと競合します。そのような場合、競合する名前への参照は、組み込みオプション側として解決されます。あいまいさを避けるために、ユーザーはプラグインオプションを --plugin-sql-

mode として指定できます。あいまいさの問題を避けるために、プラグインオプションに --plugin プリフィクスを使用することを推奨します。

- --plugin-load=plugin_list

コマンド行形式	--plugin-load=plugin_list
型	文字列

このオプションは、指定されたプラグインを起動時にロードするようサーバーに指示します。オプション値は、セミコロンで区切られた name=plugin_library のペアのリストです。それぞれの name はプラグインの名前で、plugin_library はプラグインコードを含む共有ライブラリの名前です。各ライブラリファイルは、plugin_dir システム変数によって指定されるディレクトリに配置されている必要があります。たとえば、myplug1 および myplug2 という名前のプラグインが、ライブラリファイル myplug1.so および myplug2.so を持つ場合、次のオプションを使用して、これらを起動時にロードします。

```
shell> mysqld --plugin-load="myplug1=myplug1.so;myplug2=myplug2.so"
```

ここでは引数の前後に引用符が使用されますが、これはセミコロン (;) が一部のコマンドインタプリタによって特殊文字として解釈されるためです。(たとえば UNIX シェルでは、これはコマンド終端記号として扱われます。)

複数の --plugin-load オプションが指定された場合、最後のオプションのみが使用されます。ロードする追加のプラグインは、--plugin-load-add オプションを使用して指定できます。

プラグイン名を前に付けずにプラグインライブラリを指定した場合、サーバーはライブラリ内のすべてのプラグインをロードします。

各プラグインは、mysqld の単一の呼び出しについてのみロードされます。再起動後、--plugin-load をふたたび使用しないかぎり、プラグインはロードされません。これは INSTALL PLUGIN とは対照的で、こちらは mysql.plugins テーブルに項目を追加することで、サーバーが通常起動するたびにプラグインがロードされます。

通常の起動では、サーバーは mysql.plugins システム変数を読み取ることによって、ロードするプラグインを判別します。サーバーが --skip-grant-tables オプションで起動された場合、サーバーは mysql.plugins テーブルを参照せず、そこにリストされているプラグインをロードしません。--plugin-load により、--skip-grant-tables が指定されている場合でもプラグインのロードを可能にします。--plugin-load は、プラグインを実行時にロードできない構成で、プラグインを起動時にロードできるようにします。

プラグインのロードについての追加情報は、セクション5.1.8.1「プラグインのインストールおよびアンインストール」を参照してください。

- --plugin-load-add=plugin_list

コマンド行形式	--plugin-load-add=plugin_list
導入	5.6.3
型	文字列

このオプションは --plugin-load オプションを補完します。--plugin-load-add は、起動時にロードされるプラグインのセットに 1 つまたは複数のプラグインを追加します。引数の形式は --plugin-load と同じです。--plugin-

load-add は、大量のプラグインのセットを、長くて扱いにくい単一の --plugin-load 引数として指定しないようにできます。このオプションは MySQL 5.6.3 で追加されました。

--plugin-load-add は --plugin-load がなくても指定できますが、--plugin-load はロードするプラグインのセットをリセットするため、--plugin-load の前に出現するすべての --plugin-load-add は効果がありません。つまり、次のオプションの場合、

```
--plugin-load=x --plugin-load-add=y
```

上記は次のオプションと同等です。

```
--plugin-load="x;y"
```

ただし、次のオプションの場合、

```
--plugin-load-add=y --plugin-load=x
```

上記は次のオプションと同等です。

```
--plugin-load=x
```

プラグインのロードについての追加情報は、セクション5.1.8.1「プラグインのインストールおよびアンインストール」を参照してください。

- --port=port_num, -P port_num

コマンド行形式	--port=#
システム変数	port
スコープ	グローバル
動的	いいえ
型	数値
デフォルト	3306
最小値	0
最大値	65535

TCP/IP 接続を listen するときに使用するポート番号。サーバーが root システムユーザーで開始されている場合を除き、ポート番号は 1024 以上にする必要があります。

- --port-open-timeout=num

コマンド行形式	--port-open-timeout=#
型	数値
デフォルト	0

一部のシステムでは、サーバーが停止すると、TCP/IP ポートがただちに利用できなくなることがあります。その後すぐにサーバーを再起動した場合、サーバーがポートをふたたびオープンしようとして失敗することがあります。このオプションは、TCP/IP ポートを開くことができない場合、TCP/IP ポートが開放されるまでサーバーが待機する秒数を指示します。デフォルトでは待機しません。

- --print-defaults

プログラム名と、オプションファイルから受け取るすべてのオプションを出力します。

- --remove [service_name]

コマンド行形式	--remove [service_name]
プラットフォーム固有	Windows

(Windows のみ) MySQL Windows サービスを削除します。service_name の値が指定されない場合、デフォルトサービス名は MySQL です。詳細については、セクション2.3.5.7「Windows のサービスとして MySQL を起動する」を参照してください。

- --safe-mode

コマンド行形式	--safe-mode
非推奨	はい (removed in 5.6.6)

一部の最適化ステージをスキップします。このオプションは MySQL 5.6.6 で削除されました。

- --safe-user-create

コマンド行形式	--safe-user-create
型	ブール
デフォルト	FALSE

このオプションが有効な場合、ユーザーが mysql.user テーブルまたはテーブル内の任意のカラムに対して INSERT 権限を持つ場合を除き、ユーザーは GRANT ステートメントを使用して新しい MySQL ユーザーを作成できません。あるユーザーが新規ユーザーを作成する能力を持ち、そのユーザーが付与する権利を持つ権限を、新規ユーザーが持つようにするには、そのユーザーに次の権限を付与します。

```
GRANT INSERT(user) ON mysql.user TO 'user_name'@'host_name';
```

これで、ユーザーは権限カラムを直接変更できませんが、GRANT ステートメントを使用してほかのユーザーに権限を与えることができるようになります。

- --secure-auth

コマンド行形式	--secure-auth
システム変数	secure_auth
スコープ	グローバル
動的	はい
型	ブール
デフォルト (≥ 5.6.5)	ON
デフォルト (≤ 5.6.4)	OFF

このオプションにより、サーバーは、古い (4.1 以前の) 形式で格納されているパスワードを持つアカウントを使用しようとする、クライアントによる接続をブロックします。古い形式を用いたすべてのパスワードの使用 (ネットワーク上でのセキュアでない通信) を防ぐために、これを使用します。MySQL 5.6.5 以前では、このオプションはデフォルトで無効になっています。MySQL 5.6.5 以降では、これはデフォルトで有効になっています。これを無効にするには、--skip-secure-auth を使用します。

このオプションが有効になっていて、権限テーブルが 4.1 以前の形式の場合、サーバーの起動が失敗して エラーが出ます。セクションB.5.2.4「クライアントは認証プロトコルに対応できません」を参照してください。

mysql クライアントにも --secure-auth オプションがあり、これはサーバーがそのクライアントアカウントに対して古い形式のパスワードを要求する場合にサーバーに接続しないようにします。

> **注記**
>
> 4.1 より前のハッシュ方式を使用するパスワードはネイティブのパスワードハッシュ方式を使用するパスワードよりもセキュアでないため、使用しないようにしてください。4.1 よりも前のパスワードは非推奨であり、これらのサポートは今後の MySQL リリースで削除される予定です。これにより、--skip-secure-auth を使用したセキュア認証の無効化も非推奨になります。

- --secure-file-priv=path

コマンド行形式	--secure-file-priv=path
システム変数	secure_file_priv
スコープ	グローバル
動的	いいえ

型	文字列

このオプションは、LOAD_FILE() 関数と LOAD DATA および SELECT ... INTO OUTFILE ステートメントの効果を制限し、指定されたディレクトリ内のファイルにのみ作用します。

- --shared-memory

コマンド行形式	--shared-memory-base-name=name
システム変数	shared_memory
スコープ	グローバル
動的	いいえ
プラットフォーム固有	Windows

ローカルクライアントによる共有メモリー接続を有効にします。このオプションは Windows でのみ使用できます。

- --shared-memory-base-name=name

システム変数	shared_memory_base_name
スコープ	グローバル
動的	いいえ
プラットフォーム固有	Windows

共有メモリー接続に使用する共有メモリーの名前。このオプションは Windows でのみ使用できます。デフォルト名は MYSQL です。名前は大文字と小文字を区別します。

- --skip-concurrent-insert

MyISAM テーブルに対する SELECT 文と INSERT 文の同時実行を無効化します。(これは、この機能にバグが見つかったと思われる場合のみ使用します。)セクション8.10.3「同時挿入」を参照してください。

- --skip-event-scheduler

コマンド行形式	--skip-event-scheduler --disable-event-scheduler

イベントスケジューラを OFF にします。これは、--event-scheduler=DISABLED の設定が必要なイベントスケジューラの無効化と同じではありません。詳細については、「--event-scheduler オプション」を参照してください。

- --skip-grant-tables

このオプションにより、サーバーは権限システムをまったく使用せずに開始され、サーバーへのアクセス権を持つすべてのユーザーにすべてのデータベースに対する無制限アクセス権が与えられます。サーバー実行中に、付与テーブルの行使を再開するには、mysqladmin flush-privileges または mysqladmin reload コマンドをシステムシェルから実行するか、サーバーへの接続後に MySQL FLUSH PRIVILEGES ステートメントを発行します。このオプションは、INSTALL PLUGIN ステートメント、ユーザー定義関数 (UDF)、およびスケジュールされたイベントでインストールされたプラグインのロードも抑制します。プラグインをロードするには、--plugin-load オプションを使用します。

FLUSH PRIVILEGES は起動後に実行されるほかのアクションによって暗黙的に実行される場合があります。たとえば、mysql_upgrade はアップグレード手続き中に権限をフラッシュします。

- --skip-host-cache

  名前と IP の解決を高速化するために内部ホストキャッシュの使用を無効にします。この場合、サーバーはクライアントが接続するたびに DNS ルックアップを実行します。セクション8.11.5.2「DNS ルックアップの最適化とホストキャッシュ」を参照してください。

  --skip-host-cache の使用は host_cache_size システム変数を 0 に設定することに似ていますが、host_cache_size の方が柔軟性が高く、これはサーバー起動時だけでなく実行時にもホストキャッシュのサイズを変更したり有効化または無効化したりするために使用できるためです。

  --skip-host-cache を指定してサーバーを開始しても、host_cache_size の値の変更を妨げるわけではありませんが、この変更は効果がなく、host_cache_size を 0 より大きく設定してもキャッシュはふたたび有効化されません。

- --skip-innodb

  InnoDB ストレージエンジンを無効にします。この場合、デフォルトのストレージエンジンは InnoDB であるため、--default-storage-engine および --default-tmp-storage-engine を使用して、永続テーブルと TEMPORARY テーブルの両方についてデフォルトを別のエンジンに設定しないかぎりサーバーは開始しません。

  MySQL 5.6.21 以降では、--skip-innodb オプションは非推奨です。これを使用すると警告が出ます。このオプションは今後の MySQL リリースで削除されます。

- --skip-name-resolve

  クライアント接続を検査するときにホスト名を解決しません。IP アドレスのみを使用します。このオプションを使用する場合、付与テーブルのすべての Host カラムの値は IP アドレスまたは localhost である必要があります。セクション8.11.5.2「DNS ルックアップの最適化とホストキャッシュ」を参照してください。

  システムのネットワーク構成およびアカウントの Host 値によっては、クライアントは --host=localhost、--host=127.0.0.1、--host=::1 などの明示的な --host オプションを使用して接続する必要がある場合もあります。

- --skip-networking

  TCP/IP 接続を listen しません。mysqld とのすべての対話は、名前付きパイプまたは共有メモリー (Windows の場合) または UNIX ソケットファイル (UNIX の場合) を使用して行う必要があります。このオプションは、ローカルクライアントのみが許可されているシステムで強く推奨します。セクション8.11.5.2「DNS ルックアップの最適化とホストキャッシュ」を参照してください。

- --skip-partition

コマンド行形式	--skip-partition --disable-partition

  ユーザー定義のパーティショニングを無効にします。パーティション化されたテーブルは、SHOW TABLES を使用するか、INFORMATION_SCHEMA.TABLES テーブルを照会することで表示できますが、作成または変更することはできず、そのようなテーブルのデータにアクセスすることもできません。INFORMATION_SCHEMA.PARTITIONS テーブル内のパーティション固有のすべてのカラムには NULL が表示されます。

  DROP TABLE によってテーブル定義 (.frm) ファイルが削除されるため、このオプションを使用してパーティショニングが無効化されていても、このステートメントはパーティション化されたテーブルで機能します。ただしこのような場合、このステートメントはパーティション化されたテーブルに関連付けられた .par ファイルを削除しません。このため、パーティショニングを無効化したパーティション化されたテーブルをドロップしないようにするか、孤立した .par ファイルを手動で削除してください。

- --ssl*

  --ssl で始まるオプションは、クライアントが SSL を使用して接続することを許可するかどうかを指定し、SSL 鍵および証明書を見つける場所を指定します。セクション6.3.10.4「SSL コマンドのオプション」を参照してください。

- --standalone

コマンド行形式	--standalone
プラットフォーム固有	Windows

Windows でのみ使用可能で、MySQL Server にサービスとして実行しないよう指示します。

- --super-large-pages

コマンド行形式	--super-large-pages
プラットフォーム固有	Solaris
型	ブール
デフォルト	FALSE

MySQL での標準的な大規模ページの使用では、サポートされる最大サイズである 4M バイトまでの使用が試行されます。Solaris では「超大規模ページ」機能により 256M バイトまでのページの使用が可能です。この機能は最新の SPARC プラットフォームで使用できます。これは --super-large-pages または --skip-super-large-pages オプションを使用して有効または無効にできます。

- --symbolic-links、--skip-symbolic-links

コマンド行形式	--symbolic-links

シンボリックリンクサポートを有効または無効にします。このオプションは、Windows と UNIX で異なる効果になります。

- Windows では、シンボリックリンクを有効にすることで、実際のディレクトリへのパスを含む db_name.sym ファイルを作成することによって、データベースディレクトリへのシンボリックリンクを確立できます。Windows 上のデータベースへのシンボリックリンクの使用を参照してください。
- UNIX では、シンボリックリンクを有効にすることは、CREATE TABLE ステートメントの INDEX DIRECTORY または DATA DIRECTORY オプションを使用して MyISAM インデックスファイルまたはデータファイルを別のディレクトリにリンクできることを意味します。テーブルを削除したり名前変更したりすると、そのシンボリックリンクが指定するファイルも削除されたり名前が変更されたりします。Unix 上の MyISAM へのシンボリックリンクの使用を参照してください。

- --skip-show-database

コマンド行形式	--skip-show-database
システム変数	skip_show_database
スコープ	グローバル
動的	いいえ

このオプションは、SHOW DATABASES ステートメントを使用することが許可されているユーザーを制御する skip_show_database システム変数を設定します。セクション5.1.4「サーバーシステム変数」を参照してください。

- --skip-stack-trace

コマンド行形式	--skip-stack-trace

スタックトレースを書き込みません。このオプションは、デバッガで mysqld を実行するときに役立ちます。一部のシステムでは、コアファイルを取得するために、このオプションの使用が必要になることもあります。セクション24.4「MySQL のデバッグおよび移植」 を参照してください。

- --skip-thread-priority

コマンド行形式	--skip-thread-priority
非推奨	はい (removed in 5.6.1)

応答速度を高めるために、スレッド優先順位の使用を無効にします。このオプションは使用されておらず、MySQL 5.6.1 で削除されました。

- --slow-query-log[={0|1}]

コマンド行形式	--slow-query-log
システム変数	slow_query_log

スコープ	グローバル
動的	はい
型	ブール
デフォルト	OFF

スロークエリーログの初期状態を指定します。引数がないか引数が 1 の場合、--slow-query-log オプションはログを使用可能にします。省略されるか引数が 0 の場合、オプションはログを使用不可にします。

- --slow-start-timeout=timeout

コマンド行形式	--slow-start-timeout=#
導入	5.6.5
型	数値
デフォルト	15000

このオプションは、Windows サービスコントロールマネージャーのサービス開始タイムアウトを制御します。この値は、起動時に Windows サービスを強制終了する前に、サービスコントロールマネージャーが待機する最大のミリ秒数です。デフォルト値は 15000 (15 秒) です。MySQL サービスの開始に時間がかかりすぎる場合、この値を増やすことが必要なこともあります。値 0 は、タイムアウトがないことを意味します。

- --socket=path

コマンド行形式	--socket=file_name
システム変数	socket
スコープ	グローバル
動的	いいえ
型	ファイル名
デフォルト	/tmp/mysql.sock

UNIX の場合、このオプションは、ローカル接続用の listen を行うときに使用する UNIX ソケットファイルを指定します。デフォルト値は /tmp/mysql.sock です。このオプションが指定された場合、別のディレクトリを指定する絶対パス名が指定されないかぎり、サーバーはデータディレクトリにファイルを作成します。Windows の場合、このオプションは、名前付きパイプを使用する、ローカル接続用の listen を行うときに使用するパイプ名を指定します。デフォルト値は MySQL です (大文字小文字の区別はありません)。

- --sql-mode=value[,value[,value...]]

コマンド行形式	--sql-mode=name
システム変数	sql_mode
スコープ	グローバル、セッション
動的	はい
型	セット
デフォルト (≥ 5.6.6)	NO_ENGINE_SUBSTITUTION
デフォルト (≤ 5.6.5)	''
有効な値	ALLOW_INVALID_DATES ANSI_QUOTES ERROR_FOR_DIVISION_BY_ZERO HIGH_NOT_PRECEDENCE IGNORE_SPACE NO_AUTO_CREATE_USER NO_AUTO_VALUE_ON_ZERO

	NO_BACKSLASH_ESCAPES NO_DIR_IN_CREATE NO_ENGINE_SUBSTITUTION NO_FIELD_OPTIONS NO_KEY_OPTIONS NO_TABLE_OPTIONS NO_UNSIGNED_SUBTRACTION NO_ZERO_DATE NO_ZERO_IN_DATE ONLY_FULL_GROUP_BY PAD_CHAR_TO_FULL_LENGTH PIPES_AS_CONCAT REAL_AS_FLOAT STRICT_ALL_TABLES STRICT_TRANS_TABLES

SQL モードを設定します。MySQL 5.6.6 より前では、デフォルトは '' (空の文字列) でしたが、MySQL 5.6.6 以降では、デフォルトは NO_ENGINE_SUBSTITUTION です。セクション5.1.7「サーバー SQL モード」を参照してください。

> **注記**
>
> MySQL インストールプログラムはインストールプロセス中に SQL モードを構成することがあります。たとえば、mysql_install_db は、my.cnf という名前のデフォルトオプションファイルを基本インストールディレクトリに作成します。このファイルには、SQL モードを設定する行が含まれています。セクション4.4.3「mysql_install_db — MySQL データディレクトリの初期化」を参照してください。
>
> SQL モードがデフォルトまたは期待されているモードと異なる場合、サーバーが起動時に読み取るオプションファイル内の設定を確認してください。

- --sysdate-is-now

コマンド行形式	--sysdate-is-now
型	ブール
デフォルト	FALSE

デフォルトの SYSDATE() は、この関数があるステートメントの実行が開始された時間ではなく、この関数が実行された時間を返します。これは NOW() の動作と異なります。このオプションによって、SYSDATE() を NOW() のエイリアスにできます。バイナリロギングおよびレプリケーションに対する意味については、セクション12.7「日付および時間関数」の SYSDATE() および セクション5.1.4「サーバーシステム変数」の SET TIMESTAMP についての説明を参照してください。

- --tc-heuristic-recover={COMMIT|ROLLBACK}

コマンド行形式	--tc-heuristic-recover=name
型	列挙
デフォルト	COMMIT
有効な値	COMMIT

	ROLLBACK

経験則によるリカバリプロセスで使用する決定のタイプ。現在このオプションは使用されていません。

- --temp-pool

コマンド行形式	--temp-pool
型	ブール
デフォルト	TRUE

このオプションにより、サーバーによって作成されるほとんどの一時ファイルが、新規ファイルごとに一意の名前を使用するのではなく、少数の名前のセットを使用するようになります。これは、異なる名前で多くの新規ファイルの作成を処理する Linux カーネルでの問題を回避します。Linux では、メモリーがディスクキャッシュではなくディレクトリエントリキャッシュへの割り当てになるため、従来の動作ではメモリーの「リーク」が発生しがちです。このオプションは Linux 以外では無視されます。

- --transaction-isolation=level

コマンド行形式	--transaction-isolation=name
型	列挙
デフォルト	REPEATABLE-READ
有効な値	READ-UNCOMMITTED READ-COMMITTED REPEATABLE-READ SERIALIZABLE

デフォルトのトランザクション分離レベルを設定します。level 値は、READ-UNCOMMITTED、READ-COMMITTED、REPEATABLE-READ、または SERIALIZABLE に設定できます。セクション13.3.6「SET TRANSACTION 構文」を参照してください。

デフォルトのトランザクション分離レベルは SET TRANSACTION ステートメントを使用するか、tx_isolation システム変数を設定することによって実行時にも設定できます。

- --transaction-read-only

コマンド行形式	--transaction-read-only
導入	5.6.5
型	ブール
デフォルト	OFF

デフォルトのトランザクションアクセスモードを設定します。デフォルトでは読み取り専用モードが無効化されているため、モードは読み取り/書き込みです。

デフォルトのトランザクションアクセスモードを実行時に設定するには、SET TRANSACTION ステートメントを使用するか、tx_read_only システム変数を設定します。セクション13.3.6「SET TRANSACTION 構文」を参照してください。

このオプションは MySQL 5.6.5 で追加されました。

- --tmpdir=path, -t path

コマンド行形式	--tmpdir=path
システム変数	tmpdir
スコープ	グローバル
動的	いいえ

型	ディレクトリ名

一時ファイルを作成するために使用するディレクトリのパス。これは、小さすぎて一時テーブルを保持できないパーティション上にデフォルトの /tmp ディレクトリがある場合に役立つことがあります。このオプションは、ラウンドロビン方式で使用されるいくつかのパスを受け入れます。パスは UNIX ではコロン文字 (「:」)、Windows ではセミコロン文字 (「;」) で区切るようにしてください。MySQL Server がレプリケーションスレーブとして動作する場合、--tmpdir を、メモリーベースのファイルシステム上のディレクトリや、サーバーホストが再起動したときにクリアされるディレクトリに指定するように設定しないでください。一時ファイルのストレージ位置に関しては、セクションB.5.4.4「MySQL が一時ファイルを格納する場所」を参照してください。レプリケーションスレーブは、一部の一時ファイルがマシンの再起動後も存続し、一時テーブルまたは LOAD DATA INFILE 操作を複製できるようにする必要があります。サーバーが再起動したときに一時ファイルディレクトリ内のファイルが消失した場合、レプリケーションは失敗します。

- --user={user_name|user_id}, -u {user_name|user_id}

コマンド行形式	--user=name
型	文字列

mysqld サーバーを、名前 user_name または数字ユーザー ID user_id を持つユーザーとして実行します。(このコンテキストでの「ユーザー」は、システムログインアカウントであり、付与テーブルにリストされている MySQL ユーザーではありません。)

mysqld を root として起動する場合、このオプションは必須です。サーバーは起動シーケンス中にそのユーザー ID を変更し、root ではなく特定のユーザーでこれを実行します。セクション6.1.1「セキュリティーガイドライン」を参照してください。

セキュリティーホールを回避するため、つまりユーザーが --user=root オプションを my.cnf ファイルに追加することが原因で、サーバーが root として稼働できないようにするために、mysqld で最初に指定した --user オプションだけを使用し、複数の --user オプションがあった場合に警告を生成します。/etc/my.cnf および $MYSQL_HOME/my.cnf 内のオプションは、コマンド行のオプションより先に処理することになるため、--user オプションを /etc/my.cnf に含めた上で、root 以外の値を指定することを推奨します。/etc/my.cnf 内のオプションがほかの --user オプションよりも先に検出されることになるので、サーバーは確実に root 以外のユーザーとして実行することになり、別の --user オプションが検出されると警告を出します。

- --verbose, -v

  詳細なヘルプを得るには、このオプションを --help オプションと一緒に使用します。

- --version, -V

  バージョン情報を表示して終了します。

--var_name=value の形式のオプションを使用して、サーバーシステム変数に値を割り当てることができます。たとえば、--key_buffer_size=32M は key_buffer_size 変数を 32M バイトの値に設定できます。

変数に値を割り当てた場合、MySQL は特定の範囲内にとどまるようにするために値を自動的に修正したり、特定の値のみが許可されていたりする場合は許容できるもっとも近い値に値を調整します。

実行時に SET で変数を設定可能な最大値を制限するには、--maximum-var_name=value コマンド行オプションを使用してこれを定義できます。

SET ステートメントを使用して、実行中のサーバーについてのほとんどのシステム変数の値を変更できます。セクション13.7.4「SET 構文」を参照してください。

すべての変数の詳細に加え、サーバーの起動時や実行時にこれらの設定に関する追加情報は、セクション5.1.4「サーバーシステム変数」を参照してください。システム変数の調整によるサーバーの最適化に関する情報は、セクション8.11.2「サーバーパラメータのチューニング」を参照してください。

5.1.4 サーバーシステム変数

MySQL Server には MySQL Server の構成方法を指示する多くのシステム変数が保持されています。各システム変数にはデフォルト値があります。システム変数は、コマンド行のオプションを使用するか、オプションファイルでサーバー起動時に設定できます。これらのほとんどは、SET ステートメントを使用してサーバーの実行中に動的に変更でき、これによりサーバーを停止して再起動することなくサーバーの動作を変更できます。システム変数値を式によって参照できます。

システム変数の名前と値を表示するにはいくつかの方法があります。

- サーバーのコンパイル時のデフォルトおよび読み取られるオプションファイルに基づいて、サーバーが使用する値を表示するには、次のコマンドを使用します。

```
mysqld --verbose --help
```

- すべてのオプションファイルの設定を無視し、サーバーのコンパイル時のデフォルトに基づいてサーバーが使用する値を表示するには、次のコマンドを使用します。

```
mysqld --no-defaults --verbose --help
```

- 実行中のサーバーによって使用される現在の値を表示するには、SHOW VARIABLES ステートメントを使用します。

このセクションでは各システム変数について説明します。バージョンが示されていない変数は、すべての MySQL 5.6 リリースに存在します。

次の表は使用可能なすべてのシステム変数をリストしたものです。

表 5.3 システム変数のサマリー

名前	コマンド行	オプションファイル	システム変数	変数スコープ	動的
audit_log_buffer_size	はい	はい	はい	グローバル	いいえ
audit_log_connection_policy	はい	はい	はい	グローバル	はい
audit_log_current_session			はい	両方	いいえ
audit_log_exclude_accounts	はい	はい	はい	グローバル	はい
audit_log_file	はい	はい	はい	グローバル	いいえ
audit_log_flush			はい	グローバル	はい
audit_log_format	はい	はい	はい	グローバル	いいえ
audit_log_include_accounts	はい	はい	はい	グローバル	はい
audit_log_policy	はい	はい	はい	グローバル	異なる
audit_log_rotate_on_size	はい	はい	はい	グローバル	はい
audit_log_statement_policy	はい	はい	はい	グローバル	はい
audit_log_strategy	はい	はい	はい	グローバル	いいえ
auto_increment_increment			はい	両方	はい
auto_increment_offset			はい	両方	はい
autocommit	はい	はい	はい	両方	はい
automatic_sp_privileges			はい	グローバル	はい
back_log			はい	グローバル	いいえ
basedir	はい	はい	はい	グローバル	いいえ
big_tables	はい	はい	はい	両方	はい
bind_address	はい	はい	はい	グローバル	いいえ
binlog_cache_size	はい	はい	はい	グローバル	はい
binlog_checksum			はい	グローバル	はい
binlog_direct_non_transactional_updates	はい	はい	はい	両方	はい
binlog_error_action	はい	はい	はい	両方	はい
binlog_format	はい	はい	はい	両方	はい
binlog_gtid_recovery_simplified	はい	はい	はい	グローバル	いいえ
binlog_max_flush_queue_time			はい	グローバル	はい
binlog_order_commits			はい	グローバル	はい
binlog_row_image	はい	はい	はい	両方	はい
binlog_rows_query_log_events			はい	両方	はい
binlog_stmt_cache_size	はい	はい	はい	グローバル	はい
binlogging_impossible_mode	はい	はい	はい	両方	はい

名前	コマンド行	オプションファイル	システム変数	変数スコープ	動的
block_encryption_mode	はい	はい	はい	両方	はい
bulk_insert_buffer_size	はい	はい	はい	両方	はい
character_set_client			はい	両方	はい
character_set_connection			はい	両方	はい
character_set_database (note 1)			はい	両方	はい
character_set_filesystem	はい	はい	はい	両方	はい
character_set_results			はい	両方	はい
character_set_server	はい	はい	はい	両方	はい
character_set_system			はい	グローバル	いいえ
character_sets_dir	はい	はい	はい	グローバル	いいえ
collation_connection			はい	両方	はい
collation_database (note 1)			はい	両方	はい
collation_server	はい	はい	はい	両方	はい
completion_type	はい	はい	はい	両方	はい
concurrent_insert	はい	はい	はい	グローバル	はい
connect_timeout	はい	はい	はい	グローバル	はい
core_file			はい	グローバル	いいえ
daemon_memcached_enable_binlog	はい	はい	はい	グローバル	いいえ
daemon_memcached_engine_lib_name	はい	はい	はい	グローバル	いいえ
daemon_memcached_engine_lib_path	はい	はい	はい	グローバル	いいえ
daemon_memcached_option	はい	はい	はい	グローバル	いいえ
daemon_memcached_r_batch_size	はい	はい	はい	グローバル	いいえ
daemon_memcached_w_batch_size	はい	はい	はい	グローバル	いいえ
datadir	はい	はい	はい	グローバル	いいえ
date_format			はい	グローバル	いいえ
datetime_format			はい	グローバル	いいえ
debug	はい	はい	はい	両方	はい
debug_sync			はい	セッション	はい
default_storage_engine	はい	はい	はい	両方	はい
default_tmp_storage_engine	はい	はい	はい	両方	はい
default_week_format	はい	はい	はい	両方	はい
delay_key_write	はい	はい	はい	グローバル	はい
delayed_insert_limit	はい	はい	はい	グローバル	はい
delayed_insert_timeout	はい	はい	はい	グローバル	はい
delayed_queue_size	はい	はい	はい	グローバル	はい
disable_gtid_unsafe_statements	はい	はい	はい	グローバル	いいえ
disable_gtid_unsafe_statements	はい	はい	はい	グローバル	いいえ
disconnect_on_expired_password	はい	はい	はい	セッション	いいえ
div_precision_increment	はい	はい	はい	両方	はい
end_markers_in_json			はい	両方	はい
enforce_gtid_consistency	はい	はい	はい	グローバル	いいえ
enforce_gtid_consistency	はい	はい	はい	グローバル	いいえ

名前	コマンド行	オプションファイル	システム変数	変数スコープ	動的
engine_condition_pushdown	はい	はい	はい	両方	はい
eq_range_index_dive_limit			はい	両方	はい
error_count			はい	セッション	いいえ
event_scheduler	はい	はい	はい	グローバル	はい
expire_logs_days	はい	はい	はい	グローバル	はい
explicit_defaults_for_timestamp	はい	はい	はい	セッション	いいえ
external_user			はい	セッション	いいえ
flush	はい	はい	はい	グローバル	はい
flush_time	はい	はい	はい	グローバル	はい
foreign_key_checks			はい	両方	はい
ft_boolean_syntax	はい	はい	はい	グローバル	はい
ft_max_word_len	はい	はい	はい	グローバル	いいえ
ft_min_word_len	はい	はい	はい	グローバル	いいえ
ft_query_expansion_limit	はい	はい	はい	グローバル	いいえ
ft_stopword_file	はい	はい	はい	グローバル	いいえ
general_log	はい	はい	はい	グローバル	はい
general_log_file	はい	はい	はい	グローバル	はい
group_concat_max_len	はい	はい	はい	両方	はい
gtid_done			はい	両方	いいえ
gtid_executed			はい	両方	いいえ
gtid_lost			はい	グローバル	いいえ
gtid_mode	はい	はい	はい	グローバル	いいえ
gtid_mode			はい	グローバル	いいえ
gtid_next			はい	セッション	はい
gtid_owned			はい	両方	いいえ
gtid_purged			はい	グローバル	はい
have_compress			はい	グローバル	いいえ
have_crypt			はい	グローバル	いいえ
have_csv			はい	グローバル	いいえ
have_dynamic_loading			はい	グローバル	いいえ
have_geometry			はい	グローバル	いいえ
have_innodb			はい	グローバル	いいえ
have_ndbcluster			はい	グローバル	いいえ
have_openssl			はい	グローバル	いいえ
have_partitioning			はい	グローバル	いいえ
have_profiling			はい	グローバル	いいえ
have_query_cache			はい	グローバル	いいえ
have_rtree_keys			はい	グローバル	いいえ
have_ssl			はい	グローバル	いいえ
have_symlink			はい	グローバル	いいえ
host_cache_size			はい	グローバル	はい
hostname			はい	グローバル	いいえ
identity			はい	セッション	はい

名前	コマンド行	オプションファイル	システム変数	変数スコープ	動的
ignore_builtin_innodb	はい	はい	はい	グローバル	いいえ
ignore_db_dirs			はい	グローバル	いいえ
init_connect	はい	はい	はい	グローバル	はい
init_file	はい	はい	はい	グローバル	いいえ
init_slave	はい	はい	はい	グローバル	はい
innodb_adaptive_flushing	はい	はい	はい	グローバル	はい
innodb_adaptive_flushing_lwm	はい	はい	はい	グローバル	はい
innodb_adaptive_hash_index	はい	はい	はい	グローバル	はい
innodb_adaptive_max_sleep_delay	はい	はい	はい	グローバル	はい
innodb_additional_mem_pool_size	はい	はい	はい	グローバル	いいえ
innodb_api_bk_commit_interval	はい	はい	はい	グローバル	はい
innodb_api_disable_rowlock	はい	はい	はい	グローバル	いいえ
innodb_api_enable_binlog	はい	はい	はい	グローバル	いいえ
innodb_api_enable_mdl	はい	はい	はい	グローバル	いいえ
innodb_api_trx_level	はい	はい	はい	グローバル	はい
innodb_autoextend_increment	はい	はい	はい	グローバル	はい
innodb_autoinc_lock_mode	はい	はい	はい	グローバル	いいえ
innodb_buffer_pool_dump_at_shutdown	はい	はい	はい	グローバル	はい
innodb_buffer_pool_dump_now	はい	はい	はい	グローバル	はい
innodb_buffer_pool_filename	はい	はい	はい	グローバル	はい
innodb_buffer_pool_instances	はい	はい	はい	グローバル	いいえ
innodb_buffer_pool_load_abort	はい	はい	はい	グローバル	はい
innodb_buffer_pool_load_at_startup	はい	はい	はい	グローバル	いいえ
innodb_buffer_pool_load_now	はい	はい	はい	グローバル	はい
innodb_buffer_pool_size	はい	はい	はい	グローバル	いいえ
innodb_change_buffer_max_size	はい	はい	はい	グローバル	はい
innodb_change_buffering	はい	はい	はい	グローバル	はい
innodb_checksum_algorithm	はい	はい	はい	グローバル	はい
innodb_checksums	はい	はい	はい	グローバル	いいえ
innodb_cmp_per_index_enabled	はい	はい	はい	グローバル	はい
innodb_commit_concurrency	はい	はい	はい	グローバル	はい
innodb_compression_failure_threshold_pct	はい	はい	はい	グローバル	はい
innodb_compression_level	はい	はい	はい	グローバル	はい
innodb_compression_pad_pct_max	はい	はい	はい	グローバル	はい
innodb_concurrency_tickets	はい	はい	はい	グローバル	はい
innodb_data_file_path	はい	はい	はい	グローバル	いいえ
innodb_data_home_dir	はい	はい	はい	グローバル	いいえ
innodb_disable_sort_file_cache	はい	はい	はい	グローバル	はい
innodb_doublewrite	はい	はい	はい	グローバル	いいえ
innodb_fast_shutdown	はい	はい	はい	グローバル	はい
innodb_file_format	はい	はい	はい	グローバル	はい
innodb_file_format_check	はい	はい	はい	グローバル	いいえ
innodb_file_format_max	はい	はい	はい	グローバル	はい

名前	コマンド行	オプションファイル	システム変数	変数スコープ	動的
innodb_file_per_table	はい	はい	はい	グローバル	はい
innodb_flush_log_at_timeout			はい	グローバル	はい
innodb_flush_log_at_trx_commit	はい	はい	はい	グローバル	はい
innodb_flush_method	はい	はい	はい	グローバル	いいえ
innodb_flush_neighbors	はい	はい	はい	グローバル	はい
innodb_flushing_avg_loops	はい	はい	はい	グローバル	はい
innodb_force_load_corrupted	はい	はい	はい	グローバル	いいえ
innodb_force_recovery	はい	はい	はい	グローバル	いいえ
innodb_ft_aux_table			はい	グローバル	はい
innodb_ft_cache_size	はい	はい	はい	グローバル	いいえ
innodb_ft_enable_diag_print	はい	はい	はい	グローバル	はい
innodb_ft_enable_stopword	はい	はい	はい	グローバル	はい
innodb_ft_max_token_size	はい	はい	はい	グローバル	いいえ
innodb_ft_min_token_size	はい	はい	はい	グローバル	いいえ
innodb_ft_num_word_optimize	はい	はい	はい	グローバル	はい
innodb_ft_result_cache_limit	はい	はい	はい	グローバル	はい
innodb_ft_server_stopword_table	はい	はい	はい	グローバル	はい
innodb_ft_sort_pll_degree	はい	はい	はい	グローバル	いいえ
innodb_ft_total_cache_size	はい	はい	はい	グローバル	いいえ
innodb_ft_user_stopword_table	はい	はい	はい	両方	はい
innodb_io_capacity	はい	はい	はい	グローバル	はい
innodb_io_capacity_max	はい	はい	はい	グローバル	はい
innodb_large_prefix	はい	はい	はい	グローバル	はい
innodb_lock_wait_timeout	はい	はい	はい	両方	はい
innodb_locks_unsafe_for_binlog	はい	はい	はい	グローバル	いいえ
innodb_log_buffer_size	はい	はい	はい	グローバル	いいえ
innodb_log_compressed_pages	はい	はい	はい	グローバル	はい
innodb_log_file_size	はい	はい	はい	グローバル	いいえ
innodb_log_files_in_group	はい	はい	はい	グローバル	いいえ
innodb_log_group_home_dir	はい	はい	はい	グローバル	いいえ
innodb_lru_scan_depth	はい	はい	はい	グローバル	はい
innodb_max_dirty_pages_pct	はい	はい	はい	グローバル	はい
innodb_max_dirty_pages_pct_lwm	はい	はい	はい	グローバル	はい
innodb_max_purge_lag	はい	はい	はい	グローバル	はい
innodb_max_purge_lag_delay	はい	はい	はい	グローバル	はい
innodb_mirrored_log_groups	はい	はい	はい	グローバル	いいえ
innodb_monitor_disable	はい	はい	はい	グローバル	はい
innodb_monitor_enable	はい	はい	はい	グローバル	はい
innodb_monitor_reset	はい	はい	はい	グローバル	はい
innodb_monitor_reset_all	はい	はい	はい	グローバル	はい
innodb_old_blocks_pct	はい	はい	はい	グローバル	はい
innodb_old_blocks_time	はい	はい	はい	グローバル	はい
innodb_online_alter_log_max_size	はい	はい	はい	グローバル	はい

名前	コマンド行	オプションファイル	システム変数	変数スコープ	動的
innodb_open_files	はい	はい	はい	グローバル	いいえ
innodb_optimize_fulltext_only	はい	はい	はい	グローバル	はい
innodb_page_size	はい	はい	はい	グローバル	いいえ
innodb_print_all_deadlocks	はい	はい	はい	グローバル	はい
innodb_purge_batch_size	はい	はい	はい	グローバル	はい
innodb_purge_threads	はい	はい	はい	グローバル	いいえ
innodb_random_read_ahead	はい	はい	はい	グローバル	はい
innodb_read_ahead_threshold	はい	はい	はい	グローバル	はい
innodb_read_io_threads	はい	はい	はい	グローバル	いいえ
innodb_read_only	はい	はい	はい	グローバル	いいえ
innodb_replication_delay	はい	はい	はい	グローバル	はい
innodb_rollback_on_timeout	はい	はい	はい	グローバル	いいえ
innodb_rollback_segments	はい	はい	はい	グローバル	はい
innodb_sort_buffer_size	はい	はい	はい	グローバル	いいえ
innodb_spin_wait_delay	はい	はい	はい	グローバル	はい
innodb_stats_auto_recalc	はい	はい	はい	グローバル	はい
innodb_stats_method	はい	はい	はい	グローバル	はい
innodb_stats_on_metadata	はい	はい	はい	グローバル	はい
innodb_stats_persistent	はい	はい	はい	グローバル	はい
innodb_stats_persistent_sample_pages	はい	はい	はい	グローバル	はい
innodb_stats_sample_pages	はい	はい	はい	グローバル	はい
innodb_stats_transient_sample_pages	はい	はい	はい	グローバル	はい
innodb_status_output	はい	はい	はい	グローバル	はい
innodb_status_output_locks	はい	はい	はい	グローバル	はい
innodb_strict_mode	はい	はい	はい	両方	はい
innodb_support_xa	はい	はい	はい	両方	はい
innodb_sync_array_size	はい	はい	はい	グローバル	いいえ
innodb_sync_spin_loops	はい	はい	はい	グローバル	はい
innodb_table_locks	はい	はい	はい	両方	はい
innodb_thread_concurrency	はい	はい	はい	グローバル	はい
innodb_thread_sleep_delay	はい	はい	はい	グローバル	はい
innodb_undo_directory	はい	はい	はい	グローバル	いいえ
innodb_undo_logs	はい	はい	はい	グローバル	はい
innodb_undo_tablespaces	はい	はい	はい	グローバル	いいえ
innodb_use_native_aio	はい	はい	はい	グローバル	いいえ
innodb_use_sys_malloc	はい	はい	はい	グローバル	いいえ
innodb_version			はい	グローバル	いいえ
innodb_write_io_threads	はい	はい	はい	グローバル	いいえ
insert_id			はい	セッション	はい
interactive_timeout	はい	はい	はい	両方	はい
join_buffer_size	はい	はい	はい	両方	はい
keep_files_on_create	はい	はい	はい	両方	はい
key_buffer_size	はい	はい	はい	グローバル	はい

名前	コマンド行	オプションファイル	システム変数	変数スコープ	動的
key_cache_age_threshold	はい	はい	はい	グローバル	はい
key_cache_block_size	はい	はい	はい	グローバル	はい
key_cache_division_limit	はい	はい	はい	グローバル	はい
language	はい	はい	はい	グローバル	いいえ
large_files_support			はい	グローバル	いいえ
large_page_size			はい	グローバル	いいえ
large_pages	はい	はい	はい	グローバル	いいえ
last_insert_id			はい	セッション	はい
lc_messages	はい	はい	はい	両方	はい
lc_messages_dir	はい	はい	はい	グローバル	いいえ
lc_time_names			はい	両方	はい
license			はい	グローバル	いいえ
local_infile			はい	グローバル	はい
lock_wait_timeout	はい	はい	はい	両方	はい
locked_in_memory			はい	グローバル	いいえ
log	はい	はい	はい	グローバル	はい
log_bin	はい	はい	はい	グローバル	いいえ
log_bin			はい	グローバル	いいえ
log_bin_basename			はい	グローバル	いいえ
log_bin_index			はい	グローバル	いいえ
log_bin_trust_function_creators	はい	はい	はい	グローバル	はい
log_bin_use_v1_row_events	はい	はい	はい	グローバル	いいえ
log_bin_use_v1_row_events	はい	はい	はい	グローバル	いいえ
log_error	はい	はい	はい	グローバル	いいえ
log_output	はい	はい	はい	グローバル	はい
log_queries_not_using_indexes	はい	はい	はい	グローバル	はい
log_slave_updates	はい	はい	はい	グローバル	いいえ
log_slave_updates	はい	はい	はい	グローバル	いいえ
log_slow_admin_statements			はい	グローバル	はい
log_slow_queries	はい	はい	はい	グローバル	はい
log_slow_slave_statements			はい	グローバル	はい
log_throttle_queries_not_using_indexes			はい	グローバル	はい
log_warnings	はい	はい	はい	異なる	はい
long_query_time	はい	はい	はい	両方	はい
low_priority_updates	はい	はい	はい	両方	はい
lower_case_file_system			はい	グローバル	いいえ
lower_case_table_names	はい	はい	はい	グローバル	いいえ
master_info_repository	はい	はい	はい	グローバル	はい
master_verify_checksum			はい	グローバル	はい
max_allowed_packet	はい	はい	はい	グローバル	はい
max_binlog_cache_size	はい	はい	はい	グローバル	はい
max_binlog_size	はい	はい	はい	グローバル	はい
max_binlog_stmt_cache_size	はい	はい	はい	グローバル	はい

名前	コマンド行	オプションファイル	システム変数	変数スコープ	動的
max_connect_errors	はい	はい	はい	グローバル	はい
max_connections	はい	はい	はい	グローバル	はい
max_delayed_threads	はい	はい	はい	両方	はい
max_error_count	はい	はい	はい	両方	はい
max_heap_table_size	はい	はい	はい	両方	はい
max_insert_delayed_threads			はい	両方	はい
max_join_size	はい	はい	はい	両方	はい
max_length_for_sort_data	はい	はい	はい	両方	はい
max_prepared_stmt_count	はい	はい	はい	グローバル	はい
max_relay_log_size	はい	はい	はい	グローバル	はい
max_seeks_for_key	はい	はい	はい	両方	はい
max_sort_length	はい	はい	はい	両方	はい
max_sp_recursion_depth	はい	はい	はい	両方	はい
max_user_connections	はい	はい	はい	両方	はい
max_write_lock_count	はい	はい	はい	グローバル	はい
metadata_locks_cache_size			はい	グローバル	いいえ
metadata_locks_hash_instances			はい	グローバル	いいえ
min_examined_row_limit	はい	はい	はい	両方	はい
myisam_data_pointer_size	はい	はい	はい	グローバル	はい
myisam_max_sort_file_size	はい	はい	はい	グローバル	はい
myisam_mmap_size	はい	はい	はい	グローバル	いいえ
myisam_recover_options			はい	グローバル	いいえ
myisam_repair_threads	はい	はい	はい	両方	はい
myisam_sort_buffer_size	はい	はい	はい	両方	はい
myisam_stats_method	はい	はい	はい	両方	はい
myisam_use_mmap	はい	はい	はい	グローバル	はい
named_pipe			はい	グローバル	いいえ
ndb_autoincrement_prefetch_sz	はい	はい	はい	両方	はい
ndb_batch_size	はい	はい	はい	グローバル	いいえ
ndb_blob_read_batch_bytes	はい	はい	はい	両方	はい
ndb_blob_write_batch_bytes	はい	はい	はい	両方	はい
ndb_cache_check_time	はい	はい	はい	グローバル	はい
ndb_cluster_connection_pool	はい	はい	はい	グローバル	いいえ
ndb_deferred_constraints	はい	はい	はい	両方	はい
ndb_deferred_constraints	はい	はい	はい	両方	はい
ndb_distribution	はい	はい	はい	グローバル	はい
ndb_distribution	はい	はい	はい	グローバル	はい
ndb_eventbuffer_max_alloc	はい	はい	はい	グローバル	はい
ndb_extra_logging	はい	はい	はい	グローバル	はい
ndb_force_send	はい	はい	はい	両方	はい
ndb_index_stat_cache_entries	はい	はい	はい	両方	はい
ndb_index_stat_enable	はい	はい	はい	両方	はい
ndb_index_stat_option	はい	はい	はい	両方	はい

名前	コマンド行	オプションファイル	システム変数	変数スコープ	動的
ndb_index_stat_update_freq	はい	はい	はい	両方	はい
ndb_join_pushdown			はい	両方	はい
ndb_log_apply_status	はい	はい	はい	グローバル	いいえ
ndb_log_apply_status	はい	はい	はい	グローバル	いいえ
ndb_log_bin	はい		はい	両方	はい
ndb_log_binlog_index	はい		はい	グローバル	はい
ndb_log_empty_epochs	はい	はい	はい	グローバル	はい
ndb_log_empty_epochs	はい	はい	はい	グローバル	はい
ndb_log_exclusive_reads	はい	はい	はい	両方	はい
ndb_log_exclusive_reads	はい	はい	はい	両方	はい
ndb_log_orig	はい	はい	はい	グローバル	いいえ
ndb_log_orig	はい	はい	はい	グローバル	いいえ
ndb_log_transaction_id	はい	はい	はい	グローバル	いいえ
ndb_log_transaction_id			はい	グローバル	いいえ
ndb_log_update_as_write	はい	はい	はい	グローバル	はい
ndb_log_updated_only	はい	はい	はい	グローバル	はい
ndb_optimization_delay			はい	グローバル	はい
ndb_optimized_node_selection	はい	はい	はい	グローバル	いいえ
ndb_recv_thread_activation_threshold			はい	グローバル	はい
ndb_recv_thread_cpu_mask			はい	グローバル	はい
ndb_show_foreign_key_mock_tables	はい	はい	はい	グローバル	はい
ndb_slave_conflict_role	はい	はい	はい	グローバル	はい
Ndb_slave_last_conflict_epoch			はい	グローバル	いいえ
Ndb_slave_max_replicated_epoch			はい	グローバル	いいえ
ndb_table_no_logging			はい	セッション	はい
ndb_table_temporary			はい	セッション	はい
ndb_use_copying_alter_table			はい	両方	いいえ
ndb_use_exact_count			はい	両方	はい
ndb_use_transactions	はい	はい	はい	両方	はい
ndb_version			はい	グローバル	いいえ
ndb_version_string			はい	グローバル	いいえ
ndb_wait_connected	はい	はい	はい	グローバル	いいえ
ndb_wait_setup	はい	はい	はい	グローバル	いいえ
ndbinfo_database			はい	グローバル	いいえ
ndbinfo_max_bytes	はい		はい	両方	はい
ndbinfo_max_rows	はい		はい	両方	はい
ndbinfo_offline			はい	グローバル	はい
ndbinfo_show_hidden	はい		はい	両方	はい
ndbinfo_table_prefix	はい		はい	両方	はい
ndbinfo_version			はい	グローバル	いいえ
net_buffer_length	はい	はい	はい	両方	はい
net_read_timeout	はい	はい	はい	両方	はい
net_retry_count	はい	はい	はい	両方	はい

名前	コマンド行	オプションファイル	システム変数	変数スコープ	動的
net_write_timeout	はい	はい	はい	両方	はい
new	はい	はい	はい	両方	はい
old	はい	はい	はい	グローバル	いいえ
old_alter_table	はい	はい	はい	両方	はい
old_passwords			はい	両方	はい
open_files_limit	はい	はい	はい	グローバル	いいえ
optimizer_join_cache_level	はい	はい	はい	両方	はい
optimizer_prune_level	はい	はい	はい	両方	はい
optimizer_search_depth	はい	はい	はい	両方	はい
optimizer_switch	はい	はい	はい	両方	はい
optimizer_trace			はい	両方	はい
optimizer_trace_features			はい	両方	はい
optimizer_trace_limit			はい	両方	はい
optimizer_trace_max_mem_size			はい	両方	はい
optimizer_trace_offset			はい	両方	はい
performance_schema	はい	はい	はい	グローバル	いいえ
performance_schema_accounts_size	はい	はい	はい	グローバル	いいえ
performance_schema_digests_size	はい	はい	はい	グローバル	いいえ
performance_schema_events_stages_history_long_size	はい	はい	はい	グローバル	いいえ
performance_schema_events_stages_history_size	はい	はい	はい	グローバル	いいえ
performance_schema_events_statements_history_long_size	はい	はい	はい	グローバル	いいえ
performance_schema_events_statements_history_size	はい	はい	はい	グローバル	いいえ
performance_schema_events_waits_history_long_size	はい	はい	はい	グローバル	いいえ
performance_schema_events_waits_history_size	はい	はい	はい	グローバル	いいえ
performance_schema_hosts_size	はい	はい	はい	グローバル	いいえ
performance_schema_max_cond_classes	はい	はい	はい	グローバル	いいえ
performance_schema_max_cond_instances	はい	はい	はい	グローバル	いいえ
performance_schema_max_file_classes	はい	はい	はい	グローバル	いいえ
performance_schema_max_file_handles	はい	はい	はい	グローバル	いいえ
performance_schema_max_file_instances	はい	はい	はい	グローバル	いいえ
performance_schema_max_mutex_classes	はい	はい	はい	グローバル	いいえ
performance_schema_max_mutex_instances	はい	はい	はい	グローバル	いいえ
performance_schema_max_rwlock_classes	はい	はい	はい	グローバル	いいえ
performance_schema_max_rwlock_instances	はい	はい	はい	グローバル	いいえ
performance_schema_max_socket_classes	はい	はい	はい	グローバル	いいえ
performance_schema_max_socket_instances	はい	はい	はい	グローバル	いいえ
performance_schema_max_stage_classes	はい	はい	はい	グローバル	いいえ
performance_schema_max_statement_classes	はい	はい	はい	グローバル	いいえ
performance_schema_max_table_handles	はい	はい	はい	グローバル	いいえ
performance_schema_max_table_instances	はい	はい	はい	グローバル	いいえ
performance_schema_max_thread_classes	はい	はい	はい	グローバル	いいえ
performance_schema_max_thread_instances	はい	はい	はい	グローバル	いいえ
performance_schema_session_connect_attrs_size	はい	はい	はい	グローバル	いいえ

名前	コマンド行	オプションファイル	システム変数	変数スコープ	動的
performance_schema_setup_actors_size	はい	はい	はい	グローバル	いいえ
performance_schema_setup_objects_size	はい	はい	はい	グローバル	いいえ
performance_schema_users_size	はい	はい	はい	グローバル	いいえ
pid_file	はい	はい	はい	グローバル	いいえ
plugin_dir	はい	はい	はい	グローバル	いいえ
port	はい	はい	はい	グローバル	いいえ
preload_buffer_size	はい	はい	はい	両方	はい
profiling			はい	両方	はい
profiling_history_size	はい	はい	はい	両方	はい
protocol_version			はい	グローバル	いいえ
proxy_user			はい	セッション	いいえ
pseudo_slave_mode			はい	セッション	はい
pseudo_thread_id			はい	セッション	はい
query_alloc_block_size	はい	はい	はい	両方	はい
query_cache_limit	はい	はい	はい	グローバル	はい
query_cache_min_res_unit	はい	はい	はい	グローバル	はい
query_cache_size	はい	はい	はい	グローバル	はい
query_cache_type	はい	はい	はい	両方	はい
query_cache_wlock_invalidate	はい	はい	はい	両方	はい
query_prealloc_size	はい	はい	はい	両方	はい
rand_seed1			はい	セッション	はい
rand_seed2			はい	セッション	はい
range_alloc_block_size	はい	はい	はい	両方	はい
read_buffer_size	はい	はい	はい	両方	はい
read_only	はい	はい	はい	グローバル	はい
read_rnd_buffer_size	はい	はい	はい	両方	はい
relay_log	はい	はい	はい	グローバル	いいえ
relay_log_basename			はい	グローバル	いいえ
relay_log_index	はい	はい	はい	グローバル	いいえ
relay_log_index	はい	はい	はい	グローバル	いいえ
relay_log_info_file	はい	はい	はい	グローバル	いいえ
relay_log_info_repository			はい	グローバル	はい
relay_log_purge	はい	はい	はい	グローバル	はい
relay_log_recovery	はい	はい	はい	グローバル	異なる
relay_log_space_limit	はい	はい	はい	グローバル	いいえ
report_host	はい	はい	はい	グローバル	いいえ
report_password	はい	はい	はい	グローバル	いいえ
report_port	はい	はい	はい	グローバル	いいえ
report_user	はい	はい	はい	グローバル	いいえ
rpl_semi_sync_master_enabled			はい	グローバル	はい
rpl_semi_sync_master_timeout			はい	グローバル	はい
rpl_semi_sync_master_trace_level			はい	グローバル	はい
rpl_semi_sync_master_wait_no_slave			はい	グローバル	はい

名前	コマンド行	オプションファイル	システム変数	変数スコープ	動的
rpl_semi_sync_slave_enabled			はい	グローバル	はい
rpl_semi_sync_slave_trace_level			はい	グローバル	はい
rpl_stop_slave_timeout	はい	はい	はい	グローバル	はい
secure_auth	はい	はい	はい	グローバル	はい
secure_file_priv	はい	はい	はい	グローバル	いいえ
server_id	はい	はい	はい	グローバル	はい
server_id_bits	はい	はい	はい	グローバル	いいえ
server_id_bits	はい	はい	はい	グローバル	いいえ
server_uuid			はい	グローバル	いいえ
sha256_password_private_key_path			はい	グローバル	いいえ
sha256_password_public_key_path			はい	グローバル	いいえ
shared_memory	はい	はい	はい	グローバル	いいえ
shared_memory_base_name			はい	グローバル	いいえ
simplified_binlog_gtid_recovery	はい	はい	はい	グローバル	いいえ
skip_external_locking	はい	はい	はい	グローバル	いいえ
skip_name_resolve	はい	はい	はい	グローバル	いいえ
skip_networking	はい	はい	はい	グローバル	いいえ
skip_show_database	はい	はい	はい	グローバル	いいえ
slave_allow_batching	はい	はい	はい	グローバル	はい
slave_checkpoint_group	はい	はい	はい	グローバル	はい
slave_checkpoint_period	はい	はい	はい	グローバル	はい
slave_compressed_protocol	はい	はい	はい	グローバル	はい
slave_exec_mode	はい	はい	はい	グローバル	はい
slave_load_tmpdir	はい	はい	はい	グローバル	いいえ
slave_max_allowed_packet			はい	グローバル	はい
slave_net_timeout	はい	はい	はい	グローバル	はい
slave_parallel_workers	はい		はい	グローバル	はい
slave_pending_jobs_size_max			はい	グローバル	はい
slave_rows_search_algorithms			はい	グローバル	はい
slave_skip_errors	はい	はい	はい	グローバル	いいえ
slave_sql_verify_checksum			はい	グローバル	はい
slave_transaction_retries	はい	はい	はい	グローバル	はい
slave_type_conversions	はい	はい	はい	グローバル	いいえ
slow_launch_time	はい	はい	はい	グローバル	はい
slow_query_log	はい	はい	はい	グローバル	はい
slow_query_log_file	はい	はい	はい	グローバル	はい
socket	はい	はい	はい	グローバル	いいえ
sort_buffer_size	はい	はい	はい	両方	はい
sql_auto_is_null			はい	両方	はい
sql_big_selects			はい	両方	はい
sql_big_tables			はい	両方	はい
sql_buffer_result			はい	両方	はい
sql_log_bin			はい	両方	はい

名前	コマンド行	オプションファイル	システム変数	変数スコープ	動的
sql_log_off			はい	両方	はい
sql_low_priority_updates			はい	両方	はい
sql_max_join_size			はい	両方	はい
sql_mode	はい	はい	はい	両方	はい
sql_notes			はい	両方	はい
sql_quote_show_create			はい	両方	はい
sql_safe_updates			はい	両方	はい
sql_select_limit			はい	両方	はい
sql_slave_skip_counter			はい	グローバル	はい
sql_warnings			はい	両方	はい
ssl_ca	はい	はい	はい	グローバル	いいえ
ssl_capath	はい	はい	はい	グローバル	いいえ
ssl_cert	はい	はい	はい	グローバル	いいえ
ssl_cipher	はい	はい	はい	グローバル	いいえ
ssl_crl	はい	はい	はい	グローバル	いいえ
ssl_crlpath	はい	はい	はい	グローバル	いいえ
ssl_key	はい	はい	はい	グローバル	いいえ
storage_engine			はい	両方	はい
stored_program_cache	はい	はい	はい	グローバル	はい
sync_binlog	はい	はい	はい	グローバル	はい
sync_frm	はい	はい	はい	グローバル	はい
sync_master_info	はい	はい	はい	グローバル	はい
sync_relay_log	はい	はい	はい	グローバル	はい
sync_relay_log_info	はい	はい	はい	グローバル	はい
system_time_zone			はい	グローバル	いいえ
table_definition_cache			はい	グローバル	はい
table_open_cache			はい	グローバル	はい
table_open_cache_instances			はい	グローバル	いいえ
thread_cache_size	はい	はい	はい	グローバル	はい
thread_concurrency	はい	はい	はい	グローバル	いいえ
thread_handling	はい	はい	はい	グローバル	いいえ
thread_pool_algorithm	はい	はい	はい	グローバル	いいえ
thread_pool_high_priority_connection	はい	はい	はい	両方	はい
thread_pool_max_unused_threads	はい	はい	はい	グローバル	はい
thread_pool_prio_kickup_timer	はい	はい	はい	両方	はい
thread_pool_size	はい	はい	はい	グローバル	いいえ
thread_pool_stall_limit	はい	はい	はい	グローバル	はい
thread_stack	はい	はい	はい	グローバル	いいえ
time_format			はい	グローバル	いいえ
time_zone			はい	両方	はい
timed_mutexes	はい	はい	はい	グローバル	はい
timestamp			はい	セッション	はい
tmp_table_size	はい	はい	はい	両方	はい

名前	コマンド行	オプションファイル	システム変数	変数スコープ	動的
tmpdir	はい	はい	はい	グローバル	いいえ
transaction_alloc_block_size	はい	はい	はい	両方	はい
transaction_allow_batching			はい	セッション	はい
transaction_prealloc_size	はい	はい	はい	両方	はい
tx_isolation			はい	両方	はい
tx_read_only			はい	両方	はい
unique_checks			はい	両方	はい
updatable_views_with_limit	はい	はい	はい	両方	はい
validate_password_dictionary_file			はい	グローバル	異なる
validate_password_length			はい	グローバル	はい
validate_password_mixed_case_count			はい	グローバル	はい
validate_password_number_count			はい	グローバル	はい
validate_password_policy			はい	グローバル	はい
validate_password_special_char_count			はい	グローバル	はい
validate_user_plugins			はい	グローバル	いいえ
version			はい	グローバル	いいえ
version_comment			はい	グローバル	いいえ
version_compile_machine			はい	グローバル	いいえ
version_compile_os			はい	グローバル	いいえ
wait_timeout	はい	はい	はい	両方	はい
warning_count			はい	セッション	いいえ

Notes:

1. このオプションは動的ですが、サーバーのみがこの情報を設定するようにしてください。この変数の値は手動で設定しないでください。

追加のシステム変数情報については、次のセクションを参照してください。

- セクション5.1.5「システム変数の使用」では、システム変数値の設定および表示の構文について説明します。
- セクション5.1.5.2「動的システム変数」では、実行時に設定できる変数を一覧表示しています。
- システム変数の調整に関する情報は、セクション8.11.2「サーバーパラメータのチューニング」を参照してください。
- セクション14.12「InnoDB の起動オプションおよびシステム変数」では、InnoDB システム変数を一覧表示しています。
- セクション18.3.4.3「MySQL Cluster のシステム変数」では、MySQL クラスタに固有のシステム変数をリストしています。
- レプリケーションに固有のサーバーシステム変数については、セクション17.1.4「レプリケーションおよびバイナリロギングのオプションと変数」を参照してください。

> **注記**
>
> 次の変数説明の一部では、変数を「有効にする」または「無効にする」ことについて述べています。これらの変数は SET ステートメントを ON または 1 に設定すると有効になり、あるいは OFF または 0 に設定すると無効になります。ただし、MySQL 5.6.2 より前では、このような値をコマンド行またはオプションファイルで設定するには 1 または 0 で設定する必要があり、ON または OFF に設定しても機能しません。たとえば、コマンド行において、--delay_key_write=1 は機能しますが、--delay_key_write=ON は機能しません。MySQL 5.6.2 以降では、起動時にブール変数を ON、TRUE、OFF、および FALSE に設定できます (大文字小文字を区別しない)。セクション4.2.5「プログラムオプション修飾子」を参照してください。

一部のシステム変数はバッファーまたはキャッシュのサイズを制御します。所定のバッファーについて、サーバーは内部データ構造を割り当てる必要がある場合もあります。これらの構造は、バッファーに割り当てられた合計メモリーから割り当てられ、必要なスペースの量はプラットフォームに依存することがあります。つまり、バッファーサイズを制御するシステム変数に値を割り当てたとき、実際に使用可能なスペースの量が、割り当てられた値と異なる場合もあることを意味します。一部の場合では、この量は割り当てられた値より少ないこともあります。また、サーバーが値を上方に調整することも考えられます。たとえば、最小値が 1024 の変数に値 0 を割り当てた場合、サーバーは値を 1024 に設定します。

バッファーサイズ、長さ、およびスタックサイズの値は、別途指定しないかぎりバイト単位で指定されます。

一部のシステム変数はファイル名の値を取ります。別途指定しないかぎり、値が相対パス名であれば、デフォルトのファイルの場所はデータディレクトリです。場所を明示的に指定するには、絶対パス名を使用します。たとえばデータディレクトリが /var/mysql/data だとします。ファイル値の変数が相対パス名として指定された場合、ファイルは /var/mysql/data の下に配置されます。値が絶対パス名である場合、その場所はパス名によって指定されます。

- authentication_windows_log_level

コマンド行形式	--authentication-windows-log-level
導入	5.6.10
型	数値
デフォルト	0
最小値	0
最大値	4

この変数は、authentication_windows Windows 認証プラグインが使用可能で、デバッグコードが有効な場合のみ使用できます。セクション6.3.8.6「Windows ネイティブ認証プラグイン」を参照してください。

この変数は、Windows 認証プラグインのロギングレベルを設定します。次の表は、許可される値を示しています。

値	説明
0	ロギングなし
1	エラーメッセージのみログに記録します
2	レベル 1 メッセージおよび警告メッセージをログに記録します
3	レベル 2 メッセージおよび情報メモをログに記録します
4	レベル 3 メッセージおよびデバッグメッセージをログに記録します

この変数は MySQL 5.6.10 で追加されました。

- authentication_windows_use_principal_name

コマンド行形式	--authentication-windows-use-principal-name
導入	5.6.10
型	ブール
デフォルト	ON

この変数は、authentication_windows Windows 認証プラグインが使用可能な場合のみ使用できます。セクション6.3.8.6「Windows ネイティブ認証プラグイン」を参照してください。

InitSecurityContext() 関数を使用して認証するクライアントは、接続するサービスを識別する文字列を提供する必要があります (targetName)。MySQL は、サーバーが実行するアカウントの主体名 (UPN) を使用します。UPN は user_id@computer_name という形式で、使用される場所に登録される必要はありません。この UPN は、認証ハンドシェイクの最初にサーバーによって送信されます。

この変数は、サーバーが初期チャレンジで UPN を送信するかどうかを制御します。デフォルトでは、変数は有効になっています。セキュリティー上の理由で、サーバーのアカウント名を平文でクライアントに送信しないようにするために、これを無効にできます。変数が無効な場合、サーバーは最初のチャレンジで常に 0x00 バイトを送信し、クライアントは targetName を指定せず、結果として NTLM 認証が使用されます。

サーバーがその UPN を取得できない場合 (これは Kerberos 認証をサポートしない環境で主に発生します)、UPN はサーバーによって送信されず、NTLM 認証が使用されます。

この変数は MySQL 5.6.10 で追加されました。

- autocommit

コマンド行形式	--autocommit[=#]
システム変数	autocommit
スコープ	グローバル、セッション
動的	はい
型	ブール
デフォルト	ON

自動コミットモード。1 に設定された場合、テーブルへのすべての変更はすぐに有効になります。0 に設定した場合、COMMIT を使用してトランザクションを受け入れるか、ROLLBACK でトランザクションをキャンセルする必要があります。autocommit が 0 で、これを 1 に変更した場合、MySQL はオープン中のすべてのトランザクションの自動的な COMMIT を実行します。トランザクションを始める別の方法は、START TRANSACTION または BEGIN ステートメントを利用する方法です。セクション13.3.1「START TRANSACTION、COMMIT、および ROLLBACK 構文」を参照してください。

デフォルトでは、クライアント接続は autocommit を 1 に設定して開始されます。デフォルト 0 でクライアントを開始させるには、--autocommit=0 オプションを使用してサーバーを開始することによって、グローバルな autocommit 値を設定します。オプションファイルを使用して変数を設定するには、次の行を含めます。

```
[mysqld]
autocommit=0
```

- automatic_sp_privileges

システム変数	automatic_sp_privileges
スコープ	グローバル
動的	はい
型	ブール
デフォルト	TRUE

この変数の値が 1 (デフォルト) のとき、ユーザーがルーチンを実行して変更したりドロップしたりできない場合、サーバーは自動的に EXECUTE および ALTER ROUTINE の権限をストアドルーチンの作成者に付与します。(ルーチンをドロップするには ALTER ROUTINE 権限が必要です。)ルーチンがドロップされると、サーバーはそれらの権限を作成者から自動的にドロップします。automatic_sp_privileges が 0 の場合、サーバーはこれらの権限を自動的に追加またはドロップしません。

ルーチンの作成者は、ルーチンの CREATE ステートメントを実行するために使用されるアカウントです。これは、ルーチン定義で DEFINER として名前が指定されているアカウントと同じでないことがあります。

セクション20.2.2「ストアドルーチンと MySQL 権限」も参照してください。

- back_log

システム変数	back_log
スコープ	グローバル
動的	いいえ
型	数値
デフォルト (≥ 5.6.6)	-1 (autosized)
デフォルト (≤ 5.6.5)	50
最小値	1

最大値	65535

MySQL で保持できる未処理の接続リクエストの数。これはメインの MySQL スレッドが非常に短時間で非常に多くの接続リクエストを受け取る場合に効果をあらわします。次に、メインスレッドが接続を検査し新規スレッドを開始するまで (非常に短いですが) 少し時間がかかります。back_log 値は、MySQL が新規リクエストへの回答を一時的に停止するまでの短い時間に、スタック可能なリクエストの数を示します。短い時間に大量の接続が予想される場合にかぎり、これを増加する必要があります。

つまり、この値は着信 TCP/IP 接続の listen キューのサイズです。使用しているオペレーティングシステムには、このキューのサイズについて独自の制限があります。UNIX listen() システムコールのマニュアルページに、詳細情報があります。この変数の最大値については OS のドキュメントを確認してください。back_log をオペレーティングシステムの制限を超える設定はできません。

MySQL 5.6.6 以降では、デフォルト値は次の数式に基づき、900 の上限までに制限されます。

```
50 + (max_connections / 5)
```

5.6.6 より前では、デフォルトは 50 です。

- basedir

コマンド行形式	--basedir=path
システム変数	basedir
スコープ	グローバル
動的	いいえ
型	ディレクトリ名

MySQL インストールの基本ディレクトリ。この変数は、--basedir オプションで設定できます。ほかの変数の相対パス名は、通常は基本ディレクトリを基準として解決されます。

- big_tables

コマンド行形式	--big-tables
システム変数	big_tables
スコープ	グローバル、セッション
動的	はい
型	ブール
デフォルト	OFF

1 に設定された場合、すべての一時テーブルはメモリーでなくディスクに保管されます。これはスピードが少し遅くなりますが、エラー The table tbl_name is full は、大きい一時テーブルを必要とする SELECT オペレーションでは発生しません。新規接続のデフォルト値は 0 です (インメモリーの一時テーブルを使用します)。通常は、インメモリーテーブルが必要に応じてディスクベースのテーブルに自動的に変換されるため、この変数をふたたび設定する必要はありません。

- bind_address

コマンド行形式	--bind-address=addr
システム変数 (≥ 5.6.1)	bind_address
スコープ (≥ 5.6.1)	グローバル
動的 (≥ 5.6.1)	いいえ
型	文字列
デフォルト (≥ 5.6.6)	*
デフォルト (≤ 5.6.5)	0.0.0.0

--bind-address オプションの値。この変数は MySQL 5.6.1 で追加されました。

- block_encryption_mode

コマンド行形式	--block-encryption-mode=#
導入	5.6.17
システム変数	block_encryption_mode
スコープ	グローバル、セッション
動的	はい
型	文字列
デフォルト	aes-128-ecb

この変数は、AES などのブロックベースのアルゴリズムのブロック暗号化モードを制御します。これは AES_ENCRYPT() および AES_DECRYPT() の暗号化に影響します。

block_encryption_mode は aes-keylen-mode 形式の値を取り、ここで keylen はビット単位の鍵の長さ、mode は暗号化モードです。この値は大文字と小文字を区別しません。許可される keylen 値は 128、192、および 256 です。許可される暗号化モードは MySQL が OpenSSL または yaSSL のいずれを使用して構築されているかによって異なります。

- OpenSSL の場合に許可される mode 値: ECB、CBC、CFB1、CFB8、CFB128、OFB
- yaSSL の場合に許可される mode 値: ECB、CBC

たとえば次のステートメントでは、AES 暗号化機能が 256 ビットの鍵の長さおよび CBC モードを使用します。

```
SET block_encryption_mode = 'aes-256-cbc';
```

サポートされない鍵の長さや SSL ライブラリがサポートしないモードを含む値に block_encryption_mode を設定しようとすると、エラーが発生します。

この変数は MySQL 5.6.17 で追加されました。

- bulk_insert_buffer_size

コマンド行形式	--bulk-insert-buffer-size=#
システム変数	bulk_insert_buffer_size
スコープ	グローバル、セッション
動的	はい
型	数値
デフォルト	8388608
最小値	0
最大値 (64 ビットプラットフォーム)	18446744073709551615
最大値 (32 ビットプラットフォーム)	4294967295

MyISAM では、空ではないテーブルにデータを追加するとき、INSERT ... SELECT、INSERT ... VALUES (...), (...), ...、および LOAD DATA INFILE の一括挿入をさらに高速にするツリー状の特殊なキャッシュを使用します。この変数は、スレッドあたりのバイト単位のキャッシュツリーのサイズを制限します。これを 0 に設定すると、この最適化が無効になります。デフォルトの値は 8M バイトです。

- character_set_client

システム変数	character_set_client
スコープ	グローバル、セッション
動的	はい

型	文字列

クライアントから到達するステートメントの文字セット。この変数のセッション値は、クライアントがサーバーに接続するときにクライアントによってリクエストされる文字セットを使用して設定されます。(多くのクライアントは、この文字セットを明示的に指定するための --default-character-set オプションをサポートします。セクション10.1.4「接続文字セットおよび照合順序」も参照してください。)クライアントがリクエストする値が不明または利用できないか、サーバーがクライアントリクエストを無視するように構成されている場合、セッション値を設定するよう変数のグローバル値が使用されます。

- クライアントの MySQL バージョンが MySQL 4.1 よりも古いため、文字セットをリクエストしない場合。
- クライアントがリクエストする文字セットがサーバーで認識されない場合。たとえば、日本語に対応したクライアントが、sjis サポートを構成されていないサーバーに接続するときに sjis をリクエストする場合があります。
- mysqld が --skip-character-set-client-handshake オプションを使用して開始された場合、これによってクライアント文字セット構成が無視されます。これによって MySQL 4.0 の動作が再現されるため、すべてのクライアントをアップグレードしないでサーバーをアップグレードする場合に便利です。

ucs2、utf16、utf16le、および utf32 をクライアント文字セットとして使用することはできず、つまりこれらは SET NAMES または SET CHARACTER SET でも機能しないことを意味します。

- character_set_connection

システム変数	character_set_connection
スコープ	グローバル、セッション
動的	はい
型	文字列

文字セットイントロデューサを持たないリテラル用、および数字から文字列への変換用に使用される文字セット。

- character_set_database

システム変数	character_set_database
スコープ	グローバル、セッション
動的	はい
型	文字列
脚注	このオプションは動的ですが、サーバーのみがこの情報を設定するようにしてください。この変数の値は手動で設定しないでください。

デフォルトデータベースで使用される文字セット。デフォルトのデータベースが変更されるたびに、サーバーはこの変数を設定します。デフォルトデータベースが存在しない場合、変数は character_set_server と同じ値になります。

- character_set_filesystem

コマンド行形式	--character-set-filesystem=name
システム変数	character_set_filesystem
スコープ	グローバル、セッション
動的	はい
型	文字列
デフォルト	binary

ファイルシステムの文字セット。この変数は、LOAD DATA INFILE および SELECT ... INTO OUTFILE ステートメントや LOAD_FILE() 関数などのファイル名を参照する文字列リテラルを解釈するために使用されます。このようなファイル名は、ファイルを開くよう試行する前に character_set_client から character_set_filesystem に変換されます。デフォルト値は binary で、変換が行われないことを意味します。マルチバイトファイル名が許可されるシステムについては、別の値が適切な場合もあります。たとえば、システムが UTF-8 を使用してファイル名を表現する場合、character_set_filesystem を 'utf8' に設定します。

- character_set_results

システム変数	character_set_results
スコープ	グローバル、セッション
動的	はい
型	文字列

結果セットやエラーメッセージなどのクエリー結果をクライアントに返すために使用される文字セット。

- character_set_server

コマンド行形式	--character-set-server
システム変数	character_set_server
スコープ	グローバル、セッション
動的	はい
型	文字列
デフォルト	latin1

サーバーのデフォルト文字セット。

- character_set_system

システム変数	character_set_system
スコープ	グローバル
動的	いいえ
型	文字列
デフォルト	utf8

識別子を格納するためにサーバーで使用される文字セット。この値は常に utf8 です。

- character_sets_dir

コマンド行形式	--character-sets-dir=path
システム変数	character_sets_dir
スコープ	グローバル
動的	いいえ
型	ディレクトリ名

文字セットがインストールされているディレクトリ。

- collation_connection

システム変数	collation_connection
スコープ	グローバル、セッション
動的	はい
型	文字列

接続文字セットの照合順序。

- collation_database

システム変数	collation_database
スコープ	グローバル、セッション
動的	はい
型	文字列

脚注	このオプションは動的ですが、サーバーのみがこの情報を設定するようにしてください。この変数の値は手動で設定しないでください。

デフォルトデータベースで使用される照合。デフォルトのデータベースが変更されるたびに、サーバーはこの変数を設定します。デフォルトデータベースが存在しない場合、変数は collation_server と同じ値になります。

- collation_server

コマンド行形式	--collation-server
システム変数	collation_server
スコープ	グローバル、セッション
動的	はい
型	文字列
デフォルト	latin1_swedish_ci

サーバーのデフォルトの照合順序。

- completion_type

コマンド行形式	--completion-type=#
システム変数	completion_type
スコープ	グローバル、セッション
動的	はい
型	列挙
デフォルト	NO_CHAIN
有効な値	NO_CHAIN CHAIN RELEASE 0 1 2

トランザクション完了タイプ。この変数は、次の表に示す値を取ることができます。変数は、名前の値か対応する整数値のいずれかを使用して割り当てることができます。

値	説明
NO_CHAIN (または 0)	COMMIT および ROLLBACK は影響されません。これはデフォルト値です。
CHAIN (または 1)	COMMIT および ROLLBACK は、それぞれ COMMIT AND CHAIN および ROLLBACK AND CHAIN と同等です。(終了したばかりのトランザクションと同じ分離レベルで新規トランザクションがすぐに開始します。)
RELEASE (または 2)	COMMIT および ROLLBACK は、それぞれ COMMIT RELEASE および ROLLBACK RELEASE と同等です。(サーバーはトランザクションの終了後に切断されます。)

completion_type は、START TRANSACTION または BEGIN で開始されて COMMIT または ROLLBACK で終了するトランザクションに影響します。これは、セクション13.3.3「暗黙的なコミットを発生させるステートメント」に一覧表示されているステートメントの実行から生じる暗黙的なコミットに適用されません。また、XA COMMIT や XA ROLLBACK に対して、あるいは autocommit=1 の場合にも適用されません。

- concurrent_insert

コマンド行形式	--concurrent-insert[=#]
システム変数	concurrent_insert
スコープ	グローバル

動的	はい
型	列挙
デフォルト	AUTO
有効な値	NEVER AUTO ALWAYS 0 1 2

AUTO (デフォルト) の場合、MySQL ではデータファイルの中間に空きブロックがない MyISAM テーブルに対して INSERT および SELECT ステートメントを同時に実行することが許可されます。--skip-new を指定して mysqld を開始した場合、この変数は NEVER に設定されます。

この変数は、次の表に示す値を取ることができます。変数は、名前の値か対応する整数値のいずれかを使用して割り当てることができます。

値	説明
NEVER (または 0)	同時挿入を無効にします
AUTO (または 1)	(デフォルト) 空きブロックがない MyISAM テーブルについての同時挿入を有効にします
ALWAYS (または 2)	空きブロックがあるテーブルであっても、すべての MyISAM テーブルについての同時挿入を有効にします。途中に空きブロックのあるテーブルが別のスレッドによって使用されている場合は、新しい行がテーブルの最後に挿入されます。そうでない場合は、MySQL は正常な書き込みロックを取得し、行を空きブロックに挿入します。

セクション8.10.3「同時挿入」も参照してください。

- connect_timeout

コマンド行形式	--connect-timeout=#
システム変数	connect_timeout
スコープ	グローバル
動的	はい
型	数値
デフォルト	10
最小値	2
最大値	31536000

mysqld サーバーがハンドシェイクエラーを返すまでに接続パケットを待つ秒数。デフォルトは 10 秒です。

「Lost connection to MySQL server at 'XXX', system error: errno」という形式のエラーがクライアントで頻繁に発生する場合、connect_timeout 値を増やすと役立つことがあります。

- core_file

導入	5.6.2
システム変数	core_file
スコープ	グローバル
動的	いいえ
型	ブール
デフォルト	OFF

サーバーがクラッシュした場合にコアファイルを書き込むかどうか。この変数は --core-file オプションによって設定されます。これは MySQL 5.6.2 で追加されました。

- datadir

コマンド行形式	--datadir=path
システム変数	datadir
スコープ	グローバル
動的	いいえ
型	ディレクトリ名

MySQL データディレクトリ。この変数は --datadir オプションで設定できます。

- date_format

この変数は使用されません。これは MySQL 5.6.7 以降で非推奨となり、今後の MySQL リリースで削除されます。

- datetime_format

この変数は使用されません。これは MySQL 5.6.7 以降で非推奨となり、今後の MySQL リリースで削除されます。

- debug

コマンド行形式	--debug[=debug_options]
システム変数	debug
スコープ	グローバル、セッション
動的	はい
型	文字列
デフォルト (Windows)	d:t:i:O,\mysqld.trace
デフォルト (Unix)	d:t:i:o,/tmp/mysqld.trace

この変数は現在のデバッグ設定を指定します。これはデバッグサポートを使用して構築されたサーバーについてのみ使用できます。初期値は、サーバー始動時に指定された --debug オプションのインスタンスの値から取得されます。グローバル値およびセッション値を実行時に設定でき、セッション値についても SUPER 権限が必要です。

+ または - で始まる値を割り当てると、値は現在の値に追加されたり現在の値から削除されたりします。

```
mysql> SET debug = 'T';
mysql> SELECT @@debug;
+---------+
| @@debug |
+---------+
| T       |
+---------+

mysql> SET debug = '+P';
mysql> SELECT @@debug;
+---------+
| @@debug |
+---------+
| P:T     |
+---------+

mysql> SET debug = '-P';
mysql> SELECT @@debug;
+---------+
| @@debug |
+---------+
| T       |
+---------+
```

詳細については、セクション24.4.3「DBUG パッケージ」を参照してください。

- debug_sync

システム変数	debug_sync
スコープ	セッション
動的	はい
型	文字列

この変数は、Debug Sync 機能へのユーザーインタフェースです。Debug Sync を使用するには MySQL が -DENABLE_DEBUG_SYNC=1 オプションで構成されている必要があります (セクション2.9.4「MySQL ソース構成オプション」を参照)。Debug Sync がコンパイルされていない場合、このシステム変数は使用できません。

グローバル変数値は読み取り専用で、この機能が有効かどうかを示します。デフォルトでは、Debug Sync は無効化されており、debug_sync の値は OFF です。サーバーが --debug-sync-timeout=N で開始した場合 (ここで、N は 0 より大きいタイムアウト値)、Debug Sync は有効化され、debug_sync の値は ON - current signal の後にシグナル名が続いたものになります。また、N は個々の同期点についてのデフォルトのタイムアウトになります。

セッション値はすべてのユーザーによって読み取ることができ、グローバル変数と同じ値になります。セッション値は、同期点を制御するための SUPER 権限を持つユーザーによって設定できます。

Debug Sync 機能および同期点の使用方法についての説明は、「MySQL Internals: Test Synchronization」を参照してください。

- default_storage_engine

コマンド行形式	--default-storage-engine=name
システム変数	default_storage_engine
スコープ	グローバル、セッション
動的	はい
型	列挙
デフォルト	InnoDB

デフォルトのストレージエンジン。MySQL 5.6.3 以降では、この変数は永続的なテーブルについてのみストレージエンジンを設定します。TEMPORARY テーブルについてストレージエンジンを設定するには、default_tmp_storage_engine システム変数を設定します。

使用可能かつ有効化できるストレージエンジンを表示するには、SHOW ENGINES ステートメントまたはクエリー INFORMATION_SCHEMA ENGINES テーブルを参照してください。

default_storage_engine は、非推奨となった storage_engine に優先して使用するようにしてください。

サーバー起動時のデフォルトストレージエンジンを無効にした場合、永続テーブルと TEMPORARY テーブルの両方のデフォルトエンジンを別のエンジンに設定しなければならず、そうしないとサーバーは起動しません。

- default_tmp_storage_engine

コマンド行形式	--default-tmp-storage-engine=name
導入	5.6.3
システム変数	default_tmp_storage_engine
スコープ	グローバル、セッション
動的	はい
型	列挙

デフォルト	InnoDB

TEMPORARY テーブルのデフォルトストレージエンジン (CREATE TEMPORARY TABLE で作成されたもの)。永続的なテーブルについてのストレージエンジンを設定するには、default_storage_engine システム変数を設定します。

サーバー起動時のデフォルトストレージエンジンを無効にした場合、永続テーブルと TEMPORARY テーブルの両方のデフォルトエンジンを別のエンジンに設定しなければならず、そうしないとサーバーは起動しません。

default_tmp_storage_engine は MySQL 5.6.3 で追加されました。

- default_week_format

コマンド行形式	--default-week-format=#
システム変数	default_week_format
スコープ	グローバル、セッション
動的	はい
型	数値
デフォルト	0
最小値	0
最大値	7

WEEK() 関数について使用するデフォルトモード値。セクション12.7「日付および時間関数」を参照してください。

- delay_key_write

コマンド行形式	--delay-key-write[=name]
システム変数	delay_key_write
スコープ	グローバル
動的	はい
型	列挙
デフォルト	ON
有効な値	ON OFF ALL

このオプションは、MyISAM テーブルにのみ適用されます。これは、CREATE TABLE ステートメントに使用できる DELAY_KEY_WRITE テーブルオプションの処理に影響する次のいずれかの値を指定できます。

オプション	説明
OFF	DELAY_KEY_WRITE は無視されます。
ON	MySQL は CREATE TABLE ステートメントに指定される DELAY_KEY_WRITE オプションを優先します。これはデフォルト値です。
ALL	新しくオープンしたすべてのテーブルは、DELAY_KEY_WRITE オプションを有効にして作成された場合と同様に処理されます。

テーブルの DELAY_KEY_WRITE を有効にした場合、インデックス更新のたびにそのテーブルのキーバッファーがフラッシュされるのではなく、テーブルが閉じたときだけフラッシュされます。これによりキーの書き込みが非常に高速化されますが、この機能を使用する場合、--myisam-recover-options オプションを指定してサーバーを開始することによって、すべての MyISAM テーブルの自動チェックを追加します (たとえば、--

myisam-recover-options=BACKUP,FORCE)。セクション5.1.3「サーバーコマンドオプション」およびセクション15.2.1「MyISAM 起動オプション」を参照してください。

> **警告**
>
> --external-locking で外部ロックを有効にした場合、キーの遅延書き込みを使用するテーブルについてのインデックス破損に対する保護はありません。

- delayed_insert_limit

コマンド行形式	--delayed-insert-limit=#
非推奨	5.6.7
システム変数	delayed_insert_limit
スコープ	グローバル
動的	はい
型	数値
デフォルト	100
最小値	1
最大値 (64 ビットプラットフォーム)	18446744073709551615
最大値 (32 ビットプラットフォーム)	4294967295

delayed_insert_limit 件の遅延した行を非トランザクションテーブルに挿入したあと、INSERT DELAYED ハンドラスレッドは、保留中の SELECT ステートメントがないかどうかを検査します。その場合、遅延した行の挿入を続行する前に、それらの実行を許可します。

MySQL 5.6.7 以降では、このシステム変数は非推奨となり (DELAYED 挿入が非推奨となったため)、今後のリリースで削除される予定です。

- delayed_insert_timeout

コマンド行形式	--delayed-insert-timeout=#
非推奨	5.6.7
システム変数	delayed_insert_timeout
スコープ	グローバル
動的	はい
型	数値
デフォルト	300

終了する前に、INSERT DELAYED ハンドラスレッドが INSERT ステートメントを待機する秒数。

MySQL 5.6.7 以降では、このシステム変数は非推奨となり (DELAYED 挿入が非推奨となったため)、今後のリリースで削除される予定です。

- delayed_queue_size

コマンド行形式	--delayed-queue-size=#
非推奨	5.6.7
システム変数	delayed_queue_size
スコープ	グローバル
動的	はい
型	数値
デフォルト	1000
最小値	1

最大値 (64 ビットプラットフォーム)	18446744073709551615
最大値 (32 ビットプラットフォーム)	4294967295

これは、非トランザクションテーブルに対する INSERT DELAYED ステートメントを処理するときに、キューに入れる行の数に関するテーブルごとの制限です。キューがいっぱいになると、INSERT DELAYED ステートメントを実行するすべてのクライアントは、キューにふたたび空きができるまで待機します。

MySQL 5.6.7 以降では、このシステム変数は非推奨となり (DELAYED 挿入が非推奨となったため)、今後のリリースで削除される予定です。

- disconnect_on_expired_password

コマンド行形式	--disconnect-on-expired-password=#
導入	5.6.10
システム変数	disconnect_on_expired_password
スコープ	セッション
動的	いいえ
型	ブール
デフォルト	ON

この変数は、期限切れのパスワードを持つクライアントをサーバーが処理する方法を制御します。

- クライアントが期限切れパスワードを処理できるように指定されている場合、disconnect_on_expired_password の値は無関係です。サーバーはクライアントが接続することを許可しますが、クライアントをサンドボックスモードに設定します。
- クライアントが期限切れパスワードを処理できるように指定しない場合、サーバーは disconnect_on_expired_password の値に従ってクライアントを処理します。
  - disconnect_on_expired_password: が有効な場合、サーバーはクライアントを切断します。
  - disconnect_on_expired_password: が無効な場合、サーバーはクライアントの接続を許可しますが、クライアントをサンドボックスモードに設定します。

期限切れパスワードに関するクライアントとサーバーの対話の設定の詳細については、セクション6.3.6「パスワードの期限切れとサンドボックスモード」を参照してください。

この変数は MySQL 5.6.10 で追加されました。

- div_precision_increment

コマンド行形式	--div-precision-increment=#
システム変数	div_precision_increment
スコープ	グローバル、セッション
動的	はい
型	数値
デフォルト	4
最小値	0
最大値	30

この変数は、/ 演算子で実行される除算の結果のスケールを増やす桁数を示します。デフォルト値は 4 です。最小値および最大値は、それぞれ 0 および 30 です。次の例は、デフォルト値を増やした効果を説明したものです。

```
mysql> SELECT 1/7;
+--------+
| 1/7    |
+--------+
| 0.1429 |
```

```
+--------+
mysql> SET div_precision_increment = 12;
mysql> SELECT 1/7;
+---------------+
| 1/7           |
+---------------+
| 0.142857142857 |
+---------------+
```

- engine_condition_pushdown

コマンド行形式	--engine-condition-pushdown
非推奨	はい (removed in 5.6.1); use optimizer_switch instead
システム変数	engine_condition_pushdown
スコープ	グローバル、セッション
動的	はい
型	ブール
デフォルト	ON

この変数は MySQL 5.6.1 で削除されました。optimizer_switch 変数の engine_condition_pushdown フラグを代わりに使用します。セクション8.8.5.2「切り替え可能な最適化の制御」を参照してください。

- end_markers_in_json

導入	5.6.5
システム変数	end_markers_in_json
スコープ	グローバル、セッション
動的	はい
型	ブール
デフォルト	OFF

オプティマイザ JSON 出力がエンドマーカーを追加するかどうか。

- eq_range_index_dive_limit

導入	5.6.5
システム変数	eq_range_index_dive_limit
スコープ	グローバル、セッション
動的	はい
型	数値
デフォルト	10
最小値	0
最大値	4294967295

この変数は、オプティマイザが限定する行数を推定するときに、インデックスダイブの使用からインデックス統計の使用に切り換える場合の等価比較条件内の等価範囲の数を指定します。これは次に示す同等のいずれかの形式を持つ式の評価に適用され、このときオプティマイザは一意でないインデックスを使用して col_name 値を参照します。

```
col_name IN(val1, ..., valN)
col_name = val1 OR ... OR col_name = valN
```

どちらの場合も、式に N 個の等価範囲が含まれています。オプティマイザはインデックスダイブまたはインデックス統計を使用すると行の推定を実行できます。eq_range_index_dive_limit が 0 より大きい場合、eq_range_index_dive_limit 以上の等価範囲があれば、オプティマイザはインデックスダイブの代わりに既存のインデックス統計を使用します。したがって、N 個までの等価範囲に対してインデックスダイブの使用を可能にするには、eq_range_index_dive_limit を N + 1 に設定します。インデックス統計の使用を無効にして、N に関係なくインデックスダイブを常に使用するには、eq_range_index_dive_limit を 0 に設定します。

詳細については、複数値比較の等価範囲の最適化を参照してください。

この変数は MySQL 5.6.5 で追加されました。5.6.5 より前では、オプティマイザはすべての場合にインデックスダイブを使用して行の推定を実行します。

最適な推定を行うためにテーブルインデックス統計を更新するには、ANALYZE TABLE を使用します。

- error_count

  メッセージを生成した最後のステートメントから発生したエラーの数。この変数は読み取り専用です。セクション13.7.5.18「SHOW ERRORS 構文」を参照してください。

- event_scheduler

コマンド行形式	--event-scheduler[=value]
システム変数	event_scheduler
スコープ	グローバル
動的	はい
型	列挙
デフォルト	OFF
有効な値	ON OFF DISABLED

この変数はイベントスケジューラのステータスを示し、可能な値は ON、OFF、および DISABLED で、デフォルトは OFF です。この変数と、イベントスケジューラの操作におけるその効果は、イベントの章の概要セクションでさらに詳しく説明されています。

- expire_logs_days

コマンド行形式	--expire-logs-days=#
システム変数	expire_logs_days
スコープ	グローバル
動的	はい
型	数値
デフォルト	0
最小値	0
最大値	99

バイナリログファイルの自動削除のための日数。デフォルトは 0 で 「自動削除しない」ことを意味します。削除は起動時およびバイナリログがフラッシュされるときに発生する可能性があります。ログのフラッシュは、セクション5.2「MySQL Server ログ」に記載されているように発生します。

バイナリログファイルを手動で削除するには、PURGE BINARY LOGS ステートメントを使用します。セクション13.4.1.1「PURGE BINARY LOGS 構文」を参照してください。

- explicit_defaults_for_timestamp

コマンド行形式	--explicit-defaults-for-timestamp=#
導入	5.6.6
システム変数	explicit_defaults_for_timestamp
スコープ	セッション
動的	いいえ
型	ブール

デフォルト	FALSE

MySQL では、TIMESTAMP データ型は非標準的な方式であるという点でほかのデータ型と異なります。

- NULL 属性で明示的に宣言されない TIMESTAMP カラムには、NOT NULL 属性が割り当てられます。(ほかのデータ型のカラムは、NOT NULL として明示的に宣言されない場合、NULL 値が許可されます。)そのようなカラムを NULL に設定すると、カラムは現在のタイムスタンプに設定されます。
- テーブル内の最初の TIMESTAMP カラムは、NULL 属性や明示的な DEFAULT または ON UPDATE 句で宣言されない場合、DEFAULT CURRENT_TIMESTAMP および ON UPDATE CURRENT_TIMESTAMP 属性が自動的に割り当てられます。
- 最初のカラムに続く TIMESTAMP カラムは、NULL 属性または明示的な DEFAULT 句で宣言されない場合、DEFAULT '0000-00-00 00:00:00' (「ゼロ」タイムスタンプ) が自動的に割り当てられます。そのようなカラムに対して明示的な値を指定しない挿入された行については、カラムに '0000-00-00 00:00:00' が自動的に割り当てられて、警告は発生しません。

これらの非標準の動作は TIMESTAMP についてはデフォルトのままですが、MySQL 5.6.6 以降では非推奨となり、起動時に次の警告が表示されます。

```
[Warning] TIMESTAMP with implicit DEFAULT value is deprecated.
Please use --explicit_defaults_for_timestamp server option (see
documentation for more details).
```

警告が示すように、非標準の動作をオフにするには、新しい explicit_defaults_for_timestamp システム変数を起動時に有効にします。この変数を有効にすると、サーバーは TIMESTAMP を、代わりに次のように処理します。

- 明示的に NOT NULL として宣言されない TIMESTAMP カラムでは、NULL 値が許可されます。そのようなカラムを NULL に設定することで、カラムは現在のタイムスタンプではなく NULL に設定されます。
- TIMESTAMP カラムに DEFAULT CURRENT_TIMESTAMP または ON UPDATE CURRENT_TIMESTAMP 属性が自動的に割り当てられません。これらの属性は、明示的に指定する必要があります。
- NOT NULL として宣言され、明示的な DEFAULT 句を持たない TIMESTAMP カラムは、デフォルト値を持たないものとして処理されます。そのようなカラムについて明示的な値を指定しない挿入された行の場合、結果は SQL モードによって異なります。厳密 SQL モードが有効である場合、エラーが発生します。厳密 SQL モードが有効でない場合、カラムには暗黙的なデフォルトの '0000-00-00 00:00:00' が割り当てられ、警告が発生します。これは、MySQL が DATETIME などのほかの時間型を処理する方法に類似しています。

> **注記**
>
> explicit_defaults_for_timestamp は、それ自体が非推奨です。これは、その唯一の目的が、将来の MySQL リリースで削除される現在非推奨となった TIMESTAMP 動作に対する制御を許可するためです。その削除が行われると、explicit_defaults_for_timestamp は目的を失うため、同様に削除されます。

この変数は MySQL 5.6.6 で追加されました。

- external_user

システム変数	external_user
スコープ	セッション
動的	いいえ
型	文字列

クライアントを認証するために使用されるプラグインによって設定された、認証プロセス中に使用される外部ユーザー名。ネイティブ (組み込み型) の MySQL 認証や、プラグインで値が設定されない場合、この変数は NULL です。セクション6.3.9「プロキシユーザー」を参照してください。

- flush

コマンド行形式	--flush
システム変数	flush
スコープ	グローバル

動的	はい
型	ブール
デフォルト	OFF

ON の場合、各 SQL ステートメントのあとでサーバーはすべての変更をデスクにフラッシュ (同期) します。通常、MySQL では各 SQL ステートメントの終了後にのみすべての変更内容をディスクに書き込み、ディスクへの同期はオペレーティングシステムが処理します。セクションB.5.4.2「MySQL が繰り返しクラッシュする場合の対処方法」を参照してください。--flush オプションで mysqld を起動した場合、この変数は ON に設定されます。

- flush_time

コマンド行形式	--flush-time=#
システム変数	flush_time
スコープ	グローバル
動的	はい
型	数値
デフォルト (Windows, ≥ 5.6.6)	0
デフォルト (Windows, ≤ 5.6.5)	1800
デフォルト	0
最小値	0

これがゼロ以外の値に設定されると、すべてのテーブルは flush_time 秒ごとに閉じられて、リソースが解放され、フラッシュされていないデータがディスクへ同期されます。このオプションは、リソースが非常に限定されたシステムでのみ使用することを推奨します。デフォルトは 0 ですが、MySQL 5.6.6 以前では、Windows のデフォルトは 1800 です。

- foreign_key_checks

1 (デフォルト) に設定すると、InnoDB テーブルについての外部キー制約が検査されます。0 に設定すると、そのような制約は無視されます。MySQL Cluster NDB 7.3.2 以降では、NDB テーブルでこの変数を設定することは、それを InnoDB テーブルで実行することと同じ効果が得られます。以前では、この設定は無視され、そのようなすべての検査が強制されていました (バグ #14095855)。一般的に、通常の操作中はこの設定を有効にしたままにすることで、参照整合性を強制します。外部キーの検査を無効化することは、テーブルの親子関係によって要求される順序と異なる順序でこれらのテーブルをリロードする場合に役立つことがあります。セクション14.6.6「InnoDB と FOREIGN KEY 制約」を参照してください。

foreign_key_checks を 0 に設定すると、データ定義ステートメントにも影響します。DROP SCHEMA は、スキーマの外部のテーブルによって参照されている外部キーを持つテーブルをスキーマが含む場合であってもスキーマをドロップし、DROP TABLE は、別のテーブルによって参照されている外部キーを持つテーブルをドロップします。

> **注記**
>
> foreign_key_checks を 1 に設定すると、既存のテーブルデータのスキャンがトリガーされません。したがって、foreign_key_checks = 0 のときにテーブルに追加された行は、一貫性が検証されません。

- ft_boolean_syntax

コマンド行形式	--ft-boolean-syntax=name
システム変数	ft_boolean_syntax
スコープ	グローバル
動的	はい
型	文字列

デフォルト	+ -><()~*:""&|

IN BOOLEAN MODE を使用して実行されるブール全文検索によってサポートされる演算子のリスト。セクション12.9.2「ブール全文検索」を参照してください。

デフォルトの変数値は '+ -><()~*:""&|' です。値を変更するルールは次のようになります。

- 演算子の機能は、文字列内の位置によって決定されます。
- 置換する値は 14 文字である必要があります。
- 各文字は、英数字以外の ASCII 文字である必要があります。
- 1 番目または 2 番目の文字がスペースである必要があります。
- 位置 11 および 12 にある句を引用する演算子を除き、重複は許可されません。これらの 2 つの文字は同じである必要はありませんが、同じであってもよいのはこれら 2 つだけです。
- 位置 10、13、および 14 (デフォルトで 「:」、「&」、および 「\|」 に設定) は将来の拡張用に予約されています。

- ft_max_word_len

コマンド行形式	--ft-max-word-len=#
システム変数	ft_max_word_len
スコープ	グローバル
動的	いいえ
型	数値
最小値	10

MyISAM FULLTEXT インデックスに含めることができる最大の単語の長さ。

> **注記**
>
> この変数を変更したあと、MyISAM テーブルの FULLTEXT インデックスを再構築する必要があります。REPAIR TABLE tbl_name QUICK を使用します。

- ft_min_word_len

コマンド行形式	--ft-min-word-len=#
システム変数	ft_min_word_len
スコープ	グローバル
動的	いいえ
型	数値
デフォルト	4
最小値	1

MyISAM FULLTEXT インデックスに含めることができる最小の単語の長さ。

> **注記**
>
> この変数を変更したあと、MyISAM テーブルの FULLTEXT インデックスを再構築する必要があります。REPAIR TABLE tbl_name QUICK を使用します。

- ft_query_expansion_limit

コマンド行形式	--ft-query-expansion-limit=#
システム変数	ft_query_expansion_limit
スコープ	グローバル
動的	いいえ

型	数値
デフォルト	20
最小値	0
最大値	1000

WITH QUERY EXPANSION を使用して実行する全文検索で使用する最上位の一致の数。

- ft_stopword_file

コマンド行形式	--ft-stopword-file=file_name
システム変数	ft_stopword_file
スコープ	グローバル
動的	いいえ
型	ファイル名

MyISAM テーブルの全文検索について、ストップワードのリストの読み取り元ファイル。サーバーは、別のディレクトリを指定する絶対パス名が指定されないかぎり、データディレクトリ内のファイルを検索します。ファイル内のすべての単語が使用され、コメントは受け付けされません。デフォルトでは、ストップワードの組み込みリストが使用されます (storage/myisam/ft_static.c ファイルに定義されています)。この変数を空の文字列 ('') に設定すると、ストップワードフィルタリングが無効になります。セクション12.9.4「全文ストップワード」も参照してください。

> **注記**
>
> この変数またはストップワードファイルの内容を変更したあと、MyISAM テーブルの FULLTEXT インデックスを再構築する必要があります。REPAIR TABLE tbl_name QUICK を使用します。

- general_log

コマンド行形式	--general-log
システム変数	general_log
スコープ	グローバル
動的	はい
型	ブール
デフォルト	OFF

一般クエリーログを有効にするかどうか。値が 0 (または OFF) の場合はログを無効にし、1 (または ON) の場合はログを有効にします。デフォルト値は --general_log オプションが指定されているかどうかによって異なります。ログ出力先は log_output システム変数によって制御され、この値を NONE にした場合はログが有効になっていてもログエントリは書き込まれません。

- general_log_file

コマンド行形式	--general-log-file=file_name
システム変数	general_log_file
スコープ	グローバル
動的	はい
型	ファイル名
デフォルト	host_name.log

一般クエリーログファイルの名前。デフォルト値は host_name.log ですが、初期値は --general_log_file オプションを使用すると変更できます。

- group_concat_max_len

コマンド行形式	--group-concat-max-len=#

システム変数	group_concat_max_len
スコープ	グローバル、セッション
動的	はい
型	数値
デフォルト	1024
最小値	4
最大値 (64 ビットプラットフォーム)	18446744073709551615
最大値 (32 ビットプラットフォーム)	4294967295

GROUP_CONCAT() 関数について許可されるバイト単位の最大の結果の長さ。デフォルトは 1024 です。

- have_compress

  zlib 圧縮ライブラリがサーバーで利用できる場合は YES、そうでない場合は NO。利用できない場合、COMPRESS() および UNCOMPRESS() 関数は使用できません。

- have_crypt

  crypt() システムコールがサーバーで利用できる場合は YES、そうでない場合は NO。利用できない場合、ENCRYPT() 関数は使用できません。

- have_csv

  mysqld が CSV テーブルをサポートする場合は YES、そうでない場合は NO。

  この変数は MySQL 5.6.1 で削除されました。代わりに SHOW ENGINES を使用してください。

- have_dynamic_loading

  mysqld がプラグインの動的ロードをサポートする場合は YES、そうでない場合は NO。

- have_geometry

  サーバーが空間データ型をサポートする場合は YES、そうでない場合は NO。

- have_innodb

  mysqld が InnoDB テーブルをサポートする場合は YES。--skip-innodb が使用される場合は DISABLED。

  この変数は MySQL 5.6.1 で削除されました。代わりに SHOW ENGINES を使用してください。

- have_openssl

  この変数は have_ssl のエイリアスです。

- have_partitioning

  mysqld がパーティショニングをサポートする場合は YES。

  この変数は MySQL 5.6.1 で削除されました。代わりに SHOW PLUGINS を使用してください。詳細については、第19章「パーティション化」を参照してください。

- have_profiling

  ステートメントプロファイリング機能が存在する場合は YES、そうでない場合は NO。存在する場合、この機能を有効または無効にするかが profiling システム変数によって制御されます。セクション13.7.5.32「SHOW PROFILES 構文」を参照してください。

  この変数は MySQL 5.6.8 で非推奨となり、将来の MySQL リリースで削除されます。

- have_query_cache

  mysqld がクエリーキャッシュをサポートする場合は YES、そうでない場合は NO。

- have_rtree_keys

RTREE インデックスを利用できる場合は YES、そうでない場合は NO。(これらは MyISAM テーブル内の空間インデックスで使用されます。)

- have_ssl

  mysqld が SSL 接続をサポートする場合は YES、そうでない場合は NO。DISABLED の場合、サーバーは SSL サポート付きでコンパイルされているが、適切な --ssl-xxx オプションを指定して開始されなかったことを示します。詳細については、セクション6.3.10.2「SSL を使用するための MySQL の構成」を参照してください。

- have_symlink

  シンボリックリンクサポートを有効化している場合は YES、そうでない場合はNO。これは Unix では DATA DIRECTORY および INDEX DIRECTORY のテーブルオプションをサポートし、Windows ではデータディレクトリの symlink をサポートするために必要です。--skip-symbolic-links オプションを指定してサーバーが開始された場合、この値は DISABLED です。

- host_cache_size

導入	5.6.5
システム変数	host_cache_size
スコープ	グローバル
動的	はい
型	数値
デフォルト (≥ 5.6.8)	-1 (autosized)
デフォルト (≤ 5.6.7)	128
最小値	0
最大値	65536

  内部ホストキャッシュのサイズ (セクション8.11.5.2「DNS ルックアップの最適化とホストキャッシュ」を参照してください)。サイズを 0 に設定すると、ホストキャッシュが無効になります。実行時にキャッシュサイズを変更すると、暗黙的に FLUSH HOSTS 操作によってホストキャッシュがクリアされ、host_cache テーブルが切り捨てられます。

  デフォルト値は 128 で、500 までの max_connections の値については 1 が加算され、max_connections の値が 500 を超えて 20 増えるごとに 1 が加算され、上限は 2000 までに制限されます。MySQL 5.6.8 より前では、デフォルトは 128 です。

  --skip-host-cache の使用は host_cache_size システム変数を 0 に設定することに似ていますが、host_cache_size の方が柔軟性が高く、これはサーバー起動時だけでなく実行時にもホストキャッシュのサイズを変更したり有効化または無効化したりするために使用できるためです。

  --skip-host-cache を指定してサーバーを開始しても、host_cache_size の値の変更を妨げるわけではありませんが、この変更は効果がなく、host_cache_size を 0 より大きく設定してもキャッシュはふたたび有効化されません。

  この変数は MySQL 5.6.5 で追加されました。

- hostname

システム変数	hostname
スコープ	グローバル
動的	いいえ
型	文字列

  サーバーは起動時に、この変数をサーバーホスト名に設定します。

- identity

  この変数は last_insert_id 変数のシノニムです。これはほかのデータベースシステムとの互換性のために存在します。この値は SELECT @@identity で読み取ることができ、SET identity で設定できます。

- ignore_db_dirs

導入	5.6.3
システム変数	ignore_db_dirs
スコープ	グローバル
動的	いいえ
型	文字列

データディレクトリ内でデータベースディレクトリとして考慮されない、カンマで区切られた名前のリスト。この値は、サーバー起動時に指定されるすべての --ignore-db-dir の実例によって設定されます。

この変数は MySQL 5.6.3 で追加されました。

- init_connect

コマンド行形式	--init-connect=name
システム変数	init_connect
スコープ	グローバル
動的	はい
型	文字列

接続する各クライアントに対してサーバーによって実行される文字列。文字列は 1 つ以上の SQL ステートメントで構成され、セミコロン文字で区切られます。たとえば、各クライアントセッションは、デフォルトでは自動コミットモードが有効な状態で開始します。古いサーバー (MySQL 5.5.8 より前) では、自動コミットをデフォルトで無効にするよう指定するためのグローバルな autocommit システム変数が存在しませんが、同じ効果を得るための回避策として init_connect を使用できます。

```
SET GLOBAL init_connect='SET autocommit=0';
```

init_connect 変数はコマンド行またはオプションファイルにも設定できます。ここに示されたような変数を、オプションファイルを使用して設定するには、次の行を含めます。

```
[mysqld]
init_connect='SET autocommit=0'
```

init_connect の内容は、SUPER 権限を持つユーザーに対して実行されません。これを行うのは、init_connect の値が誤っていても、すべてのクライアントの接続を妨げないようにするためです。たとえば、値に含まれているステートメントが構文エラーを含むため、クライアント接続が失敗することがあります。SUPER 権限を持つユーザーに対して init_connect を実行しないことで、これらのユーザーは接続を開き、init_connect 値を修正できます。

- init_file

コマンド行形式	--init-file=file_name
システム変数	init_file
スコープ	グローバル
動的	いいえ
型	ファイル名

サーバーを起動したときに、--init-file オプションで指定するファイルの名前。これは、サーバーが起動したときにサーバーで実行する SQL ステートメントを含むファイルにしてください。各ステートメントは単一の行にして、コメントを含めなでください。各ステートメントの末尾に ;、\g、\G などのステートメントターミネータを指定しないようにしてください。

- innodb_xxx

InnoDB システム変数は、セクション14.12「InnoDB の起動オプションおよびシステム変数」にリストされています。これらの変数は、InnoDB テーブルのストレージ、メモリー使用、および I/O パターンの多くの側面を制御し、InnoDB がデフォルトストレージエンジンになったため、特に重要です。

- insert_id

  AUTO_INCREMENT 値を挿入するときに、後に続く INSERT または ALTER TABLE ステートメントによって使用される値。これは主にバイナリログと一緒に使用されます。

- interactive_timeout

コマンド行形式	--interactive-timeout=#
システム変数	interactive_timeout
スコープ	グローバル、セッション
動的	はい
型	数値
デフォルト	28800
最小値	1

  サーバーが対話型の接続で、対話型の接続を閉じる前にアクティビティーを待機する秒数。対話型クライアントは、mysql_real_connect() で CLIENT_INTERACTIVE オプションを使用するクライアントと定義されます。wait_timeoutも参照してください。

- join_buffer_size

コマンド行形式	--join-buffer-size=#
システム変数	join_buffer_size
スコープ	グローバル、セッション
動的	はい
型	数値
デフォルト (その他, 64 ビットプラットフォーム, ≥ 5.6.6)	262144
デフォルト (その他, 64 ビットプラットフォーム, ≤ 5.6.5)	131072
デフォルト (その他, 32 ビットプラットフォーム, ≥ 5.6.6)	262144
デフォルト (その他, 32 ビットプラットフォーム, ≤ 5.6.5)	131072
デフォルト (Windows, ≥ 5.6.6)	262144
デフォルト (Windows, ≤ 5.6.5)	131072
最小値	128
最大値 (その他, 64 ビットプラットフォーム)	18446744073709547520
最大値 (その他, 32 ビットプラットフォーム)	4294967295
最大値 (Windows)	4294967295

  単純インデックススキャン、範囲インデックススキャン、およびインデックスを使用しないため完全テーブルスキャンを実行する結合について、使用されるバッファーの最小サイズ。通常の場合、高速な結合を得るための最適な方法は、インデックスを追加することです。インデックスを追加できない場合、より高速な完全結合を得るために、join_buffer_size の値を大きくします。2 つのテーブル間の完全結合 1 つに対して 1 つの結合バッファーが割り当てられます。インデックスが使用されない複数テーブル間の複雑な結合については、複数の結合バッファーが必要になることもあります。

  バッチキーアクセス (BKA) を使用しないかぎり、一致する各行を保持するために必要な量よりも大きいバッファーを設定することの利点はなく、すべての結合は少なくとも最小のサイズを割り当てるため、この変数をグローバルに大きい値に設定する場合は注意してください。グローバル設定を小さくしておき、大規模な結合を実行するセッションでのみ大きい設定に変更することを推奨します。メモリーを使用するほとんどのク工

リーによって必要なサイズよりもグローバルサイズを大きくすると、メモリー割り当て時間が原因でパフォーマンスが著しく低下することがあります。

BKA が使用される場合、join_buffer_size の値によって、ストレージエンジンへの個々のリクエストでのキーのバッチの大きさが定義されます。バッファーが大きいほど、結合操作の右側テーブルへの順次アクセスが増え、パフォーマンスを著しく向上させることができます。

デフォルトは MySQL 5.6.6 以降では 256K バイトで、それより前は 128K バイトです。join_buffer_size で許可される最大の設定値は 4G バイト −1 です。64 ビットプラットフォームの場合は大きい値が許可されます (64 ビットの Windows の場合は例外で、大きい値は 4G バイト − 1 に切り捨てられて警告が出ます)。

結合バッファリングについての追加情報は、セクション8.2.1.10「Nested Loop 結合アルゴリズム」を参照してください。バッチキーアクセスについては、セクション8.2.1.14「Block Nested Loop 結合と Batched Key Access 結合」を参照してください。

- keep_files_on_create

コマンド行形式	--keep-files-on-create=#
システム変数	keep_files_on_create
スコープ	グローバル、セッション
動的	はい
型	ブール
デフォルト	OFF

MyISAM テーブルが DATA DIRECTORY オプションなしで作成される場合、.MYD ファイルがデータベースディレクトリ内に作成されます。デフォルトでは、MyISAM が既存の .MYD ファイルを検出した場合、そのファイルを上書きします。INDEX DIRECTORY オプションを指定せずに作成されたテーブルについて、.MYI ファイルに同じことが当てはまります。この動作を抑制するには、keep_files_on_create 変数を ON (1) に設定します。この場合、MyISAM は既存のファイルを上書きせず、代わりにエラーが返されます。デフォルト値は OFF (0) です。

MyISAM テーブルが DATA DIRECTORY または INDEX DIRECTORY オプションを使用して作成され、既存の .MYD または .MYI ファイルが見つかった場合、MyISAM は常にエラーを返します。指定されたディレクトリ内のファイルは上書きされません。

- key_buffer_size

コマンド行形式	--key-buffer-size=#
システム変数	key_buffer_size
スコープ	グローバル
動的	はい
型	数値
デフォルト	8388608
最小値	8
最大値 (64 ビットプラットフォーム)	OS_PER_PROCESS_LIMIT
最大値 (32 ビットプラットフォーム)	4294967295

MyISAM テーブルのインデックスブロックはバッファリングされ、すべてのスレッドで共有されます。key_buffer_size は、インデックスブロックに使用されるバッファーのサイズです。キーバッファーはキーキャッシュとしても知られています。

32 ビットプラットフォームでは、key_buffer_size に許可される最大の設定値は 4G バイト − 1 です。64 ビットプラットフォームでは、さらに大きい値が許可されます。実質的な最大サイズは、使用可能な物理 RAM や、オペレーティングシステムまたはハードウェアプラットフォームによって課されるプロセスごとの RAM 制限によって、もっと小さいことがあります。この変数の値は、リクエストされるメモリーの量を示します。サー

バーは内部的に、この量までのできるだけ多くのメモリーを割り当てますが、実際の割り当てがもっと少なくなることもあります。

値を増やすことで、すべての読み取りおよび複数の書き込みのインデックス処理を改善できます。システムの主な機能が MyISAM ストレージエンジンを使用して MySQL を実行する場合、マシンの合計メモリーの 25% がこの変数の許容可能な値です。ただし、値を大きくし過ぎると (マシンの合計メモリーの 50% 超)、システムのページングが始まってきわめて低速になることがあります。これは MySQL がデータ読み取りのためのファイルシステムキャッシュの実行をオペレーティングシステムに依存しているためで、ファイルシステムキャッシュのためにいくらかの空きを残しておく必要があります。また、MyISAM に追加して使用するほかのストレージエンジンのメモリー要件も考慮します。

多くの行の同時書き込みなどスピードを高めるには、LOCK TABLES を使用します。セクション 8.2.2.1「INSERT ステートメントの速度」を参照してください。

キーバッファーのパフォーマンスを確認するために、SHOW STATUS ステートメントを発行し、Key_read_requests、Key_reads、Key_write_requests、および Key_writes のステータス変数を調べることができます。(セクション13.7.5「SHOW 構文」を参照してください。)Key_reads/Key_read_requests の比率は通常は 0.01 より小さくなります。操作がほとんど更新と削除だけの場合は Key_writes/Key_write_requests の比率は 1 に近くなりますが、同時に多くの行に影響を与える更新を行う場合や、DELAY_KEY_WRITE テーブルオプションを使用する場合はもっと小さくなることがあります。

使用中のキーバッファーの部分は、key_buffer_size に加えて、Key_blocks_unused ステータス変数と、key_cache_block_size システム変数から利用可能なバッファーブロックサイズを使用して決定できます。

```
1 - ((Key_blocks_unused * key_cache_block_size) / key_buffer_size)
```

キーバッファー内の一部のスペースは、管理構造の内部で割り当てられるため、この値は概算です。これらの構造についてのオーバーヘッドの量に影響する要素には、ブロックサイズおよびポインタサイズがあります。ブロックサイズが増加すると、オーバーヘッドで失われるキーバッファーのパーセントが減少する傾向にあります。ブロックが大きくなると、読み取り操作の数が少なくなりますが (読み取りあたりで取得されるキーが増えるため)、検査されないキーの読み取りが逆に増加します (ブロック内の一部のキーがクエリーに関連していない場合)。

MyISAM の複数キーキャッシュを作成できます。グループとしてではなく個別の各キャッシュに対して 4G バイトのサイズ制限が適用されます。セクション8.9.2「MyISAM キーキャッシュ」を参照してください。

- key_cache_age_threshold

コマンド行形式	--key-cache-age-threshold=#
システム変数	key_cache_age_threshold
スコープ	グローバル
動的	はい
型	数値
デフォルト	300
最小値	100
最大値 (64 ビットプラットフォーム)	18446744073709551615
最大値 (32 ビットプラットフォーム)	4294967295

この値は、キーキャッシュのホットサブリストからウォームサブリストへのバッファーの格下げを制御します。値が低いと格下げが早く行われます。最小値は 100 です。デフォルト値は 300 です。セクション 8.9.2「MyISAM キーキャッシュ」を参照してください。

- key_cache_block_size

コマンド行形式	--key-cache-block-size=#
システム変数	key_cache_block_size
スコープ	グローバル
動的	はい
型	数値

デフォルト	1024
最小値	512
最大値	16384

キーキャッシュ内のバイト単位のブロックのサイズ。デフォルト値は 1024 です。セクション8.9.2「MyISAM キーキャッシュ」を参照してください。

- key_cache_division_limit

コマンド行形式	--key-cache-division-limit=#
システム変数	key_cache_division_limit
スコープ	グローバル
動的	はい
型	数値
デフォルト	100
最小値	1
最大値	100

キーキャッシュバッファーリストのホットサブリストとウォームサブリストの間の分割点。値は、ウォームサブリスト用に使用するバッファーリストのパーセントです。許可される値の範囲は 1 から 100 です。デフォルト値は 100 です。セクション8.9.2「MyISAM キーキャッシュ」を参照してください。

- large_files_support

システム変数	large_files_support
スコープ	グローバル
動的	いいえ

大きなファイルをサポートするオプションで mysqld をコンパイルしているかどうか。

- large_pages

コマンド行形式	--large-pages
システム変数	large_pages
スコープ	グローバル
動的	いいえ
プラットフォーム固有	Linux
型	ブール
デフォルト	FALSE

大規模ページサポートが (--large-pages オプションで) 有効になっているかどうか。セクション8.11.4.2「ラージページのサポートの有効化」を参照してください。

- large_page_size

システム変数	large_page_size
スコープ	グローバル
動的	いいえ
型	数値
デフォルト	0

大規模ページサポートが有効化されている場合、これはメモリーページのサイズを示します。現在、大規模メモリーページは Linux でのみサポートされており、ほかのプラットフォームではこの変数の値は常に 0 です。セクション8.11.4.2「ラージページのサポートの有効化」を参照してください。

- last_insert_id

  LAST_INSERT_ID() から返される値。これは、テーブルを更新するステートメント内で LAST_INSERT_ID() を使用するときにバイナリログ内に格納されます。この変数を設定しても、mysql_insert_id() C API 関数によって返される値は更新されません。

- lc_messages

コマンド行形式	--lc-messages=name
システム変数	lc_messages
スコープ	グローバル、セッション
動的	はい
型	文字列
デフォルト	en_US

  エラーメッセージに使用するロケール。デフォルトは en_US です。サーバーは引数を言語名に変換し、これを lc_messages_dir の値と組み合わせてエラーメッセージファイルの場所を生成します。セクション10.2「エラーメッセージ言語の設定」を参照してください。

- lc_messages_dir

コマンド行形式	--lc-messages-dir=dir_name
システム変数	lc_messages_dir
スコープ	グローバル
動的	いいえ
型	ディレクトリ名

  エラーメッセージが配置されているディレクトリ。サーバーはこの値を lc_messages の値と一緒に使用して、エラーメッセージファイルの場所を生成します。セクション10.2「エラーメッセージ言語の設定」を参照してください。

- lc_time_names

システム変数	lc_time_names
スコープ	グローバル、セッション
動的	はい
型	文字列

  この変数は、日および月の名前と略語を表示するために使用する言語を制御するロケールを指定します。この変数は DATE_FORMAT()、DAYNAME()、および MONTHNAME() 関数の出力に影響を与えます。ロケール名は、'ja_JP' や 'pt_BR' などの POSIX 規格の値です。システムのロケール設定に関係なく、デフォルト値は 'en_US' です。詳細については、セクション10.7「MySQL Server のロケールサポート」を参照してください。

- license

システム変数	license
スコープ	グローバル
動的	いいえ
型	文字列
デフォルト	GPL

  サーバーが持つライセンスのタイプ。

- local_infile

システム変数	local_infile
スコープ	グローバル
動的	はい

型	ブール

LOAD DATA INFILE ステートメントで LOCAL がサポートされているかどうか。この変数が無効な場合、クライアントは LOAD DATA ステートメントで LOCAL を使用できません。セクション6.1.6「LOAD DATA LOCAL のセキュリティーの問題」を参照してください。

- lock_wait_timeout

コマンド行形式	--lock-wait-timeout=#
システム変数	lock_wait_timeout
スコープ	グローバル、セッション
動的	はい
型	数値
デフォルト	31536000
最小値	1
最大値	31536000

この変数は、メタデータロックを取得するための試行のタイムアウトを秒単位で指定します。許可される値の範囲は 1 から 31536000 (1 年) です。デフォルトは 31536000 です。

このタイムアウトは、メタデータロックを使用するすべてのステートメントに適用されます。これらには、テーブル、ビュー、ストアドプロシージャー、ストアドファンクションの DML 操作および DDL 操作のほかに、LOCK TABLES、FLUSH TABLES WITH READ LOCK、および HANDLER ステートメントが含まれます。

このタイムアウトは、GRANT または REVOKE ステートメントやテーブルロギングステートメントによって変更される付与テーブルなど、mysql データベース内のシステムテーブルへの暗黙的なアクセスには適用されません。タイムアウトは、SELECT や UPDATE などによって直接アクセスされるシステムテーブルに適用されます。

タイムアウト値は、メタデータロック試行ごとに別々に適用されます。ある特定のステートメントが複数のロックを必要とする場合もあるため、タイムアウトエラーを報告する前に、ステートメントが lock_wait_timeout 値よりも長くブロックする可能性もあります。ロックタイムアウトが発生すると、ER_LOCK_WAIT_TIMEOUT が報告されます。

lock_wait_timeout は、常に 1 年のタイムアウトで実行される遅延挿入に適用されません。これは不要なタイムアウトを回避するために行われます。遅延挿入を発行するセッションは、遅延挿入タイムアウトの通知を受け取らないためです。

- locked_in_memory

システム変数	locked_in_memory
スコープ	グローバル
動的	いいえ

mysqld が --memlock によってメモリー内でロックされたかどうか。

- log

コマンド行形式	--log[=file_name]
非推奨	はい (removed in 5.6.1); use general-log instead
システム変数	log
スコープ	グローバル
動的	はい
型	ファイル名

この変数は MySQL 5.6.1 で削除されました。代わりに general_log を使用してください。

- log_bin_trust_function_creators

コマンド行形式	--log-bin-trust-function-creators

システム変数	log_bin_trust_function_creators
スコープ	グローバル
動的	はい
型	ブール
デフォルト	FALSE

この変数は、バイナリロギングが有効な場合に適用されます。これは、安全ではないイベントがバイナリログに書き込まれる原因となるストアドファンクションを、ストアドファンクションの生成者が作成しないということを信頼できるかどうかを制御します。0 (デフォルト) に設定した場合、ユーザーは CREATE ROUTINE または ALTER ROUTINE 権限に加えて SUPER 権限を持たないかぎり、ストアドファンクションを作成または変更することが許可されません。0 に設定することで、関数を DETERMINISTIC 特性で、あるいは READS SQL DATA または NO SQL 特性で宣言する必要があるという制約も強制されます。変数が 1 に設定された場合、MySQL はストアドファンクション作成にこれらの制約を強制しません。この変数はトリガー作成にも適用されます。セクション20.7「ストアドプログラムのバイナリロギング」を参照してください。

- log_error

コマンド行形式	--log-error[=file_name]
システム変数	log_error
スコープ	グローバル
動的	いいえ
型	ファイル名

エラーログの場所か、サーバーが標準エラー出力にエラーメッセージを書き込む場合は空白。セクション5.2.2「エラーログ」 を参照してください。

- log_output

コマンド行形式	--log-output=name
システム変数	log_output
スコープ	グローバル
動的	はい
型	セット
デフォルト	FILE
有効な値	TABLE FILE NONE

一般クエリーログおよびスロークエリーログの出力先。値は TABLE (テーブルへのログ)、FILE (ファイルへのログ)、NONE (テーブルまたはファイルをログしない) という 1 つ以上の単語のカンマ区切りリストにできます。デフォルト値は FILE です。NONE がある場合は、ほかの指定子よりも優先されます。値が NONE の場合、ログが有効であってもログエントリは書き込まれません。ログが有効でない場合、log_output の値が NONE でなくてもロギングは実行されません。詳細については、セクション5.2.1「一般クエリーログおよびスロークエリーログの出力先の選択」を参照してください。

- log_queries_not_using_indexes

コマンド行形式	--log-queries-not-using-indexes
システム変数	log_queries_not_using_indexes
スコープ	グローバル
動的	はい
型	ブール

デフォルト	OFF

インデックスを使用しないクエリーがスロークエリーログに記録されるかどうか。セクション5.2.5「スロークエリーログ」を参照してください。

- log_slow_admin_statements

導入	5.6.11
システム変数	log_slow_admin_statements
スコープ	グローバル
動的	はい
型	ブール
デフォルト	OFF

スロークエリーログに書き込まれるステートメントにスロー管理ステートメントを含めます。管理ステートメントには、ALTER TABLE、ANALYZE TABLE、CHECK TABLE、CREATE INDEX、DROP INDEX、OPTIMIZE TABLE、および REPAIR TABLE が含まれます。

この変数は --log-slow-admin-statements オプションの置換として MySQL 5.6.11 で追加されました。システム変数はオプションと同じ方法でコマンド行またはオプションファイルに設定できるため、サーバー起動時に何らかの変更を行う必要はありませんが、システム変数は実行時に値を検査または設定することも可能です。

- log_slow_queries

コマンド行形式	--log-slow-queries[=name]
非推奨	はい (removed in 5.6.1); use slow-query-log instead
システム変数	log_slow_queries
スコープ	グローバル
動的	はい
型	ブール

この変数は MySQL 5.6.1 で削除されました。代わりに slow_query_log を使用してください。

- log_throttle_queries_not_using_indexes

導入	5.6.5
システム変数	log_throttle_queries_not_using_indexes
スコープ	グローバル
動的	はい
型	数値
デフォルト	0

log_queries_not_using_indexes が有効な場合、log_throttle_queries_not_using_indexes 変数は、スロークエリーログに書き込み可能な分あたりのクエリー数を制限します。値 0 (デフォルト) は「制限なし」を意味します。詳細は、セクション5.2.5「スロークエリーログ」を参照してください。

この変数は MySQL 5.6.5 で追加されました。

- log_warnings

コマンド行形式	--log-warnings[=#]
システム変数	log_warnings
スコープ (≥ 5.6.4)	グローバル
スコープ (≤ 5.6.3)	グローバル、セッション
動的	はい
型	数値

デフォルト	1
最小値	0
最大値 (64 ビットプラットフォーム)	18446744073709551615
最大値 (32 ビットプラットフォーム)	4294967295

追加の警告メッセージをエラーログに生成するかどうか。この変数はデフォルトで有効 (1) になっており、これを 0 に設定することによって無効にできます。値が 0 より大きい場合、サーバーはステートメントベースのロギングについて、安全ではないステートメントに関するメッセージをログに記録します。値が 1 より大きい場合、新規接続試行の接続の中止およびアクセス拒否エラーがログに記録されます。

- long_query_time

コマンド行形式	--long-query-time=#
システム変数	long_query_time
スコープ	グローバル、セッション
動的	はい
型	数値
デフォルト	10
最小値	0

クエリーの時間がこの秒数よりかかると、サーバーは Slow_queries ステータス変数を増やします。スロークエリーログが有効な場合、クエリーはスロークエリーログファイルに記録されます。この値は CPU 時間でなくリアルタイムで測定されるため、負荷の軽いシステムでしきい値を下回るクエリーが、負荷の重いシステムではしきい値を超える場合もあります。long_query_time の最小値およびデフォルト値は、それぞれ 0 および 10 です。値はマイクロ秒の精度まで指定できます。ファイルへのロギングの場合、時間はマイクロ秒の部分も含めて書き込まれます。テーブルへのロギングの場合、時間の整数部のみ書き込まれ、マイクロ秒の部分は無視されます。セクション5.2.5「スロークエリーログ」を参照してください。

- low_priority_updates

コマンド行形式	--low-priority-updates
システム変数	low_priority_updates
スコープ	グローバル、セッション
動的	はい
型	ブール
デフォルト	FALSE

1 に設定された場合、すべての INSERT、UPDATE、DELETE、および LOCK TABLE WRITE ステートメントは、影響を受けるテーブルでの保留中の SELECT または LOCK TABLE READ がなくなるまで待機します。これは、テーブルレベルロックのみを使用するストレージエンジン (MyISAM、MEMORY、および MERGE) にのみ影響を与えます。

- lower_case_file_system

システム変数	lower_case_file_system
スコープ	グローバル
動的	いいえ
型	ブール

この変数は、データディレクトリが配置されているファイルシステムでのファイル名の大文字小文字の区別を示します。OFF はファイル名が大文字小文字を区別することを意味し、ON は大文字小文字を区別しないことを意味します。この変数は、ファイルシステム属性を反映するため読み取り専用で、変数を設定してもファイルシステムに影響しません。

- lower_case_table_names

コマンド行形式	--lower-case-table-names[=#]
システム変数	lower_case_table_names
スコープ	グローバル
動的	いいえ
型	数値
デフォルト	0
最小値	0
最大値	2

0 に設定されると、テーブル名は指定したとおりに格納され、比較では大文字と小文字が区別されます。1 に設定されると、テーブル名はディスク上に小文字で格納され、比較では大文字と小文字は区別されません。2 に設定されると、テーブル名は指定したとおりに格納されますが、小文字で比較されます。このオプションはデータベース名やテーブルエイリアスにも適用されます。追加情報については セクション9.2.2「識別子の大文字と小文字の区別」を参照してください。

大文字小文字を区別しないファイル名を持つシステム (Windows や OS X など) で MySQL を実行する場合、この変数を 0 に設定しないでください。そのようなシステムでこの変数を 0 に設定し、大文字を小文字に (あるいは小文字を大文字に) 入れ替えて MyISAM テーブルスペースにアクセスすると、インデックスの破損が発生することがあります。Windows では、デフォルト値は 1 です。OS X では、デフォルト値は 2 です。

InnoDB テーブルを使用する場合、名前を強制的に小文字に変換するために、すべてのプラットフォームでこの値を 1 に設定します。

MySQL 5.6 でのこの変数の設定は、大文字小文字の区別に関するレプリケーションのフィルタ処理オプションの動作に影響します。これは以前のバージョンの MySQL からの変更点です。(バグ #51639) 詳細については、セクション17.2.3「サーバーがレプリケーションフィルタリングルールをどのように評価するか」を参照してください。

以前のバージョンの MySQL では、レプリケーションマスターとスレーブの lower_case_table_names に異なる設定を使用すると、スレーブが大文字小文字を区別するファイルシステムを使用していた場合にレプリケーションが失敗する可能性がありました。この問題は MySQL 5.6.1 で解決されました。詳細については、セクション17.4.1.34「レプリケーションと変数」を参照してください。

- max_allowed_packet

コマンド行形式	--max-allowed-packet=#
システム変数	max_allowed_packet
スコープ	グローバル
動的	はい
型	数値
デフォルト (≥ 5.6.6)	4194304
デフォルト (≤ 5.6.5)	1048576
最小値	1024

最大値	1073741824

1 つのパケット、生成された文字列または中間文字列、または mysql_stmt_send_long_data() C API 関数によって送信されたすべてのパラメータの最大サイズ。デフォルトは、MySQL 5.6.6 では 4M バイト、それより前については 1M バイトです。

パケットメッセージバッファーは net_buffer_length バイトに初期化されますが、必要に応じて max_allowed_packet バイトまで大きくできます。この値はデフォルトでは小さいため、大きい (正しくない可能性がある) パケットをキャッチできません。

大きい BLOB カラムまたは長い文字列を使用している場合、この値を大きくする必要があります。使用する最大の BLOB と同じ大きさにしてください。max_allowed_packet のプロトコル制限は 1G バイトです。値は 1024 の倍数にします。倍数でない場合、もっとも近い倍数に切り下げられます。

max_allowed_packet 変数の値を変更することによってメッセージバッファーサイズを変更するとき、クライアントプログラムでそれが可能である場合は、クライアント側のバッファーサイズも変更します。クライアントライブラリに組み込まれるデフォルトの max_allowed_packet 値は 1G バイトですが、個々のクライアントプログラムはこれをオーバーライドできます。たとえば、mysql および mysqldump のデフォルトは、それぞれ 16M バイトおよび 24M バイトです。また、コマンド行またはオプションファイル内で max_allowed_packet を設定することによって、クライアント側の値を変更することもできます。

この変数のセッションの値は、読み取り専用です。

- max_connect_errors

コマンド行形式	--max-connect-errors=#
システム変数	max_connect_errors
スコープ	グローバル
動的	はい
型	数値
デフォルト (64 ビットプラットフォーム, ≥ 5.6.6)	100
デフォルト (64 ビットプラットフォーム, ≤ 5.6.5)	10
デフォルト (32 ビットプラットフォーム, ≥ 5.6.6)	100
デフォルト (32 ビットプラットフォーム, ≤ 5.6.5)	10
最小値	1
最大値 (64 ビットプラットフォーム)	18446744073709551615
最大値 (32 ビットプラットフォーム)	4294967295

ホストからの連続する接続リクエストが、接続に成功することなくこの数を超えて中断した場合、サーバーはホストのその後の接続をブロックします。ブロックされたホストは、ホストキャッシュをフラッシュしてブロック解除できます。これを行うには、FLUSH HOSTS ステートメントを発行するか、mysqladmin flush-hosts コマンドを実行します。以前の接続が中断されたあと、max_connect_errors 回の試行よりも少ない回数で接続が正常に確立された場合、ホストのエラーカウントはゼロにクリアされます。ただし、ホストがブロックされた場合、ホストキャッシュをフラッシュすることが、ホストのブロックを解除するためのただ 1 つの方法です。デフォルトは、MySQL 5.6.6 以降では 100 で、それより前では 10 です。

- max_connections

コマンド行形式	--max-connections=#
システム変数	max_connections
スコープ	グローバル
動的	はい
型	整数

デフォルト	151
最小値	1
最大値	100000

許可される最大のクライアントの同時接続数。デフォルトでは、これは 151 です。詳細については、セクションB.5.2.7「接続が多すぎます」を参照してください。

この値を大きくすると、mysqld が要求するファイルディスクリプタの数が増加します。必要な数のディスクリプタが利用できない場合、サーバーは max_connections の値を削減します。ファイルディスクリプタの制限に関する解説は、セクション8.4.3.1「MySQL でのテーブルのオープンとクローズの方法」を参照してください。

max_connections 制限に到達したことにより接続が拒否されると、Connection_errors_max_connections ステータス変数が増加します。

- max_delayed_threads

コマンド行形式	--max-delayed-threads=#
非推奨	5.6.7
システム変数	max_delayed_threads
スコープ	グローバル、セッション
動的	はい
型	数値
デフォルト	20
最小値	0
最大値	16384

非トランザクションテーブル用の INSERT DELAYED ステートメントを処理するには、この数を超えるスレッドを開始しないでください。すべての INSERT DELAYED スレッドが使用中になったあとで、データを新規テーブルに挿入しようとした場合、DELAYED 属性が指定されていない場合と同様に行が挿入されます。これを 0 に設定すると、MySQL は DELAYED 行を処理するスレッドを作成せず、実質的に DELAYED が完全に無効になります。

この変数の SESSION 値について、有効な値は 0 または GLOBAL 値のみです。

MySQL 5.6.7 以降では、このシステム変数は非推奨となり (DELAYED 挿入が非推奨となったため)、今後のリリースで削除される予定です。

- max_error_count

コマンド行形式	--max-error-count=#
システム変数	max_error_count
スコープ	グローバル、セッション
動的	はい
型	数値
デフォルト	64
最小値	0
最大値	65535

SHOW ERRORS や SHOW WARNINGS ステートメントで表示するために格納される、エラー、警告、および注記メッセージの最大数。これは診断領域内の条件領域の数と同じで、GET DIAGNOSTICS によって調査できる条件数と同じです。

- max_heap_table_size

コマンド行形式	--max-heap-table-size=#
システム変数	max_heap_table_size
スコープ	グローバル、セッション

動的	はい
型	数値
デフォルト	16777216
最小値	16384
最大値 (64 ビットプラットフォーム)	18446744073709547520
最大値 (32 ビットプラットフォーム)	4294967295

この変数は、ユーザーが作成した MEMORY テーブルの増加が許可される最大サイズを設定します。この変数の値は MEMORY テーブルの MAX_ROWS 値を計算するために使用されます。この変数を設定しても、既存の MEMORY テーブルに影響しませんが、CREATE TABLE などのステートメントでテーブルを再作成したり、ALTER TABLE または TRUNCATE TABLE でテーブルを変更したりした場合は影響します。サーバーを再起動しても、既存の MEMORY テーブルの最大サイズがグローバルの max_heap_table_size 値に設定されます。

この変数は、内部インメモリーテーブルのサイズを制限するために tmp_table_size と一緒に使用されることもあります。セクション8.4.4「MySQL が内部一時テーブルを使用する仕組み」を参照してください。

max_heap_table_size は複製されません。詳しくは、セクション17.4.1.21「レプリケーションと MEMORY テーブル」およびセクション17.4.1.34「レプリケーションと変数」を参照してください。

- max_insert_delayed_threads

非推奨	5.6.7
システム変数	max_insert_delayed_threads
スコープ	グローバル、セッション
動的	はい
型	数値

この変数は、max_delayed_threads のシノニムです。

MySQL 5.6.7 以降では、このシステム変数は非推奨となり (DELAYED 挿入が非推奨となったため)、今後のリリースで削除される予定です。

- max_join_size

コマンド行形式	--max-join-size=#
システム変数	max_join_size
スコープ	グローバル、セッション
動的	はい
型	数値
デフォルト	18446744073709551615
最小値	1
最大値	18446744073709551615

検査が必要となる行数 (単一テーブルステートメントの場合) または行の組み合わせの数 (複数テーブルステートメントの場合) が、max_join_size をおそらく超えるか、ディスクシークが max_join_size 回を超えて実行される可能性があるステートメントを許可しません。この値を設定することで、キーが適切に使用されず長い時間がかかりそうなステートメントをキャッチできます。ユーザーが、WHERE 句のない結合、長い時間がかかる結合、または数百万行を返す結合を実行する傾向がある場合にこれを設定します。

この変数を DEFAULT 以外の値に設定すると、sql_big_selects の値が 0 にリセットされます。sql_big_selects 値を再設定すると、max_join_size 変数は無視されます。

クエリー結果がクエリーキャッシュ内にある場合、結果サイズの検査は実行されません。これは、結果は以前計算されており、結果をクライアントに送信するサーバーに負荷をかけないためです。

- max_length_for_sort_data

コマンド行形式	--max-length-for-sort-data=#
システム変数	max_length_for_sort_data
スコープ	グローバル、セッション
動的	はい
型	数値
デフォルト	1024
最小値	4
最大値	8388608

使用する filesort アルゴリズムを決定するインデックス値のサイズの制限。セクション8.2.1.15「ORDER BY の最適化」を参照してください。

- max_prepared_stmt_count

コマンド行形式	--max-prepared-stmt-count=#
システム変数	max_prepared_stmt_count
スコープ	グローバル
動的	はい
型	数値
デフォルト	16382
最小値	0
最大値	1048576

この変数は、サーバー内のプリペアドステートメントの総数を制限します。(すべてのセッションでのプリペアドステートメントの合計数。)これは、大量のステートメントを作成することによってサーバーの実行するメモリーを不足させることに基づくサービス妨害攻撃の可能性がある環境で使用できます。値が現在のプリペアドステートメントの数より低く設定された場合、既存のステートメントは影響を受けずに使用できますが、現在の数が制限を下回るまで新しいステートメントを作成できません。デフォルト値は 16,382 です。許可される値の範囲は 0 から 100 万までです。値を 0 に設定すると、プリペアドステートメントが無効になります。

- max_relay_log_size

コマンド行形式	--max-relay-log-size=#
システム変数	max_relay_log_size
スコープ	グローバル
動的	はい
型	数値
デフォルト	0
最小値	0
最大値	1073741824

レプリケーションスレーブによるリレーログへの書き込みによって、現在のログファイルサイズがこの変数の値を超えた場合、スレーブはリレーログをローテーションします (現在のファイルを閉じて新しいファイルを開きます)。max_relay_log_size が 0 の場合、サーバーはバイナリログとリレーログの両方に max_binlog_size を使用します。max_relay_log_size が 0 より大きい場合、リレーログのサイズを抑制し、2 つのログに異なるサイズを持たせることが可能になります。max_relay_log_size を 4096 バイトと 1G バイト (両端の値を含む) の間に設定するか、0 にする必要があります。デフォルト値は 0 です。セクション17.2.1「レプリケーション実装の詳細」を参照してください。

- max_seeks_for_key

コマンド行形式	--max-seeks-for-key=#
システム変数	max_seeks_for_key
スコープ	グローバル、セッション

動的	はい
型	数値
デフォルト (64 ビットプラットフォーム)	18446744073709551615
デフォルト (32 ビットプラットフォーム)	4294967295
最小値	1
最大値 (64 ビットプラットフォーム)	18446744073709551615
最大値 (32 ビットプラットフォーム)	4294967295

キーに基づいて行を参照するとき、推定されるシークの最大数を制限します。MySQL オプティマイザは、インデックスをスキャンすることによってテーブル内で一致する行を検索するとき、インデックスの実際のカーディナリティーに関係なく、この数を超えるキーシークは不要であると推定します (セクション 13.7.5.23「SHOW INDEX 構文」を参照してください)。これを低い値 (100 など) に設定することで、MySQL でテーブルスキャンよりもインデックスを優先するように強制できます。

- max_sort_length

コマンド行形式	--max-sort-length=#
システム変数	max_sort_length
スコープ	グローバル、セッション
動的	はい
型	数値
デフォルト	1024
最小値	4
最大値	8388608

データ値をソートするときに使用するバイト数。それぞれの値の最初の max_sort_length バイトだけを使用し、残りは無視されます。

MySQL 5.6.9 以降では、max_sort_length は整数、小数、浮動小数点数、および時間データ型について無視されます。

- max_sp_recursion_depth

コマンド行形式	--max-sp-recursion-depth[=#]
システム変数	max_sp_recursion_depth
スコープ	グローバル、セッション
動的	はい
型	数値
デフォルト	0
最大値	255

任意のストアドプロシージャーを再帰的に呼び出すことができる回数。このオプションのデフォルト値は 0 で、これはストアドプロシージャーの再帰を完全に無効化します。最大値は 255 です。

ストアドプロシージャーの再帰により、スレッドスタック領域の要求が増加します。max_sp_recursion_depth の値を増やした場合、サーバー起動時に thread_stack の値を増やすことによってスレッドスタックサイズを増やすことが必要な場合もあります。

- max_tmp_tables

この変数は使用されません。これは MySQL 5.6.7 以降で非推奨となり、今後の MySQL リリースで削除されます。

- max_user_connections

コマンド行形式	--max-user-connections=#
システム変数	max_user_connections
スコープ	グローバル、セッション
動的	はい
型	数値
デフォルト	0
最小値	0
最大値	4294967295

任意の MySQL ユーザーアカウントに許可された最大同時接続数。値 0 (デフォルト) は「制限なし」を意味します。

この変数は、サーバー起動時または実行時に設定できるグローバル値を持ちます。また、現在のセッションに関連付けられたアカウントに適用される、実際の同時接続制限を示す読み取り専用のセッション値も持ちます。セッション値は次のように初期化されます。

- ユーザーアカウントの MAX_USER_CONNECTIONS リソース制限がゼロでない場合、セッション max_user_connections の値はその制限値に設定されます。
- そうでない場合、セッション max_user_connections の値はグローバル値に設定されます。

アカウントのリソース制限は GRANT ステートメントによって指定されます。セクション6.3.4「アカウントリソース制限の設定」およびセクション13.7.1.4「GRANT 構文」を参照してください。

- max_write_lock_count

コマンド行形式	--max-write-lock-count=#
システム変数	max_write_lock_count
スコープ	グローバル
動的	はい
型	数値
デフォルト (64 ビットプラットフォーム)	18446744073709551615
デフォルト (32 ビットプラットフォーム)	4294967295
最小値	1
最大値 (64 ビットプラットフォーム)	18446744073709551615
最大値 (32 ビットプラットフォーム)	4294967295

この大きさの書き込みロックのあと、保留中の読み取りロックリクエストの処理を間で許可します。

- metadata_locks_cache_size

導入	5.6.4
システム変数	metadata_locks_cache_size
スコープ	グローバル
動的	いいえ
型	数値
デフォルト	1024
最小値	1

最大値	1048576

メタデータロックキャッシュのサイズ。サーバーはこのキャッシュを使用して、同期オブジェクトの作成および破棄を回避します。これは、このような操作にコストがかかる、Windows XP などのシステムで特に役立ちます。この変数は MySQL 5.6.4 で追加されました。

- metadata_locks_hash_instances

導入	5.6.8
システム変数	metadata_locks_hash_instances
スコープ	グローバル
動的	いいえ
型	数値
デフォルト	8
最小値	1
最大値	1024

一連のメタデータロックは別々のハッシュにパーティション化されて、別々のロックハッシュを使用して競合を削減するために使用する異なるオブジェクトにアクセスする接続が可能になります。metadata_locks_hash_instances システム変数は、ハッシュの数を指定します (デフォルトは 8)。この変数は MySQL 5.6.8 で追加されました。

- min_examined_row_limit

コマンド行形式	--min-examined-row-limit=#
システム変数	min_examined_row_limit
スコープ	グローバル、セッション
動的	はい
型	数値
デフォルト	0
最小値	0
最大値 (64 ビットプラットフォーム)	18446744073709551615
最大値 (32 ビットプラットフォーム)	4294967295

これよりも少ない行数を検査するクエリーは、スロークエリーログに記録されません。

- multi_range_count

コマンド行形式	--multi-range-count=#
システム変数	multi_range_count
スコープ	グローバル、セッション
動的	はい
型	数値
デフォルト	256
最小値	1
最大値	4294967295

範囲選択時にテーブルハンドラに一度に送信する範囲の最大数。デフォルト値は 256 です。複数の範囲をハンドラに一度に送信することで、一部の選択のパフォーマンスが劇的に向上します。これは NDBCLUSTER テーブルハンドラについて特に当てはまり、範囲リクエストをすべてのノードに送信する必要があります。これらのリクエストのバッチを一度に送信することで、通信コストが著しく低下します。

この変数は MySQL 5.6.7 で削除されました。

- myisam_data_pointer_size

コマンド行形式	--myisam-data-pointer-size=#
システム変数	myisam_data_pointer_size
スコープ	グローバル
動的	はい
型	数値
デフォルト	6
最小値	2
最大値	7

MAX_ROWS オプションが指定されていない場合に MyISAM テーブルの CREATE TABLE によって使用されるバイト単位のデフォルトポインタサイズ。この値を 2 より小さくしたり 7 より大きくしたりすることはできません。デフォルト値は 6 です。セクションB.5.2.12「テーブルが満杯です」を参照してください。

- myisam_max_sort_file_size

コマンド行形式	--myisam-max-sort-file-size=#
システム変数	myisam_max_sort_file_size
スコープ	グローバル
動的	はい
型	数値
デフォルト (64 ビットプラットフォーム)	9223372036854775807
デフォルト (32 ビットプラットフォーム)	2147483648

MyISAM インデックスを再作成するとき (REPAIR TABLE、ALTER TABLE、または LOAD DATA INFILE 中に)、MySQL が使用を許可されている一時ファイルの最大サイズ。ファイルサイズがこの値より大きい場合、さらに低速なキーキャッシュを代わりに使用してインデックスが作成されます。値はバイト単位で指定されます。

MyISAM インデックスファイルがこのサイズを超えて、ディスクスペースが使用できる場合、この値を大きくするとパフォーマンスが向上することがあります。このスペースは、元のインデックスファイルが配置されているディレクトリを含むファイルシステム内で利用する必要があります。

- myisam_mmap_size

コマンド行形式	--myisam-mmap-size=#
システム変数	myisam_mmap_size
スコープ	グローバル
動的	いいえ
型	数値
デフォルト (64 ビットプラットフォーム)	18446744073709551615
デフォルト (32 ビットプラットフォーム)	4294967295
最小値	7
最大値 (64 ビットプラットフォーム)	18446744073709551615
最大値 (32 ビットプラットフォーム)	4294967295

圧縮された MyISAM ファイルのメモリーマッピングに使用する最大のメモリー量。圧縮された MyISAM テーブルが多く使用される場合、この値を減らすことで、メモリースワッピングの問題が生じるおそれを低下できます。

- myisam_recover_options

システム変数	myisam_recover_options
スコープ	グローバル
動的	いいえ

--myisam-recover-options オプションの値。セクション5.1.3「サーバーコマンドオプション」を参照してください。

- myisam_repair_threads

コマンド行形式	--myisam-repair-threads=#
システム変数	myisam_repair_threads
スコープ	グローバル、セッション
動的	はい
型	数値
デフォルト	1
最小値	1
最大値 (64 ビットプラットフォーム)	18446744073709551615
最大値 (32 ビットプラットフォーム)	4294967295

この値が 1 より大きい場合、MyISAM テーブルインデックスは Repair by sorting プロセス中に並列で作成されます (各インデックスはインデックス独自のスレッド内)。デフォルト値は 1 です。

> **注記**
>
> 複数スレッドの修復は、まだベータ品質コードです。

- myisam_sort_buffer_size

コマンド行形式	--myisam-sort-buffer-size=#
システム変数	myisam_sort_buffer_size
スコープ	グローバル、セッション
動的	はい
型	数値
デフォルト	8388608
最小値	4096
最大値 (その他, 64 ビットプラットフォーム)	18446744073709551615
最大値 (その他, 32 ビットプラットフォーム)	4294967295
最大値 (Windows)	4294967295

REPAIR TABLE 中に MyISAM インデックスをソートするときや、CREATE INDEX または ALTER TABLE を使用してインデックスを作成するときに割り当てられるバッファーのサイズ。

myisam_sort_buffer_size に対して許可される最大の設定値は 4G バイト − 1 です。64 ビットプラットフォームの場合は大きい値が許可されます (64 ビットの Windows の場合は例外で、大きい値は 4G バイト − 1 に切り捨てられて警告が出ます)。

- myisam_stats_method

コマンド行形式	--myisam-stats-method=name
システム変数	myisam_stats_method

スコープ	グローバル、セッション
動的	はい
型	列挙
デフォルト	nulls_unequal
有効な値	nulls_equal nulls_unequal nulls_ignored

MyISAM テーブルのインデックス値の分布に関する統計を収集するときに、サーバーが NULL 値を扱う方法。この変数は、nulls_equal、nulls_unequal、および nulls_ignored の 3 つの値を指定できます。nulls_equal の場合、すべての NULL インデックス値を同等として扱い、NULL 値の数とサイズが同等の単一値グループを生成します。nulls_unequal の場合、NULL 値同士を同等として扱わず、それぞれの NULL はサイズが 1 の別個のグループを生成します。nulls_ignored の場合、NULL 値は無視されます。

テーブル統計を生成するために使用する方法は、セクション8.3.7「InnoDB および MyISAM インデックス統計コレクション」に記載されているように、オプティマイザがクエリー実行のためのインデックスを選択する方法に影響を与えます。

- myisam_use_mmap

コマンド行形式	--myisam-use-mmap
システム変数	myisam_use_mmap
スコープ	グローバル
動的	はい
型	ブール
デフォルト	OFF

MyISAM テーブルの読み取りおよび書き込みでメモリーマッピングを使用します。

- named_pipe

システム変数	named_pipe
スコープ	グローバル
動的	いいえ
プラットフォーム固有	Windows
型	ブール
デフォルト	OFF

(Windows のみ。)サーバーが名前付きパイプでの接続をサポートしているかどうかを指定します。

- net_buffer_length

コマンド行形式	--net-buffer-length=#
システム変数	net_buffer_length
スコープ	グローバル、セッション
動的	はい
型	数値
デフォルト	16384
最小値	1024

最大値	1048576

各クライアントスレッドは、接続バッファーおよび結果バッファーに関連付けられています。両者は net_buffer_length で与えられたサイズで開始されますが、必要に応じて、max_allowed_packet バイトまで動的に拡大できます。結果バッファーは、各 SQL ステートメントのあとで net_buffer_length に縮小されます。

この変数は通常は変更しませんが、メモリーが非常に少ない場合、クライアントによって送信される予想されるステートメントの長さに設定できます。ステートメントがこの長さを超えた場合、接続バッファーは自動的に拡大されます。net_buffer_length の最大値は 1M バイトに設定できます。

この変数のセッションの値は、読み取り専用です。

- net_read_timeout

コマンド行形式	--net-read-timeout=#
システム変数	net_read_timeout
スコープ	グローバル、セッション
動的	はい
型	数値
デフォルト	30
最小値	1

読み取りを中止する前に接続からのデータを待機する秒数。サーバーがクライアントからの読み込みを行うとき、net_read_timeout は中止するタイミングを制御するタイムアウト値です。サーバーがクライアントに書き込みを行うとき、net_write_timeout は中止するタイミングを制御するタイムアウト値です。slave_net_timeoutも参照してください。

- net_retry_count

コマンド行形式	--net-retry-count=#
システム変数	net_retry_count
スコープ	グローバル、セッション
動的	はい
型	数値
デフォルト	10
最小値	1
最大値 (64 ビットプラットフォーム)	18446744073709551615
最大値 (32 ビットプラットフォーム)	4294967295

通信ポートでの読み取りまたは書き込みが中断された場合、停止するまでこの回数だけ再試行します。FreeBSD では内部の中断がすべてのスレッドに送信されるため、この値をきわめて高く設定するようにしてください。

- net_write_timeout

コマンド行形式	--net-write-timeout=#
システム変数	net_write_timeout
スコープ	グローバル、セッション
動的	はい
型	数値
デフォルト	60
最小値	1

書き込みを中止する前にブロックが接続に書き込まれるのを待機する秒数。net_read_timeout も参照してください。

- new

コマンド行形式	--new
システム変数	new
スコープ	グローバル、セッション
動的	はい
無効化	skip-new
型	ブール
デフォルト	FALSE

この変数は、一部の 4.1 の動作をオンにするために MySQL 4.0 で使用されており、下位互換性のために保持されています。MySQL 5.6 では、この値は常に OFF です。

- old

コマンド行形式	--old
システム変数	old
スコープ	グローバル
動的	いいえ

old は互換性変数です。これはデフォルトでは無効化されていますが、以前のバージョンに存在した動作にサーバーを戻すために、起動時に有効にできます。

現時点では、old が有効化された場合、インデックスヒントのデフォルトのスコープを、MySQL 5.1.17 以前に使用されていたものに変更します。つまり、FOR 句を使用しないインデックスヒントは、インデックスが行の取得に使用する方法についてのみ適用され、ORDER BY 句または GROUP BY 句の解決には適用されません。(セクション13.2.9.3「インデックスヒントの構文」を参照してください。)レプリケーションのセットアップでこれを有効にする場合は注意してください。ステートメントベースのバイナリロギングで、マスターとスレーブに異なるモードを指定するとレプリケーションエラーが発生する場合があります。

- old_alter_table

コマンド行形式	--old-alter-table
システム変数	old_alter_table
スコープ	グローバル、セッション
動的	はい
型	ブール
デフォルト	OFF

この変数を有効にすると、サーバーは ALTER TABLE 操作を処理する最適化された方法を使用しません。一時テーブルの使用に戻り、データのコピー後、MySQL 5.0 以前で使用されていたように、一時テーブルを元のテーブルの名前に変更します。ALTER TABLE の操作について詳しくは、セクション13.1.7「ALTER TABLE 構文」を参照してください。

- old_passwords

システム変数	old_passwords
スコープ	グローバル、セッション
動的	はい
型 (≥ 5.6.6)	列挙
型 (≤ 5.6.5)	ブール
デフォルト	0
有効な値	0 1 2

この変数は、PASSWORD() 関数によって使用されるパスワードハッシュ方式を制御します。これは、IDENTIFIED BY 句を使用してパスワードを指定する CREATE USER および GRANT ステートメントによって実行されるパスワードハッシュにも影響します。

次の表は、old_passwords の許可される値、それぞれの値に対するパスワードハッシュ方式、およびそれぞれの方式でハッシュされたパスワードを使用する認証プラグインを示します。これらの値は MySQL 5.6.6 以降で許可されます。5.6.6 より前では、許可される値は 0 (または OFF) および 1 (または ON) です。

値	パスワードハッシュ方式	関連付けられた認証プラグイン
0	MySQL 4.1 ネイティブハッシュ	mysql_native_password
1	4.1 以前の (「古い」) ハッシュ	mysql_old_password
2	SHA-256 ハッシュ	sha256_password

old_passwords=1 の場合、PASSWORD(str) は OLD_PASSWORD(str) と同じ値を返します。後者の関数は old_passwords の値によって影響を受けません。

old_passwords=2 を設定する場合、セクション6.3.8.4「SHA-256 認証プラグイン」の sha256_password プラグインを使用するための指示に従ってください。

MySQL 5.6.6 以降では、サーバーは起動中に、デフォルトの認証プラグインによって必要となるパスワードハッシュ方式と整合性がとれるようにグローバルの old_passwords 値を設定します。--default-authentication-plugin オプションが別のものに設定されないかぎり、デフォルトプラグインは mysql_native_password です。

MySQL 5.6.10 以降では、クライアントがサーバーに正常に接続すると、サーバーはアカウント認証方式について適切なセッション old_passwords 値を設定します。たとえば、アカウントが sha256_password 認証プラグインを使用する場合、サーバーは old_passwords=2 を設定します。

認証プラグインおよびハッシュ形式についての追加情報は、セクション6.3.7「プラガブル認証」およびセクション6.1.2.4「MySQL でのパスワードハッシュ」を参照してください。

> **注記**
>
> 4.1 より前のハッシュ方式を使用するパスワードはネイティブのパスワードハッシュ方式を使用するパスワードよりもセキュアでないため、使用しないようにしてください。4.1 よりも前のパスワードは非推奨であり、これらのサポートは今後の MySQL リリースで削除される予定です。その結果、PASSWORD() で 4.1 以前のパスワードハッシュを生成する old_passwords=1 も非推奨となります。アカウントのアップグレード手順については、セクション6.3.8.3「4.1 よりも前のパスワードハッシュ方式と mysql_old_password プラグインからの移行」を参照してください。

- open_files_limit

コマンド行形式	--open-files-limit=#
システム変数	open_files_limit
スコープ	グローバル
動的	いいえ
型	数値
デフォルト (≥ 5.6.8)	5000, with possible adjustment
デフォルト (≤ 5.6.7)	0
最小値	0
最大値	platform dependent

オペレーティングシステムで mysqld が開くことを許可するファイル数。実行時でのこの変数の値はシステムによって許可される実際の値であるため、サーバー起動時に指定した値と異なる場合があります。MySQL がオープンファイルの数を変更できないシステムでは、値は 0 です。

実際の open_files_limit の値は、システム起動時に指定された値 (ある場合) と、max_connections および table_open_cache の値に基づき、次の式を使用します。

```
1) 10 + max_connections + (table_open_cache * 2)
2) max_connections * 5
```

3) open_files_limit value specified at startup, 5000 if none

サーバーはこれらの 3 つの値の最大値を使用して、ファイルディスクリプタの数を取得しようとします。その数のディスクリプタが取得できない場合、サーバーはシステムに許可されるできるだけ多くの数を取得しようとします。

- optimizer_join_cache_level

コマンド行形式	--optimizer-join-cache-level=#
導入	5.6.1
削除	5.6.3
システム変数	optimizer_join_cache_level
スコープ	グローバル、セッション
動的	はい
型	数値
デフォルト	4
最小値	0
最大値	4

MySQL 5.6.3 より前では、この変数は結合バッファー管理に使用されます。これは結合操作のための結合バッファーの使用法を制御します。MySQL 5.6.3 以降では、これは削除されて optimizer_switch 変数が代わりに使用されます。セクション8.2.1.14「Block Nested Loop 結合と Batched Key Access 結合」を参照してください。

次の表は、許可される optimizer_join_cache_level 値を示します。

オプション	説明
0	すべての結合操作で結合バッファーは使用されません。この設定は、結合バッファリングの使用を可能にする非ゼロ値でのパフォーマンスと比較した、結合の基準パフォーマンスの評価に役立てることができます。
1	これはデフォルト値です。結合バッファーは、元の Block Nested-Loop (BNL) 結合アルゴリズムによって実行される内部結合についてのみ利用されます。このアルゴリズムが適用された場合、内部テーブルの行は、テーブルスキャン、単純インデックススキャン、または範囲インデックススキャンによってアクセスされます。
2	サーバーは、その最初のオペランドが結合バッファー自体を使用する結合操作によって生成される場合、結合操作に対して増分結合バッファーを使用します。
3	1 つの内部テーブルを持つ外部結合操作および内部結合について、BNL アルゴリズムが使用されます。
4	BNL アルゴリズムは、内部テーブルに対して増分バッファーを使用します。この場合、BNL アルゴリズムはネストされた外部結合 (いくつかの内部テーブルを持つ外部結合) について使用できます。このような操作は、最初のものを除くすべての内部テーブルを結合するために増分結合バッファーが使用される場合にのみ実行できます。

- optimizer_prune_level

コマンド行形式	--optimizer-prune-level[=#]
システム変数	optimizer_prune_level
スコープ	グローバル、セッション
動的	はい
型	ブール
デフォルト	1

見込みのない部分的プランをオプティマイザ検索スペースから削除するために、クエリー最適化中に適用される経験則を制御します。値 0 は、オプティマイザが網羅的な検索を実行できるよう経験則を無効にします。値 1 は、中間プランによって取得された行の数に基づいて、オプティマイザにプランを削除させます。

- optimizer_search_depth

コマンド行形式	--optimizer-search-depth[=#]
システム変数	optimizer_search_depth
スコープ	グローバル、セッション
動的	はい
型	数値
デフォルト	62
最小値	0
最大値	62

クエリーオプティマイザによって実行される検索の最大の深さ。クエリー内の関係の数より値が大きいと、適切なクエリー計画が得られますが、クエリーの実行計画の生成に時間がかかります。クエリー内の関係の数より値が小さいと、実行プランがすばやく返されますが、結果のプランがまったく最適にならないことがあります。0 に設定された場合、システムは合理的な値を自動的に選択します。

- optimizer_switch

コマンド行形式	--optimizer-switch=value
システム変数	optimizer_switch
スコープ	グローバル、セッション
動的	はい
型	セット
有効な値 (≥ 5.6.9)	batched_key_access={on|off} block_nested_loop={on|off} engine_condition_pushdown={on|off} firstmatch={on|off} index_condition_pushdown={on|off} index_merge={on|off} index_merge_intersection={on|off} index_merge_sort_union={on|off} index_merge_union={on|off} loosescan={on|off} materialization={on|off} mrr={on|off} mrr_cost_based={on|off} semijoin={on|off} subquery_materialization_cost_based={on|off} use_index_extensions={on|off}
有効な値 (≥ 5.6.7, ≤ 5.6.8)	batched_key_access={on|off} block_nested_loop={on|off} engine_condition_pushdown={on|off} firstmatch={on|off}

	index_condition_pushdown={on|off} index_merge={on|off} index_merge_intersection={on|off} index_merge_sort_union={on|off} index_merge_union={on|off} loosescan={on|off} materialization={on|off} mrr={on|off} mrr_cost_based={on|off} semijoin={on|off} subquery_materialization_cost_based={on|off}
有効な値 (≥ 5.6.5, ≤ 5.6.6)	batched_key_access={on|off} block_nested_loop={on|off} engine_condition_pushdown={on|off} firstmatch={on|off} index_condition_pushdown={on|off} index_merge={on|off} index_merge_intersection={on|off} index_merge_sort_union={on|off} index_merge_union={on|off} loosescan={on|off} materialization={on|off} mrr={on|off} mrr_cost_based={on|off} semijoin={on|off}
有効な値 (≥ 5.6.3, ≤ 5.6.4)	batched_key_access={on|off} block_nested_loop={on|off} engine_condition_pushdown={on|off} index_condition_pushdown={on|off} index_merge={on|off} index_merge_intersection={on|off} index_merge_sort_union={on|off} index_merge_union={on|off} mrr={on|off} mrr_cost_based={on|off}

有効な値 (≥ 5.6.1, ≤ 5.6.2)	engine_condition_pushdown={on|off} index_condition_pushdown={on|off} index_merge={on|off} index_merge_intersection={on|off} index_merge_sort_union={on|off} index_merge_union={on|off} mrr={on|off} mrr_cost_based={on|off}
有効な値 (5.6.0)	engine_condition_pushdown={on|off} index_merge={on|off} index_merge_intersection={on|off} index_merge_sort_union={on|off} index_merge_union={on|off}

optimizer_switch システム変数を使用するとオプティマイザの動作を制御できます。この変数の値はフラグのセットで、各フラグは対応するオプティマイザの動作の有効または無効を示す on または off を値を持ちます。この変数はグローバル値およびセッション値を持ち、実行時に変更できます。グローバル値のデフォルトはサーバーの起動時に設定できます。

オプティマイザの現在のフラグセットを表示するには、変数値を選択します。

```
mysql> SELECT @@optimizer_switch\G
*************************** 1. row ***************************
@@optimizer_switch: index_merge=on,index_merge_union=on,
                    index_merge_sort_union=on,
                    index_merge_intersection=on,
                    engine_condition_pushdown=on,
                    index_condition_pushdown=on,
                    mrr=on,mrr_cost_based=on,
                    block_nested_loop=on,batched_key_access=off,
                    materialization=on,semijoin=on,loosescan=on,
                    firstmatch=on,
                    subquery_materialization_cost_based=on,
                    use_index_extensions=on
```

この変数の構文と、制御するオプティマイザの動作の詳細については、セクション8.8.5.2「切り替え可能な最適化の制御」を参照してください。

- optimizer_trace

導入	5.6.3
システム変数	optimizer_trace
スコープ	グローバル、セッション
動的	はい
型	文字列

この変数はオプティマイザのトレースを制御します。詳細については、「MySQL Internals: Tracing the Optimizer」を参照してください。この変数は MySQL 5.6.3 で追加されました。

- optimizer_trace_features

導入	5.6.3
システム変数	optimizer_trace_features
スコープ	グローバル、セッション
動的	はい

型	文字列

この変数は選択されたオプティマイザトレース機能を有効または無効にします。詳細については、「MySQL Internals: Tracing the Optimizer」を参照してください。この変数は MySQL 5.6.3 で追加されました。

- optimizer_trace_limit

導入	5.6.3
システム変数	optimizer_trace_limit
スコープ	グローバル、セッション
動的	はい
型	数値
デフォルト	1

表示するオプティマイザトレースの最大数。詳細については、「MySQL Internals: Tracing the Optimizer」を参照してください。この変数は MySQL 5.6.3 で追加されました。

- optimizer_trace_max_mem_size

導入	5.6.3
システム変数	optimizer_trace_max_mem_size
スコープ	グローバル、セッション
動的	はい
型	数値
デフォルト	16384

格納されるオプティマイザトレースの最大累積サイズ。詳細については、「MySQL Internals: Tracing the Optimizer」を参照してください。この変数は MySQL 5.6.3 で追加されました。

- optimizer_trace_offset

導入	5.6.3
システム変数	optimizer_trace_offset
スコープ	グローバル、セッション
動的	はい
型	数値
デフォルト	-1

表示するオプティマイザトレースのオフセット。詳細については、「MySQL Internals: Tracing the Optimizer」を参照してください。この変数は MySQL 5.6.3 で追加されました。

- performance_schema_xxx

パフォーマンススキーマのシステム変数は、セクション22.12「パフォーマンススキーマシステム変数」にリストされています。これらの変数は、パフォーマンススキーマ操作を構成するために使用されることもあります。

- pid_file

コマンド行形式	--pid-file=file_name
システム変数	pid_file
スコープ	グローバル
動的	いいえ
型	ファイル名

プロセス ID (PID) ファイルのパス名。この変数は、--pid-file オプションで設定できます。

- plugin_dir

コマンド行形式	--plugin-dir=path
システム変数	plugin_dir
スコープ	グローバル
動的	いいえ
型	ディレクトリ名
デフォルト	BASEDIR/lib/plugin

プラグインディレクトリのパス名。

プラグインディレクトリがサーバーによって書き込み可能な場合、ユーザーは SELECT ... INTO DUMPFILE を使用して、ディレクトリ内のファイルに実行可能コードを書き込むことができます。これを防ぐために、plugin_dir をサーバーに対して読み取り専用にしたり、SELECT 書き込みが安全に実行できるディレクトリに --secure-file-priv を設定したりできます。

- port

コマンド行形式	--port=#
システム変数	port
スコープ	グローバル
動的	いいえ
型	数値
デフォルト	3306
最小値	0
最大値	65535

サーバーが TCP/IP 接続を listen するポートの数。この変数は、--port オプションで設定できます。

- preload_buffer_size

コマンド行形式	--preload-buffer-size=#
システム変数	preload_buffer_size
スコープ	グローバル、セッション
動的	はい
型	数値
デフォルト	32768
最小値	1024
最大値	1073741824

インデックスをプリロードしたときに割り当てられるバッファーのサイズ。

- profiling

0 または OFF (デフォルト) に設定した場合、ステートメントのプロファイリングは無効になります。1 または ON に設定した場合、ステートメントのプロファイリングは有効になり、SHOW PROFILE および SHOW PROFILES ステートメントはプロファイリング情報へのアクセスを提供します。セクション13.7.5.32「SHOW PROFILES 構文」を参照してください。

この変数は MySQL 5.6.8 で非推奨となり、将来の MySQL リリースで削除されます。

- profiling_history_size

profiling が有効な場合にプロファイリング情報を保持する対象となるステートメントの数。デフォルト値は 15 です。最大値は 100 です。値を 0 に設定すると、プロファイリングは実質的に無効になります。セクション13.7.5.32「SHOW PROFILES 構文」を参照してください。

この変数は MySQL 5.6.8 で非推奨となり、将来の MySQL リリースで削除されます。

- protocol_version

システム変数	protocol_version
スコープ	グローバル
動的	いいえ
型	数値

MySQL Server によって使用されるクライアント/サーバープロトコルのバージョン。

- proxy_user

システム変数	proxy_user
スコープ	セッション
動的	いいえ
型	文字列

現在のクライアントが別のユーザーのプロキシの場合、この変数はプロキシユーザーのアカウント名です。そうでない場合、この変数は NULL です。セクション6.3.9「プロキシユーザー」を参照してください。

- pseudo_slave_mode

導入	5.6.10
システム変数	pseudo_slave_mode
スコープ	セッション
動的	はい
型	数値

この変数は内部サーバーで使用します。これは MySQL 5.6.10 で追加されました。

- pseudo_thread_id

システム変数	pseudo_thread_id
スコープ	セッション
動的	はい
型	数値

この変数は内部サーバーで使用します。

- query_alloc_block_size

コマンド行形式	--query-alloc-block-size=#
システム変数	query_alloc_block_size
スコープ	グローバル、セッション
動的	はい
型	数値
デフォルト	8192
最小値	1024
最大値 (64 ビットプラットフォーム)	18446744073709551615
最大値 (32 ビットプラットフォーム)	4294967295
ブロックサイズ	1024

ステートメントの解析および実行中に作成されるオブジェクトに対して割り当てられるメモリーブロックの割り当てサイズ。メモリーのフラグメント化について問題がある場合、このパラメータを増やすと役立つ場合があります。

- query_cache_limit

コマンド行形式	--query-cache-limit=#
システム変数	query_cache_limit
スコープ	グローバル
動的	はい
型	数値
デフォルト	1048576
最小値	0
最大値 (64 ビットプラットフォーム)	18446744073709551615
最大値 (32 ビットプラットフォーム)	4294967295

このバイト数より大きい結果をキャッシュしません。デフォルト値は 1M バイトです。

- query_cache_min_res_unit

コマンド行形式	--query-cache-min-res-unit=#
システム変数	query_cache_min_res_unit
スコープ	グローバル
動的	はい
型	数値
デフォルト	4096
最小値	512
最大値 (64 ビットプラットフォーム)	18446744073709551615
最大値 (32 ビットプラットフォーム)	4294967295

クエリーキャッシュによって割り当てられたブロックの最小サイズ (バイト単位)。デフォルト値は 4096 (4K バイト) です。この変数の調整についての情報は、セクション8.9.3.3「クエリーキャッシュの構成」に提供されています。

- query_cache_size

コマンド行形式	--query-cache-size=#
システム変数	query_cache_size
スコープ	グローバル
動的	はい
型	数値
デフォルト (64 ビットプラットフォーム, ≥ 5.6.8)	1048576
デフォルト (64 ビットプラットフォーム, ≤ 5.6.7)	0
デフォルト (32 ビットプラットフォーム, ≥ 5.6.8)	1048576
デフォルト (32 ビットプラットフォーム, ≤ 5.6.7)	0
最小値	0
最大値 (64 ビットプラットフォーム)	18446744073709551615

最大値 (32 ビットプラットフォーム)	4294967295

クエリー結果をキャッシュするために割り当てられたメモリーの量。デフォルトでは、クエリーキャッシュは無効化されます。これはデフォルト値の 1M と、query_cache_type のデフォルトの 0 を使用することによって実行できます。(MySQL 5.6.8 より前では、デフォルトサイズは 0 で、デフォルトの query_cache_type は 1 です。クエリーキャッシュを使用しない場合、オーバーヘッドを大幅に削減するには query_cache_type=0 でサーバーを開始する必要もあります。)

許可される値は 1024 の倍数で、その他の値はもっとも近い倍数に切り下げられます。query_cache_type が 0 に設定されても、query_cache_size バイトのメモリーは割り当てられることに注意してください。詳細については、セクション8.9.3.3「クエリーキャッシュの構成」を参照してください。

クエリーキャッシュはその構造を割り当てるために最低約 40K バイトのサイズが必要です。(正確なサイズはシステムアーキテクチャーによります。)query_cache_size の値を小さく設定しすぎると、セクション8.9.3.3「クエリーキャッシュの構成」に記載されているような警告が発生します。

- query_cache_type

コマンド行形式	--query-cache-type=#
システム変数	query_cache_type
スコープ	グローバル、セッション
動的	はい
型	列挙
デフォルト (≥ 5.6.8)	0
デフォルト (≤ 5.6.7)	1
有効な値	0 1 2

クエリーキャッシュタイプを設定します。GLOBAL 値を設定すると、これ以降に接続するクライアントのタイプが設定されます。個別のクライアントは SESSION 値を設定することで、クエリーキャッシュの独自の使用に影響を及ぼします。設定可能な値を次の表に示します。

オプション	説明
0 または OFF	クエリーキャッシュに結果をキャッシュしたり、クエリーキャッシュから結果を取得したりしません。これはクエリーキャッシュバッファーを割り当て解除しません。これを行うには query_cache_size を 0 に設定します。
1 または ON	SELECT SQL_NO_CACHE で始まるものを除くキャッシュ可能なすべてのクエリー結果をキャッシュします。
2 または DEMAND	SELECT SQL_CACHE で始まるキャッシュ可能なクエリーのみ結果をキャッシュします。

この変数のデフォルトは、MySQL 5.6.8 以降では OFF で、それより前は ON です。

query_cache_type を 0 に設定してサーバーを開始した場合、クエリーキャッシュ相互排他ロックを取得できません。これは、クエリーキャッシュを実行時に有効化できず、クエリー実行のオーバーヘッドが削減されることを意味します。

- query_cache_wlock_invalidate

コマンド行形式	--query-cache-wlock-invalidate
システム変数	query_cache_wlock_invalidate
スコープ	グローバル、セッション
動的	はい
型	ブール
デフォルト	FALSE

通常、あるクライアントが MyISAM テーブルの WRITE ロックを獲得した場合、クエリー結果がクエリーキャッシュ内にあると、別のクライアントがそのテーブルから読み取るステートメントの発行はブロックされません。この値を 1 にした場合、テーブルに対する WRITE ロックが獲得されて、そのテーブルを参照するクエリーキャッシュ内のすべてのクエリーが無効化されます。これにより、そのテーブルにアクセスしようとするほかのクライアントは、ロック有効時に待機するよう強制されます。

- query_prealloc_size

コマンド行形式	--query-prealloc-size=#
システム変数	query_prealloc_size
スコープ	グローバル、セッション
動的	はい
型	数値
デフォルト	8192
最小値	8192
最大値 (64 ビットプラットフォーム)	18446744073709551615
最大値 (32 ビットプラットフォーム)	4294967295
ブロックサイズ	1024

ステートメントの解析および実行に使用される永続バッファーのサイズ。このバッファーは、ステートメント間で解放されません。複雑なクエリーを発行する場合、query_prealloc_size の値を大きくすると、クエリー実行操作時にサーバーがメモリー割り当てを実行する必要性が低くなるため、パフォーマンスの向上に役立つ場合があります。

- rand_seed1

rand_seed1 および rand_seed2 変数は、セッション変数としてのみ存在し、設定はできますが読み取ることはできません。変数は SHOW VARIABLES の出力に表示されますが、その値は表示されません。

これらの変数の目的は、RAND() 関数のレプリケーションをサポートすることです。RAND() を呼び出すステートメントでは、マスターは 2 つの値をスレーブに渡し、スレーブではこれらの値は乱数ジェネレータにシードを指定するために使用されます。スレーブはこれらの値を使用して、セッション変数 rand_seed1 および rand_seed2 を設定し、スレーブの RAND() はマスターと同じ値を生成します。

- rand_seed2

rand_seed1 の説明を参照してください。

- range_alloc_block_size

コマンド行形式	--range-alloc-block-size=#
システム変数	range_alloc_block_size
スコープ	グローバル、セッション
動的	はい
型	数値
デフォルト (64 ビットプラットフォーム)	2048
デフォルト (64 ビットプラットフォーム)	4096
デフォルト (32 ビットプラットフォーム)	4096
デフォルト (32 ビットプラットフォーム)	2048
最小値 (64 ビットプラットフォーム)	2048

最小値 (64 ビットプラットフォーム)	4096
最小値 (32 ビットプラットフォーム)	4096
最小値 (32 ビットプラットフォーム)	2048
最大値 (64 ビットプラットフォーム)	18446744073709551615
最大値 (64 ビットプラットフォーム)	18446744073709547520
最大値 (32 ビットプラットフォーム)	4294967295
最大値 (32 ビットプラットフォーム)	4294967295
ブロックサイズ	1024

範囲の最適化を行うときに割り当てられるブロックのサイズ。

- read_buffer_size

コマンド行形式	--read-buffer-size=#
システム変数	read_buffer_size
スコープ	グローバル、セッション
動的	はい
型	数値
デフォルト	131072
最小値	8200
最大値	2147479552

MyISAM テーブルの順次スキャンを実行する各スレッドは、スキャンする各テーブルにこのサイズ (バイト単位) のバッファーを割り当てます。多くの順次スキャンを実行する場合、この値を増やした方がよい場合もあり、デフォルトは 131072 です。この変数の値は 4K バイトの倍数にしてください。これが 4K バイトの倍数でない値に設定された場合、値は 4K バイトにもっとも近い倍数に切り下げられます。

このオプションは、すべての検索エンジンの次のコンテキストでも使用されます。

- ORDER BY で行をソートするとき、インデックスを一時ファイル (一時テーブルではない) にキャッシュする場合。
- パーティションに一括挿入する場合。
- ネストされたクエリーの結果をキャッシュする場合。

さらに、ストレージエンジンに固有の 1 つの方法、つまり MEMORY テーブルのメモリーブロックサイズを決定するためにも使用されます。

read_buffer_size で許可される最大の設定は 2G バイトです。

さまざまな操作中でのメモリー使用についての詳細は、セクション8.11.4.1「MySQL のメモリーの使用方法」を参照してください。

- read_only

コマンド行形式	--read-only
システム変数	read_only
スコープ	グローバル
動的	はい
型	ブール

デフォルト	false

この変数はデフォルトでオフにされています。これが有効にされた場合、サーバーは SUPER 権限を持つユーザーか、(スレーブサーバー上で) スレーブスレッドによって実行される更新を除く更新を許可しません。レプリケーションセットアップでは、スレーブサーバーで read_only を有効にして、スレーブがマスターサーバーからの更新のみ受け入れ、クライアントからは受け入れないようにすると便利です。

read_only は TEMPORARY テーブルに適用されず、サーバーがログテーブルに行を挿入しないようにすることもできません (セクション5.2.1「一般クエリーログおよびスロークエリーログの出力先の選択」を参照してください)。この変数の目的は、テーブルの構造または内容への変更を防ぐことであるため、ANALYZE TABLE または OPTIMIZE TABLE ステートメントの使用を妨げることはありません。分析および最適化は、そのような変更の条件を備えていません。つまり、例を挙げると、読み取り専用スレーブでの一貫性検査は、mysqlcheck --all-databases --analyze を使用して実行できるということを意味します。

read_only は GLOBAL 変数としてのみ存在するため、値を変更するには SUPER 権限が必要です。マスターサーバー上での read_only への変更は、スレーブサーバーに複製されません。マスター上での設定に関係なく、値をスレーブサーバー上で設定できます。

> **重要**
>
> MySQL 5.6 で、read_only を有効にすると、SUPER 権限を持っていないすべてのユーザーは、SET PASSWORD ステートメントを使用できなくなります。これは必ずしもすべての MySQL リリースシリーズに当てはまるわけではありません。1 つの MySQL リリースシリーズから別のリリースシリーズ (たとえば、MySQL 5.0 マスターから MySQL 5.1 あるいはそれ以降のスレーブ) にレプリケーションするとき、マスターとスレーブの両方が実行するバージョンのドキュメントを確認し、この read_only の動作が同じか同じでないかを判定し、異なる場合は、アプリケーションに影響があるかどうかを調べます。

次の条件が適用されます。

- 明示的なロック (LOCK TABLES で取得) または保留中のトランザクションがある場合に read_only を有効にしようとすると、エラーが発生します。
- 他のクライアントが明示的なテーブルロックを保持しているか、保留中のトランザクションを持っている場合に read_only の有効化を試行すると、ロックが解放されてトランザクションが終了するまで試行がブロックされます。read_only の有効化の試行が保留されているとき、ほかのクライアントによるテーブルロックあるいはトランザクションの開始のリクエストもまた read_only が設定されるまでブロックされます。
- グローバル読み取りロック (FLUSH TABLES WITH READ LOCK で取得) にはテーブルロックが含まれていないため、read_only を有効化できます。

MySQL 5.6 で、メタデータロックを保持するアクティブなトランザクションに対して read_only を設定する試行は、それらのトランザクションが終了するまでブロックされます。

- read_rnd_buffer_size

コマンド行形式	--read-rnd-buffer-size=#
システム変数	read_rnd_buffer_size
スコープ	グローバル、セッション
動的	はい
型	数値
デフォルト	262144
最小値	1
最大値	2147483647

この変数は、MyISAM テーブルからの読み取り、ストレージエンジン、および Multi-Range Read の最適化のために使用されます。

キーソート操作のあとで、MyISAM テーブルの行をソート順に読み取るとき、ディスクシークを回避するため行はこのバッファーから読み取られます。セクション8.2.1.15「ORDER BY の最適化」を参照してください。この変数を大きい値に設定すると、ORDER BY のパフォーマンスを大幅に向上できます。ただし、これは各ク

ライアントに割り当てられるバッファーであるため、グローバル変数を大きい値に設定しないでください。代わりに、大規模なクエリーを実行する必要があるクライアント内からのみセッション変数を変更します。

read_rnd_buffer_size の許可される最大の設定は 2G バイトです。

さまざまな操作中でのメモリー使用についての詳細は、セクション8.11.4.1「MySQL のメモリーの使用方法」を参照してください。Multi-Range Read の最適化については、セクション8.2.1.13「Multi-Range Read の最適化」を参照してください。

- relay_log_purge

コマンド行形式	--relay-log-purge
システム変数	relay_log_purge
スコープ	グローバル
動的	はい
型	ブール
デフォルト	TRUE

リレーログファイルが不要になったときに自動的にパージするよう無効または有効にします。デフォルト値は 1 (ON) です。

- relay_log_space_limit

コマンド行形式	--relay-log-space-limit=#
システム変数	relay_log_space_limit
スコープ	グローバル
動的	いいえ
型	数値
デフォルト	0
最小値	0
最大値 (64 ビットプラットフォーム)	18446744073709551615
最大値 (32 ビットプラットフォーム)	4294967295

すべてのリレーログに使用するスペースの最大量。

- report_host

コマンド行形式	--report-host=host_name
システム変数	report_host
スコープ	グローバル
動的	いいえ
型	文字列

--report-host オプションの値。

- report_password

コマンド行形式	--report-password=name
システム変数	report_password
スコープ	グローバル
動的	いいえ
型	文字列

--report-password オプションの値。MySQL レプリケーションユーザーアカウントについて使用されるパスワードと同じではありません。

- report_port

コマンド行形式	--report-port=#
システム変数	report_port
スコープ	グローバル
動的	いいえ
型	数値
デフォルト (≥ 5.6.5)	[slave_port]
デフォルト (≤ 5.6.4)	0
最小値	0
最大値	65535

--report-port オプションの値。

- report_user

コマンド行形式	--report-user=name
システム変数	report_user
スコープ	グローバル
動的	いいえ
型	文字列

--report-user オプションの値。MySQL レプリケーションユーザーアカウントについての名前と同じではありません。

- rpl_semi_sync_master_enabled

システム変数	rpl_semi_sync_master_enabled
スコープ	グローバル
動的	はい
型	ブール
デフォルト	OFF

準同期レプリケーションがマスター上で有効かどうかを制御します。プラグインを有効または無効にするには、この変数を ON または OFF (あるいは 1 または 0) にそれぞれ設定します。デフォルトは OFF です。

この変数は、マスター側の準同期レプリケーションプラグインがインストールされている場合のみ利用できます。

- rpl_semi_sync_master_timeout

システム変数	rpl_semi_sync_master_timeout
スコープ	グローバル
動的	はい
型	数値
デフォルト	10000

タイムアウトが発生して非同期レプリケーションに戻すまでに、スレーブからの肯定応答のコミットをマスターが待機する時間を制御する、ミリ秒単位の値。デフォルト値は 10000 (10 秒) です。

この変数は、マスター側の準同期レプリケーションプラグインがインストールされている場合のみ利用できます。

- rpl_semi_sync_master_trace_level

システム変数	rpl_semi_sync_master_trace_level

スコープ	グローバル
動的	はい
型	数値
デフォルト	32

マスターの準同期レプリケーションデバッグトレースレベル。現在は 4 つのレベルが定義されています。

- 1 = 一般レベル (時間関数の失敗など)
- 16 = 詳細レベル (詳細情報)
- 32 = ネット待機レベル (ネットワーク待機についての詳細情報)
- 64 = 関数レベル (関数の入口および出口についての情報)

この変数は、マスター側の準同期レプリケーションプラグインがインストールされている場合のみ利用できます。

- rpl_semi_sync_master_wait_no_slave

システム変数	rpl_semi_sync_master_wait_no_slave
スコープ	グローバル
動的	はい
型	ブール
デフォルト	ON

準同期レプリケーションでは、各トランザクションについて、マスターはいずれかの準同期スレーブからの受け取りの認証を、タイムアウトになるまで待機します。この期間中に応答がなければ、マスターは通常のレプリケーションに戻ります。この変数は、タイムアウト期間中にスレーブカウントが減少してゼロになったとしても、マスターは通常のレプリケーションに戻る前に、タイムアウトが終了するまで待機するかどうかを制御します。

値が ON (デフォルト) の場合、タイムアウト期間中に (たとえばスレーブが未接続となった場合)、スレーブカウントが減少してゼロになるよう許可されます。マスターは引き続きタイムアウトを待機しているため、タイムアウト期間中にいずれかのスレーブが再接続してトランザクションを認証した場合、準同期レプリケーションは続行します。

値が OFF の場合、タイムアウト期間中にスレーブカウントが減少してゼロになると、マスターは通常のレプリケーションに戻ります。

この変数は、マスター側の準同期レプリケーションプラグインがインストールされている場合のみ利用できます。

- rpl_semi_sync_slave_enabled

システム変数	rpl_semi_sync_slave_enabled
スコープ	グローバル
動的	はい
型	ブール
デフォルト	OFF

準同期レプリケーションをスレーブ上で有効にするかどうかを制御します。プラグインを有効または無効にするには、この変数を ON または OFF (あるいは 1 または 0) にそれぞれ設定します。デフォルトは OFF です。

この変数は、スレーブ側の準同期レプリケーションプラグインがインストールされている場合にのみ利用できます。

- rpl_semi_sync_slave_trace_level

システム変数	rpl_semi_sync_slave_trace_level
スコープ	グローバル

動的	はい
型	数値
デフォルト	32

スレーブの準同期レプリケーションデバッグトレースレベル。許可できる値については、rpl_semi_sync_master_trace_level を参照してください。

この変数は、スレーブ側の準同期レプリケーションプラグインがインストールされている場合にのみ利用できます。

- secure_auth

コマンド行形式	--secure-auth
システム変数	secure_auth
スコープ	グローバル
動的	はい
型	ブール
デフォルト (≥ 5.6.5)	ON
デフォルト (≤ 5.6.4)	OFF

この変数が有効な場合、サーバーは、古い (4.1 以前の) 形式で格納されているパスワードを持つアカウントを使用しようとしたクライアントの接続をブロックします。

古い形式を使用したパスワードの使用 (セキュリティー保護されていないネットワーク上での通信) を防ぐには、この変数を有効にします。MySQL 5.6.5 より前では、この変数はデフォルトで無効です。MySQL 5.6.5 以降では、これはデフォルトで有効です。

この変数が有効で、権限テーブルが 4.1 以前の形式である場合、サーバーの起動が失敗します。セクション B.5.2.4「クライアントは認証プロトコルに対応できません」を参照してください。

> **注記**
>
> 4.1 より前のハッシュ方式を使用するパスワードはネイティブのパスワードハッシュ方式を使用するパスワードよりもセキュアでないため、使用しないようにしてください。4.1 よりも前のパスワードは非推奨であり、これらのサポートは今後の MySQL リリースで削除される予定です。そのため、secure_auth の無効化も非推奨です。

- secure_file_priv

コマンド行形式	--secure-file-priv=path
システム変数	secure_file_priv
スコープ	グローバル
動的	いいえ
型	文字列

デフォルトでは、この変数は空です。ディレクトリの名前に設定すると、LOAD_FILE() 関数と、LOAD DATA および SELECT ... INTO OUTFILE ステートメントの効果を制限し、そのディレクトリ内のファイルにのみ機能します。

- server_id

コマンド行形式	--server-id=#
システム変数	server_id
スコープ	グローバル
動的	はい
型	数値
デフォルト	0
最小値	0

最大値	4294967295

マスターおよびスレーブそれぞれに一意の識別子を付与するレプリケーションで使用されるサーバー ID。この変数は、--server-id オプションによって設定されます。レプリケーションに参加する各サーバーは、そのサーバーの ID となる 1 から $2^{32} - 1$ の範囲の正の整数を、選択します。

- sha256_password_private_key_path

導入	5.6.6
システム変数	sha256_password_private_key_path
スコープ	グローバル
動的	いいえ
型	ファイル名
デフォルト	private_key.pem

sha256_password 認証プラグイン用の RSA 秘密鍵ファイルのパス名。ファイル名が相対パスとして指定された場合、サーバーのデータディレクトリを基準として解釈されます。ファイルは PEM 形式である必要があります。このファイルは秘密鍵を格納しているため、MySQL Server のみがファイルを読み取りできるようにファイルのアクセスモードを制限します。

RSA 鍵ファイルの作成の説明を含む sha256_password についての情報は、セクション6.3.8.4「SHA-256 認証プラグイン」を参照してください。

この変数は、MySQL が OpenSSL を使用して構築されている場合のみ利用できます。これは MySQL 5.6.6 で追加されました。(MySQL Community Edition は yaSSL を使用して構築されています。)

- sha256_password_public_key_path

導入	5.6.6
システム変数	sha256_password_public_key_path
スコープ	グローバル
動的	いいえ
型	ファイル名
デフォルト	public_key.pem

sha256_password 認証プラグイン用の RSA 公開鍵ファイルのパス名。ファイル名が相対パスとして指定された場合、サーバーのデータディレクトリを基準として解釈されます。ファイルは PEM 形式である必要があります。このファイルは公開鍵を格納しているため、クライアントユーザーに対してコピーを自由に配布できます。(RSA パスワード暗号化を使用してサーバーに接続するときに公開鍵を明示的に指定するクライアントは、サーバーで使用されるものと同じ公開鍵を使用する必要があります。)

RSA 鍵ファイルの作成と、クライアントが RSA 公開鍵を指定する方法についての説明を含む、sha256_password についての情報は、セクション6.3.8.4「SHA-256 認証プラグイン」を参照してください。

この変数は、MySQL が OpenSSL を使用して構築されている場合のみ利用できます。これは MySQL 5.6.6 で追加されました。(MySQL Community Edition は yaSSL を使用して構築されています。)

- shared_memory

コマンド行形式	--shared-memory-base-name=name
システム変数	shared_memory
スコープ	グローバル
動的	いいえ
プラットフォーム固有	Windows

(Windows のみ。)サーバーが共有メモリー接続を許可するかどうか。

- shared_memory_base_name

システム変数	shared_memory_base_name
スコープ	グローバル
動的	いいえ
プラットフォーム固有	Windows

(Windows のみ。)共有メモリー接続に使用する共有メモリーの名前。これは、単一の物理マシン上で複数の MySQL インスタンスを実行する場合に便利です。デフォルト名は MYSQL です。名前は大文字と小文字を区別します。

- skip_external_locking

コマンド行形式	--skip-external-locking
システム変数	skip_external_locking
スコープ	グローバル
動的	いいえ
型	ブール
デフォルト	ON

これは、mysqld が外部ロック (システムロック) を使用する場合は OFF で、外部ロックが無効な場合は ON です。これは、MyISAM テーブルアクセスにのみ影響します。

この変数は、--external-locking または --skip-external-locking オプションによって設定されます。MySQL 4.0 以降では、外部ロックはデフォルトで無効化されています。

外部ロックは MyISAM テーブルアクセスにのみ影響します。使用できるまたはできない状況も含めた詳細情報については、セクション8.10.5「外部ロック」を参照してください。

- skip_name_resolve

コマンド行形式	--skip-name-resolve
システム変数	skip_name_resolve
スコープ	グローバル
動的	いいえ
型	ブール
デフォルト	OFF

この変数は、--skip-name-resolve オプションの値から設定されます。これが OFF の場合、mysqld はクライアント接続を検査するときにホスト名を解決します。これが ON の場合、mysqld は IP 番号のみを使用し、この状況では、付与テーブル内のすべての Host カラム値が IP アドレスまたは localhost である必要があります。セクション8.11.5.2「DNS ルックアップの最適化とホストキャッシュ」を参照してください。

- skip_networking

コマンド行形式	--skip-networking
システム変数	skip_networking
スコープ	グローバル
動的	いいえ

サーバーがローカル接続 (TCP/IP 以外) のみ許可する場合、これは ON です。Unix の場合、ローカル接続には Unix ソケットファイルが使用されます。Windows の場合、ローカル接続には名前付きパイプまたは共有メモリーが使用されます。この変数は、--skip-networking オプションを使用して ON に設定できます。

- skip_show_database

コマンド行形式	--skip-show-database
システム変数	skip_show_database

スコープ	グローバル
動的	いいえ

これは、SHOW DATABASES 権限を持っていないユーザーが SHOW DATABASES ステートメントを使用することを防ぎます。ほかのユーザーに属するデータベースをユーザーが表示できることに不安がある場合に、セキュリティーを高めることができます。この効果は SHOW DATABASES 権限によって異なります。変数の値が ON の場合、SHOW DATABASES ステートメントは SHOW DATABASES 権限を持つユーザーにのみ許可され、ステートメントはすべてのデータベース名を表示します。値が OFF の場合、SHOW DATABASES はすべてのユーザーに許可されますが、ユーザーが SHOW DATABASES またはほかの権限を持つデータベースの名前のみが表示されます。(すべてのグローバル権限がデータベースのための権限とみなされることに注意してください。)

- slow_launch_time

コマンド行形式	--slow-launch-time=#
システム変数	slow_launch_time
スコープ	グローバル
動的	はい
型	数値
デフォルト	2

スレッドを作成する時間がこの秒数より長くかかると、サーバーは Slow_launch_threads ステータス変数を増やします。

- slow_query_log

コマンド行形式	--slow-query-log
システム変数	slow_query_log
スコープ	グローバル
動的	はい
型	ブール
デフォルト	OFF

スロークエリーログを有効にするかどうか。値が 0 (または OFF) の場合はログを無効にし、1 (または ON) の場合はログを有効にします。デフォルト値は、--slow_query_log オプションが指定されているかどうかによって異なります。ログ出力先は log_output システム変数によって制御され、この値を NONE にした場合はログが有効になっていてもログエントリは書き込まれません。

「スロー」の程度は、long_query_time 変数の値によって決定されます。セクション5.2.5「スロークエリーログ」を参照してください。

- slow_query_log_file

コマンド行形式	--slow-query-log-file=file_name
システム変数	slow_query_log_file
スコープ	グローバル
動的	はい
型	ファイル名
デフォルト	host_name-slow.log

スロークエリーログファイルの名前。デフォルト値は host_name-slow.log ですが、初期値は --slow_query_log_file オプションを使用すると変更できます。

- socket

コマンド行形式	--socket=file_name
システム変数	socket

スコープ	グローバル
動的	いいえ
型	ファイル名
デフォルト	/tmp/mysql.sock

Unix プラットフォームでは、この変数は、ローカルクライアント接続に使用されるソケットファイルの名前です。デフォルトは /tmp/mysql.sock です。(一部の配布形式ではディレクトリが異なる場合があり、たとえば RPM の場合は /var/lib/mysql です。)

Windows では、この変数は、ローカルクライアント接続に使用される名前付きパイプの名前です。デフォルト値は MySQL です (大文字小文字の区別はありません)。

- sort_buffer_size

コマンド行形式	--sort-buffer-size=#
システム変数	sort_buffer_size
スコープ	グローバル、セッション
動的	はい
型	数値
デフォルト (その他, 64 ビットプラットフォーム, ≥ 5.6.4)	262144
デフォルト (その他, 64 ビットプラットフォーム, ≤ 5.6.3)	2097144
デフォルト (その他, 32 ビットプラットフォーム, ≥ 5.6.4)	262144
デフォルト (その他, 32 ビットプラットフォーム, ≤ 5.6.3)	2097144
デフォルト (Windows, ≥ 5.6.4)	262144
デフォルト (Windows, ≤ 5.6.3)	2097144
最小値	32768
最大値 (その他, 64 ビットプラットフォーム)	18446744073709551615
最大値 (その他, 32 ビットプラットフォーム)	4294967295
最大値 (Windows)	4294967295

ソートを実行する必要がある各セッションは、このサイズのバッファーを割り当てます。sort_buffer_size はいずれかのストレージエンジンに固有ではなく、一般的な方法で最適化に適用されます。例については、セクション8.2.1.15「ORDER BY の最適化」を参照してください。

SHOW GLOBAL STATUS の出力に表示される秒あたりの Sort_merge_passes の数が多い場合、sort_buffer_size 値を増やすことで、クエリー最適化またはインデックスの改善によって改善できない ORDER BY または GROUP BY 操作を高速化することを検討できます。

MySQL 5.6.4 以降では、オプティマイザは必要なスペースを算出しようとしますが、さらに多く上限まで割り当てることができます。MySQL 5.6.4 より前では、オプティマイザはバッファーのすべてが必要ない場合でも、バッファー全体を割り当てます。いずれの場合も、これを必要以上に大きくグローバルに設定すると、ソートを実行するほとんどのクエリーが低速化します。これはセッション設定として増やし、かつ大きいサイズを必要とするセッションに制限することを推奨します。Linux の場合、256K バイトおよび 2M バイトのしきい値があり、それより大きい値にするとメモリー割り当てが著しく低速になるため、これらのいずれかの値より低くすることを検討してください。実験して、ワークロードに最適な値を見つけてください。セクション B.5.4.4「MySQL が一時ファイルを格納する場所」を参照してください。

許可される sort_buffer_size の最大の設定値は 4G バイト − 1 です。64 ビットプラットフォームの場合は大きい値が許可されます (64 ビットの Windows の場合は例外で、大きい値は 4G バイト − 1 に切り捨てられて警告が出ます)。

- sql_auto_is_null

システム変数	sql_auto_is_null
スコープ	グローバル、セッション
動的	はい
型	ブール
デフォルト	0

この変数が 1 に設定されている場合は、自動的に生成された AUTO_INCREMENT 値を正常に挿入するステートメントの後に、次の形式のステートメントを発行すれば、その値を検索できます。

```
SELECT * FROM tbl_name WHERE auto_col IS NULL
```

ステートメントが行を返す場合、返される値は LAST_INSERT_ID() 関数を呼び出した場合と同じです。複数行の挿入後の戻り値などについての詳細は、セクション12.14「情報関数」を参照してください。AUTO_INCREMENT 値を正常に挿入できなかった場合、SELECT ステートメントは行を返しません。

IS NULL 比較を使用して AUTO_INCREMENT 値を取得する動作は、Access などの一部の ODBC プログラムによって使用されます。Obtaining Auto-Increment Valuesを参照してください。この動作は sql_auto_is_null を 0 に設定することによって無効化できます。

MySQL 5.6 では sql_auto_is_null のデフォルト値は 0 です。

- sql_big_selects

システム変数	sql_big_selects
スコープ	グローバル、セッション
動的	はい
型	ブール
デフォルト	1

0 に設定すると、MySQL は、実行に非常に時間がかかる SELECT ステートメント (つまり、調査される行数が max_join_size の値を超えるとオプティマイザが推定したステートメント) を中止します。これは、推奨されない WHERE ステートメントが発行されたときに便利です。新規接続についてのデフォルト値は 1 で、これはすべての SELECT ステートメントを許可します。

max_join_size システム変数を DEFAULT 以外の値に設定すると、sql_big_selects は 0 に設定されます。

- sql_buffer_result

システム変数	sql_buffer_result
スコープ	グローバル、セッション
動的	はい
型	ブール
デフォルト	0

1 に設定すると、sql_buffer_result は SELECT ステートメントからの結果を一時テーブルに配置するよう強制します。これは、MySQL でテーブルロックを早期に解放するのに役立ち、クライアントに結果を送信するのに長い時間がかかる場合に適していることがあります。デフォルト値は 0 です。

- sql_log_bin

システム変数	sql_log_bin
スコープ	グローバル、セッション
動的	はい
型	ブール

この変数は、バイナリログへの口ギングを実行するかどうかを制御します。デフォルト値は 1 (口ギングを実行する) です。現在のセッションの口ギングを変更するには、この変数のセッション値を変更します。この変数を設定するには、セッションユーザーが SUPER 権限を持つ必要があります。

MySQL 5.6 では、トランザクションまたはサブクエリー内に @@SESSION.sql_log_bin を設定できません。(バグ #53437)

- sql_log_off

システム変数	sql_log_off
スコープ	グローバル、セッション
動的	はい
型	ブール
デフォルト	0

この変数は、一般クエリーログへのロギングが実行されるかどうかを制御します。デフォルト値は 0 (ロギングを実行する) です。現在のセッションのロギングを変更するには、この変数のセッション値を変更します。このオプションを設定するには、セッションユーザーが SUPER 権限を持つ必要があります。デフォルト値は 0 です。

- sql_mode

コマンド行形式	--sql-mode=name
システム変数	sql_mode
スコープ	グローバル、セッション
動的	はい
型	セット
デフォルト (≥ 5.6.6)	NO_ENGINE_SUBSTITUTION
デフォルト (≤ 5.6.5)	''
有効な値	ALLOW_INVALID_DATES ANSI_QUOTES ERROR_FOR_DIVISION_BY_ZERO HIGH_NOT_PRECEDENCE IGNORE_SPACE NO_AUTO_CREATE_USER NO_AUTO_VALUE_ON_ZERO NO_BACKSLASH_ESCAPES NO_DIR_IN_CREATE NO_ENGINE_SUBSTITUTION NO_FIELD_OPTIONS NO_KEY_OPTIONS NO_TABLE_OPTIONS NO_UNSIGNED_SUBTRACTION NO_ZERO_DATE NO_ZERO_IN_DATE ONLY_FULL_GROUP_BY PAD_CHAR_TO_FULL_LENGTH PIPES_AS_CONCAT

	REAL_AS_FLOAT STRICT_ALL_TABLES STRICT_TRANS_TABLES

現在のサーバー SQL モードで、動的に設定できます。MySQL 5.6.6 以降のデフォルトは NO_ENGINE_SUBSTITUTION で、以前は空の文字列でした。セクション5.1.7「サーバー SQL モード」を参照してください。

> **注記**
>
> MySQL インストールプログラムはインストールプロセス中に SQL モードを構成することがあります。たとえば、mysql_install_db は、my.cnf という名前のデフォルトオプションファイルを基本インストールディレクトリに作成します。このファイルには、SQL モードを設定する行が含まれています。セクション4.4.3「mysql_install_db — MySQL データディレクトリの初期化」を参照してください。
>
> SQL モードがデフォルトまたは期待されているモードと異なる場合、サーバーが起動時に読み取るオプションファイル内の設定を確認してください。

- sql_notes

  1 (デフォルト) に設定した場合、Note レベルの警告によって warning_count が増加し、サーバーがこれらを記録します。0 に設定した場合、Note の警告によって warning_count は増加せず、サーバーはこれらを記録しません。mysqldump には、この変数を 0 に設定するための出力が含まれているため、ダンプファイルをリロードしても、リロード操作の整合性に影響しないイベントについて、警告が生成されません。

- sql_quote_show_create

  1 (デフォルト) に設定すると、サーバーは SHOW CREATE TABLE と SHOW CREATE DATABASE ステートメントに識別子を引用します。0 に設定した場合、引用は無効化されます。このオプションはデフォルトで有効化されているため、引用が必要な識別子に対してレプリケーションが機能します。セクション13.7.5.12「SHOW CREATE TABLE 構文」およびセクション13.7.5.8「SHOW CREATE DATABASE 構文」を参照してください。

- sql_safe_updates

  1 に設定すると、MySQL は WHERE 句または LIMIT 句の内部でキーを使用しない UPDATE または DELETE ステートメントを中止します。(特に、UPDATE ステートメントは、キーを使用する WHERE 句または LIMIT 句、あるいはその両方を持つ必要があります。DELETE ステートメントは両方を持つ必要があります。)これにより、キーが正しく利用されないため多数の行を変更または削除する可能性がある UPDATE または DELETE ステートメントのキャッチが可能になります。デフォルト値は 0 です。

- sql_select_limit

システム変数	sql_select_limit
スコープ	グローバル、セッション
動的	はい
型	数値

SELECT ステートメントから返される最大行数。新規接続についてのデフォルト値は、サーバーがテーブルあたりで許可する最大行数です。標準的なデフォルト値は $(2^{32})-1$ または $(2^{64})-1$ です。制限を変更した場合、デフォルト値は DEFAULT の値を割り当てることでリストアできます。

SELECT に LIMIT 句がある場合、LIMIT が sql_select_limit の値に優先されます。

- sql_warnings

  この変数は、警告が発生する場合に、単一行の INSERT ステートメントが情報文字列を生成するかどうかを制御します。デフォルトは 0 です。この値を 1 に設定すると、情報文字列が生成されます。

- ssl_ca

コマンド行形式	--ssl-ca=file_name

システム変数	ssl_ca
スコープ	グローバル
動的	いいえ
型	ファイル名

信頼された SSL CA のリストを含むファイルへのパス。

- ssl_capath

コマンド行形式	--ssl-capath=dir_name
システム変数	ssl_capath
スコープ	グローバル
動的	いいえ
型	ディレクトリ名

PEM 形式の信頼された SSL CA 証明書を格納するディレクトリのパス。

- ssl_cert

コマンド行形式	--ssl-cert=file_name
システム変数	ssl_cert
スコープ	グローバル
動的	いいえ
型	ファイル名

セキュアな接続を確立するために使用する SSL 証明書ファイルの名前。

- ssl_cipher

コマンド行形式	--ssl-cipher=name
システム変数	ssl_cipher
スコープ	グローバル
動的	いいえ
型	文字列

SSL 暗号化に使用する許可されている暗号のリスト。

- ssl_crl

コマンド行形式	--ssl-crl=file_name
導入	5.6.3
システム変数	ssl_crl
スコープ	グローバル
動的	いいえ
型	ファイル名

PEM 形式での証明書失効リストを含むファイルへのパス。失効リストは、OpenSSL に対してコンパイルされた MySQL 配布で機能します (yaSSL では機能しません)。

この変数は MySQL 5.6.3 で追加されました。

- ssl_crlpath

コマンド行形式	--ssl-crlpath=dir_name
導入	5.6.3
システム変数	ssl_crlpath

スコープ	グローバル
動的	いいえ
型	ディレクトリ名

PEM 形式での証明書失効リストを含むファイルを格納するディレクトリへのパス。失効リストは、OpenSSL に対してコンパイルされた MySQL 配布で機能します (yaSSL では機能しません)。

この変数は MySQL 5.6.3 で追加されました。

- ssl_key

コマンド行形式	--ssl-key=file_name
システム変数	ssl_key
スコープ	グローバル
動的	いいえ
型	ファイル名

セキュアな接続を確立するために使用する SSL 鍵ファイルの名前。

- storage_engine

システム変数	storage_engine
スコープ	グローバル、セッション
動的	はい
型	列挙
デフォルト	InnoDB

デフォルトストレージエンジン (テーブル型)。サーバー起動時にストレージエンジンを設定するには、--default-storage-engine オプションを使用します。セクション5.1.3「サーバーコマンドオプション」を参照してください。

この変数は非推奨です。代わりに default_storage_engine を使用してください。

- stored_program_cache

コマンド行形式	--stored-program-cache=#
導入	5.6.5
システム変数	stored_program_cache
スコープ	グローバル
動的	はい
型	数値
デフォルト	256
最小値	256
最大値	524288

接続あたりでキャッシュされるストアドルーチンの数について、上側のソフトリミットを設定します。この変数の値は、ストアドプロシージャーおよびストアドファンクションで、MySQL Server によって維持される 2 つのキャッシュそれぞれに保持されるストアドルーチンの数に関して指定します。

ストアドルーチンが実行されると、ルーチン内の先頭または最上位レベルのステートメントが解析される前に、このキャッシュサイズが検査されます。同じタイプのルーチン (どちらが実行されているかによってストアドプロシージャーまたはストアドファンクション) の数が、この変数によって指定される制限を超える場合、対応するキャッシュがフラッシュされ、キャッシュされたオブジェクトに対して以前割り当てられていたメモリーが解放されます。これにより、ストアドルーチン間に依存関係がある場合でも、キャッシュを安全にフラッシュできます。

- sync_frm

コマンド行形式	--sync-frm
システム変数	sync_frm
スコープ	グローバル
動的	はい
型	ブール
デフォルト	TRUE

この変数が 1 に設定された場合、一時テーブル以外のテーブルが作成されると、その .frm ファイルは (fdatasync() を使用して) ディスクに同期されます。これは遅いですが、クラッシュした場合はより安全です。デフォルトは 1 です。

- system_time_zone

システム変数	system_time_zone
スコープ	グローバル
動的	いいえ
型	文字列

サーバーシステムタイムゾーン。サーバーは実行を開始するとき、マシンのデフォルトからタイムゾーン設定を継承し、サーバーを実行するために使用されるアカウントの環境または起動スクリプトによって変更されることがあります。値は system_time_zone を設定するために使用されます。通常、タイムゾーンは TZ 環境変数で指定されます。または mysqld_safe スクリプトの --timezone オプションを使用しても指定できます。

system_time_zone 変数は time_zone と異なります。これらは同じ値になることもありますが、後者の変数は、接続する各クライアントのタイムゾーンを初期化するために使用されます。セクション10.6「MySQL Server でのタイムゾーンのサポート」を参照してください。

- table_definition_cache

システム変数	table_definition_cache
スコープ	グローバル
動的	はい
型	数値
デフォルト (≥ 5.6.8)	-1 (autosized)
デフォルト (≤ 5.6.7)	400
最小値	400
最大値	524288

定義キャッシュに格納可能な (.frm ファイルからの) テーブル定義の数。多数のテーブルを使用する場合、大きいテーブル定義キャッシュを作成して、テーブルを開くことを高速化できます。標準のテーブルキャッシュと異なり、テーブル定義キャッシュは占有スペースが少なくファイルディスクリプタを使用しません。最小値は 400 です。デフォルト値は次の式に基づき、2000 までに制限されています。

```
400 + (table_open_cache / 2)
```

MySQL 5.6.8 より前では、デフォルトは 400 です。

InnoDB の場合、table_definition_cache は、InnoDB データディクショナリキャッシュ内の開いているテーブルインスタンスの数のソフト制限として機能します。開いているテーブルインスタンスの数が table_definition_cache 設定を超えた場合、LRU メカニズムはエビクション用のテーブルインスタンスにマークを付け、最終的にデータディクショナリキャッシュから削除されます。この制限は、次回サーバー開始までに使用頻度が低いテーブルインスタンスをキャッシュするために大量のメモリーが使用されるような状況に対処するのに役立ちます。キャッシュされたメタデータを持つテーブルインスタンスの数が、table_definition_cache によって定義された制限よりも多い場合があります。これは、InnoDB システムテー

ブルインスタンスと、外部キー関係を持つ親および子のテーブルインスタンスが LRU リストに配置されず、メモリーから エビクションされないためです。

さらに、table_definition_cache は、一度に開くことができる、InnoDB file-per-table テーブルスペースの数のソフト制限を定義し、これは innodb_open_files によっても制御されます。table_definition_cache および innodb_open_files の両方が設定される場合、高い方の設定値が使用されます。どちらの変数も設定されない場合、デフォルト値が高い table_definition_cache が使用されます。オープンテーブルスペースファイルハンドルの数が、table_definition_cache または innodb_open_files によって定義された制限を超える場合、LRU メカニズムは、テーブルスペースファイル LRU リストを検索して、完全にフラッシュされて現在延長されていないファイルを探します。この処理は、新しいテーブルスペースが開くたびに実行されます。「非アクティブな」テーブルスペースがない場合、テーブルスペースファイルはクローズされません。

- table_open_cache

システム変数	table_open_cache
スコープ	グローバル
動的	はい
型	数値
デフォルト (≥ 5.6.8)	2000
デフォルト (≤ 5.6.7)	400
最小値	1
最大値	524288

すべてのスレッドについて開いているテーブルの数。この値を大きくすると、mysqld が要求するファイルディスクリプタの数が増加します。Opened_tables ステータス変数を検査して、テーブルキャッシュを増やす必要があるかどうかを確認できます。セクション5.1.6「サーバーステータス変数」を参照してください。Opened_tables の値が大きく、FLUSH TABLES をあまり使用しない場合 (すべてのテーブルのクローズおよび再オープンの強制のみを実行する)、table_open_cache 変数の値を増やします。テーブルキャッシュに関する詳細は、セクション8.4.3.1「MySQL でのテーブルのオープンとクローズの方法」を参照してください。

- table_open_cache_instances

導入	5.6.6
システム変数	table_open_cache_instances
スコープ	グローバル
動的	いいえ
型	数値
デフォルト	1

開いているテーブルキャッシュインスタンスの数 (デフォルトは 1)。セッション間の競合を減少させることでスケーラビリティーを改善するために、開いているテーブルキャッシュを、サイズが table_open_cache / table_open_cache_instances のいくつかの小さいキャッシュインスタンスにパーティション化できます。DML ステートメントでは、セッションはインスタンスにアクセスするために、1 つのインスタンスのみロックする必要があります。このセグメントキャッシュは複数インスタンスにわたってアクセスし、多くのセッションがテーブルにアクセスする場合にキャッシュを使用する演算の高いパフォーマンスが可能になります。(DDL ステートメントでは引き続きキャッシュ全体のロックが必要ですが、そのようなステートメントは DML ステートメントよりも頻度がずっと低くなります。)

通常 16 以上のコアを使用するシステムでは、8 または 16 の値が推奨されます。

この変数は MySQL 5.6.6 で追加されました。

- thread_cache_size

コマンド行形式	--thread-cache-size=#
システム変数	thread_cache_size
スコープ	グローバル
動的	はい
型	数値

デフォルト (≥ 5.6.8)	-1 (autosized)
デフォルト (≤ 5.6.7)	0
最小値	0
最大値	16384

サーバーが再使用のためにキャッシュするスレッドの数。クライアントが接続を切断したとき、スレッド数が thread_cache_size より少なければ、クライアントのスレッドはキャッシュに配置されます。スレッドのリクエストは、可能であれば、キャッシュからのスレッドを再使用することによって満たされ、キャッシュが空の場合のみ新しいスレッドが作成されます。多くの新しい接続がある場合、この変数を増やしてパフォーマンスを向上できます。スレッドの実装が適切な場合、通常はパフォーマンスが著しく改善されることはありません。ただし、1 秒あたり数百件の接続がサーバーで見られる場合、通常は thread_cache_size を十分に高く設定すると、ほとんどの新しい接続でキャッシュされたスレッドを使用できます。ステータス変数 Connections と Threads_created の差異を調査することで、スレッドキャッシュの効率性を確認できます。詳細については、セクション5.1.6「サーバーステータス変数」を参照してください。

デフォルト値は次の式に基づいており、上限は 100 に制限されています。

```
8 + (max_connections / 100)
```

MySQL 5.6.8 より前では、デフォルトは 0 です。

- thread_concurrency

コマンド行形式	--thread-concurrency=#
非推奨	5.6.1
システム変数	thread_concurrency
スコープ	グローバル
動的	いいえ
型	数値
デフォルト	10
最小値	1
最大値	512

この変数は Solaris 8 以前のシステムに固有のもので、mysqld はこのシステムに対し、変数値を指定して thr_setconcurrency() 関数を呼び出します。この関数によって、アプリケーションは、同時に実行するのが望ましいスレッド数についてのヒントをスレッドシステムに指示できます。現在の Solaris バージョンのドキュメントには、これは効果がないと記載されています。

この変数は MySQL 5.6.1 以降で非推奨となり、MySQL 5.7 で削除されています。これを見つけたときは、Solaris 8 以前のためのものでないかぎり、MySQL 構成ファイルから削除してください。

- thread_handling

コマンド行形式	--thread-handling=name
システム変数	thread_handling
スコープ	グローバル
動的	いいえ
型	列挙
デフォルト	one-thread-per-connection
有効な値	no-threads one-thread-per-connection dynamically-loaded

接続スレッドのサーバーによって使用されるスレッド処理モデル。許可される値は no-threads (サーバーは 1 つの接続を処理するために単一スレッドを使用する) および one-thread-per-connection (サーバーはそれぞれのク

ライアント接続を処理するために 1 つのスレッドを使用する) です。Linux でのデバッグには no-threads が便利です。セクション24.4「MySQL のデバッグおよび移植」を参照してください。

スレッドプールプラグインが有効な場合、サーバーは thread_handling 値を dynamically-loaded に設定します。セクション8.11.6.1「スレッドプールコンポーネントとインストール」を参照してください。

- thread_pool_algorithm

コマンド行形式	--thread-pool-algorithm=#
導入	5.6.10
システム変数	thread_pool_algorithm
スコープ	グローバル
動的	いいえ
型	数値
デフォルト	0
最小値	0
最大値	1

この変数は、スレッドプールプラグインが使用するアルゴリズムを制御します。

- 値 0 (デフォルト) では、並列性の低い保守的なアルゴリズムが使用されます。これはもっとも良く検査されていて、非常に良好な結果を生成することが知られています。
- 値 1 では並列性が高まり、より積極的なアルゴリズムが使用されます。このアルゴリズムは、最適なスレッドカウントでパフォーマンスが 5 – 10% 高まりますが、接続数が増えるにつれてパフォーマンスが低下することが知られています。この使用は実験的であり、サポートされないものとみなすようにしてください。

  この変数は MySQL 5.6.10 で追加されました。これはスレッドプールプラグインが有効な場合のみ使用できます。セクション8.11.6「スレッドプールプラグイン」を参照してください

- thread_pool_high_priority_connection

コマンド行形式	--thread-pool-high-priority-connection=#
導入	5.6.10
システム変数	thread_pool_high_priority_connection
スコープ	グローバル、セッション
動的	はい
型	数値
デフォルト	0
最小値	0
最大値	1

この変数は、実行前の新規ステートメントのキューイングに影響します。値が 0 (false、デフォルト) の場合、ステートメントのキューイングでは優先度の低いキューと優先度の高いキューの両方が使用されます。値が 1 (true) の場合、キューに入れられるステートメントは常に優先度の高いキューに入ります。

この変数は MySQL 5.6.10 で追加されました。これはスレッドプールプラグインが有効な場合のみ使用できます。セクション8.11.6「スレッドプールプラグイン」を参照してください

- thread_pool_max_unused_threads

コマンド行形式	--thread-pool-max-unused-threads=#
導入	5.6.10
システム変数	thread_pool_max_unused_threads
スコープ	グローバル
動的	はい
型	数値

デフォルト	0
最小値	0
最大値	4096

スレッドプール内で許可される最大の未使用スレッド数。この変数により、スリープ状態のスレッドによって使用されるメモリーの量を制限できます。

値 0 (デフォルト) は、スリープ状態のスレッドの数を制限しないことを意味します。値 N は、N が 0 より大きい場合、1 つのコンシューマスレッドと、N−1 個の予約スレッドを意味します。この状況で、スレッドがスリープ状態に入ろうとしたが、スリープ状態のスレッド数がすでに最大値に到達している場合、スレッドはスリープ状態に入らずに存在します。

スリープ状態のスレッドは、コンシューマスレッドまたは予約スレッドのいずれかとしてスリープ状態になります。スレッドプールでは、1 つのスレッドがスリープ状態のコンシューマスレッドであることを許可します。あるスレッドがスリープ状態になり、コンシューマスレッドが存在しない場合、そのスレッドはコンシューマスレッドとしてスリープ状態になります。スレッドをウェイクアップさせる必要があるとき、コンシューマスレッドが存在すれば、そのコンシューマスレッドが選択されます。ウェイクアップするコンシューマスレッドがない場合にのみ予約スレッドが選択されます。

この変数は MySQL 5.6.10 で追加されました。これはスレッドプールプラグインが有効な場合のみ使用できます。セクション8.11.6「スレッドプールプラグイン」を参照してください

- thread_pool_prio_kickup_timer

コマンド行形式	--thread-pool-prio-kickup-timer=#
導入	5.6.10
システム変数	thread_pool_prio_kickup_timer
スコープ	グローバル、セッション
動的	はい
型	数値
デフォルト	1000
最小値	0
最大値	4294967294

この変数は、優先度が低いキューで実行を待機するステートメントに影響します。この値は、待機中のステートメントが優先度の高いキューに移されるまでのミリ秒数です。デフォルトは 1000 (1 秒) です。値の範囲は 0 から $2^{32} - 2$ です。

この変数は MySQL 5.6.10 で追加されました。これはスレッドプールプラグインが有効な場合のみ使用できます。セクション8.11.6「スレッドプールプラグイン」を参照してください

- thread_pool_size

コマンド行形式	--thread-pool-size=#
導入	5.6.10
システム変数	thread_pool_size
スコープ	グローバル
動的	いいえ
型	数値
デフォルト	16
最小値	1
最大値	64

スレッドプール内のスレッドグループの数。これはスレッドプールのパフォーマンスを制御するもっとも重要なパラメータです。同時に実行できるステートメントの数に影響します。デフォルト値は 16 で、許可される値

の範囲は 1 から 64 です。この範囲の外側の値が指定された場合、スレッドプールプラグインはロードされず、サーバーはエラーログにメッセージを書き込みます。

この変数は MySQL 5.6.10 で追加されました。これはスレッドプールプラグインが有効な場合のみ使用できます。セクション8.11.6「スレッドプールプラグイン」を参照してください

- thread_pool_stall_limit

コマンド行形式	--thread-pool-stall-limit=#
導入	5.6.10
システム変数	thread_pool_stall_limit
スコープ	グローバル
動的	はい
型	数値
デフォルト	6
最小値	4
最大値	600

この変数はステートメントの実行に影響します。この値は、ステートメントが実行を開始したあと、ステートメントが停滞していると定義される前に終了する時間量で、その時点で、スレッドプールはスレッドグループは別のステートメントの実行の開始を許可します。この値は 10 ミリ秒単位で測定されるため、値 6 (デフォルト) は 60 ミリ秒を意味します。値の範囲は 4 から 600 (40 ミリ秒から 6 秒) です。待機の値が短いと、スレッドはよりすみやかに開始できます。短い値はデッドロック状況を回避により適しています。長い待機の値は、長時間実行するステートメントを含むワークロードで有用で、現在のステートメントの実行時に多数の新しいステートメントが開始しないようにします。

この変数は MySQL 5.6.10 で追加されました。これはスレッドプールプラグインが有効な場合のみ使用できます。セクション8.11.6「スレッドプールプラグイン」を参照してください

- thread_stack

コマンド行形式	--thread-stack=#
システム変数	thread_stack
スコープ	グローバル
動的	いいえ
型	数値
デフォルト (64 ビットプラットフォーム)	262144
デフォルト (32 ビットプラットフォーム)	196608
最小値	131072
最大値 (64 ビットプラットフォーム)	18446744073709551615
最大値 (32 ビットプラットフォーム)	4294967295
ブロックサイズ	1024

各スレッドのスタックサイズ。crash-me テストによって検出される制限の多くは、この値に依存します。セクション8.12.2「MySQL ベンチマークスイート」を参照してください。デフォルトの 192K バイト (64 ビットシステムの場合は 256K バイト) は、通常の操作では十分な大きさです。スレッドスタックサイズが小さすぎると、サーバーで処理できる SQL ステートメントの複雑さ、ストアドプロシージャーの再帰の深さなど、メモリーを大量に消費する処理が制限されます。

- time_format

この変数は使用されません。これは MySQL 5.6.7 以降で非推奨となり、今後の MySQL リリースで削除されます。

- time_zone

システム変数	time_zone
スコープ	グローバル、セッション
動的	はい
型	文字列

現在のタイムゾーン。この変数は、接続する各クライアントのタイムゾーンを初期化するために使用されます。デフォルトでは、この初期値は 'SYSTEM' です (「system_time_zone の値を使用する」ことを意味します)。この値はサーバー起動時に --default-time-zone オプションで明示的に指定できます。セクション 10.6「MySQL Server でのタイムゾーンのサポート」を参照してください。

- timed_mutexes

コマンド行形式	--timed-mutexes
非推奨	5.6.20
システム変数	timed_mutexes
スコープ	グローバル
動的	はい
型	ブール
デフォルト	OFF

MySQL 5.6 で、この変数は非推奨になり、必要ありません。これは今後の MySQL リリースで削除されます。

- timestamp

システム変数	timestamp
スコープ	セッション
動的	はい
型	数値

このクライアントの時間を設定します。これはバイナリログを使用して行をリストアする場合に元のタイムスタンプを取得するために使用されます。timestamp_value は Unix エポックタイムスタンプ (UNIX_TIMESTAMP() で返されるような値で、'YYYY-MM-DD hh:mm:ss' 形式の値ではありません) または DEFAULT になります。

timestamp を定数値に設定すると、ふたたび変更されるまでその値が保持されます。timestamp を DEFAULT に設定すると、その値はアクセスを受けた時点での現在の日付および時間になります。

MySQL 5.6.4 以降では、timestamp は BIGINT でなく DOUBLE で、これはこの値がマイクロ秒部分を含むためです。

SET timestamp は NOW() によって戻された値に影響を及ぼしますが、SYSDATE() によって戻された値には影響しません。つまり、バイナリログのタイムスタンプ設定は、SYSDATE() の呼び出しに影響しないことを意味します。サーバーを --sysdate-is-now オプションで開始して、SYSDATE() を NOW() のエイリアスにでき、この場合 SET timestamp が両方の関数に影響します。

- tmp_table_size

コマンド行形式	--tmp-table-size=#
システム変数	tmp_table_size
スコープ	グローバル、セッション
動的	はい
型	数値
デフォルト	16777216
最小値	1024

最大値	18446744073709551615

内部インメモリーの一時テーブルの最大サイズ。(実際の制限値は tmp_table_size と max_heap_table_size の最小値として決定されます。)インメモリーの一時テーブルが制限値を超えると、MySQL はこれを自動的にディスク上の MyISAM テーブルにします。多数の高度な GROUP BY クエリーを実行する場合にメモリーが多くあるときは、tmp_table_size (さらに必要に応じて max_heap_table_size) の値を増やします。この変数はユーザーが作成した MEMORY テーブルには適用されません。

Created_tmp_disk_tables 変数と Created_tmp_tables 変数の値を比較することによって、作成された内部のディスク上の一時テーブル数と、作成された内部の一時テーブルの総数を比較できます。

セクション8.4.4「MySQL が内部一時テーブルを使用する仕組み」も参照してください。

- tmpdir

コマンド行形式	--tmpdir=path
システム変数	tmpdir
スコープ	グローバル
動的	いいえ
型	ディレクトリ名

一時ファイルおよび一時テーブル用に使用されるディレクトリ。この変数は、ラウンドロビン方式で使用されるいくつかのパスのリストとして設定できます。パスは UNIX ではコロン文字 (「:」)、Windows ではセミコロン文字 (「;」) で区切るようにしてください。

複数ディレクトリ機能を使用すると、いくつかの物理ディスクに負荷を分散できます。MySQL Server がレプリケーションスレーブとして動作する場合、tmpdir を、メモリーベースのファイルシステム上のディレクトリや、サーバーホストが再起動したときにクリアされるディレクトリに指定するように設定しないでください。レプリケーションスレーブは、一部の一時ファイルがマシンの再起動後も存続し、一時テーブルまたは LOAD DATA INFILE 操作を複製できるようにする必要があります。サーバーが再起動したときに一時ファイルディレクトリ内のファイルが消失した場合、レプリケーションは失敗します。slave_load_tmpdir 変数を使用すると、スレーブの一時ディレクトリを設定できます。その場合、スレーブは一般的な tmpdir 値を使用せず、tmpdir を非永続的な場所に設定できます。

- transaction_alloc_block_size

コマンド行形式	--transaction-alloc-block-size=#
システム変数	transaction_alloc_block_size
スコープ	グローバル、セッション
動的	はい
型	数値
デフォルト	8192
最小値	1024
最大値 (64 ビットプラットフォーム)	18446744073709551615
最大値 (32 ビットプラットフォーム)	4294967295
ブロックサイズ	1024

メモリーを必要とするトランザクションごとのメモリープールを増やす、バイト単位の増加量。transaction_prealloc_size の説明を参照してください。

- transaction_prealloc_size

コマンド行形式	--transaction-prealloc-size=#
システム変数	transaction_prealloc_size
スコープ	グローバル、セッション
動的	はい

型	数値
デフォルト	4096
最小値	1024
最大値 (64 ビットプラットフォーム)	18446744073709551615
最大値 (32 ビットプラットフォーム)	4294967295
ブロックサイズ	1024

トランザクションに関するさまざまな割り当てでメモリーの取得元となる、トランザクションごとのメモリープールが存在します。プールのバイト単位の初期サイズは transaction_prealloc_size です。利用できるメモリーが不足しているためプールから十分に行えない各割り当てについて、プールは transaction_alloc_block_size バイトだけ増加されます。トランザクションが終了すると、プールは transaction_prealloc_size バイトに切り捨てられます。

単一トランザクション内のすべてのステートメントを含めるように transaction_prealloc_size を十分に大きくすると、多数の malloc() コールを避けることができます。

- tx_isolation

システム変数	tx_isolation
スコープ	グローバル、セッション
動的	はい
型	列挙
デフォルト	REPEATABLE-READ
有効な値	READ-UNCOMMITTED READ-COMMITTED REPEATABLE-READ SERIALIZABLE

デフォルトのトランザクション分離レベル。デフォルト値は REPEATABLE-READ。

この変数は直接設定することも、SET TRANSACTION ステートメントなどを使用して間接的に設定することもできます。セクション13.3.6「SET TRANSACTION 構文」を参照してください。スペースを含む分離レベル名に tx_isolation を直接設定する場合、名前を引用符記号で囲み、スペースをダッシュで置き換えるようにします。例:

```
SET tx_isolation = 'READ-COMMITTED';
```

この変数の値を設定するために、一意となる任意の有効な値のプリフィクスを使用できます。

デフォルトのトランザクション分離レベルは --transaction-isolation サーバーオプションを使用すると、起動時にも設定できます。

- tx_read_only

導入	5.6.5
システム変数	tx_read_only
スコープ	グローバル、セッション
動的	はい
型	ブール

デフォルト	OFF

デフォルトのトランザクションアクセスモード。値は OFF (読み取り/書き込み、デフォルト) または ON (読み取り専用) を指定できます。

この変数は直接設定することも、SET TRANSACTION ステートメントなどを使用して間接的に設定することもできます。セクション13.3.6「SET TRANSACTION 構文」を参照してください。

デフォルトのトランザクションアクセスモードを起動時に設定するには、--transaction-read-only サーバーオプションを使用します。

この変数は MySQL 5.6.5 で追加されました。

- unique_checks

システム変数	unique_checks
スコープ	グローバル、セッション
動的	はい
型	ブール
デフォルト	1

1 に設定した場合 (デフォルト)、InnoDB テーブルのセカンダリインデックスの一意性チェックが行われます。0 に設定した場合、ストレージエンジンでは、重複したキーが入力データに存在しないことが想定されます。一意性違反がデータにないことが確実にわかっている場合、これを 0 に設定して InnoDB への大きいテーブルのインポートを高速化できます。

この変数を 0 に設定しても、ストレージエンジンが重複キーを無視する必要があるわけではありません。エンジンは引き続き、重複キーの存在を検査し、検出された場合に重複キーエラーが生成されます。

- updatable_views_with_limit

コマンド行形式	--updatable-views-with-limit=#
システム変数	updatable_views_with_limit
スコープ	グローバル、セッション
動的	はい
型	ブール
デフォルト	1

この変数は、基礎テーブルで定義した主キーのすべてのカラムがビューに含まれていない場合に、更新ステートメントに LIMIT 句が含まれているとき、そのビューの更新を行えるかどうかを制御します。(このような更新は GUI ツールによって頻繁に生成されます。)更新は UPDATE または DELETE ステートメントのことです。ここでの主キーとは PRIMARY KEY か、カラムに NULL を含むことができない UNIQUE インデックスです。

変数は 2 つの値に設定できます。

- 1 または YES: 警告のみ発行します (エラーメッセージではない)。これはデフォルト値です。
- 0 または NO: 更新を禁止します。

- validate_password_xxx

validate_password プラグインは、validate_password_xxx 形式の名前を持つシステム変数のセットを実装します。これらの変数は、そのプラグインによるパスワードテストの影響を受けます。パスワード検証プラグインのオプションおよび変数を参照してください。

- validate_user_plugins

導入	5.6.11
システム変数	validate_user_plugins
スコープ	グローバル
動的	いいえ

型	ブール
デフォルト	ON

この変数が有効な場合 (デフォルト)、サーバーは各ユーザーアカウントを検査し、アカウントが使用できなくなる条件が検出された場合に警告を生成します。

- アカウントが、ロードされていない認証プラグインを必要としている。
- アカウントは sha256_password 認証プラグインを必要としているが、このプラグインによって必要とされる SSL および RSA をいずれも有効化しないでサーバーが開始された。

validate_user_plugins を有効にすると、サーバー初期化および FLUSH PRIVILEGES の速度が低下します。追加の検査が必要ない場合、この変数を起動時に無効化するとパフォーマンス低下を防ぐことができます。

この変数は MySQL 5.6.11 で追加されました。

- version

サーバーのバージョン番号。値には、サーバーの構築情報または構成情報を示すサフィクスを含めることも可能です。-log は、 1 つ以上の一般ログ、スロークエリーログ、またはバイナリログを有効化できることを示します。-debug は、デバッグサポートを有効にしてサーバーが構築されたことを示します。

- version_comment

システム変数	version_comment
スコープ	グローバル
動的	いいえ
型	文字列

CMake 構成プログラムには、MySQL の構築時にコメントの指定を可能にする COMPILATION_COMMENT オプションがあります。この変数は、そのコメントの値を格納します。セクション2.9.4「MySQL ソース構成オプション」を参照してください。

- version_compile_machine

システム変数	version_compile_machine
スコープ	グローバル
動的	いいえ
型	文字列

サーバーバイナリのタイプ。

- version_compile_os

システム変数	version_compile_os
スコープ	グローバル
動的	いいえ
型	文字列

MySQL が構築されているオペレーティングシステムのタイプ。

- wait_timeout

コマンド行形式	--wait-timeout=#
システム変数	wait_timeout
スコープ	グローバル、セッション
動的	はい
型	数値
デフォルト	28800

最小値	1
最大値 (その他)	31536000
最大値 (Windows)	2147483

非インタラクティブな接続を閉じる前に、サーバーがその接続上でアクティビティーを待機する秒数。

スレッド開始時に、セッションの wait_timeout 値は、wait_timeout グローバル値または interactive_timeout グローバル値で初期化されますが、いずれになるかはクライアントのタイプ (mysql_real_connect() に対する CLIENT_INTERACTIVE 接続オプションによって定義される) によって決まります。interactive_timeoutも参照してください。

- warning_count

  メッセージを生成した最後のステートメントから得られたエラー、警告、および注意の数。この変数は読み取り専用です。セクション13.7.5.41「SHOW WARNINGS 構文」を参照してください。

5.1.5 システム変数の使用

MySQL Server には、その構成方法を指示する多くのシステム変数が保持されています。セクション5.1.4「サーバーシステム変数」に、これらの変数の意味が記載されています。各システム変数にはデフォルト値があります。システム変数は、コマンド行のオプションを使用するか、オプションファイルでサーバー起動時に設定できます。これらのほとんどは、SET ステートメントを使用してサーバーの実行中に動的に変更でき、これによりサーバーを停止して再起動することなくサーバーの動作を変更できます。システム変数値を式によって参照できます。

サーバーには 2 種類のシステム変数が保持されています。グローバル変数は、サーバーの操作全体に影響します。セッション変数は、個々のクライアント接続の操作に影響します。所定のシステム変数は、グローバル値とセッション値の両方を持つことができます。グローバルシステム変数とセッションシステム変数は、次のように関連しています。

- サーバーが開始すると、サーバーはすべてのグローバル変数をデフォルト値に初期化します。これらのデフォルトは、コマンド行で指定されるオプションまたはオプションファイルで変更できます。(セクション4.2.3「プログラムオプションの指定」を参照してください。)
- サーバーは、接続する各クライアントのセッション変数のセットも保持しています。クライアントのセッション変数は、対応するグローバル変数の現在の値を使用して、接続時に初期化されます。たとえば、クライアントの SQL モードはセッション sql_mode 値で制御されますが、この値はクライアントが sql_mode のグローバル値に接続するときに初期化されます。

システム変数値は、コマンド行のオプションまたはオプションファイルを使用すると、サーバー起動時にグローバルに設定できます。起動オプションを使用して、数値を取る値を設定するとき、値には 1024、1024^2 または 1024^3 の倍数を示す、K、M、または G のサフィクス (大文字あるは小文字) を付けて指定でき、それぞれがキロバイト、メガバイト、またはギガバイトの単位を示します。これにより、次のコマンドは 16M バイトのクエリーキャッシュサイズと、1G バイトの最大パケットサイズでサーバーを開始します。

```
mysqld --query_cache_size=16M --max_allowed_packet=1G
```

オプションファイル内で、これらの変数は次のように設定されます。

```
[mysqld]
query_cache_size=16M
max_allowed_packet=1G
```

サフィクスの大文字、小文字の区別は問わず、16M と 16m、1G と 1g を同等とします。

SET ステートメントを使用して、実行時にシステム変数に設定できる最大値を制限するには、サーバー起動時に --maximum-var_name=value の形式のオプションを使用すると、この最大値を指定できます。たとえば、query_cache_size の値が実行時に 32M バイトを超えないようにするには、オプション --maximum-query_cache_size=32M を使用します。

多くのシステム変数は動的であり、SET ステートメントを利用してサーバーが実行している間に変更できます。リストついては、セクション5.1.5.2「動的システム変数」を参照してください。SET を利用してシステム変数を変更するには、任意で修飾子が先行する var_name としてシステム変数を参照してください。

- 変数がグローバル変数であることを明示的に指示するためには、その名前の前に GLOBAL または @@GLOBAL. を付けます。グローバル変数を設定するには SUPER 権限が必要です。

- 変数がセッション変数であることを明示的に指示するには、その名前の前に SESSION、@@SESSION.、または @@ を付けます。セッション変数を設定するのに特別な権限は必要ありませんが、クライアントは自分自身のセッション変数のみ変更でき、別のクライアントの変数は変更できません。
- LOCAL と @@LOCAL. は SESSION と @@SESSION. のシノニムです。
- 修飾子が何もなければ、SET はセッション変数を変更します。

SET ステートメントは、カンマで区切られた複数の変数割り当てを含むことができます。複数のシステム変数を設定した場合、ステートメント内のいちばん最近の GLOBAL または SESSION 修飾子が、指定された修飾子を持たない後続の変数に利用されます。

例:

```
SET sort_buffer_size=10000;
SET @@LOCAL.sort_buffer_size=10000;
SET GLOBAL sort_buffer_size=1000000, SESSION sort_buffer_size=1000000;
SET @@sort_buffer_size=1000000;
SET @@GLOBAL.sort_buffer_size=1000000, @@LOCAL.sort_buffer_size=1000000;
```

システム変数の @@var_name 構文では、ほかの一部のデータベースシステムとの互換性をサポートしています。

セッションシステム変数を変更すると、セッションが終了するまで、または変数を異なる値に変更するまではその値は有効になります。別のクライアントは変更を見ることができません。

グローバルシステム変数を変更すると、その値はサーバーが再起動するまでの間記憶され、新しい接続に利用されます。(グローバルシステム変数を永続的に設定するには、オプションファイルに設定する必要があります。) そのグローバル変数にアクセスするすべてのクライアントが変更を確認できます。ただし変更は、変更後に接続するクライアントの対応するセッション変数にのみ影響を与えます。グローバル変数の変更は、現在接続中のクライアントのセッション変数に影響を与えません (SET GLOBAL ステートメントを発行するクライアントのセッション変数にも影響を与えません)。

誤用を防ぐために、SET SESSION とのみ利用できる変数とともに SET GLOBAL を使用したり、グローバル変数の設定時に GLOBAL (または @@GLOBAL.) を指定しなかったりした場合に、MySQL でエラーが生じます。

SESSION 変数を GLOBAL 値に設定したり、GLOBAL 値をコンパイル時の MySQL のデフォルト値に設定したりするには、DEFAULT キーワードを使用します。たとえば、次の 2 つのステートメントは、max_join_size のセッション値をグローバル値に設定する上で同一です。

```
SET max_join_size=DEFAULT;
SET @@SESSION.max_join_size=@@GLOBAL.max_join_size;
```

すべてのシステム変数を DEFAULT に設定できるわけではありません。そのような場合、DEFAULT を使用すると エラーが発生します。

いずれかの @@ 修飾子を使用することによって、特定のグローバルシステム変数またはセッションシステム変数の値を式で参照できます。たとえば、次のようにして SELECT ステートメントで値を取得できます。

```
SELECT @@GLOBAL.sql_mode, @@SESSION.sql_mode, @@sql_mode;
```

@@var_name のような式でシステム変数を参照するとき (つまり、@@GLOBAL. または @@SESSION. を指定しない場合)、MySQL はセッション値が存在すればそれを返し、それ以外の場合はグローバル値を返します。(これは、常にセッション値を参照する SET @@var_name = value とは異なります。)

> **注記**
>
> SHOW VARIABLES によって表示される一部の変数は、SELECT @@var_name 構文で使用できない場合があり、「不明なシステム変数です」と表示されます。その場合の回避方法として、SHOW VARIABLES LIKE 'var_name' を使用できます。

値乗数を指定するサフィクスは、サーバーの起動時に変数を設定するときに使用できますが、実行時に SET で値を設定するためには使用できません。一方、SET を使用すると、式を使用して変数の値を割り当てることができますが、サーバーの起動時に変数を設定するときには使用できません。たとえば、サーバーの起動時に次の 1 行目は有効ですが 2 行目は無効です。

```
shell> mysql --max_allowed_packet=16M
shell> mysql --max_allowed_packet=16*1024*1024
```

逆に、実行時に次の 2 行目は有効ですが 1 行目は無効です。

```
mysql> SET GLOBAL max_allowed_packet=16M;
```

```
mysql> SET GLOBAL max_allowed_packet=16*1024*1024;
```

> **注記**
>
> 一部のシステム変数は、SET ステートメントで ON または 1 に設定することで有効化され、OFF または 0 に設定することで無効化されます。ただし、このような変数をコマンド行またはオプションファイルで設定するには、1 または 0 で設定する必要があり、ON または OFF に設定しても機能しません。たとえば、コマンド行において、--delay_key_write=1 は機能しますが、--delay_key_write=ON は機能しません。

システム変数名と値を表示するには、SHOW VARIABLES ステートメントを使用します。

```
mysql> SHOW VARIABLES;
+---------------------------------+----------------------------------+
| Variable_name                   | Value                            |
+---------------------------------+----------------------------------+
| auto_increment_increment        | 1                                |
| auto_increment_offset           | 1                                |
| automatic_sp_privileges         | ON                               |
| back_log                        | 50                               |
| basedir                         | /home/mysql/                     |
| binlog_cache_size               | 32768                            |
| bulk_insert_buffer_size         | 8388608                          |
| character_set_client            | latin1                           |
| character_set_connection        | latin1                           |
| character_set_database          | latin1                           |
| character_set_results           | latin1                           |
| character_set_server            | latin1                           |
| character_set_system            | utf8                             |
| character_sets_dir              | /home/mysql/share/mysql/charsets/ |
| collation_connection            | latin1_swedish_ci                |
| collation_database              | latin1_swedish_ci                |
| collation_server                | latin1_swedish_ci                |
...
| innodb_additional_mem_pool_size | 1048576                          |
| innodb_autoextend_increment     | 8                                |
| innodb_buffer_pool_size         | 8388608                          |
| innodb_checksums                | ON                               |
| innodb_commit_concurrency       | 0                                |
| innodb_concurrency_tickets      | 500                              |
| innodb_data_file_path           | ibdata1:10M:autoextend           |
| innodb_data_home_dir            |                                  |
...
| version                         | 5.1.6-alpha-log                  |
| version_comment                 | Source distribution              |
| version_compile_machine         | i686                             |
| version_compile_os              | suse-linux                       |
| wait_timeout                    | 28800                            |
+---------------------------------+----------------------------------+
```

LIKE 句では、ステートメントはパターンに一致する変数のみを表示します。特定の変数名を取得するには、LIKE 句を次のように使用します。

```
SHOW VARIABLES LIKE 'max_join_size';
SHOW SESSION VARIABLES LIKE 'max_join_size';
```

名前がパターンと一致する変数のリストを取得するには、LIKE 句の中で 「%」 のワイルドカード文字を使用します。

```
SHOW VARIABLES LIKE '%size%';
SHOW GLOBAL VARIABLES LIKE '%size%';
```

ワイルドカード文字は、照合されるパターン内のどの場所でも利用できます。厳密には、「_」 は任意の 1 文字と一致するワイルドカードであるため、文字どおりに照合するには 「_」 としてエスケープしてください。実際には、これはほとんど必要ありません。

SHOW VARIABLES で GLOBAL および SESSION をいずれも指定しない場合、MySQL は SESSION 値を返します。

GLOBAL 専用の変数を設定するときに GLOBAL キーワードが必要であるのに、それらを取得するときにキーワードが不要である理由は、今後の問題を防ぐためです。GLOBAL 変数と同じ名前を持つ SESSION 変数を削除した場合、SUPER 権限を持つクライアントが、その接続に対する SESSION 変数だけでなく、誤って GLOBAL 変数も変更してしまう可能性があります。SESSION 変数を GLOBAL 変数と同じ名前で追加する場合、クライアントが GLOBAL 変数を変更しようとしたが、クライアントの SESSION 変数のみが変更されてしまう可能性があります。

5.1.5.1 構造化システム変数

構造化システム変数は 2 つの点で通常のシステム変数と異なります。

- この値は、密接に関連すると考えられるサーバーパラメータを指定するコンポーネントを持つ構造です。
- あるタイプの構造化変数に複数のインスタンスがある場合もあります。それぞれが異なる名前を持ち、サーバーによって保持される異なるリソースを参照します。

MySQL 5.6 では 1 つの構造化変数タイプがサポートされており、この構造化変数タイプはキーキャッシュの操作を制御するパラメータを指定します。キーキャッシュ構造化変数には次のコンポーネントがあります。

- key_buffer_size
- key_cache_block_size
- key_cache_division_limit
- key_cache_age_threshold

このセクションでは、構造化変数を参照するための構文について説明します。キーキャッシュ変数は構文の例で使用されますが、キーキャッシュの操作方法についての具体的な詳細は、セクション8.9.2「MyISAM キーキャッシュ」に記載されています。

構造化変数インスタンスのコンポーネントを参照するには、instance_name.component_name 形式の複合名を使用できます。例:

```
hot_cache.key_buffer_size
hot_cache.key_cache_block_size
cold_cache.key_cache_block_size
```

それぞれの構造化システム変数には、default という名前のインスタンスが常に事前定義されます。インスタンス名を付けずに構造化変数のコンポーネントを参照した場合、default インスタンスが使用されます。つまり、default.key_buffer_size および key_buffer_size は両方とも同じシステム変数を指します。

構造化変数インスタンスおよびコンポーネントは次の命名ルールに従います。

- あるタイプの構造化変数について、それぞれのインスタンスは、そのタイプの変数の範囲内で一意の名前を持つ必要があります。ただし、インスタンス名は構造化変数タイプをまたいで一意である必要はありません。たとえば、各構造化変数には default という名前のインスタンスがあるため、default は変数タイプをまたいで一意ではありません。
- 各構造化変数タイプのコンポーネントの名前は、すべてのシステム変数名で一意である必要があります。このようにならない場合 (つまり、2 つの異なる構造化変数のタイプがコンポーネントメンバー名を共有する場合)、インスタンス名によって修飾されないメンバー名への参照に使用するデフォルトの構造化変数が明確でなくなります。
- 構造化変数インスタンス名が引用符で囲まれていない識別子として有効でない場合、逆引用符を使用した、引用符で囲まれた識別子としてこれを指定します。たとえば、hot-cache は有効ではありませんが、`hot-cache` は有効です。
- global、session、local は有効なインスタンス名ではありません。これは、非構造化システム変数を参照するための @@GLOBAL.var_name などの表記法での競合を回避します。

現時点では、構造化変数タイプはキーキャッシュのものだけであるため、最初の 2 つのルールが違反される可能性はありません。これらのルールは、ほかの何らかの構造化変数のタイプが将来作成された場合に重要性を帯びることになります。

1 つの例外はありますが、単純な変数名を指定できるあらゆるコンテキストで、複合名を使用すると構造化変数コンポーネントを参照できます。たとえば、コマンド行オプションを使用すると、構造化変数に値を割り当てることができます。

```
shell> mysqld --hot_cache.key_buffer_size=64K
```

オプションファイルでは、次の構文を使用します。

```
[mysqld]
hot_cache.key_buffer_size=64K
```

このオプションでサーバーを起動する場合、デフォルトサイズが 8M バイトのデフォルトのキーキャッシュに加えて、サイズが 64K バイトの hot_cache という名前のキーキャッシュが作成されます。

次のようにサーバーを開始したとします。

```
shell> mysqld --key_buffer_size=256K \
       --extra_cache.key_buffer_size=128K \
       --extra_cache.key_cache_block_size=2048
```

この場合、サーバーはデフォルトキーキャッシュのサイズを 256K バイトに設定します。(--default.key_buffer_size=256K と記述することもできます)。さらに、このサーバーは、128K バイトのサイズを持つ extra_cache という名前の 2 番目のキーキャッシュを作成し、テーブルインデックスブロックのキャッシュ用のブロックバッファーのサイズを 2048 バイトに設定します。

次の例では、サイズの比が 3:1:1 である 3 つの異なるキーキャッシュを指定してサーバーを開始します。

```
shell> mysqld --key_buffer_size=6M \
       --hot_cache.key_buffer_size=2M \
       --cold_cache.key_buffer_size=2M
```

構造化変数値は実行時にも設定および取得できます。たとえば、hot_cache という名前のキーキャッシュを 10M バイトのサイズに設定するには、次のステートメントのどちらかを使用します。

```
mysql> SET GLOBAL hot_cache.key_buffer_size = 10*1024*1024;
mysql> SET @@GLOBAL.hot_cache.key_buffer_size = 10*1024*1024;
```

キャッシュサイズを取得するには、次のようにします。

```
mysql> SELECT @@GLOBAL.hot_cache.key_buffer_size;
```

ただし、次のステートメントは機能しません。この変数は複合名として解釈されず、LIKE のパターンマッチング操作の単純文字列として解釈されます。

```
mysql> SHOW GLOBAL VARIABLES LIKE 'hot_cache.key_buffer_size';
```

これは、単純な変数名を指定できるすべての場所で構造化変数を使用できる例外です。

5.1.5.2 動的システム変数

多くのサーバーシステム変数は動的で、SET GLOBAL または SET SESSION を使用すると実行時に設定できます。また、SELECT を使用してこれらの値を取得することも可能です。セクション5.1.5「システム変数の使用」を参照してください。

次の表に、すべての動的システム変数の完全なリストを示します。最後のカラムは、それぞれの変数に対して GLOBAL または SESSION のいずれが (あるいは両方が) 適用されるかを示します。テーブルには、SET ステートメントで設定できるセッションオプションもリストしています。これらのオプションについては、セクション5.1.4「サーバーシステム変数」に説明があります。

「string」 型の変数は文字列値を取ります。「numeric」 型の変数は数値を取ります。「boolean」 型の変数は、0、1、ON、OFF に設定できます。(これらをコマンド行またはオプションファイルで設定する場合は数値を使用します。)「enumeration」 と記載されている変数は、通常はその変数に対して使用可能ないずれかの値に設定しますが、目的とする列挙値に相当する数値も設定できます。列挙されたシステム変数について、最初の列挙値は 0 になります。これは ENUM カラムと異なり、この場合の最初の列挙値は 1 になります。

表 5.4 動的変数のサマリー

変数名	変数型	変数スコープ
audit_log_connection_policy	列挙	グローバル
audit_log_exclude_accounts	文字列	グローバル
audit_log_flush	ブール	グローバル
audit_log_include_accounts	文字列	グローバル
audit_log_policy	列挙	グローバル
audit_log_rotate_on_size	数値	グローバル
audit_log_statement_policy	列挙	グローバル
auto_increment_increment	数値	両方
auto_increment_offset	数値	両方
autocommit	ブール	両方
automatic_sp_privileges	ブール	グローバル
big_tables	ブール	両方
binlog_cache_size	数値	グローバル

変数名	変数型	変数スコープ
binlog_checksum	文字列	グローバル
binlog_direct_non_transactional_updates	ブール	両方
binlog_error_action	列挙	両方
binlog_format	列挙	両方
binlog_max_flush_queue_time	数値	グローバル
binlog_order_commits	ブール	グローバル
binlog_row_image=image_type	列挙	両方
binlog_rows_query_log_events	ブール	両方
binlog_stmt_cache_size	数値	グローバル
binlogging_impossible_mode	列挙	両方
block_encryption_mode	文字列	両方
bulk_insert_buffer_size	数値	両方
character_set_client	文字列	両方
character_set_connection	文字列	両方
character_set_database	文字列	両方
character_set_filesystem	文字列	両方
character_set_results	文字列	両方
character_set_server	文字列	両方
collation_connection	文字列	両方
collation_database	文字列	両方
collation_server	文字列	両方
completion_type	列挙	両方
concurrent_insert	列挙	グローバル
connect_timeout	数値	グローバル
debug	文字列	両方
debug_sync	文字列	セッション
default_storage_engine	列挙	両方
default_tmp_storage_engine	列挙	両方
default_week_format	数値	両方
delay_key_write	列挙	グローバル
delayed_insert_limit	数値	グローバル
delayed_insert_timeout	数値	グローバル
delayed_queue_size	数値	グローバル
div_precision_increment	数値	両方
end_markers_in_json	ブール	両方
engine_condition_pushdown	ブール	両方
eq_range_index_dive_limit	数値	両方
event_scheduler	列挙	グローバル
expire_logs_days	数値	グローバル
flush	ブール	グローバル
flush_time	数値	グローバル
foreign_key_checks	ブール	両方
ft_boolean_syntax	文字列	グローバル
general_log	ブール	グローバル

変数名	変数型	変数スコープ
general_log_file	ファイル名	グローバル
group_concat_max_len	数値	両方
gtid_next	列挙	セッション
gtid_purged	文字列	グローバル
host_cache_size	数値	グローバル
identity	数値	セッション
init_connect	文字列	グローバル
init_slave	文字列	グローバル
innodb_adaptive_flushing	ブール	グローバル
innodb_adaptive_flushing_lwm	数値	グローバル
innodb_adaptive_hash_index	ブール	グローバル
innodb_adaptive_max_sleep_delay	数値	グローバル
innodb_api_bk_commit_interval	数値	グローバル
innodb_api_trx_level	数値	グローバル
innodb_autoextend_increment	数値	グローバル
innodb_buffer_pool_dump_at_shutdown	ブール	グローバル
innodb_buffer_pool_dump_now	ブール	グローバル
innodb_buffer_pool_filename	ファイル名	グローバル
innodb_buffer_pool_load_abort	ブール	グローバル
innodb_buffer_pool_load_now	ブール	グローバル
innodb_change_buffer_max_size	数値	グローバル
innodb_change_buffering	列挙	グローバル
innodb_checksum_algorithm	列挙	グローバル
innodb_cmp_per_index_enabled	ブール	グローバル
innodb_commit_concurrency	数値	グローバル
innodb_compression_failure_threshold_pct	数値	グローバル
innodb_compression_level	数値	グローバル
innodb_compression_pad_pct_max	数値	グローバル
innodb_concurrency_tickets	数値	グローバル
innodb_disable_sort_file_cache	ブール	グローバル
innodb_fast_shutdown	数値	グローバル
innodb_file_format	文字列	グローバル
innodb_file_format_max	文字列	グローバル
innodb_file_per_table	ブール	グローバル
innodb_flush_log_at_timeout	数値	グローバル
innodb_flush_log_at_trx_commit	列挙	グローバル
innodb_flush_neighbors	列挙	グローバル
innodb_flushing_avg_loops	数値	グローバル
innodb_ft_aux_table	文字列	グローバル
innodb_ft_enable_diag_print	ブール	グローバル
innodb_ft_enable_stopword	ブール	グローバル
innodb_ft_num_word_optimize	数値	グローバル
innodb_ft_result_cache_limit	数値	グローバル
innodb_ft_server_stopword_table	文字列	グローバル

変数名	変数型	変数スコープ
innodb_ft_user_stopword_table	文字列	両方
innodb_io_capacity	数値	グローバル
innodb_io_capacity_max	数値	グローバル
innodb_large_prefix	ブール	グローバル
innodb_lock_wait_timeout	数値	両方
innodb_log_compressed_pages	ブール	グローバル
innodb_lru_scan_depth	数値	グローバル
innodb_max_dirty_pages_pct	数値	グローバル
innodb_max_dirty_pages_pct_lwm	数値	グローバル
innodb_max_purge_lag	数値	グローバル
innodb_max_purge_lag_delay	数値	グローバル
innodb_monitor_disable	文字列	グローバル
innodb_monitor_enable	文字列	グローバル
innodb_monitor_reset	文字列	グローバル
innodb_monitor_reset_all	文字列	グローバル
innodb_old_blocks_pct	数値	グローバル
innodb_old_blocks_time	数値	グローバル
innodb_online_alter_log_max_size	数値	グローバル
innodb_optimize_fulltext_only	ブール	グローバル
innodb_print_all_deadlocks	ブール	グローバル
innodb_purge_batch_size	数値	グローバル
innodb_random_read_ahead	ブール	グローバル
innodb_read_ahead_threshold	数値	グローバル
innodb_replication_delay	数値	グローバル
innodb_rollback_segments	数値	グローバル
innodb_spin_wait_delay	数値	グローバル
innodb_stats_auto_recalc	ブール	グローバル
innodb_stats_method	列挙	グローバル
innodb_stats_on_metadata	ブール	グローバル
innodb_stats_persistent	ブール	グローバル
innodb_stats_persistent_sample_pages	数値	グローバル
innodb_stats_sample_pages	数値	グローバル
innodb_stats_transient_sample_pages	数値	グローバル
innodb_status_output	ブール	グローバル
innodb_status_output_locks	ブール	グローバル
innodb_strict_mode	ブール	両方
innodb_support_xa	ブール	両方
innodb_sync_spin_loops	数値	グローバル
innodb_table_locks	ブール	両方
innodb_thread_concurrency	数値	グローバル
innodb_thread_sleep_delay	数値	グローバル
innodb_undo_logs	数値	グローバル
insert_id	数値	セッション
interactive_timeout	数値	両方

変数名	変数型	変数スコープ
join_buffer_size	数値	両方
keep_files_on_create	ブール	両方
key_buffer_size	数値	グローバル
key_cache_age_threshold	数値	グローバル
key_cache_block_size	数値	グローバル
key_cache_division_limit	数値	グローバル
last_insert_id	数値	セッション
lc_messages	文字列	両方
lc_time_names	文字列	両方
local_infile	ブール	グローバル
lock_wait_timeout	数値	両方
log	ファイル名	グローバル
log_bin_trust_function_creators	ブール	グローバル
log_output	セット	グローバル
log_queries_not_using_indexes	ブール	グローバル
log_slow_admin_statements	ブール	グローバル
log_slow_queries	ブール	グローバル
log_slow_slave_statements	ブール	グローバル
log_throttle_queries_not_using_indexes	数値	グローバル
log_warnings	数値	異なる
long_query_time	数値	両方
low_priority_updates	ブール	両方
master_info_repository	文字列	グローバル
master_verify_checksum	ブール	グローバル
max_allowed_packet	数値	グローバル
max_binlog_cache_size	数値	グローバル
max_binlog_size	数値	グローバル
max_binlog_stmt_cache_size	数値	グローバル
max_connect_errors	数値	グローバル
max_connections	整数	グローバル
max_delayed_threads	数値	両方
max_error_count	数値	両方
max_heap_table_size	数値	両方
max_insert_delayed_threads	数値	両方
max_join_size	数値	両方
max_length_for_sort_data	数値	両方
max_prepared_stmt_count	数値	グローバル
max_relay_log_size	数値	グローバル
max_seeks_for_key	数値	両方
max_sort_length	数値	両方
max_sp_recursion_depth	数値	両方
max_user_connections	数値	両方
max_write_lock_count	数値	グローバル
min_examined_row_limit	数値	両方

変数名	変数型	変数スコープ
myisam_data_pointer_size	数値	グローバル
myisam_max_sort_file_size	数値	グローバル
myisam_repair_threads	数値	両方
myisam_sort_buffer_size	数値	両方
myisam_stats_method	列挙	両方
myisam_use_mmap	ブール	グローバル
ndb_autoincrement_prefetch_sz	数値	両方
ndb_blob_read_batch_bytes	数値	両方
ndb_blob_write_batch_bytes	数値	両方
ndb_cache_check_time	数値	グローバル
ndb_deferred_constraints	整数	両方
ndb_deferred_constraints	整数	両方
ndb_distribution	列挙	グローバル
ndb_distribution={KEYHASH|LINHASH}	列挙	グローバル
ndb_eventbuffer_max_alloc	数値	グローバル
ndb_extra_logging	数値	グローバル
ndb_force_send	ブール	両方
ndb_index_stat_cache_entries	数値	両方
ndb_index_stat_enable	ブール	両方
ndb_index_stat_option	文字列	両方
ndb_index_stat_update_freq	数値	両方
ndb_join_pushdown	ブール	両方
ndb_log_bin	ブール	両方
ndb_log_binlog_index	ブール	グローバル
ndb_log_empty_epochs	ブール	グローバル
ndb_log_empty_epochs	ブール	グローバル
ndb_log_exclusive_reads	ブール	両方
ndb_log_exclusive_reads	ブール	両方
ndb_log_update_as_write	ブール	グローバル
ndb_log_updated_only	ブール	グローバル
ndb_optimization_delay	数値	グローバル
ndb_recv_thread_activation_threshold	数値	グローバル
ndb_recv_thread_cpu_mask	ビットマップ	グローバル
ndb_show_foreign_key_mock_tables	ブール	グローバル
ndb_slave_last_conflict_epoch	列挙	グローバル
ndb_table_no_logging	ブール	セッション
ndb_table_temporary	ブール	セッション
ndb_use_exact_count	ブール	両方
ndb_use_transactions	ブール	両方
ndbinfo_max_bytes	数値	両方
ndbinfo_max_rows	数値	両方
ndbinfo_offline	ブール	グローバル
ndbinfo_show_hidden	ブール	両方
ndbinfo_table_prefix	文字列	両方

変数名	変数型	変数スコープ
net_buffer_length	数値	両方
net_read_timeout	数値	両方
net_retry_count	数値	両方
net_write_timeout	数値	両方
new	ブール	両方
old_alter_table	ブール	両方
old_passwords	列挙	両方
optimizer_join_cache_level	数値	両方
optimizer_prune_level	ブール	両方
optimizer_search_depth	数値	両方
optimizer_switch	セット	両方
optimizer_trace	文字列	両方
optimizer_trace_features	文字列	両方
optimizer_trace_limit	数値	両方
optimizer_trace_max_mem_size	数値	両方
optimizer_trace_offset	数値	両方
preload_buffer_size	数値	両方
profiling	ブール	両方
profiling_history_size	数値	両方
pseudo_slave_mode	数値	セッション
pseudo_thread_id	数値	セッション
query_alloc_block_size	数値	両方
query_cache_limit	数値	グローバル
query_cache_min_res_unit	数値	グローバル
query_cache_size	数値	グローバル
query_cache_type	列挙	両方
query_cache_wlock_invalidate	ブール	両方
query_prealloc_size	数値	両方
rand_seed1	数値	セッション
rand_seed2	数値	セッション
range_alloc_block_size	数値	両方
read_buffer_size	数値	両方
read_only	ブール	グローバル
read_rnd_buffer_size	数値	両方
relay_log_info_repository	文字列	グローバル
relay_log_purge	ブール	グローバル
relay_log_recovery	ブール	グローバル
rpl_semi_sync_master_enabled	ブール	グローバル
rpl_semi_sync_master_timeout	数値	グローバル
rpl_semi_sync_master_trace_level	数値	グローバル
rpl_semi_sync_master_wait_no_slave	ブール	グローバル
rpl_semi_sync_slave_enabled	ブール	グローバル
rpl_semi_sync_slave_trace_level	数値	グローバル
rpl_stop_slave_timeout	整数	グローバル

変数名	変数型	変数スコープ
secure_auth	ブール	グローバル
server_id	数値	グローバル
slave_allow_batching	ブール	グローバル
slave_checkpoint_group=#	数値	グローバル
slave_checkpoint_period=#	数値	グローバル
slave_compressed_protocol	ブール	グローバル
slave_exec_mode	列挙	グローバル
slave_max_allowed_packet	数値	グローバル
slave_net_timeout	数値	グローバル
slave_parallel_workers	数値	グローバル
slave_pending_jobs_size_max	数値	グローバル
slave_rows_search_algorithms=list	セット	グローバル
slave_sql_verify_checksum	ブール	グローバル
slave_transaction_retries	数値	グローバル
slow_launch_time	数値	グローバル
slow_query_log	ブール	グローバル
slow_query_log_file	ファイル名	グローバル
sort_buffer_size	数値	両方
sql_auto_is_null	ブール	両方
sql_big_selects	ブール	両方
sql_big_tables	ブール	両方
sql_buffer_result	ブール	両方
sql_log_bin	ブール	両方
sql_log_off	ブール	両方
sql_low_priority_updates	ブール	両方
sql_max_join_size	数値	両方
sql_mode	セット	両方
sql_notes	ブール	両方
sql_quote_show_create	ブール	両方
sql_safe_updates	ブール	両方
sql_select_limit	数値	両方
sql_slave_skip_counter	数値	グローバル
sql_warnings	ブール	両方
storage_engine	列挙	両方
stored_program_cache	数値	グローバル
sync_binlog	数値	グローバル
sync_frm	ブール	グローバル
sync_master_info	数値	グローバル
sync_relay_log	数値	グローバル
sync_relay_log_info	数値	グローバル
table_definition_cache	数値	グローバル
table_open_cache	数値	グローバル
thread_cache_size	数値	グローバル
thread_pool_high_priority_connection	数値	両方

変数名	変数型	変数スコープ
thread_pool_max_unused_threads	数値	グローバル
thread_pool_prio_kickup_timer	数値	両方
thread_pool_stall_limit	数値	グローバル
time_zone	文字列	両方
timed_mutexes	ブール	グローバル
timestamp	数値	セッション
tmp_table_size	数値	両方
transaction_alloc_block_size	数値	両方
transaction_allow_batching	ブール	セッション
transaction_prealloc_size	数値	両方
tx_isolation	列挙	両方
tx_read_only	ブール	両方
unique_checks	ブール	両方
updatable_views_with_limit	ブール	両方
validate_password_dictionary_file	ファイル名	グローバル
validate_password_length	数値	グローバル
validate_password_mixed_case_count	数値	グローバル
validate_password_number_count	数値	グローバル
validate_password_policy	列挙	グローバル
validate_password_special_char_count	数値	グローバル
wait_timeout	数値	両方

5.1.6 サーバーステータス変数

サーバーには、その操作についての情報を提供する多くのステータス変数が保持されています。これらの変数およびその値は、SHOW [GLOBAL | SESSION] STATUS ステートメントを使用して表示できます (セクション 13.7.5.36「SHOW STATUS 構文」を参照してください)。オプションの GLOBAL キーワードはすべての接続にわたって値を集計し、SESSION は現在の接続についての値を表示します。

```
mysql> SHOW GLOBAL STATUS;
+-----------------------------------+------------+
| Variable_name                     | Value      |
+-----------------------------------+------------+
| Aborted_clients                   | 0          |
| Aborted_connects                  | 0          |
| Bytes_received                    | 155372598  |
| Bytes_sent                        | 1176560426 |
...
| Connections                       | 30023      |
| Created_tmp_disk_tables           | 0          |
| Created_tmp_files                 | 3          |
| Created_tmp_tables                | 2          |
...
| Threads_created                   | 217        |
| Threads_running                   | 88         |
| Uptime                            | 1389872    |
+-----------------------------------+------------+
```

多くのステータス変数は、FLUSH STATUS ステートメントで 0 にリセットされます。

次の表には使用可能なすべてのサーバーステータス変数がリストされています。

表 5.5 ステータス変数のサマリー

変数名	変数型	変数スコープ
Aborted_clients	数値	グローバル
Aborted_connects	数値	グローバル

変数名	変数型	変数スコープ
Audit_log_current_size	整数	グローバル
Audit_log_event_max_drop_size	整数	グローバル
Audit_log_events	整数	グローバル
Audit_log_events_filtered	整数	グローバル
Audit_log_events_lost	整数	グローバル
Audit_log_events_written	整数	グローバル
Audit_log_total_size	整数	グローバル
Audit_log_write_waits	整数	グローバル
Binlog_cache_disk_use	数値	グローバル
Binlog_cache_use	数値	グローバル
Binlog_stmt_cache_disk_use	数値	グローバル
Binlog_stmt_cache_use	数値	グローバル
Bytes_received	数値	両方
Bytes_sent	数値	両方
Com_admin_commands	数値	両方
Com_alter_db	数値	両方
Com_alter_db_upgrade	数値	両方
Com_alter_event	数値	両方
Com_alter_function	数値	両方
Com_alter_procedure	数値	両方
Com_alter_server	数値	両方
Com_alter_table	数値	両方
Com_alter_tablespace	数値	両方
Com_alter_user	数値	両方
Com_analyze	数値	両方
Com_assign_to_keycache	数値	両方
Com_begin	数値	両方
Com_binlog	数値	両方
Com_call_procedure	数値	両方
Com_change_db	数値	両方
Com_change_master	数値	両方
Com_check	数値	両方
Com_checksum	数値	両方
Com_commit	数値	両方
Com_create_db	数値	両方
Com_create_event	数値	両方
Com_create_function	数値	両方
Com_create_index	数値	両方
Com_create_procedure	数値	両方
Com_create_server	数値	両方
Com_create_table	数値	両方
Com_create_trigger	数値	両方
Com_create_udf	数値	両方
Com_create_user	数値	両方

変数名	変数型	変数スコープ
Com_create_view	数値	両方
Com_dealloc_sql	数値	両方
Com_delete	数値	両方
Com_delete_multi	数値	両方
Com_do	数値	両方
Com_drop_db	数値	両方
Com_drop_event	数値	両方
Com_drop_function	数値	両方
Com_drop_index	数値	両方
Com_drop_procedure	数値	両方
Com_drop_server	数値	両方
Com_drop_table	数値	両方
Com_drop_trigger	数値	両方
Com_drop_user	数値	両方
Com_drop_view	数値	両方
Com_empty_query	数値	両方
Com_execute_sql	数値	両方
Com_flush	数値	両方
Com_get_diagnostics	数値	両方
Com_grant	数値	両方
Com_ha_close	数値	両方
Com_ha_open	数値	両方
Com_ha_read	数値	両方
Com_help	数値	両方
Com_insert	数値	両方
Com_insert_select	数値	両方
Com_install_plugin	数値	両方
Com_kill	数値	両方
Com_load	数値	両方
Com_lock_tables	数値	両方
Com_optimize	数値	両方
Com_preload_keys	数値	両方
Com_prepare_sql	数値	両方
Com_purge	数値	両方
Com_purge_before_date	数値	両方
Com_release_savepoint	数値	両方
Com_rename_table	数値	両方
Com_rename_user	数値	両方
Com_repair	数値	両方
Com_replace	数値	両方
Com_replace_select	数値	両方
Com_reset	数値	両方
Com_resignal	数値	両方
Com_revoke	数値	両方

変数名	変数型	変数スコープ
Com_revoke_all	数値	両方
Com_rollback	数値	両方
Com_rollback_to_savepoint	数値	両方
Com_savepoint	数値	両方
Com_select	数値	両方
Com_set_option	数値	両方
Com_show_authors	数値	両方
Com_show_binlog_events	数値	両方
Com_show_binlogs	数値	両方
Com_show_charsets	数値	両方
Com_show_collations	数値	両方
Com_show_contributors	数値	両方
Com_show_create_db	数値	両方
Com_show_create_event	数値	両方
Com_show_create_func	数値	両方
Com_show_create_proc	数値	両方
Com_show_create_table	数値	両方
Com_show_create_trigger	数値	両方
Com_show_databases	数値	両方
Com_show_engine_logs	数値	両方
Com_show_engine_mutex	数値	両方
Com_show_engine_status	数値	両方
Com_show_errors	数値	両方
Com_show_events	数値	両方
Com_show_fields	数値	両方
Com_show_function_code	数値	両方
Com_show_function_status	数値	両方
Com_show_grants	数値	両方
Com_show_keys	数値	両方
Com_show_master_status	数値	両方
Com_show_ndb_status	数値	両方
Com_show_new_master	数値	両方
Com_show_open_tables	数値	両方
Com_show_plugins	数値	両方
Com_show_privileges	数値	両方
Com_show_procedure_code	数値	両方
Com_show_procedure_status	数値	両方
Com_show_processlist	数値	両方
Com_show_profile	数値	両方
Com_show_profiles	数値	両方
Com_show_relaylog_events	数値	両方
Com_show_slave_hosts	数値	両方
Com_show_slave_status	数値	両方
Com_show_status	数値	両方

変数名	変数型	変数スコープ
Com_show_storage_engines	数値	両方
Com_show_table_status	数値	両方
Com_show_tables	数値	両方
Com_show_triggers	数値	両方
Com_show_variables	数値	両方
Com_show_warnings	数値	両方
Com_signal	数値	両方
Com_slave_start	数値	両方
Com_slave_stop	数値	両方
Com_stmt_close	数値	両方
Com_stmt_execute	数値	両方
Com_stmt_fetch	数値	両方
Com_stmt_prepare	数値	両方
Com_stmt_reprepare	数値	両方
Com_stmt_reset	数値	両方
Com_stmt_send_long_data	数値	両方
Com_truncate	数値	両方
Com_uninstall_plugin	数値	両方
Com_unlock_tables	数値	両方
Com_update	数値	両方
Com_update_multi	数値	両方
Com_xa_commit	数値	両方
Com_xa_end	数値	両方
Com_xa_prepare	数値	両方
Com_xa_recover	数値	両方
Com_xa_rollback	数値	両方
Com_xa_start	数値	両方
Compression	数値	セッション
Connection_errors_accept	数値	グローバル
Connection_errors_internal	数値	グローバル
Connection_errors_max_connections	数値	グローバル
Connection_errors_peer_addr	数値	グローバル
Connection_errors_select	数値	グローバル
Connection_errors_tcpwrap	数値	グローバル
Connections	数値	グローバル
Created_tmp_disk_tables	数値	両方
Created_tmp_files	数値	グローバル
Created_tmp_tables	数値	両方
Delayed_errors	数値	グローバル
Delayed_insert_threads	数値	グローバル
Delayed_writes	数値	グローバル
Flush_commands	数値	グローバル
Handler_commit	数値	両方
Handler_delete	数値	両方

変数名	変数型	変数スコープ
Handler_discover	数値	両方
Handler_external_lock	数値	両方
Handler_mrr_init	数値	両方
Handler_prepare	数値	両方
Handler_read_first	数値	両方
Handler_read_key	数値	両方
Handler_read_last	数値	両方
Handler_read_next	数値	両方
Handler_read_prev	数値	両方
Handler_read_rnd	数値	両方
Handler_read_rnd_next	数値	両方
Handler_rollback	数値	両方
Handler_savepoint	数値	両方
Handler_savepoint_rollback	数値	両方
Handler_update	数値	両方
Handler_write	数値	両方
Innodb_available_undo_logs	数値	グローバル
Innodb_buffer_pool_bytes_data	数値	グローバル
Innodb_buffer_pool_bytes_dirty	数値	グローバル
Innodb_buffer_pool_dump_status	数値	グローバル
Innodb_buffer_pool_load_status	数値	グローバル
Innodb_buffer_pool_pages_data	数値	グローバル
Innodb_buffer_pool_pages_dirty	数値	グローバル
Innodb_buffer_pool_pages_flushed	数値	グローバル
Innodb_buffer_pool_pages_free	数値	グローバル
Innodb_buffer_pool_pages_latched	数値	グローバル
Innodb_buffer_pool_pages_misc	数値	グローバル
Innodb_buffer_pool_pages_total	数値	グローバル
Innodb_buffer_pool_read_ahead	数値	グローバル
Innodb_buffer_pool_read_ahead_evicted	数値	グローバル
Innodb_buffer_pool_read_requests	数値	グローバル
Innodb_buffer_pool_reads	数値	グローバル
Innodb_buffer_pool_wait_free	数値	グローバル
Innodb_buffer_pool_write_requests	数値	グローバル
Innodb_data_fsyncs	数値	グローバル
Innodb_data_pending_fsyncs	数値	グローバル
Innodb_data_pending_reads	数値	グローバル
Innodb_data_pending_writes	数値	グローバル
Innodb_data_read	数値	グローバル
Innodb_data_reads	数値	グローバル
Innodb_data_writes	数値	グローバル
Innodb_data_written	数値	グローバル
Innodb_dblwr_pages_written	数値	グローバル
Innodb_dblwr_writes	数値	グローバル

変数名	変数型	変数スコープ
Innodb_have_atomic_builtins	数値	グローバル
Innodb_log_waits	数値	グローバル
Innodb_log_write_requests	数値	グローバル
Innodb_log_writes	数値	グローバル
Innodb_num_open_files	数値	グローバル
Innodb_os_log_fsyncs	数値	グローバル
Innodb_os_log_pending_fsyncs	数値	グローバル
Innodb_os_log_pending_writes	数値	グローバル
Innodb_os_log_written	数値	グローバル
Innodb_page_size	数値	グローバル
Innodb_pages_created	数値	グローバル
Innodb_pages_read	数値	グローバル
Innodb_pages_written	数値	グローバル
Innodb_row_lock_current_waits	数値	グローバル
Innodb_row_lock_time	数値	グローバル
Innodb_row_lock_time_avg	数値	グローバル
Innodb_row_lock_time_max	数値	グローバル
Innodb_row_lock_waits	数値	グローバル
Innodb_rows_deleted	数値	グローバル
Innodb_rows_inserted	数値	グローバル
Innodb_rows_read	数値	グローバル
Innodb_rows_updated	数値	グローバル
Innodb_truncated_status_writes	数値	グローバル
Key_blocks_not_flushed	数値	グローバル
Key_blocks_unused	数値	グローバル
Key_blocks_used	数値	グローバル
Key_read_requests	数値	グローバル
Key_reads	数値	グローバル
Key_write_requests	数値	グローバル
Key_writes	数値	グローバル
Last_query_cost	数値	セッション
Last_query_partial_plans	数値	セッション
Max_used_connections	数値	グローバル
Ndb_api_bytes_received_count	数値	グローバル
Ndb_api_bytes_received_count_session	数値	セッション
Ndb_api_bytes_received_count_slave	数値	グローバル
Ndb_api_bytes_sent_count	数値	グローバル
Ndb_api_bytes_sent_count_session	数値	セッション
Ndb_api_bytes_sent_count_slave	数値	グローバル
Ndb_api_event_bytes_count	数値	グローバル
Ndb_api_event_bytes_count_injector	数値	グローバル
Ndb_api_event_data_count	数値	グローバル
Ndb_api_event_data_count_injector	数値	グローバル
Ndb_api_event_nondata_count	数値	グローバル

変数名	変数型	変数スコープ
Ndb_api_event_nondata_count_injector	数値	グローバル
Ndb_api_pk_op_count	数値	グローバル
Ndb_api_pk_op_count_session	数値	セッション
Ndb_api_pk_op_count_slave	数値	グローバル
Ndb_api_pruned_scan_count	数値	グローバル
Ndb_api_pruned_scan_count_session	数値	セッション
Ndb_api_pruned_scan_count_slave	数値	グローバル
Ndb_api_range_scan_count	数値	グローバル
Ndb_api_range_scan_count_session	数値	セッション
Ndb_api_range_scan_count_slave	数値	グローバル
Ndb_api_read_row_count	数値	グローバル
Ndb_api_read_row_count_session	数値	セッション
Ndb_api_read_row_count_slave	数値	グローバル
Ndb_api_scan_batch_count	数値	グローバル
Ndb_api_scan_batch_count_session	数値	セッション
Ndb_api_scan_batch_count_slave	数値	グローバル
Ndb_api_table_scan_count	数値	グローバル
Ndb_api_table_scan_count_session	数値	セッション
Ndb_api_table_scan_count_slave	数値	グローバル
Ndb_api_trans_abort_count	数値	グローバル
Ndb_api_trans_abort_count_session	数値	セッション
Ndb_api_trans_abort_count_slave	数値	グローバル
Ndb_api_trans_close_count	数値	グローバル
Ndb_api_trans_close_count_session	数値	セッション
Ndb_api_trans_close_count_slave	数値	グローバル
Ndb_api_trans_commit_count	数値	グローバル
Ndb_api_trans_commit_count_session	数値	セッション
Ndb_api_trans_commit_count_slave	数値	グローバル
Ndb_api_trans_local_read_row_count	数値	グローバル
Ndb_api_trans_local_read_row_count_session	数値	セッション
Ndb_api_trans_local_read_row_count_slave	数値	グローバル
Ndb_api_trans_start_count	数値	グローバル
Ndb_api_trans_start_count_session	数値	セッション
Ndb_api_trans_start_count_slave	数値	グローバル
Ndb_api_uk_op_count	数値	グローバル
Ndb_api_uk_op_count_session	数値	セッション
Ndb_api_uk_op_count_slave	数値	グローバル
Ndb_api_wait_exec_complete_count	数値	グローバル
Ndb_api_wait_exec_complete_count_session	数値	セッション
Ndb_api_wait_exec_complete_count_slave	数値	グローバル
Ndb_api_wait_meta_request_count	数値	グローバル
Ndb_api_wait_meta_request_count_session	数値	セッション
Ndb_api_wait_meta_request_count_slave	数値	グローバル
Ndb_api_wait_nanos_count	数値	グローバル

変数名	変数型	変数スコープ
Ndb_api_wait_nanos_count_session	数値	セッション
Ndb_api_wait_nanos_count_slave	数値	グローバル
Ndb_api_wait_scan_result_count	数値	グローバル
Ndb_api_wait_scan_result_count_session	数値	セッション
Ndb_api_wait_scan_result_count_slave	数値	グローバル
Ndb_cluster_node_id	数値	両方
Ndb_config_from_host	数値	両方
Ndb_config_from_port	数値	両方
Ndb_conflict_fn_epoch	数値	グローバル
Ndb_conflict_fn_epoch_trans	数値	グローバル
Ndb_conflict_fn_max	数値	グローバル
Ndb_conflict_fn_old	数値	グローバル
Ndb_conflict_trans_conflict_commit_count	整数	グローバル
Ndb_conflict_trans_detect_iter_count	整数	グローバル
Ndb_conflict_trans_reject_count	数値	グローバル
Ndb_conflict_trans_row_conflict_count	数値	グローバル
Ndb_conflict_trans_row_reject_count	数値	グローバル
Ndb_execute_count	数値	グローバル
Ndb_last_commit_epoch_server	整数	グローバル
Ndb_last_commit_epoch_session	整数	セッション
Ndb_cluster_node_id	数値	グローバル
Ndb_number_of_data_nodes	整数	グローバル
Ndb_pruned_scan_count	数値	グローバル
Ndb_pushed_queries_defined	数値	グローバル
Ndb_pushed_queries_dropped	数値	グローバル
Ndb_pushed_queries_executed	数値	グローバル
Ndb_pushed_reads	数値	グローバル
Ndb_scan_count	数値	グローバル
Not_flushed_delayed_rows	数値	グローバル
Open_files	数値	グローバル
Open_streams	数値	グローバル
Open_table_definitions	数値	グローバル
Open_tables	数値	両方
Opened_files	数値	グローバル
Opened_table_definitions	数値	両方
Opened_tables	数値	両方
Performance_schema_accounts_lost	数値	グローバル
Performance_schema_cond_classes_lost	数値	グローバル
Performance_schema_cond_instances_lost	数値	グローバル
Performance_schema_digest_lost	数値	グローバル
Performance_schema_file_classes_lost	数値	グローバル
Performance_schema_file_handles_lost	数値	グローバル
Performance_schema_file_instances_lost	数値	グローバル
Performance_schema_hosts_lost	数値	グローバル

変数名	変数型	変数スコープ
Performance_schema_locker_lost	数値	グローバル
Performance_schema_mutex_classes_lost	数値	グローバル
Performance_schema_mutex_instances_lost	数値	グローバル
Performance_schema_rwlock_classes_lost	数値	グローバル
Performance_schema_rwlock_instances_lost	数値	グローバル
Performance_schema_session_connect_attrs_lost	数値	グローバル
Performance_schema_socket_classes_lost	数値	グローバル
Performance_schema_socket_instances_lost	数値	グローバル
Performance_schema_stage_classes_lost	数値	グローバル
Performance_schema_statement_classes_lost	数値	グローバル
Performance_schema_table_handles_lost	数値	グローバル
Performance_schema_table_instances_lost	数値	グローバル
Performance_schema_thread_classes_lost	数値	グローバル
Performance_schema_thread_instances_lost	数値	グローバル
Performance_schema_users_lost	数値	グローバル
Prepared_stmt_count	数値	グローバル
Qcache_free_blocks	数値	グローバル
Qcache_free_memory	数値	グローバル
Qcache_hits	数値	グローバル
Qcache_inserts	数値	グローバル
Qcache_lowmem_prunes	数値	グローバル
Qcache_not_cached	数値	グローバル
Qcache_queries_in_cache	数値	グローバル
Qcache_total_blocks	数値	グローバル
Queries	数値	両方
Questions	数値	両方
Rpl_semi_sync_master_clients	数値	グローバル
Rpl_semi_sync_master_net_avg_wait_time	数値	グローバル
Rpl_semi_sync_master_net_wait_time	数値	グローバル
Rpl_semi_sync_master_net_waits	数値	グローバル
Rpl_semi_sync_master_no_times	数値	グローバル
Rpl_semi_sync_master_no_tx	数値	グローバル
Rpl_semi_sync_master_status	ブール	グローバル
Rpl_semi_sync_master_timefunc_failures	数値	グローバル
Rpl_semi_sync_master_tx_avg_wait_time	数値	グローバル
Rpl_semi_sync_master_tx_wait_time	数値	グローバル
Rpl_semi_sync_master_tx_waits	数値	グローバル
Rpl_semi_sync_master_wait_pos_backtraverse	数値	グローバル
Rpl_semi_sync_master_wait_sessions	数値	グローバル
Rpl_semi_sync_master_yes_tx	数値	グローバル
Rpl_semi_sync_slave_status	ブール	グローバル
Rsa_public_key	文字列	グローバル
Select_full_join	数値	両方
Select_full_range_join	数値	両方

変数名	変数型	変数スコープ
Select_range	数値	両方
Select_range_check	数値	両方
Select_scan	数値	両方
Slave_heartbeat_period	数値	グローバル
Slave_last_heartbeat	日時	グローバル
Slave_open_temp_tables	数値	グローバル
Slave_received_heartbeats	整数	グローバル
Slave_retried_transactions	数値	グローバル
Slave_running	ブール	グローバル
Slow_launch_threads	数値	両方
Slow_queries	数値	両方
Sort_merge_passes	数値	両方
Sort_range	数値	両方
Sort_rows	数値	両方
Sort_scan	数値	両方
Ssl_accept_renegotiates	数値	グローバル
Ssl_accepts	数値	グローバル
Ssl_callback_cache_hits	数値	グローバル
Ssl_cipher	文字列	両方
Ssl_cipher_list	文字列	両方
Ssl_client_connects	数値	グローバル
Ssl_connect_renegotiates	数値	グローバル
Ssl_ctx_verify_depth	数値	グローバル
Ssl_ctx_verify_mode	数値	グローバル
Ssl_default_timeout	数値	両方
Ssl_finished_accepts	数値	グローバル
Ssl_finished_connects	数値	グローバル
Ssl_server_not_after	数値	両方
Ssl_server_not_before	数値	両方
Ssl_session_cache_hits	数値	グローバル
Ssl_session_cache_misses	数値	グローバル
Ssl_session_cache_mode	文字列	グローバル
Ssl_session_cache_overflows	数値	グローバル
Ssl_session_cache_size	数値	グローバル
Ssl_session_cache_timeouts	数値	グローバル
Ssl_sessions_reused	数値	両方
Ssl_used_session_cache_entries	数値	グローバル
Ssl_verify_depth	数値	両方
Ssl_verify_mode	数値	両方
Ssl_version	文字列	両方
Table_locks_immediate	数値	グローバル
Table_locks_waited	数値	グローバル
Table_open_cache_hits	数値	両方
Table_open_cache_misses	数値	両方

変数名	変数型	変数スコープ
Table_open_cache_overflows	数値	両方
Tc_log_max_pages_used	数値	グローバル
Tc_log_page_size	数値	グローバル
Tc_log_page_waits	数値	グローバル
Threads_cached	数値	グローバル
Threads_connected	数値	グローバル
Threads_created	数値	グローバル
Threads_running	数値	グローバル
Uptime	数値	グローバル
Uptime_since_flush_status	数値	グローバル

ステータス変数には次の意味があります。MySQL Cluster に固有のステータス変数の意味については、セクション18.3.4.4「MySQL Cluster のステータス変数」を参照してください。

- Aborted_clients

  クライアントが接続を適切に閉じることなく終了したため中止された接続の数。セクションB.5.2.11「通信エラーおよび中止された接続」を参照してください。

- Aborted_connects

  MySQL Server への接続に失敗した試行数。セクションB.5.2.11「通信エラーおよび中止された接続」を参照してください。

  接続に関する追加情報については、Connection_errors_xxx ステータス変数および host_cache テーブルを確認してください。

- Binlog_cache_disk_use

  一時バイナリログキャッシュを使用したが、binlog_cache_size の値を超えたため、一時ファイルを使用してトランザクションからのステートメントを保管したトランザクション数。

  バイナリログトランザクションキャッシュがディスクに書き込まれた非トランザクションステートメントの数は、Binlog_stmt_cache_disk_use ステータス変数で別途追跡されます。

- Binlog_cache_use

  バイナリログキャッシュを使用したトランザクション数。

- Binlog_stmt_cache_disk_use

  バイナリログステートメントキャッシュを使用したが、binlog_stmt_cache_size の値を超えたため、一時ファイルを使用してこれらのステートメントを保管した、非トランザクションステートメントの数。

- Binlog_stmt_cache_use

  バイナリログステートメントキャッシュを使用した非トランザクションステートメントの数。

- Bytes_received

  すべてのクライアントから受信したバイト数。

- Bytes_sent

  すべてのクライアントに送信されたバイト数。

- Com_xxx

  Com_xxx ステートメントカウンタ変数は、それぞれの xxx ステートメントが実行された回数を示します。ステートメントのタイプごとにステータス変数が 1 つあります。たとえば、Com_delete および Com_update はそれぞれ DELETE および UPDATE ステートメントをカウントします。Com_delete_multi および Com_update_multi は類似していますが、複数テーブル構文を使用する DELETE および UPDATE ステートメントに適用されます。

クエリー結果がクエリーキャッシュから返される場合、サーバーは Com_select ではなく Qcache_hits ステータス変数を増やします。セクション8.9.3.4「クエリーキャッシュのステータスと保守」を参照してください。

プリペアドステートメントの引数が不明であったり、実行中にエラーが発生したりした場合であっても、すべての Com_stmt_xxx 変数が増加します。つまり、これらの値は発行されたリクエスト数に対応し、正常に完了したリクエスト数に対応しません。

Com_stmt_xxx ステータス変数は次のとおりです。

- Com_stmt_prepare
- Com_stmt_execute
- Com_stmt_fetch
- Com_stmt_send_long_data
- Com_stmt_reset
- Com_stmt_close

これらの変数は、プリペアドステートメントコマンドを表します。これらの名前は、ネットワーク層で使用される COM_xxx コマンドセットを示します。つまり、mysql_stmt_prepare() や mysql_stmt_execute() などのプリペアドステートメントの API コールを実行すると、これらの値は増加します。ただし、Com_stmt_prepare、Com_stmt_execute、および Com_stmt_close も、PREPARE、EXECUTE、または DEALLOCATE PREPARE に対してそれぞれ増加します。さらに、古いステートメントカウンタ変数の値 Com_prepare_sql、Com_execute_sql、および Com_dealloc_sql は、PREPARE、EXECUTE、および DEALLOCATE PREPARE ステートメントに対して増加します。Com_stmt_fetch はカーソルからフェッチしたときに発行されるネットワーク往復の合計回数のことです。

Com_stmt_reprepare は、ステートメントによって参照されるテーブルまたはビューへのメタデータの変更後に、ステートメントがサーバーによって自動的に再作成された回数を示します。再作成操作は Com_stmt_reprepare および Com_stmt_prepare を増加させます。

- Compression

  クライアント接続で、クライアント/サーバープロトコルの圧縮を使用するかどうか。

- Connection_errors_xxx

  これらの変数は、クライアント接続プロセス中に発生したエラーについての情報を提供します。これらはグローバル専用で、すべてのホストからの接続全体で集計したエラー数を表します。これらの変数は、ホストキャッシュによって説明されないエラーを追跡し (セクション8.11.5.2「DNS ルックアップの最適化とホストキャッシュ」を参照してください)、たとえば、TCP 接続に関連付けられないエラーや、接続プロセスのきわめて早期に (IP アドレスが既知となる前も含めて) 発生するエラー、または特定の IP アドレスに固有でない (メモリー不足の状況などの) エラーなどです。これらの変数は MySQL 5.6.5 で追加されました。

  - Connection_errors_accept

    リスニングポートでの accept() への呼び出し中に発生したエラーの数。

  - Connection_errors_internal

    新しいスレッドの開始のエラーやメモリー不足状況など、サーバーの内部エラーが原因で拒否された接続の数。

  - Connection_errors_max_connections

    サーバーの max_connections 制限に到達したため拒否された接続の数。

  - Connection_errors_peer_addr

    クライアント IP アドレスへの接続の検索中に発生したエラーの数。

  - Connection_errors_select

    リスニングポートでの select() または poll() への呼び出し中に発生したエラーの数。(この操作に失敗したことは、クライアント接続が拒否されたことを必ずしも意味しません。)

  - Connection_errors_tcpwrap

    libwrap ライブラリによって拒否された接続の数。

- Connections

  MySQL Server への (成功またはそれ以外の) 接続の試行数。

- Created_tmp_disk_tables

  ステートメントの実行中にサーバーによって作成された、ディスク上の内部一時テーブルの数。

  内部一時テーブルが最初にインメモリーテーブルとして作成されたが、これが大きくなりすぎた場合、MySQL はこれを自動的にディスク上のテーブルに変換します。インメモリー一時テーブルの最大サイズは、tmp_table_size と max_heap_table_size の最小値です。Created_tmp_disk_tables が大きい場合、メモリー内の内部一時テーブルがディスク上のテーブルに変換される可能性を低くするため、tmp_table_size または max_heap_table_size の値を増加させた方がよいこともあります。

  Created_tmp_disk_tables 変数と Created_tmp_tables 変数の値を比較することによって、作成された内部のディスク上の一時テーブル数と、作成された内部の一時テーブルの総数を比較できます。

  セクション8.4.4「MySQL が内部一時テーブルを使用する仕組み」も参照してください。

- Created_tmp_files

  mysqld が生成した一時ファイルの数。

- Created_tmp_tables

  ステートメントの実行中にサーバーによって作成された、内部一時テーブルの数。

  Created_tmp_disk_tables 変数と Created_tmp_tables 変数の値を比較することによって、作成された内部のディスク上の一時テーブル数と、作成された内部の一時テーブルの総数を比較できます。

  セクション8.4.4「MySQL が内部一時テーブルを使用する仕組み」も参照してください。

  SHOW STATUS ステートメントを呼び出すたびに内部一時テーブルが使用され、グローバルの Created_tmp_tables 値が増加します。

- Delayed_errors

  一部のエラーが発生した (多くの場合は duplicate key)、INSERT DELAYED で書き込まれるレコードの数。

  MySQL 5.6.7 以降では、このステータス変数は非推奨となり (DELAYED 挿入が非推奨となったため)、今後のリリースで削除される予定です。

- Delayed_insert_threads

  非トランザクションテーブルに対して使用中の INSERT DELAYED ハンドラスレッドの数。

  MySQL 5.6.7 以降では、このステータス変数は非推奨となり (DELAYED 挿入が非推奨となったため)、今後のリリースで削除される予定です。

- Delayed_writes

  非トランザクションテーブルに書き込まれる INSERT DELAYED 行の数。

  MySQL 5.6.7 以降では、このステータス変数は非推奨となり (DELAYED 挿入が非推奨となったため)、今後のリリースで削除される予定です。

- Flush_commands

  ユーザーが FLUSH TABLES ステートメントを実行したか、内部のサーバー動作が原因で、サーバーがテーブルをフラッシュする回数。これは COM_REFRESH パケットの受信によっても増加します。これは Com_flush とは対照的で、FLUSH TABLES、FLUSH LOGS などのいずれかの FLUSH ステートメントが実行された回数を示します。

- Handler_commit

  内部 COMMIT ステートメントの数。

- Handler_delete

  テーブルから行が削除された回数。

- Handler_external_lock

  サーバーは external_lock() 関数への呼び出しごとにこの変数を増加し、この呼び出しは通常、テーブルインスタンスへのアクセスの最初と最後に発生します。ストレージエンジンによって相違がある場合があります。この変数は、たとえばパーティション化されたテーブルにアクセスするステートメントについて、ロックが発生する前に削除されたパーティション数を検出するために使用されます。ステートメントについてカウンタがいくら増加したかを確認し、2 を減算し (テーブルそのものに対する 2 件の呼び出し)、2 で除算して、ロックされたパーティション数を取得します。この変数は MySQL 5.6.2 で追加されました。

- Handler_mrr_init

  サーバーがテーブルへのアクセスでストレージエンジン独自の Multi-Range Read 実装を使用する回数。この変数は MySQL 5.6.1 で追加されました。

- Handler_prepare

  2 フェーズコミット操作の準備フェーズのカウンタ。

- Handler_read_first

  インデックスの最初のエントリが読み取られた回数。この値が大きい場合、サーバーは多数のフルインデックススキャンを実行している可能性があり、たとえば、SELECT col1 FROM foo で col1 がインデックス付けされている場合などがあります。

- Handler_read_key

  キーに基づいて行を読み取るリクエスト数。この値が高いことは、クエリーに対してテーブルが適切にインデックス付けされていることのよい目安になります。

- Handler_read_last

  インデックスの最後のキーを読み取るリクエスト数。ORDER BY の場合、サーバーは先頭キーのリクエストのあとでいくつかの次のキーのリクエストを発行し、ORDER BY DESC の場合、サーバーは最終キーのリクエストのあとで前のキーのリクエストを発行します。この変数は MySQL 5.6.1 で追加されました。

- Handler_read_next

  キー順で次の行の読み取りリクエスト数。この値は、範囲制約を持つインデックスカラムにクエリーを実行するか、インデックススキャンを実行する場合に増加します。

- Handler_read_prev

  キー順で前の行の読み取りリクエスト数。この読み取り方法は、ORDER BY ... DESC を最適化するために主に使用されます。

- Handler_read_rnd

  固定された位置に基づいた行読み取りリクエスト数。この値は、結果のソートが必要となる多くのクエリーを実行する場合に高くなります。MySQL がテーブル全体をスキャンする必要がある多くのクエリーが存在する可能性があるか、キーが適切に使用されない結合があります。

- Handler_read_rnd_next

  データファイル内で次の行の読み取りリクエスト数。多くのテーブルスキャンを実行すると、この値は高くなります。一般的に、これはテーブルが正しくインデックス付けされていないか、作成したインデックスを利用するようにクエリーが記述されていないことを示します。

- Handler_rollback

  ロールバック操作を実行するためのストレージエンジンのリクエスト数。

- Handler_savepoint

セーブポイントを配置するためのストレージエンジンへのリクエスト数。

- Handler_savepoint_rollback

  セーブポイントをロールバックするためのストレージエンジンへのリクエスト数。

- Handler_update

  テーブルの行更新リクエスト数。

- Handler_write

  テーブルへの行挿入リクエスト数。

- Innodb_available_undo_logs

  使用可能な InnoDB undo ログの総数。アクティブな undo ログの数を報告する、innodb_undo_logs システム変数を補います。

- Innodb_buffer_pool_dump_status

  InnoDB バッファープールに保持されるページを記録するための操作の進捗状況で、innodb_buffer_pool_dump_at_shutdown または innodb_buffer_pool_dump_now の設定によってトリガーされます。

- Innodb_buffer_pool_load_status

  以前の時点のものに対応するページのセットを読み取ることによって、InnoDB バッファープールをウォームアップする操作の進捗状況で、innodb_buffer_pool_load_at_startup または innodb_buffer_pool_load_now の設定によってトリガーされます。操作によってもたらされるオーバーヘッドが多すぎる場合、innodb_buffer_pool_load_abort を設定すると取り消しできます。

- Innodb_buffer_pool_bytes_data

  データを含む InnoDB バッファープール内のバイトの総数。ダーティーページとクリーンページの両方が含まれます。圧縮テーブルによってバッファープールが異なるサイズのページを保持する場合に、Innodb_buffer_pool_pages_data を使用するよりも正確なメモリー使用量を計算するために使用します。

- Innodb_buffer_pool_pages_data

  データを含む InnoDB バッファープール内のページ数。ダーティーページとクリーンページの両方が含まれます。

- Innodb_buffer_pool_bytes_dirty

  InnoDB バッファープール内のダーティーページに保持されている現在の合計バイト数。圧縮テーブルによってバッファープールが異なるサイズのページを保持する場合に、Innodb_buffer_pool_pages_dirty を使用するよりも正確なメモリー使用量を計算するために使用します。

- Innodb_buffer_pool_pages_dirty

  InnoDB バッファープール内のダーティーページの現在の数。

- Innodb_buffer_pool_pages_flushed

  InnoDB バッファープールからページをフラッシュするためのリクエスト数。

- Innodb_buffer_pool_pages_free

  InnoDB バッファープール内の空きページの数。

- Innodb_buffer_pool_pages_latched

  InnoDB バッファープール内のラッチされたページの数。これらは現在読み取りまたは書き込み中であるか、ほかの何らかの理由でフラッシュまたは削除できないページです。この変数の計算にはコストがかかるため、UNIV_DEBUG システムがサーバー構築時に定義される場合のみ利用できます。

- Innodb_buffer_pool_pages_misc

  行ロックやアダプティブハッシュインデックスなど、管理オーバーヘッドに割り当てられているためビジー状態になっている、InnoDB バッファープール内のページ数。この値は Innodb_buffer_pool_pages_total − Innodb_buffer_pool_pages_free − Innodb_buffer_pool_pages_data として計算することもできます。

- Innodb_buffer_pool_pages_total

  InnoDB バッファープールの合計サイズ (ページ単位)。

- Innodb_buffer_pool_read_ahead

  先読みバックグラウンドスレッドによって InnoDB バッファープールに読み取られたページ数。

- Innodb_buffer_pool_read_ahead_evicted

  クエリーによってアクセスされずにあとで消去された先読みバックグラウンドスレッドによって InnoDB バッファープールに読み取られたページ数。

- Innodb_buffer_pool_read_requests

  論理読み取りリクエスト数。

- Innodb_buffer_pool_reads

  InnoDB がバッファープールから満たすことができず、ディスクから直接読み取る必要があった論理読み取りの数。

- Innodb_buffer_pool_wait_free

  通常は、InnoDB バッファープールへの書き込みは、バックグラウンドで行われます。InnoDB がページを読み取るか作成する必要があって、クリーンページが利用できない場合、InnoDB は一部のダーティーページを最初にフラッシュし、その操作の完了まで待機します。このカウンタはこれらの待機のインスタンスをカウントします。innodb_buffer_pool_size が適切に設定されていれば、この値は小さくなります。

- Innodb_buffer_pool_write_requests

  InnoDB バッファープールに対して実行される書き込みの数。

- Innodb_data_fsyncs

  これまでの fsync() 操作数。fsync() 呼び出しの頻度は innodb_flush_method 構成オプションの設定に影響されます。

- Innodb_data_pending_fsyncs

  現在保留中の fsync() 操作の数。fsync() 呼び出しの頻度は innodb_flush_method 構成オプションの設定に影響されます。

- Innodb_data_pending_reads

  現在保留中の読み取りの数。

- Innodb_data_pending_writes

  現在保留中の書き込み数。

- Innodb_data_read

  サーバーが開始してから読み取られたデータ量。

- Innodb_data_reads

  データ読み取りの合計数。

- Innodb_data_writes

  データ書き込みの合計数。

- Innodb_data_written

  これまでに書き込まれたデータ量 (バイト単位)。

- Innodb_dblwr_pages_written

  ダブル書き込みバッファーに書き込まれたページ数。セクション14.10.1「InnoDB ディスク I/O」を参照してください。

- Innodb_dblwr_writes

  実行されたダブル書き込み操作の数。セクション14.10.1「InnoDB ディスク I/O」を参照してください。

- Innodb_have_atomic_builtins

  サーバーがアトミック命令で構築されたかどうかを示します。

- Innodb_log_waits

  The number of times that the ログバッファーが小さすぎるため、続行する前にフラッシュするために待機が必要だった回数。

- Innodb_log_write_requests

  InnoDB Redo ログの書き込みリクエストの数。

- Innodb_log_writes

  InnoDB Redo ログファイルへの物理書き込みの数。

- Innodb_num_open_files

  InnoDB で現在開いたままになっているファイルの数。

- Innodb_os_log_fsyncs

  InnoDB Redo ログファイルに対して実行される fsync() 書き込みの数。

- Innodb_os_log_pending_fsyncs

  InnoDB Redo ログファイルに対する保留中の fsync() 操作の数。

- Innodb_os_log_pending_writes

  InnoDB Redo ログファイルに対する保留中の書き込み数。

- Innodb_os_log_written

  InnoDB Redo ログファイルに書き込まれたバイト数。

- Innodb_page_size

  InnoDB のページサイズ (デフォルトは 16K バイト)。ページには多くの値がカウントされ、ページサイズは簡単にバイトに換算できます。

- Innodb_pages_created

  InnoDB テーブルの操作によって作成されるページ数。

- Innodb_pages_read

  InnoDB テーブルの操作によって読み取られるページ数。

- Innodb_pages_written

  InnoDB テーブルの操作によって書き込まれるページ数。

- Innodb_row_lock_current_waits

  InnoDB テーブルの操作によって現在待機中の行ロックの数。

- Innodb_row_lock_time

  InnoDB テーブルの行ロックの取得に要した合計時間 (ミリ秒単位)。

- Innodb_row_lock_time_avg

InnoDB テーブルの行ロックの取得に要した平均時間 (ミリ秒)。

- Innodb_row_lock_time_max

  InnoDB テーブルの行ロックの取得に要した最大時間 (ミリ秒)。

- Innodb_row_lock_waits

  InnoDB テーブル上の操作が行ロックを待機した回数。

- Innodb_rows_deleted

  InnoDB テーブルから削除された行数。

- Innodb_rows_inserted

  InnoDB テーブルに挿入された行数。

- Innodb_rows_read

  InnoDB テーブルから読み取られた行数。

- Innodb_rows_updated

  InnoDB テーブル内で更新された行数。

- Innodb_truncated_status_writes

  SHOW ENGINE INNODB STATUS ステートメントからの出力が切り捨てられた回数。

- Key_blocks_not_flushed

  変更されたがまだディスクにフラッシュされていない MyISAM キーキャッシュ内のキーブロック数。

- Key_blocks_unused

  MyISAM キーキャッシュ内の未使用ブロック数。この値を使用して、使用中のキーキャッシュの量を判別できます。セクション5.1.4「サーバーシステム変数」の key_buffer_sizeに関する説明を参照してください。

- Key_blocks_used

  MyISAM キーキャッシュ内の使用済みブロック数。この値は、一度に使用された今までの最大ブロック数を示す高位境界値です。

- Key_read_requests

  MyISAM キーキャッシュからキーブロックを読み取るリクエスト数。

- Key_reads

  ディスクから MyISAM キーキャッシュへのキーブロックの物理的な読み取りの数。Key_reads が大きい場合、key_buffer_size の値が小さすぎる可能性があります。キャッシュミス率は Key_reads/Key_read_requests と計算できます。

- Key_write_requests

  MyISAM キーキャッシュにキーブロックを書き込むリクエスト数。

- Key_writes

  MyISAM キーキャッシュからディスクへのキーブロックの物理的な書き込みの数。

- Last_query_cost

  クエリーオプティマイザによって計算された、最後にコンパイルされたクエリーの合計コスト。これは同じクエリーに対して異なるクエリー計画のコストを比較するために役立ちます。デフォルト値の 0 は、クエリーがまだコンパイルされていないことを意味します。デフォルト値は 0 です。Last_query_cost はセッションスコープを持ちます。

  Last_query_cost 値は単純な「フラット」クエリーについてのみ正確に計算でき、サブクエリーまたは UNION を持つような複雑なクエリーには該当しません。後者の場合、値は 0 に設定されます。

- Last_query_partial_plans

  クエリーオプティマイザが前のクエリーの実行計画の構築で実行した反復数。Last_query_cost はセッションスコープを持ちます。この変数は MySQL 5.6.5 で追加されました。

- Max_used_connections

  サーバーが開始されてから同時に使用された接続の最大数。

- Not_flushed_delayed_rows

  INSERT DELAYED クエリーで非トランザクションテーブルへの書き込みを待機している行数。

  MySQL 5.6.7 以降では、このステータス変数は非推奨となり (DELAYED 挿入が非推奨となったため)、今後のリリースで削除される予定です。

- Open_files

  開いているファイルの数。このカウントにはサーバーによって開いた通常のファイルが含まれます。ソケットやパイプなどのほかのタイプのファイルは含まれません。またこのカウントには、サーバーレベルに実行を依頼するのではなく、ストレージエンジンがそれら独自の内部関数を使用して開いたファイルは含まれません。

- Open_streams

  開いているストリーム数 (主にロギングに使用)。

- Open_table_definitions

  キャッシュされた .frm ファイルの数。

- Open_tables

  開いているテーブルの数。

- Opened_files

  my_open() (mysys ライブラリ関数) によって開いたファイルの数。この関数を使用せずにファイルを開くサーバーの一部は、カウントを増加させません。

- Opened_table_definitions

  キャッシュされた .frm ファイルの数。

- Opened_tables

  開いているテーブル数。Opened_tables の値が大きい場合、table_open_cache の値が小さいすぎる可能性があります。

- Performance_schema_xxx

  パフォーマンススキーマのステータス変数は、セクション22.13「パフォーマンススキーマステータス変数」にリストされています。これらの変数は、メモリー制約のためロードまたは作成できないにインストゥルメンテーションついての情報を提供します。

- Prepared_stmt_count

  現在のプリペアドステートメントの数。(ステートメントの最大数は、max_prepared_stmt_count システム変数によって指定されます。)

- Qcache_free_blocks

  クエリーキャッシュ内の空きメモリーブロックの数。

- Qcache_free_memory

  クエリーキャッシュ用の空きメモリーの量。

- Qcache_hits

  クエリーキャッシュヒットの数。

- Qcache_inserts

  クエリーキャッシュに追加されるクエリーの数。

- Qcache_lowmem_prunes

  メモリーが少ないためクエリーキャッシュから削除されたクエリーの数。

- Qcache_not_cached

  非キャッシュクエリーの数 (キャッシュできないか、query_cache_type 設定のためキャッシュされない)。

- Qcache_queries_in_cache

  クエリーキャッシュ内に登録されたクエリーの数。

- Qcache_total_blocks

  クエリーキャッシュ内のブロックの合計数。

- Queries

  サーバーによって実行されたステートメントの数。この変数は Questions 変数と異なり、ストアドプロシージャー内で実行されるステートメントを含みます。COM_PING または COM_STATISTICS コマンドをカウントしません。

- Questions

  サーバーによって実行されたステートメントの数。これは Queries 変数とは異なり、クライアントによってサーバーに送信されたステートメントのみを含み、ストアドプロシージャー内で実行されたステートメントは含みません。この変数は、COM_PING、COM_STATISTICS、COM_STMT_PREPARE、COM_STMT_CLOSE、または COM_STMT_RESET コマンドをカウントしません。

- Rpl_semi_sync_master_clients

  準同期スレーブの数。

  この変数は、マスター側の準同期レプリケーションプラグインがインストールされている場合のみ利用できます。

- Rpl_semi_sync_master_net_avg_wait_time

  マスターがスレーブ返信を待機する平均時間 (マイクロ秒)。

  この変数は、マスター側の準同期レプリケーションプラグインがインストールされている場合のみ利用できます。

- Rpl_semi_sync_master_net_wait_time

  マスターがスレーブ返信を待機する合計時間 (マイクロ秒)。

  この変数は、マスター側の準同期レプリケーションプラグインがインストールされている場合のみ利用できます。

- Rpl_semi_sync_master_net_waits

  マスターがスレーブ返信を待機する合計回数。

  この変数は、マスター側の準同期レプリケーションプラグインがインストールされている場合のみ利用できます。

- Rpl_semi_sync_master_no_times

  マスターが準同期レプリケーションをオフにした回数。

  この変数は、マスター側の準同期レプリケーションプラグインがインストールされている場合のみ利用できます。

- Rpl_semi_sync_master_no_tx

スレーブによって正しく認証されなかったコミット数。

この変数は、マスター側の準同期レプリケーションプラグインがインストールされている場合のみ利用できます。

- Rpl_semi_sync_master_status

  準同期レプリケーションがマスター上で現在動作中であるかどうか。プラグインが有効で、コミット認証が発生した場合、この値は ON です。プラグインが有効でないか、コミット認証タイムアウトのため、マスターが非同期レプリケーションにフォールバックする場合、OFF です。

  この変数は、マスター側の準同期レプリケーションプラグインがインストールされている場合のみ利用できます。

- Rpl_semi_sync_master_timefunc_failures

  gettimeofday() などの時間関数を呼び出すときにマスターが失敗した回数。

  この変数は、マスター側の準同期レプリケーションプラグインがインストールされている場合のみ利用できます。

- Rpl_semi_sync_master_tx_avg_wait_time

  マスターが各トランザクションを待機する平均時間 (マイクロ秒)。

  この変数は、マスター側の準同期レプリケーションプラグインがインストールされている場合のみ利用できます。

- Rpl_semi_sync_master_tx_wait_time

  マスターがトランザクションを待機する合計時間 (マイクロ秒)。

  この変数は、マスター側の準同期レプリケーションプラグインがインストールされている場合のみ利用できます。

- Rpl_semi_sync_master_tx_waits

  マスターがトランザクションを待機した合計回数。

  この変数は、マスター側の準同期レプリケーションプラグインがインストールされている場合のみ利用できます。

- Rpl_semi_sync_master_wait_pos_backtraverse

  以前待機したイベントよりも低いバイナリ座標を持つイベントをマスターが待機した合計回数。これは、トランザクションが応答の待機を開始した順序が、バイナリログイベントが書き込まれた順序と異なる場合に発生することがあります。

  この変数は、マスター側の準同期レプリケーションプラグインがインストールされている場合のみ利用できます。

- Rpl_semi_sync_master_wait_sessions

  スレーブ返信を現在待機しているセッション数。

  この変数は、マスター側の準同期レプリケーションプラグインがインストールされている場合のみ利用できます。

- Rpl_semi_sync_master_yes_tx

  スレーブによって正しく認証されたコミットの数。

  この変数は、マスター側の準同期レプリケーションプラグインがインストールされている場合のみ利用できます。

- Rpl_semi_sync_slave_status

  準同期レプリケーションがスレーブ上で現在動作中かどうか。プラグインが有効であり、スレーブ I/O スレッドが実行中の場合は ON、それ以外の場合は OFF です。

この変数は、スレーブ側の準同期レプリケーションプラグインがインストールされている場合にのみ利用できます。

- Rsa_public_key

  sha256_password 認証プラグインによって使用される RSA 公開キー値。この値は、sha256_password_private_key_path および sha256_password_public_key_path システム変数によって名前が指定されるファイル内の秘密鍵および公開鍵をサーバーが正常に初期化した場合のみ空ではありません。Rsa_public_key の値は後者のファイルから得られます。

  sha256_password については、セクション6.3.8.4「SHA-256 認証プラグイン」を参照してください。

  この変数は、MySQL が OpenSSL を使用して構築されている場合のみ利用できます。これは MySQL 5.6.6 で追加されました。(MySQL Community Edition は yaSSL を使用して構築されています。)

- Select_full_join

  インデックスを使用しないためテーブルスキャンを実行する結合の数。この値が 0 でない場合、テーブルのインデックスを慎重に検査してください。

- Select_full_range_join

  参照テーブル上で範囲検索を使用した結合の数。

- Select_range

  最初のテーブルの範囲が使用された結合の数。値がきわめて大きい場合でも、これは通常重大な問題ではありません。

- Select_range_check

  各行のあとにキーの使用法がチェックされるキーなしの結合数。これが 0 でない場合、テーブルのインデックスを慎重に検査してください。

- Select_scan

  最初のテーブルのフルスキャンが実行された結合の数。

- Slave_heartbeat_period

  レプリケーションスレーブのレプリケーションハートビート間隔 (秒単位) を示します。

- Slave_last_heartbeat

  最近のハートビート信号がレプリケーションスレーブによって受信された時期を TIMESTAMP 値で示します。

- Slave_open_temp_tables

  スレーブ SQL スレッドが現在開いている一時テーブルの数。この値がゼロより大きい場合、スレーブをシャットダウンすることは安全ではありません。セクション17.4.1.22「レプリケーションと一時テーブル」を参照してください。

- Slave_received_heartbeats

  このカウンタは、スレーブが再起動またはリセットされたか、CHANGE MASTER TO ステートメントが発行された以降に、レプリケーションスレーブによってレプリケーションハートビートが受信されるごとに増加します。

- Slave_retried_transactions

  起動以降、レプリケーションスレーブ SQL スレッドがトランザクションを再試行した合計回数。

- Slave_running

  このサーバーがレプリケーションマスターに接続されているレプリケーションスレーブで、I/O および SQL スレッドの両方が実行中の場合、これは ON で、それ以外の場合は OFF です。

- Slow_launch_threads

  作成に要した時間が slow_launch_time 秒を超えたスレッドの数。

- Slow_queries

  long_query_time 秒よりも時間を要したクエリーの数。このカウンタは、スロークエリーログが有効かどうかに関係なく増加します。このログについては、セクション5.2.5「スロークエリーログ」を参照してください。

- Sort_merge_passes

  ソートアルゴリズムが実行する必要があったマージパスの数。この値が大きい場合、sort_buffer_size システム変数を増やすことを検討してください。

- Sort_range

  範囲を使用して実行されたソートの数。

- Sort_rows

  ソートされた行の数。

- Sort_scan

  テーブルをスキャンすることで実行されたソートの数。

- Ssl_accept_renegotiates

  接続を確立するために必要なネゴシエーションの数。

- Ssl_accepts

  受け入れられた SSL 接続の数。

- Ssl_callback_cache_hits

  コールバックキャッシュのヒット数。

- Ssl_cipher

  現在の SSL 暗号 (非 SSL 接続の場合は空)。

- Ssl_cipher_list

  利用可能な SSL 暗号のリスト (非 SSL 接続の場合は空)。

- Ssl_client_connects

  SSL 対応マスターに対する SSL 接続試行数。

- Ssl_connect_renegotiates

  SSL 対応マスターへの接続を確立するために必要なネゴシエーションの数。

- Ssl_ctx_verify_depth

  SSL コンテキスト検証の深さ (テストされるチェーン内の証明書数)。

- Ssl_ctx_verify_mode

  SSL コンテキスト検証モード。

- Ssl_default_timeout

  デフォルトの SSL タイムアウト。

- Ssl_finished_accepts

  サーバーへの正常な SSL 接続数。

- Ssl_finished_connects

  SSL 対応マスターへのスレーブ接続の成功数。

- Ssl_server_not_after

  SSL 証明書が有効な最終日。この変数は MySQL 5.6.3 で追加されました。

- Ssl_server_not_before

  SSL 証明書が有効な最初の日。この変数は MySQL 5.6.3 で追加されました。

- Ssl_session_cache_hits

  SSL セッションキャッシュのヒット数。

- Ssl_session_cache_misses

  SSL セッションキャッシュのミス数。

- Ssl_session_cache_mode

  SSL セッションキャッシュモード。

- Ssl_session_cache_overflows

  SSL セッションキャッシュのオーバーフロー数。

- Ssl_session_cache_size

  SSL セッションキャッシュサイズ。

- Ssl_session_cache_timeouts

  SSL セッションキャッシュのタイムアウト数。

- Ssl_sessions_reused

  キャッシュから再使用された SSL 接続の数。

- Ssl_used_session_cache_entries

  使用された SSL セッションキャッシュエントリの数。

- Ssl_verify_depth

  レプリケーション SSL 接続の検証の深さ。

- Ssl_verify_mode

  レプリケーション SSL 接続の検証モード。

- Ssl_version

  接続の SSL プロトコルバージョン。

- Table_locks_immediate

  テーブルロックのリクエストが即座に付与された回数。

- Table_locks_waited

  テーブルロックのリクエストが即座に付与されず、待機が必要だった回数。これが高く、パフォーマンスに問題がある場合、最初にクエリーを最適化し、次に 1 つ以上のテーブルを分割するか、レプリケーションを使用してください。

- Table_open_cache_hits

  開いたテーブルのキャッシュルックアップのヒット数。この変数は MySQL 5.6.6 で追加されました。

- Table_open_cache_misses

  開いたテーブルのキャッシュルックアップのミス数。この変数は MySQL 5.6.6 で追加されました。

- Table_open_cache_overflows

  開いたテーブルのキャッシュのオーバーフロー数。これはテーブルが開くか閉じたあとにキャッシュインスタンスが未使用のエントリを持ち、インスタンスのサイズが table_open_cache / table_open_cache_instances より大きい場合の回数です。この変数は MySQL 5.6.6 で追加されました。

- Tc_log_max_pages_used

  この変数は、mysqld が内部の XA トランザクションのリカバリのためのトランザクションコーディネータとしての役割を果たすとき、mysqld によって使用されるログのメモリーマップ実装に対して、サーバーが起動してからログに使用された最大のページ数を示します。Tc_log_max_pages_used と Tc_log_page_size の積が常にログサイズよりも極端に小さい場合、そのサイズが必要以上に大きいため削減できます。(このサイズは --log-tc-size オプションで指定できます。現在この変数は使用されません。バイナリログベースのリカバリは不要で、2 フェーズコミットが可能なストレージエンジンの数が 1 より大きい場合を除き、メモリーマップリカバリログ方式は使用されません。(InnoDB のみが該当するエンジンです。)

- Tc_log_page_size

  XA リカバリログのメモリーマップ実装に使用されるページサイズ。デフォルト値は getpagesize() を使用して決定されます。この変数は、Tc_log_max_pages_used について記述したものと同じ理由で現在使用されません。

- Tc_log_page_waits

  この変数は、リカバリログのメモリーマップ実装で、サーバーがトランザクションをコミットできず、ログ内の空きページを待機する必要がある場合に毎回増加します。この値が大きい場合、(--log-tc-size オプションで) ログサイズを増加した方がよい場合もあります。この変数は、バイナリログベースのリカバリで、2 フェーズコミットが進行中のためバイナリログをクローズできない場合に毎回増加します。(クローズ操作は、このようなトランザクションがすべて終了するまで待機します。)

- Threads_cached

  スレッドキャッシュ内のスレッド数。

- Threads_connected

  現在開いている接続の数。

- Threads_created

  接続を処理するために作成されたスレッドの数。Threads_created が大きい場合、thread_cache_size の値を大きくした方がよい場合もあります。キャッシュミス率は Threads_created/Connections として計算できます。

- Threads_running

  スリープ状態ではないスレッド数。

- Uptime

  サーバーが作動している秒数。

- Uptime_since_flush_status

  最新の FLUSH STATUS ステートメントの秒数。

5.1.7 サーバー SQL モード

MySQL Server は異なる SQL モードで動作でき、sql_mode システム変数の値に応じて異なるクライアントにこれらの異なるモードを適用できます。DBA はサイトサーバーの動作要件に一致するグローバル SQL モードを設定でき、各アプリケーションはアプリケーションのセッション SQL モードをアプリケーション独自の要件に設定できます。

モードは MySQL がサポートする SQL 構文と、MySQL が実行するデータ検証に影響します。これにより、MySQL をさまざまな環境で使用したり、MySQL をほかのデータベースサーバーと一緒に使用したりすることが、さらに容易になります。

- Setting the SQL Mode
- The Most Important SQL Modes
- Full List of SQL Modes
- Strict SQL Mode
- Combination SQL Modes

MySQL のサーバー SQL モードについてのよくある質問に対する回答は、セクションA.3「MySQL 5.6 FAQ: サーバー SQL モード」を参照してください。

InnoDB テーブルを操作するとき、innodb_strict_mode システム変数についても考慮してください。これによって、InnoDB テーブルの追加のエラー検査が可能になります。

SQL モードの設定

MySQL 5.6.6 以降でのデフォルトの SQL モードは NO_ENGINE_SUBSTITUTION で、MySQL 5.6.5 以前では、これは空白です (モードの設定なし)。

サーバー起動時に SQL モードを設定するには、コマンド行で --sql-mode="modes" オプションを使用するか、my.cnf (Unix オペレーティングシステム) または my.ini (Windows) などのオプションファイル内で sql-mode="modes" を使用します。modes は、カンマで区切られるさまざまなモードのリストです。SQL モードを明示的にクリアするには、コマンド行で --sql-mode="" を使用するかオプションファイル内で sql-mode="" を使用して、SQL モードを空の文字列に設定します。

> **注記**
>
> MySQL インストールプログラムはインストールプロセス中に SQL モードを構成することがあります。たとえば、mysql_install_db は、my.cnf という名前のデフォルトオプションファイルを基本インストールディレクトリに作成します。このファイルには、SQL モードを設定する行が含まれています。セクション4.4.3「mysql_install_db — MySQL データディレクトリの初期化」を参照してください。
>
> SQL モードがデフォルトまたは期待されているモードと異なる場合、サーバーが起動時に読み取るオプションファイル内の設定を確認してください。

SQL モードを実行時に変更するには、SET ステートメントを使用して、グローバルまたはセッションの sql_mode システム変数を設定します。

```
SET GLOBAL sql_mode = 'modes';
SET SESSION sql_mode = 'modes';
```

GLOBAL 変数を設定するには SUPER 権限が必要で、この設定はその時点以降に接続するすべてのクライアントの動作に影響します。SESSION 変数を設定すると、現在のクライアントにのみ影響します。すべてのクライアントは、自分のセッションの sql_mode 値をいつでも変更できます。

現在のグローバルまたはセッションの sql_mode 値を確認するには、次のステートメントを使用します。

```
SELECT @@GLOBAL.sql_mode;
SELECT @@SESSION.sql_mode;
```

> **重要**
>
> SQL モードおよびユーザー定義のパーティショニング　パーティション化されたテーブルを作成してデータを挿入したあとでサーバー SQL モードを変更すると、このようなテーブルの動作が大きく変更される可能性があり、データが失われたり破損したりすることがあります。ユーザー定義のパーティショニングを使用したテーブルを作成したら、SQL モードを変更しないことを強くお勧めします。
>
> パーティション化されたテーブルをレプリケーションするとき、マスターとスレーブの SQL モードが違うことが問題につながることもあります。最適な結果を得るために、マスターとスレーブとで常に同じサーバー SQL モードを使用してください。
>
> 詳細については、セクション19.6「パーティショニングの制約と制限」を参照してください。

もっとも重要な SQL モード

次に、多くの場合でもっとも重要な sql_mode 値を示します。

- ANSI

  このモードは、構文および動作が標準の SQL にさらに緊密に準拠するように変更します。これは、このセクションの末尾にリストされている、特殊な組み合わせモードの 1 つです。

- STRICT_TRANS_TABLES

  値を指定したとおりにトランザクションテーブルに挿入できない場合、ステートメントを中止します。非トランザクションテーブルの場合、値が単一行ステートメントで発生するか、複数行ステートメントの先頭行で発生した場合、ステートメントを中止します。詳細については、このセクションのあとの方で説明します。

- TRADITIONAL

  MySQL を 「従来型の」 SQL データベースシステムのように動作させます。このモードを簡単に説明すると、カラムに不正な値を挿入したときに「警告ではなくエラーを返し」ます。これは、このセクションの末尾にリストされている、特殊な組み合わせモードの 1 つです。

  > **注記**
  >
  > INSERT または UPDATE は、エラーが見つかるとすぐに中止します。非トランザクションストレージエンジンを使用している場合、これは期待する動作でない場合もあります。エラーの前に行われたデータ変更はロールバックされず、「部分的に実行された」更新になることがあるためです。

このマニュアルの「厳密モード」とは、STRICT_TRANS_TABLES または STRICT_ALL_TABLES のいずれかあるいは両方が有効なモードを意味します。

SQL モードの完全なリスト

次のリストは、サポートされるすべての SQL モードについて説明しています。

- ALLOW_INVALID_DATES

  日付の完全な検査を実行しません。月が 1 から 12 までの範囲にあることと、日が 1 から 31 までの範囲にあることのみ検査します。これは、年、月、および日を 3 つの異なるフィールドで取得し、ユーザーが挿入したデータを (日付の検証を行わずに) そのまま格納する Web アプリケーションでは非常に便利です。このモードは DATE および DATETIME カラムに適用されます。TIMESTAMP カラムは有効な日付が常に必要なため、このカラムには適用されません。

  サーバーは、月と日の値がそれぞれ 1 から 12 または 1 から 31 の範囲にあることだけでなく、正しい値であることを要求します。厳密モードが無効になっていると、'2004-04-31' のような無効な日付は '0000-00-00' に変換され、警告メッセージが表示されます。厳密モードが有効なときは、無効な日付によってエラーが発生します。このような日付を許可するには、ALLOW_INVALID_DATES を有効にします。

- ANSI_QUOTES

  「"」 を (「`」 引用符文字のような) 識別子引用符文字として扱い、文字列引用符文字として扱いません。このモードを有効にして、識別子を引用するために 「`」 を引き続き使用できます。ANSI_QUOTES を有効にすると、二重引用符を使用してリテラル文字列を引用できません (リテラル文字列が識別子として解釈されるため)。

- ERROR_FOR_DIVISION_BY_ZERO

  ERROR_FOR_DIVISION_BY_ZERO モードは、MOD(N,0) を含むゼロ除算の処理に影響します。データ変更操作 (INSERT、UPDATE) の場合、この効果は厳密 SQL モードが有効であるかどうかにもよります。

  - このモードが有効でない場合、ゼロによる除算は NULL を挿入し、警告は生成されません。
  - このモードが有効な場合、ゼロによる除算は NULL を挿入し、警告が生成されます。
  - このモードおよび厳密モードが有効な場合、ゼロによる除算はエラーを生成しますが、IGNORE も指定されている場合は例外です。INSERT IGNORE および UPDATE IGNORE の場合、ゼロによる除算は NULL を挿入し、警告が生成されます。

  SELECT の場合、ゼロによる除算は NULL を返します。ERROR_FOR_DIVISION_BY_ZERO を有効にすると、厳密モードが有効かどうかに関係なく警告も生成されます。

  MySQL 5.6.17 以降では、ERROR_FOR_DIVISION_BY_ZERO は非推奨となり、これを含むように sql_mode 値を設定すると警告が生成されます。MySQL 5.7 では、このモードは何も行いません。その代わり、この効果は厳密モードの効果に含められています。

- HIGH_NOT_PRECEDENCE

NOT 演算子の存在によって、NOT a BETWEEN b AND c のような式は NOT (a BETWEEN b AND c) として構文解析されます。一部の古い MySQL バージョンでは、この式は (NOT a) BETWEEN b AND c として構文解析されます。優先順位を高める以前の動作は、HIGH_NOT_PRECEDENCE の SQL モードを有効にすることによって取得できます。

```
mysql> SET sql_mode = '';
mysql> SELECT NOT 1 BETWEEN -5 AND 5;
       -> 0
mysql> SET sql_mode = 'HIGH_NOT_PRECEDENCE';
mysql> SELECT NOT 1 BETWEEN -5 AND 5;
       -> 1
```

- IGNORE_SPACE

  関数名と 「(」 文字の間にスペースを許可します。これにより、組み込み関数名が予約語として扱われます。その結果、関数名と同じ識別子は、セクション9.2「スキーマオブジェクト名」に記載されているように引用符で囲む必要があります。たとえば、COUNT() 関数があるため、次のステートメントで count をテーブル名として使用すると、エラーが発生します。

  ```
  mysql> CREATE TABLE count (i INT);
  ERROR 1064 (42000): You have an error in your SQL syntax
  ```

  テーブル名を引用符で囲んでください。

  ```
  mysql> CREATE TABLE `count` (i INT);
  Query OK, 0 rows affected (0.00 sec)
  ```

  IGNORE_SPACE SQL モードは、ユーザー定義関数またはストアドファンクションではなく、組み込み関数に適用されます。IGNORE_SPACE が有効かどうかにかかわらず、UDF またはストアドファンクション名のあとにスペースを入れることが常に許可されます。

  IGNORE_SPACE に関する詳細は、セクション9.2.4「関数名の構文解析と解決」を参照してください。

- NO_AUTO_CREATE_USER

  認証情報が指定される場合を除き、ほかの方法で実行される場合は、GRANT ステートメントで新規ユーザーを自動的に作成しません。ステートメントは IDENTIFIED BY を使用した空以外のパスワードを指定するか、IDENTIFIED WITH を使用した認証プラグインを指定する必要があります。

- NO_AUTO_VALUE_ON_ZERO

  NO_AUTO_VALUE_ON_ZERO は AUTO_INCREMENT カラムの処理に影響します。通常は、NULL または 0 をカラムに挿入することによって、カラムの次のシーケンス番号を生成します。NO_AUTO_VALUE_ON_ZERO は 0 のこの動作を抑制するため、NULL のみが次のシーケンス番号を生成します。

  このモードは、テーブルの AUTO_INCREMENT カラムに 0 が格納されている場合に便利なことがあります。(ただし、0 を格納することは、推奨される方法ではありません。)たとえば、mysqldump でテーブルをダンプして、テーブルをリロードする場合、MySQL は通常、0 という値を検出すると、新たなシーケンス番号を生成するため、その結果、ダンプされたものとは異なる内容を持つテーブルになります。ダンプファイルをリロードする前に NO_AUTO_VALUE_ON_ZERO を有効にすると、この問題が解決されます。この問題を防ぐために、mysqldump には現在、NO_AUTO_VALUE_ON_ZERO を有効にするステートメントがその出力に自動的に含まれるようになりました。

- NO_BACKSLASH_ESCAPES

  バックスラッシュ文字 (「\」) を文字列内でエスケープ文字として使用することを無効にします。このモードを有効にすると、バックスラッシュはほかの文字のように通常の文字になります。

- NO_DIR_IN_CREATE

  テーブルを作成するとき、INDEX DIRECTORY および DATA DIRECTORY ディレクティブをすべて無視します。このオプションは、スレーブレプリケーションサーバー上で役立ちます。

- NO_ENGINE_SUBSTITUTION

  CREATE TABLE または ALTER TABLE などのステートメントが無効またはコンパイルされていないストレージエンジンを指定したとき、デフォルトのストレージエンジンの自動置換を制御します。

  ストレージエンジンは実行時にプラガブルであるため、利用できないエンジンも同様に扱われます。

NO_ENGINE_SUBSTITUTION を無効にすると、CREATE TABLE については、目的のエンジンが利用できない場合にデフォルトエンジンが使用されて警告が発生します。ALTER TABLE では、警告が発生してテーブルは変更されません。

NO_ENGINE_SUBSTITUTION を有効にすると、目的のエンジンが利用できない場合にエラーが発生し、テーブルは作成または変更されません。

- NO_FIELD_OPTIONS

  SHOW CREATE TABLE の出力に MySQL 固有のカラムオプションを出力しません。このモードはポータビリティモードで mysqldump によって使用されます。

- NO_KEY_OPTIONS

  SHOW CREATE TABLE の出力で MySQL 固有のインデックスオプションを出力しません。このモードはポータビリティモードで mysqldump によって使用されます。

- NO_TABLE_OPTIONS

  SHOW CREATE TABLE の出力で MySQL 固有のテーブルオプション (ENGINE など) を出力しません。このモードはポータビリティモードで mysqldump によって使用されます。

- NO_UNSIGNED_SUBTRACTION

  デフォルトでは、いずれかのオペランドが UNSIGNED の場合、整数オペランド間の減算は UNSIGNED の結果を生成します。NO_UNSIGNED_SUBTRACTION が有効な場合、いずれかのオペランドが符号なしであっても、減算の結果は符号付きになります。たとえば、テーブル t1 のカラム c2 のタイプと、テーブル t2 のカラム c2 のタイプを比較します。

```
mysql> SET sql_mode='';
mysql> CREATE TABLE test (c1 BIGINT UNSIGNED NOT NULL);
mysql> CREATE TABLE t1 SELECT c1 - 1 AS c2 FROM test;
mysql> DESCRIBE t1;
+-------+---------------------+------+-----+---------+-------+
| Field | Type                | Null | Key | Default | Extra |
+-------+---------------------+------+-----+---------+-------+
| c2    | bigint(21) unsigned |      |     | 0       |       |
+-------+---------------------+------+-----+---------+-------+

mysql> SET sql_mode='NO_UNSIGNED_SUBTRACTION';
mysql> CREATE TABLE t2 SELECT c1 - 1 AS c2 FROM test;
mysql> DESCRIBE t2;
+-------+------------+------+-----+---------+-------+
| Field | Type       | Null | Key | Default | Extra |
+-------+------------+------+-----+---------+-------+
| c2    | bigint(21) |      |     | 0       |       |
+-------+------------+------+-----+---------+-------+
```

このことは、BIGINT UNSIGNED がすべてのコンテキストで 100% 使用可能ではないことを意味します。セクション12.10「キャスト関数と演算子」を参照してください。

```
mysql> SET sql_mode = '';
mysql> SELECT CAST(0 AS UNSIGNED) - 1;
+------------------------+
| CAST(0 AS UNSIGNED) - 1 |
+------------------------+
|   18446744073709551615 |
+------------------------+

mysql> SET sql_mode = 'NO_UNSIGNED_SUBTRACTION';
mysql> SELECT CAST(0 AS UNSIGNED) - 1;
+------------------------+
| CAST(0 AS UNSIGNED) - 1 |
+------------------------+
|                     -1 |
+------------------------+
```

- NO_ZERO_DATE

  NO_ZERO_DATE モードは、サーバーが '0000-00-00' を有効な日付として許可するかどうかに影響します。この影響は、厳密 SQL モードが有効かどうかにも依存します。

  - このモードが有効でない場合、'0000-00-00' は許可され、挿入によって警告が生成されません。

- このモードが有効な場合、'0000-00-00' は許可され、挿入によって警告が生成されます。
- このモードおよび厳密モードが有効な場合、IGNORE も指定されている場合を除き、'0000-00-00' は許可されず、挿入によってエラーが生成されます。INSERT IGNORE および UPDATE IGNORE の場合、'0000-00-00' は許可され、挿入によって警告が生成されます。

MySQL 5.6.17 以降では、NO_ZERO_DATE は非推奨となり、これを含むように sql_mode 値を設定すると警告が生成されます。MySQL 5.7 では、このモードは何も行いません。その代わり、この効果は厳密モードの効果に含められています。

- NO_ZERO_IN_DATE

NO_ZERO_IN_DATE モードは、年の部分は非ゼロであるが月または日の部分が 0 である日付をサーバーが許可するかどうかに影響します。(このモードは '2010-00-01' や '2010-01-00' などの日付に影響しますが、'0000-00-00' には影響しません。サーバーが '0000-00-00' を許可するかどうかを制御するには、NO_ZERO_DATE モードを使用してください。)NO_ZERO_IN_DATE の影響は、厳密 SQL モードが有効かどうかにも依存します。

- このモードが有効でない場合、ゼロ部分を含む日付は許可され、挿入によって警告が生成されません。
- このモードが有効な場合、ゼロ部分を含む日付は '0000-00-00' として挿入され、警告が生成されます。
- このモードおよび厳密モードが有効な場合は、IGNORE も指定されている場合を除き、ゼロ部分を含む日付は許可されず、挿入によってエラーが生成されます。INSERT IGNORE および UPDATE IGNORE の場合、ゼロ部分を含む日付は '0000-00-00' として挿入され、警告が生成されます。

MySQL 5.6.17 以降では、NO_ZERO_IN_DATE は非推奨となり、これを含むように sql_mode 値を設定すると警告が生成されます。MySQL 5.7 では、このモードは何も行いません。その代わり、この効果は厳密モードの効果に含められています。

- ONLY_FULL_GROUP_BY

GROUP BY 句で名前が指定されていない非集約カラムを、選択リスト、HAVING 条件、または (MySQL 5.6.5 以降で) ORDER リストが参照するクエリーを拒否します。

ONLY_FULL_GROUP_BY が有効な場合、次のクエリーは無効です。1 番目は、選択リスト内の非集約の address カラムが GROUP BY 句で名前を指定されておらず、2 番目は、HAVING 句の max_age が GROUP BY 句で名前を指定されていないため、ともに無効になります。

```
mysql> SELECT name, address, MAX(age) FROM t GROUP BY name;
ERROR 1055 (42000): 't.address' isn't in GROUP BY
```

```
mysql> SELECT name, MAX(age) AS max_age FROM t GROUP BY name
    -> HAVING max_age < 30;
Empty set (0.00 sec)
ERROR 1463 (42000): Non-grouping field 'max_age' is used in HAVING clause
```

2 番目の例では、HAVING MAX(age) を使用するようにクエリーを書き換えることで、集約関数で名前を指定されているカラムが参照されるようになります。(max_age は集約関数そのものであるため失敗します。)

クエリーに集約関数があって GROUP BY 句がない場合、ONLY_FULL_GROUP_BY が有効なときに、クエリーは選択リストまたは ORDER BY リストに非集約カラムを含めることができません。

```
mysql> SELECT name, MAX(age) FROM t;
ERROR 1140 (42000): Mixing of GROUP columns (MIN(),MAX(),COUNT(),...)
with no GROUP columns is illegal if there is no GROUP BY clause
```

追加の説明については、セクション12.19.3「MySQL での GROUP BY の処理」を参照してください。

- PAD_CHAR_TO_FULL_LENGTH

デフォルトでは、末尾のスペースは、取得時に CHAR カラム値から削除されます。PAD_CHAR_TO_FULL_LENGTH が有効な場合、削除は行われず、取得された CHAR 値は完全な長さになるまでパディングされます。このモードは VARCHAR カラムには適用されず、この場合、末尾のスペースは取得時に保持されます。

```
mysql> CREATE TABLE t1 (c1 CHAR(10));
Query OK, 0 rows affected (0.37 sec)

mysql> INSERT INTO t1 (c1) VALUES('xy');
Query OK, 1 row affected (0.01 sec)

mysql> SET sql_mode = '';
Query OK, 0 rows affected (0.00 sec)

mysql> SELECT c1, CHAR_LENGTH(c1) FROM t1;
+------+----------------+
| c1   | CHAR_LENGTH(c1) |
+------+----------------+
| xy   |              2 |
+------+----------------+
1 row in set (0.00 sec)

mysql> SET sql_mode = 'PAD_CHAR_TO_FULL_LENGTH';
Query OK, 0 rows affected (0.00 sec)

mysql> SELECT c1, CHAR_LENGTH(c1) FROM t1;
+------------+----------------+
| c1         | CHAR_LENGTH(c1) |
+------------+----------------+
| xy         |             10 |
+------------+----------------+
1 row in set (0.00 sec)
```

- PIPES_AS_CONCAT

  || を、OR のシノニムとしてではなく (CONCAT() と同様に) 文字列連結演算子として扱います。

- REAL_AS_FLOAT

  REAL を FLOAT のシノニムとして扱います。デフォルトでは、MySQL は REAL を DOUBLE のシノニムとして扱います。

- STRICT_ALL_TABLES

  すべてのストレージエンジンについて厳密モードを有効にします。無効なデータ値は拒否されます。追加の詳細は、あとの方で説明します。

- STRICT_TRANS_TABLES

  トランザクションストレージエンジンの厳密モードを有効にし、可能な場合は非トランザクションストレージエンジンでも有効にします。追加の詳細は、あとの方で説明します。

厳密な SQL モード

厳密モードは、MySQL が INSERT や UPDATE などのデータ変更ステートメントで無効な値または欠落した値を処理する方法を制御します。値はいくつかの理由で無効になることがあります。たとえば、カラムに対して正しくないデータ型を持っていたり、範囲外であったりすることがあります。値の欠落が発生するのは、挿入される新しい行の非 NULL カラムに値が含まれておらず、そのカラムに明示的な DEFAULT 句が定義されていない場合です。(NULL カラムの場合、値が欠落しているときは NULL が挿入されます。)

厳密モードが有効でない場合、MySQL は無効または欠落した値に対して調整された値を挿入し、警告を生成します (セクション13.7.5.41「SHOW WARNINGS 構文」を参照してください)。厳密モードでは、INSERT IGNORE または UPDATE IGNORE を使用すると、この動作を実行できます。

データを変更しない SELECT などのステートメントの場合、厳密モードでは無効な値はエラーでなく警告を生成します。

厳密モードは、外部キー制約が検査されるかどうかに影響されません。foreign_key_checks を検査に使用できます。(セクション5.1.4「サーバーシステム変数」を参照してください。)

厳密な SQL モードは、STRICT_ALL_TABLES または STRICT_TRANS_TABLES のいずれかが有効な場合に有効になりますが、これらのモードの影響はいくらか異なります。

- トランザクションテーブルの場合、STRICT_ALL_TABLES または STRICT_TRANS_TABLES のいずれかが有効なとき、データ変更ステートメント内の無効な値または欠落した値に対してエラーが発生します。ステートメントは中止されてロールバックされます。
- 非トランザクションテーブルの場合、挿入または更新される最初の行に不適切な値があるとき、どちらのモードでも動作は同じになり、ステートメントが中止されて、テーブルはそのまま変更されません。ステートメン

トが複数行を挿入または変更し、2 行目以降に不適切な値がある場合、どちらの厳密モードが有効になっているかによって結果は異なります。

- STRICT_ALL_TABLES では、MySQL はエラーを返し、残りの行を無視します。ただし、それより前の行が挿入または更新されているため、結果は部分更新となります。これを防ぐには、テーブルを変更することなく中止できる単一行ステートメントを使用します。
- STRICT_TRANS_TABLES では、MySQL は無効な値をカラムについてのもっとも近い有効な値に変換し、調整された値を挿入します。値が欠落している場合、MySQL はカラムデータ型の暗黙のデフォルト値を挿入します。いずれの状況でも MySQL はエラーでなく警告を生成し、ステートメントの処理を続行します。暗黙的なデフォルトについては、セクション11.6「データ型デフォルト値」に記載されています。

厳密モードは ERROR_FOR_DIVISION_BY_ZERO、NO_ZERO_DATE、および NO_ZERO_IN_DATE モードに関連して、ゼロによる除算、ゼロ日付、および日付内のゼロの処理にも影響を及ぼします。詳細については、これらのモードの説明を参照してください。

組み合わせ SQL モード

次の特殊なモードは、前リストのモード値の組み合わせを表す省略表現として提供されています。

- ANSI

  REAL_AS_FLOAT、PIPES_AS_CONCAT、ANSI_QUOTES、IGNORE_SPACE と同等です。

  また ANSI モードは、外部参照 S(outer_ref) を持つ設定関数 S が、外部参照が解決される外部クエリー内で集約できない場合のクエリーに、サーバーがエラーを返します。このようなクエリーを次に示します。

  ```
  SELECT * FROM t1 WHERE t1.a IN (SELECT MAX(t1.b) FROM t2 WHERE ...);
  ```

  ここで、MAX(t1.b) はそのクエリーの WHERE 句に指定されているため、外部クエリーで集約できません。標準的な SQL では、この状況ではエラーになります。ANSI モードが有効でない場合、サーバーはそのようなクエリー内の S(outer_ref) を、S(const) を解釈する同じ方法で扱います。

  セクション1.7「MySQL の標準への準拠」を参照してください。

- DB2

  PIPES_AS_CONCAT、ANSI_QUOTES、IGNORE_SPACE、NO_KEY_OPTIONS、NO_TABLE_OPTIONS、NO_FIELD_OPT
  と同等です。

- MAXDB

  PIPES_AS_CONCAT、ANSI_QUOTES、IGNORE_SPACE、NO_KEY_OPTIONS、NO_TABLE_OPTIONS、NO_FIELD_OPT
  と同等です。

- MSSQL

  PIPES_AS_CONCAT、ANSI_QUOTES、IGNORE_SPACE、NO_KEY_OPTIONS、NO_TABLE_OPTIONS、NO_FIELD_OPT
  と同等です。

- MYSQL323

  NO_FIELD_OPTIONS、HIGH_NOT_PRECEDENCE と同等です。

- MYSQL40

  NO_FIELD_OPTIONS、HIGH_NOT_PRECEDENCE と同等です。

- ORACLE

  PIPES_AS_CONCAT、ANSI_QUOTES、IGNORE_SPACE、NO_KEY_OPTIONS、NO_TABLE_OPTIONS、NO_FIELD_OPT
  と同等です。

- POSTGRESQL

  PIPES_AS_CONCAT、ANSI_QUOTES、IGNORE_SPACE、NO_KEY_OPTIONS、NO_TABLE_OPTIONS、NO_FIELD_OPT
  と同等です。

- TRADITIONAL

STRICT_TRANS_TABLES、STRICT_ALL_TABLES、NO_ZERO_IN_DATE、NO_ZERO_DATE、ERROR_FOR_DIVISIO
および NO_ENGINE_SUBSTITUTION と同等です。

5.1.8 サーバープラグイン

MySQL は、サーバーコンポーネントの作成を可能にするプラグイン API をサポートします。プラグインはサーバー起動時にロードしたり、実行時にサーバーを再起動せずにロードおよびアンロードしたりできます。このインタフェースによってサポートされるコンポーネントは、これらに限定されませんが、ストレージエンジン、全文パーサープラグイン、パーティショニングサポート、およびサーバー拡張機能があります。

5.1.8.1 プラグインのインストールおよびアンインストール

サーバープラグインを使用する前に、サーバープラグインをサーバーにロードする必要があります。MySQL では、ユーザーはサーバー起動時または実行時にプラグインをロードできます。また、ロードされたプラグインの有効化を起動時に制御したり、実行時にプラグインをアンロードしたりすることも可能です。

- プラグインのインストール
- プラグインの有効化の制御
- プラグインのアンインストール

プラグインのインストール

サーバープラグインを使用する前に、サーバープラグインがサーバーに認識されている必要があります。ここで説明されているように、プラグインはいくつかの方法で認識させることができます。次の説明で、plugin_name は innodb や csv などのプラグイン名を表します。

組み込みプラグイン:

サーバーに組み込まれているプラグインは、サーバーによって自動的に認識されます。通常、サーバーはプラグインを起動時に有効にしますが、--plugin_name オプションでこれを変更できます。

mysql.plugin テーブルに登録済みのプラグイン:

mysql.plugin テーブルはプラグインのレジストリの役割を担っています。サーバーは通常、テーブルにリストされている各プラグインを起動時に有効にしますが、特定のプラグインを有効化するかどうかは --plugin_name オプションで変更できます。サーバーが --skip-grant-tables オプションで開始された場合、サーバーはこのテーブルを参照せず、ここにリストされているプラグインをロードしません。

コマンド行オプションで名前を指定されたプラグイン:

プラグインライブラリファイルに存在するプラグインは、--plugin-load オプションを使用するとサーバー起動時にロードできます。通常、サーバーはプラグインを起動時に有効にしますが、--plugin_name オプションでこれを変更できます。

オプション値は、セミコロンで区切られた name=plugin_library のペアのリストです。それぞれの name はプラグインの名前で、plugin_library はプラグインコードを含む共有ライブラリの名前です。プラグイン名を前に付けずにプラグインライブラリを指定した場合、サーバーはライブラリ内のすべてのプラグインをロードします。各ライブラリファイルは、plugin_dir システム変数によって指定されるディレクトリに配置されている必要があります。

このオプションは、mysql.plugin テーブルにプラグインを登録しません。後続の再起動では、--plugin-load がふたたび指定された場合のみ、サーバーはプラグインを再ロードします。つまり、このオプションは 1 回のサーバー起動にしか継続しない 1 回のインストールに影響します。

--plugin-load は、(サーバーが mysql.plugin テーブルを無視するようにする) --skip-grant-tables が指定されている場合でも、プラグインのロードを可能にします。--plugin-load はまた、プラグインが実行時にロードできない構成で、プラグインを起動時にロードできるようにします。

--plugin-load-add オプションは --plugin-load オプションを補完します。--plugin-load-add は、起動時にロードされるプラグインのセットに 1 つまたは複数のプラグインを追加します。引数の形式は --plugin-load と同じです。--plugin-load-add を使用すれば、大量のプラグインのセットを、長くて扱いにくい単一の --plugin-load 引数として指定することを回避できます。--plugin-load-add は --plugin-load がなくても指定できますが、--plugin-load は口ー

ドするプラグインのセットをリセットするため、--plugin-load の前にあるすべての --plugin-load-add のインスタンスは無効になります。つまり、次のオプションの場合、

```
--plugin-load=x --plugin-load-add=y
```

上記は次のオプションと同等です。

```
--plugin-load="x;y"
```

ただし、次のオプションの場合、

```
--plugin-load-add=y --plugin-load=x
```

上記は次のオプションと同等です。

```
--plugin-load=x
```

INSTALL PLUGIN ステートメントでインストールされるプラグイン:

プラグインライブラリファイルに存在するプラグインは、INSTALL PLUGIN ステートメントで実行時にロードできます。このステートメントはさらにプラグインを mysql.plugin テーブルに登録し、その後再起動してサーバーからロードします。このため、INSTALL PLUGIN には、mysql.plugin テーブルに対する INSERT 権限が必要です。

プラグインが --plugin-load オプションと mysql.plugin テーブルの両方を使用して指定された場合、サーバーは起動しますが、次のメッセージがエラーログに書き込まれます。

```
100310 19:15:44 [ERROR] Function 'plugin_name' already exists
100310 19:15:44 [Warning] Couldn't load plugin named 'plugin_name'
with soname 'plugin_object_file'.
```

例: --plugin-load オプションはサーバー起動時にプラグインをインストールします。myplugin という名前のプラグインを somepluglib.so という名前のプラグインライブラリファイルにインストールするには、my.cnf ファイル内で次の行を使用します。

```
[mysqld]
plugin-load=myplugin=somepluglib.so
```

この場合、プラグインは mysql.plugin に登録されません。--plugin-load オプションを付けずにサーバーを再起動すると、プラグインは起動時にロードされません。

一方、INSTALL PLUGIN ステートメントでは、サーバーは起動時にライブラリファイルからプラグインコードをロードします。

```
mysql> INSTALL PLUGIN myplugin SONAME 'somepluglib.so';
```

また、INSTALL PLUGIN は「永続的な」プラグイン登録を実行します。サーバーはプラグインを mysql.plugin テーブルにリストすることで、その後サーバーを再起動するとプラグインがロードされます。

多くのプラグインは、サーバーの起動時または実行時のいずれかにロードできます。ただし、サーバー起動時にプラグインがロードおよび初期化されるように設計されている場合、INSTALL PLUGIN ではなく --plugin-load を使用してください。

プラグインがロードされているとき、プラグインについての情報は、INFORMATION_SCHEMA.PLUGINS テーブルや SHOW PLUGINS ステートメントなどのいくつかのソースから実行時に入手できます。詳細については、セクション5.1.8.2「サーバープラグイン情報の取得」を参照してください。

プラグインの有効化の制御

サーバーが起動するときにプラグインを認識している場合 (たとえば、プラグインが --plugin-load オプションを使用して指定されていたり、mysql.plugin テーブルに登録されていたりする場合)、サーバーはデフォルトでは、プラグインをロードして有効にします。プラグインのあとに --plugin_name[=value] 起動オプションを使用すると、これらのプラグインの有効化を制御できます。次の説明で、plugin_name は innodb や csv などのプラグイン名を表します。ほかのオプションと同じように、オプション名のダッシュとアンダースコアは交換可能です。たとえば、--my_plugin=ON と --my-plugin=ON は同等です。

- --plugin_name=OFF

  プラグインを無効にするようサーバーに指示します。

- --plugin_name[=ON]

  プラグインを有効にするようサーバーに指示します。(値を付けずにオプションを --plugin_name と指定しても効果は同じです。)プラグインが初期化に失敗した場合、サーバーはプラグインを無効にして実行します。

- --plugin_name=FORCE

  プラグインを有効にするようサーバーに指示しますが、プラグインの初期化が失敗した場合、サーバーは開始しません。つまり、このオプションはプラグインを有効にしてサーバーを実行するか、何もしないかのいずれかを強制します。

- --plugin_name=FORCE_PLUS_PERMANENT

  FORCE と似ていますが、さらにプラグインが実行時にアンロードされないようにします。ユーザーが UNINSTALL PLUGIN を使用してこの操作を実行しようとすると、エラーが発生します。

OFF、ON、FORCE、および FORCE_PLUS_PERMANENT の値は大文字小文字を区別しません。

プラグインの有効化状態は、INFORMATION_SCHEMA.PLUGINS テーブルの LOAD_OPTION カラムに表示されます。

かりに、CSV、BLACKHOLE、および ARCHIVE がプラガブルな組み込みストレージエンジンで、サーバーが起動時にこれらをロードするようにし、次の条件が与えられているとします。つまり、CSV の初期化に失敗しても、サーバーは実行を許可されるが、BLACKHOLE の初期化に成功することが必要で、ARCHIVE が無効化されていなければならないという条件です。これを満たすには、オプションファイル内で次の行を使用します。

```
[mysqld]
csv=ON
blackhole=FORCE
archive=OFF
```

--enable-plugin_name オプション形式は --plugin_name=ON のシノニムとしてサポートされます。--disable-plugin_name および --skip-plugin_name オプション形式は、--plugin_name=OFF のシノニムとしてサポートされます。

プラグインが OFF で明示的に無効化されるか、ON で有効化されたが初期化に失敗したため暗黙的に無効化される場合、プラグインを必要とするサーバー操作について変更されます。たとえば、プラグインがストレージエンジンを実装する場合、そのストレージエンジンの既存のテーブルはアクセス不能になり、そのストレージエンジンに対して新規テーブルを作成しようとすると、代わりにエラーを発生させる NO_ENGINE_SUBSTITUTION SQL モードが有効になっていないかぎり、デフォルトのストレージエンジンを使用するテーブルが生成されます。

プラグインの無効化は、ほかのオプションの調整を要することもあります。たとえば、--skip-innodb を使用してサーバーを開始して、InnoDB を無効化した場合、ほかの innodb_xxx オプションを起動コマンドから省略することが必要になる可能性があります。さらに、InnoDB はデフォルトのストレージエンジンであるため、使用可能な別のストレージエンジンを --default_storage_engine で指定しないかぎり開始しません。MySQL 5.6.3 以降では、--default_tmp_storage_engine も設定する必要があります。

プラグインのアンインストール

サーバーに認識されているプラグインは、UNINSTALL PLUGIN ステートメントで実行時にアンインストールすると無効化できます。このステートメントはプラグインをアンロードし、プラグインが mysql.plugin テーブルに登録されている場合はそこから削除します。このため、UNINSTALL PLUGIN ステートメントには、mysql.plugin テーブルに対する DELETE 権限が必要です。プラグインがテーブルに登録されていない場合、サーバーはその後の再起動でプラグインを自動的にロードしません。

UNINSTALL PLUGIN は、プラグインが INSTALL PLUGIN または --plugin-load のいずれでロードされたかに関係なく、プラグインをアンロードできます。

UNINSTALL PLUGIN には次の例外が該当します。

- サーバーに組み込まれているプラグインをアンロードできません。これらは INFORMATION_SCHEMA.PLUGINS または SHOW PLUGINS からの出力で、ライブラリ名が NULL のプラグインとして識別できます。

- 実行時にプラグインをアンロードしないようにする --plugin_name=FORCE_PLUS_PERMANENT を使用すると、サーバーが開始された目的のプラグインをアンロードできません。これらは INFORMATION_SCHEMA.PLUGINS テーブルの LOAD_OPTION カラムから識別できます。

5.1.8.2 サーバープラグイン情報の取得

サーバーにインストールされているプラグインを調べるにはいくつかの方法があります。

- INFORMATION_SCHEMA.PLUGINS テーブルには、ロードされているプラグインの行が含まれています。PLUGIN_LIBRARY 値が NULL のプラグインは組み込み型であり、アンロードできません。

```
mysql> SELECT * FROM information_schema.PLUGINS\G
*************************** 1. row ***************************
           PLUGIN_NAME: binlog
        PLUGIN_VERSION: 1.0
         PLUGIN_STATUS: ACTIVE
           PLUGIN_TYPE: STORAGE ENGINE
   PLUGIN_TYPE_VERSION: 50158.0
        PLUGIN_LIBRARY: NULL
PLUGIN_LIBRARY_VERSION: NULL
         PLUGIN_AUTHOR: MySQL AB
    PLUGIN_DESCRIPTION: This is a pseudo storage engine to represent the binlog in a transaction
        PLUGIN_LICENSE: GPL
          LOAD_OPTION: FORCE
...
*************************** 10. row ***************************
           PLUGIN_NAME: InnoDB
        PLUGIN_VERSION: 1.0
         PLUGIN_STATUS: ACTIVE
           PLUGIN_TYPE: STORAGE ENGINE
   PLUGIN_TYPE_VERSION: 50158.0
        PLUGIN_LIBRARY: ha_innodb_plugin.so
PLUGIN_LIBRARY_VERSION: 1.0
         PLUGIN_AUTHOR: Innobase Oy
    PLUGIN_DESCRIPTION: Supports transactions, row-level locking,
                        and foreign keys
        PLUGIN_LICENSE: GPL
          LOAD_OPTION: ON
...
```

- SHOW PLUGINS ステートメントは、ロードされている各プラグインの行を表示します。Library 値が NULL のプラグインは組み込み型であり、アンロードできません。

```
mysql> SHOW PLUGINS\G
*************************** 1. row ***************************
   Name: binlog
 Status: ACTIVE
   Type: STORAGE ENGINE
Library: NULL
License: GPL
...
*************************** 10. row ***************************
   Name: InnoDB
 Status: ACTIVE
   Type: STORAGE ENGINE
Library: ha_innodb_plugin.so
License: GPL
...
```

- mysql.plugin テーブルには、INSTALL PLUGIN で登録されたプラグインを表示します。このテーブルにはプラグイン名およびライブラリファイル名のみが含まれているため、PLUGINS テーブルまたは SHOW PLUGINS ステートメントほどの多くの情報は提供されません。

5.1.9 IPv6 サポート

MySQL での IPv6 のサポートには次の機能があります。

- MySQL Server は、IPv6 を介して接続するクライアントからの TCP/IP 接続を受け入れることができます。たとえば、次のコマンドは、ローカルホスト上の MySQL Server に IPv6 を介して接続します。

```
shell> mysql -h ::1
```

この機能を使用するには、2 つのことが満たされている必要があります。

- システムが IPv6 をサポートするように構成されている必要があります。セクション5.1.9.1「IPv6 用のシステムサポートの確認」を参照してください。
- MySQL 5.6.6 以降では、デフォルトの MySQL Server 構成は、IPv4 接続に加えて IPv6 接続を許可します。5.6.6 より前の場合、デフォルトは IPv4 接続のみが許可されます。デフォルト構成を変更するには、適

切な --bind-address オプションを使用してサーバーを開始します。セクション5.1.4「サーバーシステム変数」を参照してください。

- MySQL アカウント名には、IPv6 を介してサーバーに接続するクライアントの権限を DBA が指定できるようにするために、IPv6 アドレスが許可されます。セクション6.2.3「アカウント名の指定」を参照してください。IPv6 アドレスは、CREATE USER、GRANT、REVOKE などのステートメント内のアカウント名に指定できます。例:

```
mysql> CREATE USER 'bill'@'::1' IDENTIFIED BY 'secret';
mysql> GRANT SELECT ON mydb.* TO 'bill'@'::1';
```

- IPv6 関数は、文字列と内部形式の IPv6 アドレス形式の間の変換が可能で、値が有効な IPv6 アドレスを表現しているかどうかを検査できます。たとえば、INET6_ATON() および INET6_NTOA() は INET_ATON() および INET_NTOA() に類似していますが、IPv4 アドレスに加えて IPv6 アドレスも処理します。セクション12.18「その他の関数」を参照してください。

以降のセクションでは、クライアントが IPv6 を介してサーバーに接続できるようにするために、MySQL をセットアップする方法について説明します。

5.1.9.1 IPv6 用のシステムサポートの確認

MySQL Server が IPv6 接続を受け入れるには、サーバーホスト上のオペレーティングシステムが IPv6 をサポートしている必要があります。このことが当てはまるかどうかを判別する簡単なテストとして、次のコマンドを試してみてください。

```
shell> ping6 ::1
16 bytes from ::1, icmp_seq=0 hlim=64 time=0.171 ms
16 bytes from ::1, icmp_seq=1 hlim=64 time=0.077 ms
...
```

システムのネットワークインタフェースの説明を生成するには、ifconfig -a を呼び出して、出力に IPv6 がないか探します。

ホストが IPv6 をサポートしない場合、システムのドキュメントを調べて IPv6 を有効にするための手順を確認します。既存のネットワークインタフェースの再構成後 IPv6 アドレスの追加のみが必要な場合もあります。また、IPv6 オプションを有効にしてカーネルを再構築するなどの大がかりな変更が必要な場合もあります。

IPv6 さまざまなプラットフォームで設定するとき、次のリンクが役立つことがあります。

- Windows XP
- Gentoo Linux
- Ubuntu Linux
- Linux (汎用)
- OS X

5.1.9.2 IPv6 接続を許可するための MySQL Server の構成

MySQL Server は、TCP/IP 接続について単一ネットワークソケットを listen します。このソケットは単一アドレスにバインドされますが、あるアドレスを複数のネットワークインタフェースにマップできます。アドレスを指定するには、サーバー起動時に --bind-address=addr オプションを使用します。ここで、addr は IPv4 または IPv6 アドレスあるいはホスト名です。(IPv6 アドレスは MySQL 5.5.3 より前ではサポートされません。)addr がホスト名の場合、サーバーはこの名前を IP アドレスに解決し、そのアドレスにバインドします。

サーバーはさまざまなタイプのアドレスを次のように処理します。

- アドレスが * の場合、サーバーホストが IPv6 アドレスをサポートする場合はすべてのサーバーホストの IPv6 および IPv4 インタフェース上の TCP/IP 接続を受け入れ、そうでない場合はすべての IPv4 アドレスの TCP/IP 接続を受け入れます。すべてのサーバーインタフェース上の IPv4 および IPv6 の両方の接続を許可するには、このアドレスを使用します。この値は、MySQL 5.6.6 以降で許可されています (またデフォルトです)。
- アドレスが 0.0.0.0 の場合、サーバーはすべてのサーバーホスト IPv4 インタフェース上の TCP/IP 接続を受け入れます。これは MySQL 5.6.6 以前のデフォルトです。
- アドレスが :: の場合、サーバーはすべてのサーバーホスト IPv4 および IPv6 インタフェース上の TCP/IP 接続を受け入れます。すべてのサーバーインタフェース上の IPv4 および IPv6 の両方の接続を許可するには、このアドレスを使用します。

- アドレスが IPv4 にマップ済みのアドレスの場合、サーバーは IPv4 または IPv6 のいずれかの形式で、そのアドレスの TCP/IP 接続を受け入れます。たとえば、サーバーが ::ffff:127.0.0.1 にバインドされている場合、クライアントは --host=127.0.0.1 または --host=::ffff:127.0.0.1 のいずれかを使用して接続できます。
- アドレスが「通常の」 IPv4 または IPv6 アドレスの場合 (127.0.0.1 や ::1 など)、サーバーはその IPv4 または IPv6 アドレスについてのみ TCP/IP 接続を受け入れます。

サーバーを特定のアドレスにバインドする予定の場合、そのアドレスに接続するために使用できる管理者権限を持つアカウントが mysql.user 付与テーブルに含まれていることを確認します。そうでない場合、サーバーをシャットダウンできません。たとえば、サーバーを * にバインドしている場合、すべての既存のアカウントを使用して接続できます。ただし、サーバーを ::1 にバインドしている場合、そのアドレスの接続のみ受け入れます。この場合、'root'@'::1' アカウントが mysql.user テーブルに存在することをまず確認して、サーバーに接続してシャットダウンできることをたしかめます。

5.1.9.3 IPv6 ローカルホストアドレスを使用した接続

次の手順では、::1 のローカルホストアドレスを使用してローカルサーバーに接続するクライアントによる IPv6 接続を許可するために、MySQL を構成する方法を示します。ここに示す手順は、システムが IPv6 をサポートしていることを想定しています。

1. IPv6 接続を受け入れることを許可するための適切な --bind-address オプションを使用して、MySQL Server を開始します。たとえば、次の行をサーバーオプションファイルに入れて、サーバーを再起動します。

   ```
   [mysqld]
   bind-address = *   # before 5.6.6, use :: rather than *
   ```

   または、サーバーを ::1 にバインドできますが、その場合 TCP/IP 接続のサーバー制限がさらに多くなります。その単一のアドレスについての IPv6 接続のみ受け入れ、IPv4 接続が拒否されます。詳細については、セクション5.1.9.2「IPv6 接続を許可するための MySQL Server の構成」を参照してください。

2. 管理者としてサーバーに接続し、::1 のローカル IPv6 ホストアドレスから接続するローカルユーザーのアカウントを作成します。

   ```
   mysql> CREATE USER 'ipv6user'@'::1' IDENTIFIED BY 'ipv6pass';
   ```

   アカウント名で許可される IPv6 アドレスの構文については、セクション6.2.3「アカウント名の指定」を参照してください。CREATE USER ステートメントに加えて、特定の権限をアカウントに付与する GRANT ステートメントを発行できます。ただし、この手順の残りのステップには不要です。

3. mysql クライアントを呼び出し、新しいアカウントを使用するサーバーに接続します。

   ```
   shell> mysql -h ::1 -u ipv6user -pipv6pass
   ```

4. 接続情報を表示する単純なステートメントを試してみます。

   ```
   mysql> STATUS
   ...
   Connection:   ::1 via TCP/IP
   ...

   mysql> SELECT CURRENT_USER(), @@bind_address;
   +---------------+---------------+
   | CURRENT_USER() | @@bind_address |
   +---------------+---------------+
   | ipv6user@::1   | ::            |
   +---------------+---------------+
   ```

5.1.9.4 IPv6 非ローカルホストアドレスを使用した接続

次の手順では、リモートクライアントによる IPv6 接続を許可するために MySQL を構成する方法を示します。これは前に示したローカルクライアントについての手順と似ていますが、サーバーとクライアントホストが別個であり、それぞれがローカル以外の独自の IPv6 アドレスを持ちます。この例では次のアドレスが使用されます。

```
Server host: 2001:db8:0:f101::1
Client host: 2001:db8:0:f101::2
```

これらのアドレスは、文書化目的で IANA によって推奨されるルーティングできないアドレス範囲から選択され、ローカルネットワークでの試験向けには十分です。ローカルネットワーク外部のクライアントから IPv6 接続を受け入れるには、サーバーホストが公開アドレスを持つ必要があります。ネットワークプロバイダによって IPv6 アドレスが割り当てられている場合、それを使用できます。そうでない場合、アドレスを取得するための別の方法は、IPv6 ブローカを使用する方法で、セクション5.1.9.5「ブローカからの IPv6 アドレスの入手」を参照してください。

1. IPv6 接続を受け入れることを許可するための適切な --bind-address オプションを使用して、MySQL Server を開始します。たとえば、次の行をサーバーオプションファイルに入れて、サーバーを再起動します。

```
[mysqld]
bind-address = *   # before 5.6.6, use :: rather than *
```

   または、サーバーを 2001:db8:0:f101::1 にバインドできますが、その場合 TCP/IP 接続のサーバーの制限がさらに多くなります。その単一のアドレスについての IPv6 接続のみ受け入れ、IPv4 接続が拒否されます。詳細については、セクション5.1.9.2「IPv6 接続を許可するための MySQL Server の構成」を参照してください。

2. サーバーホスト (2001:db8:0:f101::1) 上で、クライアントホスト (2001:db8:0:f101::2) から接続するユーザーにアカウントを作成します。

```
mysql> CREATE USER 'remoteipv6user'@'2001:db8:0:f101::2' IDENTIFIED BY 'remoteipv6pass';
```

3. クライアントホスト (2001:db8:0:f101::2) 上で、mysql クライアントを呼び出し、新しいアカウントを使用するサーバーに接続します。

```
shell> mysql -h 2001:db8:0:f101::1 -u remoteipv6user -premoteipv6pass
```

4. 接続情報を表示する単純なステートメントを試してみます。

```
mysql> STATUS
...
Connection:   2001:db8:0:f101::1 via TCP/IP
...

mysql> SELECT CURRENT_USER(), @@bind_address;
+-----------------------------------+---------------+
| CURRENT_USER()                    | @@bind_address |
+-----------------------------------+---------------+
| remoteipv6user@2001:db8:0:f101::2 | ::            |
+-----------------------------------+---------------+
```

5.1.9.5 ブローカからの IPv6 アドレスの入手

システムが IPv6 を介してローカルネットワークの外部と通信できるようにするための公開 IPv6 アドレスがない場合、IPv6 ブローカから入手できます。Wikipedia IPv6 Tunnel Broker page には、いくつかのブローカとその特徴 (静的アドレスおよびサポートされるルーティングプロトコルを提供するかどうかなど) をリストします。

ブローカが提供した IPv6 アドレスを使用するようにサーバーホストを構成したあと、適切な --bind-address オプションを使用して MySQL Server を開始すると、サーバーで IPv6 接続を受け入れることが許可されます。たとえば、次の行をサーバーオプションファイルに入れて、サーバーを再起動します。

```
[mysqld]
bind-address = *   # before 5.6.6, use :: rather than *
```

またはサーバーをブローカによって提供される特定の IPv6 にバインドできますが、その場合 TCP/IP 接続のサーバーの制限がさらに多くなります。その単一のアドレスについての IPv6 接続のみ受け入れ、IPv4 接続が拒否されます。詳細については、セクション5.1.9.2「IPv6 接続を許可するための MySQL Server の構成」を参照してください。さらに、ブローカによって動的アドレスが割り当てられた場合、ブローカに次回接続するときに、システムに指定されるアドレスが変更されることもあります。その場合、ユーザーが作成した、元のアドレスを指定するアカウントが無効になります。特定のアドレスにバインドするとき、このアドレス変更の問題を回避するには、静的 IPv6 アドレスを使用するようブローカと取り決めることができる場合もあります。

次の例は、Freenet6 をブローカとして使用し、gogoc IPv6 クライアントパッケージを Gentoo Linux 上で使用する方法を示します。

1. 次の URL にアクセスしてサインアップして、Freenet6 でアカウントを作成します。

```
http://gogonet.gogo6.com
```

2. アカウントを作成したあと、この URL に移動してサインインし、IPv6 ブローカ用のユーザー ID およびパスワードを作成します。

```
http://gogonet.gogo6.com/page/freenet6-registration
```

3. root として、gogoc をインストールします。

```
shell> emerge gogoc
```

4. /etc/gogoc/gogoc.conf を編集して userid および password の値を設定します。例:

```
userid=gogouser
```

```
passwd=gogopass
```

5. gogoc を開始します。

```
shell> /etc/init.d/gogoc start
```

システムが起動するたびに gogoc を開始するには、次のコマンドを実行します。

```
shell> rc-update add gogoc default
```

6. ping6 を使用してホストに ping を実行します。

```
shell> ping6 ipv6.google.com
```

7. IPv6 アドレスを表示するには、次のようにします。

```
shell> ifconfig tun
```

5.1.10 サーバー側のヘルプ

MySQL Server は、MySQL リファレンスマニュアルからオンライン情報を表示する HELP ステートメントをサポートしています (セクション13.8.3「HELP 構文」を参照してください)。このステートメントを適切に操作するには、mysql データベースのヘルプテーブルが、ヘルプトピック情報で初期化されることが必要で、これは fill_help_tables.sql スクリプトの内容を処理することによって行います。

Unix でバイナリまたはソース配布を使用して MySQL をインストールした場合、mysql_install_db を実行したときにヘルプテーブルの内容が初期化されます。Linux の RPM 配布または Windows のバイナリ配布の場合、コンテンツの初期化は MySQL インストールプロセスの一部として実行されます。

バイナリ配布を使用して MySQL をアップグレードする場合、ヘルプテーブルの内容は自動的にアップグレードされませんが、手動でアップグレードできます。share または share/mysql ディレクトリ内から fill_help_tables.sql ファイルを見つけます。場所をそのディレクトリに変更し、mysql クライアントを使用してファイルを次のように処理します。

```
shell> mysql -u root mysql < fill_help_tables.sql
```

ヘルプテーブルをアップグレードするために、最新の fill_help_tables.sql をいつでも取得できます。MySQL のバージョンの適切なファイルを https://dev.mysql.com/doc/index-other.html からダウンロードします。ファイルをダウンロードして圧縮解除したあと、上記のように mysql を使用してファイルを処理します。

Bazaar および MySQL 開発ソースツリーを操作している場合、ソースツリーには「スタブ」バージョンしか含まれていないため、ダウンロードされた fill_help_tables.sql ファイルのコピーを使用する必要があります。

> **注記**
>
> レプリケーションに参加するサーバーの場合、ヘルプテーブルの内容のアップグレードプロセスには複数のサーバーが関与します。詳細については、セクション17.4.1.27「サーバー側ヘルプテーブルのレプリケーション」を参照してください。

5.1.11 シグナルへのサーバー応答

Unix では、シグナルをプロセスに送信できます。mysqld は、プロセスに送信されたシグナルに次のように応答します。

- SIGTERM によってサーバーはシャットダウンします。
- SIGHUP によって、サーバーは付与テーブルをリロードし、テーブル、ログ、スレッドキャッシュ、およびホストキャッシュをフラッシュします。これらのアクションは、FLUSH ステートメントのさまざまな形式に似ています。サーバーは、次の形式を持つステータスレポートをエラーログに書き込みます。

```
Status information:

Current dir: /var/mysql/data/
Running threads: 0  Stack size: 196608
Current locks:

Key caches:
default
Buffer_size:       8388600
Block_size:           1024
Division_limit:        100
Age_limit:             300
```

```
blocks used:            0
not flushed:            0
w_requests:             0
writes:                 0
r_requests:             0
reads:                  0

handler status:
read_key:           0
read_next:          0
read_rnd            0
read_first:         1
write:              0
delete              0
update:             0

Table status:
Opened tables:          5
Open tables:            0
Open files:             7
Open streams:           0

Alarm status:
Active alarms:  1
Max used alarms: 2
Next alarm time: 67
```

一部の OS X 10.3 バージョンでは、mysqld は SIGHUP および SIGQUIT を無視します。

5.1.12 シャットダウンプロセス

サーバーのシャットダウンプロセスは、次のように実行されます。

1. シャットダウンプロセスが開始されます。

   これはいくつかの方法で開始できます。たとえば、SHUTDOWN 権限を持つユーザーは mysqladmin shutdown コマンドを実行できます。mysqladmin は、MySQL によってサポートされているすべてのプラットフォーム上で使用できます。オペレーティングシステムに固有のほかのシャットダウン開始方式も使用可能で、Unix ではサーバーが SIGTERM シグナルを受け取るとサーバーがシャットダウンします。Windows 上でサービスとして実行中のサーバーは、サービスマネージャーからシャットダウンを指示されるとシャットダウンします。

2. サーバーは必要に応じてシャットダウンスレッドを作成します。

   シャットダウンが開始された方法によっては、シャットダウンプロセスを処理するためのスレッドをサーバーが作成することがあります。シャットダウンがクライアントによってリクエストされた場合、シャットダウンスレッドが作成されます。シャットダウンが SIGTERM を受信したことによるものである場合、シグナルスレッドそれ自体がシャットダウンを処理したり、処理を行うための別のスレッドを作成したりすることがあります。サーバーがシャットダウンスレッドを作成しようとしたが作成できない場合 (メモリーが不足する場合など)、診断メッセージが発行されてエラーログに示されます。

   ```
   Error: Can't create thread to kill server
   ```

3. サーバーは新しい接続の受け入れを停止します。

   シャットダウン中に新しいアクティビティーが開始されるのを防ぐために、サーバーは通常、接続を listen するネットワークインタフェースのハンドラを閉じて新しいクライアント接続の受け入れを停止します。接続には TCP/IP ポート、Unix ソケットファイル、Windows 名前付きパイプ、および Windows の共有メモリーがあります。

4. サーバーは現在のアクティビティーを終了します。

   クライアント接続に関連付けられている各スレッドについて、サーバーはクライアントへの接続を切断し、スレッドに強制終了のマークを付けます。スレッドは、そのようなマークを付けられたことが通知されると終了します。アイドル状態の接続のスレッドは、ただちに終了します。ステートメントを現在処理中のスレッドは、その状態を定期的に検査するため、終了するのに時間がかかります。スレッド終了についての追加情報について、特に、MyISAM テーブルで強制終了された REPAIR TABLE または OPTIMIZE TABLE 操作に関する指示については、セクション13.7.6.4「KILL 構文」を参照してください。

   オープン中のトランザクションがあるスレッドでは、トランザクションがロールバックされます。スレッドが非トランザクションテーブルを更新している場合、複数行の UPDATE または INSERT などの操作は完了前に終了できるため、そのテーブルが一部更新されたままになることがあります。

サーバーがマスターレプリケーションサーバーの場合、サーバーは現在接続されているスレーブに関連付けられているスレッドを、ほかのクライアントのスレッドにように扱います。つまり、それぞれに強制終了のマークを付け、その状態を次回検査するときに終了します。

サーバーがスレーブレプリケーションサーバーの場合、I/O および SQL スレッドがアクティブであれば、クライアントスレッドに強制終了のマークを付ける前にこれらを停止します。SQL スレッドは、(レプリケーションの問題を起こすことを防ぐために) 現在のステートメントを完了することが可能で、そのあとで終了します。SQL スレッドがこの時点でトランザクションの途中である場合、サーバーは現在のレプリケーションイベントグループ (ある場合) が実行を完了するか、ユーザーが KILL QUERY または KILL CONNECTION ステートメントを発行するまで待機します。セクション13.4.2.6「STOP SLAVE 構文」も参照してください。非トランザクションステートメントはロールバックできないため、クラッシュに対する安全性を持つレプリケーションを保証するにはトランザクションテーブルのみを使用してください。

> **注記**
>
> スレーブでのクラッシュに対する安全性を保証するには、--relay-log-recovery を有効にしてスレーブを実行することも必要です。

セクション17.2.2「レプリケーションリレーおよびステータスログ」も参照してください。

5. サーバーはシャットダウンするかストレージエンジンを閉じます。

   この段階で、サーバーはテーブルキャッシュをフラッシュして、オープン中のすべてのテーブルを閉じます。

   各ストレージエンジンは、管理するテーブルに必要なすべての動作を実行します。 InnoDB はバッファープールをディスクにフラッシュし (innodb_fast_shutdown が 2 である場合を除く)、現在の LSN をテーブルスペースに書き込み、その独自の内部スレッドを終了します。MyISAM は、テーブルの保留中のインデックス書き込みをフラッシュします。

6. サーバーが終了します。

5.2 MySQL Server ログ

MySQL Server には、実行中のアクティビティーを検出するのに役立ついくつかのログがあります。

ログのタイプ	ログに書き込まれる情報
エラーログ	mysqld の起動、実行、および停止で発生した問題
一般クエリーログ	確立されたクライアント接続およびクライアントから受け取ったステートメント
バイナリログ	データを変更するステートメント (レプリケーションにも使用される)
リレーログ	レプリケーションマスターサーバーから受け取ったデータ変更
スロークエリーログ	実行するのに long_query_time 秒よりも時間を要したクエリー
DDL ログ (メタデータログ)	DDL ステートメントによって実行されたメタデータ操作

デフォルトでは、Windows 上のエラーログを除いてログは有効化されていません。(DDL ログは必要な場合に常に作成され、ユーザーが構成可能なオプションはありません。セクション5.2.6「DDL ログ」を参照してください。)このあとに続くログに固有のセクションでは、ロギングを有効にするためのサーバーオプションに関する情報を提供します。

デフォルトでは、サーバーは有効化されたすべてのログに対してデータディレクトリ内にファイルを書き込みます。ログをフラッシュすることによって、サーバーがログファイルを閉じて再オープンする (または新しいログファイルに切り替える) ことを強制的に実行できます。ログのフラッシュは、FLUSH LOGS ステートメントを発行したり、mysqladmin に flush-logs または refresh 引数を指定して実行したり、mysqldump に --flush-logs または --master-data オプションを指定して実行したりしたときに実行されます。セクション13.7.6.3「FLUSH 構文」、セクション4.5.2「mysqladmin — MySQL サーバーの管理を行うクライアント」、およびセクション4.5.4「mysqldump — データベースバックアッププログラム」を参照してください。さらに、バイナリログは、サイズが max_binlog_size システム変数の値に達するとフラッシュされます。

一般クエリーログおよびスロークエリーログを実行時に制御することができます。ロギングを有効化または無効化したり、ログファイル名を変更したりできます。一般クエリーエントリおよびスロークエリーエントリを、ログテーブル、ログファイル、または両方に書き込むようにサーバーに指示することができます。詳細については、セクション5.2.1「一般クエリーログおよびスロークエリーログの出力先の選択」、セクション5.2.3「一般クエリーログ」、およびセクション5.2.5「スロークエリーログ」を参照してください。

リレーログはスレーブレプリケーションサーバー上でのみ使用され、スレーブ上でも実行する必要があるマスターサーバーからのデータ変更を保持します。リレーログの内容および構成については、セクション17.2.2.1「スレーブリレーログ」を参照してください。

古いログファイルの有効期限などのログの保守操作についての情報は、セクション5.2.7「サーバーログの保守」を参照してください。

ログのセキュリティー保護についての情報は、セクション6.1.2.3「パスワードおよびロギング」を参照してください。

5.2.1 一般クエリーログおよびスロークエリーログの出力先の選択

MySQL Server では、一般クエリーログおよびスロークエリーログが有効化されている場合、これらのログへの出力先を柔軟に制御できます。ログエントリの可能な出力先は、ログファイル、または mysql データベース内の general_log および slow_log テーブルです。いずれかまたは両方の出力先を選択できます。

サーバー起動時のログ制御。--log-output オプションは、ログ出力の出力先を指定します。このオプション自体はログを有効化しません。この構文は、--log-output[=value,...] です。

- --log-output に値が指定される場合、値は TABLE (テーブルへのログ)、FILE (ファイルへのログ)、または NONE (テーブルにもファイルにもログを出力しない) のうち 1 つ以上の単語のカンマ区切りリストにします。NONE がある場合は、ほかの指定子よりも優先されます。
- --log-output が省略された場合、デフォルトのロギング出力先は、FILE です。

general_log システム変数は、選択されたログ出力先についての一般クエリーログへのロギングを制御します。サーバー起動時に指定された場合、general_log は、ログを有効化または無効化するためのオプション引数 1 または 0 を取ります。ファイルロギングについて、デフォルト以外のファイル名を指定するには、general_log_file 変数を設定します。同様に、slow_query_log 変数は、選択された出力先についてのスロークエリーログへのロギングを制御し、slow_query_log_file の設定は、ファイルロギングのためのファイル名を指定します。いずれかのログが有効化された場合、サーバーは対応するログファイルを開き、ログファイルに起動メッセージを書き込みます。ただし、FILE ログの出力先が選択されないかぎり、ファイルに対するそれ以上のクエリーのロギングは実行されません。

例:

- 一般クエリーログエントリをログテーブルおよびログファイルに書き込むには、--log-output=TABLE,FILE を使用して両方のログ出力先を選択し、--general_log を使用して、一般クエリーログを有効化します。
- 一般クエリーログエントリおよびスロークエリーログエントリをログテーブルにのみ書き込むには、--log-output=TABLE を使用してテーブルをログ出力先として選択し、--general_log および --slow_query_log を使用して両方のログを有効化します。
- スロークエリーログエントリをログファイルにのみ書き込むには、--log-output=FILE を使用してファイルをログ出力先として選択し、--slow_query_log を使用してスロークエリーログを有効化します。(この場合、デフォルトのログ出力先は FILE であるため、--log-output オプションを省略できます。)

実行時のログ制御。ログテーブルおよびファイルに関連付けられたシステム変数によって、ロギングへの実行時制御が可能になります。

- グローバルな log_output システム変数は、現在のロギング出力先を示します。出力先を変更するために、これを実行時に変更できます。
- グローバルな general_log および slow_query_log 変数は、一般クエリーログおよびスロークエリーログを有効化 (ON) または無効化 (OFF) するかどうかを指示します。これらの変数を実行時に設定して、ログを有効化するかどうかを制御することができます。
- グローバルな general_log_file および slow_query_log_file 変数は、一般クエリーログファイルおよびスロークエリーログファイルの名前を指示します。これらの変数をサーバー起動時または実行時に設定して、ログファイルの名前を変更することができます。
- 現在の接続のための一般クエリーロギングを無効化または有効化するには、セッションの sql_log_off 変数を ON または OFF に設定します。

ログ出力用のテーブルを使用することには、次の利点があります。

- ログエントリが標準形式を持ちます。ログテーブルの現在の構造を表示するには、次のステートメントを使用します。

```
SHOW CREATE TABLE mysql.general_log;
```

```
SHOW CREATE TABLE mysql.slow_log;
```

- ログ内容に SQL ステートメントを使用してアクセスできます。これにより、特定の基準を満たすログエントリのみ選択するクエリーを使用することができます。たとえば、特定のクライアントに関連したログ内容を選択するには (そのクライアントからの問題のあるクエリーを特定するために役立つことがあります)、ログファイルよりもログテーブルを使用して行う方が簡単です。
- サーバーに接続してクエリーを発行できるすべてのクライアントを介して、ログにリモートからアクセスできます (クライアントが適切なログテーブル権限を持つ場合)。サーバーホストにログインしてファイルシステムに直接アクセスする必要はありません。

ログテーブルの実装には次の特徴があります。

- 一般的に、ログテーブルの主な目的は、サーバーのランタイム実行を観察するユーザーにインタフェースを提供することで、サーバーのランタイム実行を妨げません。
- CREATE TABLE、ALTER TABLE、および DROP TABLE はログテーブル上での有効な操作です。ALTER TABLE および DROP TABLE の場合、ログテーブルは使用中であってはならず、あとで説明するように無効にする必要があります。
- デフォルトでは、ログテーブルは、カンマ区切り値形式でデータを書き込む CSV ストレージエンジンを使用します。ログテーブルデータを含む .CSV ファイルにアクセスするユーザーの場合、CSV 入力を処理できるスプレッドシートなどのほかのプログラムにファイルを簡単にインポートできます。

  ログテーブルは、MyISAM ストレージエンジンを使用するように変更することができます。使用中のログテーブルを変更するために、ALTER TABLE を使用することはできません。ログを最初に無効にする必要があります。CSV または MyISAM 以外のすべてのエンジンは、ログテーブルについて適正ではありません。
- ログテーブルを変更 (または削除) できるようにロギングを無効化するには、次の方法を使用することができます。この例では一般クエリーログを使用しており、スロークエリーログについての手順も類似していますが、slow_log テーブルおよび slow_query_log システム変数を使用します。

```
SET @old_log_state = @@GLOBAL.general_log;
SET GLOBAL general_log = 'OFF';
ALTER TABLE mysql.general_log ENGINE = MyISAM;
SET GLOBAL general_log = @old_log_state;
```

- TRUNCATE TABLE は、ログテーブル上での有効な操作です。ログエントリを期限切れにするために使用できます。
- RENAME TABLE は、ログテーブル上での有効な操作です。次の方法を使用して、(たとえばログローテーションを実行するために) ログテーブルを原子的に名前変更できます。

```
USE mysql;
DROP TABLE IF EXISTS general_log2;
CREATE TABLE general_log2 LIKE general_log;
RENAME TABLE general_log TO general_log_backup, general_log2 TO general_log;
```

- CHECK TABLE は、ログテーブル上での有効な操作です。
- LOCK TABLES をログテーブル上で使用することはできません。
- INSERT、DELETE、および UPDATE をログテーブル上で使用することはできません。これらの操作は、サーバー自体の内部でのみ許可されます。
- FLUSH TABLES WITH READ LOCK およびグローバルな read_only システム変数の状態は、ログテーブルに影響を及ぼしません。サーバーは常にログテーブルに書き込むことができます。
- ログテーブルに書き込まれたエントリはバイナリログに書き込まれないため、スレーブサーバーにレプリケーションされません。(MySQL 5.6.9 より前では、これは常に正しく強制されるわけではありませんでした。Bug #14741537 を参照してください。)
- ログテーブルまたはログファイルをフラッシュするには、FLUSH TABLES または FLUSH LOGS をそれぞれ使用します。
- ログテーブルのパーティション化は許可されません。
- MySQL 5.6.6 より前では、mysqldump は mysql データベースのダンプとして general_log テーブルも slow_query_log テーブルもダンプしません。5.6.6 以降では、ダンプにはそれらのテーブルを再作成するためのステートメントが含まれているため、ダンプファイルを再ロードしたあとにそれらのテーブルが失われません。ログテーブルの内容はダンプされません。

5.2.2 エラーログ

エラーログには mysqld が開始および停止された時期を示す情報と、サーバーが実行中に発生したあらゆるクリティカルエラーが格納されます。自動的にチェックまたは修復することが必要なテーブルが mysqld で検出された場合、エラーログに警告メッセージが書き込まれます。

オペレーティングシステムによっては、mysqld が異常終了した場合にエラーログにスタックトレースが記録されます。トレースを使用して、mysqld が異常終了したかどうかを判断することができます。セクション 24.4「MySQL のデバッグおよび移植」 を参照してください。

mysqld_safe を使用して mysqld が開始され、mysqld が予期せず異常終了した場合、mysqld_safe はこれを検出し、mysqld を再起動し、restarted mysqld メッセージをエラーログに書き込みます。

次の説明で、「コンソール」は標準エラー出力 stderr を意味し、標準エラー出力がリダイレクトされていないかぎり、これはユーザーのターミナルウィンドウまたはコンソールウィンドウです。(たとえば、--syslog オプションを指定して起動した場合、あとで説明するように、mysqld_safe ではサーバーの stderr が syslog に送信されるような調整が行われます。)

Windows で、--log-error および --console オプションは、両方ともエラーロギングに影響します。

- --log-error を指定しない場合、mysqld はデータディレクトリ内の host_name.err にエラーメッセージを書き込みます。
- --log-error[=file_name] を指定した場合、mysqld はエラーログファイルにエラーメッセージを書き込みます。指定されたファイルが存在する場合、サーバーはそのファイルを使用し、別のディレクトリを指定する絶対パス名が指定されている場合を除き、そのファイルをデータディレクトリ内に作成します。ファイルが指定されない場合、デフォルト名は、データディレクトリ内の host_name.err です。
- --console を指定した場合、mysqld は、--log-error がさらに指定される場合を除き、エラーメッセージをコンソールに書き込みます。両方のオプションが存在する場合、--console は無視されて効果がなくなります。それらの順序は関係なく、--log-error が優先されて、エラーメッセージがログファイルに記録されます。

さらに Windows では、サーバーはイベントおよびエラーメッセージをアプリケーションログ内の Windows イベントログに書き込みます。Warning および Note のマークが付いたエントリは、イベントログに書き込まれますが、個々のストレージエンジンからの情報ステートメントなどの情報メッセージは書き込まれません。これらのログエントリのソースは MySQL です。Windows イベントログへの情報の書き込みを無効にすることはできません。

Unix および Unix に類似したシステムでは、mysqld はエラーログメッセージを次のように書き込みます。

- --log-error を指定しない場合、mysqld はエラーメッセージをコンソールに書き込みます。
- --log-error[=file_name] を指定した場合、mysqld はエラーログファイルにエラーメッセージを書き込みます。指定されたファイルが存在する場合、サーバーはそのファイルを使用し、別のディレクトリを指定する絶対パス名が指定されている場合を除き、そのファイルをデータディレクトリ内に作成します。ファイルが指定されない場合、デフォルト名は、データディレクトリ内の host_name.err です。

> **注記**
>
> Yum または APT パッケージインストールでは、サーバー構成ファイル内の log-error=/var/log/mysqld.log のようなエントリによって、エラーログの場所を /var/log の下に構成することが一般的です。エントリからファイル名を削除すると、エラーログの場所がそのデフォルト設定であるデータディレクトリに戻されます。

エラー出力がファイルに書き込まれる場合、実行時に、log_error システム変数がエラーログファイル名を示します。

オプションファイル内の [mysqld]、[server]、または [mysqld_safe] セクションに --log-error を指定した場合、mysqld_safe はオプションを検出して使用します。

FLUSH LOGS または mysqladmin flush-logs を使用してログをフラッシュし、mysqld がエラーログをファイルに書き込む場合 (たとえば、--log-error オプションを使用して開始された場合)、サーバーはログファイルを閉じて再オープンします。ファイルを名前変更するには、フラッシュする前にこれを手動で実行します。そのあとでログをフラッシュすると、新しいファイルが元のファイル名で再オープンします。たとえば、次のコマンドを使用して、ファイルを名前変更し、新しいファイルを作成することができます。

```
shell> mv host_name.err host_name.err-old
shell> mysqladmin flush-logs
shell> mv host_name.err-old backup-directory
```

Windows では、mv の代わりに rename を使用してください。

サーバーが指定されたファイルに書き込まない場合、ログがフラッシュされたときにエラーログの名前変更は実行されません。

mysqld_safe を使用して mysqld を開始した場合、mysqld_safe では mysqld がエラーメッセージをログファイルまたは syslog に書き込むような調整が行われます。mysqld_safe には --syslog、--skip-syslog、および --log-error の 3 つのエラーロギングオプションがあります。ロギングオプションを指定しないか --skip-syslog を指定した場合のデフォルトは、デフォルトログファイルを使用することです。エラーログファイルの使用を明示的に指定する場合、--log-error=file_name を mysqld_safe に指定すると、mysqld_safe では mysqld がメッセージをログファイルに書き込むような調整が行われます。syslog を代わりに使用するには、--syslog オプションを指定します。

--log-warnings オプションまたは log_warnings システム変数を使用すると、エラーログへの警告ロギングを制御できます。デフォルト値は有効化 (1) です。値 0 を使用して警告ロギングを無効にすることができます。値が 1 より大きい場合、中止された接続がエラーログに書き込まれ、新しい接続の試行についてのアクセス拒否エラーが書き込まれます。セクションB.5.2.11「通信エラーおよび中止された接続」を参照してください。

5.2.3 一般クエリーログ

一般クエリーログは、mysqld の実行内容の一般的な記録です。サーバーは、クライアントが接続または接続解除したときに情報をこのログに書き込み、クライアントから受け取った各 SQL ステートメントをログに記録します。一般クエリーログは、クライアント側でエラーが疑われるとき、クライアントが mysqld に送信した内容を正確に知りたい場合に非常に役立つことがあります。

mysqld は、ステートメントを受け取った順にクエリーログに書き込みますが、ステートメントが実行された順番とは異なることがあります。このロギング順序はバイナリログの順序とは対照的で、バイナリログの場合、ステートメントはそれが実行されたあと、ロックがリリースされる前に書き込まれます。さらに、クエリーログはデータを選択するだけのステートメントを格納することもあり、そのようなステートメントはバイナリログには一切書き込まれません。

ステートメントベースのロギングを使用する場合、すべてのステートメントはクエリーログに書き込まれますが、行ベースのロギングを使用する場合、更新は SQL ステートメントではなく行の変更として送信されるため、binlog_format が ROW のときは、これらのステートメントはクエリーログに一切書き込まれません。使用されるステートメントによっては、この変数が MIXED に設定された場合、所定の更新がクエリーログに書き込まれないこともあります。詳細については、セクション17.1.2.1「ステートメントベースおよび行ベースレプリケーションのメリットとデメリット」を参照してください。

デフォルトでは、一般クエリーログは無効になっています。初期の一般クエリーログ状態を明示的に指定するには、--general_log[={0|1}] を使用します。引数を指定しないか、引数が 1 の場合、--general_log によってログが有効になります。引数が 0 の場合、このオプションによってログが無効になります。ログファイル名を指定するには、--general_log_file=file_name を使用します。ログの出力先を指定するには、(セクション5.2.1「一般クエリーログおよびスロークエリーログの出力先の選択」で説明されている) --log-output を使用します。

一般クエリーログファイルの名前を指定しない場合、デフォルト名は host_name.log です。サーバーは、別のディレクトリを指定する絶対パス名が指定されないかぎり、データディレクトリ内にファイルを作成します。

実行時に一般クエリーログを無効化または有効化したり、ログファイル名を変更したりするには、グローバルな general_log および general_log_file システム変数を使用します。general_log を 0 (または OFF) に設定するとログが無効化され、1 (または ON) にすると有効化されます。ログファイルの名前を指定するには、general_log_file を指定します。ログファイルがすでに開いている場合、ログファイルが閉じて新しいファイルが開きます。

一般クエリーログが有効な場合、サーバーは --log-output オプションまたは log_output システム変数によって指定されたあらゆる出力先に出力を書き込みます。ログを有効にすると、サーバーはログファイルを開き、ログファイルに起動メッセージを書き込みます。ただし、FILE ログの出力先が選択されないかぎり、ファイルに対するそれ以上のクエリーのロギングは実行されません。出力先が NONE の場合、一般ログが有効な場合であってもサーバーはクエリーを書き込みません。ログ出力先の値に FILE が含まれていない場合、ログファイル名を設定してもロギングへの影響はありません。

サーバー再起動およびログフラッシュを行なっても、新しい一般クエリーログファイルは生成されません (ただし、フラッシュではファイルが閉じて再オープンします)。ファイルを名前変更して新しいファイルを作成するには、次のコマンドを使用します。

```
shell> mv host_name.log host_name-old.log
shell> mysqladmin flush-logs
shell> mv host_name-old.log backup-directory
```

Windows では、mv の代わりに rename を使用してください。

また、ログを無効にすることによって、実行時に一般クエリーログファイルを名前変更することができます。

```
SET GLOBAL general_log = 'OFF';
```

ログが無効化されている場合、コマンド行などの外部からログファイルを名前変更してください。そのあと、ログをふたたび有効にします。

```
SET GLOBAL general_log = 'ON';
```

この方法はすべてのプラットフォームで動作し、サーバー再起動を必要としません。

セッション変数 sql_log_off を ON または OFF に設定して、現在の接続についての一般クエリーロギングを無効化または有効化することができます。

MySQL 5.6.3 以降では、一般クエリーログに書き込まれるステートメントのパスワードはサーバーによって書き換えられ、文字どおりに平文で表示されることはありません。一般クエリーログについてのパスワードの書き換えは、--log-raw オプションでサーバーを起動することによって抑制できます。このオプションは、サーバーによって受け取られるステートメントの正確なテキストを表示する際の診断目的で役立つ場合がありますが、セキュリティー上の理由で本番用途では推奨されません。

MySQL 5.6.3 より前では、ステートメント内のパスワードは書き換えられないため、一般クエリーログを保護するようにしてください。セクション6.1.2.3「パスワードおよびロギング」を参照してください。

MySQL 5.6.3 でのパスワード書き換えの導入による 1 つの影響として、(構文エラーなどが理由で) 構文解析できないステートメントがパスワードなしであることを認識できないため、一般クエリーログに書き込まれなくなるということがあります。エラーを持つものを含むすべてのステートメントのロギングを必要とする使用例では、--log-raw オプションを使用しますが、このオプションはパスワードの書き込みをバイパスすることにも留意してください。

5.2.4 バイナリログ

バイナリログには、テーブル作成操作やテーブルデータへの変更などのデータベース変更を記述する「イベント」が格納されます。また、行ベースのロギングが使用される場合を除き、(一致する行のない DELETE などの) 潜在的に変更を行おうとしたステートメントについてのイベントも格納されます。バイナリログには、データを更新した各ステートメントに要した時間に関する情報も格納されます。バイナリログには 2 つの重要な目的があります。

- レプリケーションについて、マスターレプリケーションサーバー上のバイナリログは、スレーブサーバーに送信されるデータ変更のレコードを提供します。マスターサーバーは、そのバイナリログに格納されているイベントをそのスレーブに送信し、スレーブはこれらのイベントを実行して、マスター上で実行されたものと同じデータ変更を実行します。セクション17.2「レプリケーションの実装」を参照してください。
- ある特定のデータリカバリ操作には、バイナリログの使用が必要です。バックアップがリストアされたあと、バックアップが実行されたあとに記録されたバイナリログ内のイベントが再実行されます。これらのイベントは、データベースをバックアップのポイントから最新の状態に持って行きます。セクション7.5「バイナリログを使用したポイントインタイム (増分) リカバリ」を参照してください。

バイナリログは、データを変更しない SELECT や SHOW などのステートメントでは使用されません。(問題となるクエリーを特定するなどのために) すべてのステートメントを口グに記録するには、一般クエリーログを使用します。セクション5.2.3「一般クエリーログ」を参照してください。

バイナリロギングを有効にしてサーバーを実行すると、パフォーマンスがいくらか低下します。ただし、レプリケーションをセットアップでき、リストア操作に対応できるというバイナリログの利点は、一般的にこのパフォーマンスの減少よりも重要です。

MySQL 5.6.2 以降では、バイナリログはクラッシュセーフになっています。完全なイベントまたはトランザクションのみがログに記録されたり、または読み戻されたりします。

MySQL 5.6.3 以降では、バイナリログに書き込まれるステートメントのパスワードはサーバーによって書き換えられ、文字どおりに平文で表示されることはありません。MySQL 5.6.3 より前では、ステートメント内のパスワードは書き換えられないため、バイナリログを保護するようにしてください。セクション6.1.2.3「パスワードおよびロギング」を参照してください。

次の説明では、バイナリロギングの操作に影響する一部のサーバーオプションおよび変数について記述します。完全なリストについては、セクション17.1.4.4「バイナリログのオプションと変数」を参照してください。

バイナリログを有効にするには、--log-bin[=base_name] オプションでサーバーを起動します。base_name 値が指定されない場合、デフォルト名は、pid-file オプションの値 (デフォルトではこれはホストマシンの名前) のあとに -bin が続きます。ベース名が指定される場合、別のディレクトリを指定する絶対パス名が前に付いたベース名が指定されないかぎり、サーバーはファイルをデータディレクトリに書き込みます。デフォルトのホスト名を使用

せず、ベース名を明示的に指定することをお勧めします。その理由については、セクションB.5.8「MySQL の既知の問題」を参照してください。

ログ名に拡張子を指定した場合 (--log-bin=base_name.extension など)、拡張子は暗黙的に削除されて無視されます。

mysqld はバイナリログファイル名を生成するために、バイナリログベース名に数値拡張を付加します。数値はサーバーが新しいログファイルを作成するたびに増加し、順序付きの一連のファイルが作成されます。その一連のファイル内の新しいファイルは、サーバーが起動するかログをフラッシュするたびにサーバーによって作成されます。また、サーバーは現在のログのサイズが max_binlog_size に到達したあと、新しいバイナリログファイルを自動的に作成します。大きなトランザクションを使用する場合、トランザクションがひとまとまりでファイルに書き込まれて、複数のファイルに分割されないため、バイナリログファイルが max_binlog_size を超えることがあります。

使用されたバイナリログファイルの追跡を行うために、mysqld は、使用されたすべてのバイナリログファイルの名前を格納するバイナリログインデックスファイルも作成します。デフォルトでは、これはバイナリログファイルと同じベース名を持ち、拡張子 '.index' が付いています。バイナリログインデックスファイルの名前は、--log-bin-index[=file_name] オプションを使用して変更できます。mysqld の動作中にこのファイルを手動で変更しないでください。変更すると、mysqld を混乱させることになります。

「バイナリログファイル」という用語は一般的に、データベースイベントを格納する、番号付けされた個々のファイルを指します。「バイナリログ」という用語は、番号付けされたバイナリログファイルとインデックスファイルのセットをひとまとめにしたものを指します。

SUPER 権限を持つクライアントは、SET sql_log_bin=0 ステートメントを使用することによって、自分自身のステートメントのバイナリロギングを無効にすることができます。セクション5.1.4「サーバーシステム変数」を参照してください。

デフォルトでは、サーバーはイベント自体だけでなくイベントの長さもログに記録し、イベントが正しく書き込まれたことを検証するためにこれを使用します。また、binlog_checksum システム変数を設定することによって、サーバーがイベントのチェックサムを書き込むようにすることもできます。バイナリログから読み戻すとき、マスターはデフォルトではイベントの長さを使用しますが、master_verify_checksum システム変数を有効にすることによって、使用可能であればチェックサムを使用するように変更できます。スレーブ I/O スレッドも、マスターから受け取ったイベントを検証します。slave_sql_verify_checksum システム変数を有効にすることによって、リレーログから読み取るときにチェックサムが使用可能な場合、スレーブ SQL スレッドがチェックサムを使用するようにすることもできます。

バイナリログに記録されるイベントの形式は、バイナリロギング形式に依存します。行ベースのロギング、ステートメントベースのロギング、および混合ベースのロギングの 3 つのタイプの形式がサポートされます。使用されるバイナリロギング形式は、MySQL のバージョンに依存します。ロギング形式の一般的な説明については、セクション5.2.4.1「バイナリロギング形式」を参照してください。バイナリログの形式についての詳細な説明は、「MySQL Internals: The Binary Log」を参照してください。

サーバーは、--binlog-do-db および --binlog-ignore-db オプションを評価する際、それが --replicate-do-db および --replicate-ignore-db オプションを評価する場合と同じ方法で行います。これを行う方法については、セクション17.2.3.1「データベースレベルレプリケーションオプションおよびバイナリロギングオプションの評価」を参照してください。

レプリケーションスレーブサーバーは、デフォルトでは、レプリケーションマスターから受け取ったすべてのデータ変更を、それ自身のバイナリログに書き込みません。これらの変更をログに記録するには、--log-bin オプションに加えて --log-slave-updates オプションを指定してスレーブを起動します (セクション17.1.4.3「レプリケーションスレーブのオプションと変数」を参照してください)。これは、スレーブがチェーン型レプリケーションで、ほかのスレーブのマスターとしての役割も果たす場合に実行されます。

RESET MASTER ステートメントですべてのバイナリログファイルを削除したり、PURGE BINARY LOGS でそのサブセットを削除したりすることができます。セクション13.7.6.6「RESET 構文」およびセクション13.4.1.1「PURGE BINARY LOGS 構文」を参照してください。

レプリケーションを使用している場合、マスター上の古いバイナリログファイルを使用する必要があるスレーブがなくなったことを確認するまでは、それらのファイルを削除しないようにしてください。たとえば、スレーブが 3 日を超えて遅れて実行されることがない場合、mysqladmin flush-logs をマスター上で 1 日に 1 回実行し、3 日分よりも古いログを削除することができます。ファイルを手動で削除することができますが、PURGE BINARY LOGS を使用することが推奨され、この操作によってバイナリログインデックスファイルも安全に更新されます (さらに日付引数を使用できます)。セクション13.4.1.1「PURGE BINARY LOGS 構文」を参照してください。

mysqlbinlog ユーティリティーを使用して、バイナリログファイルの内容を表示できます。これはリカバリ操作のために log 内のステートメントを再処理するときに役立ちます。たとえば、次のようにしてバイナリログから MySQL Server を更新できます。

```
shell> mysqlbinlog log_file | mysql -h server_name
```

mysqlbinlog は、レプリケーションスレーブのリレーログファイルの内容を表示するためにも使用できます。これは、これらがバイナリログファイルと同じ形式を使用して書き込まれるためです。mysqlbinlog ユーティリティーとその使用方法についての詳細は、セクション4.6.8「mysqlbinlog — バイナリログファイルを処理するためのユーティリティー」を参照してください。バイナリログおよびリカバリ操作の詳細については、セクション7.5「バイナリログを使用したポイントインタイム (増分) リカバリ」を参照してください。

バイナリロギングは、ステートメントまたはトランザクションの完了後すぐに行われますが、すべてのロックがリリースされるかコミットが実行されるよりも前になります。これにより、ログがコミット順に記録されることが保証されます。

非トランザクションテーブルへの更新は、実行後すぐにバイナリログに格納されます。

コミットなしのトランザクション内では、InnoDB テーブルなどのトランザクションテーブルを変更するすべての更新 (UPDATE、DELETE、INSERT) は、サーバーによって COMMIT ステートメントが受け取られるまでキャッシュされます。その時点で、COMMIT が実行される前に mysqld はトランザクション全体をバイナリログに書き込みます。

非トランザクションテーブルへの変更はロールバックできません。ロールバックされるトランザクションに非トランザクションテーブルへの変更が含まれている場合は、非トランザクションテーブルへの変更が確実にレプリケーションされるようにするために、最後に ROLLBACK ステートメントを使用してトランザクション全体がログに記録されます。

トランザクションを処理するスレッドが開始すると、スレッドは binlog_cache_size のバッファーをバッファーステートメントに割り当てます。ステートメントがこれより大きい場合、スレッドはトランザクションを格納する一時ファイルを開きます。スレッドが終了すると、一時ファイルは削除されます。

Binlog_cache_use ステータス変数は、ステートメントを格納するために、このバッファー (および場合によっては一時ファイル) を使用したトランザクションの数を表示します。Binlog_cache_disk_use ステータス変数は、それらのトランザクションのうち、実際に一時ファイルを使用する必要があったものの数を表示します。これらの 2 つの変数は、一時ファイルの使用を避けるために十分な値になるよう binlog_cache_size を調整するために使用することができます。

max_binlog_cache_size システム変数 (デフォルトは最大値の 4G バイト) を使用して、複数ステートメントのトランザクションをキャッシュするために使用する合計サイズを制限することができます。トランザクションがこのバイト数より大きくなると、失敗してロールバックします。最小値は 4096 です。

バイナリログおよび行ベースのロギングを使用している場合、並列挿入は CREATE ... SELECT または INSERT ... SELECT ステートメントの一般的な挿入に変換されます。これは、バックアップ操作中にログを適用することでテーブルの正確なコピーを確実に再作成できるようにするために行われます。ステートメントベースのロギングを使用している場合、元のステートメントがログに書き込まれます。

バイナリログ形式には、バックアップからのリカバリに影響する可能性があるいくつかの既知の制約があります。セクション17.4.1「レプリケーションの機能と問題」を参照してください。

ストアドプログラムのバイナリロギングは、セクション20.7「ストアドプログラムのバイナリロギング」で説明しているように行われます。

MySQL 5.6 のバイナリログ形式は、レプリケーションの機能拡張により以前のバージョンの MySQL とは異なることに注意してください。セクション17.4.2「MySQL バージョン間のレプリケーション互換性」を参照してください。

バイナリログファイルとバイナリログインデックスファイルへの書き込みは、MyISAM テーブルへの書き込みと同じ方法で処理されます。セクションB.5.4.3「MySQL が満杯のディスクを処理する方法」を参照してください。

デフォルトでは、バイナリログは毎回の書き込みごとにディスクと同期されるわけではありません。したがって、(MySQL Server だけでなく) オペレーティングシステムまたはマシンがクラッシュした場合、バイナリログの最後のステートメントが失われる可能性があります。これを防ぐには、sync_binlog システム変数を使用して、N 件のコミットグループが済むごとにバイナリログをディスクに同期させます。セクション5.1.4「サーバーシステム変数」を参照してください。もっとも安全な sync_binlog の値は 1 ですが、もっとも遅い値でもあります。sync_binlog が 1 に設定されたとしても、クラッシュ時にテーブルの内容とバイナリログの内容の間に不一致が生じる可能性があります。

たとえば、InnoDB テーブルを使用していて MySQL Server が COMMIT ステートメントを処理した場合、MySQL Server は作成済みの多くのトランザクションをバイナリログに順番に書き込み、バイナリログを同期させ、次にこのトランザクションを InnoDB にコミットします。これらの 2 つの操作を処理している間にサーバーがクラッシュすると、このトランザクションは再起動時に InnoDB によってロールバックされますが、バイナリログにはまだ存在します。このような問題は、--innodb_support_xa がデフォルトの 1 に設定されていたと仮定すれば解決

されます。このオプションは InnoDB 内の XA トランザクションのサポートに関係しますが、このオプションは、バイナリログおよび InnoDB データファイルが同期されていることも保証します。このオプションによって、より高い安全性が提供されるようにするために、MySQL Server はトランザクションをコミットする前に、バイナリログおよび InnoDB ログをディスクに同期するようにも構成されるようにします。InnoDB ログはデフォルトで同期されるため、バイナリログを同期するために sync_binlog=1 を使用することができます。このオプションの影響は、クラッシュ後の再起動時にトランザクションのロールバックを行なったあと、MySQL Server はロールバック済みの InnoDB トランザクションをバイナリログから削除する、ということになります。これにより、バイナリログは InnoDB テーブルの正確なデータを反映し、したがって、スレーブはロールバック済みのステートメントを受け取らないため、マスターとの同期状態が維持されます。

バイナリログが、必要な長さよりも短いということを、MySQL Server がクラッシュリカバリ中に検出した場合、正常にコミットされた InnoDB トランザクションが、少なくとも 1 つバイナリログから欠落していることを示しています。これは sync_binlog=1 の場合は発生するはずがなく、ディスクまたはファイルシステムは、リクエストされた場合は (されない場合もあります) 実際の同期を実行するため、サーバーは「The binary log file_name is shorter than its expected size」というエラーメッセージを出力します。この場合、このバイナリログは正しくないため、マスターのデータのフレッシュスナップショットからレプリケーションを再起動するようにしてください。

次のシステム変数のセッション値はバイナリログに書き込まれ、バイナリログを構文解析するときにレプリケーションスレーブによって受け付けられます。

- sql_mode (NO_DIR_IN_CREATE モードがレプリケーションされない場合を除きます。セクション 17.4.1.34「レプリケーションと変数」を参照してください)
- foreign_key_checks
- unique_checks
- character_set_client
- collation_connection
- collation_database
- collation_server
- sql_auto_is_null

5.2.4.1 バイナリロギング形式

サーバーはバイナリログに情報を記録するために、いくつかのロギング形式を使用します。採用される形式は、使用される MySQL のバージョンによって異なります。ロギング形式は 3 つあります。

- MySQL のもともとのレプリケーション機能は、マスターからスレーブへの SQL ステートメントの伝播に基づいていました。これはステートメントベースのロギングと呼ばれます。--binlog-format=STATEMENT を指定してサーバーを起動することによって、この形式を使用できます。
- 行ベースのロギングでは、マスターは個々のテーブル行が受ける影響を示すイベントをバイナリログに書き込みます。--binlog-format=ROW を指定してサーバーを起動することによって、サーバーが行ベースのロギングを使用するようにすることができます。
- 3 番目のオプションである混合形式ロギングも選択できます。混合形式ロギングの場合、デフォルトではステートメントベースのロギングが使用されますが、次に示すような特定の状況では ロギングモードが自動的に行ベースに切り替わります。--binlog-format=MIXED オプションを指定して mysqld を開始することによって、MySQL に混合形式ロギングを使用させることができます。

MySQL 5.6 では、デフォルトのバイナリロギング形式は STATEMENT です。

ロギング形式は、使用されているストレージエンジンによって設定または制限される可能性があります。これによって、異なるストレージエンジンを使用しているマスターとスレーブの間で特定のステートメントをレプリケーションするときの問題を除去するのに役立ちます。

ステートメントベースのレプリケーションでは、非決定的なステートメントのレプリケーションに関して問題があることがあります。所定のステートメントがステートメントベースのレプリケーションについて安全かどうかを判断するために、MySQL は、ステートメントベースのロギングを使用してステートメントをレプリケーションできることを保証できるかどうかを判断します。MySQL がこれを保証できない場合、潜在的に信頼できないステートメントにマークを付け、警告 Statement may not be safe to log in statement format を発行します。

これらの問題は、代わりに MySQL の行ベースのレプリケーションを使用することで回避できます。

5.2.4.2 バイナリログ形式の設定

MySQL Server を --binlog-format=type で起動することによって、バイナリロギング形式を明示的に選択することができます。type については次の値がサポートされます。

- STATEMENT の場合、ロギングはステートメントに基づきます。
- ROW の場合、ロギングは行に基づきます。
- MIXED の場合、ロギングは混合形式を使用します。

MySQL 5.6 では、デフォルトのバイナリロギング形式は STATEMENT です。

ロギング形式は実行時でも変更できます。すべてのクライアントについてグローバルに形式を指定するには、binlog_format システム変数のグローバル値を設定します。

```
mysql> SET GLOBAL binlog_format = 'STATEMENT';
mysql> SET GLOBAL binlog_format = 'ROW';
mysql> SET GLOBAL binlog_format = 'MIXED';
```

個別クライアントは binlog_format のセッション値を設定することによって、クライアント自身のステートメントについてのロギング形式を制御することができます。

```
mysql> SET SESSION binlog_format = 'STATEMENT';
mysql> SET SESSION binlog_format = 'ROW';
mysql> SET SESSION binlog_format = 'MIXED';
```

> **注記**
>
> それぞれの MySQL Server は、サーバー独自のバイナリロギング形式のみを設定できます (binlog_format がグローバルスコープまたはセッションスコープのいずれで設定される場合にも当てはまります)。これは、レプリケーションマスターのロギング形式を変更しても、スレーブがそのロギング形式を一致するように変更するわけではないことを意味しています。(STATEMENT モードを使用中の場合、binlog_format システム変数はレプリケーションされず、MIXED または ROW ロギングモードを使用中の場合、レプリケーションされますが、スレーブによって無視されます。)レプリケーションの進行中にマスター上のバイナリロギング形式を変更したり、スレーブ上でそれを変更しなかったりすると、予期しない結果を招いたり、レプリケーションの失敗を招くことがあります。

グローバル値またはセッション値 binlog_format を変更するには、SUPER 権限が必要です。

ロギング形式の手動による切り替えのほかに、スレーブサーバーが自動的にその形式を変更する場合もあります。これは、サーバーが STATEMENT または MIXED のいずれかの形式で実行していて、ROW ロギング形式で書き込まれたイベントをバイナリログ内で検出した場合に発生します。この場合、スレーブはそのイベントについて行ベースのレプリケーションに一時的に切り替わり、そのあとで以前の元の形式に切り替わります。

クライアントがセッションごとにバイナリロギングを設定することには、いくつかの理由があります。

- 多くの小さい変更をデータベースに行うセッションでは、行ベースのロギングを使用した方がよい場合があります。
- WHERE 句で多くの行と一致する更新を実行するセッションでは、多くの行よりも 2、3 件のステートメントをログに記録する方が効率的であるため、ステートメントベースのロギングを使用した方がよい場合があります。
- 一部のステートメントでは、マスター上で多くの実行時間を必要とするが、変更される行がわずか数行ということがあります。したがって、行ベースのロギングを使用してそれらの行をレプリケーションする方が有益なことがあります。

レプリケーション形式を実行時に切り替えることができない例外もあります。

- ストアドファンクションまたはトリガーから実行する場合
- NDB ストレージエンジンが有効な場合
- セッションが現在行ベースのレプリケーションモードで、一時テーブルが開いている場合

これらのいずれかの場合に形式を切り替えようとすると、結果はエラーになります。

InnoDB テーブルを使用中で、トランザクション分離レベルが READ COMMITTED または READ UNCOMMITTED の場合、行ベースのロギングのみを使用することができます。ロギング形式を STATEMENT に

変更することは可能ですが、InnoDB は挿入を実行できないため、実行時にこれを行うと、非常に速くエラーが発生します。

一時テーブルが存在する場合にレプリケーション形式を実行時に切り替えることは推奨されません。これは、一時テーブルはステートメントベースのレプリケーションを使用中の場合にのみログに記録され、行ベースのレプリケーションではこれらはログに記録されないためです。混合形式のレプリケーションでは、一時テーブルは通常はログに記録されます。ユーザー定義関数 (UDF) および UUID() 関数については例外が発生します。

バイナリログ形式を ROW に設定すると、多くの変更は行ベースの形式を使用してバイナリログに書き込まれます。しかし、一部の変更ではステートメントベース形式が使用されます。たとえば、CREATE TABLE、ALTER TABLE、DROP TABLE などのすべての DDL (データ定義言語) ステートメントがこれに該当します。

--binlog-row-event-max-size オプションは、行ベースのレプリケーションが可能なサーバーで使用できます。行は、このオプションの値を越えないバイト単位のサイズを持つチャンクとして、バイナリログに格納されます。この値は 256 の倍数である必要があります。デフォルト値は 1024 です。

> **警告**
>
> レプリケーション用にステートメントベースのロギングを使用している場合、ステートメントの設計方法が、データ変更が非決定的である (つまりクエリーオプティマイザの意向に委ねられるようになっている) ときに、マスターとスレーブのデータが異なることがあります。一般的に、これはレプリケーションの領域外であっても適切なやり方ではありません。この問題についての詳細な説明は、セクションB.5.8「MySQL の既知の問題」を参照してください。

レプリケーションスレーブによって保持されるログについては、セクション17.2.2「レプリケーションリレーおよびステータスログ」を参照してください。

5.2.4.3 混合形式のバイナリロギング形式

MIXED のロギング形式で実行すると、サーバーは次の条件のときにステートメントベースのロギングから行ベースのロギングに自動的に切り替わります。

- 関数に UUID() が含まれているとき。
- AUTO_INCREMENT カラムを含む 1 つ以上のテーブルが更新され、トリガーまたはストアドファンクションが呼び出されたとき。ほかのすべての安全でないステートメントのように、binlog_format = STATEMENT の場合にこれによって警告が生成されます。

  詳細については、セクション17.4.1.1「レプリケーションと AUTO_INCREMENT」を参照してください。
- ビューの本体が行ベースのレプリケーションを必要とするときに、ビューを作成するステートメントもそれを使用するとき。たとえば、ビューを作成するステートメントが UUID() 関数を使用するときに発生します。
- UDF の呼び出しが含まれるとき。
- 非トランザクションテーブルに対して INSERT DELAYED が実行されるとき。
- ステートメントが行ごとにログに記録され、ステートメントを実行したセッションがいずれかの一時テーブルを持つ場合、そのセッションによって使用中のすべての一時テーブルが削除されるまでは、後続のすべてのステートメントで行ごとのロギングが使用されます (一時テーブルにアクセスするステートメントを除きます)。

  これは、いずれかの一時テーブルが実際にログに記録されているかどうかにかかわらず当てはまります。

  行ベースの形式を使用して一時テーブルをログに記録することはできないため、行ベースのロギングが使用されると、そのテーブルを使用する後続のすべてのステートメントは安全でなくなります。セッションが一時テーブルを保持しなくなるまで、サーバーはセッション中に実行されるすべてのステートメントを扱うことによって、この状況を安全でないものとして推定します。
- FOUND_ROWS() または ROW_COUNT() が使用されるとき。(Bug #12092、Bug #30244)
- USER()、CURRENT_USER()、または CURRENT_USER が使用されるとき。(Bug #28086)
- ステートメントが 1 つ以上のシステム変数を参照するとき。(Bug #31168)

  例外　次のシステム変数がセッションスコープ (のみ) で使用された場合、ロギング形式の切り替えは発生しません。

  - auto_increment_increment

- auto_increment_offset
- character_set_client
- character_set_connection
- character_set_database
- character_set_server
- collation_connection
- collation_database
- collation_server
- foreign_key_checks
- identity
- last_insert_id
- lc_time_names
- pseudo_thread_id
- sql_auto_is_null
- time_zone
- timestamp
- unique_checks

  システム変数スコープを決定することについては、セクション5.1.5「システム変数の使用」を参照してください。

  レプリケーションが sql_mode を処理する方法については、セクション17.4.1.34「レプリケーションと変数」を参照してください。

- 関係するテーブルの 1 つが mysql データベース内のログテーブルのとき。
- LOAD_FILE() 関数が使用されるとき。(Bug #39701)

> **注記**
>
> 行ベースのロギングを使用して記述されるべきステートメントをステートメントベースのロギングを使用して実行しようとすると、警告が生成されます。警告は、クライアント (SHOW WARNINGS の出力内) および mysqld エラーログの両方に表示されます。そのようなステートメントが実行されるごとに警告が SHOW WARNINGS テーブルに追加されます。ただし、ログがいっぱいになるのを防ぐために、各クライアントセッションについて警告を生成した最初のステートメントのみがエラーログに書き込まれます。

前述の判断のほかに、テーブル内の情報が更新されるときに使用されるロギング形式が、個々のエンジンによって決定される場合もあります。個々のエンジンのロギング機能は、次のように定義することができます。

- エンジンが行ベースのロギングをサポートする場合、そのエンジンは行ロギング対応といいます。
- エンジンがステートメントベースのロギングをサポートする場合、そのエンジンはステートメントロギング対応といいます。

ある特定のストレージエンジンは、いずれかまたは両方のロギング形式をサポートできます。次の表に、各エンジンによってサポートされる形式を示します。

ストレージエンジン	行ロギングのサポート	ステートメントロギングのサポート
ARCHIVE	はい	はい
BLACKHOLE	はい	はい

ストレージエンジン	行ロギングのサポート	ステートメントロギングのサポート
CSV	はい	はい
EXAMPLE	はい	いいえ
FEDERATED	はい	はい
HEAP	はい	はい
InnoDB	はい	トランザクション分離レベルが REPEATABLE READ または SERIALIZABLE の場合は「はい」、それ以外の場合は「いいえ」。
MyISAM	はい	はい
MERGE	はい	はい
NDB	はい	いいえ

MySQL 5.6 では、ステートメントがログに記録されるかどうか、および使用されるロギングモードは、ステートメントのタイプ (安全、安全でない、またはバイナリインジェクション)、バイナリロギング形式 (STATEMENT、ROW、または MIXED)、およびストレージエンジンのロギング機能 (ステートメント対応、行対応、両方、またはいずれでもない) に応じて決定されます。(バイナリインジェクションとは、ROW 形式を使用してログに記録する必要がある変更のロギングのことをいいます。)

ステートメントがログに記録されるときに警告を出す場合と出さない場合があります。失敗したステートメントはログに記録されませんが、ログにエラーが生成されます。これを次の決定表に示します。ここで、SLC は「ステートメントロギング対応」を示し、RLC は「行ロギング対応」を指します。

条件				アクション	
型	binlog_format	SLC	RLC	エラーまたは警告	ロギング形式
*	*	いいえ	いいえ	Error: Cannot execute statement: 行ロギングにもステートメントロギングにも対応していないエンジンが少なくとも 1 つあるためバイナリロギングは不可能です。	-
安全	STATEMENT	はい	いいえ	-	STATEMENT
安全	MIXED	はい	いいえ	-	STATEMENT
安全	ROW	はい	いいえ	Error: Cannot execute statement: BINLOG_FORMAT = ROW であり、少なくとも 1 つのテーブルが、行ベースのロギングに対応しないストレージエンジンを使用しているため、バイナリロギングは不可能です。	-
安全でない	STATEMENT	はい	いいえ	Warning: Unsafe statement binlogged in statement format: BINLOG_FORMAT = STATEMENTであるため。	STATEMENT
安全でない	MIXED	はい	いいえ	Error: Cannot execute statement: BINLOG_FORMAT	-

条件				アクション	
型	binlog_format	SLC	RLC	エラーまたは警告	ロギング形式
				= MIXED であっても、ストレージエンジンがステートメントベースのロギングに限定されている場合、安全でないステートメントのバイナリロギングは不可能です。	
安全でない	ROW	はい	いいえ	Error: Cannot execute statement: BINLOG_FORMAT = ROW であり、少なくとも 1 つのテーブルが、行ベースのロギングに対応しないストレージエンジンを使用しているため、バイナリロギングは不可能です。	-
行インジェクション	STATEMENT	はい	いいえ	Error: Cannot execute row injection: 少なくとも 1 つのテーブルが、行ベースのロギングに対応していないストレージエンジンを使用しているため、バイナリロギングは不可能です。	-
行インジェクション	MIXED	はい	いいえ	Error: Cannot execute row injection: 少なくとも 1 つのテーブルが、行ベースのロギングに対応していないストレージエンジンを使用しているため、バイナリロギングは不可能です。	-
行インジェクション	ROW	はい	いいえ	Error: Cannot execute row injection: 少なくとも 1 つのテーブルが、行ベースのロギングに対応していないストレージエンジンを使用しているため、バイナリロギングは不可能です。	-
安全	STATEMENT	いいえ	はい	Error: Cannot execute statement: BINLOG_FORMAT = STATEMENT であり、少なくとも 1 つのテーブルが、ステートメントベースのロギングに対応しないストレージエン	-

条件				アクション	
型	binlog_format	SLC	RLC	エラーまたは警告	ロギング形式
				ジンを使用しているため、バイナリロギングは不可能です。	
安全	MIXED	いいえ	はい	-	ROW
安全	ROW	いいえ	はい	-	ROW
安全でない	STATEMENT	いいえ	はい	Error: Cannot execute statement: BINLOG_FORMAT = STATEMENT であり、少なくとも 1 つのテーブルが、ステートメントベースのロギングに対応しないストレージエンジンを使用しているため、バイナリロギングは不可能です。	-
安全でない	MIXED	いいえ	はい	-	ROW
安全でない	ROW	いいえ	はい	-	ROW
行インジェクション	STATEMENT	いいえ	はい	Error: Cannot execute row injection: BINLOG_FORMAT = STATEMENTのため、バイナリロギングは不可能です。	-
行インジェクション	MIXED	いいえ	はい	-	ROW
行インジェクション	ROW	いいえ	はい	-	ROW
安全	STATEMENT	はい	はい	-	STATEMENT
安全	MIXED	はい	はい	-	STATEMENT
安全	ROW	はい	はい	-	ROW
安全でない	STATEMENT	はい	はい	Warning: Unsafe statement binlogged in statement format: BINLOG_FORMAT = STATEMENTであるため。	STATEMENT
安全でない	MIXED	はい	はい	-	ROW
安全でない	ROW	はい	はい	-	ROW
行インジェクション	STATEMENT	はい	はい	Error: Cannot execute row injection: BINLOG_FORMAT = STATEMENTのため、バイナリロギングは不可能です。	-

条件				アクション	
型	binlog_format	SLC	RLC	エラーまたは警告	ロギング形式
行インジェクション	MIXED	はい	はい	-	ROW
行インジェクション	ROW	はい	はい	-	ROW

決定によって警告が生成される場合、標準の MySQL 警告が生成されます (警告は SHOW WARNINGS を使用して確認できます)。この情報は mysqld エラーログにも書き込まれます。ログがいっぱいになるのを防ぐために、エラーは各クライアント接続のエラー発生ごとに 1 つだけログに記録されます。ログメッセージには試行された SQL ステートメントが含められます。

スレーブサーバーが --log-warnings を有効にして起動された場合、スレーブは、スレーブのステータスに関する情報を提供するためのメッセージをエラーログに出力し、この情報には、スレーブがジョブを開始したときのバイナリログおよびリレーログの座標、別のリレーログに切り替える時期、切断後に再接続する時期などがあります。

5.2.4.4 mysql データベーステーブルへの変更に対するロギング形式

mysql データベース内の付与テーブルの内容は、直接的に (INSERT や DELETE などを使用して) または間接的に (GRANT や CREATE USER などを使用して) 変更することができます。mysql データベーステーブルに影響するステートメントは、次のルールを使用してバイナリログに書き込まれます。

- mysql データベーステーブル内のデータを直接変更するデータ操作ステートメントは binlog_format システム変数の設定に従ってログに記録されます。これが関係するステートメントは、INSERT、UPDATE、DELETE、REPLACE、DO、LOAD DATA INFILE、SELECT、および TRUNCATE TABLE などです。
- mysql データベースを間接的に変更するステートメントは、binlog_format の値にかかわらずステートメントとしてログに記録されます。これが関係するステートメントは、GRANT、REVOKE、SET PASSWORD、RENAME USER、CREATE (CREATE TABLE ... SELECT を除くすべての形式)、ALTER (すべての形式)、および DROP (すべての形式) などです。

CREATE TABLE ... SELECT はデータ定義とデータ操作の組み合わせです。CREATE TABLE 部分はステートメント形式を使用してログに記録され、SELECT 部分は binlog_format の値に従ってログに記録されます。

5.2.5 スロークエリーログ

スロークエリーログは、実行に要した時間が long_query_time 秒を超え、少なくとも min_examined_row_limit 行を検査する必要があった SQL ステートメントで構成されます。long_query_time の最小値およびデフォルト値は、それぞれ 0 および 10 です。値はマイクロ秒の精度まで指定できます。ファイルへのロギングの場合、時間はマイクロ秒の部分も含めて書き込まれます。テーブルへのロギングの場合、時間の整数部のみ書き込まれ、マイクロ秒の部分は無視されます。

デフォルトでは、管理ステートメントはログに記録されず、参照にインデックスを使用しないクエリーも記録されません。あとで説明するように、この動作は log_slow_admin_statements および log_queries_not_using_indexes を使用して変更することができます。

初期ロックを取得する時間は実行時間として計算されません。mysqld がスロークエリーログにステートメントを書き込むのは、ステートメントが実行されて、すべてのロックが解放されたあとであるため、ログの順序が実行順と異なる場合があります。

デフォルトでは、スロークエリーログは無効になっています。初期のスロークエリーログ状態を明示的に指定するには、--slow_query_log[={0|1}] を使用します。引数を指定しないか、引数が 1 の場合、--slow_query_log によってログが有効になります。引数が 0 の場合、このオプションによってログが無効になります。ログファイル名を指定するには、--slow_query_log_file=file_name を使用します。ログの出力先を指定するには、(セクション 5.2.1「一般クエリーログおよびスロークエリーログの出力先の選択」で説明されている) --log-output を使用します。

スロークエリーログファイルの名前を指定しない場合、デフォルト名は host_name-slow.log です。サーバーは、別のディレクトリを指定する絶対パス名が指定されないかぎり、データディレクトリ内にファイルを作成します。

実行時にスロークエリーログを無効化または有効化したり、ログファイル名を変更したりするには、グローバルな slow_query_log および slow_query_log_file システム変数を使用します。slow_query_log を 0 (または OFF) にすると、ログが無効化し、1 (または ON) で有効化します。ログファイルの名前を指定するには、slow_query_log_file を指定します。ログファイルがすでに開いている場合、ログファイルが閉じて新しいファイルが開きます。

スロークエリーログが有効な場合、サーバーは --log-output オプションまたは log_output システム変数によって指定されたあらゆる出力先に出力を書き込みます。ログを有効にすると、サーバーはログファイルを開き、ログファイルに起動メッセージを書き込みます。ただし、FILE ログの出力先が選択されないかぎり、ファイルに対するそれ以上のクエリーのロギングは実行されません。出力先が NONE の場合、スロークエリーログが有効な場合であってもサーバーはクエリーを書き込みません。ログ出力先の値に FILE が含まれていない場合、ログファイル名を設定してもロギングへの影響はありません。

--log-short-format オプションを使用する場合、サーバーによってスロークエリーログ (およびバイナリログ) に書き込まれる情報が少なくなります。

スロークエリーログに書き込まれるステートメントに低速の管理ステートメントを含めるには、log_slow_admin_statements システム変数を使用します。管理ステートメントには、ALTER TABLE、ANALYZE TABLE、CHECK TABLE、CREATE INDEX、DROP INDEX、OPTIMIZE TABLE、および REPAIR TABLE が含まれます。

スロークエリーログに書き込まれるステートメントに、行参照についてインデックスを使用しないクエリーを含めるには、log_queries_not_using_indexes システム変数を有効にします。そのようなクエリーがログに記録されると、スロークエリーログが急速に増大することがあります。log_throttle_queries_not_using_indexes システム変数を設定することによって、これらのクエリーに速度制限を課すことが可能です。デフォルトでは、この変数は 0 で、制限がないことを意味します。正の値を指定すると、インデックスを使用しないクエリーのロギングについて分あたりの制限が課されます。そのような最初のクエリーによって 60 秒間のウィンドウが開き、その期間内でサーバーはクエリーを所定の制限までログに記録し、そのあと、追加のクエリーを抑制します。ウィンドウが終了したときに抑制されたクエリーが存在する場合、サーバーはクエリーが存在した数と、それらに要した集計時間とを示すサマリーをログに記録します。インデックスを使用しない次のクエリーをサーバーがログに記録するとき、別の 60 秒間のウィンドウが開始されます。

サーバーは、スロークエリーログにクエリーを書き込むかどうかを判断するために、制御パラメータを次の順序で使用します。

1. クエリーは管理ステートメントでないか、log_slow_admin_statements が有効になっている必要がある。
2. クエリーに少なくとも long_query_time 秒かかっているか、log_queries_not_using_indexes が有効であって、クエリーは行参照にインデックスを使用していない。
3. クエリーは少なくとも min_examined_row_limit 行を検査している必要がある。
4. クエリーは、log_throttle_queries_not_using_indexes 設定によって抑制されていてはならない。

サーバーは、クエリーキャッシュによって処理されるクエリーをスロークエリーログに書き込まず、テーブルに 0 行または 1 行しかないことからインデックスがあることのメリットがないようなクエリーもスロークエリーログに書き込みません。

デフォルトでは、レプリケーションスレーブはレプリケーションされたクエリーをスロークエリーログに書き込みません。これを変更するには、log_slow_slave_statements システム変数を使用します。

MySQL 5.6.3 以降では、スロークエリーログに書き込まれるステートメントのパスワードはサーバーによって書き換えられ、文字どおりに平文で表示されることはありません。MySQL 5.6.3 より前では、ステートメント内のパスワードは書き換えされないため、スロークエリーログを保護するようにしてください。セクション 6.1.2.3「パスワードおよびロギング」を参照してください。

スロークエリーログは、実行に長い時間がかかっているため最適化の候補となるクエリーを見つけるために使用できます。ただし、長いスロークエリーログを調査することは、難しいタスクになる場合があります。これを簡単にするために、mysqldumpslow コマンドを使用してスロークエリーログファイルを処理し、ログに表示されるクエリーを要約することができます。セクション4.6.9「mysqldumpslow — スロークエリーログファイルの要約」を参照してください。

5.2.6 DDL ログ

DDL ログまたはメタデータログは、DROP TABLE や ALTER TABLE などのデータ定義ステートメントによって生成されるメタデータ操作を記録します。MySQL では、メタデータ操作の最中に発生するクラッシュからのリカバリのためにこのログが使用されます。ステートメント DROP TABLE t1, t2 を実行するとき、t1 および t2 の両

方が削除され、それぞれのテーブル削除が完了するようにする必要があります。このタイプの SQL ステートメントの別の例として、ALTER TABLE t3 DROP PARTITION p2 があり、ここではパーティションが完全に削除されることと、この定義がテーブル t3 のパーティションのリストから削除されるようにする必要があります。

前述で説明したようなメタデータ操作の記録は MySQL データディレクトリ内のファイル ddl_log.log に書き込まれます。これはバイナリファイルであって、人間が読むためのものでないため、何らかの形で変更を試みないようにしてください。

ddl_log.log は、メタデータステートメントを記録するために実際に必要になるまで作成されないため、まったく正常な方法で動作している MySQL Server ではこのファイルが存在しない場合があります。

このファイルに関連付けられた、ユーザーが構成可能なサーバーオプションまたは変数はありません。

5.2.7 サーバーログの保守

セクション5.2「MySQL Server ログ」で説明したように、MySQL Server は実行中のアクティビティーの内容を確認するのに役立ついくつかの異なるログファイルを作成することができます。ただし、多くのディスクスペースを占有しすぎないようにするために、これらのファイルを定期的にクリーンアップする必要があります。

ロギングを有効にして MySQL を使用しているとき、古いログファイルをときどきバックアップおよび削除して、新しいファイルへのロギングを開始するよう MySQL に指示することが必要な場合があります。セクション7.2「データベースバックアップ方法」を参照してください。

Linux (Red Hat) のインストールでは、これを行うために mysql-log-rotate スクリプトを使用できます。RPM 配布から MySQL をインストールした場合、このスクリプトは自動的にインストールされているはずです。レプリケーション用にバイナリログを使用している場合、このスクリプトには注意が必要です。バイナリログの内容がすべてのスレーブによって処理されたことに確信が持てるまでは、バイナリログを削除しないでください。

ほかのシステムでは、ログファイルを処理するための、cron (またはその同等物) で開始する短いスクリプトを自分でインストールする必要があります。

バイナリログの場合、指定した日数後にバイナリログファイルを自動的に期限切れにする expire_logs_days システム変数を設定できます (セクション5.1.4「サーバーシステム変数」を参照してください)。レプリケーションを使用している場合、スレーブがマスターよりも遅延できる最大日数を下回らないような変数を設定するようにしてください。バイナリログをオンデマンドで削除するには、PURGE BINARY LOGS ステートメントを使用します (セクション13.4.1.1「PURGE BINARY LOGS 構文」を参照してください)。

ログをフラッシュすることによって、MySQL が新しいログファイルを使用することを強制的に開始できます。ログのフラッシュは、FLUSH LOGS ステートメントを発行したり、mysqladmin flush-logs、mysqladmin refresh、mysqldump --flush-logs、または mysqldump --master-data コマンドを実行したりしたときに実行されます。セクション13.7.6.3「FLUSH 構文」、セクション4.5.2「mysqladmin — MySQL サーバーの管理を行うクライアント」、およびセクション4.5.4「mysqldump — データベースバックアッププログラム」を参照してください。さらに、バイナリログは、サイズが max_binlog_size システム変数の値に達するとフラッシュされます。

FLUSH LOGS は、個々のログの選択的なフラッシュを可能にするためのオプションの修飾子をサポートします (FLUSH BINARY LOGS など)。

ログのフラッシュ操作は、次のことを実行します。

- ログファイルに対する一般クエリーロギングまたはスロークエリーロギングが有効化されている場合、サーバーは一般クエリーログファイルまたはスロークエリーログファイルを閉じて再オープンします。
- バイナリロギングが有効化されている場合、サーバーは現在のバイナリログファイルを閉じ、新しいログファイルを次のシーケンス番号で開きます。
- エラーログがファイルに書き込まれるようにするためにサーバーが --log-error オプションで開始されている場合、サーバーはログファイルを閉じて再オープンします。

ログをフラッシュすると、サーバーは新しいバイナリログファイルを作成します。ただし、一般クエリーログファイルおよびスロークエリーログファイルについては、それらを閉じて再オープンするだけです。Unix で新しいファイルが作成されるようにするには、現在のログファイルをフラッシュする前にそれらを名前変更します。フラッシュ時に、サーバーは新しいログファイルを元の名前で開きます。たとえば、一般クエリーログファイルおよびスロークエリーログファイルの名前が mysql.log および mysql-slow.log の場合、次のような一連のコマンドを使用することができます。

```
shell> cd mysql-data-directory
shell> mv mysql.log mysql.old
shell> mv mysql-slow.log mysql-slow.old
shell> mysqladmin flush-logs
```

Windows では、mv の代わりに rename を使用してください。

この時点で、mysql.old および mysql-slow.old のバックアップを作成し、そのあとこれらをディスクから削除することができます。

エラーログファイルがある場合、同様の方法を使用してこれらをバックアップできます。

ログを無効化することによって、一般クエリーログまたはスロークエリーログを実行時に名前変更できます。

```
SET GLOBAL general_log = 'OFF';
SET GLOBAL slow_query_log = 'OFF';
```

ログが無効化されている場合、コマンド行などの外部からログファイルを名前変更してください。次に、ログをふたたび有効にします。

```
SET GLOBAL general_log = 'ON';
SET GLOBAL slow_query_log = 'ON';
```

この方法はすべてのプラットフォームで動作し、サーバー再起動を必要としません。

5.3 1 つのマシン上での複数の MySQL インスタンスの実行

状況によっては、MySQL の複数インスタンスを単一マシン上で実行する場合もあります。既存の本番設定をそのままにして、新しい MySQL リリースをテストすることもできます。または、ユーザーが自分で管理する異なる mysqld サーバーへのアクセス権を別々のユーザーに与える場合もあります。(たとえば、ユーザーは独立した MySQL インストールを異なるカスタマ用に提供するインターネットサービスプロバイダである場合もあります。)

インスタンスごとに異なる MySQL Server バイナリを使用したり、複数のインスタンスに対して同じバイナリを使用したり、この 2 つの方法を組み合わせたりすることが可能です。たとえば、MySQL 5.5 と MySQL 5.6 からそれぞれサーバーを実行し、異なるバージョンによって所定のワークロードがどのように処理されるかを確認することもできます。または、現在の本番バージョンの複数インスタンスを実行し、それぞれが異なるデータベースのセットを管理する場合もあります。

別個のサーバーバイナリを使用するかどうかにかかわらず、実行する各インスタンスは、いくつかの操作パラメータについて一意の値を使用して構成される必要があります。これにより、インスタンス間で競合するおそれがなくなります。パラメータは、コマンド行、オプションファイル、または環境変数の設定によって設定できます。セクション4.2.3「プログラムオプションの指定」を参照してください。所定のインスタンスによって使用される値を表示するには、インスタンスに接続して、SHOW VARIABLES ステートメントを実行します。

MySQL インスタンスによって管理される主なリソースは、データディレクトリです。各インスタンスが異なるデータディレクトリを使用するようにし、その場所は --datadir=path オプションを使用して指定されます。各インスタンスをインスタンス独自のデータディレクトリで構成する方法と、構成を行わないことの危険についての警告は、セクション5.3.1「複数のデータディレクトリのセットアップ」を参照してください。

異なるデータディレクトリを使用することに加えて、いくつかのほかのオプションは、各サーバーインスタンスについて異なる値を持つ必要があります。

- --port=port_num

  --port は、TCP/IP 接続のポート番号を制御します。または、ホストに複数のネットワークアドレスがある場合、--bind-address を使用して、各サーバーが異なるアドレスを待機するようにすることができます。

- --socket=path

  --socket は、Unix 上の Unix ソケットファイルパスまたは Windows 上の名前付きパイプ名を制御します。Windows の場合、名前付きパイプ接続を許可するように構成されたサーバーについてのみ、個別のパイプ名を指定することが必要です。

- --shared-memory-base-name=name

  このオプションは Windows でのみ使用されます。これは、クライアントが共有メモリーを使用して接続できるようにするために、Windows サーバーによって使用される共有メモリー名を指定します。共有メモリー接続を許可するように構成されたサーバーについてのみ、個別の共有メモリー名を指定することが必要です。

- --pid-file=file_name

  このオプションは、サーバーがプロセス ID を書き込むファイルのパス名を示します。

次のログファイルオプションを使用した場合、これらの値はサーバーごとに異なっている必要があります。

- --general_log_file=file_name
- --log-bin[=file_name]
- --slow_query_log_file=file_name
- --log-error[=file_name]

ログファイルオプションについての詳細な説明は、セクション5.2「MySQL Server ログ」を参照してください。

パフォーマンスを高めるには、次のオプションをサーバーごとに異なるやり方で指定して、いくつかの物理ディスクに負荷を分散させることができます。

- --tmpdir=path

異なる一時ディレクトリを作成すると、特定の一時ファイルを作成した MySQL Server を判別しやすくなります。

複数の MySQL インストールが異なる場所にある場合、--basedir=path オプションを使用して、インストールごとの基本ディレクトリを指定することができます。これにより、各インスタンスは自動的に異なるデータディレクトリ、ログファイル、および PID ファイルを使用します。この理由は、これらの各パラメータのデフォルトが、基本ディレクトリに対して相対的に指定されるためです。この場合、指定する必要があるほかのオプションは、--socket および --port オプションのみです。tar ファイルバイナリ配布を使用して、異なるバージョンの MySQL をインストールするとします。これらは別の場所にインストールされるため、各インストールについてのサーバーを、対応する基本ディレクトリの下でコマンド bin/mysqld_safe を使用して開始することができます。mysqld_safe によって、mysqld に渡される適切な --basedir オプションが決定され、--socket および --port オプションのみを mysqld_safe に指定する必要があります。

あとのセクションで説明するように、適切なコマンドオプションを指定するか、環境変数を設定することによって、追加のサーバーを開始することができます。ただし、複数のサーバーをより永続的に実行する必要がある場合は、オプションファイルを使用して、サーバーに一意となる必要があるオプション値を各サーバーに指定する方が簡単です。--defaults-file オプションは、このために役立ちます。

5.3.1 複数のデータディレクトリのセットアップ

マシン上の各 MySQL インスタンスには、独自のデータディレクトリを持たせるようにします。場所は --datadir=path オプションを使用して指定されます。

新しいインスタンス用のデータディレクトリをセットアップするには、さまざまな方法があります。

- 新しいデータディレクトリを作成する。
- 既存のデータディレクトリをコピーする。

次に、それぞれの方法について詳しく説明します。

> **警告**
>
> 通常、2 つのサーバーが同じデータベース内のデータを更新するようにするべきではありません。このようにすると、使用しているオペレーティングシステムが、障害のないシステムロックをサポートしない場合に、好ましくない意外な結果が生じるおそれがあります。(この警告にもかかわらず) 同じデータディレクトリを使用する複数のサーバーを実行し、これらの口ギングが有効な場合、適切なオプションを使用して、各サーバーについて一意となる口グファイル名を指定する必要があります。そうでなければ、サーバーは同じファイルにログを記録しようとします。
>
> 前述の予防策が遵守されたとしても、この種類のセットアップは MyISAM および MERGE テーブルでのみ機能し、ほかのストレージエンジンでは機能しません。さらに、データディレクトリを複数のサーバー間で共有することに対するこの警告は、NFS 環境では常に該当します。複数の MySQL Server が共通のデータディレクトリに NFS 経由でアクセスできるようにすることは、非常によくない方法です。主な問題は、NFS が速度のボトルネックになることです。これはそのように使用するためのものではありません。NFS のもう 1 つのリスクは、2 つ以上のサーバーが相互に干渉しないような方法を考案する必要があるということです。通常、NFS ファイルロックは lockd デーモンで処理されますが、現在のところ、どのような状況でも 100% の信頼性でロックを実行できるプラットフォームは存在しません。

新規データディレクトリの作成

この方法を使用すると、データディレクトリは MySQL を最初にインストールしたときと同じ状態になります。これは MySQL アカウントのデフォルトセットを持ち、ユーザーデータはありません。

Unix では、mysql_install_db を実行してデータディレクトリを初期化します。セクション2.10.1「Unix 類似システムでのインストール後の手順」を参照してください。

Windows では、データディレクトリは MySQL 配布に含まれています。

- Windows 用の MySQL Zip アーカイブ配布には、未変更のままのデータディレクトリが格納されています。そのような配布を一時的な場所に解凍し、新規インスタンスをセットアップする場所にその data ディレクトリをコピーします。
- Windows MSI パッケージインストーラは、インストールされたサーバーが使用するデータディレクトリを作成してセットアップするだけでなく、インストールディレクトリの下に data という名前の新しい「テンプレート」データディレクトリも作成します。MSI パッケージを使用してインストールが実行されたあと、テンプレートデータディレクトリをコピーして、追加の MySQL インスタンスをセットアップすることができます。

既存のデータディレクトリのコピー

この方法を使用すると、データディレクトリ内に存在するすべての MySQL アカウントまたはユーザーデータは新しいデータディレクトリに引き継がれます。

1. データディレクトリを使用する既存の MySQL インスタンスを停止します。これは、保留中の変更をインスタンスがディスクにフラッシュするクリーンシャットダウンである必要があります。
2. 新規データディレクトリがあるべき場所にデータディレクトリをコピーします。
3. 既存のインスタンスによって使用されるオプションファイル my.cnf または my.ini をコピーします。これは新規インスタンスの基礎となります。
4. 元のデータディレクトリを参照するすべてのパス名が新規データディレクトリを参照するように、新規オプションファイルを変更します。さらに、インスタンスごとに一意でなければならない TCP/IP ポート番号や口グファイルなど、ほかのオプションも変更します。インスタンスごとに一意でなければならないパラメータのリストについては、セクション5.3「1 つのマシン上での複数の MySQL インスタンスの実行」を参照してください。
5. 新規インスタンスを起動し、新規オプションファイルを使用するよう指示します。

5.3.2 Windows 上での複数の MySQL インスタンスの実行

Windows 上で複数のサーバーを実行するには、適切な操作パラメータを付けてコマンド行からそれらを手動で起動するか、複数のサーバーを Windows サービスとしてインストールしてそのように実行することによって行うことができます。MySQL をコマンド行から実行するか、サービスとして実行するための一般的な手順については、セクション2.3「Microsoft Windows に MySQL をインストールする」に記載されています。次のセクションでは、サーバーごとに一意でなければならないデータディレクトリなどのオプションに対して異なる値を指定して、各サーバーを起動する方法について説明します。これらのオプションは、セクション5.3「1 つのマシン上での複数の MySQL インスタンスの実行」にリストされています。

5.3.2.1 Windows コマンド行での複数の MySQL インスタンスの起動

単一の MySQL Server をコマンド行から手動で起動するための手順はセクション2.3.5.5「Windows のコマンド行からの MySQL の起動」に記載されています。複数のサーバーをこの方法で起動する場合、コマンド行またはオプションファイルで適切なオプションを指定することができます。オプションをオプションファイルに配置する方が便利ですが、各サーバーが確実にそれ独自のオプションのセットを取得するようにすることが必要です。これを行うには、サーバーごとにオプションファイルを作成し、サーバーを実行するときに --defaults-file オプションを使用してファイル名をサーバーに指示します。

たとえば mysqld をポート 3307 でデータディレクトリ C:\mydata1 を使用して実行し、mysqld-debug をポート 3308 でデータディレクトリ C:\mydata2 を使用して実行するとします。次の手順を使用します。

1. 各データディレクトリが存在し、付与テーブルを格納する mysql データベースの独自のコピーも含まれていることを確認します。
2. 2 つのオプションファイルを作成します。たとえば、次のような C:\my-opts1.cnf という名前のファイルを作成します。

```
[mysqld]
```

```
datadir = C:/mydata1
port = 3307
```

そして、次のような C:\my-opts2.cnf という名前の 2 番目のファイルを作成します。

```
[mysqld]
datadir = C:/mydata2
port = 3308
```

3. --defaults-file オプションを使用して、サーバー独自のオプションファイルを使用して各サーバーを起動します。

```
C:\> C:\mysql\bin\mysqld --defaults-file=C:\my-opts1.cnf
C:\> C:\mysql\bin\mysqld-debug --defaults-file=C:\my-opts2.cnf
```

各サーバーはフォアグラウンドで起動するため (あとでサーバーが終了するまで新しいプロンプトは表示されない)、これらの 2 つのコマンドを別のコンソールウィンドウで発行する必要があります。

サーバーをシャットダウンするには、適切なポート番号を使用して各サーバーに接続します。

```
C:\> C:\mysql\bin\mysqladmin --port=3307 shutdown
C:\> C:\mysql\bin\mysqladmin --port=3308 shutdown
```

上述のように構成されたサーバーは、クライアントが TCP/IP 経由で接続することを許可します。使用している Windows のバージョンが名前付きパイプをサポートし、名前付きパイプ接続も許可する場合、mysqld または mysqld-debug サーバーを使用し、名前付きパイプを有効にしてその名前を指定するオプションを指定します。名前付きパイプ接続をサポートする各サーバーは、一意のパイプ名を使用する必要があります。たとえば、C:\my-opts1.cnf ファイルは次のように修正されることがあります。

```
[mysqld]
datadir = C:/mydata1
port = 3307
enable-named-pipe
socket = mypipe1
```

2 番目のサーバーで使用する C:\my-opts2.cnf も同様に修正します。そのあと、前に説明したようにサーバーを起動します。

共有メモリー接続を許可するサーバーについても同様の手順が適用されます。--shared-memory オプションでそのような接続を有効化し、--shared-memory-base-name オプションでそれぞれのサーバーに対して一意の共有メモリー名を指定します。

5.3.2.2 Windows サービスとして複数の MySQL インスタンスの起動

Windows 上では、MySQL Server は Windows サービスとして実行することができます。単一の MySQL サービスをインストール、制御、および削除する手順は、セクション2.3.5.7「Windows のサービスとして MySQL を起動する」に記載されています。

複数の MySQL サービスをセットアップするには、各インスタンスが、インスタンスごとに一意でなければならないほかのパラメータを使用することに加えて、異なるサービス名を使用するようにする必要があります。

次の手順について、mysqld サーバーを、C:\mysql-5.5.9 および C:\mysql-5.6.23 にそれぞれインストールされている 2 つの異なるバージョンの MySQL から実行するとします。(5.5.9 を本番サーバーとして実行しているが、5.6.23 を使用したテストも実行したい場合、このような状況になることがあります。)

MySQL を Windows サービスとしてインストールするには、--install または --install-manual オプションを使用します。これらのオプションについては、セクション2.3.5.7「Windows のサービスとして MySQL を起動する」を参照してください。

前述の説明によれば、複数のサービスをセットアップするにはいくつかの方法があります。次の手順では、いくつかの例を説明します。これらのいずれかを試行する前に、既存の MySQL サービスがあればシャットダウンして削除してください。

- 方法 1: いずれかの標準オプションファイル内ですべてのサービスのオプションを指定します。これを行うには、サーバーごとに異なるサービス名を使用します。5.5.9 mysqld をサービス名 mysqld1 で実行し、5.6.23 mysqld をサービス名 mysqld2 で実行するとします。この場合、[mysqld1] グループを 5.5.9 に対して使用し、[mysqld2] グループを 5.6.23 に対して使用することができます。たとえば、C:\my.cnf を次のようにセットアップすることができます。

```
# options for mysqld1 service
[mysqld1]
```

```
basedir = C:/mysql-5.5.9
port = 3307
enable-named-pipe
socket = mypipe1

# options for mysqld2 service
[mysqld2]
basedir = C:/mysql-5.6.23
port = 3308
enable-named-pipe
socket = mypipe2
```

各サービスに対して Windows が正しい実行可能プログラムを登録するようにするために、完全なサーバーパス名を使用して、サービスを次のようにインストールします。

```
C:\> C:\mysql-5.5.9\bin\mysqld --install mysqld1
C:\> C:\mysql-5.6.23\bin\mysqld --install mysqld2
```

サービスを起動するには、サービスマネージャーを使用するか、または該当するサービス名を指定して NET START を使用します。

```
C:\> NET START mysqld1
C:\> NET START mysqld2
```

サービスを停止するには、サービスマネージャーを使用するか、または該当するサービス名を指定して NET STOP を使用します。

```
C:\> NET STOP mysqld1
C:\> NET STOP mysqld2
```

- 方法 2: 各サーバーのオプションを別々のファイルに指定し、サービスをインストールするときに --defaults-file を使用して、使用するファイルを各サーバーに指示します。この場合、それぞれのファイルで [mysqld] グループを使用してオプションをリストするようにします。

  この方法を使用する場合、5.5.9 mysqld のオプションを指定するには、次のようなファイル C:\my-opts1.cnf を作成します。

  ```
  [mysqld]
  basedir = C:/mysql-5.5.9
  port = 3307
  enable-named-pipe
  socket = mypipe1
  ```

  5.6.23 mysqld については、次のようなファイル C:\my-opts2.cnf を作成します。

  ```
  [mysqld]
  basedir = C:/mysql-5.6.23
  port = 3308
  enable-named-pipe
  socket = mypipe2
  ```

  次のようにしてサービスをインストールします (各コマンドを単一行に入力します)。

  ```
  C:\> C:\mysql-5.5.9\bin\mysqld --install mysqld1
          --defaults-file=C:\my-opts1.cnf
  C:\> C:\mysql-5.6.23\bin\mysqld --install mysqld2
          --defaults-file=C:\my-opts2.cnf
  ```

  MySQL Server をサービスとしてインストールし、--defaults-file オプションを使用する場合、サービス名がオプションより前になければなりません。

  サービスをインストールしたあと、前の例と同じ方法でサービスを起動および停止します。

複数のサービスを削除するには、それぞれのサービスに対して mysqld --remove を使用し、サービス名のあとに --remove オプションを指定します。サービス名がデフォルト (MySQL) の場合、サービス名を省略できます。

5.3.3 Unix 上での複数の MySQL インスタンスの実行

Unix 上で複数の MySQL インスタンスを実行するための 1 つの方法は、デフォルトの TCP/IP ポートおよび Unix ソケットファイルが異なる別々のサーバーをコンパイルして、それぞれのサーバーが別々のネットワークインタフェースを待機するようにすることです。インストールごとに異なる基本ディレクトリ内にコンパイルすることで、コンパイル済みのデータディレクトリ、ログファイル、および PID ファイルの場所がサーバーごとに自動的に別々になります。

デフォルトの TCP/IP ポート番号 (3306) および Unix ソケットファイル (/tmp/mysql.sock) に対して既存の 5.5 サーバーが構成されていると仮定します。別の操作パラメータを持つ新しい 5.6.23 サーバーを構成するには、次のような CMake コマンドを使用します。

```
shell> cmake . -DMYSQL_TCP_PORT=port_number \
         -DMYSQL_UNIX_ADDR=file_name \
         -DCMAKE_INSTALL_PREFIX=/usr/local/mysql-5.6.23
```

ここで、port_number および file_name は、デフォルトの TCP/IP ポート番号および Unix ソケットファイルパス名と異なっている必要があり、CMAKE_INSTALL_PREFIX 値は、既存の MySQL インストールが存在する場所とは異なるインストールディレクトリを指定します。

MySQL Server が所定のポート番号を待機している場合、次のコマンドを使用して、基本ディレクトリおよび Unix ソケットファイル名などのいくつかの重要な構成可能変数に対して MySQL Server が使用中の操作パラメータを検出できます。

```
shell> mysqladmin --host=host_name --port=port_number variables
```

このコマンドによって表示される情報を使用すれば、追加のサーバーを構成するときに使用しないオプション値を見分けることができます。

ホスト名として localhost を指定した場合、mysqladmin は TCP/IP ではなく Unix ソケットファイル接続をデフォルトで使用します。接続プロトコルを明示的に指定するには、--protocol={TCP|SOCKET|PIPE|MEMORY} オプションを使用します。

異なる Unix ソケットファイルおよび TCP/IP ポート番号を使用して起動するためだけの理由で新しい MySQL Server をコンパイルする必要はありません。同じサーバーバイナリを使用し、実行時に異なるパラメータ値を使用してそれぞれのバイナリの起動を開始することも可能です。これを行う 1 つの方法は、コマンド行オプションを使用する方法です。

```
shell> mysqld_safe --socket=file_name --port=port_number
```

2 番目のサーバーを起動するには、異なる --socket および --port オプション値を指定し、mysqld_safe に --datadir=path オプションを渡すことによってサーバーが異なるデータディレクトリを使用するようにします。

または、サーバーごとのオプションを別々のオプションファイルに配置し、適切なオプションファイルへのパスを指定する --defaults-file オプションを使用して各サーバーを起動します。たとえば、2 つのサーバーインスタンスのオプションファイルの名前が /usr/local/mysql/my.cnf および /usr/local/mysql/my.cnf2 の場合、次のようなコマンドでサーバーを起動します。

```
shell> mysqld_safe --defaults-file=/usr/local/mysql/my.cnf
shell> mysqld_safe --defaults-file=/usr/local/mysql/my.cnf2
```

同様な効果を得るための別の方法は、環境変数を使用して Unix ソケットファイル名および TCP/IP ポート番号を設定する方法です。

```
shell> MYSQL_UNIX_PORT=/tmp/mysqld-new.sock
shell> MYSQL_TCP_PORT=3307
shell> export MYSQL_UNIX_PORT MYSQL_TCP_PORT
shell> mysql_install_db --user=mysql
shell> mysqld_safe --datadir=/path/to/datadir &
```

これはテスト用に使用するための 2 番目のサーバーを起動する簡単な方法です。この方法の利点は、同じシェルから起動するすべてのクライアントプログラムに対して環境変数設定が適用されるということです。したがって、これらのクライアントに対する接続は 2 番目のサーバーに自動的に送信されます。

MySQL プログラムに影響を及ぼすために使用できるほかの環境変数のリストは、セクション2.12「環境変数」に記載されています。

Unix の場合、複数のサーバーを起動する別の方法として、mysqld_multi スクリプトがあります。セクション 4.3.4「mysqld_multi — 複数の MySQL サーバーの管理」を参照してください。

5.3.4 複数サーバー環境でのクライアントプログラムの使用

クライアント内にコンパイルされたものとは異なるネットワークインタフェースを待機している MySQL Server に対してクライアントプログラムを使用して接続するために、次のいずれかの方法を使用することができます。

- クライアントを起動する際、--host=host_name --port=port_number を指定することによって TCP/IP を使用してリモートサーバーに接続するか、--host=127.0.0.1 --port=port_number を指定することによって TCP/IP を使用してローカルサーバーに接続するか、--host=localhost --socket=file_name を指定することによって Unix ソケットファイルまたは Windows 名前付きパイプを使用してローカルサーバーに接続します。

- クライアントを起動する際、--protocol=TCP を指定することによって TCP/IP を使用して接続するか、--protocol=SOCKET を指定することによって Unix ソケットファイルを使用して接続するか、--protocol=PIPE を指定することによって名前付きパイプを使用して接続するか、--protocol=MEMORY を指定することによって共有メモリーを使用して接続します。TCP/IP 接続では、--host オプションと --port オプションも指定することが必要な場合もあります。ほかの接続タイプでは、--socket オプションを指定して Unix ソケットファイルまたは Windows 名前付きパイプ名を指定したり、--shared-memory-base-name オプションで共有メモリー名を指定したりすることが必要になることもあります。共有メモリー接続は Windows でのみサポートされます。

- Unix の場合、MYSQL_UNIX_PORT および MYSQL_TCP_PORT 環境変数を設定して、Unix ソケットファイルおよび TCP/IP ポート番号を指示してからクライアントを起動します。通常、特定のソケットファイルまたはポート番号を使用する場合、これらの環境変数を設定するためのコマンドを .login ファイルに配置して、ログインするたびにこれらが適用されるようにすることができます。セクション2.12「環境変数」を参照してください。

- デフォルトの Unix ソケットファイルおよび TCP/IP ポート番号をオプションファイルの [client] グループに指定します。たとえば、Windows の C:\my.cnf や、Unix のホームディレクトリにある .my.cnf ファイルを使用できます。セクション4.2.6「オプションファイルの使用」を参照してください。

- C プログラムでは、ソケットファイルまたはポート番号の引数を mysql_real_connect() の呼び出しで指定できます。また、mysql_options() を呼び出して、プログラムにオプションファイルを読み取らせることもできます。セクション23.7.7「C API 関数の説明」を参照してください。

- Perl の DBD::mysql モジュールを使用している場合、MySQL オプションファイルからオプションを読み取ることができます。例:

```
$dsn = "DBI:mysql:test;mysql_read_default_group=client;"
       . "mysql_read_default_file=/usr/local/mysql/data/my.cnf";
$dbh = DBI->connect($dsn, $user, $password);
```

  セクション23.9「MySQL Perl API」を参照してください。

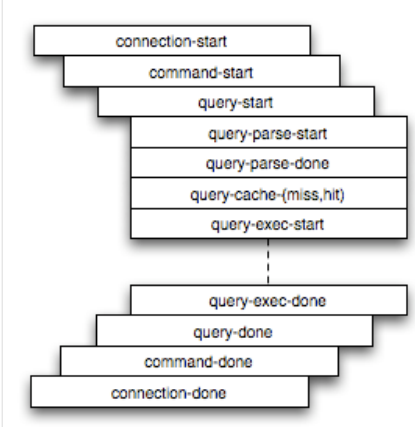
  ほかのプログラミングインタフェースでも、オプションファイルの読み取りのための同様の機能を利用できることがあります。

5.4 DTrace を使用した mysqld のトレース

MySQL Server の DTrace プローブは、MySQL 内でのクエリーの実行と、その処理中に使用されるシステムのさまざまな領域についての情報を提供するように設計されています。プローブの編成およびトリガーが持つ意味として、クエリー全体の実行は 1 レベルのプローブ (query-start および query-done) でモニターできるが、ほかのプローブをモニターすることによって、使用されるロック、ソート方法、さらに行ごとおよびストレージエンジンレベルの実行情報といった観点から、クエリーの実行に関するさらに詳細な情報を連続的に取得できるということがあります。

DTrace プローブは、クライアントからの接続のポイント、クエリーの実行、行レベルの操作、およびバックアウト操作といったクエリー全体のプロセスを追うことができるように編成されています。次の図に示すように、プローブは、典型的なクライアントの接続、実行、接続解除のシーケンス中に特定のシーケンス内で起動されるものと考えることができます。

図 5.1 プラガブルストレージエンジンを使用する MySQL アーキテクチャー



グローバル情報は、さまざまなレベルで DTrace プローブへの引数内で提供されます。グローバル情報とは、接続 ID、ユーザーとホスト、および関係する場合はクエリー文字列で、これらの情報は、キーレベル (connection-start、command-start、query-start、および query-exec-start) で提供されます。プローブが深くなるにつれて、個々の実行にのみ関心を持つか (行レベルのプローブは、データベースおよびテーブル名のみについての情報を提供します)、または行レベルのプローブと概念上の親プローブを組み合わせて特定のクエリーについての情報を提供することが想定されます。この例は、各プローブの形式および引数が提示されるときに提供されます。

MySQL 5.6 では、これらのプラットフォーム上での DTrace プローブについてのサポートが含まれます。

- SPARC、x86、および x86_64 プラットフォーム上での Solaris 10 Update 5 (Solaris 5/08)
- OS X 10.4 以上
- UEK カーネルを持つ Oracle Linux 6 以上 (MySQL 5.6.20 以降)

これらのプラットフォーム上ではプローブの有効化は自動的であるはずです。ビルド中にプローブを明示的に有効または無効にするには、CMake に -DENABLE_DTRACE=1 または -DENABLE_DTRACE=0 オプションを使用してください。

Solaris 以外のプラットフォームに DTrace サポートが含まれている場合、そのプラットフォーム上での mysqld のビルドには、DTrace サポートが含まれます。

追加のリソース

- DTrace および DTrace スクリプトの作成の詳細は、『DTrace ユーザーガイド』をお読みください。
- DTrace の概要については、MySQL Dev Zone の記事「Getting started with DTracing MySQL」を参照してください。

5.4.1 mysqld DTrace プローブリファレンス

MySQL は、機能グループに編成される次の静的プローブをサポートします。

表 5.6 MySQL DTrace プローブ

グループ	プローブ
接続	connection-start、connection-done
コマンド	command-start、command-done
クエリー	query-start、query-done
クエリー解析	query-parse-start、query-parse-done
クエリーキャッシュ	query-cache-hit、query-cache-miss
クエリー実行	query-exec-start、query-exec-done
行レベル	insert-row-start、insert-row-done
	update-row-start、update-row-done
	delete-row-start、delete-row-done
行読み取り	read-row-start、read-row-done
インデックス読み取り	index-read-row-start、index-read-row-done
ロック	handler-rdlock-start、handler-rdlock-done
	handler-wrlock-start、handler-wrlock-done
	handler-unlock-start、handler-unlock-done
ファイルソート	filesort-start、filesort-done
ステートメント	select-start、select-done
	insert-start、insert-done
	insert-select-start、insert-select-done
	update-start、update-done

グループ	プローブ
	multi-update-start、multi-update-done
	delete-start、delete-done
	multi-delete-start、multi-delete-done
ネットワーク	net-read-start、net-read-done、net-write-start、net-write-done
キーキャッシュ	keycache-read-start、keycache-read-block、keycache-read-done、keycache-read-hit、keycache-read-miss、keycache-write-start、keycache-write-block、keycache-write-done

> 注記
>
> プローブから引数データを抽出するとき、各引数は arg0 から始まる argN として使用できます。定義内の各引数を識別するために、これらには記述的な名前が提供されますが、対応する argN パラメータを使用して情報にアクセスする必要があります。

5.4.1.1 接続プローブ

connection-start および connection-done プローブは、接続がソケット経由かネットワーク接続経由かに関係なく、クライアントからの接続を囲みます。

```
connection-start(connectionid, user, host)
connection-done(status, connectionid)
```

- connection-start: 接続および正常なログインおよび認証がクライアントによって実行されたあとでトリガーされます。引数には次の接続情報が含まれます。
  - connectionid: 接続 ID を格納する unsigned long。これは SHOW PROCESSLIST からの出力の Id 値に表示されるプロセス ID と同じです。
  - user: 認証に使用されるユーザー名。匿名ユーザーの場合、値はブランクです。
  - host: クライアント接続のホスト。UNIX ソケットを使用して実行される接続の場合、値はブランクです。
- connection-done: クライアントへの接続がクローズしたときにトリガーされます。引数は次のとおりです。
  - status: 接続がクローズされたときのそのステータス。ログアウト操作は値 0 で、それ以外の接続の終了はゼロ以外の値です。
  - connectionid: クローズされた接続の接続 ID。

次の D スクリプトは個々の接続の平均期間を定量化して要約し、カウントを数値化して 60 秒ごとに情報をダンプします。

```
#!/usr/sbin/dtrace -s


mysql*:::connection-start
{
  self->start = timestamp;
}

mysql*:::connection-done
/self->start/
{
  @ = quantize(((timestamp - self->start)/1000000));
  self->start = 0;
}

tick-60s
{
  printa(@);
}
```

多数のクライアントを持つサーバー上で実行すると、次のような出力が表示されることがあります。

```
  1  57413                     :tick-60s

           value  ------------- Distribution ------------- count
              -1 |                                         0
               0 |@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@ 30011
```

```
     1 |                    59
     2 |                     5
     4 |                    20
     8 |                    29
    16 |                    18
    32 |                    27
    64 |                    30
   128 |                    11
   256 |                    10
   512 |                     1
  1024 |                     6
  2048 |                     8
  4096 |                     9
  8192 |                     8
 16384 |                     2
 32768 |                     1
 65536 |                     1
131072 |                     0
262144 |                     1
524288 |                     0
```

5.4.1.2 コマンドプローブ

コマンドプローブは、その期間中に実行されることがある SQL ステートメントを含む、クライアントコマンドが実行される前と実行されたあとに実行されます。コマンドに含まれる操作として、DB の初期化、COM_CHANGE_USER 操作の使用 (MySQL プロトコルによってサポートされる)、プリペアドステートメントの操作などがあります。これらのコマンドの多くは、PHP や Java などのさまざまなコネクタからの MySQL クライアント API によってのみ使用されます。

```
command-start(connectionid, command, user, host)
command-done(status)
```

- command-start: コマンドがサーバーに送信されたときにトリガーされます。
  - connectionid: コマンドを実行したクライアントの接続 ID。
  - command: 実行されたコマンドを表す整数。設定可能な値を次の表に示します。

値	名前	説明
00	COM_SLEEP	内部スレッド状態
01	COM_QUIT	接続のクローズ
02	COM_INIT_DB	データベースの選択 (USE ...)
03	COM_QUERY	クエリーの実行
04	COM_FIELD_LIST	フィールドのリストの取得
05	COM_CREATE_DB	データベースの作成 (非推奨)
06	COM_DROP_DB	データベースの削除 (非推奨)
07	COM_REFRESH	接続のリフレッシュ
08	COM_SHUTDOWN	サーバーのシャットダウン
09	COM_STATISTICS	統計の取得
10	COM_PROCESS_INFO	プロセスの取得 (SHOW PROCESSLIST)
11	COM_CONNECT	接続の初期化
12	COM_PROCESS_KILL	プロセスの強制終了
13	COM_DEBUG	デバッグ情報の取得
14	COM_PING	Ping
15	COM_TIME	内部スレッド状態
16	COM_DELAYED_INSERT	内部スレッド状態
17	COM_CHANGE_USER	ユーザーの変更
18	COM_BINLOG_DUMP	レプリケーションスレーブまたは mysqlbinlog によって使用されてバイナリログ読み取りを初期化する
19	COM_TABLE_DUMP	レプリケーションスレーブによって使用されてマスターテーブル情報を取得する

値	名前	説明
20	COM_CONNECT_OUT	レプリケーションスレーブによって使用されてサーバーへの接続を口グに記録する
21	COM_REGISTER_SLAVE	登録中にレプリケーションスレーブによって使用される
22	COM_STMT_PREPARE	ステートメントの作成
23	COM_STMT_EXECUTE	ステートメントの実行
24	COM_STMT_SEND_LONG_DATA	拡張データをリクエストするときにクライアントによって使用される
25	COM_STMT_CLOSE	プリペアドステートメントのクローズ
26	COM_STMT_RESET	プリペアドステートメントのリセット
27	COM_SET_OPTION	サーバーオプションの設定
28	COM_STMT_FETCH	プリペアドステートメントのフェッチ

  - user: コマンドを実行するユーザー。
  - host: クライアントホスト。
- command-done: コマンド実行が完了したときにトリガーされます。status 引数には、コマンドが正常に実行されると 0 が格納され、正常に完了する前にステートメントが終了した場合は 1 が格納されます。

command-start および command-done プローブが最適に使用されるのは、ステートメントプローブと組み合わせて実行時間の概要を取得するときです。

5.4.1.3 クエリープローブ

query-start および query-done プローブは、特定のクエリーがサーバーによって受け取られ、クエリーが実行され、情報がクライアントに正常に送信されたときにトリガーされます。

```
query-start(query, connectionid, database, user, host)
query-done(status)
```

- query-start: クライアントからクエリー文字列を受け取ったあとトリガーされます。引数は次のとおりです。
  - query: 送信されたクエリーの完全なテキスト。
  - connectionid: クエリーを送信したクライアントの接続 ID。この接続 ID は、クライアントが最初に接続したときに返される接続 ID と、SHOW PROCESSLIST からの出力の Id 値と同じです。
  - database: クエリーが実行されるデータベース名。
  - user: サーバーへの接続に使用されるユーザー名。
  - host: クライアントのホスト名。
- query-done: クエリーが実行されて、クライアントに情報が返されたときにトリガーされます。このプローブは単一の引数 status を格納し、クエリーが正常に実行されると 0 を返し、エラーが発生した場合は 1 を返します。

次の D スクリプトを使用して、各クエリーの実行時間についての単純なレポートを取得することができます。

```
#!/usr/sbin/dtrace -s

#pragma D option quiet

dtrace:::BEGIN
{
  printf("%-20s %-20s %-40s %-9s\n", "Who", "Database", "Query", "Time(ms)");
}

mysql*:::query-start
{
  self->query = copyinstr(arg0);
  self->connid = arg1;
  self->db    = copyinstr(arg2);
  self->who   = strjoin(copyinstr(arg3),strjoin("@",copyinstr(arg4)));
  self->querystart = timestamp;
}
```

```
mysql*:::query-done
{
  printf("%-20s %-20s %-40s %-9d\n",self->who,self->db,self->query,
       (timestamp - self->querystart) / 1000000);
}
```

前述のスクリプトを実行すると、クエリーの実行時間についての基本的な概要が理解できるはずです。

```
shell> ./query.d
Who                 Database            Query                                    Time(ms)
root@localhost      test                select * from t1 order by i limit 10     0
root@localhost      test                set global query_cache_size=0            0
root@localhost      test                select * from t1 order by i limit 10     776
root@localhost      test                select * from t1 order by i limit 10     773
root@localhost      test                select * from t1 order by i desc limit 10 795
```

5.4.1.4 クエリー解析プローブ

クエリー解析プローブは、元の SQL ステートメントが解析される前と、ステートメントの解析およびステートメントの処理に必要な実行モデルの決定が完了したときにトリガーされます。

```
query-parse-start(query)
query-parse-done(status)
```

- query-parse-start: ステートメントが MySQL クエリーパーサーによって解析される直前にトリガーされます。単一の引数 query は、元のクエリーの完全なテキストを格納する文字列です。
- query-parse-done: 元のステートメントの解析が完了したときにトリガーされます。status は、操作のステータスを記述する整数です。0 は、クエリーが正常に解析されたことを示します。1 は、クエリーの解析が失敗したことを示します。

たとえば、次の D スクリプトを使用して、所定のクエリーを解析するための実行時間をモニターすることができます。

```
#!/usr/sbin/dtrace -s

#pragma D option quiet

mysql*:::query-parse-start
{
  self->parsestart = timestamp;
  self->parsequery = copyinstr(arg0);
}

mysql*:::query-parse-done
/arg0 == 0/
{
  printf("Parsing %s: %d microseconds\n", self->parsequery,((timestamp - self->parsestart)/1000));
}

mysql*:::query-parse-done
/arg0 != 0/
{
  printf("Error parsing %s: %d microseconds\n", self->parsequery,((timestamp - self->parsestart)/1000));
}
```

前述のスクリプトで、query-parse-done に述語を使用することで、プローブのステータス値に基づいて異なる出力が生成されます。

スクリプトを実行して実行をモニターすると、次のようになります。

```
shell> ./query-parsing.d
Error parsing select from t1 join (t2) on (t1.i = t2.i) order by t1.s,t1.i limit 10: 36 ms
Parsing select * from t1 join (t2) on (t1.i = t2.i) order by t1.s,t1.i limit 10: 176 ms
```

5.4.1.5 クエリーキャッシュプローブ

クエリーキャッシュプローブは、いずれかのクエリーを実行するときに起動します。query-cache-hit クエリーは、クエリーがクエリーキャッシュ内に存在するときにトリガーされ、クエリーキャッシュ情報を返すために使用できます。引数には、元のクエリーテキストおよびクエリーについてクエリーキャッシュから返される行数が格納されます。クエリーがクエリーキャッシュ内に存在しないか、クエリーキャッシュが使用可能でない場合、query-cache-miss プローブが代わりにトリガーされます。

```
query-cache-hit(query, rows)
```

```
query-cache-miss(query)
```

- query-cache-hit: クエリーがクエリーキャッシュ内に見つかったときにトリガーされます。最初の引数 query にはクエリーの元のテキストが格納されます。2 番目の引数 rows は、キャッシュされたクエリー内の行数を格納する整数です。
- query-cache-miss: クエリーがクエリーキャッシュ内に見つからなかったときにトリガーされます。最初の引数 query にはクエリーの元のテキストが格納されます。

クエリーキャッシュプローブをメインクエリーのプローブと最適に組み合わせることによって、指定されたクエリーについてクエリーキャッシュを使用したときと使用しないときの時間の違いを調べることができます。たとえば、次の D スクリプトでは、クエリーおよびクエリーキャッシュ情報を組み合わせて、モニター中に出力される情報となります。

```
#!/usr/sbin/dtrace -s

#pragma D option quiet

dtrace:::BEGIN
{
  printf("%-20s %-20s %-40s %2s %-9s\n", "Who", "Database", "Query", "QC", "Time(ms)");
}

mysql*:::query-start
{
  self->query = copyinstr(arg0);
  self->connid = arg1;
  self->db    = copyinstr(arg2);
  self->who   = strjoin(copyinstr(arg3),strjoin("@",copyinstr(arg4)));
  self->querystart = timestamp;
  self->qc = 0;
}

mysql*:::query-cache-hit
{
  self->qc = 1;
}

mysql*:::query-cache-miss
{
  self->qc = 0;
}

mysql*:::query-done
{
  printf("%-20s %-20s %-40s %-2s %-9d\n",self->who,self->db,self->query,(self->qc ? "Y" : "N"),
       (timestamp - self->querystart) / 1000000);
}
```

スクリプトを実行すると、クエリーキャッシュの影響を理解することができます。最初は、クエリーキャッシュは無効化されています。クエリーキャッシュサイズを設定し、クエリーを複数回実行すると、クエリーデータを返すためにクエリーキャッシュが使用されていることが表示されるはずです。

```
shell> ./query-cache.d
root@localhost       test              select * from t1 order by i limit 10     N  1072
root@localhost                       set global query_cache_size=262144       N  0
root@localhost       test              select * from t1 order by i limit 10     N  781
root@localhost       test              select * from t1 order by i limit 10     Y  0
```

5.4.1.6 クエリー実行プローブ

クエリー実行プローブは、クエリーの実行が実際に開始されるときにトリガーされ、トリガーの時期はクエリーキャッシュの解析およびチェックのあとですが、権限チェックまたは最適化より前になります。start プローブと done プローブの差を比較することによって、(クエリーの解析およびその他の要素をただ処理するのではなく) クエリーにサービスを提供するために実際に費やした時間をモニターすることができます。

```
query-exec-start(query, connectionid, database, user, host, exec_type)
query-exec-done(status)
```

> **注記**
>
> query-start および query-exec-start の引数に提供される情報はほぼ同じで、ユーザー、クライアント、および実行されるクエリーについてのコア情報を公開しながら、(query-start を使用して) クエリープロセス全体をモニターするか、(query-exec-start を使用して) 実行のみモニターするかを選択できるように設計されています。

- query-exec-start: 個々のクエリーの実行が開始されたときにトリガーされます。引数は次のとおりです。
  - query: 送信されたクエリーの完全なテキスト。
  - connectionid: クエリーを送信したクライアントの接続 ID。この接続 ID は、クライアントが最初に接続したときに返される接続 ID と、SHOW PROCESSLIST からの出力の Id 値と同じです。
  - database: クエリーが実行されるデータベース名。
  - user: サーバーへの接続に使用されるユーザー名。
  - host: クライアントのホスト名。
  - exec_type: 実行のタイプ。実行タイプはクエリーの内容と送信先によって決定されます。各タイプの値を次の表に示します。

値	説明
0	最上位クエリーである sql_parse から実行されたクエリー。
1	実行されたプリペアドステートメント
2	実行されたカーソルステートメント
3	ストアドプロシージャー内の実行されたクエリー

- query-exec-done: クエリーの実行が完了したときにトリガーされます。このプローブは単一の引数 status を格納し、クエリーが正常に実行されると 0 を返し、エラーが発生した場合は 1 を返します。

5.4.1.7 行レベルプローブ

*row-{start,done} プローブは、行操作がストレージエンジンにプッシュダウンされるたびトリガーされます。たとえば、100 行のデータについて INSERT ステートメントを実行するとき、それぞれの行挿入に対して insert-row-start および insert-row-done プローブがどちらも 100 回トリガーされます。

```
insert-row-start(database, table)
insert-row-done(status)

update-row-start(database, table)
update-row-done(status)

delete-row-start(database, table)
delete-row-done(status)
```

- insert-row-start: 行がテーブルに挿入される前にトリガーされます。
- insert-row-done: 行がテーブルに挿入されたあとにトリガーされます。
- update-row-start: 行がテーブル内で更新される前にトリガーされます。
- update-row-done: 行がテーブル内で更新される前にトリガーされます。
- delete-row-start: 行がテーブルから削除される前にトリガーされます。
- delete-row-done: 行がテーブルから削除される前にトリガーされます。

プローブによってサポートされる引数は、それぞれの場合で対応する start および done プローブと一貫性があります。

- database: データベース名。
- table: テーブル名。
- status: ステータスで、正常なら 0、失敗なら 1。

行レベルのプローブは個々の行アクセスごとにトリガーされるため、これらのプローブは毎秒数千回トリガーされる可能性があり、モニタリングスクリプトおよび MySQL の両方に有害な影響を及ぼす場合があります。DTrace 環境ではパフォーマンスへの悪影響を防ぐために、これらのプローブのトリガーを制限するようにしてください。プローブの使用を控えるようにするか、これらのプローブをレポートするためのカウンタ関数または集計関数を使用し、スクリプトが終了したときか、query-done または query-exec-done プローブの一部としてサマリーを提供するようにしてください。

次のスクリプト例は、大規模なクエリー内でのそれぞれの行操作の期間を要約します。

```
#!/usr/sbin/dtrace -s

#pragma D option quiet

dtrace:::BEGIN
{
  printf("%-2s %-10s %-10s %9s %9s %-s \n",
       "St", "Who", "DB", "ConnID", "Dur ms", "Query");
}

mysql*:::query-start
{
  self->query = copyinstr(arg0);
  self->who   = strjoin(copyinstr(arg3),strjoin("@",copyinstr(arg4)));
  self->db    = copyinstr(arg2);
  self->connid = arg1;
  self->querystart = timestamp;
  self->rowdur = 0;
}

mysql*:::query-done
{
  this->elapsed = (timestamp - self->querystart) /1000000;
  printf("%2d %-10s %-10s %9d %9d %s\n",
       arg0, self->who, self->db,
       self->connid, this->elapsed, self->query);
}

mysql*:::query-done
/ self->rowdur /
{
  printf("%34s %9d %s\n", "", (self->rowdur/1000000), "-> Row ops");
}

mysql*:::insert-row-start
{
  self->rowstart = timestamp;
}

mysql*:::delete-row-start
{
  self->rowstart = timestamp;
}

mysql*:::update-row-start
{
  self->rowstart = timestamp;
}

mysql*:::insert-row-done
{
  self->rowdur += (timestamp-self->rowstart);
}

mysql*:::delete-row-done
{
  self->rowdur += (timestamp-self->rowstart);
}

mysql*:::update-row-done
{
  self->rowdur += (timestamp-self->rowstart);
}
```

データをテーブルに挿入するクエリーと一緒に前述のスクリプトを実行すると、生の行挿入を実行するのに費やした正確な時間をモニターすることができます。

```
St Who       DB           ConnID    Dur ms Query
 0 @localhost test            13     20767 insert into t1(select * from t2)
                                     4827 -> Row ops
```

5.4.1.8 行読み取りプローブ

行読み取りプローブは、行読み取り操作が発生するごとにストレージエンジンレベルでトリガーされます。これらのプローブは、(ストレージエンジンインタフェース内に存在する *row-start プローブとは対照的に) ストレージエンジン内で指定されます。したがって、これらのプローブは、個々のストレージエンジンの行レベル操作およ

びパフォーマンスをモニターするために使用することができます。これらのプローブは、ストレージエンジンの行読み取りインタフェースに関してトリガーされるため、基本クエリー中に非常に多くの回数だけヒットされることがあります。

```
read-row-start(database, table, scan_flag)
read-row-done(status)
```

- read-row-start: 指定された database および table からストレージエンジンによって行が読み取られるときにトリガーされます。scan_flag は、読み取りがテーブルスキャンの一部 (すなわち順次読み取り) の場合に 1 (true) に設定され、読み取りが特定レコードについての場合は 0 (false) に設定されます。
- read-row-done: ストレージエンジン内の行読み取り操作が完了したときにトリガーされます。status は成功の場合は 0 を返し、失敗の場合は正の値を返します。

5.4.1.9 インデックスプローブ

インデックスプローブは、指定されたテーブルについて、いずれかのインデックスを使用して行が読み取られるたびにトリガーされます。プローブは、テーブルに対応するストレージエンジン内でトリガーされます。

```
index-read-row-start(database, table)
index-read-row-done(status)
```

- index-read-row-start: 指定された database および table からストレージエンジンによって行が読み取られたときにトリガーされます。
- index-read-row-done: ストレージエンジン内で、インデックス付けされた行読み取り操作が完了したときにトリガーされます。status は成功の場合は 0 を返し、失敗の場合は正の値を返します。

5.4.1.10 ロックプローブ

ロックプローブは、テーブルのエンジンタイプによって定義されるテーブル上の対応するロックメカニズムを使用する、テーブルについての外部ロックが MySQL によってリクエストされるたびに呼び出されます。ロックには、読み取りロック、書き込みロック、およびロック解除操作という 3 つの異なるタイプがあります。プローブを使用することで、外部ロックルーチンの期間 (つまり、別のロックが解放されるのを待機する時間も含む、ストレージエンジンがロックを実装するために要する時間) およびロックとロック解除プロセスの合計期間を調査することができます。

```
handler-rdlock-start(database, table)
handler-rdlock-done(status)

handler-wrlock-start(database, table)
handler-wrlock-done(status)

handler-unlock-start(database, table)
handler-unlock-done(status)
```

- handler-rdlock-start: 指定された database および table に読み取りロックがリクエストされたときにトリガーされます。
- handler-wrlock-start: 指定された database および table に書き込みロックがリクエストされたときにトリガーされます。
- handler-unlock-start: 指定された database および table にロック解除リクエストが実行されたときにトリガーされます。
- handler-rdlock-done: 読み取りロックリクエストが完了したときにトリガーされます。status は、ロック操作が正常な場合は 0 で、失敗した場合は >0 です。
- handler-wrlock-done: 書き込みロックリクエストが完了したときにトリガーされます。status は、ロック操作が正常な場合は 0 で、失敗した場合は >0 です。
- handler-unlock-done: ロック解除リクエストが完了したときにトリガーされます。status は、ロック解除操作に成功した場合は 0 で、失敗した場合は >0 です。

次のスクリプトを使用して、個々のテーブルのロックおよびロック解除をモニターするための配列を使用し、テーブルロック全体の期間を計算することができます。

```
#!/usr/sbin/dtrace -s

#pragma D option quiet

mysql*:::handler-rdlock-start
```

```
{
  self->rdlockstart = timestamp;
  this->lockref = strjoin(copyinstr(arg0),strjoin("@",copyinstr(arg1)));
  self->lockmap[this->lockref] = self->rdlockstart;
  printf("Start: Lock->Read   %s.%s\n",copyinstr(arg0),copyinstr(arg1));
}

mysql*:::handler-wrlock-start
{
  self->wrlockstart = timestamp;
  this->lockref = strjoin(copyinstr(arg0),strjoin("@",copyinstr(arg1)));
  self->lockmap[this->lockref] = self->rdlockstart;
  printf("Start: Lock->Write  %s.%s\n",copyinstr(arg0),copyinstr(arg1));
}

mysql*:::handler-unlock-start
{
  self->unlockstart = timestamp;
  this->lockref = strjoin(copyinstr(arg0),strjoin("@",copyinstr(arg1)));
  printf("Start: Lock->Unlock %s.%s (%d ms lock duration)\n",
        copyinstr(arg0),copyinstr(arg1),
        (timestamp - self->lockmap[this->lockref])/1000000);
}

mysql*:::handler-rdlock-done
{
  printf("End:   Lock->Read   %d ms\n",
        (timestamp - self->rdlockstart)/1000000);
}

mysql*:::handler-wrlock-done
{
  printf("End:   Lock->Write  %d ms\n",
        (timestamp - self->wrlockstart)/1000000);
}

mysql*:::handler-unlock-done
{
  printf("End:   Lock->Unlock %d ms\n",
        (timestamp - self->unlockstart)/1000000);
}
```

これを実行すると、ロック処理自体の期間と、特定のテーブルのロックの期間の両方に関する情報が得られるはずです。

```
Start: Lock->Read   test.t2
End:   Lock->Read   0 ms
Start: Lock->Unlock test.t2 (25743 ms lock duration)
End:   Lock->Unlock 0 ms
Start: Lock->Read   test.t2
End:   Lock->Read   0 ms
Start: Lock->Unlock test.t2 (1 ms lock duration)
End:   Lock->Unlock 0 ms
Start: Lock->Read   test.t2
End:   Lock->Read   0 ms
Start: Lock->Unlock test.t2 (1 ms lock duration)
End:   Lock->Unlock 0 ms
Start: Lock->Read   test.t2
End:   Lock->Read   0 ms
```

5.4.1.11 ファイルソートプローブ

ファイルソートプローブは、ファイルソート操作がテーブルに適用されるたびにトリガーされます。ファイルソートとそれが発生する条件について詳しくは、セクション8.2.1.15「ORDER BY の最適化」を参照してください。

```
filesort-start(database, table)
filesort-done(status, rows)
```

- filesort-start: テーブル上でファイルソート操作が開始したときにトリガーされます。プローブに対する 2 つの引数である database および table は、ソートされるテーブルを識別します。
- filesort-done: ファイルソート操作が完了するとトリガーされます。2 つの引数が提供され、これらは、status (成功の場合は 0、失敗の場合は 1)、およびファイルソート処理中にソートされた行数です。

次のスクリプトはこの例を示すもので、メインクエリーの期間に加えてファイルソート処理の期間を追跡します。

```
#!/usr/sbin/dtrace -s

#pragma D option quiet

dtrace:::BEGIN
{
  printf("%-2s %-10s %-10s %9s %18s %-s \n",
        "St", "Who", "DB", "ConnID", "Dur microsec", "Query");
}

mysql*:::query-start
{
  self->query = copyinstr(arg0);
  self->who   = strjoin(copyinstr(arg3),strjoin("@",copyinstr(arg4)));
  self->db    = copyinstr(arg2);
  self->connid = arg1;
  self->querystart = timestamp;
  self->filesort = 0;
  self->fsdb = "";
  self->fstable = "";
}

mysql*:::filesort-start
{
 self->filesort = timestamp;
 self->fsdb = copyinstr(arg0);
 self->fstable = copyinstr(arg1);
}

mysql*:::filesort-done
{
  this->elapsed = (timestamp - self->filesort) /1000;
  printf("%2d %-10s %-10s %9d %18d Filesort on %s\n",
        arg0, self->who, self->fsdb,
        self->connid, this->elapsed, self->fstable);
}

mysql*:::query-done
{
  this->elapsed = (timestamp - self->querystart) /1000;
  printf("%2d %-10s %-10s %9d %18d %s\n",
        arg0, self->who, self->db,
        self->connid, this->elapsed, self->query);
}
```

ファイルソートをトリガーする ORDER BY 句を持つクエリーを大規模なテーブル上で実行し、次にテーブル上にインデックスを作成して同じクエリーを繰り返すことで、実行速度の違いを確認することができます。

```
St Who       DB          ConnID      Dur microsec Query
 0 @localhost test            14          11335469 Filesort on t1
 0 @localhost test            14          11335787 select * from t1 order by i limit 100
 0 @localhost test            14         466734378 create index t1a on t1 (i)
 0 @localhost test            14             26472 select * from t1 order by i limit 100
```

5.4.1.12 ステートメントプローブ

個々のステートメントプローブは、さまざまなステートメントタイプについての固有の情報を付与するために提供されます。start プローブの場合、クエリーの文字列のみが引数として提供されます。ステートメントのタイプに応じて、対応する done プローブによって提供される情報は異なります。すべての done プローブについて、操作のステータス (成功の場合は 0、失敗の場合は >0) が提供されます。SELECT、INSERT、INSERT ... (SELECT FROM ...)、DELETE、および DELETE FROM t1,t2 操作の場合、影響を受けた行数が返されます。

UPDATE および UPDATE t1,t2 ... ステートメントの場合、一致した行数と実際に変更された行数が提供されます。これは、対応する WHERE 句によって実際に一致した行数と、変更された行数が、異なる場合があるためです。値が新しい設定値とすでに一致している場合、MySQL では行の値は更新されません。

```
select-start(query)
select-done(status,rows)

insert-start(query)
insert-done(status,rows)

insert-select-start(query)
insert-select-done(status,rows)

update-start(query)
update-done(status,rowsmatched,rowschanged)
```

```
multi-update-start(query)
multi-update-done(status,rowsmatched,rowschanged)

delete-start(query)
delete-done(status,rows)

multi-delete-start(query)
multi-delete-done(status,rows)
```

- select-start: SELECT ステートメントの前にトリガーされます。
- select-done: SELECT ステートメントの終了時にトリガーされます。
- insert-start: INSERT ステートメントの前にトリガーされます。
- insert-done: INSERT ステートメントの終了時にトリガーされます。
- insert-select-start: INSERT ... SELECT ステートメントの前にトリガーされます。
- insert-select-done: INSERT ... SELECT ステートメントの終了時にトリガーされます。
- update-start: UPDATE ステートメントの前にトリガーされます。
- update-done: UPDATE ステートメントの終了時にトリガーされます。
- multi-update-start: 複数テーブルを伴う UPDATE ステートメントの前にトリガーされます。
- multi-update-done: 複数テーブルを伴う UPDATE ステートメントの終了時にトリガーされます。
- delete-start: DELETE ステートメントの前にトリガーされます。
- delete-done: DELETE ステートメントの終了時にトリガーされます。
- multi-delete-start: 複数テーブルを伴う DELETE ステートメントの前にトリガーされます。
- multi-delete-done: 複数テーブルを伴う DELETE ステートメントの終了時にトリガーされます。

ステートメントプローブの引数は次のとおりです。

- query: クエリー文字列。
- status: クエリーのステータス。成功の場合は 0、失敗の場合は >0。
- rows: ステートメントによって影響を受けた行数。これは SELECT について検出された行数、DELETE について削除された行数、および INSERT について正常に挿入された行数を返します。
- rowsmatched: UPDATE 操作の WHERE 句によって一致した行数。
- rowschanged: UPDATE 操作中に実際に変更された行数。

これらのプローブを使用して、ステートメントを実行するユーザーまたはクライアントをモニターすることなくこれらのステートメントタイプの実行をモニターします。この簡単な例は、実行時間を追跡することです。

```
#!/usr/sbin/dtrace -s

#pragma D option quiet

dtrace:::BEGIN
{
  printf("%-60s %-8s %-8s %-8s\n", "Query", "RowsU", "RowsM", "Dur (ms)");
}

mysql*:::update-start, mysql*:::insert-start,
mysql*:::delete-start, mysql*:::multi-delete-start,
mysql*:::multi-delete-done, mysql*:::select-start,
mysql*:::insert-select-start, mysql*:::multi-update-start
{
   self->query = copyinstr(arg0);
   self->querystart = timestamp;
}

mysql*:::insert-done, mysql*:::select-done,
mysql*:::delete-done, mysql*:::multi-delete-done, mysql*:::insert-select-done
/ self->querystart /
{
```

```
    this->elapsed = ((timestamp - self->querystart)/1000000);
    printf("%-60s %-8d %-8d %d\n",
          self->query,
          0,
          arg1,
          this->elapsed);
    self->querystart = 0;
}

mysql*:::update-done, mysql*:::multi-update-done
/ self->querystart /
{
    this->elapsed = ((timestamp - self->querystart)/1000000);
    printf("%-60s %-8d %-8d %d\n",
          self->query,
          arg1,
          arg2,
          this->elapsed);
    self->querystart = 0;
}
```

実行すると、基本的な実行時間と一致した行数を確認できます。

```
Query                                        RowsU    RowsM    Dur (ms)
select * from t2                               0        275      0
insert into t2 (select * from t2)                0        275      9
update t2 set i=5 where i > 75                     110      110      8
update t2 set i=5 where i < 25                     254      134      12
delete from t2 where i < 5                       0        0        0
```

別の方法として、DTrace の集計関数を使用して、個々のステートメントの実行時間を集計する方法があります。

```
#!/usr/sbin/dtrace -s

#pragma D option quiet


mysql*:::update-start, mysql*:::insert-start,
mysql*:::delete-start, mysql*:::multi-delete-start,
mysql*:::multi-delete-done, mysql*:::select-start,
mysql*:::insert-select-start, mysql*:::multi-update-start
{
    self->querystart = timestamp;
}

mysql*:::select-done
{
        @statements["select"] = sum(((timestamp - self->querystart)/1000000));
}

mysql*:::insert-done, mysql*:::insert-select-done
{
        @statements["insert"] = sum(((timestamp - self->querystart)/1000000));
}

mysql*:::update-done, mysql*:::multi-update-done
{
        @statements["update"] = sum(((timestamp - self->querystart)/1000000));
}

mysql*:::delete-done, mysql*:::multi-delete-done
{
        @statements["delete"] = sum(((timestamp - self->querystart)/1000000));
}

tick-30s
{
        printa(@statements);
}
```

前述のスクリプトは各操作の実行に費やした時間を集計し、これを使用すると標準的な一連のテストのベンチマークに役立つことがあります。

```
 delete                                          0
  update                                           0
  insert                                          23
  select                                        2484

  delete                                           0
```

```
  update                                0
  insert                               39
  select                            10744

  delete                                0
  update                               26
  insert                               56
  select                            10944

  delete                                0
  update                               26
  insert                             2287
  select                            15985
```

5.4.1.13 ネットワークプローブ

ネットワークプローブは、MySQL Server およびすべてのタイプのクライアントからのネットワーク経由での情報の転送をモニターします。プローブは次のように定義されます。

```
net-read-start()
net-read-done(status, bytes)
net-write-start(bytes)
net-write-done(status)
```

- net-read-start: ネットワーク読み取り操作が開始されたときにトリガーされます。
- net-read-done: ネットワーク読み取り操作が完了したときにトリガーされます。status は操作の戻りステータスを表す integer で、成功した場合は 0、失敗した場合は 1 です。bytes 引数は、プロセス中に読み取られたバイト数を指定する整数です。
- net-start-bytes: データがネットワークソケットに書き込まれたときにトリガーされます。単一の引数 bytes は、ネットワークソケットに書き込まれるバイト数を指定します。
- net-write-done: ネットワーク書き込み操作が完了したときにトリガーされます。単一の引数 status は操作の戻りステータスを表す整数で、成功した場合は 0 で、失敗した場合は 1 です。

ネットワークプローブを使用して、実行中にネットワーククライアントからの読み取りおよびネットワーククライアントへの書き込みに費やした時間をモニターすることができます。次の D スクリプトでこの例を示します。読み取りまたは書き込みの累積時間とバイト数が計算されます。ネットワークの読み取りまたは書き込み用の個々のプローブが急激に起動されることに対処するために、動的変数サイズが (dynvarsize オプションを使用して) 増加されていることに注意してください。

```
#!/usr/sbin/dtrace -s

#pragma D option quiet
#pragma D option dynvarsize=4m

dtrace:::BEGIN
{
  printf("%-2s %-30s %-10s %9s %18s %-s \n",
        "St", "Who", "DB", "ConnID", "Dur microsec", "Query");
}

mysql*:::query-start
{
  self->query = copyinstr(arg0);
  self->who   = strjoin(copyinstr(arg3),strjoin("@",copyinstr(arg4)));
  self->db    = copyinstr(arg2);
  self->connid = arg1;
  self->querystart = timestamp;
  self->netwrite = 0;
  self->netwritecum = 0;
  self->netwritebase = 0;
  self->netread = 0;
  self->netreadcum = 0;
  self->netreadbase = 0;
}

mysql*:::net-write-start
{
  self->netwrite += arg0;
  self->netwritebase = timestamp;
}

mysql*:::net-write-done
{
```

```
  self->netwritecum += (timestamp - self->netwritebase);
  self->netwritebase = 0;
}

mysql*:::net-read-start
{
  self->netreadbase = timestamp;
}

mysql*:::net-read-done
{
  self->netread += arg1;
  self->netreadcum += (timestamp - self->netreadbase);
  self->netreadbase = 0;
}

mysql*:::query-done
{
  this->elapsed = (timestamp - self->querystart) /1000000;
  printf("%2d %-30s %-10s %9d %18d %s\n",
        arg0, self->who, self->db,
        self->connid, this->elapsed, self->query);
  printf("Net read: %d bytes (%d ms) write: %d bytes (%d ms)\n",
           self->netread, (self->netreadcum/1000000),
           self->netwrite, (self->netwritecum/1000000));
}
```

リモートクライアントを持つマシン上で前述のスクリプトを実行すると、クエリーの実行にかかる時間の約 3 分の 1 が、クライアントに戻すクエリー結果を書き込むことに関連していることが理解できます。

```
St Who                          DB          ConnID      Dur microsec Query
 0 root@::ffff:192.168.0.108    test            31             3495 select * from t1 limit 1000000
Net read: 0 bytes (0 ms) write: 10000075 bytes (1220 ms)
```

5.4.1.14 キーキャッシュプローブ

キーキャッシュプローブは、MyISAM ストレージエンジンと一緒に使用されるインデックスキーキャッシュを使用するときにトリガーされます。プローブは、データがキーキャッシュに読み取られる場合、キャッシュされたキーデータがキャッシュからキャッシュファイルに書き込まれる場合、またはキーキャッシュにアクセスする場合にモニターするために存在します。

キーキャッシュの使用状況は、インデックスファイルからキャッシュにデータが読み取られたか書き込まれた時期を示し、キーキャッシュに割り当てられたメモリーが効率的に使用されているかモニターするために使用できます。あるクエリーの範囲においてキーキャッシュの読み取り数が大きいとき、アクセスされるデータのサイズに対してキーキャッシュが小さすぎることを示している場合があります。

```
keycache-read-start(filepath, bytes, mem_used, mem_free)
keycache-read-block(bytes)
keycache-read-hit()
keycache-read-miss()
keycache-read-done(mem_used, mem_free)
keycache-write-start(filepath, bytes, mem_used, mem_free)
keycache-write-block(bytes)
keycache-write-done(mem_used, mem_free)
```

データをインデックスファイルからキーキャッシュに読み取るとき、プロセスは最初に (keycache-read-start によって指示される) 読み取り操作を初期化し、データのブロックをロードします (keycache-read-block)。このとき、読み取りブロックが識別されたデータと一致したか (keycache-read-hit)、データをさらに読み取る必要がある (keycache-read-miss) 場合があります。読み取り操作が完了したら、keycache-read-done によって読み取りが停止します。

データがインデックスファイルからキーキャッシュに読み取られるのは、指定されたキーがキーキャッシュにまだ存在しない場合に限られます。

- keycache-read-start: キーキャッシュ読み取り操作が開始されたときにトリガーされます。データは指定された filepath から読み取られ、指定された bytes 数だけ読み取られます。mem_used および mem_avail は、キーキャッシュによって現在使用されているメモリーと、キーキャッシュ内で使用可能なメモリーの量を示します。
- keycache-read-block: キーキャッシュが指定された bytes 数のデータのブロックをインデックスファイルからキーキャッシュに読み取ったときにトリガーされます。
- keycache-read-hit: インデックスファイルから読み取ったデータのブロックが、リクエストされたキーデータと一致したときにトリガーされます。

- keycache-read-miss: インデックスファイルから読み取ったデータのブロックが、必要なキーデータと一致しないときにトリガーされます。
- keycache-read-done: キーキャッシュ読み取り操作が完了したときにトリガーされます。mem_used および mem_avail は、キーキャッシュによって現在使用されているメモリーと、キーキャッシュ内で使用可能なメモリーの量を示します。

キーキャッシュ書き込みは、INSERT、UPDATE、または DELETE 操作中にインデックス情報が更新され、キャッシュされているキー情報がインデックスファイルにフラッシュバックされるときに発生します。

- keycache-write-start: キーキャッシュ書き込み操作が開始されるときにトリガーされます。データは指定された filepath に書き込まれ、指定された bytes 数が読み取られます。mem_used および mem_avail は、キーキャッシュによって現在使用されているメモリーと、キーキャッシュ内で使用可能なメモリーの量を示します。
- keycache-write-block: キーキャッシュが指定された bytes 数のデータのブロックをキーキャッシュからインデックスファイルに書き込むときにトリガーされます。
- keycache-write-done: キーキャッシュ書き込み操作が完了したときにトリガーされます。mem_used および mem_avail は、キーキャッシュによって現在使用されているメモリーと、キーキャッシュ内で使用可能なメモリーの量を示します。

第 6 章 セキュリティー

目次

6.1 一般的なセキュリティーの問題 643
6.1.1 セキュリティーガイドライン 644
6.1.2 パスワードをセキュアな状態にする 645
6.1.3 攻撃者に対する MySQL のセキュアな状態の維持 656
6.1.4 セキュリティー関連の mysqld オプションおよび変数 658
6.1.5 MySQL を通常ユーザーとして実行する方法 658
6.1.6 LOAD DATA LOCAL のセキュリティーの問題 659
6.1.7 クライアントプログラミングのセキュリティーガイドライン 660
6.2 MySQL アクセス権限システム 661
6.2.1 MySQL で提供される権限 662
6.2.2 権限システム付与テーブル 666
6.2.3 アカウント名の指定 670
6.2.4 アクセス制御、ステージ 1: 接続の検証 672
6.2.5 アクセス制御、ステージ 2: リクエストの確認 674
6.2.6 権限変更が有効化される時期 676
6.2.7 アクセス拒否エラーの原因 676
6.3 MySQL ユーザーアカウントの管理 680
6.3.1 ユーザー名とパスワード 681
6.3.2 ユーザーアカウントの追加 682
6.3.3 ユーザーアカウントの削除 685
6.3.4 アカウントリソース制限の設定 685
6.3.5 アカウントパスワードの割り当て 687
6.3.6 パスワードの期限切れとサンドボックスモード 688
6.3.7 プラガブル認証 690
6.3.8 MySQL で使用可能な認証プラグイン 693
6.3.9 プロキシユーザー 712
6.3.10 セキュアな接続のための SSL の使用 715
6.3.11 SSH を使用した Windows から MySQL へのリモート接続 725
6.3.12 MySQL Enterprise Audit ログプラグイン 726
6.3.13 SQL ベースの MySQL アカウントアクティビティーの監査 747

MySQL インストール内のセキュリティーについて検討するときは、可能性のあるさまざまなトピックについて考慮し、それらが MySQL サーバーおよび関連するアプリケーションのセキュリティーに及ぼす影響について考慮するようにしてください。

- セキュリティーに影響する一般的な要因。これらには、適切なパスワードの選択、不要な権限をユーザーに付与しないこと、SQL インジェクションおよびデータ損失を防ぐことによるアプリケーションセキュリティーの確保などが含まれます。セクション6.1「一般的なセキュリティーの問題」を参照してください。

- インストール自体のセキュリティー。インストールにおけるデータファイル、ログファイル、およびすべてのアプリケーションファイルを保護することで、許可のない人物による読み取りまたは書き込みができないようにします。詳細については、セクション2.10「インストール後のセットアップとテスト」を参照してください。

- データベースおよびデータベース内で使用中のビューおよびストアドプログラムへのアクセス権限を付与されたユーザーおよびデータベースを含む、データベースシステム自体の内部におけるアクセス制御およびセキュリティー。詳細については、セクション6.2「MySQL アクセス権限システム」およびセクション6.3「MySQL ユーザーアカウントの管理」を参照してください。

- MySQL およびシステムのネットワークセキュリティー。セキュリティーは個々のユーザーに対する権限付与に関係しますが、MySQL が、MySQL サーバーホスト上でローカルからのみ利用できるか、限定されたほかのホストのセットについてのみ利用できるように MySQL を制限したい場合もあります。

- データベースファイル、構成、およびログファイルの十分かつ適切なバックアップを用意します。また、リカバリソリューションを用意するようにし、バックアップから情報を正しくリカバリできることをテストしてください。第7章「バックアップとリカバリ」を参照してください。

6.1 一般的なセキュリティーの問題

このセクションでは、留意するべき一般的なセキュリティーの問題と、攻撃または悪用に対して MySQL インストールをさらにセキュアな状態にするために実行可能なアクションについて説明します。ユーザーアカウントのセットアップおよびデータベースアクセスの検査のために MySQL が使用するアクセス制御システムについての詳細は、セクション2.10「インストール後のセットアップとテスト」を参照してください。

MySQL Server のセキュリティーの問題について、よくある質問のうちのいくつかに対する回答は、セクション A.9「MySQL 5.6 FAQ: セキュリティー」を参照してください。

6.1.1 セキュリティーガイドライン

インターネットに接続したコンピュータ上で MySQL を使用するすべてのユーザーは、セキュリティーに関するもっとも一般的な間違いを回避するために、このセクションを読むようにしてください。

セキュリティーについて検討する際、該当するすべての種類の攻撃 (盗聴、改変、プレイバック、およびサービス妨害) から、(MySQL サーバーだけでなく) サーバーホスト全体を完全に保護することを考慮する必要があります。ここでは可用性およびフォールトトレランスのすべての側面について扱うことはしません。

MySQL では、ユーザーが実行を試行できるすべての接続、クエリー、およびその他の操作に対して、アクセス制御リスト (ACL) に基づくセキュリティーが使用されています。また、MySQL クライアントとサーバーの間で SSL に対応した接続のサポートもあります。ここで説明されている多くの概念は MySQL に固有のものではなく、同じような一般的な考え方は、ほぼすべてのアプリケーションに該当します。

MySQL を実行するときは、次のガイドラインに従ってください。

- mysql データベース内の user テーブルに対するアクセス権限を (MySQL root アカウント以外の) すべてのユーザーに絶対に付与しないでください。これはきわめて重要です。
- MySQL アクセス権限システムのしくみについて学習してください (セクション6.2「MySQL アクセス権限システム」を参照してください)。MySQL へのアクセスを制御するには、GRANT および REVOKE ステートメントを使用します。必要以上の権限を付与しないでください。すべてのホストに権限を付与してはいけません。

  チェックリスト:

  - mysql -u root を試してください。パスワードを尋ねられずにサーバーへの接続に成功する場合、すべてのユーザーが、完全な権限を持つ MySQL root ユーザーとして MySQL サーバーに接続できます。root パスワードの設定に関する情報に特に注意して、MySQL インストール手順を見直してください。セクション 2.10.2「最初の MySQL アカウントのセキュリティー設定」を参照してください。
  - SHOW GRANTS ステートメントを使用して、どのアカウントが何にアクセスできるかをチェックします。REVOKE ステートメントを使用して、不要な権限を削除します。
- 平文パスワードをデータベースに保管しないでください。コンピュータのセキュリティーが損なわれた場合、侵入者はすべてのパスワードのリストを取得して使用することができます。代わりに、SHA2()、SHA1()、MD5()、またはその他の一方向のハッシュ機能を使用して、ハッシュ値を保管してください。

  レインボーテーブルを使用したパスワードのリカバリを避けるために、これらの関数をプレーンテキストパスワードに使用しないようにしてください。代わりに、ソルトとして使用する何らかの文字列を選択して、hash(hash(パスワード)+ソルト) 値を使用してください。
- 辞書からパスワードを選択しないでください。パスワードを解読する特殊なプログラムが存在します。「xfish98」 のようなパスワードでさえも、非常に悪いものです。同じ 「fish」 という単語を、標準 QWERTY キーボードでキー 1 個分左にずらしてタイプした 「duag98」 の方が、ずっと優れています。別の方法は、文の各単語の先頭文字を取ったパスワードを使用することです (たとえば、「Four score and seven years ago」 からは 「Fsasya」 というパスワードができます)。パスワードは覚えやすく入力も簡単ですが、文を知らない人は推測が困難です。この事例で、数字を示す単語をさらに数値に置き換えて 「4 score and 7 years ago」 という句を作成し、「4sa7ya」 というさらに推測困難なパスワードを得ることができます。
- ファイアウォールに投資します。これにより、あらゆる種類のソフトウェアの悪用のうち、少なくとも 50% から保護されます。MySQL をファイアウォールの背後または非武装地帯 (DMZ) に配置します。

  チェックリスト:

  - nmap などのツールを使用して、インターネットから自分のポートをスキャンしてみてください。MySQL はデフォルトでポート 3306 を使用します。このポートは信頼できないホストからアクセス可能であってはいけません。MySQL ポートが開いているかどうかを検査する簡単な方法として、いずれかのリモートマシンから

次のコマンドを試行します。ここで、server_host は MySQL サーバーが実行しているホストのホスト名または IP アドレスです。

```
shell> telnet server_host 3306
```

telnet がハングするか、接続が拒否されれば、ポートはブロックされており、これは期待どおりの結果です。接続を取得して、何らかの文字化けした文字が得られた場合、ポートは開いているため、ポートを開いたままにしておく十分な理由が実際にある場合を除き、ファイアウォールまたはルーターで閉じるようにしてください。

- MySQL にアクセスするアプリケーションは、ユーザーから入力されるすべてのデータを信頼しないようにし、適切な防御的プログラミング技術を使用して記述するようにします。セクション6.1.7「クライアントプログラミングのセキュリティーガイドライン」を参照してください。
- プレーンの (暗号化されていない) データをインターネット経由で送信しないでください。この情報は、情報を傍受する時間と能力を持ち、自身の目的のために情報を使用するすべての人物からアクセスできます。代わりに、SSL や SSH などの暗号化されたプロトコルを使用してください。MySQL は、内部 SSL 接続をサポートします。別の技術として、SSH ポートフォワーディングを使用して、暗号化された (および圧縮された) 通信用トンネルを作成する方法があります。
- tcpdump や strings などのユーティリティーの使用法について学習します。ほとんどの場合、次のようなコマンドを発行することによって、MySQL データストリームが暗号化されていない状態であるかどうかを検査できます。

```
shell> tcpdump -l -i eth0 -w - src or dst port 3306 | strings
```

これは Linux で動作するほか、ほかのシステムでもわずかな変更を行うことで動作するはずです。

> **警告**
>
> 平文データが表示されない場合、これは情報が実際に暗号化されていることを必ずしも意味しているわけではありません。セキュリティーを強化する必要がある場合、セキュリティーの専門家に相談してください。

6.1.2 パスワードをセキュアな状態にする

パスワードは MySQL のいくつかのコンテキストで現れます。次のセクションでは、エンドユーザーおよび管理者がこれらのパスワードをセキュアな状態にし、公開しないようにするためのガイドラインを提供します。また、MySQL がパスワードハッシュを内部で使用する方法や、厳密なパスワードを強制するために使用できるプラグインについても説明します。

6.1.2.1 パスワードセキュリティーのためのエンドユーザーガイドライン

MySQL ユーザーは、パスワードをセキュアな状態にするために次のガイドラインを使用することをお勧めします。

クライアントプログラムを実行して MySQL サーバーに接続する場合、ほかのユーザーからの検出によって公開されるような方法でパスワードを指定することはお勧めできません。クライアントプログラムを実行するときにパスワードを指定するために使用できる方法と、それぞれの方法のリスクの評価について、次の一覧で示します。簡単に言えば、もっとも安全な方法は、クライアントプログラムがパスワードを求めるプロンプトを出すようにするか、適切に保護されたオプションファイルにパスワードを指定する方法です。

- mysql_config_editor ユーティリティーを使用します。これは、.mylogin.cnf という名前の暗号化されたログインファイルに認証情報を格納できます。このファイルは、MySQL Server に接続するための認証情報を取得するために、MySQL クライアントプログラムによってあとで読み取ることができます。セクション4.6.6「mysql_config_editor — MySQL 構成ユーティリティー」を参照してください。
- コマンド行で -pyour_pass または --password=your_pass オプションを使用します。例:

```
shell> mysql -u francis -pfrank db_name
```

これは便利ですがセキュアではありません。一部のシステムでは、使用しているパスワードが、コマンド行を表示するためにほかのユーザーによって起動できる ps などのシステムステータスプログラムによって表示可能になります。MySQL クライアントは通常、クライアントの初期化シーケンス中にコマンド行パスワード引数をゼロで上書きします。ただし、まだ値が表示可能な短い期間があります。また、一部のシステムではこの上書きの方法には効果がなく、パスワードは ps から表示可能になったままになります。(SystemV Unix システムおよびおそらくほかのシステムでもこの問題の影響があります。)

ターミナルウィンドウのタイトルバーに現在のコマンドを表示するようにオペレーティング環境がセットアップされている場合、コマンドがウィンドウのコンテンツ領域からスクロールアウトされて表示されなくなっても、コマンドが実行中であるかぎりパスワードが表示されたままになります。

- コマンド行でパスワード値を指定せずに -p または --password オプションを使用します。この場合、クライアントプログラムはパスワードを対話的に要求します。

```
shell> mysql -u francis -p db_name
Enter password: ********
```

「*」 文字はパスワードを入力したことを示しています。パスワードは入力時に表示されません。

この方法でパスワードを入力する方が、コマンド行でパスワードを指定するよりもセキュアです。これは、パスワードがほかのユーザーに表示されないためです。ただし、このパスワード入力方法は、対話的に実行するプログラムについてのみ適しています。非対話的に実行するスクリプトからクライアントを呼び出す場合、キーボードからパスワードを入力する機会はありません。一部のシステムでは、スクリプトの 1 行目が (誤って) パスワードとして読み取られて解釈されることすらあります。

- パスワードをオプションファイルに保管します。たとえば Unix の場合、ホームディレクトリの .my.cnf ファイルの [client] セクションにパスワードを一覧表示することができます。

```
[client]
password=your_pass
```

パスワードを安全に保持するには、自分以外のすべてのユーザーからファイルにアクセス可能にしてはいけません。このようにするには、ファイルのアクセスモードを 400 または 600 に設定します。例:

```
shell> chmod 600 .my.cnf
```

パスワードを格納する特定のオプションファイルをコマンド行から指定するには、--defaults-file=file_name オプションを使用します。ここで file_name はファイルへのフルパス名です。例:

```
shell> mysql --defaults-file=/home/francis/mysql-opts
```

セクション4.2.6「オプションファイルの使用」には、オプションファイルについてさらに詳しく記載されています。

- MYSQL_PWD 環境変数にパスワードを保存します。セクション2.12「環境変数」を参照してください。

この方法で MySQL パスワードを指定することは非常に危険であるため、使用するべきではありません。ps のバージョンによっては、実行プロセスの環境を表示するオプションがあります。一部のシステムで、MYSQL_PWD を設定した場合、パスワードは ps を実行するすべてのユーザーに公開されます。そのようなバージョンの ps を持たないシステムであっても、ユーザーが処理環境を調査できるほかの方法がないと想定することは賢明ではありません。

Unix の場合、mysql クライアントは実行済みステートメントを履歴ファイルに書き込みます (セクション4.5.1.3「mysql の ロギング」を参照してください)。デフォルトでは、このファイルは .mysql_history という名前で、ユーザーのホームディレクトリに作成されます。パスワードは、CREATE USER、GRANT、SET PASSWORD などの SQL ステートメントではプレーンテキストとして書き込むことができるため、これらのステートメントを使用する場合、ステートメントは履歴ファイルに記録されます。このファイルを安全に保持するには、以前 .my.cnf ファイルについて説明したのと同じ方法である制限アクセスモードを使用します。

コマンドインタプリタが履歴を保持するように構成されている場合、コマンドが保存されるすべてのファイルには、コマンド行で入力された MySQL パスワードが格納されます。たとえば、bash は ~/.bash_history を使用します。そのようなすべてのファイルは、制限アクセスモードにするようにしてください。

6.1.2.2 パスワードセキュリティーについての管理者ガイドライン

データベース管理者は、パスワードをセキュアな状態にするための次のガイドラインを使用するようにしてください。

MySQL はユーザーアカウント用のパスワードを mysql.user テーブルに保管します。このテーブルへのアクセス権を、管理者以外のすべてのアカウントに決して付与しないでください。

アカウントパスワードは期限切れとなることがあり、ユーザーはそれらをリセットする必要があります。セクション6.3.6「パスワードの期限切れとサンドボックスモード」を参照してください。

validate_password プラグインを使用して、許容可能なパスワードについてのポリシーを強制することができます。セクション6.1.2.6「パスワード検証プラグイン」を参照してください。

プラグインディレクトリ (plugin_dir システム変数の値) またはプラグインディレクトリの場所を指定する my.cnf ファイルを変更するためのアクセス権を持つユーザーは、認証プラグインなどのプラグインを置換して、プラグインによって提供される機能を変更することができます。

パスワードが書き込まれる可能性があるログファイルなどのファイルを保護するようにしてください。セクション6.1.2.3「パスワードおよびロギング」を参照してください。

6.1.2.3 パスワードおよびロギング

パスワードは、CREATE USER、GRANT、SET PASSWORD などの SQL ステートメントや、PASSWORD() 関数を呼び出すステートメントにプレーンテキストとして書き込むことができます。これらのステートメントが MySQL サーバーによって、書き込まれたとおりにログに記録された場合、それらのパスワードはログにアクセス可能なすべてのユーザーから利用できるようになります。

MySQL 5.6.3 からステートメントロギングが変更され、次のステートメントについてはパスワードがプレーンテキストで表示されなくなりました。

```
CREATE USER ... IDENTIFIED BY ...
GRANT ... IDENTIFIED BY ...
SET PASSWORD ...
SLAVE START ... PASSWORD = ...            (as of 5.6.4)
CREATE SERVER ... OPTIONS(... PASSWORD ...) (as of 5.6.9)
ALTER SERVER ... OPTIONS(... PASSWORD ...)  (as of 5.6.9)
```

これらのステートメント内のパスワードは、一般クエリーログ、スロークエリーログ、およびバイナリログについて、ステートメントテキストの文字どおりに表示されないように書き換えられます。ほかのステートメントについては書き換えが適用されません。

一般クエリーログの場合、パスワードの書き換えは、--log-raw オプションを使用してサーバーを起動することによって抑制することができます。このオプションは、サーバーによって受け取られるステートメントの正確なテキストを表示する際の診断目的で役立つ場合がありますが、セキュリティー上の理由で本番用途では推奨されません。

監査ログプラグインによって生成される監査ログファイルの内容は、暗号化されません。セキュリティー上の理由から、このファイルは MySQL サーバーおよびログを表示する正当な理由を持つユーザーからのみアクセス可能なディレクトリに書き込むようにしてください。セクション6.3.12.2「監査ログプラグインのセキュリティーに関する考慮事項」を参照してください。

ログファイルが不当に公開されないよう保護するには、サーバーおよびデータベース管理者にのみアクセスを制限したディレクトリにログファイルを配置するようにしてください。mysql データベース内のテーブルにログを記録する場合、これらのテーブルへのアクセス権を、管理者以外のアカウントに決して付与してはいけません。

レプリケーションスレーブは、レプリケーションマスターのパスワードをマスター情報リポジトリに格納し、このリポジトリは、ファイルまたはテーブルのいずれかになります (セクション17.2.2「レプリケーションリレーおよびステータスログ」を参照してください)。このリポジトリはデータベース管理者からのみアクセス可能になるようにします。MySQL 5.6.4 の時点では、パスワードのファイルへの保管に代わる方法は、START SLAVE ステートメントを使用して、マスターに接続するための資格情報を指定する方法です。

パスワードを保管するテーブルまたはログファイルを含むデータベースバックアップは、制限アクセスモードを使用して保護するようにしてください。

6.1.2.4 MySQL でのパスワードハッシュ

MySQL では、mysql データベースの user テーブルにユーザーアカウントがリストされます。各 MySQL アカウントにパスワードを割り当てることができますが、user テーブルはパスワードの平文バージョンを格納せず、パスワードから計算されたハッシュ値を格納します。

MySQL では、クライアントとサーバーの通信の 2 つのフェーズでパスワードが使用されます。

- クライアントがサーバーに接続しようとすると、初期認証ステップがあり、そのステップでは、クライアントが使用するアカウントについての user テーブルに格納されたハッシュ値に一致するハッシュ値を持つパスワードをクライアントが提供する必要があります。
- クライアントが接続したあと、クライアントは (十分な権限がある場合に) user テーブルにリストされているアカウントについてのパスワードハッシュを設定または変更することができます。クライアントは、PASSWORD() 関数を使用してパスワードハッシュを生成するか、パスワード生成ステートメント

(CREATE USER、GRANT、または SET PASSWORD) を使用することによって、これを行うことができます。

言い換えると、クライアントが最初に接続しようとしたとき、サーバーは認証中にハッシュ値を検査します。接続されたクライアントが PASSWORD() 関数を呼び出すか、パスワード生成ステートメントを使用してパスワードを設定または変更する場合、サーバーはハッシュ値を生成します。

MySQL のパスワードハッシュ方式には、次に記述するような歴史があります。これらの変更は、パスワードハッシュ値を計算する PASSWORD() 関数からの結果と、パスワードが格納される user テーブルの構造の変更によって説明されます。

元の (4.1 より前の) ハッシュ方式

元のハッシュ方式では 16 バイト文字列が生成されていました。そのようなハッシュは、次のようになります。

```
mysql> SELECT PASSWORD('mypass');
+--------------------+
| PASSWORD('mypass') |
+--------------------+
| 6f8c114b58f2ce9e   |
+--------------------+
```

アカウントパスワードを格納するために、user テーブルの Password カラムは、この時点で 16 バイト長でした。

4.1 ハッシュ方式

MySQL 4.1 では、セキュリティーを高めてパスワードが傍受されるリスクを低下させる、パスワードハッシュが導入されました。この変更にはさまざまな側面がありました。

- PASSWORD() 関数結果の形式の変更
- Password カラムの拡張
- デフォルトのハッシュ方式による制御
- サーバーに接続しようとするクライアントについて、許可されたハッシュ方式による制御

MySQL 4.1 の変更は、2 段階で行われました。

- MySQL 4.1.0 では 4.1 ハッシュ方式の予備バージョンが使用されていました。この方式は使用期間がたいへん短かったため、以後の説明ではこれについては言及しません。
- MySQL 4.1.1 ではハッシュ方式が変更され、41 バイトの長いハッシュ値が生成されるようになりました。

```
mysql> SELECT PASSWORD('mypass');
+-------------------------------------------+
| PASSWORD('mypass')                        |
+-------------------------------------------+
| *6C8989366EAF75BB670AD8EA7A7FC1176A95CEF4 |
+-------------------------------------------+
```

長いパスワードハッシュ形式は暗号化特性に優れ、長いハッシュに基づくクライアント認証の方が短いハッシュに基づく認証よりセキュアです。

長いパスワードハッシュに対応するために、user テーブルの Password カラムはこの時点で現在の長さである 41 バイトに変更されました。

拡張された Password カラムは、4.1 より前と 4.1 の両方の形式でパスワードハッシュを格納できます。特定のハッシュ値の形式は、2 つの方法で決定されます。

- 長さ: 4.1 および 4.1 より前のハッシュは、それぞれ 41 バイトと 16 バイトです。
- 4.1 形式のパスワードハッシュは常に 「*」 文字で始まりますが、4.1 より前の形式のパスワードでそうなることはありません。

4.1 より前のパスワードハッシュを明示的に生成できるようにするために、2 つの追加変更が行われました。

- ハッシュ値を 16 バイト形式で返す OLD_PASSWORD() 関数が追加されました。
- 互換性目的で、DBA およびアプリケーションがハッシュ方式を制御できるようにする old_passwords システム変数が追加されました。old_passwords 値がデフォルトの 0 のときは、ハッシュで 4.1 方式を使用し、(41

バイトのハッシュ値)、old_passwords=1 と設定すると、ハッシュで 4.1 より前の方式を使用します。この場合、PASSWORD() は 16 バイト値を生成し、OLD_PASSWORD() と同等です

クライアントが接続できる方法を DBA が制御できるようにするために、secure_auth システム変数が追加されました。この変数を無効または有効にしてサーバーを起動することにより、クライアントが 4.1 より前の古いパスワードハッシュ方式を使用して接続することを許可または禁止します。MySQL 5.6.5 より前では、secure_auth はデフォルトで無効になっています。5.6.5 の時点では、よりセキュアなデフォルト構成を促進するために secure_auth はデフォルトで有効になっています (DBA は自らの裁量によりこれを無効化できますが、これは推奨されません)。4.1 より前のパスワードハッシュは非推奨であるため使用しないようにしてください。(アカウントのアップグレード手順については、セクション6.3.8.3「4.1 よりも前のパスワードハッシュ方式と mysql_old_password プラグインからの移行」を参照してください。)

さらに、mysql クライアントは、secure_auth に類似した --secure-auth オプションをサポートしますが、これはクライアント側から指定します。これは 4.1 より前のパスワードハッシュを使用する、セキュアではないアカウントへの接続を防ぐために使用できます。このオプションは、MySQL 5.6.7 より前ではデフォルトで無効になっていますが、それ以降では有効になっています。

ハッシュ方式に関する互換性の問題

MySQL 4.1 での 16 バイトから 41 バイトへの Password カラムの拡張は、インストールまたはアップグレード操作に次のように影響します。

- MySQL の新規インストールを実行する場合、Password カラムは自動的に 41 バイトの長さになります。
- MySQL 4.1 以降から現在のバージョンの MySQL へのアップグレードでは、どちらのバージョンも同じカラム長とパスワードハッシュ方式を使用しているため、Password カラムに関して問題は何も発生しないはずです。
- 4.1 より前のリリースから 4.1 以降へのアップグレードの場合、アップグレード後にシステムテーブルをアップグレードする必要があります。(セクション4.4.7「mysql_upgrade — MySQL テーブルのチェックとアップグレード」を参照してください。)

4.1 ハッシュ方式は MySQL 4.1 (およびそれ以降の) サーバーとクライアントによってのみ認識されるため、互換性の問題が発生する可能性があります。4.1 またはそれより新しいクライアントは、4.1 より前と 4.1 のパスワードハッシュ方式の両方を認識するため、4.1 より前のサーバーに接続できます。ただし、4.1 より前のクライアントが 4.1 またはそれより新しいサーバーに接続しようとすると、問題が発生することがあります。たとえば、4.0 の mysql クライアントは次のエラーメッセージを出して失敗することがあります。

```
shell> mysql -h localhost -u root
Client does not support authentication protocol requested
by server; consider upgrading MySQL client
```

この現象は、MySQL 4.1 以降にアップグレードしたあとで、古い PHP mysql 拡張を使用しようとしたときにも発生します。(Common Problems with MySQL and PHPを参照してください。)

次の説明では、4.1 より前と 4.1 のハッシュ方式の違いについて、およびサーバーをアップグレードしたが 4.1 より前のクライアントとの下位互換性を保持する必要がある場合にどうすればよいかについて記載します。(ただし、古いクライアントによる接続を維持することは推奨されず、可能な場合は避けるようにしてください。)セクションB.5.2.4「クライアントは認証プロトコルに対応できません」から追加情報を見つけることができます。この情報は、MySQL データベースを 4.1 より古いバージョンから 4.1 以降にマイグレーションを実行する PHP プログラマにとって特に重要です。

短いパスワードハッシュと長いパスワードハッシュの違いは、サーバーが認証中にパスワードを使用する方法と、パスワード変更操作を実行する接続対象のクライアントに対してサーバーがパスワードハッシュを生成する方法の両方に関係します。

サーバーが認証中にパスワードハッシュを使用する方法は、Password カラムの幅によって影響します。

- カラムが短い場合、短いハッシュ認証のみが使用されます。
- カラムが長い場合、短いハッシュまたは長いハッシュのいずれかを保持でき、サーバーはいずれかの形式を使用できます。
  - 4.1 より前のクライアントは接続できますが、4.1 より前のハッシュ方式のみ認識するため、短いハッシュを持つアカウントを使用してのみ認証できます。
  - 4.1 以降のクライアントは、短いハッシュまたは長いハッシュを持つアカウントを使用して認証できます。

短いハッシュのアカウントであっても、認証プロセスは、古いクライアントよりも 4.1 以降のクライアントの方が実際には少しセキュアです。セキュリティーの観点で、セキュアでないものからもっともセキュアなものの順に並べると、次のようになります。

- 短いパスワードハッシュで認証する 4.1 より前のクライアント
- 短いパスワードハッシュで認証する 4.1 以降のクライアント
- 長いパスワードハッシュで認証する 4.1 以降のクライアント

接続対象のクライアントに対してサーバーがパスワードハッシュを生成する方法は、Password カラムの幅と、old_passwords システム変数に影響されます。4.1 以降のサーバーは、特定の条件が満たされた場合にのみ長いハッシュを生成し、この条件とは、Password カラムが長い値を保持するための十分な幅を持つこと、および old_passwords が 1 に設定されていてはいけないということです。

これらの条件は次のように適用されます。

- Password カラムは、長いハッシュ (41 バイト) を保持するための十分な幅を持つ必要があります。カラムが更新されておらず、4.1 より前の 16 バイトの幅のままである場合、サーバーは長いハッシュがカラムに適合しないことを認識し、クライアントが PASSWORD() 関数またはパスワード生成ステートメントを使用してパスワード変更操作を実行したときに、サーバーは短いハッシュのみ生成します。これは、4.1 より古い MySQL バージョンから 4.1 以降にアップグレードしたが、Password カラムを拡張するための mysql_upgrade プログラムをまだ実行していない場合に発生する動作です。
- Password カラムが広い場合、短いパスワードハッシュまたは長いパスワードハッシュのいずれかを格納できます。この場合は、代わりに短いパスワードハッシュを生成することをサーバーに強制するために old_passwords システム変数を 1 に設定してサーバーが起動されないかぎり、PASSWORD() 関数およびパスワード生成ステートメントは長いハッシュを生成します。

old_passwords システム変数の目的は、ほかの状況ではサーバーが長いパスワードハッシュを生成するような状況で、4.1 より前のクライアントとの下位互換性を許可することです。このオプションは認証に影響しませんが (4.1 以降のクライアントは、長いパスワードハッシュを持つアカウントを引き続き使用できます)、パスワード変更操作の結果として user テーブル内で長いパスワードハッシュが作成されないようにします。この発生が許される場合、アカウントは 4.1 より前のクライアントによって使用できなくなります。old_passwords を無効にすると、次のような好ましくないシナリオが発生する可能性があります。

- 4.1 より前の古いクライアントが、短いパスワードハッシュを持つアカウントに接続します。
- クライアントが自分のパスワードを変更します。old_passwords が無効なとき、これによりアカウントは長いパスワードハッシュを持つようになります。
- 古いクライアントがアカウントに次回接続しようとしたとき、古いクライアントは接続できません。これはアカウントが、認証中に 4.1 ハッシュ方式を必要とする長いパスワードハッシュを持つためです。(アカウントがユーザーテーブルに長いパスワードハッシュを持つようになると、4.1 より前のクライアントは長いハッシュを認識しないため、4.1 以降のクライアントのみが認証できます。)

このシナリオでは、4.1 より前の古いクライアントをサポートする必要がある場合、old_passwords を 1 に設定しないで 4.1 またはそれより新しいサーバーを実行することは問題があるということを示しています。old_passwords=1 を指定してサーバーを実行することにより、パスワード変更操作は長いパスワードハッシュを生成せず、アカウントが古いクライアントからアクセスできなくなることはありません。(これらのクライアントは、自分のパスワードを変更し、長いパスワードハッシュを結局得ることによって誤って自分自身をロックアウトしてしまうことはありません。)

old_passwords=1 の欠点は、4.1 以降のクライアントについても、作成または変更されたすべてのパスワードが短いハッシュを使用することです。つまり、長いパスワードハッシュによって提供される追加のセキュリティーが失われます。長いハッシュ (たとえば 4.1 クライアントによって使用するためのもの) を持つアカウントを作成するか、長いパスワードハッシュを使用するための既存のアカウントを変更するために、管理者は old_passwords のセッション値を 0 に設定し、グローバル値を 1 に設定したままにすることができます。

```
mysql> SET @@SESSION.old_passwords = 0;
Query OK, 0 rows affected (0.00 sec)

mysql> SELECT @@SESSION.old_passwords, @@GLOBAL.old_passwords;
+--------------------------+--------------------------+
| @@SESSION.old_passwords | @@GLOBAL.old_passwords |
+--------------------------+--------------------------+
|                        0 |                        1 |
+--------------------------+--------------------------+
1 row in set (0.00 sec)

mysql> CREATE USER 'newuser'@'localhost' IDENTIFIED BY 'newpass';
Query OK, 0 rows affected (0.03 sec)

mysql> SET PASSWORD FOR 'existinguser'@'localhost' = PASSWORD('existingpass');
```

```
Query OK, 0 rows affected (0.00 sec)
```

MySQL 4.1 以降では次のシナリオが発生する可能性があります。要因は、Password カラムが短いか長いかということと、長い場合、サーバーが old_passwords を有効または無効のいずれに指定して起動したかということです。

シナリオ 1: ユーザーテーブルの Password カラムが短い場合:

- Password カラムには短いハッシュのみ格納できます。
- サーバーはクライアント認証中に短いハッシュのみ使用します。
- 接続対象のクライアントの場合、PASSWORD() 関数またはパスワード生成ステートメントを含むパスワードハッシュ生成操作では、短いハッシュだけが使用されます。アカウントのパスワードに何らかの変更を行うと、そのアカウントは短いパスワードハッシュを持つことになります。
- Password カラムが短いと、サーバーは短いパスワードハッシュのみ生成するため、old_passwords の値は無関係です。

このシナリオは、4.1 より前の MySQL インストールが 4.1 以降にアップグレードされたが、mysql_upgrade がまだ実行されておらず、mysql データベース内のシステムテーブルがアップグレードされていない場合に発生します。(これはよりセキュアな 4.1 のパスワードハッシュを使用できなくなるため、推奨される構成ではありません。)

シナリオ 2: Password カラムが長く、サーバーは old_passwords=1 を指定して起動される場合:

- Password カラムには短いハッシュまたは長いハッシュを格納できます。
- 4.1 以降のクライアントは、短いハッシュまたは長いハッシュを持つアカウントについて認証できます。
- 4.1 より前のクライアントは、短いハッシュを持つアカウントについてのみ認証できます。
- 接続対象のクライアントの場合、PASSWORD() 関数またはパスワード生成ステートメントを含むパスワードハッシュ生成操作では、短いハッシュだけが使用されます。アカウントのパスワードに何らかの変更を行うと、そのアカウントは短いパスワードハッシュを持つことになります。

このシナリオでは、old_passwords=1 によって長いハッシュを生成できないため、新しく作成されるアカウントは短いパスワードハッシュを持ちます。また、old_passwords を 1 に設定する前に長いハッシュを持つアカウントを作成した場合、old_passwords=1 のときにアカウントのパスワードを変更すると、アカウントには短いパスワードが提供され、長いハッシュの持つセキュリティー上の利点を失うことになります。

長いパスワードハッシュを持つ新しいアカウントを作成するか、長いハッシュを使用するための既存のアカウントを変更するには、以前説明したように、まず old_passwords のセッション値を 0 に設定し、グローバル値を 1 に設定したままにします。

このシナリオでは、サーバーの Password カラムは最新状態ですが、サーバーは 4.1 より前のハッシュ値を生成するようにデフォルトのパスワードハッシュ方式を設定して実行されます。これは推奨される構成ではありませんが、4.1 より前のクライアントおよびパスワードが 4.1 以降にアップグレードされる移行期間に役立つことがあります。これが完了したら、old_passwords=0 および secure_auth=1 を指定してサーバーを実行することが推奨されます。

シナリオ 3: Password カラムが長く、サーバーは old_passwords=0 を指定して起動される場合:

- Password カラムには短いハッシュまたは長いハッシュを格納できます。
- 4.1 以降のクライアントは、短いハッシュまたは長いハッシュを持つアカウントを使用して認証できます。
- 4.1 より前のクライアントは、短いハッシュを持つアカウントを使用する場合のみ認証できます。
- 接続対象のクライアントの場合、PASSWORD() 関数またはパスワード生成ステートメントを含むパスワードハッシュ生成操作では、長いハッシュだけが使用されます。アカウントのパスワードに変更を行うと、そのアカウントは長いパスワードハッシュを持つことになります。

以前示したように、このシナリオの危険性は、短いパスワードハッシュを持つアカウントが 4.1 より前のクライアントからアクセスできなくなることがあるということです。PASSWORD() 関数またはパスワード生成ステートメントを使用して実行したそのようなアカウントのパスワードの変更により、アカウントに長いパスワードハッシュが提供されることになります。その時点以降では、4.1 より前のクライアントはそのアカウントを使用してサーバーに接続できなくなります。クライアントは 4.1 以降にアップグレードする必要があります。

これが問題になる場合、特別な方法でパスワードを変更することができます。たとえば、通常、アカウントのパスワード変更には、SET PASSWORD を次のように使用しています。

```
SET PASSWORD FOR 'some_user'@'some_host' = PASSWORD('mypass');
```

パスワードを変更するが、短いハッシュを作成する場合、代わりに OLD_PASSWORD() 関数を使用します。

```
SET PASSWORD FOR 'some_user'@'some_host' = OLD_PASSWORD('mypass');
```

OLD_PASSWORD() は、短いハッシュを明示的に生成する必要があるような状況で役立ちます。

前の各シナリオの欠点は、次のように要約できます。

シナリオ 1 では、よりセキュアな認証を提供する長いハッシュを活用することができません。

シナリオ 2 では、old_passwords=1 によって、短いパスワードを持つアカウントをアクセス不能にすることはできませんが、注意を払って old_passwords のセッション値を最初に 0 に変更しないかぎり、パスワード変更操作によって、長いハッシュを持つアカウントは短いハッシュに戻されます。

シナリオ 3 では、短いハッシュを使用するアカウントは OLD_PASSWORD() を明示的に使用せずにパスワードを変更した場合、4.1 より前のクライアントからアクセスできなくなります。

短いパスワードハッシュに関連する互換性の問題を回避するための最良の方法は、短いパスワードハッシュを使用しない方法です。

- すべてのクライアントプログラムを MySQL 4.1 以降にアップグレードする。
- old_passwords=0 を指定してサーバーを実行する。
- 長いパスワードハッシュを使用するために、短いパスワードハッシュを持つすべてのアカウントについてのパスワードをリセットする。
- セキュリティーを強化するために、secure_auth=1 を指定してサーバーを実行する。

6.1.2.5 MySQL 4.1 でのパスワードハッシュ変更がアプリケーションプログラムに及ぼす影響

アプリケーション固有の目的で PASSWORD() を使用してパスワードを生成するアプリケーションの場合、MySQL バージョン 4.1 以降にアップグレードすると互換性の問題が生じる可能性があります。PASSWORD() は MySQL アカウントのパスワード管理専用のものであるため、アプリケーションではこれを行うべきではありません。ただし、一部のアプリケーションではそれら固有の目的で PASSWORD() を使用しています。

4.1 より前の MySQL バージョンを 4.1 以降にアップグレードして、長いパスワードハッシュを生成する条件でサーバーを実行すると、アプリケーション固有のパスワード用に PASSWORD() を使用するアプリケーションは破損します。そのような状況で推奨される一連のアクションは、アプリケーションを変更して、ハッシュ値を生成する SHA2()、SHA1()、MD5() などの関数を使用するようにすることです。これが不可能な場合は、古い形式の短いハッシュを生成するために提供されている OLD_PASSWORD() 関数を使用することができます。ただし、OLD_PASSWORD() は将来サポートされなくなる可能性があることに留意してください。

サーバーが old_passwords=1 を指定して実行している場合、サーバーは短いハッシュを生成し、OLD_PASSWORD() は PASSWORD() と同等になります。

MySQL データベースをバージョン 4.0 以前からバージョン 4.1 以降にマイグレーションを実行する PHP プログラマは、MySQL and PHPを参照してください。

6.1.2.6 パスワード検証プラグイン

validate_password プラグイン (MySQL 5.6.6 の時点で使用可能) は、パスワードをテストしてセキュリティーを向上させるために使用することができます。このプラグインは、2 つの機能を実装します。

- 平文の値として指定されるパスワードを割り当てるステートメントで、値は現在のパスワードポリシーと照合して検査され、弱い場合は拒否されます (ステートメントは ER_NOT_VALID_PASSWORD エラーを返します)。これは、CREATE USER、GRANT、および SET PASSWORD ステートメントに影響します。PASSWORD() および OLD_PASSWORD() 関数への引数として指定されるパスワードも検査されます。
- パスワード候補の強さは、VALIDATE_PASSWORD_STRENGTH() SQL 関数を使用して評価でき、これはパスワード引数を取得して、0 (弱い) から 100 (強い) までの整数を返します。

たとえば、次のステートメントの平文パスワードが検査されます。デフォルトのパスワードポリシーではパスワードに最低 8 文字の長さが要求されるため、パスワードが弱いことからステートメントはエラーを生成します。

```
mysql> SET PASSWORD = PASSWORD('abc');
ERROR 1819 (HY000): Your password does not satisfy the current policy
```

```
requirements
```

ハッシュ済みの値として指定されたパスワードは、元のパスワード値が得られないため検査されません。

```
mysql> SET PASSWORD = '*0D3CED9BEC10A777AEC23CCC353A8C08A633045E';
Query OK, 0 rows affected (0.01 sec)
```

パスワード検査を制御するパラメータは、validate_password_xxx の形式の名前を持つシステム変数の値として利用できます。これらの変数を変更してパスワード検査を構成することができます。パスワード検証プラグインのオプションおよび変数を参照してください。

パスワード検査の 3 つのレベルは、LOW、MEDIUM、および STRONG です。デフォルトは MEDIUM で、これを変更するには、validate_password_policy の値を変更します。これらのポリシーにより、実装されるパスワードテストはますます厳密になります。次の記述はデフォルトのパラメータ値を示しており、これらは適切なシステム変数を変更することによって変更できます。

- LOW ポリシーは、パスワードの長さのみテストします。パスワードは少なくとも 8 文字の長さでなければなりません。
- MEDIUM ポリシーは、パスワードが最低 1 つの数値文字を含み、1 つの小文字および大文字を含み、1 つの特殊文字 (英数字以外) を含む必要があるという条件を追加します。
- STRONG ポリシーは、パスワードの 4 文字以上の部分文字列が、(辞書ファイルが指定された場合に) 辞書ファイル内の単語と一致してはならないという条件を追加します。

validate_password プラグインがインストールされていない場合、validate_password_xxx システム変数は利用できず、ステートメント内のパスワードは検査されず、VALIDATE_PASSWORD_STRENGTH() は常に 0 を返します。たとえば、アカウントに 8 文字より短いパスワードを割り当てることができます。

パスワード検証プラグインのインストール

パスワード検証プラグインの名前は validate_password です。サーバーから使用できるようにするには、プラグインライブラリのオブジェクトファイルが MySQL プラグインディレクトリ (plugin_dir システム変数によって指定されたディレクトリ) に存在する必要があります。必要に応じて、サーバーの起動時に、プラグインディレクトリの場所をサーバーに指示する plugin_dir の値を設定します。

サーバー起動時にプラグインをロードするには、--plugin-load オプションを使用して、プラグインを格納するオブジェクトファイルの名前を指定します。このプラグインのロード方式では、サーバーを起動するたびにオプションを指定する必要があります。たとえば、my.cnf ファイルに次の行を入力します。

```
[mysqld]
plugin-load=validate_password.so
```

システム上のオブジェクトファイルのサフィクスが .so とは異なる場合、正しいサフィクスに置き換えてください (たとえば Windows の場合は .dll)。

または、プラグインを実行時に登録するには (必要に応じて拡張子を変更して) 次のステートメントを使用します。

```
mysql> INSTALL PLUGIN validate_password SONAME 'validate_password.so';
```

INSTALL PLUGIN は、プラグインをロードします。また、後続の通常のサーバー起動のたびにプラグインがロードされるように、そのプラグインを mysql.plugins テーブルに登録します。

プラグインが INSTALL PLUGIN によって以前登録されているか、--plugin-load を指定してロードされた場合、--validate-password オプションをサーバー起動時に使用して、プラグインの有効化を制御できます。たとえば、プラグインをロードして、実行時に削除されないようにするには、次のオプションを使用します。

```
[mysqld]
plugin-load=validate_password.so
validate-password=FORCE_PLUS_PERMANENT
```

パスワード検証プラグインを使用せずにサーバーが実行することを回避することが望まれる場合、--validate-password に FORCE または FORCE_PLUS_PERMANENT の値を指定して使用することで、プラグインが正常に初期化しない場合にサーバー起動を強制的に失敗させるようにします。

プラグインのインストールについての一般的な情報は、セクション5.1.8「サーバープラグイン」を参照してください。プラグインのインストールを検証するには、INFORMATION_SCHEMA.PLUGINS テーブルを調査するか、SHOW PLUGINS ステートメントを使用します。セクション5.1.8.2「サーバープラグイン情報の取得」を参照してください。

パスワード検証プラグインのオプションおよび変数

validate_password プラグインの有効化を制御するには、次のオプションを使用します。

- --validate-password[=value]

コマンド行形式	--validate-password[=value]
導入	5.6.6
型	列挙
デフォルト	ON
有効な値	ON OFF FORCE FORCE_PLUS_PERMANENT

このオプションは、サーバーが起動時に validate_password プラグインをロードする方法を制御します。値はセクション5.1.8.1「プラグインのインストールおよびアンインストール」に記載されているような、プラグインロードオプション用に指定可能ないずれかの値とする必要があります。たとえば、--validate-password=FORCE_PLUS_PERMANENT の場合、プラグインをロードし、サーバーの実行中にプラグインが削除されないようにするようサーバーに指示します。

このオプションは、validate_password プラグインが INSTALL PLUGIN で以前登録されていたか、--plugin-load でロードされている場合にのみ利用できます。パスワード検証プラグインのインストールを参照してください。

validate_password プラグインがインストールされている場合、パスワード検査を制御するパラメータを示すいくつかのシステム変数を公開します。

```
mysql> SHOW VARIABLES LIKE 'validate_password%';
+--------------------------------------+--------+
| Variable_name                        | Value  |
+--------------------------------------+--------+
| validate_password_dictionary_file    |        |
| validate_password_length             | 8      |
| validate_password_mixed_case_count   | 1      |
| validate_password_number_count       | 1      |
| validate_password_policy             | MEDIUM |
| validate_password_special_char_count | 1      |
+--------------------------------------+--------+
```

パスワードの検査方法を変更するために、これらの変数はいずれもサーバー起動時に設定でき、これらのほとんどは実行時にも設定できます。次のリストは、各変数の意味を説明したものです。

- validate_password_dictionary_file

導入	5.6.6
システム変数	validate_password_dictionary_file
スコープ	グローバル
動的 (≥ 5.6.26)	はい
動的 (≤ 5.6.25)	いいえ
型	ファイル名

パスワードを検査するために validate_password プラグインによって使用される辞書ファイルのパス名。この変数は、そのプラグインがインストールされないかぎり利用できません。

デフォルトでは、この変数は空の値を持ち、辞書検査は実行されません。辞書検査を有効にするには、この変数を空白でない値に設定する必要があります。ファイル名が相対パスとして指定された場合、サーバーのデータディレクトリを基準として解釈されます。この内容は小文字で記載し、1 行に 1 つの単語としてください。内容は、utf8 の文字セットを持つものとして処理されます。許可される最大のファイルサイズは 1M バイトです。

パスワード検査中に辞書ファイルが使用されるようにするには、パスワードポリシーを 2 (STRONG) に設定する必要があります。validate_password_policy システム変数の説明を参照してください。これが true である場

合、長さが 4 から 100 までのパスワードの各部分文字列が辞書ファイル内の単語と比較されます。いずれかが一致すると、パスワードが拒否されます。比較では大文字と小文字が区別されません。

VALIDATE_PASSWORD_STRENGTH() の場合、パスワードは STRONG を含むすべてのポリシーと照合して検査されるため、強さの評価には validate_password_policy 値に関係なく辞書検査が含まれます。

サーバーが実行中に辞書ファイルに変更を行なった場合、変更を認識させるにはサーバーの再起動が必要です。

- validate_password_length

導入	5.6.6
システム変数	validate_password_length
スコープ	グローバル
動的	はい
型	数値
デフォルト	8
最小値	0

validate_password プラグインによって検査されるパスワードに必要な最小の文字数。この変数は、そのプラグインがインストールされないかぎり利用できません。

validate_password_length の最小値は、関連するほかのいくつかのシステム変数の関数です。MySQL 5.6.10 の時点では、サーバーでは次の式の値より小さい値は設定されません。

```
validate_password_number_count
+ validate_password_special_char_count
+ (2 * validate_password_mixed_case_count)
```

前述の制約のため validate_password プラグインが validate_password_length の値を調整した場合、プラグインはエラーログファイルにメッセージを書き込みます。

- validate_password_mixed_case_count

導入	5.6.6
システム変数	validate_password_mixed_case_count
スコープ	グローバル
動的	はい
型	数値
デフォルト	1
最小値	0

パスワードポリシーが MEDIUM またはそれより強い場合、validate_password プラグインによって検査されるパスワードに必要な小文字および大文字の最大数。この変数は、そのプラグインがインストールされないかぎり利用できません。

- validate_password_number_count

導入	5.6.6
システム変数	validate_password_number_count
スコープ	グローバル
動的	はい
型	数値
デフォルト	1
最小値	0

パスワードポリシーが MEDIUM またはそれより強い場合、validate_password プラグインによって検査されるパスワードに必要な数値文字 (数字) の最大数。この変数は、そのプラグインがインストールされないかぎり利用できません。

- validate_password_policy

導入	5.6.6
システム変数	validate_password_policy
スコープ	グローバル
動的	はい
型	列挙
デフォルト	1
有効な値	0 1 2

validate_password プラグインによって適用されるパスワードポリシー。この変数は、そのプラグインがインストールされないかぎり利用できません。

validate_password_policy 値は、数値 0、1、2 または対応する記号値 LOW、MEDIUM、STRONG を使用して指定できます。次の表では、それぞれのポリシーに対して実施されるテストについて説明します。長さテストの場合、必要な長さは validate_password_length システム変数の値です。同様に、ほかのテストについて必要な値は、ほかの validate_password_xxx 変数によって提供されます。

ポリシー	実施されるテスト
0 または LOW	長さ
1 または MEDIUM	長さ。数値、小文字、大文字、および特殊文字
2 または STRONG	長さ。数値、小文字、大文字、および特殊文字。辞書ファイル

> **注記**
>
> MySQL 5.6.10 以前では、validate_password_policy の名前は validate_password_policy_number でした。

- validate_password_special_char_count

導入	5.6.6
システム変数	validate_password_special_char_count
スコープ	グローバル
動的	はい
型	数値
デフォルト	1
最小値	0

パスワードポリシーが MEDIUM またはそれより強い場合、validate_password プラグインによって検査されるパスワードに必要な英数字以外の文字の最大数。この変数は、そのプラグインがインストールされないかぎり利用できません。

6.1.3 攻撃者に対する MySQL のセキュアな状態の維持

MySQL サーバーに接続するときは、パスワードを使用するようにしてください。接続において、パスワードは平文で送信されません。クライアント接続シーケンス中のパスワード処理は、きわめてセキュアな状態を維持するように MySQL 4.1.1 でアップグレードされました。4.1.1 より前の形式のパスワードをまだ使用している場合、暗号化アルゴリズムは新しいアルゴリズムほど強くありません。クライアントとサーバーの間のトラフィックを傍受できる利口な攻撃者は、多少の労力をかければパスワードを解読することができます。(さまざまなパスワード処理方法についての説明は、セクション6.1.2.4「MySQL でのパスワードハッシュ」を参照してください。)

ほかのすべての情報はテキストとして送信され、接続を観察できるすべてのユーザーによって読み取ることができます。クライアントとサーバーの間の接続が、信頼できないネットワークを介して行われ、そのことに不安がある場合、圧縮されたプロトコルを使用して、トラフィックの解読をさらに困難にすることができます。また、MySQL の内部 SSL サポートを使用して、接続をさらにセキュアな状態にすることもできます。セクション

6.3.10「セキュアな接続のための SSL の使用」を参照してください。または SSH を使用して、MySQL サーバーと MySQL クライアントの間で暗号化された TCP/IP 接続を実現します。オープンソース SSH クライアントは http://www.openssh.org/ から見つけることができ、オープンソースと商用の SSH クライアントの比較は http://en.wikipedia.org/wiki/Comparison_of_SSH_clients にあります。

MySQL システムをセキュアな状態にするには、次の推奨事項についてよく検討するようにしてください。

- すべての MySQL アカウントがパスワードを持つことを要求します。クライアントプログラムは、それを実行中の人物の ID を必ずしも認識しているわけではありません。クライアント/サーバーアプリケーションでは、ユーザーがクライアントプログラムに任意のユーザー名を指定できることが一般的です。たとえば、other_user にパスワードがない場合、mysql プログラムを mysql -u other_user db_name として呼び出すことによって、すべてのユーザーがこのプログラムを使用してほかのユーザーとして接続することができます。すべてのアカウントにパスワードがある場合、ほかのユーザーのアカウントを使用した接続は、もっと難しくなります。

  パスワードの設定方法についての説明は、セクション6.3.5「アカウントパスワードの割り当て」を参照してください。

- データベースディレクトリ内の読み取りまたは書き込み権限を持つ Unix ユーザーアカウントのみが、mysqld の実行に使用されるアカウントであるようにしてください。

- MySQL サーバーを Unix root ユーザーとして絶対に実行しないでください。これを行うと、FILE 権限を持つすべてのユーザーが、root としてサーバーにファイルを作成させることができるため (~root/.bashrc など)、非常に危険です。これを防ぐために、mysqld は --user=root オプションを使用して明示的に指定された場合を除き、root として実行することを拒否します。

  mysqld は、権限のない普通のユーザーとしても実行できます (また、そのように実行するべきです)。mysql という名前の別の Unix アカウントを作成して、すべてをさらにセキュアな状態にすることができます。このアカウントは、MySQL の管理にのみ使用してください。mysqld を別の Unix ユーザーとして開始するには、サーバーオプションを指定した my.cnf オプションファイルの [mysqld] グループ内のユーザー名を指定する user オプションを追加します。例:

  ```
  [mysqld]
  user=mysql
  ```

  これにより、サーバーを手動で起動した場合も、mysqld_safe または mysql.server を使用して起動した場合も、指定のユーザーでサーバーが起動します。詳細は、セクション6.1.5「MySQL を通常ユーザーとして実行する方法」を参照してください。

  root 以外の Unix ユーザーとして mysqld を実行しても、user テーブル内の root ユーザー名を変更する必要があるということを意味するわけではありません。MySQL アカウントのユーザー名は、Unix アカウントのユーザー名とは何の関係もありません。

- 管理者以外のユーザーに FILE 権限を付与しないでください。この権限を持つすべてのユーザーは、mysqld デーモンの権限で、ファイルシステムのあらゆる場所のファイルに書き込むことができます。これは、権限テーブルを実装するファイルを格納するサーバーのデータディレクトリを含みます。FILE 権限の操作をもう少し安全にするために、SELECT ... INTO OUTFILE で生成されたファイルは既存のファイルを上書きせず、すべてのユーザーによって書き込み可能になります。

  FILE 権限は、すべてのユーザーが読み取り可能であるか、サーバーを実行している Unix ユーザーがアクセスできる、すべてのファイルを読み取る場合にも使用できます。この権限を使用して、すべてのファイルをデータベーステーブルに読み取ることができます。これは不正使用される可能性があり、たとえば LOAD DATA を使用して /etc/passwd をテーブルにロードし、次に SELECT を使用してこれを表示することができます。

  ファイルを読み取りおよび書き込みできる場所を制限するには、secure_file_priv システムを特定のディレクトリに設定します。セクション5.1.4「サーバーシステム変数」を参照してください。

- 管理者以外のユーザーに PROCESS または SUPER 権限を付与しないでください。mysqladmin processlist および SHOW PROCESSLIST の出力には、現在実行されているすべてのステートメントのテキストが表示されるため、サーバープロセスリストを表示できるすべてのユーザーが、ほかのユーザーによって発行された UPDATE user SET password=PASSWORD('not_secure') などのステートメントを表示できる場合があります。

  mysqld は、SUPER 権限を持つユーザー用に特別な接続を確保しているため、通常の接続がすべて使用中の場合でも、MySQL root ユーザーはログインしてサーバーのアクティビティーを検査することができます。

  SUPER 権限は、クライアント接続を終了したり、システム変数の値を変更することによってサーバー操作を変更したり、レプリケーションサーバーを制御したりするために使用することができます。

- テーブルへのシンボリックリンクを許可しないでください。(この機能は --skip-symbolic-links オプションで無効にできます。)このことは、mysqld を root として実行する場合に特に重要です。これは、サーバーのデータディレクトリへの書き込みアクセス権限があるすべてのユーザーは、システムのすべてのファイルを削除できることになるためです。Unix 上の MyISAM へのシンボリックリンクの使用を参照してください。
- ストアドプログラムおよびビューは、セクション20.6「ストアドプログラムおよびビューのアクセスコントロール」に記載されているセキュリティーガイドラインを使用して記述するようにしてください。
- DNS を信頼していない場合、付与テーブル内でホスト名の代わりに IP アドレスを使用するようにしてください。いずれの場合も、ワイルドカードを含むホスト名の値を使用して付与テーブルエントリを作成することについては、十分に注意するようにしてください。
- 単一アカウントに対して許可される接続数を制限するには、mysqld の max_user_connections 変数を設定することによってこれを実行できます。GRANT ステートメントは、アカウントに対して許可されるサーバー使用の範囲を制限するためのリソース制御オプションもサポートします。セクション13.7.1.4「GRANT 構文」を参照してください。
- プラグインディレクトリがサーバーによって書き込み可能な場合、ユーザーは SELECT ... INTO DUMPFILE を使用して、ディレクトリ内のファイルに実行可能コードを書き込むことができます。これを防ぐために、plugin_dir をサーバーに対して読み取り専用にしたり、SELECT 書き込みが安全に実行できるディレクトリに --secure-file-priv を設定したりできます。

6.1.4 セキュリティー関連の mysqld オプションおよび変数

次の表は、セキュリティーに影響する mysqld オプションおよびシステム変数を示します。これらの個別の説明は、セクション5.1.3「サーバーコマンドオプション」およびセクション5.1.4「サーバーシステム変数」を参照してください。

表 6.1 セキュリティーオプションと変数のサマリー

名前	コマンド行	オプションファイル	システム変数	ステータス変数	変数スコープ	動的
allow-suspicious-udfs	はい	はい				
automatic_sp_privileges			はい		グローバル	はい
chroot	はい	はい				
des-key-file	はい	はい				
local_infile			はい		グローバル	はい
old_passwords			はい		両方	はい
safe-user-create	はい	はい				
secure_auth	はい	はい	はい		グローバル	はい
secure_file_priv	はい	はい	はい		グローバル	いいえ
skip-grant-tables	はい	はい				
skip_name_resolve	はい	はい	はい		グローバル	いいえ
skip_networking	はい	はい	はい		グローバル	いいえ
skip_show_database	はい	はい	はい		グローバル	いいえ

6.1.5 MySQL を通常ユーザーとして実行する方法

Windows 上では、通常のユーザーアカウントを使用して Windows サービスとしてサーバーを実行できます。

Unix では、MySQL サーバー mysqld を起動でき、すべてのユーザーが実行できます。しかし、セキュリティー上の理由から、Unix root ユーザーとしてサーバーを実行することは避けてください。mysqld を変更し、権限のない普通の Unix ユーザー user_name として実行するには、次のことを行う必要があります。

1. サーバーが稼働していれば、サーバーを停止します (mysqladmin shutdown を使用します)。
2. データベースディレクトリとファイルを変更して、user_name がそこのファイルの読み取りおよび書き込みを行う権限を与えます (この操作は Unix root ユーザーとして行う必要があります)。

```
shell> chown -R user_name /path/to/mysql/datadir
```

この操作を行わない場合、サーバーは user_name として実行するとき、データベースまたはテーブルにアクセスできません。

MySQL データディレクトリ内のディレクトリまたはファイルがシンボリックリンクの場合、chown -R がシンボリックリンク先を参照しないことがあります。参照しない場合、それらのリンクを参照し、指定先のディレクトリおよびファイルを変更する必要もあります。

3. user_name というユーザーでサーバーを起動します。代わりに、Unix root ユーザーとして --user=user_name オプションを使用して mysqld を起動する方法もあります。mysqld が起動すると、接続を受け入れる前に Unix ユーザー user_name としての実行に切り替わります。

4. システム起動時に指定されたユーザーとしてサーバーを起動するには、/etc/my.cnf オプションファイルまたはサーバーのデータディレクトリに格納されている my.cnf オプションファイルの [mysqld] グループに、user オプションを追加することによってユーザー名を指定します。例:

```
[mysqld]
user=user_name
```

Unix マシン自体がセキュアな状態でない場合、付与テーブルの MySQL root アカウントにパスワードを割り当てるようにします。これをしないと、そのマシンの口グインアカウントを持つすべてのユーザーが、--user=root オプションを使用して mysql クライアントを実行でき、あらゆる操作を行うことができます。(すべての場合に MySQL アカウントにパスワードを割り当てることはよい考えですが、ほかの口グインアカウントがサーバーホスト上に存在する場合は特に重要です。)セクション2.10.2「最初の MySQL アカウントのセキュリティー設定」を参照してください。

6.1.6 LOAD DATA LOCAL のセキュリティーの問題

LOAD DATA ステートメントは、サーバーホストに配置されているファイルをロードしたり、LOCAL キーワードが指定された場合に、クライアントホストに配置されているファイルをロードしたりできます。

LOAD DATA ステートメントの LOCAL バージョンのサポートに関しては、セキュリティーについての潜在的な問題が 2 つあります。

- クライアントホストからサーバーホストへのファイルの送信は、MySQL サーバーによって開始されます。理論的には、パッチ適用済みサーバーを構築して、LOAD DATA ステートメントでクライアントによって指定されたファイルでなく、サーバーが選択するファイルを転送するようサーバーがクライアントプログラムに指示するようにすることができます。そのようなサーバーは、クライアントユーザーが読み取りアクセス権を持つクライアントホスト上のすべてのファイルにアクセスできます。

- クライアントが Web サーバーから接続する Web 環境で、ユーザーは LOAD DATA LOCAL を使用して、Web サーバープロセスが読み取りアクセス権を持つすべてのファイルを読み取ることができます (ユーザーが SQL Server に対してあらゆるコマンドを実行できる場合)。この環境では、MySQL サーバーを基準にしたときのクライアントは実際には Web サーバーであって、Web サーバーに接続するユーザーによって実行されているリモートプログラムではありません。

これらの問題に対処するために、MySQL 3.23.49 と MySQL 4.0.2 (Windows では 4.0.13) 以降で LOAD DATA LOCAL の処理方法が変更されました。

- デフォルトでは、バイナリ配布内のすべての MySQL クライアントおよびライブラリは -DENABLED_LOCAL_INFILE=1 オプションでコンパイルされ、MySQL 3.23.48 以前との互換性が保持されています。

- MySQL をソースからビルドしたが、-DENABLED_LOCAL_INFILE=1 オプションを指定して CMake を呼び出さない場合、LOAD DATA LOCAL は mysql_options(... MYSQL_OPT_LOCAL_INFILE, 0) を呼び出すように明示的に記述される場合を除いて、いずれのクライアントからも使用できません。セクション23.7.7.49「mysql_options()」を参照してください。

- --local-infile=0 オプションを指定して mysqld を起動することによって、サーバー側からのすべての LOAD DATA LOCAL ステートメントを無効にすることができます。

- mysql コマンド行クライアントの場合、--local-infile[=1] オプションを指定することによって LOAD DATA LOCAL を有効にするか、--local-infile=0 オプションを指定することによってこれを無効にします。mysqlimport の場合、ローカルデータファイルのロードはデフォルトでオフになっており、--local または -L オプションを使用してこれを有効にします。いずれの場合でも、ローカルロード操作を正常に使用するには、サーバーがこの操作を許可していることが必要。

- オプションファイルから [client] グループを読み取る Perl スクリプトまたはその他のプログラムで LOAD DATA LOCAL を使用する場合、local-infile=1 オプションをそのグループに追加できます。ただし、local-infile を認識しないプログラムで問題が発生しないようにするために、loose- プリフィクスを使用してこれを指定します。

```
[client]
loose-local-infile=1
```

- サーバーまたはクライアントのいずれかで LOAD DATA LOCAL が無効な場合、そのようなステートメントを発行しようとしたクライアントは次のエラーメッセージを受け取ります。

```
ERROR 1148: The used command is not allowed with this MySQL version
```

6.1.7 クライアントプログラミングのセキュリティーガイドライン

MySQL にアクセスするアプリケーションは、ユーザーによって入力されるあらゆるデータを信頼しないようにしてください。ユーザーは Web フォーム、URL、または構築されたあらゆるアプリケーションに特殊文字またはエスケープ文字のシーケンスを入力することによってコードを欺くことを試すことができます。ユーザーが 「; DROP DATABASE mysql;」 のような入力を行なっても、アプリケーションがセキュアな状態に保たれるようにしてください。これは極端な例ですが、同様の技術を使用するハッカーに備えていない場合、結果として大規模なセキュリティーリークおよびデータ損失が発生することがあります。

よくある過ちは、文字列データ値のみ保護することです。数値データも忘れずに検査してください。ユーザーが 234 という値を入力したとき、アプリケーションが SELECT * FROM table WHERE ID=234 のようなクエリーを生成する場合、ユーザーは 234 OR 1=1 という値を入力して、SELECT * FROM table WHERE ID=234 OR 1=1 というクエリーをアプリケーションに生成させることができます。その結果、サーバーはテーブル内のすべての行を取得します。これはすべての行を公開し、サーバーに過剰な負荷がかかります。この種類の攻撃から保護するためのもっとも簡単な方法は、数値定数を囲む単一引用符を使用して、SELECT * FROM table WHERE ID='234' とする方法です。ユーザーが余分の情報を入力すると、その情報はすべて文字列の一部となります。数値コンテキストでは、MySQL は自動的にこの文字列を数値に変換し、あとに続く数値以外のすべての文字を取り除きます。

データベースに格納されているデータが公開されているもののみであれば、保護は不要だと思われることもあります。これは正しくありません。データベース内のどの行も表示が許可されている場合であっても、サービス妨害攻撃 (前のパラグラフの技術に基づいた、サーバーにリソースを浪費させるものなど) から保護するべきです。そうしない場合、サーバーは正当なユーザーに対して応答不能になります。

チェックリスト:

- 厳密な SQL モードを有効にして、サーバーが受け入れるデータ値の制限を厳しくするようサーバーに指示します。セクション5.1.7「サーバー SQL モード」を参照してください。
- すべての Web フォームに単一引用符または二重引用符 (「'」 および 「"」) を入力してみます。何らかの種類の MySQL エラーが出る場合、すぐに問題を調査してください。
- 動的 URL に %22 (「"」)、%23 (「#」)、および %27 (「'」) を追加して、これらを変更してみます。
- 前の例で示した文字を使用して、動的 URL のデータ型を数値型から文字型に変更してみます。これらの攻撃や類似の攻撃に対してアプリケーションが安全であるようにしてください。
- 数値フィールドに、数値でなく文字、スペース、および特殊記号を入力してみます。アプリケーションはこれらを MySQL に渡す前にこれらを削除するか、またはエラーを生成します。検査済みでない値を MySQL に渡すことは非常に危険です。
- データを MySQL に渡す前にデータのサイズを検査してください。
- アプリケーションがデータベースに接続するときは、管理目的で使用するものとは異なるユーザー名を使用するようにしてください。アプリケーションに不要なアクセス権限を付与しないでください。

多くのアプリケーションプログラミングインタフェースには、データ値の特殊文字をエスケープする手段が備わっています。適切に使用すれば、これにより、意図とは異なる効果を持つステートメントをアプリケーションに生成させる値を、アプリケーションユーザーが入力できないようにすることができます。

- MySQL C API: mysql_real_escape_string() API コールを使用してください。
- MySQL++: クエリーストリームに対して escape および quote 修飾子を使用してください。
- PHP: mysqli または pdo_mysql 拡張子を使用し、古い ext/mysql 拡張子を使用しないでください。推奨される API は、プレースホルダを持つプリペアドステートメントのほかに、改善された MySQL 認証プロトコルおよびパスワードをサポートします。Choosing an APIも参照してください。

古い ext/mysql 拡張子を使用する必要がある場合、エスケープのために、mysql_real_escape_string() 関数を使用し、mysql_escape_string() または addslashes() を使用しないでください。これは、mysql_real_escape_string() のみが文字セットを認識し、ほかの関数は、(無効な) マルチバイト文字セットを使用したときに「バイパスされる」可能性があるためです。

- Perl DBI: プレースホルダまたは quote() メソッドを使用してください。
- Ruby DBI: プレースホルダまたは quote() メソッドを使用してください。
- Java JDBC: PreparedStatement オブジェクトおよびプレースホルダを使用してください。

ほかのプログラミングインタフェースも似たような機能を持っている場合があります。

6.2 MySQL アクセス権限システム

MySQL 権限システムの主な役割は、特定のホストから接続するユーザーを認証すること、およびそのユーザーを、SELECT、INSERT、UPDATE、DELETE などのデータベースにおける権限に関連付けることです。追加機能として、匿名ユーザーを持つこと、そして、管理操作や LOAD DATA INFILE などの MySQL 特有の機能についての権限を与えることなどがあります。

MySQL 権限システムでは実行できないこともあります。

- 特定ユーザーのアクセスを拒否するように明示的に指定することはできません。つまり、ユーザーを明示的に突き合わせて接続を拒否することはできません。
- ユーザーがデータベースのテーブルを作成または削除できるが、データベース自体の作成または削除はできないような権限をユーザーが持つように指定することはできません。
- パスワードはアカウントに対してグローバルに適用されます。データベース、テーブル、ルーチンなどの特定のオブジェクトにパスワードを関連付けることはできません。

MySQL 権限システムへのユーザーインタフェースは、CREATE USER、GRANT、REVOKE などの SQL ステートメントで構成されます。セクション13.7.1「アカウント管理ステートメント」を参照してください。

内部的には、サーバーは権限情報を mysql データベース (つまり mysql という名前のデータベース) の付与テーブルに格納します。MySQL サーバーはこれらのテーブルの内容を起動時にメモリーに読み取り、付与テーブルのインメモリーコピーに基づいてアクセス制御を決定します。

MySQL 権限システムによって、すべてのユーザーは自分に許可された操作のみ実行可能になります。ユーザーとして MySQL サーバーに接続すると、ユーザーの ID は、接続元のホストおよび指定したユーザー名によって決定されます。接続後にリクエストを発行すると、システムは、ユーザー ID と ユーザーが行う操作に応じて権限を付与します。

MySQL ではホスト名とユーザー名の両方を考慮に入れてユーザーを特定しますが、これは、特定のユーザー名がすべてのホストで同一人物に属すると想定する根拠がないためです。たとえば、office.example.com から接続したユーザー joe は、home.example.com から接続した joe と同一人物とは限りません。MySQL では、たまたま同じ名前を持った異なるホスト上のユーザーを識別できるようにすることによってこれを処理します。つまり、office.example.com からの joe による接続に対して 1 つの権限セットを付与し、home.example.com からの joe による接続に対して別の権限セットを提供することができます。特定のアカウントが持つ権限を表示するには、SHOW GRANTS ステートメントを使用します。例:

```
SHOW GRANTS FOR 'joe'@'office.example.com';
SHOW GRANTS FOR 'joe'@'home.example.com';
```

サーバーに接続するクライアントプログラムを実行するとき、MySQL アクセス制御には 2 つのステージがあります。

ステージ 1: サーバーは、ユーザーの ID および正しいパスワードを指定することによって ID を検証できるかどうかに基づいて、接続を受け入れるか拒否します。

ステージ 2: 接続できる場合、サーバーはユーザーが発行する各ステートメントを検査して、ステートメントを実行するだけの十分な権限をユーザーが持っているかどうかを判別します。たとえば、データベースのテーブルからレコードを選択したり、データベースのテーブルを削除したりしようとすると、サーバーはユーザーにそのテーブルの SELECT 権限があるかどうか、またはデータベースの DROP 権限があるかどうかを検証します。

各ステージで発生する動作の詳細な説明については、セクション6.2.4「アクセス制御、ステージ 1: 接続の検証」およびセクション6.2.5「アクセス制御、ステージ 2: リクエストの確認」を参照してください。

ユーザーの接続中に (ユーザー自身または別のだれかによって) 権限が変更された場合、それらの変更は、ユーザーが発行する次のステートメントで必ずしもすぐに有効になるわけではありません。サーバーが付与テーブルをリロードする条件についての詳細は、セクション6.2.6「権限変更が有効化される時期」を参照してください。

セキュリティーに関連する一般的な助言については、セクション6.1「一般的なセキュリティーの問題」を参照してください。権限に関連した問題の診断についての支援情報は、セクション6.2.7「アクセス拒否エラーの原因」を参照してください。

6.2.1 MySQL で提供される権限

MySQL では、さまざまなコンテキストおよびさまざまな動作レベルに適用される権限が提供されます。

- 管理権限によって、ユーザーは MySQL サーバーの動作を管理できます。これらの権限は特定のデータベースに固有でないため、グローバルです。
- データベース権限は、データベースおよびデータベース内のすべてのオブジェクトに適用されます。これらの権限は、特定のデータベースに付与したり、すべてのデータベースに適用されるようにグローバルに付与したりすることができます。
- テーブル、インデックス、ビュー、ストアドルーチンなどのデータベースオブジェクト向けの権限は、データベース内の特定のオブジェクト、データベース内の特定タイプのすべてのオブジェクト (たとえばデータベース内のすべてのテーブル)、またはすべてのデータベース内の特定タイプのすべてのオブジェクトにグローバルに付与することができます。

アカウント権限に関する情報は、mysql データベース内の user、db、tables_priv、columns_priv、および procs_priv テーブルに格納されます (セクション6.2.2「権限システム付与テーブル」を参照してください)。MySQL サーバーはこれらのテーブルの内容を起動時にメモリーに読み取り、セクション6.2.6「権限変更が有効化される時期」に示す条件でこれらを再ロードします。アクセス制御決定は、付与テーブルのインメモリーコピーに基づきます。

MySQL の一部のリリースでは、新たな権限や機能を追加するために付与テーブルの構造に変更を加えているものもあります。すべての新しい機能を確実に活用できるようにするには、新しいバージョンの MySQL に更新するときに常に付与テーブルを更新して、最新の構造を持つようにします。セクション4.4.7「mysql_upgrade — MySQL テーブルのチェックとアップグレード」を参照してください。

次の表には、GRANT および REVOKE ステートメント内で SQL レベルで使用される権限名のほか、付与テーブル内の各権限に関連付けられたカラムおよび権限が適用されるコンテキストが示されています。

表 6.2 GRANT および REVOKE に対して許容可能な権限

権限	カラム	コンテキスト
CREATE	Create_priv	データベース、テーブル、またはインデックス
DROP	Drop_priv	データベース、テーブル、またはビュー
GRANT OPTION	Grant_priv	データベース、テーブル、またはストアドルーチン
LOCK TABLES	Lock_tables_priv	データベース
REFERENCES	References_priv	データベースまたはテーブル
EVENT	Event_priv	データベース
ALTER	Alter_priv	テーブル
DELETE	Delete_priv	テーブル
INDEX	Index_priv	テーブル
INSERT	Insert_priv	テーブルまたはカラム
SELECT	Select_priv	テーブルまたはカラム
UPDATE	Update_priv	テーブルまたはカラム
CREATE TEMPORARY TABLES	Create_tmp_table_priv	テーブル
TRIGGER	Trigger_priv	テーブル
CREATE VIEW	Create_view_priv	ビュー

権限	カラム	コンテキスト
SHOW VIEW	Show_view_priv	ビュー
ALTER ROUTINE	Alter_routine_priv	ストアドルーチン
CREATE ROUTINE	Create_routine_priv	ストアドルーチン
EXECUTE	Execute_priv	ストアドルーチン
FILE	File_priv	サーバーホストでのファイルアクセス
CREATE TABLESPACE	Create_tablespace_priv	サーバー管理
CREATE USER	Create_user_priv	サーバー管理
PROCESS	Process_priv	サーバー管理
PROXY	proxies_priv テーブルを参照	サーバー管理
RELOAD	Reload_priv	サーバー管理
REPLICATION CLIENT	Repl_client_priv	サーバー管理
REPLICATION SLAVE	Repl_slave_priv	サーバー管理
SHOW DATABASES	Show_db_priv	サーバー管理
SHUTDOWN	Shutdown_priv	サーバー管理
SUPER	Super_priv	サーバー管理
ALL [PRIVILEGES]		サーバー管理
USAGE		サーバー管理

次のリストには、MySQL で使用可能な各権限の一般的な説明が提供されています。特定の SQL ステートメントには、ここに示されているよりも具体的な権限要件がある場合もあります。そのような場合、該当するステートメントの説明で詳細を示します。

- ALL または ALL PRIVILEGES 権限指定子は、短縮形です。これは、「特定の権限レベルで使用可能なすべての権限」 (GRANT OPTION を除く) を表します。たとえば、グローバルまたはテーブルレベルで ALL を付与すると、すべてのグローバル権限またはテーブルレベルのすべての権限が付与されます。

- ALTER 権限は、テーブルの構造を変更するための ALTER TABLE の使用を可能にします。ALTER TABLE には CREATE および INSERT 権限も必要です。テーブルを名前変更するには、古いテーブル側で ALTER および DROP と、新しいテーブル側で ALTER、CREATE、および INSERT 権限が必要です。

- ストアドルーチン (プロシージャーおよび関数) を変更または削除するには、ALTER ROUTINE 権限が必要です。

- CREATE 権限は、新規データベースおよびテーブルの作成を可能にします。

- ストアドルーチン (プロシージャーおよび関数) を作成するには、CREATE ROUTINE 権限が必要です。

- テーブルスペースおよびログファイルグループを作成、変更、または削除するには、CREATE TABLESPACE 権限が必要です。

- CREATE TEMPORARY TABLES 権限は、CREATE TEMPORARY TABLE ステートメントを使用した一時テーブルの作成を可能にします。

  MySQL 5.6.3 の時点では、セッションによって一時テーブルが作成されると、サーバーはテーブル上の権限チェックを追加実行しません。セッションの作成によって、DROP TABLE、INSERT、UPDATE、SELECT などのあらゆる操作をテーブル上で実行できます。

  この動作の 1 つの影響として、現在のユーザーが一時テーブルを作成する権限を持たなくても、セッションが一時テーブルを操作できるということがあります。現在のユーザーが CREATE TEMPORARY TABLES 権限を持たないが、CREATE TEMPORARY TABLES を持つユーザーの権限で実行して一時テーブルを作成する、DEFINER コンテキストのストアドプロシージャーを実行できるとします。プロシージャーの実行中、セッションは定義側ユーザーの権限を使用します。プロシージャーが復帰したあと、有効な権限は現在のユーザーの権限に戻り、これによって引き続き一時テーブルを表示し、一時テーブルに対してあらゆる操作を実行できることになります。

  MySQL 5.6.3 より前では、INSERT、UPDATE、SELECT などの一時テーブル上でのほかの操作には、一時テーブルを格納するデータベースに対する操作について、または同じ名前の非一時テーブルについての追加の権限が必要です。

一時テーブルと非一時テーブルの権限を分離するための、この状況についての一般的な回避策は、一時テーブルを使用するための専用データベースを作成することです。そうすることで、そのデータベースについて、CREATE TEMPORARY TABLES 権限と、ユーザーによって実行される一時テーブル操作に必要なほかのすべての権限を、そのユーザーに付与することができます。

- CREATE USER 権限は、CREATE USER、DROP USER、RENAME USER、および REVOKE ALL PRIVILEGES の使用を可能にします。
- CREATE VIEW 権限は、CREATE VIEW の使用を可能にします。
- DELETE 権限は、データベースのテーブルからの行の削除を可能にします。
- DROP 権限は、既存のデータベース、テーブル、およびビューのドロップを可能にします。パーティション化されたテーブルで ALTER TABLE ... DROP PARTITION ステートメントを使用するには DROP 権限が必要です。DROP 権限は TRUNCATE TABLE のためにも必要です。mysql データベースに対する DROP 権限をユーザーに付与すると、MySQL アクセス権限が格納されているデータベースをユーザーが削除することができます。
- イベントスケジューラについてのイベントを作成、変更、削除、または表示するには、EVENT 権限が必要です。
- ストアドルーチン (プロシージャーおよび関数) を実行するには、EXECUTE 権限が必要です。
- FILE 権限は、LOAD DATA INFILE および SELECT ... INTO OUTFILE ステートメントと LOAD_FILE() 関数を使用するサーバーホスト上でファイルの読み取りおよび書き込みを実行するための許可をユーザーに与えます。FILE 権限を持つユーザーは、すべてのユーザーから読み取り可能であるか、MySQL サーバーによって読み取り可能なサーバーホスト上のすべてのファイルを読み取ることができます。(このことは暗黙的に、データベースディレクトリ内のあらゆるファイルにサーバーからアクセスできるため、ユーザーはそれらのすべてのファイルを読み取ることができることを意味します。)さらに FILE 権限によって、ユーザーは MySQL サーバーが書き込みアクセス権限を持つあらゆるディレクトリに新しいファイルを作成できます。これは、権限テーブルを実装するファイルを格納するサーバーのデータディレクトリを含みます。セキュリティー対策として、サーバーは既存のファイルを上書きしません。

  ファイルを読み取りおよび書き込みできる場所を制限するには、secure_file_priv システムを特定のディレクトリに設定します。セクション5.1.4「サーバーシステム変数」を参照してください。
- GRANT OPTION 権限によって、ユーザーは自分自身が所有する権限をほかのユーザーに付与したり、ほかのユーザーから削除したりすることができます。
- INDEX 権限によって、ユーザーはインデックスを作成またはドロップできます。INDEX は既存のテーブルに適用されます。テーブルに対する CREATE 権限を持つ場合、CREATE TABLE ステートメントにインデックス定義を含めることも可能です。
- INSERT 権限は、データベースのテーブルへの行の挿入を可能にします。ANALYZE TABLE、OPTIMIZE TABLE、REPAIR TABLE などのテーブル保守に関するステートメントにも、INSERT 権限が必要です。
- LOCK TABLES 権限は、ユーザーが SELECT 権限を持つテーブルをロックするために、明示的な LOCK TABLES ステートメントの使用を可能にします。これには、ロックされたテーブルをほかのセッションから読み取らせないようにする書き込みロックの使用が含まれます。
- PROCESS 権限は、サーバー内で実行するスレッドについての情報の表示に関係します (つまり、セッションによって実行されるステートメントについての情報)。この権限は、SHOW PROCESSLIST または mysqladmin processlist を使用して、ほかのアカウントに属するスレッドを表示することを可能にし、自分自身のスレッドを表示することもできます。PROCESS 権限は、SHOW ENGINE の使用も可能にします。
- PROXY 権限によって、ユーザーは別のユーザーになりすましたり、別のユーザーとして認識されるようにしたりすることができます。セクション6.3.9「プロキシユーザー」を参照してください。
- REFERENCES 権限は現在使用されていません。
- RELOAD 権限は、FLUSH ステートメントの使用を可能にします。また、これは FLUSH 操作 (flush-hosts、flush-logs、flush-privileges、flush-status、flush-tables、flush-threads、refresh、および reload) と同等な mysqladmin コマンドを使用可能にします。

  reload コマンドは、付与テーブルをメモリーにリロードするようにサーバーに指示します。flush-privileges は reload のシノニムです。refresh コマンドは、ログファイルを閉じて再オープンし、すべてのテーブルをフラッシュします。ほかの flush-xxx コマンドは refresh に類似した機能を実行しますが、より具体的であるため一部の状況では好ましい場合があります。たとえば、ログファイルだけをフラッシュするときは、refresh よりも flush-logs を選択することをお勧めします。

- REPLICATION CLIENT 権限は、SHOW MASTER STATUS および SHOW SLAVE STATUS の使用を可能にします。MySQL 5.6.6 以降では、これは SHOW BINARY LOGS ステートメントの使用も可能にします。
- REPLICATION SLAVE 権限は、マスターとしての現在のサーバーに接続するときにスレーブサーバーが使用するアカウントに対して与えるようにしてください。この権限がない場合、スレーブは、マスターサーバー上のデータベースに対して実行された更新をリクエストすることができません。
- SELECT 権限によって、ユーザーはデータベース内のテーブルから行を選択できます。SELECT ステートメントで SELECT 権限が必要となるのは、ステートメントがテーブルから実際にレコードを取得する場合です。一部の SELECT ステートメントはテーブルにアクセスしないため、あらゆるデータベースへのアクセス権がなくても実行できます。たとえば、テーブルを参照しない式を評価するための単純な計算機として SELECT を使用することができます。

```
SELECT 1+1;
SELECT PI()*2;
```

  SELECT 権限は、カラム値を読み取るほかのステートメントについても必要です。たとえば、UPDATE ステートメントの代入 col_name=expr の右辺で参照されるカラムや、DELETE または UPDATE ステートメントの WHERE 句で指定されるカラムについて、SELECT が必要です。
- SHOW DATABASES 権限によって、アカウントは SHOW DATABASE ステートメントを発行することでデータベース名を表示することができます。この権限を持たないアカウントには、アカウントが一部の権限を持つデータベースしか表示されず、サーバーが --skip-show-database オプションで起動されている場合はステートメントを一切使用できません。すべてのグローバル権限は、データベースに対する権限だということに注意してください。
- SHOW VIEW 権限は、SHOW CREATE VIEW の使用を可能にします。
- SHUTDOWN 権限は、mysqladmin shutdown コマンドの使用を可能にします。対応する SQL ステートメントはありません。
- SUPER 権限によって、アカウントはほかのアカウントに属するスレッドを強制終了するための CHANGE MASTER TO、KILL、または mysqladmin kill (自分のスレッドは常に強制終了できます)、PURGE BINARY LOGS、グローバルシステム変数を変更するための SET GLOBAL を使用した構成変更、mysqladmin debug コマンド、ロギングの有効化または無効化、read_only システム変数が有効な場合の更新の実行、スレーブサーバー上でのレプリケーションの開始と停止、ストアドプログラムおよびビューの DEFINER 属性内のすべてのアカウントの指定を使用することができ、ユーザーは max_connections システム変数によって制御される接続制限に達している場合でも (一度) 接続することができます。

  バイナリロギングを有効にした場合にストアドファンクションを作成または変更するとき、セクション 20.7「ストアドプログラムのバイナリロギング」に記載されているように SUPER 権限がやはり必要になることがあります。
- TRIGGER 権限はトリガー操作を有効にします。テーブルのトリガーを作成、削除、または実行するには、そのテーブルに対してこの権限を持つ必要があります。
- UPDATE 権限は、データベース内のテーブルの行の更新を可能にします。
- USAGE 権限指定子は、「権限なし」を表します。これは GRANT と一緒にグローバルレベルで使用されて、既存のアカウント権限に影響を及ぼすことなくリソース制限や SSL 特性などのアカウント属性を変更します。

アカウントには必要な権限のみを付与することをお勧めします。FILE 権限と管理権限の付与については十分に注意するようにしてください。

- FILE 権限を悪用して、MySQL サーバーがサーバーホスト上で読み取ることができるあらゆるファイルをデータベーステーブルから読み取ることができます。これにはすべてのユーザーが読み取り可能なすべてのファイルと、サーバーのデータディレクトリ内のファイルが含まれます。そのあと、SELECT を使用してそのテーブルにアクセスし、テーブルの内容をクライアントホストに送信することができます。
- GRANT OPTION 権限によって、ユーザーはほかのユーザーに権限を付与することができます。異なる権限を持つ 2 人のユーザーが GRANT OPTION 権限を持っていれば、権限を組み合わせることができます。
- ALTER 権限を使用して、テーブルの名前を変更することによって権限システムを壊すことができます。
- SHUTDOWN 権限を悪用して、サーバーを終了することによって、ほかのユーザーへのサービスを完全に妨害することができます。
- PROCESS 権限は、パスワードの設定や変更を行うステートメントなどを含め、現在実行中のステートメントのプレーンテキストを表示することができます。

- SUPER 権限は、ほかのセッションを終了したり、サーバーの動作方法を変更したりするために使用できます。
- mysql データベース自体に付与した権限を使用して、パスワードおよびその他のアクセス権限情報を変更することができます。パスワードは暗号化されて保管されているため、悪意のあるユーザーは単純にそのパスワードを見てプレーンテキストパスワードを知ることはできません。ただし、user テーブルの Password カラムへの書き込みアクセス権限を持つユーザーは、アカウントのパスワードを変更し、そのアカウントを使用して MySQL サーバーに接続することができます。

6.2.2 権限システム付与テーブル

通常では、アカウントをセットアップして各アカウントで使用可能な権限を制御するためには、GRANT や REVOKE などのステートメントを使用することによって、mysql データベース内の付与テーブルの内容を間接的に操作します。セクション13.7.1「アカウント管理ステートメント」を参照してください。ここでは付与テーブルの基本構造と、サーバーがクライアントと対話するときに付与テーブルの内容をどのように使用するかについて説明します。

次の mysql データベーステーブルには付与情報が格納されています。

- user: ユーザーアカウント、グローバル権限、および権限以外のその他のカラムが含まれています。
- db: データベースレベルの権限が格納されています。
- host: 使用されなくなりました。MySQL 5.6.7 以降の新しい MySQL インストールではこのテーブルが作成されなくなりました。
- tables_priv: テーブルレベルの権限が格納されています。
- columns_priv: カラムレベルの権限が格納されています。
- procs_priv: ストアドプロシージャーおよびストアドファンクションの権限が格納されています。
- proxies_priv: プロキシユーザー権限が格納されています。

mysql データベース内のほかのテーブルには付与情報が保持されておらず、ほかの場所で説明されています。

- event: イベントスケジューラについての情報が格納されています。セクション20.4「イベントスケジューラの使用」を参照してください。
- func: ユーザー定義関数についての情報が格納されています。セクション24.3「MySQL への新しい関数の追加」を参照してください。
- help_xxx: これらのテーブルはサーバー側のヘルプに使用されます。セクション5.1.10「サーバー側のヘルプ」を参照してください。
- plugin: サーバープラグインについての情報が格納されています。セクション5.1.8.1「プラグインのインストールおよびアンインストール」およびセクション24.2「MySQL プラグイン API」を参照してください。
- proc: ストアドプロシージャーおよびストアドファンクションについての情報が格納されています。セクション20.2「ストアドルーチン (プロシージャーと関数) の使用」を参照してください。
- servers: FEDERATED ストレージエンジンによって使用されます。セクション15.8.2.2「CREATE SERVER を使用した FEDERATED テーブルの作成」を参照してください。
- time_zone_xxx: これらのテーブルにはタイムゾーン情報が格納されています。セクション10.6「MySQL Server でのタイムゾーンのサポート」を参照してください。
- 名前に _log が付いているテーブルは、ロギングに使用されます。セクション5.2「MySQL Server ログ」を参照してください。

> **注記**
>
> mysql データベース内のテーブルの変更は、CREATE USER、GRANT、CREATE PROCEDURE などのステートメントの応答としてサーバーによって通常実行されます。INSERT、UPDATE、DELETE などのステートメントを使用してこれらのテーブルを直接変更することは推奨されません。これらの変更の結果として誤った形式となった行を、サーバーは随意で無視します。

各付与テーブルにはスコープカラムと権限カラムがあります。

- スコープカラムはテーブル内の各行 (エントリ) のスコープ、つまり行が適用されるコンテキストを決定します。たとえば、Host と User の値が 'thomas.loc.gov' と 'bob' である user テーブル行は、bob というユーザー名

を指定するクライアントによるホスト thomas.loc.gov からサーバーに対して実行する認証接続のために使用されます。同様に、Host、User、および Db カラム値が 'thomas.loc.gov'、'bob'、および 'reports' である db テーブル行は、bob が thomas.loc.gov ホストから reports データベースに接続するときに使用されます。tables_priv テーブルおよび columns_priv テーブルには、それぞれの行に適用されるテーブルまたはテーブルとカラムの組み合わせを示すスコープカラムがあります。procs_priv スコープカラムは、それぞれの行に適用されるストアドルーチンを示します。

- 権限カラムは、テーブル行によって付与される権限、つまり実行可能な操作を指定します。サーバーはさまざまな付与テーブル内の情報を組み合わせてユーザーの権限の完全な記述を構成します。この実行に使用されるルールの説明は、セクション6.2.5「アクセス制御、ステージ 2: リクエストの確認」にあります。

サーバーは次の方法で付与テーブルを使用します。

- user テーブルのスコープカラムは、入接続を拒否または許可するかを決定します。許可される接続について、user テーブル内で付与されるすべての権限は、ユーザーのグローバル権限を示します。このテーブル内で付与されるすべての権限は、サーバー上のすべてのデータベースに適用されます。

  > **注記**
  >
  > あらゆるグローバル権限は、すべてのデータベースに対する権限とみなされるため、あらゆるグローバル権限を持つユーザーは、SHOW DATABASES を使用したり、または INFORMATION_SCHEMA の SCHEMATA テーブルを調べたりすることで、すべてのデータベース名を表示できるようになります。

- db テーブルのスコープカラムは、どのユーザーがどのホストからどのデータベースにアクセスできるかを決定します。権限カラムは、許可される操作を決定します。データベースレベルで付与される権限は、データベースのほかテーブルやストアドプログラムなどのデータベース内のすべてのオブジェクトに適用されます。

- tables_priv および columns_priv テーブルは db テーブルと似ていますが、これらはさらに粒度が細かく、データベースレベルでなくテーブルレベルおよびカラムレベルに適用されます。テーブルレベルで付与される権限は、テーブルおよびそのすべてのカラムに適用されます。カラムレベルで付与される権限は、特定のカラムにのみ適用されます。

- procs_priv テーブルはストアドルーチンに適用されます。ルーチンレベルで付与される権限は、単一ルーチンにのみ適用されます。

- proxies_priv テーブルは、ほかのユーザーのプロキシの役割を担うことができるユーザーがだれか、およびプロキシユーザーが PROXY 権限をほかのユーザーに付与できるかどうかを指定します。

サーバーは mysql データベース内の user および db テーブルを、アクセス制御のステージ 1 とステージ 2 の両方で使用します (セクション6.2「MySQL アクセス権限システム」を参照してください)。user と db のテーブルのカラムをここで示します。

表 6.3 user テーブルおよび db テーブルのカラム

テーブル名	user	db
スコープカラム	Host	Host
	User	Db
	Password	User
権限カラム	Select_priv	Select_priv
	Insert_priv	Insert_priv
	Update_priv	Update_priv
	Delete_priv	Delete_priv
	Index_priv	Index_priv
	Alter_priv	Alter_priv
	Create_priv	Create_priv
	Drop_priv	Drop_priv
	Grant_priv	Grant_priv
	Create_view_priv	Create_view_priv
	Show_view_priv	Show_view_priv
	Create_routine_priv	Create_routine_priv

テーブル名	user	db
	Alter_routine_priv	Alter_routine_priv
	Execute_priv	Execute_priv
	Trigger_priv	Trigger_priv
	Event_priv	Event_priv
	Create_tmp_table_priv	Create_tmp_table_priv
	Lock_tables_priv	Lock_tables_priv
	References_priv	References_priv
	Reload_priv	
	Shutdown_priv	
	Process_priv	
	File_priv	
	Show_db_priv	
	Super_priv	
	Repl_slave_priv	
	Repl_client_priv	
	Create_user_priv	
	Create_tablespace_priv	
セキュリティーカラム	ssl_type	
	ssl_cipher	
	x509_issuer	
	x509_subject	
	plugin	
	authentication_string	
	password_expired	
リソース制御カラム	max_questions	
	max_updates	
	max_connections	
	max_user_connections	

mysql.user テーブルの plugin および authentication_string カラムは認証プラグイン情報を格納します。

アカウント行の plugin カラムが空の場合、サーバーは Password カラムのパスワードハッシュの形式に応じて、mysql_native_password または mysql_old_password プラグインを暗黙的に使用してアカウントを認証します。Password 値が空または 4.1 のパスワードハッシュ (41 文字) である場合、サーバーは mysql_native_password を使用します。パスワード値が 4.1 より前のパスワードハッシュ (16 文字) の場合、サーバーは mysql_old_password を使用します。(これらのハッシュ形式についての追加情報は、セクション 6.1.2.4「MySQL でのパスワードハッシュ」を参照してください。)クライアントは、アカウント行の Password カラム内のパスワードと一致する必要があります。

アカウント行で plugin カラムにプラグインが指定された場合、サーバーはこれを使用して、アカウントに対する接続の試行を認証します。プラグインが Password カラム内の値を使用するかどうかはプラグインによって異なります。

password_expired カラムは、DBA がアカウントパスワードを期限切れにして、ユーザーにパスワードをリセットするよう求めることができるようにするために、MySQL 5.6.6 で追加されました。デフォルトの password_expired 値は 'N' ですが、ALTER USER ステートメントを使用して 'Y' に設定できます。アカウントのパスワードの有効期限が切れたあと、サーバーへの以降の接続で、そのアカウントによって実行されるすべての操作は、ユーザーが SET PASSWORD ステートメントを発行して新しいアカウントパスワードを確立するまで、エラーになります。セクション13.7.1.1「ALTER USER 構文」を参照してください。

パスワードの有効期限が切れたあと、SET PASSWORD を使用してパスワードを現在の値に設定することによって、パスワードを「リセット」することができます。適切なポリシーとして、別のパスワードを選択することをお勧めします。

> **注意**
>
> MySQL 5.6.6 で、ALTER USER は Password カラムを空の文字列に設定するため、このステートメントは 5.6.7 まで使用しないでください。

アクセス制御のステージ 2 で、サーバーはリクエスト検証を実行して、クライアントが発行した各リクエストに対してそれぞれのクライアントが十分な権限を持つことを確認します。user および db 付与テーブルに加えて、テーブルに関するリクエストの場合、サーバーは tables_priv および columns_priv テーブルを参照することもあります。後者のテーブルは、テーブルレベルおよびカラムレベルでの細かい権限制御を提供します。これらには、次の表に示すカラムがあります。

表 6.4 tables_priv テーブルおよび columns_priv テーブルのカラム

テーブル名	tables_priv	columns_priv
スコープカラム	Host	Host
	Db	Db
	User	User
	Table_name	Table_name
		Column_name
権限カラム	Table_priv	Column_priv
	Column_priv	
その他のカラム	Timestamp	Timestamp
	Grantor	

Timestamp カラムおよび Grantor カラムは、それぞれ現在のタイムスタンプおよび CURRENT_USER 値に設定されます。ただし、これらは未使用のため、ここではこれ以上説明しません。

ストアドルーチンに関するリクエストを検証するために、サーバーは procs_priv テーブルを参照することがあり、このテーブルには次の表に示すカラムがあります。

表 6.5 procs_priv テーブルのカラム

テーブル名	procs_priv
スコープカラム	Host
	Db
	User
	Routine_name
	Routine_type
権限カラム	Proc_priv
その他のカラム	Timestamp
	Grantor

Routine_type カラムは、'FUNCTION' または 'PROCEDURE' の値を持つ ENUM カラムであり、その行が示すルーチンのタイプを指します。このカラムにより、同じ名前を持つ関数とプロシージャーに別々に権限を付与することができます。

Timestamp カラムと Grantor カラムは現在未使用のため、ここではこれ以上説明しません。

proxies_priv テーブルはプロキシユーザーについての情報を記録します。これには次のカラムがあります。

- Host、User: これらのカラムは、プロキシ設定されるアカウントに対して PROXY 権限を持つユーザーアカウントを示します。
- Proxied_host、Proxied_user: これらのカラムは、プロキシ設定されるユーザーのアカウントを示します。
- Grantor: 現在未使用です。
- Timestamp: 現在未使用です。
- With_grant: このカラムは、プロキシアカウントが PROXY 権限を別のアカウントに付与できるかどうかを示します。

付与テーブルのスコープカラムには文字列が格納されています。これらは次に示すように宣言され、それぞれのデフォルト値は空の文字列です。

表 6.6 付与テーブルのスコープカラムの型

カラム名	型
Host、Proxied_host	CHAR(60)
User、Proxied_user	CHAR(16)
Password	CHAR(41)
Db	CHAR(64)
Table_name	CHAR(64)
Column_name	CHAR(64)
Routine_name	CHAR(64)

アクセスチェックのために、User、Proxied_user、Password、Db、および Table_name 値の比較は大文字小文字を区別します。Host、Proxied_host、Column_name、および Routine_name 値の比較は大文字小文字を区別しません。

user および db テーブルでは、各権限は、ENUM('N','Y') DEFAULT 'N' として宣言される別個のカラムにリストされます。つまり、各権限は無効または有効にすることができ、デフォルトは無効です。

tables_priv、columns_priv、および procs_priv のテーブルでは、権限カラムは SET カラムとして宣言されます。これらのカラムの値は、テーブルによって制御されるあらゆる組み合わせの権限を含むことができます。カラム値にリストされている権限のみが入力されます。

表 6.7 Set タイプ権限のカラム値

テーブル名	カラム名	可能な Set 要素
tables_priv	Table_priv	'Select', 'Insert', 'Update', 'Delete', 'Create', 'Drop', 'Grant', 'References', 'Index', 'Alter', 'Create View', 'Show view', 'Trigger'
tables_priv	Column_priv	'Select', 'Insert', 'Update', 'References'
columns_priv	Column_priv	'Select', 'Insert', 'Update', 'References'
procs_priv	Proc_priv	'Execute', 'Alter Routine', 'Grant'

管理者権限 (RELOAD、SHUTDOWN など) は、user テーブルのみで指定されます。管理操作はサーバー自体での操作であって、データベース固有でないため、これらの権限をほかの付与テーブルにリストする理由はありません。したがって、ユーザーが管理操作を実行できるかどうかを判別するには、サーバーは user テーブルのみ参照する必要があります。

FILE 権限も、user テーブルでのみ指定されます。これはそもそも管理権限ではありませんが、サーバーホスト上のファイルを読み取りまたは書き込みするための能力は、アクセスするデータベースとは無関係です。

mysqld サーバーは、起動時に付与テーブルの内容をメモリーに読み取ります。FLUSH PRIVILEGES ステートメントを発行するか、mysqladmin flush-privileges または mysqladmin reload コマンドを実行することによって、テーブルをリロードするようサーバーに指示することができます。付与テーブルへの変更は、セクション 6.2.6「権限変更が有効化される時期」で示すように反映されます。

アカウントの権限を変更するとき、変更によって権限が希望どおりにセットアップされるかどうかを確認することをお勧めします。特定のアカウントの権限をチェックするには、SHOW GRANTS ステートメントを使用します (セクション13.7.5.22「SHOW GRANTS 構文」を参照してください)。たとえば、ユーザー名およびホスト名の値がそれぞれ bob および pc84.example.com のアカウントに付与された権限を調べるには、次のステートメントを使用します。

```
SHOW GRANTS FOR 'bob'@'pc84.example.com';
```

6.2.3 アカウント名の指定

MySQL アカウント名はユーザー名とホスト名で構成されます。これにより、別々のホストから接続可能な同じ名前を持つユーザーのためのアカウントを作成できます。このセクションでは、特殊な値やワイルドカードルールを含む、アカウント名の記述方法について説明します。

CREATE USER、GRANT、SET PASSWORD などの SQL ステートメントでは、次のルールを使用してアカウント名を記述してください。

- アカウント名の構文は 'user_name'@'host_name' です。
- ユーザー名のみで構成されるアカウント名は、'user_name'@'%' と同等です。たとえば、'me' は 'me'@'%' と同等です。
- ユーザー名およびホスト名は、それらが引用符なしの識別子として有効な場合、引用する必要はありません。引用符が必要なのは、user_name 文字列が特殊文字 (「-」 など) を含んでいたり、host_name 文字列が特殊文字またはワイルドカード文字 (「%」 など) を含んでいたりする場合で、たとえば 'test-user'@'%.com' のようになります。
- ユーザー名およびホスト名は、逆引用符 (「`」)、単一引用符 (「'」)、または二重引用符 (「"」) のいずれかを使用して、識別子または文字列として引用符で囲みます。
- ユーザー名およびホスト名の部分が引用符で囲まれる場合、別々に囲んでください。つまり、'me'@'localhost' と記述し、'me@localhost' とは記述しません。後者は 'me@localhost'@'%' と解釈されます。
- CURRENT_USER または CURRENT_USER() 関数への参照は、現行クライアントのユーザー名およびホスト名の文字を指定することと同等です。

MySQL は、mysql データベースの付与テーブルにアカウント名を格納するとき、ユーザー名とホスト名の部分に別々のカラムを使用します。

- user テーブルにはアカウントごとに 1 行が格納されます。User および Host カラムはユーザー名およびホスト名を格納します。このテーブルには、アカウントが持つグローバル権限も指定されます。
- ほかの付与テーブルには、データベースおよびデータベース内のオブジェクトに対してアカウントが持つ権限が示されます。これらのテーブルにはアカウント名を格納するための User および Host カラムがあります。これらのテーブルの各行は、同じ User および Host の値を持つ user テーブル内のアカウントに関連付けられています。

付与テーブル構造に関する追加の詳細については、セクション6.2.2「権限システム付与テーブル」を参照してください。

ユーザー名およびホスト名は、次に示すような特殊な値が使用されたりワイルドカード規則が適用されたりします。

ユーザー名は、入接続を試行するユーザー名と文字が一致するブランクでない値であるか、あらゆるユーザー名と一致するブランク値 (空の文字列) です。ブランクのユーザー名を持つアカウントは匿名ユーザーです。SQL ステートメントで匿名ユーザーを指定するには、''@'localhost' のように引用符で囲んだ空のユーザー名部分を使用します。

アカウント名のホスト名部分は多くの形式を持つことができ、ワイルドカードが許可されます。

- ホスト値はホスト名または IP アドレス (IPv4 または IPv6) とすることができます。'localhost' という名前はローカルホストを示します。IP アドレス '127.0.0.1' は IPv4 ループバックインタフェースを示します。IP アドレス '::1' は、IPv6 ループバックインタフェースを示します。
- ワイルドカード文字 「%」 および 「_」 を、ホスト名または IP アドレスの値に使用できます。これらは LIKE 演算子で実行されるパターンマッチング演算と同じ意味を持ちます。たとえば、ホスト値 '%' はあらゆるホスト名に一致し、'%.mysql.com' という値は mysql.com ドメイン内のすべてのホストに一致し、'192.168.1.%' は 192.168.1 のクラス C ネットワークのあらゆるホストに一致します。

  IP ワイルドカード値をホスト値に使用できるため (サブネット上のすべてのホストに一致する '192.168.1.%' など)、一部のユーザーがホスト 192.168.1.somewhere.com を指定することによって、この機能を悪用しようとする可能性があります。このような試みを阻止するために、MySQL では、数字およびドットで始まるホスト名との一致が許可されていません。したがって、1.2.example.com のような名前を持つホストがある場合、その名前はアカウント名のホスト部分と決して一致しません。IP ワイルドカード値は、ホスト名でなく IP アドレスのみと一致することができます。
- IPv4 アドレスで指定されるホスト値について、ネットワーク番号に使用するアドレスビットの数を示すネットマスクを指定することができます。ネットマスク表記は IPv6 アドレスについては使用できません。

  構文は host_ip/netmask です。例:

  ```
  CREATE USER 'david'@'192.58.197.0/255.255.255.0';
  ```

  これにより david は、次の条件が true となる IP アドレス client_ip を持つすべてのクライアントホストから接続できます。

```
client_ip & netmask = host_ip
```

つまり、次のような CREATE USER ステートメントがあるとします。

```
client_ip & 255.255.255.0 = 192.58.197.0
```

この条件を満たし、MySQL サーバーに接続できる IP アドレスは、192.58.197.0 から 192.58.197.255 までの範囲のものです。

ネットマスクは、8、16、24、または 32 ビットのアドレスを使用するようサーバーに指示するためにのみ使用できます。例:

- 192.0.0.0/255.0.0.0: 192 のクラス A ネットワーク上のすべてのホスト
- 192.168.0.0/255.255.0.0: 192.168 のクラス B ネットワーク上のすべてのホスト
- 192.168.1.0/255.255.255.0: 192.168.1 のクラス C ネットワーク上のすべてのホスト
- 192.168.1.1: この特定の IP アドレスを持つホストのみ

次のネットマスクは 28 ビットをマスクしますが、28 は 8 の倍数でないため機能しません。

```
192.168.0.1/255.255.255.240
```

サーバーは、クライアントホスト名または IP アドレス用のシステム DNS リゾルバによって返された値を使用して、アカウント名のホスト値の突き合わせをクライアントホストに対して実行します。アカウントホスト値がネットマスク記法を使用して指定される場合を除き、この比較は IP アドレスとして指定されるアカウントホスト値であっても文字列の突き合わせとして実行されます。つまり、DNS によって使用されるのと同じ形式でアカウントホスト値を指定しなければならないということを意味します。留意するべき問題の例を次に示します。

- ローカルネットワーク上のホストの完全修飾名が host1.example.com だとします。DNS がこのホストの名前参照を host1.example.com として返す場合、アカウントホスト値にこの名前を使用します。ただし、DNS が host1 のみを返す場合、代わりに host1 を使用します。
- DNS が特定のホストの IP アドレスとして 192.168.1.2 を返す場合、これはアカウントホスト値 192.168.1.2 と一致しますが 192.168.01.2 とは一致しません。同様に、これは 192.168.1.% のようなアカウントホストパターンに一致しますが 192.168.01.% には一致しません。

このような問題を避けるために、DNS がホスト名およびアドレスを返す際の形式を確認し、同じ形式の値を MySQL アカウント名に使用するようにすることをお勧めします。

6.2.4 アクセス制御、ステージ 1: 接続の検証

ユーザーが MySQL サーバーに接続しようとすると、サーバーはユーザーの ID と、正しいパスワードを指定することでユーザーが自分の ID を検証できるかどうかに基づいて、接続を受け入れるか拒否します。できない場合、サーバーはアクセスを完全に拒否します。それ以外の場合、サーバーは接続を受け入れてステージ 2 に進み、リクエストを待機します。

ユーザーの ID は 2 つの部分の情報に基づきます。

- 接続元のクライアントホスト
- MySQL ユーザー名

ID チェックは、user テーブルの 3 つのスコープカラム (Host、User、Password) を使用して実施されます。サーバーは、一部の user テーブル行の Host および User カラムがクライアントホスト名およびユーザー名と一致し、その行に指定されているパスワードがクライアントから提供された場合のみ、接続を受け入れます。許容可能な Host および User 値についてのルールは、セクション6.2.3「アカウント名の指定」にあります。

User カラム値がブランクでない場合、入接続のユーザー名は正確に一致する必要があります。User 値がブランクの場合、これはすべてのユーザー名と一致します。入接続と一致する user テーブル行のユーザー名がブランクである場合、ユーザーはクライアントが実際に指定した名前を持つユーザーでなく、名前のない匿名ユーザーとみなされます。つまり、接続期間中の (つまりステージ 2 での) 今後のすべてのアクセスチェックでブランクのユーザー名が使用されることを意味します。

Password カラムはブランクにできます。これはワイルドカードではなく、あらゆるパスワードが一致するという意味ではありません。これは、ユーザーはパスワードを指定せずに接続しなければならないことを意味します。サーバーがプラグインを使用してクライアントを認証する場合、プラグインが実装する認証方式で、Password カ

ラムのパスワードが使用される場合もそうでない場合もあります。この場合、MySQL サーバーへの認証を行う際に、外部パスワードも使用される可能性があります。

user テーブル内のブランクでない Password 値は暗号化パスワードを表します。MySQL は、すべてのユーザーが表示できるプレーンテキスト形式のパスワードを格納しません。代わりに、接続しようとしたユーザーが入力したパスワードは (PASSWORD() 関数を使用して) 暗号化されます。そのあと、暗号化パスワードは、接続プロセス中にパスワードが正しいかどうかをチェックするときに使用されます。これは、暗号化パスワードが接続を介してやりとりされずに実行されます。セクション6.3.1「ユーザー名とパスワード」を参照してください。

MySQL から見ると、暗号化パスワードが実際のパスワードであるため、暗号化パスワードへのアクセス権限をすべてのユーザーに付与しないようにしてください。特に、管理者以外のユーザーに、mysql データベース内のテーブルへの読み取りアクセス権限を付与しないでください。

次の表では、user テーブルエントリの Host 値および User 値のさまざまな組み合わせが入接続にどのように適用されるかを示しています。

Host 値	User 値	許容される接続
'thomas.loc.gov'	'fred'	thomas.loc.gov から接続する fred
'thomas.loc.gov'	''	thomas.loc.gov から接続するすべてのユーザー
'%'	'fred'	任意のホストから接続する fred
'%'	''	任意のホストから接続する任意のユーザー
'%.loc.gov'	'fred'	loc.gov ドメイン内の任意のホストから接続する fred
'x.y.%'	'fred'	x.y.net、x.y.com、x.y.edu などから接続する fred。おそらく有用ではありません
'144.155.166.177'	'fred'	IP アドレス 144.155.166.177 のホストから接続する fred
'144.155.166.%'	'fred'	144.155.166 のクラス C サブネットの任意のホストから接続する fred
'144.155.166.0/255.255.255.0'	'fred'	前の例と同じ

入接続のクライアントホスト名およびユーザー名が user テーブルの複数行と一致することもあります。このことは前の一連の例で示されており、示されている複数のエントリが、fred による thomas.loc.gov からの接続に一致します。

複数の一致が可能な場合、サーバーはいずれを使用するかを決定する必要があります。この問題は、次のように解決されます。

- サーバーが user テーブルをメモリーに読み取るとき、行を毎回ソートします。
- クライアントが接続しようとすると、サーバーは行をソート順に参照します。
- サーバーは、クライアントホスト名およびユーザー名が一致した最初の行を使用します。

サーバーは、具体性がもっとも高い Host 値が先になるように行を並べるソートルールを使用します。リテラルのホスト名および IP アドレスは具体性がもっとも高くなります。(リテラルの IP アドレスの具体性は IP アドレスがネットマスクを持つかどうかによって影響されないため、192.168.1.13 と 192.168.1.0/255.255.255.0 の具体性は同等とみなされます。)パターン '%' は「任意のホスト」を意味するため、具体性はもっとも低くなります。空の文字列 '' も「任意のホスト」を意味しますが、'%' のあとにソートされます。同じ Host 値を持つ行は、もっとも具体的な User 値が最初になるよう並べられます (ブランクの User 値は 「任意のユーザー」を意味し、具体性がもっとも低くなります)。Host および User 値の具体性が等しい行については、順序は不確定です。

user テーブルが次の内容であると仮定して、これがどのように作用するかを説明します。

```
+-----------+----------+-
| Host      | User     | ...
+-----------+----------+-
| %         | root     | ...
| %         | jeffrey  | ...
| localhost | root     | ...
| localhost |          | ...
+-----------+----------+-
```

サーバーがテーブルをメモリーに読み取るとき、サーバーは前に記載したルールを使用して行をソートします。ソート後の結果は次のようになります。

```
+-----------+----------+-
```

```
| Host      | User     | ...
+-----------+----------+-
| localhost | root     | ...
| localhost |          | ...
| %         | jeffrey  | ...
| %         | root     | ...
+-----------+----------+-
```

クライアントが接続しようとすると、サーバーはソート済みの行を参照し、見つかった最初の一致を使用します。jeffrey による localhost からの接続の場合、テーブルの 2 つの行が一致し、すなわち Host および User 値が 'localhost' および '' であるものと、値が '%' および 'jeffrey' であるものが一致します。ソート順では 'localhost' 行が最初になるため、サーバーはこの行を使用します。

次に別の例を示します。user テーブルが次のようになっていると仮定します。

```
+---------------+----------+-
| Host          | User     | ...
+---------------+----------+-
| %             | jeffrey  | ...
| thomas.loc.gov |          | ...
+---------------+----------+-
```

ソート済みテーブルは次のようになります。

```
+---------------+----------+-
| Host          | User     | ...
+---------------+----------+-
| thomas.loc.gov |          | ...
| %             | jeffrey  | ...
+---------------+----------+-
```

jeffrey による thomas.loc.gov からの接続は最初の行に一致しますが、jeffrey による任意のホストからの接続は 2 番目の行に一致します。

> **注記**
>
> よくある誤解として、ある特定のユーザー名についてサーバーが接続に対する一致を検出しようとしたとき、そのユーザーの名前を明示的に指定するすべての行が最初に使用されるという認識があります。これは正しくありません。このことは前述の例によって説明され、jeffrey による thomas.loc.gov からの接続は、User カラム値として 'jeffrey' を格納している行ではなく、ユーザー名のない行が最初に一致します。その結果、jeffrey は接続するときにユーザー名を指定したにもかかわらず、匿名ユーザーとして認証されます。

サーバーに接続できても権限が期待したものとは異なる場合、おそらくほかのアカウントとして認証されています。サーバーがユーザーの認識に使用したアカウントを見つけるには、CURRENT_USER() 関数を使用します。(セクション12.14「情報関数」を参照してください。)これは、一致する user テーブル行の User および Host 値を示す、user_name@host_name 形式の値を返します。たとえば、jeffrey が接続して、次のクエリーを発行したとします。

```
mysql> SELECT CURRENT_USER();
+----------------+
| CURRENT_USER() |
+----------------+
| @localhost     |
+----------------+
```

ここで表示された結果は、一致した user テーブル行の User カラム値がブランクであることを示しています。つまり、サーバーは jeffrey を匿名ユーザーとして扱っています。

認証の問題を診断するための別の方法は、user テーブルを出力し、テーブルを手作業でソートして、最初の一致が行われた行を確認する方法です。

6.2.5 アクセス制御、ステージ 2: リクエストの確認

接続が確立されたあと、サーバーはアクセス制御のステージ 2 に入ります。接続を介してユーザーが発行する各リクエストに対し、サーバーはユーザーが実行する操作を判別し、その操作を行う十分な権限をユーザーが持つかどうかを検査します。ここで、付与テーブルの権限カラムが役立ちます。これらの権限は、user、db、tables_priv、columns_priv、または procs_priv テーブルから取得される可能性があります。(それぞれの付与テーブルにあるカラムの一覧を示す、セクション6.2.2「権限システム付与テーブル」を参照すると役立つことがあります。)

user テーブルは、ユーザーにグローバルに割り当てられ、デフォルトのデータベースに関係なく適用される権限を付与します。たとえば、user テーブルによってユーザーに DELETE 権限が付与された場合、そのサーバーホスト上のあらゆるデータベースのあらゆるテーブルの行を削除できてしまいます。user テーブルの権限は、データベース管理者などの権限を必要とするユーザーに対してのみ付与するのが賢明です。ほかのユーザーについては、user テーブルのすべての権限を 'N' に設定しておき、具体的なレベルでのみ権限を付与するようにします。特定のデータベース、テーブル、カラム、またはルーチンに対して権限を付与することができます。

db テーブルは、データベース固有の権限を付与します。このテーブルのスコープカラムの値は、次の形式を取ることができます。

- ブランクの User 値は匿名ユーザーに一致します。ブランク以外の値は文字どおりに一致し、ユーザー名にワイルドカードはありません。
- Host および Db カラム内でワイルドカード文字「%」および「_」を使用することができます。これらは LIKE 演算子で実行されるパターンマッチング演算と同じ意味を持ちます。権限を付与するときにいずれかの文字を文字どおりに使用する場合、文字をバックスラッシュでエスケープする必要があります。たとえば、データベース名の一部としてアンダースコア文字 (「_」) を含めるには、GRANT ステートメント内でこれを 「_」と指定します。
- '%' またはブランクの Host 値は、「任意のホスト」を意味します。
- '%' またはブランクの Db 値は、「任意のデータベース」を意味します。

サーバーは user テーブルを読み取るのと同時に、db テーブルをメモリーに読み取ってソートします。サーバーは、Host、Db、および User スコープカラムに基づき db テーブルをソートします。user テーブルと同じように、ソートでは、もっとも具体的な値が最初に配置され、もっとも具体的でない値が最後に配置され、サーバーが一致するエントリを参照するときは、見つかった最初の一致が使用されます。

tables_priv、columns_priv、および procs_priv テーブルは、テーブル固有、カラム固有、およびルーチン固有の権限を付与します。これらのテーブルのスコープカラムの値は、次の形式を取ることができます。

- Host カラム内でワイルドカード文字「%」および「_」を使用することができます。これらは LIKE 演算子で実行されるパターンマッチング演算と同じ意味を持ちます。
- '%' またはブランクの Host 値は、「任意のホスト」を意味します。
- Db、Table_name、Column_name、および Routine_name カラムは、ワイルドカードを含めたり、ブランクにしたりすることができません。

サーバーは、Host、Db、および User カラムに基づき、tables_priv、columns_priv、および procs_priv テーブルをソートします。これは db テーブルのソートと同様ですが、Host カラムのみワイルドカードを含めることができるため、よりシンプルです。

サーバーはソート済みテーブルを使用して、サーバーが受け取るリクエストを検証します。SHUTDOWN や RELOAD などの管理権限が必要なリクエストでは、サーバーは user テーブル行のみチェックします。これは、管理権限を指定するのはこのテーブルだけであるためです。リクエストされた操作がその行によって許可される場合、サーバーはアクセス権限を付与し、そうでない場合はアクセスを拒否します。たとえば、ユーザーが mysqladmin shutdown を実行したいが、user テーブル行によって SHUTDOWN 権限がユーザーに付与されていない場合、サーバーは db テーブルさえもチェックせずにアクセスを拒否します。(これには Shutdown_priv カラムがないため、実行は不要です。)

データベース関連のリクエスト (INSERT、UPDATE など) の場合、サーバーは user テーブル行を参照することによって、ユーザーのグローバル権限を最初にチェックします。リクエストされた操作が行で許可されている場合、アクセス権限が付与されます。user テーブル内のグローバル権限が不十分な場合、サーバーは db テーブルをチェックすることによってユーザーのデータベース固有の権限を調べます。

サーバーは、db テーブルを参照し Host、Db、および User カラムの一致がないかチェックします。Host および User カラムは、接続するユーザーのホスト名および MySQL ユーザー名に突き合わせされます。Db カラムは、ユーザーがアクセスしようとしているデータベースに突き合わせされます。Host および User に該当する行がない場合、アクセスは拒否されます。

db テーブルエントリによって付与されるデータベース固有の権限を判別したあと、サーバーは user テーブルによって付与されるグローバル権限にそれらの権限を追加します。その結果、リクエストされた操作が許可される場合、アクセス権限が付与されます。そうでない場合、サーバーは続けて tables_priv および columns_priv テーブル内でユーザーのテーブル権限およびカラム権限をチェックし、これらをユーザーの権限に追加し、結果に基づいてアクセスを許可または拒否します。ストアドルーチン操作の場合、サーバーは tables_priv および columns_priv の代わりに procs_priv テーブルを使用します。

ブール条件によって表現すると、ユーザーの権限を計算する方法についての前述の説明は、次のように要約できます。

```
global privileges
OR (database privileges AND host privileges)
OR table privileges
OR column privileges
OR routine privileges
```

リクエストされた操作に対して user 行のグローバル権限が最初に十分ではないことがわかっている場合に、サーバーがそれらの権限をあとでデータベース権限、テーブル権限、およびカラム権限と組み合わせることの理由が明確でないかもしれません。この理由は、1 つのリクエストが複数のタイプの権限を必要とすることがあるためです。たとえば、INSERT INTO ... SELECT ステートメントを実行する場合、INSERT および SELECT 権限が両方必要です。使用する権限が、user テーブル行で 1 つの権限を付与し、db テーブル行でもう 1 つの権限を付与するようになっている場合があります。この状況で、リクエストを実行するために必要な権限はありますが、サーバーはいずれのテーブルから来たものかを見分けることができず、両方のテーブルのエントリによって付与される権限を組み合わせる必要があります。

6.2.6 権限変更が有効化される時期

mysqld を起動すると、すべての付与テーブルの内容がメモリーに読み取られます。インメモリーテーブルは、その時点でアクセス制御に有効になります。

ユーザーが GRANT、REVOKE、SET PASSWORD、RENAME USER などのアカウント管理ステートメントを使用して、付与テーブルを間接的に変更した場合、サーバーはそれらの変更を認識し、再びすぐに付与テーブルをメモリーにロードします。

INSERT、UPDATE、DELETE などのステートメントを使用して、付与テーブルを直接変更する場合、サーバーを再起動するか、テーブルをリロードするようサーバーに指示するまで、変更内容は権限チェックに影響しません。付与テーブルを直接変更したが、それらをリロードし忘れた場合、サーバーを再起動するまで変更は影響しません。このため、変更したのに違いが現れないことを不思議に思うことがあるかもしれません。

付与テーブルをリロードするようサーバーに指示するには、フラッシュ権限操作を実行します。これは、FLUSH PRIVILEGES ステートメントを発行するか、mysqladmin flush-privileges または mysqladmin reload コマンドを実行することによって行うことができます。

付与テーブルのリロードは、既存のクライアント接続のための権限に次のような影響を及ぼします。

- テーブルおよびカラムの権限の変更は、クライアントの次回のリクエストで有効になります。
- データベース権限の変更は、クライアントが次回 USE db_name ステートメントを実行したときに有効になります。

  > **注記**
  >
  > クライアントアプリケーションがデータベース名をキャッシュしている場合があるため、この影響は、実際に別のデータベースに変更したり、権限をフラッシュしたりしないかぎり、クライアントアプリケーションに認識されないことがあります。

- 接続済みクライアントについてのグローバルな権限およびパスワードは影響されません。これらの変更は、後続の接続についてのみ有効になります。

サーバーが --skip-grant-tables オプションを指定して起動された場合、サーバーは付与テーブルを読み取ったりアクセス制御を実装したりしません。すべてのユーザーが接続してあらゆることが可能であるため、セキュアではありません。そのように起動されたサーバーが、テーブルを読み取ってアクセスチェックを有効にするようにするには、権限をフラッシュします。

6.2.7 アクセス拒否エラーの原因

MySQL サーバーへの接続を試行したときに問題が発生した場合に問題を修正するために実行できる一連のアクションについて、次の項目で説明します。

- サーバーが実行中であることを確認します。そうでない場合、クライアントは接続できません。たとえば、サーバーに接続しようとして次のいずれかのようなメッセージで失敗した場合、サーバーが実行中でないことが 1 つの原因であることがあります。

```
shell> mysql
ERROR 2003: Can't connect to MySQL server on 'host_name' (111)
shell> mysql
```

```
ERROR 2002: Can't connect to local MySQL server through socket
'/tmp/mysql.sock' (111)
```

- サーバーは実行しているが、サーバーが待機しているのと異なる TCP/IP ポート、名前付きパイプ、または Unix ソケットファイルを使用して接続しようとしている場合もあります。これを修正するには、クライアントプログラムを呼び出すときに、適切なポート番号を指すように --port オプションを指定するか、--socket で適切な名前付きパイプまたは Unix ソケットファイルを指定します。ソケットファイルがある場所を見つけるには、次のコマンドを使用できます。

```
shell> netstat -ln | grep mysql
```

- サーバーがネットワーク接続を無視するように構成されていないこと、または (リモート側から接続しようとする場合に) サーバーのネットワークインタフェース上でローカル側でのみ待機するように構成されていないことを確認します。サーバーが --skip-networking を指定して起動された場合、サーバーは TCP/IP 接続を受け入れません。サーバーが --bind-address=127.0.0.1 を指定して起動された場合、サーバーはローカルのループバックインタフェース上でのみ TCP/IP 接続を待機するためリモート接続を受け入れません。

- ファイアウォールが MySQL へのアクセスをブロックしていないか確認します。ファイアウォールは、実行中のアプリケーションまたは MySQL によって通信用に使用されるポート番号 (デフォルトは 3306) を基準として構成されることがあります。Linux または Unix の場合、IP テーブル (または同様の機能の) 構成を調べてポートがブロックされていないことを確認します。Windows の場合、ZoneAlarm や Windows XP パーソナルファイアウォールなどのアプリケーションが MySQL ポートをブロックしないようにこれらを構成することが必要な場合があります。

- 付与テーブルが適切にセットアップされており、サーバーがこれをアクセス制御に使用できるようになっていることが必要です。一部の配布タイプ (Windows のバイナリ配布または Linux の RPM 配布など) では、インストールプロセスで、付与テーブルがある mysql データベースが初期化されます。これを行わない配布の場合、mysql_install_db プログラムを実行して付与テーブルを手動で初期化する必要があります。詳細については、セクション2.10.1「Unix 類似システムでのインストール後の手順」を参照してください。

  付与テーブルの初期化が必要かどうかを判別するには、データディレクトリの下にある mysql ディレクトリを参照します。(通常、データディレクトリは data または var という名前で、MySQL のインストールディレクトリの下にあります。)mysql データベースディレクトリに user.MYD という名前のファイルがあることを確認してください。ない場合、mysql_install_db プログラムを実行します。このプログラムを実行してサーバーを起動したあと、次のコマンドを実行して初期の権限をテストします。

```
shell> mysql -u root test
```

  サーバーはエラーを出さずにユーザーを接続するはずです。

- 新規インストールのあと、サーバーに接続してユーザーおよびユーザーのアクセス権限をセットアップするようにします。

```
shell> mysql -u root mysql
```

  MySQL root ユーザーには最初からパスワードがないため、サーバーはユーザーを接続するはずです。これにはセキュリティー面でのリスクが伴うため、別の MySQL アカウントをセットアップする際は、root アカウントのパスワードを設定することをお勧めします。初期パスワードの設定手順に関しては、セクション2.10.2「最初の MySQL アカウントのセキュリティー設定」を参照してください。

- 既存の MySQL インストールを新しいバージョンに更新した場合、mysql_upgrade スクリプトを実行したかどうかを確認します。行なっていない場合は実行します。付与テーブルの構造は、新機能が追加されるときにしばしば変更されるため、アップグレードしたあとは常に、テーブルの構造が最新であることを確認することをお勧めします。その手順は、セクション4.4.7「mysql_upgrade — MySQL テーブルのチェックとアップグレード」を参照してください。

- クライアントプログラムが接続しようとしたときに次のエラーメッセージを受け取る場合、サーバーはクライアントが生成可能なものよりも新しい形式のパスワードを予期していることを意味します。

```
shell> mysql
Client does not support authentication protocol requested
by server; consider upgrading MySQL client
```

  これに対処する方法については、セクション6.1.2.4「MySQL でのパスワードハッシュ」およびセクション B.5.2.4「クライアントは認証プロトコルに対応できません」を参照してください。

- クライアントプログラムはオプションファイルまたは環境変数に指定された接続パラメータを使用することに留意してください。コマンド行に接続パラメータを指定しないときにクライアントプログラムが間違ったデフォルトの接続パラメータを送信していると思われる場合、該当するオプションファイルおよび環境を確認し

てください。たとえば、オプションなしでクライアントを実行するときに Access denied を受け取る場合、いずれかのオプションファイルで古いパスワードを指定していないか確認してください。

--no-defaults を指定してクライアントプログラムを呼び出すことによって、オプションファイルの使用をクライアントプログラムによって抑制することができます。例:

```
shell> mysqladmin --no-defaults -u root version
```

クライアントが使用するオプションファイルの一覧は、セクション4.2.6「オプションファイルの使用」にあります。環境変数の一覧は、セクション2.12「環境変数」にあります。

- 次のエラーが出る場合、誤った root パスワードを使用していることを示しています。

```
shell> mysqladmin -u root -pxxxx ver
Access denied for user 'root'@'localhost' (using password: YES)
```

パスワードを指定していないのに前述のエラーが発生する場合、いずれかのオプションファイルに間違ったパスワードがリストされていることを意味します。前述の項目で説明したように、--no-defaults オプションを試してみてください。

パスワードの変更に関する情報は、セクション6.3.5「アカウントパスワードの割り当て」を参照してください。

root パスワードを紛失したか忘れた場合、セクションB.5.4.1「root のパスワードをリセットする方法」を参照してください。

- SET PASSWORD、INSERT、または UPDATE を使用してパスワードを変更する場合、PASSWORD() 関数を使用して、そのパスワードを暗号化する必要があります。これらのステートメントに対して PASSWORD() 関数を使用しないと、パスワードは機能しません。たとえば、次のステートメントはパスワードを割り当てますが、パスワードが暗号化されないため、ユーザーはあとで接続できません。

```
SET PASSWORD FOR 'abe'@'host_name' = 'eagle';
```

代わりに、パスワードを次のように設定します。

```
SET PASSWORD FOR 'abe'@'host_name' = PASSWORD('eagle');
```

CREATE USER または GRANT ステートメントまたは mysqladmin password コマンドを使用してパスワードを指定するときは、PASSWORD() 関数は不要です。これらはいずれも PASSWORD() を自動的に使用してパスワードを暗号化します。セクション6.3.5「アカウントパスワードの割り当て」およびセクション13.7.1.2「CREATE USER 構文」を参照してください。

- localhost はローカルホスト名のシノニムで、ホストを明示的に指定しない場合にクライアントが接続を試行するデフォルトホストでもあります。

そのようなシステムで前述の問題を回避するために、--host=127.0.0.1 オプションを使用して、サーバーホストを明示的に指定することができます。これにより、ローカルの mysqld サーバーへの TCP/IP 接続が作成されます。また、ローカルホストの実際のホスト名を使用する --host オプションを指定することによって TCP/IP を使用することもできます。この場合、サーバーと同じホスト上でクライアントプログラムを実行していても、ホスト名がサーバーホスト上の user テーブル行に指定されていなければなりません。

- Access denied というエラーメッセージは、ログインしようとしているユーザー名、接続を試行しているクライアントホスト、およびパスワードを使用したかどうかを通知します。通常では、エラーメッセージ内で指定されたホスト名およびユーザー名に正確に一致する 1 つの行を user テーブル内に持つようにします。たとえば、using password: NO というメッセージを含むエラーメッセージを受け取る場合、パスワードなしでログインしようとしたことを意味します。

- mysql -u user_name を使用してデータベースに接続しようとしたときに Access denied エラーを受け取った場合、user テーブルにおそらく問題があります。これをチェックするには、mysql -u root mysql を実行し、次の SQL ステートメントを発行します。

```
SELECT * FROM user;
```

この結果には、クライアントのホスト名および使用中の MySQL ユーザー名に一致する Host および User カラムを持つ行が含まれているはずです。

- MySQL サーバーを実行しているホストではないホストから接続しようとして次のエラーが発生する場合、クライアントホストと一致する Host 値を持つ行が user テーブルにないということを意味しています。

```
Host ... is not allowed to connect to this MySQL server
```

これは、接続しようとするときに使用するクライアントホスト名およびユーザー名の組み合わせに対するアカウントをセットアップすることによって修正できます。

接続元のマシンの IP アドレスまたはホスト名がわからない場合、Host カラム値が '%' の行を user テーブル内に作成するようにします。そして、そのクライアントマシンから接続しようとしたあとで、SELECT USER() クエリーを使用して、実際にどのように接続したかを確認します。そのあと、user テーブル行の '%' を、ログに表示されている実際のホスト名に変更します。そうしない場合、特定のユーザー名について任意のホストからの接続が可能になるため、システムはセキュアでない状態のままになります。

Linux では、このエラーが発生する可能性がある別の理由として、使用中のバージョンとは異なるバージョンの glibc ライブラリでコンパイルされたバイナリ MySQL バージョンを使用しているということがあります。この場合、オペレーティングシステムまたは glibc をアップグレードするか、MySQL のソース配布バージョンをダウンロードして自分でコンパイルします。ソース RPM のコンパイルおよびインストールは通常簡単であるため、これは大きな問題ではありません。

- 接続しようとしたときにホスト名を指定したが、ホスト名が非表示または IP アドレスとなっているエラーメッセージを受け取った場合、MySQL サーバーはクライアントホストの IP アドレスを名前に解決しようとしたときにエラーを受け取ったことを意味します。

```
shell> mysqladmin -u root -pxxxx -h some_hostname ver
Access denied for user 'root'@'' (using password: YES)
```

root として接続しようとして次のエラーを受け取った場合、User カラム値が 'root' の行が user テーブルになく、mysqld がクライアントに対してホスト名を解決できないことを意味します。

```
Access denied for user ''@'unknown'
```

これらのエラーは DNS の問題を示しています。これを修正するには、mysqladmin flush-hosts を実行して内部 DNS ホストキャッシュをリセットします。セクション8.11.5.2「DNS ルックアップの最適化とホストキャッシュ」を参照してください。

いくつかの永続的な解決策を次に示します。

- DNS サーバーの問題を判別して修正します。
- MySQL 付与テーブルにホスト名の代わりに IP アドレスを指定します。
- クライアントマシン名に対するエントリを、Unix の場合は /etc/hosts に、Windows の場合は \windows\hosts に配置します。
- mysqld を --skip-name-resolve オプションで起動します。
- mysqld を --skip-host-cache オプションで起動します。
- Unix で、サーバーとクライアントを同じマシンで実行している場合、localhost に接続します。localhost への Unix 接続は、TCP/IP ではなく Unix ソケットファイルを使用します。
- Windows で、サーバーとクライアントを同じマシンで実行していて、サーバーが名前付きパイプ接続をサポートしている場合、ホスト名 . (ピリオド) に接続します。. への接続には、TCP/IP ではなく名前付きパイプが使用されます。

- mysql -u root test は動作するが、mysql -h your_hostname -u root test が Access denied になる場合 (ここで、your_hostname はローカルホストの実際のホスト名)、ホスト名に対する正しい名前が user テーブル内にない可能性があります。このときよくある問題として、user テーブル行の Host 値は、修飾されていないホスト名を指定しているが、システムの名前解決ルーチンは完全修飾ドメイン名を返すということ (またはその逆) があります。たとえば、user テーブルにホスト 'pluto' のエントリがあるが、DNS が MySQL に対してホスト名が 'pluto.example.com' であると指示する場合、エントリは無効になります。ホストの IP アドレスを Host カラム値として格納するエントリを user テーブルに追加してみてください。(または、'pluto.%' などのワイルドカードを含む Host 値を持つエントリを user テーブルに追加することもできます。ただし、「%」で終わる Host 値を使用することは安全でないため推奨されません。)

- mysql -u user_name test が動作するが、mysql -u user_name other_db が動作しない場合、other_db という名前のデータベースについて、特定のユーザーにアクセス権限を付与していません。

- mysql -u user_name をサーバーホスト上で実行したときに動作するが、mysql -h host_name -u user_name をリモートクライアントホスト上で実行したときに動作しない場合、特定のユーザー名についてリモートホストからサーバーへのアクセスを有効にしていません。

- Access denied を受け取る理由がわからない場合、ワイルドカードを含む Host 値を持つすべてのエントリ ('%' または '_' 文字を含むエントリ) を user テーブルから削除します。非常によくある間違いは、Host='%' および User='some_user' という新しいエントリを挿入して、これにより、ユーザーは同じマシンから接続するよう localhost に指定できると思い込むことです。これが機能しない理由は、デフォルト権限に、Host = 'localhost' および User = '' というエントリを含むためです。このエントリには、'%' よりもさらに具体的な 'localhost' という Host 値があるため、localhost から接続するときは新しいエントリよりもこちらが優先して使用されます。正しい手順としては、Host = 'localhost' および User = 'some_user' という第 2 のエントリを挿入するか、Host = 'localhost' および User = '' のエントリを削除します。このエントリを削除したら、FLUSH PRIVILEGES ステートメントを発行して、付与テーブルのリロードを必ず行なってください。セクション6.2.4「アクセス制御、ステージ 1: 接続の検証」も参照してください。

- MySQL サーバーに接続できるが、SELECT ... INTO OUTFILE または LOAD DATA INFILE ステートメントを発行するたびに Access denied メッセージを受け取る場合、user テーブル内のエントリで FILE 権限が有効になっていません。

- 付与テーブルを直接 (たとえば、INSERT、UPDATE、または DELETE ステートメントを使用して) 変更し、変更が無視されたように思われる場合、サーバーに権限テーブルをリロードさせるために FLUSH PRIVILEGES ステートメントまたは mysqladmin flush-privileges コマンドを実行する必要があることを覚えておいてください。そうしない場合、サーバーが次回再起動するまで変更の影響はありません。root パスワードを UPDATE ステートメントで変更したあと、権限をフラッシュするまで新規パスワードを指定する必要はありません。これは、パスワードが変更されたことがサーバーにまだ認識されないためです。

- セッションの最中に権限が変更されたと思われる場合、MySQL 管理者によって権限が変更された可能性があります。付与テーブルのリロードは、新しいクライアント接続に影響しますが、セクション6.2.6「権限変更が有効化される時期」に示すように既存の接続にも影響します。

- Perl、PHP、Python、または ODBC プログラムとのアクセスに問題がある場合、mysql -u user_name db_name または mysql -u user_name -pyour_pass db_name を使用してサーバーに接続してみてください。mysql クライアントを使用して接続できる場合、問題はアクセス権限ではなくプログラムにあります。(-p とパスワードの間にスペースはありません。--password=your_pass 構文を使用してパスワードを指定することもできます。-p または --password オプションを使用してパスワード値を指定しない場合、MySQL はパスワードを要求します。)

- テスト目的で、--skip-grant-tables オプションを指定して mysqld サーバーを起動します。これにより、MySQL 付与テーブルを変更でき、SHOW GRANTS ステートメントを使用して、変更による望ましい影響があるかどうかを確認することができます。変更に満足したら、mysqladmin flush-privileges を実行して、権限をリロードするよう mysqld サーバーに指示します。これにより、サーバーを停止して再起動することなく新しい付与テーブルの内容の使用を開始することができます。

- すべてが失敗する場合、mysqld サーバーをデバッグオプション (--debug=d,general,query など) で起動します。これによって、発行された各コマンドに関する情報のほかに、試行された接続についてのホストおよびユーザー情報が出力されます。セクション24.4.3「DBUG パッケージ」を参照してください。

- MySQL 付与テーブルに関して何かほかの問題があって、メーリングリストに問題を投稿する必要があると思われる場合、MySQL 付与テーブルのダンプを常に提供してください。mysqldump mysql コマンドでテーブルをダンプできます。バグレポートを提出するには、セクション1.6「質問またはバグをレポートする方法」の説明を参照してください。mysqldump を実行するには --skip-grant-tables を指定して mysqld を再起動することが必要な場合もあります。

6.3 MySQL ユーザーアカウントの管理

このセクションでは、MySQL サーバーのクライアント用にアカウントを設定する方法について説明します。次のトピックについて説明します。

- MySQL で使用されるアカウント名とパスワードの意味、およびオペレーティングシステムで使用される名前とパスワードと比較する方法
- 新しいアカウントを設定し、既存のアカウントを削除する方法
- パスワードを変更する方法
- パスワードをセキュアに使用するためのガイドライン
- SSL によるセキュアな接続を使用する方法

すべてのユーザー管理 SQL ステートメントの構文および使用について説明されているセクション13.7.1「アカウント管理ステートメント」も参照してください。

6.3.1 ユーザー名とパスワード

MySQL では、mysql データベースの user テーブルにアカウントが格納されます。アカウントは、ユーザー名およびユーザーがサーバーに接続できるクライアントホスト (複数の場合あり) に関して定義されます。アカウントはパスワードを持っている場合もあります。user テーブルでのアカウントの表示については、セクション6.2.2「権限システム付与テーブル」を参照してください。MySQL 5.6 では、認証プラグインがサポートされているため、アカウントは一部の外部認証方式を使用して認証できます。セクション6.3.7「プラガブル認証」を参照してください。

ユーザー名とパスワードが MySQL で使用される方法と、オペレーティングシステムで使用される方法との間には、いくつかの違いがあります。

- MySQL で認証目的に使用されるユーザー名と、Windows または Unix で使用されるユーザー名 (ログイン名) とには、まったく関係がありません。Unix では、ほとんどの MySQL クライアントがデフォルトで、現在の Unix ユーザー名を MySQL ユーザー名として使用してログインを試みますが、これは便宜上の目的に過ぎません。クライアントプログラムでは、-u または --user オプションを使用して任意のユーザー名を指定することが許可されているため、簡単にデフォルトをオーバーライドできます。これは、ユーザー名を使用すればだれでもサーバーへの接続を試みることができることを意味するため、すべての MySQL アカウントがパスワードを持っていなければ、どのような方法でもデータベースをセキュアにすることはできません。パスワードを持っていないアカウントにユーザー名を指定するユーザーはだれでも、正常にサーバーに接続できます。

- MySQL ユーザー名は、最大で 16 文字の長さまで指定できます。オペレーティングシステムのユーザー名は、MySQL ユーザー名とは完全に無関係であるため、最大長が異なる可能性があります。たとえば、Unix ユーザー名は通常、8 文字までに制限されています。

  > **警告**
  >
  > MySQL サーバーおよびクライアントでは、MySQL ユーザー名の長さの制限がハードコードされているため、mysql データベース内のテーブル定義を変更してこれを回避しようとしても、効果がありません。
  >
  > セクション4.4.7「mysql_upgrade — MySQL テーブルのチェックとアップグレード」で説明する手順を使用する以外は、どのような方法でも mysql データベース内のテーブルは変更するべきではありません。その他の方法で MySQL のシステムテーブルを再定義しようとすると、未定義 (および未サポート) の動作が発生します。

- サーバーは (mysql.user テーブルに格納されたパスワードと照合する) MySQL ネイティブ認証を使用してクライアント接続を認証する際に、user テーブルに格納された MySQL パスワードを使用します。これらのパスワードと、オペレーティングシステムにログインするためのパスワードとには、まったく関係がありません。Windows または Unix マシンにログインする際に使用される「外部」パスワードと、そのマシン上の MySQL サーバーにアクセスする際に使用されるパスワードとの間には、必要な関係はありません。

  サーバーがプラグインを使用してクライアントを認証する場合は、プラグインで実装されている認証方式で、user テーブル内のパスワードが使用される場合と、使用されない場合があります。この場合、MySQL サーバーへの認証を行う際に、外部パスワードも使用される可能性があります。

- MySQL では独自アルゴリズムを使用して、user テーブルに格納されたパスワードが暗号化されます。この暗号化は、PASSWORD() SQL 関数で実装されたものと同じですが、Unix のログインプロセス時に使用されるものとは異なります。Unix のパスワード暗号化は、ENCRYPT() SQL 関数で実装されたものと同じです。PASSWORD() および ENCRYPT() 関数については、セクション12.13「暗号化関数と圧縮関数」を参照してください。

  MySQL バージョン 4.1 以降では、旧バージョンよりも強固な認証方式が採用され、接続プロセス時のパスワード保護が改善されています。TCP/IP パケットが盗聴されたり、mysql データベースが乗っ取られたりしてもセキュアです。(旧バージョンでは、パスワードが暗号化された形式で user テーブルに格納されますが、MySQL サーバーに接続する際に、暗号化されたパスワード値を知っていることが利用される可能性があります。)セクション6.1.2.4「MySQL でのパスワードハッシュ」では、パスワードの暗号化について詳細に説明します。

- ユーザー名およびパスワードに ASCII 文字のみが含まれている場合は、文字セットの設定に関係なく、サーバーに接続できます。ユーザー名またはパスワードに ASCII 文字以外が含まれているときに接続するには、クライアントは MYSQL_SET_CHARSET_NAME オプションと引数として適切な文字セット名を指定して、mysql_options() C API 関数を呼び出すようにしてください。これにより、指定された文字セットを使用した認証が実行されます。それ以外の場合、サーバーのデフォルト文字セットが認証のデフォルトエンコーディングと同じでなければ、認証に失敗します。

  標準の MySQL クライアントプログラムでは、先ほど説明したように、mysql_options() が呼び出される --default-character-set オプションがサポートされています。さらに、セクション10.1.4「接続文字セットおよび

照合順序」で説明したように、文字セットの自動検出もサポートされています。C API に基づいていないコネクタを使用するプログラムでは、mysql_options() と同等のものがコネクタで提供されている場合があり、それを代わりに使用できます。コネクタのドキュメントを確認してください。

前述の注は、クライアントの文字セットとして許可されていない ucs2、utf16、および utf32 には適用されません。

MySQL をインストールすると、付与テーブルにアカウントの初期セットが移入されます。これらのアカウントの名前およびアクセス権限については、セクション2.10.2「最初の MySQL アカウントのセキュリティー設定」で説明されています。ここでは、パスワードを割り当てる方法についても説明されています。そのあと、通常は CREATE USER、GRANT、および REVOKE などのステートメントを使用して、MySQL アカウントを設定、変更、および削除します。セクション13.7.1「アカウント管理ステートメント」を参照してください。

コマンド行クライアントを使用して MySQL サーバーに接続する場合は、使用するアカウントでの必要に応じて、ユーザー名とパスワードを指定します。

```
shell> mysql --user=monty --password=password db_name
```

短いオプションを好む場合は、コマンドは次のようになります。

```
shell> mysql -u monty -ppassword db_name
```

-p オプションとそれに続くパスワード値との間には、空白文字を入れないでください。

コマンド行で --password または -p オプションに続く password 値を省略した場合は、値を 1 つ指定するように求めるプロンプトがクライアントに表示されます。

コマンド行でパスワードを指定することは、セキュアでないと考えるべきです。セクション6.1.2.1「パスワードセキュリティーのためのエンドユーザーガイドライン」を参照してください。オプションファイルを使用すれば、コマンド行でパスワードを指定することを回避できます。

ユーザー名、パスワード、およびその他の接続パラメータの指定に関する追加情報については、セクション4.2.2「MySQL サーバーへの接続」を参照してください。

6.3.2 ユーザーアカウントの追加

MySQL アカウントは、次の 2 つの方法で作成できます。

- アカウントを作成するためのステートメント (CREATE USER や GRANT など) を使用する。これらのステートメントを発行すると、サーバーによって付与テーブルへの適切な変更が行われます。
- INSERT、UPDATE、DELETE などのステートメントを使用して、MySQL 付与テーブルを直接操作する。

アカウント作成のステートメントを使用する方が、付与テーブルを直接操作するよりも簡潔で、エラーの発生率も低いため、推奨される方法です。CREATE USER および GRANT については、セクション13.7.1「アカウント管理ステートメント」で説明されています。

アカウントを作成するためのもう 1 つのオプションは、GUI ツール MySQL Workbench を使用する方法です。または、MySQL アカウント管理の機能を提供する使用可能な複数のサードパーティープログラムのいずれかを使用します。phpMyAdmin は、このようなプログラムの 1 つです。

次の例では、mysql クライアントプログラムを使用して、新しいアカウントを設定する方法を示します。これらの例は、セクション2.10.2「最初の MySQL アカウントのセキュリティー設定」で説明するデフォルトに従って、権限が設定されていることが前提となっています。つまり、変更を行うには、MySQL root ユーザーとして MySQL サーバーに接続する必要があり、root アカウントは mysql データベースに対する INSERT 権限および RELOAD 管理権限を持っている必要があります。

該当する例で注記したように、特定の制約が有効になるようにサーバーの SQL モードが設定されている場合は、一部のステートメントに失敗します。特に、厳密モード (STRICT_TRANS_TABLES、STRICT_ALL_TABLES、および NO_AUTO_CREATE_USER) では、サーバーが一部のステートメントを受け入れることが回避されます。このような場合のために、回避策を示します。SQL モードおよびそれによる付与テーブルの操作への影響についての詳細は、セクション5.1.7「サーバー SQL モード」およびセクション13.7.1.4「GRANT 構文」を参照してください。

まず、mysql プログラムを使用して、MySQL root ユーザーとしてサーバーに接続します。

```
shell> mysql --user=root mysql
```

root アカウントにパスワードを割り当てた場合は、この mysql コマンドと、このセクションで後述するコマンドの両方に、--password または -p オプションを指定する必要もあります。

root としてサーバーに接続したあとは、新しいアカウントを追加できます。次のステートメントは、GRANT を使用して 4 つの新しいアカウントを設定します。

```
mysql> CREATE USER 'monty'@'localhost' IDENTIFIED BY 'some_pass';
mysql> GRANT ALL PRIVILEGES ON *.* TO 'monty'@'localhost'
    ->     WITH GRANT OPTION;
mysql> CREATE USER 'monty'@'%' IDENTIFIED BY 'some_pass';
mysql> GRANT ALL PRIVILEGES ON *.* TO 'monty'@'%'
    ->     WITH GRANT OPTION;
mysql> CREATE USER 'admin'@'localhost';
mysql> GRANT RELOAD,PROCESS ON *.* TO 'admin'@'localhost';
mysql> CREATE USER 'dummy'@'localhost';
```

これらのステートメントで作成されたアカウントには、次のプロパティーがあります。

- アカウントのうち 2 つは、ユーザー名が monty、パスワードが some_pass です。どちらのアカウントも、すべてを実行するための完全な権限を持っているスーパーユーザーアカウントです。'monty'@'localhost' アカウントは、ローカルホストから接続する際にのみ使用できます。'monty'@'%' アカウントでは、ホスト部分に '%' ワイルドカードが使用されているため、これを使用すれば任意のホストから接続できます。

  monty として任意の場所から接続できるようにするには、monty に両方のアカウントが必要です。localhost アカウントがない場合は、monty がローカルホストから接続する際に、mysql_install_db で作成された localhost の匿名ユーザーアカウントが優先されます。その結果、monty は匿名ユーザーとして処理されます。その理由は、匿名ユーザーアカウントが 'monty'@'%' アカウントよりも固有の Host カラム値を持っているため、user テーブルのソート順でより早く表示されるためです。(user テーブルのソートについては、セクション6.2.4「アクセス制御、ステージ 1: 接続の検証」で説明されています。)

- 'admin'@'localhost' アカウントにはパスワードがありません。このアカウントは、admin がローカルホストから接続する際にのみ使用できます。これには、RELOAD および PROCESS の管理者権限が付与されます。これらの権限を持つ admin ユーザーは、mysqladmin reload、mysqladmin refresh、mysqladmin flush-xxx コマンド、および mysqladmin processlist を実行できます。任意のデータベースにアクセスするための権限は付与されません。その他の GRANT ステートメントを発行すれば、あとでこのような権限を追加できることがあります。

- 'dummy'@'localhost' アカウントにはパスワードがありません。このアカウントは、ローカルホストから接続する際にのみ使用できます。権限は付与されません。あとでアカウントに特定の権限を付与することを前提としています。

NO_AUTO_CREATE_USER SQL モードが有効になっている場合は、パスワードなしでアカウントを作成するステートメントに失敗します。これに対処するには、空でないパスワードを指定する IDENTIFIED BY 句を使用します。

アカウントに対する権限をチェックするには、SHOW GRANTS を使用します。

```
mysql> SHOW GRANTS FOR 'admin'@'localhost';
+-----------------------------------------------------+
| Grants for admin@localhost                          |
+-----------------------------------------------------+
| GRANT RELOAD, PROCESS ON *.* TO 'admin'@'localhost' |
+-----------------------------------------------------+
```

CREATE USER および GRANT の代わりとして、直接 INSERT ステートメントを発行してから、FLUSH PRIVILEGES を使用して付与テーブルを再ロードするようにサーバーに指示することで、同じアカウントを作成できます。

```
shell> mysql --user=root mysql
mysql> INSERT INTO user
    ->     VALUES('localhost','monty',PASSWORD('some_pass'),
    ->     'Y','Y','Y','Y','Y','Y','Y','Y','Y','Y','Y','Y','Y','Y');
mysql> INSERT INTO user
    ->     VALUES('%','monty',PASSWORD('some_pass'),
    ->     'Y','Y','Y','Y','Y','Y','Y','Y','Y','Y','Y','Y','Y','Y',
    ->     'Y','Y','Y','Y','Y','Y','Y','Y','Y','Y','Y','Y','Y','Y',
    ->     '','','','',0,0,0,0);
mysql> INSERT INTO user SET Host='localhost',User='admin',
    ->     Reload_priv='Y', Process_priv='Y';
mysql> INSERT INTO user (Host,User,Password)
    ->     VALUES('localhost','dummy','');
```

```
mysql> FLUSH PRIVILEGES;
```

INSERT を使用してアカウントを作成する場合は、FLUSH PRIVILEGES を使用して、付与テーブルを再ロードするようにサーバーに指示する必要があります。そうしなければ、サーバーを再起動するまで変更が認識されません。CREATE USER では、FLUSH PRIVILEGES が必要ありません。

INSERT で PASSWORD() 関数を使用する理由は、パスワードを暗号化するためです。CREATE USER ステートメントでは自動的にパスワードが暗号化されるため、PASSWORD() は必要ありません。

'Y' 値を指定すると、アカウントに対する権限が有効になります。MySQL のバージョンによっては、最初の 2 つの INSERT ステートメントに異なる数の 'Y' 値を使用する必要がある場合もあります。admin アカウントに対する INSERT ステートメントでは、SET を使用することで、さらに読み取り可能になるように拡張された INSERT 構文が採用されています。

dummy アカウントに対する INSERT ステートメントでは、user テーブル行内の Host、User、および Password カラムにのみ、値が割り当てられます。権限カラムは明示的に設定されないため、MySQL によってすべてのカラムに、デフォルト値の 'N' が割り当てられます。これは、CREATE USER の動作と同等です。

厳密な SQL モードが有効になっている場合は、デフォルト値を持たないすべてのカラムに値を指定する必要があります。この場合、INSERT ステートメントは、明示的に ssl_cipher、x509_issuer、および x509_subject カラムに値を指定する必要があります。

スーパーユーザーアカウントを設定するために必要な操作は、すべての権限カラムが 'Y' に設定された user テーブル行を挿入することだけです。user テーブルの権限はグローバルであるため、その他の付与テーブルのいずれにもエントリは必要ありません。

次の例では、3 つのアカウントを作成し、それらに特定のデータベースへのアクセス権を付与します。それぞれのユーザー名は custom で、パスワードは obscure です。

CREATE USER および GRANT を使用してアカウントを作成するには、次のステートメントを使用します。

```
shell> mysql --user=root mysql
mysql> CREATE USER 'custom'@'localhost' IDENTIFIED BY 'obscure';
mysql> GRANT SELECT,INSERT,UPDATE,DELETE,CREATE,DROP
    ->     ON bankaccount.*
    ->     TO 'custom'@'localhost';
mysql> CREATE USER 'custom'@'host47.example.com' IDENTIFIED BY 'obscure';
mysql> GRANT SELECT,INSERT,UPDATE,DELETE,CREATE,DROP
    ->     ON expenses.*
    ->     TO 'custom'@'host47.example.com';
mysql> CREATE USER 'custom'@'server.domain' IDENTIFIED BY 'obscure';
mysql> GRANT SELECT,INSERT,UPDATE,DELETE,CREATE,DROP
    ->     ON customer.*
    ->     TO 'custom'@'server.domain';
```

3 つのアカウントは、次のように使用できます。

- 1 番目のアカウントは、bankaccount データベースにアクセスできますが、ローカルホストからに限定されます。
- 2 番目のアカウントは、expenses データベースにアクセスできますが、host47.example.com ホストからに限定されます。
- 3 番目のアカウントは、customer データベースにアクセスできますが、server.domain ホストからに限定されます。

GRANT を使用せずに custom アカウントを設定するには、次のように INSERT ステートメントを使用して、付与テーブルを直接変更します。

```
shell> mysql --user=root mysql
mysql> INSERT INTO user (Host,User,Password)
    ->     VALUES('localhost','custom',PASSWORD('obscure'));
mysql> INSERT INTO user (Host,User,Password)
    ->     VALUES('host47.example.com','custom',PASSWORD('obscure'));
mysql> INSERT INTO user (Host,User,Password)
    ->     VALUES('server.domain','custom',PASSWORD('obscure'));
mysql> INSERT INTO db
    ->     (Host,Db,User,Select_priv,Insert_priv,
    ->     Update_priv,Delete_priv,Create_priv,Drop_priv)
    ->     VALUES('localhost','bankaccount','custom',
    ->     'Y','Y','Y','Y','Y','Y');
```

```
mysql> INSERT INTO db
    ->     (Host,Db,User,Select_priv,Insert_priv,
    ->     Update_priv,Delete_priv,Create_priv,Drop_priv)
    ->     VALUES('host47.example.com','expenses','custom',
    ->     'Y','Y','Y','Y','Y','Y');
mysql> INSERT INTO db
    ->     (Host,Db,User,Select_priv,Insert_priv,
    ->     Update_priv,Delete_priv,Create_priv,Drop_priv)
    ->     VALUES('server.domain','customer','custom',
    ->     'Y','Y','Y','Y','Y','Y');
mysql> FLUSH PRIVILEGES;
```

最初の 3 つの INSERT ステートメントは、付与されたパスワードを使用してさまざまなホストから接続することをユーザー custom に許可する user テーブルエントリを追加しますが、グローバルな権限は付与しません (すべての権限はデフォルト値の 'N' に設定されます)。次の 3 つの INSERT ステートメントは、bankaccount、expenses、および customer データベースに対する権限を custom に付与するが、適切なホストからのアクセス時に限定される db テーブルエントリを追加します。通常どおりに付与テーブルを直接変更する場合は、権限の変更が有効になるように、FLUSH PRIVILEGES を使用して再ロードするようにサーバーに指示する必要があります。

特定のドメイン (mydomain.com など) 内のすべてのマシンからのアクセス権を持つユーザーを作成する際には、アカウント名のホスト部分に 「%」 ワイルドカード文字を使用できます。

```
mysql> CREATE USER 'myname'@'%.mydomain.com' IDENTIFIED BY 'mypass';
```

付与テーブルを直接変更することで同じ処理を行うには、次のように実行します。

```
mysql> INSERT INTO user (Host,User,Password,...)
    ->     VALUES('%.mydomain.com','myname',PASSWORD('mypass'),...);
mysql> FLUSH PRIVILEGES;
```

6.3.3 ユーザーアカウントの削除

アカウントを削除するには、セクション13.7.1.3「DROP USER 構文」で説明されている DROP USER ステートメントを使用します。

6.3.4 アカウントリソース制限の設定

MySQL サーバーリソースの使用を制限する方法の 1 つは、グローバルな max_user_connections システム変数をゼロ以外の値に設定することです。これにより、任意の特定のアカウントで実行できる同時接続の数が制限されますが、一度クライアントが接続したら、実行内容には制限が課されません。さらに、max_user_connections を設定しても、各アカウントの管理は有効になりません。どちらのタイプの制御も、多くの MySQL 管理者 (特に、インターネットサービスプロバイダの従業員) にとって関心のあるものです。

MySQL 5.6 では、各アカウントに対して次のサーバーリソースの使用を制限できます。

- アカウントが 1 時間ごとに発行できるクエリーの数
- アカウントが 1 時間ごとに発行できる更新の数
- アカウントが 1 時間ごとにサーバーに接続できる回数
- アカウントによるサーバーへの同時接続の数

クエリー制限に対して、クライアントが発行できるステートメントがすべてカウントされます (その結果がクエリーキャッシュから提供される場合は除きます)。更新制限に対して、データベースまたはテーブルを変更するステートメントのみがカウントされます。

このコンテキストでは、「アカウント」は mysql.user テーブル内の行に対応しています。つまり、接続に適用される user テーブル行内の User および Host 値に対して、接続が評価されます。たとえば、アカウント 'usera'@'%.example.com' は、example.com ドメイン内の任意のホストから接続することを usera に許可するために、usera および %.example.com の User および Host 値を持つ user テーブル内の行に対応しています。この場合、このような接続ではすべて同じアカウントが使用されるため、サーバーは、usera による example.com ドメイン内の任意のホストからのすべての接続に、この行のリソース制限をまとめて適用します。

MySQL 5.0.3 よりも前では、ユーザーの接続元である実際のホストに対して、「アカウント」が評価されていました。--old-style-user-limits オプションを付けてサーバーを起動すれば、この古いアカウント方式を選択できます。この場合、usera が host1.example.com と host2.example.com から同時に接続すると、サーバーは各接続に

個別にアカウントリソースの制限を適用します。usera が host1.example.com から再度接続すると、サーバーはそのホストからの既存の接続とともに、その接続に対する制限を適用します。

アカウントに対してリソース制限を設定するには、GRANT ステートメントを使用します (セクション 13.7.1.4「GRANT 構文」を参照してください)。制限される各リソースの名前を指定する WITH 句を指定します。各制限のデフォルト値は、ゼロ (制限なし) です。たとえば、制限された方法でのみ customer データベースにアクセスできる新しいアカウントを作成するには、次のようなステートメントを発行します。

```
mysql> CREATE USER 'francis'@'localhost' IDENTIFIED BY 'frank';
mysql> GRANT ALL ON customer.* TO 'francis'@'localhost'
    ->     WITH MAX_QUERIES_PER_HOUR 20
    ->          MAX_UPDATES_PER_HOUR 10
    ->          MAX_CONNECTIONS_PER_HOUR 5
    ->          MAX_USER_CONNECTIONS 2;
```

制限タイプの名前をすべて WITH 句に指定する必要はありませんが、名前を指定したものは任意の順序で表示できます。1 時間ごとの各制限の値は、1 時間当たりの回数を表す整数にするようにしてください。MAX_USER_CONNECTIONS では、制限はアカウントによる同時接続の最大数を表す整数です。この制限がゼロに設定されている場合は、グローバルな max_user_connections システム変数の値によって同時接続の数が決定されます。max_user_connections もゼロである場合は、アカウントに制限がありません。

アカウントに対する既存の制約を変更するには、グローバルレベル (ON *.*) で GRANT USAGE ステートメントを使用します。次のステートメントは、francis に対するクエリー制限を 100 に変更します。

```
mysql> GRANT USAGE ON *.* TO 'francis'@'localhost'
    ->     WITH MAX_QUERIES_PER_HOUR 100;
```

このステートメントは、指定された制限値のみを変更し、それ以外のアカウントは未変更のままにします。

制限を削除するには、その値をゼロに設定します。たとえば、francis が接続できる 1 時間当たりの回数に対する制限を削除するには、次のステートメントを使用します。

```
mysql> GRANT USAGE ON *.* TO 'francis'@'localhost'
    ->     WITH MAX_CONNECTIONS_PER_HOUR 0;
```

すでに説明したように、アカウントに対する同時接続の制限は、MAX_USER_CONNECTIONS 制限および max_user_connections システム変数によって決定されます。グローバルな max_user_connections 値が 10 であり、3 つのアカウントが GRANT を使用して指定されたリソース制限を持っていると仮定します。

```
GRANT ... TO 'user1'@'localhost' WITH MAX_USER_CONNECTIONS 0;
GRANT ... TO 'user2'@'localhost' WITH MAX_USER_CONNECTIONS 5;
GRANT ... TO 'user3'@'localhost' WITH MAX_USER_CONNECTIONS 20;
```

user1 の MAX_USER_CONNECTIONS はゼロであるため、接続の制限は 10 (グローバルな max_user_connections 値) です。user2 と user3 の MAX_USER_CONNECTIONS はゼロ以外であるため、接続の制限はそれぞれ 5 と 20 です。

サーバーはアカウントに対応する user テーブル行に、アカウントに対するリソース制限を格納します。max_questions、max_updates、または max_connections カラムには、1 時間当たりの制限が格納され、max_user_connections カラムには、MAX_USER_CONNECTIONS の制限が格納されます。セクション 6.2.2「権限システム付与テーブル」を参照してください。

任意のアカウントによるリソースのいずれかの使用に対してゼロ以外の制限が設定されている場合は、リソース使用のカウントが発生します。

サーバーが実行されると、各アカウントがリソースを使用する回数がカウントされます。過去 1 時間以内にアカウントが接続数に対する制限に達した場合は、その時間が経過するまで、そのアカウントによる以降の接続が拒否されます。同様に、アカウントがクエリーまたは更新の回数に対する制限に達した場合も、その時間が経過するまで、以降のクエリーまたは更新が拒否されます。このような場合はすべて、適切なエラーメッセージが発行されます。

リソースのカウントはクライアントごとではなく、アカウントごとに実行されます。たとえば、アカウントのクエリー制限が 50 である場合は、サーバーへの 2 つの同時クライアント接続を作成しても、制限を 100 に増加できません。両方の接続で発行されたクエリーは、まとめてカウントされます。

現在の 1 時間ごとのリソース使用のカウントは、すべてのアカウントに対してグローバルにリセットすることも、特定のアカウントごとに個別にリセットすることもできます。

- すべてのアカウントに対して現在のカウントをゼロにリセットするには、FLUSH USER_RESOURCES ステートメント発行します。また、(たとえば、FLUSH PRIVILEGES ステートメントまたは mysqladmin reload コマンドを使用して) 付与テーブルを再ロードして、カウントをリセットすることもできます。

- 各アカウントの制限のいずれかを再度付与すれば、そのカウントをゼロに設定できます。これを行うには、前述のように GRANT USAGE を使用して、アカウントが現在持っている値と等しい制限値を指定します。

カウンタをリセットしても、MAX_USER_CONNECTIONS の制限は影響を受けません。

サーバーが起動されると、すべてのカウントがゼロで始まります。再起動後は、カウントが繰り越されません。

MAX_USER_CONNECTIONS の制限では、アカウントが許可されている接続の最大数を現在開いている場合に、エッジケースが発生する可能性があります。接続が発生する時点までにサーバーで切断が完全に処理されていない場合に、切断後にすぐに接続すると、エラー (ER_TOO_MANY_USER_CONNECTIONS または ER_USER_LIMIT_REACHED) が発生する可能性があります。サーバーで切断処理が終了すると、別の接続がもう一度許可されます。

6.3.5 アカウントパスワードの割り当て

MySQL サーバーに接続するクライアントに必要な証明書には、パスワードを含めることができます。このセクションでは、MySQL アカウントにパスワードを割り当てる方法について説明します。クライアント認証は、プラグインを使用することで発生します。セクション6.3.7「プラガブル認証」を参照してください。

CREATE USER を使用して新しいアカウントを作成するときに、パスワードを割り当てるには、IDENTIFIED BY 句を含めます。

```
mysql> CREATE USER 'jeffrey'@'localhost'
    -> IDENTIFIED BY 'mypass';
```

既存のアカウントにパスワードを割り当てる方法またはそれを変更する方法の 1 つは、SET PASSWORD ステートメントを発行することです。

```
mysql> SET PASSWORD FOR
    -> 'jeffrey'@'localhost' = PASSWORD('mypass');
```

MySQL では、mysql データベースの user テーブルにパスワードが格納されます。mysql データベースに対する UPDATE 権限を持っている root などのユーザーのみが、ほかのユーザーのパスワードを変更できます。匿名ユーザーとして接続していない場合は、FOR 句を省略すれば自分自身のパスワードを変更できます。

```
mysql> SET PASSWORD = PASSWORD('mypass');
```

old_passwords システム変数の値によって、PASSWORD() で使用されるハッシュ化方式が決定されます。その関数を使用しているパスワードを指定しても、SET PASSWORD でそのパスワードの形式が正しくないとして拒否される場合は、old_passwords を設定してハッシュ化方式を変更する必要がある場合もあります。許可されている値については、セクション5.1.4「サーバーシステム変数」を参照してください。

read_only システム変数を有効にすると、SUPER 権限を持っていない任意のユーザーによる SET PASSWORD ステートメントの使用が回避されます。

また、グローバルレベル (ON *.*) で GRANT USAGE ステートメントを使用すると、アカウントの現在の権限に影響を与えずに、アカウントにパスワードを割り当てることができます。

```
mysql> GRANT USAGE ON *.* TO 'jeffrey'@'localhost'
    -> IDENTIFIED BY 'mypass';
```

コマンド行からパスワードを割り当てるには、mysqladmin コマンドを使用します。

```
shell> mysqladmin -u user_name -h host_name password "newpwd"
```

このコマンドでパスワードが設定されるアカウントは、User カラム内の user_name に一致する user テーブル行を持つアカウント、および Host カラム内の接続元のクライアントホストです。

認証中にクライアントがサーバーに接続すると、MySQL では user テーブル内のパスワードが暗号化されたハッシュ値 (PASSWORD() でパスワードに返される値) として処理されます。アカウントにパスワードを割り当てる際は、平文のパスワードではなく、暗号化された値を格納することが重要です。次のガイドラインを使用します。

- CREATE USER、IDENTIFIED BY 句を含む GRANT、または mysqladmin password コマンドを使用してパスワードを割り当てると、自動的にパスワードが暗号化されます。リテラルの平文パスワードを指定します。

  ```
  mysql> CREATE USER 'jeffrey'@'localhost'
      -> IDENTIFIED BY 'mypass';
  ```

- CREATE USER または GRANT では、PASSWORD() でパスワードに返されるハッシュ値がわかっている場合に、平文パスワードの送信を回避できます。前にキーワード PASSWORD を付けたハッシュ値を指定します。

```
mysql> CREATE USER 'jeffrey'@'localhost'
    -> IDENTIFIED BY PASSWORD '*90E462C37378CED12064BB3388827D2BA3A9B689';
```

- SET PASSWORD を使用して、アカウントに空でないパスワードを割り当てる場合は、PASSWORD() 機能を使用してパスワードを暗号化する必要があります。それ以外の場合は、パスワードが平文として格納されます。次のようなパスワードを割り当てると仮定します。

```
mysql> SET PASSWORD FOR
    -> 'jeffrey'@'localhost' = 'mypass';
```

その結果、暗号化された値ではなく、リテラル値の 'mypass' がパスワードとして user テーブルに格納されます。jeffrey がこのパスワードを使用してサーバーへの接続を試みると、その値が暗号化され、user テーブルに格納された値と比較されます。ただし、格納されている値はリテラル文字列の 'mypass' であるため、比較に失敗し、サーバーは接続を拒否し、「アクセスが拒否されました」というエラーを返します。

> **注記**
>
> PASSWORD() の暗号化は、Unix のパスワード暗号化とは異なります。セクション 6.3.1「ユーザー名とパスワード」を参照してください。

SET PASSWORD、GRANT、または mysqladmin を使用してパスワードを割り当てることが望ましいですが、user テーブルを直接変更することもできます。この場合、FLUSH PRIVILEGES を使用して、サーバーで付与テーブルを再ロードさせる必要もあります。それ以外の場合は、サーバーを再起動するまで変更が認識されないままです。

6.3.6 パスワードの期限切れとサンドボックスモード

MySQL 5.6 では、データベース管理者がアカウントのパスワードを期限切れにして、ユーザーに自分のパスワードをリセットするように要求できるパスワード期限切れ機能が導入されています。直後の説明では、パスワードの期限切れの現在の動作について記述します。そのあと、どの機能がどのような状況で使用できるのかを理解する際に役立つ背景として、この機能の開発について、複数のバージョンにわたって詳細に説明します。ただし、すべての機能と修正を活用できるようにするには、可能なかぎり、MySQL 5.6 の最新バージョンを使用するようにしてください。

パスワードの期限切れの動作

アカウントのパスワードを期限切れにするには、ALTER USER ステートメントを使用します。例:

```
ALTER USER 'myuser'@'localhost' PASSWORD EXPIRE;
```

このステートメントは、password_expired カラムを 'Y' に設定することで、指定されたアカウントに関連付けられた mysql.user テーブルの行を変更します。これにより、アカウントが開いている現在の接続は影響を受けません。そのアカウントを使用する後続の各接続では、サーバーはクライアントを切断するか、「サンドボックスモード」でクライアントを処理します。このモードでは、期限切れのパスワードをリセットするために必要な操作のみが、サーバーからクライアントに許可されます。(サーバーで実行されるアクションは、クライアントとサーバー両方の設定に依存します。)

サーバーがクライアントを切断すると、ER_MUST_CHANGE_PASSWORD_LOGIN エラーが返されます。

```
shell> mysql -u myuser -p
Password: ******
ERROR 1862 (HY000): Your password has expired. To log in you must
change it using a client that supports expired passwords.
```

サーバーがクライアントをサンドボックスモードに移行すると、これらの操作がクライアントセッション内で許可されます。

- クライアントは SET PASSWORD を使用して、アカウントのパスワードをリセットできます。これは、password_expired カラムを 'N' に設定することで、現在のアカウントに関連付けられた mysql.user テーブルの行を変更します。パスワードがリセットされると、サーバーはそのセッション、およびアカウントを使用する後続の接続への通常のアクセスをリストアします。

  パスワードを現在の値に設定すれば、「リセット」できます。適切なポリシーとして、別のパスワードを選択することをお勧めします。

- クライアントは SET ステートメントを使用できます。たとえば、old_passwords システム変数をデフォルトとは異なる値に設定する必要のあるハッシュ化形式がアカウントのパスワードで使用されている場合は、パスワードをリセットする前に、これが必要になることがあります。

セッション内で許可されていない操作の場合、サーバーは ER_MUST_CHANGE_PASSWORD エラーを返します。

```
mysql> USE test;
ERROR 1820 (HY000): You must SET PASSWORD before executing this statement
```

すでに説明したように、サーバーがパスワードの期限が切れているクライアントを切断するのか、サンドボックスモードに移行するのかは、クライアント設定とサーバー設定の組み合わせによって異なります。次の説明では、関連する設定と、それらがどのように相互作用するのかについて記述します。

クライアント側では、特定のクライアントが期限切れパスワードに対してサンドボックスモードを処理できるかどうかを示します。C クライアントライブラリを使用するクライアントの場合、これを実行するための方法が 2 つあります。

- 接続前に MYSQL_OPT_CAN_HANDLE_EXPIRED_PASSWORDS フラグを mysql_options() に渡します。

```
arg = 1;
result = mysql_options(mysql,
                       MYSQL_OPT_CAN_HANDLE_EXPIRED_PASSWORDS, &arg);
```

- 接続時に CLIENT_CAN_HANDLE_EXPIRED_PASSWORDS フラグを mysql_real_connect() に渡します。

```
mysql = mysql_real_connect(mysql,
                           host, user, password, "test",
                           port, unix_socket,
                           CLIENT_CAN_HANDLE_EXPIRED_PASSWORDS);
```

その他の MySQL コネクタには、サンドボックスモードを処理する準備ができていることを示す独自の規則が用意されています。関連するコネクタのドキュメントを参照してください。

サーバー側では、クライアントが期限切れパスワードを処理できることを示している場合、サーバーはサンドボックスモードに移行します。

クライアントが期限切れパスワードを処理できることを示していない場合 (または、そのように示すことができない古いバージョンのクライアントライブラリを使用している場合)、サーバーのアクションは disconnect_on_expired_password システム変数の値によって異なります。

- disconnect_on_expired_password が有効になっている (デフォルト) 場合、サーバーはクライアントを切断し、ER_MUST_CHANGE_PASSWORD_LOGIN エラーを返します。
- disconnect_on_expired_password が無効になっている場合、サーバーはクライアントをサンドボックスモードに移行します。

前述のクライアント設定とサーバー設定は、期限切れパスワードを持つアカウントにのみ適用されます。クライアントが期限切れでないパスワードを使用して接続すれば、サーバーはクライアントを通常どおりに処理します。

パスワードの期限切れ機能の開発

次のタイムラインには、さまざまなパスワードの期限切れ機能が追加されたバージョンを記載します。

MySQL 5.6.6: パスワードの期限切れの初期実装

DBA がアカウントのパスワードを期限切れにすることができるように、password_expired カラムが mysql.user テーブルに導入されました。このカラムのデフォルト値は、'N' (期限切れなし) です。

password_expired カラムを 'Y' に設定するための SQL インタフェースとして、ALTER USER ステートメントが導入されました。

期限切れパスワードを持つアカウントを使用して接続すると、SET PASSWORD ステートメントのみが許可される「サンドボックスモード」に移行します。その他のステートメントの場合、サーバーは ER_MUST_CHANGE_PASSWORD エラーを返します。この目的は、サーバーでその他の操作が許可される前に、クライアントにパスワードのリセットを強制することです。SET PASSWORD はアカウントのパスワードをリセットし、password_expired を 'N' に設定します。

初期実装には、ALTER USER を使用すると mysql.user テーブル内の Password カラムが空の文字列に設定されるというバグがあります。つまり、このステートメントを使用するには、ユーザーは MySQL 5.6.7 まで待つべきです。

MySQL 5.6.7:

Password カラムが空の文字列に設定されないように、ALTER USER が固定されました。

MySQL 5.6.8:

ALTER USER は、準備されたステートメントとして使用できます。

mysqladmin password は、期限切れのネイティブまたは古いネイティブのパスワードを持つアカウントのパスワードを設定できるようになりました。

SET PASSWORD に加えて、クライアントが SET ステートメントを実行することが許可されるように、サンドボックスモードが変更されました。SET の禁止により、old_passwords を設定する必要があるクライアントがパスワードをリセットできませんでした。また、これにより、接続時に SET を広範囲に使用してセッション環境を初期化する一部のコネクタも使用できませんでした。

MySQL 5.6.9:

ステートメントで指定されたアカウントとクライアントが認証したアカウントが一致する場合にのみ、SET PASSWORD が許可されるように、サンドボックスモードが変更されました。

MySQL 5.6.10:

期限切れパスワードを持つアカウントのクライアント接続をサーバーが処理する方法をより適切に制御することが許可され、クライアントが期限切れパスワードを処理できるどうかを示すことが許可されるように、サンドボックスモードが変更されました。

- サーバーが期限切れパスワードを持つアカウントを処理する方法を制御する disconnect_on_expired_password システム変数が追加されました。
- C API クライアントライブラリに、mysql_options() 用の MYSQL_OPT_CAN_HANDLE_EXPIRED_PASSWORDS と mysql_real_connect() 用の CLIENT_CAN_HANDLE_EXPIRED_PASSWORDS の 2 つのフラグが追加されました。各フラグを使用すると、クライアントプログラムが期限切れパスワードを持つアカウントに対してサンドボックスモードを処理できるかどうかを示すことができます。

  MYSQL_OPT_CAN_HANDLE_EXPIRED_PASSWORDS が mysqltest では無条件に、mysql ではインタラクティブモードで、mysqladmin では最初のコマンドが password の場合に有効になりました。
- ER_MUST_CHANGE_PASSWORD_LOGIN エラーが追加されました。クライアントが切断されると、サーバーはこのエラーを返します。

MySQL サーバーと C API クライアントライブラリにおけるサンドボックスモードへのこれらの変更に合わせて、コネクタを変更に適合させるための修正の取り組みも始まっています。

6.3.7 プラガブル認証

クライアントが MySQL サーバーに接続すると、サーバーはクライアントおよびクライアントホストで提供されたユーザー名を使用して、mysql.user テーブルから適切なアカウント行を選択します。そのあと、サーバーは認証プラグインでクライアントに適用されるアカウント行から決定することで、クライアントを認証します。

- アカウント行にプラグインが指定されている場合、サーバーはそれを呼び出すことで、ユーザーを認証します。サーバーがプラグインを検出できない場合は、エラーが発生します。
- アカウント行にプラグイン名が指定されていない場合、Password カラム内のパスワードハッシュ値でネイティブのハッシュ化方式と、4.1 よりも前の古いハッシュ化方式のどちらが使用されているのかに応じて、サーバーは mysql_native_password と mysql_old_password のいずれかのプラグインを使用してアカウントを認証します。クライアントは、アカウント行の Password カラム内のパスワードと一致する必要があります。

ユーザーが接続を許可されているかどうかを示すステータスがプラグインからサーバーに返されます。

プラガブル認証を使用すると、次の 2 つの重要な機能が有効になります。

- 外部認証: プラガブル認証を使用すると、mysql.user テーブルに格納されているパスワードに基づいたネイティブ認証以外の認証方式に適した証明書を使用して、クライアントが MySQL サーバーに接続できます。たとえば、PAM、Windows のログイン ID、LDAP、Kerberos などの外部認証方式を使用するプラグインを作成できます。

- プロキシユーザー: ユーザーが接続を許可されている場合、接続中のユーザーが別のユーザーのプロキシであることを示すために、認証プラグインは接続中のユーザーの名前と異なるユーザー名をサーバーに返すことができます。接続が存続している間は、アクセス制御のために、プロキシユーザーは別のユーザーの権限を持っているものとして処理されます。実際に、あるユーザーは別のユーザーを偽装します。詳細については、セクション6.3.9「プロキシユーザー」を参照してください。

MySQL では、複数の認証プラグインが使用可能です。

- アカウント行の Password カラムと照合してパスワードを一致させるネイティブ認証を実行するプラグイン。mysql_native_password プラグインには、ネイティブのパスワードハッシュ化方式に基づいた認証が実装されています。mysql_old_password プラグインには、古い (4.1 よりも前の) パスワードハッシュ化方式 (現在は非推奨です) に基づいたネイティブ認証が実装されています。セクション6.3.8.1「ネイティブ認証プラグイン」およびセクション6.3.8.2「古いネイティブ認証プラグイン」を参照してください。サーバーの起動時に --default-authentication-plugin オプションが設定されている場合を除き、mysql_native_password を使用したネイティブ認証が新しいアカウントのデフォルトです。

- SHA-256 のパスワードハッシュ化を使用して認証を実行するプラグイン。このプラグインは、アカウント行の authentication_string カラムと照合してパスワードを一致させます。これは、ネイティブ認証で実現できるよりも強力な暗号化です。セクション6.3.8.4「SHA-256 認証プラグイン」を参照してください。

- PAM (Pluggable Authentication Module) と照合して外部認証を実行するプラグイン。これを使用すると、MySQL サーバーが PAM を使用して MySQL ユーザーを認証できます。このプラグインでは、プロキシユーザーもサポートされています。セクション6.3.8.5「PAM 認証プラグイン」を参照してください。

- Windows で外部認証を実行するプラグイン。これを使用すると、MySQL サーバーがネイティブの Windows サービスを使用して、クライアント接続を認証できます。Windows にログインしたユーザーは、追加のパスワードを指定せずに、自分の環境内の情報に基づいて MySQL クライアントプログラムからサーバーに接続できます。このプラグインでは、プロキシユーザーもサポートされています。セクション6.3.8.6「Windows ネイティブ認証プラグイン」を参照してください。

- ハッシュ化または暗号化を行わずに、サーバーにパスワードを送信するクライアント側のプラグイン。このプラグインは、クライアントユーザーから提供されたものとまったく同じパスワードにアクセスする必要があるサーバー側のプラグインで使用できます。セクション6.3.8.7「クライアント側のクリアテキスト認証プラグイン」を参照してください。

- Unix ソケットファイルを使用してローカルホストから接続するクライアントを認証するプラグイン。セクション6.3.8.8「ソケットピア証明書認証プラグイン」を参照してください。

- MySQL のネイティブ認証を使用して認証するテストプラグイン。このプラグインは、テストおよび開発のために、認証プラグインを作成する方法を示す例として使用されます。セクション6.3.8.9「テスト認証プラグイン」を参照してください。

> **注記**
>
> プラガブル認証の使用に対する現在の制約 (どのコネクタがどのプラグインをサポートしているのかなど) については、セクションD.9「プラガブルな認証の制約」を参照してください。
>
> サードパーティー製コネクタの開発者は、コネクタがプラガブル認証機能を活用できる範囲と、より準拠させるために実行する手順を確認するために、そのセクションを読むべきです。

独自の認証プラグインを作成することに関心がある場合は、セクション24.2.4.9「認証プラグインの作成」を参照してください。

認証プラグインの使用手順

このセクションでは、認証プラグインをインストールおよび使用するための一般的な手順を示します。

通常、プラガブル認証では、サーバー側とクライアント側で対応するプラグインが使用されます。そのため、次のような特定の認証方式を使用します。

- サーバーホストでは、必要に応じて、適切なサーバープラグインを含むライブラリをインストールします。これにより、サーバーはこれを使用して、クライアント接続を認証できます。同様に、各クライアントホストでは、クライアントプログラムで使用される適切なクライアントプラグインを含むライブラリをインストールします。
- 認証用のプラグインの使用を指定する MySQL アカウントを作成します。
- クライアントが接続すると、サーバープラグインからクライアントプログラムに、認証に使用されるクライアントプラグインが指示されます。

ここで示す指示では、MySQL 配布に含まれるサンプルの認証プラグインが使用されています (セクション 6.3.8.9「テスト認証プラグイン」を参照してください)。この手順は、その他の認証プラグインに似ています。適切なプラグインとファイル名に置き換えてください。

サンプルの認証プラグインには、次のような特性があります。

- サーバー側のプラグイン名は test_plugin_server です。
- クライアント側のプラグイン名は auth_test_plugin です。
- どちらのプラグインも、プラグインディレクトリ (plugin_dir システム変数で指定されたディレクトリ) 内の auth_test_plugin.so という名前の共有ライブラリオブジェクトファイルに配置されています。ファイル名のサフィクスは、システムによって異なる場合があります。

次のように、サンプルの認証プラグインをインストールして使用します。

1. サーバーホストおよびクライアントホスト上に、プラグインライブラリがインストールされていることを確認します。
2. サーバーの起動時または実行時に、サーバー側のテストプラグインをインストールします。
   - 起動時にプラグインをインストールするには、--plugin-load オプションを使用します。このプラグインのロード方式では、サーバーを起動するたびにオプションを指定する必要があります。たとえば、my.cnf オプションファイル内で次の行を使用します。

     ```
     [mysqld]
     plugin-load=test_plugin_server=auth_test_plugin.so
     ```

   - 実行時にプラグインをインストールするには、INSTALL PLUGIN ステートメントを使用します。

     ```
     mysql> INSTALL PLUGIN test_plugin_server SONAME 'auth_test_plugin.so';
     ```

     これにより、プラグインが永続的にインストールされるため、1 回しか実行する必要がありません。
3. プラグインがインストールされていることを確認します。たとえば、SHOW PLUGINS を使用します。

   ```
   mysql> SHOW PLUGINS\G
   ...
   *************************** 21. row ***************************
     Name: test_plugin_server
   Status: ACTIVE
     Type: AUTHENTICATION
   Library: auth_test_plugin.so
   License: GPL
   ```

   プラグインをチェックするためのその他の方法については、セクション5.1.8.2「サーバープラグイン情報の取得」を参照してください。
4. 特定のサーバープラグインを使用して MySQL ユーザーを認証する必要があることを指定するには、ユーザーを作成する CREATE USER ステートメントの IDENTIFIED WITH 句にプラグインの名前を指定します。

   ```
   CREATE USER 'testuser'@'localhost' IDENTIFIED WITH test_plugin_server;
   ```

5. クライアントプログラムを使用して、サーバーに接続します。テストプラグインは、ネイティブの MySQL 認証と同じ方法で認証を実行します。したがって、サーバーに接続する際に一般的に使用される通常の --user および --password オプションを指定します。例:

   ```
   shell> mysql --user=your_name --password=your_pass
   ```

   testuser による接続の場合、サーバーは test_plugin_server という名前のサーバー側のプラグインを使用してアカウントを認証する必要があることを認識し、クライアント側のプラグイン (この場合、auth_test_plugin) を使用する必要のあるクライアントプログラムと通信します。

アカウントがサーバープログラムとクライアントプログラムの両方のデフォルトである認証方式を使用する場合、サーバーはプラグインを使用するクライアントと通信する必要がなく、クライアントとサーバーのネゴシエーションでラウンドトリップが発生することを回避できます。現在、これはネイティブの MySQL 認証 (mysql_native_password) を使用するアカウントに適用されます。

mysql コマンド行で --default-auth=plugin_name オプションを指定すると、プログラムで使用するように要求できるクライアント側のプラグインを明示的に作成できます。ただし、ユーザーアカウントが別のプラグインを要求した場合、これはサーバーによってオーバーライドされます。

クライアントプログラムでプラグインが見つからない場合は、プラグインが配置されている場所を示す --plugin-dir=dir_name オプションを指定します。

> 注記
>
> --skip-grant-tables オプションを付けてサーバーを起動した場合、サーバーはクライアント認証を実行せず、任意のクライアントが接続することを許可するため、認証プラグインはロードされたとしても使用されません。これはセキュアではないため、リモートクライアントによる接続を回避するために、--skip-networking と組み合わせて --skip-grant-tables を使用する必要がある場合もあります。

6.3.8 MySQL で使用可能な認証プラグイン

次のセクションでは、MySQL で使用可能な認証プラグインについて説明します。

6.3.8.1 ネイティブ認証プラグイン

MySQL には、ネイティブ認証 (つまり、mysql.user テーブルの Password カラムに格納されたパスワードと照合する認証) を実装する 2 つのプラグインが含まれています。このセクションでは、ネイティブのパスワードハッシュ化方式を使用して、mysql.user テーブルと照合する認証が実装された mysql_native_password について説明します。古い (4.1 よりも前の) パスワードハッシュ化方式を使用した認証が実装された mysql_old_password については、セクション6.3.8.2「古いネイティブ認証プラグイン」を参照してください。これらのパスワードハッシュ化方式については、セクション6.1.2.4「MySQL でのパスワードハッシュ」を参照してください。

mysql_native_password ネイティブ認証プラグインには、下位互換性があります。MySQL 5.5.7 よりも古いクライアントでは、認証プラグインがサポートされていませんが、ネイティブ認証プロトコルは使用されます。そのため、MySQL 5.5.7 以上からサーバーに接続できます。

次の表に、サーバー側とクライアント側のプラグイン名を示します。

表 6.8 MySQL ネイティブパスワード認証プラグイン

サーバー側のプラグイン名	mysql_native_password
クライアント側のプラグイン名	mysql_native_password
ライブラリオブジェクトファイル名	なし (プラグインは組み込み済み)

プラグインは、クライアントとサーバーの両方の形式で存在します。

- サーバー側のプラグインはサーバーに組み込まれているため、明示的にロードする必要はなく、アンロードしても無効にすることができません。
- クライアント側のプラグインは、MySQL 5.5.7 の時点で libmysqlclient クライアントライブラリに組み込まれ、そのバージョン以降では libmysqlclient に対してリンクされている任意のプログラムで使用可能です。
- MySQL クライアントプログラムでは、デフォルトで mysql_native_password が使用されます。--default-auth オプションを使用すると、プラグインを明示的に指定できます。

```
shell> mysql --default-auth=mysql_native_password ...
```

アカウント行にプラグイン名が指定されていない場合、Password カラム内のパスワードハッシュ値でネイティブのハッシュ化方式と、4.1 よりも前の古いハッシュ化方式のどちらが使用されるのかに応じて、サーバーは mysql_native_password と mysql_old_password のいずれかのプラグインを使用してアカウントを認証します。クライアントは、アカウント行の Password カラム内のパスワードと一致する必要があります。

MySQL のプラガブル認証に関する一般的な情報については、セクション6.3.7「プラガブル認証」を参照してください。

6.3.8.2 古いネイティブ認証プラグイン

MySQL には、ネイティブ認証 (つまり、mysql.user テーブルの Password カラムに格納されたパスワードと照合する認証) を実装する 2 つのプラグインが含まれています。このセクションでは、古い (4.1 よりも前の) パスワードハッシュ化方式を使用して、mysql.user テーブルと照合する認証が実装された mysql_old_password について説明します。ネイティブのパスワードハッシュ化方式を使用した認証が実装された mysql_native_password については、セクション6.3.8.1「ネイティブ認証プラグイン」を参照してください。これらのパスワードハッシュ化方式については、セクション6.1.2.4「MySQL でのパスワードハッシュ」を参照してください。

> **注記**
>
> 4.1 よりも前のハッシュ化方式を使用したパスワードは、ネイティブのパスワードハッシュ化方式を使用したパスワードよりもセキュアではありません。4.1 よりも前のパスワードは非推奨であり、これらのサポートは今後の MySQL リリースで削除される予定です。アカウントのアップグレード手順については、セクション6.3.8.3「4.1 よりも前のパスワードハッシュ方式と mysql_old_password プラグインからの移行」を参照してください。

mysql_old_password ネイティブ認証プラグインには、下位互換性があります。MySQL 5.5.7 よりも古いクライアントでは、認証プラグインがサポートされていませんが、ネイティブ認証プロトコルは使用されます。そのため、MySQL 5.5.7 以上からサーバーに接続できます。

次の表に、サーバー側とクライアント側のプラグイン名を示します。

表 6.9 MySQL の古いネイティブ認証プラグイン

サーバー側のプラグイン名	mysql_old_password
クライアント側のプラグイン名	mysql_old_password
ライブラリオブジェクトファイル名	なし (プラグインは組み込み済み)

プラグインは、クライアントとサーバーの両方の形式で存在します。

- サーバー側のプラグインはサーバーに組み込まれているため、明示的にロードする必要はなく、アンロードしても無効にすることができません。
- クライアント側のプラグインは、MySQL 5.5.7 の時点で libmysqlclient クライアントライブラリに組み込まれ、そのバージョン以降では libmysqlclient に対してリンクされている任意のプログラムで使用可能です。
- MySQL クライアントプログラムは、--default-auth オプションを使用して、mysql_old_password プラグインを明示的に指定できます。

```
shell> mysql --default-auth=mysql_old_password ...
```

アカウント行にプラグイン名が指定されていない場合、Password カラム内のパスワードハッシュ値でネイティブのハッシュ化方式と、4.1 よりも前の古いハッシュ化方式のどちらが使用されるのかに応じて、サーバーは mysql_native_password と mysql_old_password のいずれかのプラグインを使用してアカウントを認証します。クライアントは、アカウント行の Password カラム内のパスワードと一致する必要があります。

MySQL のプラガブル認証に関する一般的な情報については、セクション6.3.7「プラガブル認証」を参照してください。

6.3.8.3 4.1 よりも前のパスワードハッシュ方式と mysql_old_password プラグインからの移行

MySQL サーバーは、plugin カラムで名前が指定された認証プラグインを使用して、mysql.user テーブルに一覧表示されたアカウントごとに接続の試行を認証します。plugin カラムが空である場合、サーバーは次のように認証を実行します。

- MySQL 5.7.2 よりも前では、Password カラム内のパスワードハッシュ形式に応じて、サーバーは暗黙的に mysql_native_password または mysql_old_password プラグインを使用します。Password 値が空または 4.1 のパスワードハッシュ (41 文字) である場合、サーバーは mysql_native_password を使用します。パスワード値が 4.1 より前のパスワードハッシュ (16 文字) の場合、サーバーは mysql_old_password を使用します。(これらの

ハッシュ形式に関する追加情報については、セクション6.1.2.4「MySQL でのパスワードハッシュ」を参照してください。)

- MySQL 5.7.2 の時点では、plugin カラムを空以外にする必要があり、サーバーは空の plugin 値を持つアカウントを無効にします。5.7.5 の時点では、mysql_old_password のサポートが削除されています。

4.1 よりも前のパスワードハッシュおよび mysql_old_password プラグインは、MySQL 5.6.5 の時点で非推奨となり、4.1 のパスワードハッシュ化および mysql_native_password プラグインによって提供されるよりも低いセキュリティーレベルを提供します。plugin カラムを空以外にする必要のある MySQL 5.7.2 の要件と、5.7.5 で mysql_old_password のサポートが削除されたことを合わせて考えると、DBA は次のようにアカウントをアップグレードすることをお勧めします。

- mysql_native_password を明示的に使用するために、それを暗黙的に使用しているアカウントをアップグレードします。
- mysql_native_password を明示的に使用するために、(暗黙的または明示的に) mysql_old_password を使用しているアカウントをアップグレードします。

このセクションの手順では、これらのアップグレードを実行する方法について説明します。その結果、アカウントは空の plugin 値を持たず、4.1 よりも前のパスワードハッシュ化または mysql_old_password プラグインを使用しません。

これらの手順のバリエーションとして、DBA は、SHA-256 パスワードハッシュを使用して認証する sha256_password プラグインにアップグレードするという選択肢をユーザーに提示することがあります。このプラグインについては、セクション6.3.8.4「SHA-256 認証プラグイン」を参照してください。

次の表には、この説明で検討される mysql.user アカウントのタイプを一覧表示します。

plugin カラム	Password カラム	認証結果	アップグレードアクション
空	空	暗黙的に mysql_native_password を使用する	プラグインの割り当て
空	4.1 のハッシュ	暗黙的に mysql_native_password を使用する	プラグインの割り当て
空	4.1 よりも前のハッシュ	暗黙的に mysql_old_password を使用する	プラグインを割り当て、パスワードを再ハッシュ化する
mysql_native_password	空	明示的に mysql_native_password を使用する	なし
mysql_native_password	4.1 のハッシュ	明示的に mysql_native_password を使用する	なし
mysql_old_password	空	明示的に mysql_old_password を使用する	プラグインのアップグレード
mysql_old_password	4.1 よりも前のハッシュ	明示的に mysql_old_password を使用する	プラグインをアップグレードし、パスワードを再ハッシュ化する

mysql_native_password プラグインの行に対応するアカウントでは、(プラグインやハッシュ形式を変更する必要がないため) アップグレードアクションは必要ありません。パスワードが空の行に対応するアカウントでは、DBA は、パスワードを選択するようにアカウントの所有者に要求することを検討するべきです (または、ALTER USER を使用して空のアカウントパスワードを期限切れにすることで、それを要求するべきです)。

mysql_native_password の暗黙的な使用から明示的な使用へのアップグレード

空のプラグインと 4.1 のパスワードハッシュを持つアカウントは、暗黙的に mysql_native_password を使用します。mysql_native_password を明示的に使用するようにアカウントをアップグレードするには、DBA は次のステートメントを実行するようにしてください。

```
UPDATE mysql.user SET plugin = 'mysql_native_password'
WHERE plugin = '' AND (Password = '' OR LENGTH(Password) = 41);
FLUSH PRIVILEGES;
```

MySQL 5.7.2 以降では、DBA は、そのアップグレードアクション間で同じ処理を行う mysql_upgrade を実行できます。5.7.2 よりも前では、DBA はこれらのステートメントを実行して、アカウントをプロアクティブにアップグレードできます。

注:

- この手順は、すでに暗黙的に使用しているアカウントに対してのみ、mysql_native_password プラグインを明示的にするため、いつ実行しても安全です。
- この手順ではパスワードを変更する必要がないため、DBA はユーザーに影響を与えたり、アップグレードプロセスに関与するようにユーザーに要求したりせずに、このアクションを実行できます。

mysql_old_password から mysql_native_password へのアカウントのアップグレード

mysql_native_password を明示的に使用するには、(暗黙的または明示的に) mysql_old_password を使用しているアカウントをアップグレードするようにしてください。これを行うには、プラグインを変更かつ4.1 よりも前から 4.1 のハッシュ形式にパスワードを変更する必要があります。

この手順で説明したアカウントをアップグレードする必要がある場合は、次の条件のいずれかに当てはまります。

- plugin カラムが空であり、パスワードハッシュ形式が 4.1 よりも前 (16 文字) であるため、アカウントは暗黙的に mysql_old_password を使用します。
- アカウントは明示的に mysql_old_password を使用します。

このようなアカウント識別するには、次のクエリーを使用します。

```
SELECT User, Host, Password FROM mysql.user
WHERE (plugin = '' AND LENGTH(Password) = 16)
OR plugin = 'mysql_old_password';
```

次の説明では、そのアカウントセットを更新するための 2 つの方法を示します。これらは異なる特性を持っているため、DBA は両方とも読み、特定の MySQL インストールに最適な方を決定するようにしてください。

方法 1.

この方法の特性:

- すべてのユーザーが mysql_native_password にアップグレードされるまで、secure_auth=0 を指定してサーバーおよびクライアントを実行する必要があります。(それ以外の場合、ユーザーは新しい形式のハッシュにアップグレードするために、古い形式のパスワードハッシュを使用してサーバーに接続できません。)
- MySQL 5.5 から 5.7.1 までで、動作します。5.7.2 の時点では、サーバーは空でないプラグインを持つようにアカウントに要求し、そうでない場合はそれらのアカウントを無効にするため、動作しません。したがって、5.7.2 以降にアップグレードした場合は、方法 2 を選択してください。

DBA は、サーバーが secure_auth=0 を指定して実行されていることを確認するようにしてください。

明示的に mysql_old_password を使用するすべてのアカウントでは、DBA は空のプラグインに設定するようにしてください。

```
UPDATE mysql.user SET plugin = ''
WHERE plugin = 'mysql_old_password';
FLUSH PRIVILEGES;
```

また、影響を受けるアカウントのパスワードを期限切れにするには、次のステートメントを使用します。

```
UPDATE mysql.user SET plugin = '', password_expired = 'Y'
WHERE plugin = 'mysql_old_password';
FLUSH PRIVILEGES;
```

この時点で、影響を受けるユーザーはサーバーに接続し、4.1 のハッシュ化が使用されるようにパスワードをリセットできます。DBA は空のプラグインを現在持っている各ユーザーに、接続して次のステートメントを実行するように要求するようにしてください。

```
SET old_passwords = 0;
SET PASSWORD = PASSWORD('user-chosen-password');
```

> **注記**
>
> MySQL 5.6.5 以降では、クライアント側の --secure-auth がデフォルトで有効になっているため、DBA はこれを無効にするようにユーザーに思い出させるようにしてください。そうしなければ、ユーザーは接続できなくなります。
>
> ```
> shell> mysql -u user_name -p --secure-auth=0
> ```

影響を受けるユーザーがこれらのステートメントを実行したあとに、DBA は対応するアカウントプラグインを mysql_native_password に設定すると、プラグインを明示的にすることができます。また、DBA は次のステートメントを定期的に実行すると、影響を受けるユーザーが自分のパスワードをリセットした任意のアカウントを検索および修正できます。

```
UPDATE mysql.user SET plugin = 'mysql_native_password'
WHERE plugin = '' AND (Password = '' OR LENGTH(Password) = 41);
FLUSH PRIVILEGES;
```

空のプラグインを持つアカウントがなくなると、このクエリーは空の結果を返します。

```
SELECT User, Host, Password FROM mysql.user
WHERE (plugin = '' AND LENGTH(Password) = 16);
```

この時点で、すべてのアカウントは 4.1 よりも前のパスワードハッシュ化から移行され、secure_auth=0 を指定してサーバーを実行する必要がなくなります。

方法 2.

この方法の特性:

- DBA は、影響を受ける各アカウントに新しいパスワードを割り当てます。そのため、DBA はこのような各ユーザーに新しいパスワードを通知し、新しいパスワードを選択するようにユーザーに要求する必要があります。DBA からユーザーへのパスワードの通知は、MySQL のスコープ外です。DBA はパスワードを慎重に通知するようにしてください。
- secure_auth=0 を指定してサーバーまたはクライアントを実行する必要はありません。
- MySQL 5.5 以降のどのバージョンでも動作します。

この方法では、パスワードを個別に設定する必要があるため、DBA は各アカウントを別々に更新します。DBA は、アカウントごとに別々のパスワードを選択するようにしてください。

'user1'@'localhost' がアップグレードされるアカウントの 1 つであると仮定します。DBA は、次のように変更するようにしてください。

```
SET old_passwords = 0;
UPDATE mysql.user SET plugin = 'mysql_native_password',
Password = PASSWORD('DBA-chosen-password')
WHERE (User, Host) = ('user1', 'localhost');
FLUSH PRIVILEGES;
```

また、パスワードを期限切れにするには、代わりに次のステートメントを使用します。

```
SET old_passwords = 0;
UPDATE mysql.user SET plugin = 'mysql_native_password',
Password = PASSWORD('DBA-chosen-password'), password_expired = 'Y'
WHERE (User, Host) = ('user1', 'localhost');
FLUSH PRIVILEGES;
```

次に DBA は、ユーザーに新しいパスワードを通知して、そのパスワードを使用してサーバーに接続し、新しいパスワードを選択するために次のステートメントを実行するようにユーザーに要求するようにしてください。

```
SET old_passwords = 0;
SET PASSWORD = PASSWORD('user-chosen-password');
```

アップグレードするアカウントごとに繰り返します。

6.3.8.4 SHA-256 認証プラグイン

MySQL 5.6.6 の時点で、MySQL では、ユーザーアカウントのパスワード用に SHA-256 ハッシュ化が実装された認証プラグインが提供されています。

> **重要**
>
> sha256_password プラグインで認証するアカウントを使用してサーバーに接続するには、このセクションの後半で説明するように、SSL 接続または、RSA を使用してパスワードを暗号化する単純な接続を使用する必要があります。どちらの方法でも、sha256_password プラグインを使用するには、SSL の機能を使用して MySQL を構築する必要があります。セクション6.3.10「セキュアな接続のための SSL の使用」を参照してください。

次の表に、サーバー側とクライアント側のプラグイン名を示します。

表 6.10 MySQL SHA-256 認証プラグイン

サーバー側のプラグイン名	sha256_password
クライアント側のプラグイン名	sha256_password
ライブラリオブジェクトファイル名	なし (プラグインは組み込み済み)

サーバー側の sha256_password プラグインはサーバーに組み込まれているため、明示的にロードする必要はなく、アンロードしても無効にすることはできません。同様に、クライアントはクライアント側のプラグインの場所を指定する必要がありません。

SHA-256 パスワードのハッシュ化を使用するアカウントを設定するには、次の手順を使用します。

1. アカウントを作成し、sha256_password プラグインを使用して認証するように指定します。

```
CREATE USER 'sha256user'@'localhost' IDENTIFIED WITH sha256_password;
```

2. PASSWORD () 関数でパスワード文字列の SHA-256 ハッシュ化が使用されるように、old_passwords システム変数を 2 に設定してから、アカウントのパスワードを設定します。

```
SET old_passwords = 2;
SET PASSWORD FOR 'sha256user'@'localhost' = PASSWORD('Sh@256Pa33');
```

または、sha256_password に設定されたデフォルトの認証プラグインを使用して、サーバーを起動します。たとえば、サーバーのオプションファイルに次の行を挿入します。

```
[mysqld]
default-authentication-plugin=sha256_password
```

これにより、新しいアカウント用に sha256_password プラグインがデフォルトで使用され、old_passwords が 2 に設定されます。その結果、アカウント作成時に CREATE USER ステートメントで IDENTIFIED BY 句を使用して、パスワードを設定できます。

```
mysql> CREATE USER 'sha256user2'@'localhost' IDENTIFIED BY 'Sh@256Pa33';
Query OK, 0 rows affected (0.06 sec)
```

この場合、サーバーは sha256_password プラグインをアカウントに割り当て、SHA-256 を使用してパスワードを暗号化します。(もう 1 つの結果として、sha256_password とは異なる認証プラグインを使用するアカウントを作成するには、CREATE USER ステートメントで IDENTIFIED BY 句を使用して、そのプラグインを指定し、プラグインに old_passwords を適切に設定してから、SET PASSWORD を使用してアカウントのパスワードを設定する必要があります。)

old_passwords および PASSWORD() についての詳細は、セクション5.1.4「サーバーシステム変数」およびセクション12.13「暗号化関数と圧縮関数」を参照してください。

sha256_password プラグインを使用して認証する任意のアカウントのパスワードを変更するには、SET PASSWORD を使用する前に、old_passwords の値が 2 になっていることを確認します。old_passwords の値が 2 以外になっている場合は、パスワードを設定しようとすると エラーが発生します。

```
mysql> SET old_passwords = 0;
mysql> SET PASSWORD FOR 'sha256user'@'localhost' = PASSWORD('NewSh@256Pa33');
ERROR 1827 (HY000): The password hash doesn't have the expected format.
Check if the correct password algorithm is being used with the
PASSWORD() function.
```

SHA-256 のパスワードを使用する mysql.user テーブル内のアカウントは、plugin カラム内の 'sha256_password' および authentication_string カラム内の SHA-256 パスワードハッシュを含む行として識別できます。

MySQL は yaSSL と OpenSSL のいずれかを使用して構築でき、sha256_password プラグインは、いずれかのパッケージを使用して構築された配布で動作します。デフォルトは、yaSSL を使用する方法です。代わりに OpenSSL を使用して MySQL が構築されている場合は、RSA 暗号化が使用可能であり、次のリストに示す追加の機能が sha256_password に実装されます。(これらの機能を有効にするには、このセクションの後半で示す RSA の構成手順に従う必要もあります。)

- あとで説明するように、クライアント接続プロセス中に RSA 暗号化を使用すると、クライアントがサーバーにパスワードを送信できます。
- サーバーは sha256_password_private_key_path と sha256_password_public_key_path の 2 つの追加のシステム変数を公開します。これは、サーバーの起動時にデータベース管理者が、これらを RSA 秘密鍵と公開鍵ファイルの名前に設定するために使用されます。
- サーバーは、RSA 公開鍵の値を示すステータス変数 Rsa_public_key を公開します。
- mysql および mysqltest クライアントプログラムでは、RSA 公開鍵ファイルを明示的に指定するための --server-public-key-path オプションがサポートされています。(このオプションは、--server-public-key という名前で MySQL 5.6.6 に追加されましたが、5.6.7 で --server-public-key-path という名前に変更されました。)

sha256_password プラグインを使用するクライアントでは、サーバーへの接続時にパスワードがクリアテキストとして公開されません。パスワードの送信がどのように発生するのかは、SSL 接続が使用されるのかどうか、および RSA 暗号化が使用可能であるかどうかによって異なります。

- SSL 接続が使用される場合、パスワードはクリアテキストとして送信されますが、接続は SSL を使用して暗号化されるため、覗き見られる可能性はありません。
- SSL 接続は使用されないが、RSA 暗号化が使用可能である場合、パスワードは暗号化されていない接続内に送信されます。ただし、パスワードは覗き見られないように RSA で暗号化されます。サーバーはパスワードを受信すると、それを復号化します。繰り返し攻撃を防ぐために、スクランブルが暗号化で使用されます。
- SSL 接続が使用されず、RSA 暗号化が使用可能でない場合、パスワードはクリアテキストとして公開されずに送信できないため、sha256_password プラグインによる接続の試行に失敗します。

すでに説明したように、RSA のパスワード暗号化は、OpenSSL を使用して MySQL が構築された場合にのみ使用可能です。yaSSL を使用して MySQL 配布を構築したということは、クライアントが SSL 接続を使用してサーバーにアクセスする際にのみ、SHA-256 のパスワードを使用できることを意味します。SSL を使用したサーバーへの接続については、セクション6.3.10「セキュアな接続のための SSL の使用」を参照してください。

次の手順では、MySQL が OpenSSL を使用して構築されたと仮定して、クライアント接続プロセス中にパスワードの RSA 暗号化を有効にする方法について説明します。

1. RSA の秘密鍵および公開鍵ファイルを作成します。MySQL サーバーを実行する際に使用されるシステムアカウントにログインしている間に、次のコマンドを実行します。これにより、ファイルはそのアカウントによって所有されます。

   ```
   openssl genrsa -out mykey.pem 1024
   openssl rsa -in mykey.pem -pubout -out mykey.pub
   ```

   これらのコマンドでは、1,024 ビットの鍵が作成されます。より強固な鍵を作成するには、2,048 のような大きな値を使用します。

2. 鍵ファイルのアクセスモードを設定します。秘密鍵は、サーバーからのみ読み取り可能にするべきです。一方で、公開鍵は、クライアントユーザーに自由に配布できます。

   ```
   chmod 400 mykey.pem
   chmod 444 mykey.pub
   ```

3. サーバーのオプションファイルで、鍵ファイルの名前を使用して適切なシステム変数を構成します。サーバーのデータディレクトリにファイルを配置する場合は、完全パス名を指定する必要がありません。

   ```
   [mysqld]
   sha256_password_private_key_path=mykey.pem
   sha256_password_public_key_path=mykey.pub
   ```

   ファイルがデータディレクトリ内にない場合や、システム変数の値でそれらの場所を明示的に指定する場合は、完全パス名を使用します。

```
[mysqld]
sha256_password_private_key_path=/usr/local/mysql/mykey.pem
sha256_password_public_key_path=/usr/local/mysql/mykey.pub
```

4. サーバーを再起動してから、それに接続し、Rsa_public_key ステータス変数の値をチェックします。値はここで示すものとは異なりますが、空以外を指定するようにしてください。

```
mysql> SHOW STATUS LIKE 'Rsa_public_key'\G
*************************** 1. row ***************************
Variable_name: Rsa_public_key
        Value: -----BEGIN PUBLIC KEY-----
MIGfMA0GCSqGSIb3DQEBAQUAA4GNADCBiQKBgQDO9nRUDd+KvSZgY7cNBZMNpwX6
MvE1PbJFXO7u18nJ9lwc99Du/E7lw6CVXw7VKrXPeHbVQUzGyUNkf45Nz/ckaaJa
aLgJOBCIDmNVnyU54OT/1lcs2xiyfaDMe8fCJ64ZwTnKbY2gkt1IMjUAB5Ogd5kJ
g8aV7EtKwyhHb0c30QIDAQAB
-----END PUBLIC KEY-----
```

   値が空の場合は、鍵ファイルに関するいくつかの問題がサーバーで見つかっています。エラーログをチェックして、診断情報を確認してください。

サーバーに RSA 鍵ファイルが構成されると、クライアントはそれらのファイルを使用して、sha256_password プラグインで認証するアカウントを使用したサーバーに接続できます。すでに説明したように、このようなアカウントは SSL 接続 (この場合、RSA が使用されません)、または RSA を使用してパスワードを暗号化する単純な接続を使用できます。次の説明は、SSL が使用されないことが前提となっています。サーバーに接続するために、クライアント側で特別な準備をする必要はありません。例:

```
shell> mysql -u sha256user -p
Enter password: Sh@256Pa33
```

sha256user によって接続を試みる場合、サーバーは sha256_password が適切な認証プラグインであると判断し、それを呼び出します。接続で SSL が使用されないために、RSA 暗号化を使用してパスワードを送信する必要があることがプラグインによって検出されます。RSA 公開鍵がクライアントに送信され、この鍵を使用してパスワードが暗号化され、結果がサーバーに返されます。このプラグインは、サーバー側の RSA 鍵を使用してパスワードを復号化し、パスワードが正しいかどうかに基づいて接続を承認または拒否します。

サーバーは必要に応じて公開鍵をクライアントに送信しますが、クライアントホスト上で RSA 公開鍵のコピーが使用可能である場合は、クライアントはその鍵を使用して、クライアント/サーバープロトコルにラウンドトリップを保存できます。

```
shell> mysql -u sha256user -p --server-public-key-path=file_name
```

--server-public-key-path オプションで指定されたファイル内の公開鍵値は、sha256_password_public_key_path システム変数で指定されたサーバー側のファイル内の鍵値と同じにするようにしてください。鍵ファイルに有効な公開鍵値が含まれているが、その値が正しくない場合は、アクセス拒否のエラーが発生します。鍵ファイルに有効な公開鍵が含まれていない場合は、その鍵をクライアントプログラムで使用できません。この場合、サーバーは --server-public-key-path オプションが指定された場合と同様に、クライアントに公開鍵を送信します。

クライアントユーザーは、次の 2 つの方法で RSA 公開鍵を取得できます。

- データベース管理者は、公開鍵ファイルのコピーを提供できます。
- その他の方法でサーバーに接続できるクライアントユーザーは、SHOW STATUS LIKE 'Rsa_public_key' ステートメントを使用し、返された鍵値をファイル内に保存できます。

6.3.8.5 PAM 認証プラグイン

> **注記**
>
> PAM 認証プラグインは、商用の拡張機能です。商用の製品 (MySQL Enterprise Edition) についてさらに学習するには、https://www.mysql.com/products/ を参照してください。

MySQL 5.6.10 の時点で、MySQL の商用配布には、MySQL サーバーが PAM (Pluggable Authentication Module) を使用して MySQL ユーザーを認証できる認証プラグインが含まれています。PAM を使用すると、システムは標準インタフェースを使用して、さまざまな種類の認証方式 (Unix パスワードや LDAP ディレクトリなど) にアクセスできます。

PAM プラグインでは、MySQL サーバーによって渡される情報 (ユーザー名、ホスト名、パスワード、認証文字列など) に加えて、PAM 検索で使用可能な任意の方式が使用されます。このプラグインは、PAM と照合して

ユーザー資格情報をチェックし、「Authentication succeeded, Username is user_name」または「Authentication failed」を返します。

PAM 認証プラグインには、次のような機能が備わっています。

- 外部認証: このプラグインを使用すると、MySQL サーバーは、MySQL 付与テーブルの外部で定義されたユーザーからの接続を受け入れることができます。
- プロキシユーザーのサポート: このプラグインは、外部ユーザーが属するグループおよび指定された認証文字列に基づいて、ログインユーザーとは異なるユーザー名を MySQL に返すことができます。つまり、このプラグインは、外部 PAM で認証されたユーザーが持つべき権限を定義する MySQL ユーザーを返すことができます。たとえば、joe という名前の PAM ユーザーは接続して、developer という名前の MySQL ユーザーの権限を持つことができます。

次の表には、プラグインおよびライブラリファイルの名前を示します。ファイル名のサフィクスは、システムによって異なる場合があります。ファイルの場所は、plugin_dir システム変数で指定されたディレクトリである必要があります。インストールについては、PAM 認証プラグインのインストールを参照してください。

表 6.11 MySQL PAM 認証プラグイン

サーバー側のプラグイン名	authentication_pam
クライアント側のプラグイン名	mysql_clear_password
ライブラリオブジェクトファイル名	authentication_pam.so

ライブラリファイルには、サーバー側のプラグインのみが含まれています。クライアント側のプラグインは、libmysqlclient クライアントライブラリに組み込まれています。セクション6.3.8.7「クライアント側のクリアテキスト認証プラグイン」を参照してください。

サーバー側の PAM 認証プラグインは、商用配布にのみ含まれています。MySQL コミュニティー配布には含まれていません。サーバー側のプラグインと通信するクライアント側のクリアテキストプラグインは、MySQL クライアントライブラリに組み込まれ、コミュニティー配布を含むすべての配布に含まれています。これにより、5.6.10 以降の任意の配布から、サーバー側のプラグインがロードされたサーバーに接続することがクライアントに許可されます。

PAM 認証プラグインは、Linux および Mac OS X 上でテストされています。これには、MySQL Server 5.6.10 以降が必要です。

MySQL のプラガブル認証に関する一般的な情報については、セクション6.3.7「プラガブル認証」を参照してください。プロキシユーザーについては、セクション6.3.9「プロキシユーザー」を参照してください。

PAM 認証プラグインのインストール

PAM 認証プラグインは、MySQL プラグインディレクトリ (plugin_dir システム変数で指定されたディレクトリ) に配置する必要があります。必要に応じて、サーバーの起動時に、プラグインディレクトリの場所をサーバーに指示する plugin_dir の値を設定します。

プラグインを有効にするには、--plugin-load オプションを付けてサーバーを起動します。たとえば、my.cnf ファイルに次の行を挿入します。オブジェクトファイルのサフィクスがシステム上の .so と異なる場合は、正しいサフィクスに置き換えてください。

```
[mysqld]
plugin-load=authentication_pam.so
```

このプラグインを使用して認証するべき MySQL アカウントには、CREATE USER または GRANT ステートメントの IDENTIFIED WITH 句の authentication_pam というプラグイン名を使用します。

プラグインのインストールを確認するには、INFORMATION_SCHEMA.PLUGINS テーブルを調査するか、または SHOW PLUGINS ステートメントを使用します。セクション5.1.8.2「サーバープラグイン情報の取得」を参照してください。

PAM 認証プラグインの使用

このセクションでは、PAM 認証プラグインを使用して、MySQL クライアントプログラムからサーバーに接続する方法について説明します。サーバー側のプラグインが有効になっていて、クライアント側のプラグインを追加するのに十分新しいクライアントプログラムであることが前提となっています。

> 注記
>
> PAM プラグインの通信相手であるクライアント側のプラグインは、PAM に渡すことができるように、単純にパスワードをクリアテキストでサーバーに送信します。これにより、一部の構成でセキュリティーの問題が発生する可能性がありますが、サーバー側の PAM ライブラリを使用するためには必要です。パスワードが傍受される可能性がある場合に、問題を回避するには、クライアントは SSL を使用して MySQL サーバーに接続するようにしてください。セクション6.3.8.7「クライアント側のクリアテキスト認証プラグイン」を参照してください。

CREATE USER または GRANT ステートメントの IDENTIFIED WITH 句で PAM 認証プラグインを参照するには、authentication_pam という名前を使用します。例:

```
CREATE USER user
  IDENTIFIED WITH authentication_pam
  AS 'authentication_string';
```

認証文字列には、次のタイプの情報が指定されます。

- PAM では、システム管理者が特定のアプリケーション用の認証方式を構成する際に使用できる名前である「サービス名」の概念がサポートされています。このような複数の「アプリケーション」が単一のデータベースサーバーインスタンスに関連付けられている可能性もあるため、サービス名の選択は SQL アプリケーションの開発者に任されます。PAM を使用して認証するべきアカウントを定義する際は、認証文字列でサービス名を指定します。
- PAM では、ログイン時に指定されたログイン名以外の MySQL ユーザー名を PAM モジュールからサーバーに返すための方法が提供されています。認証文字列を使用して、ログイン名と MySQL ユーザー名とのマッピングを制御します。プロキシユーザーの機能を活用するには、この種類のマッピングを認証文字列に含める必要があります。

たとえば、サービス名が mysql であり、root および users PAM グループ内のユーザーを developer および data_entry ユーザーにマップする必要がある場合は、それぞれ次のようなステートメントを使用します。

```
CREATE USER user
  IDENTIFIED WITH authentication_pam
  AS 'mysql, root=developer, users=data_entry';
```

PAM 認証プラグインでの認証文字列の構文は、次のようなルールに従っています。

- 文字列は PAM サービス名で構成され、それぞれグループ名と SQL ユーザー名を指定する 1 つ以上のキーワードと値のペアで構成されるグループマッピングリストがオプションで続きます。

  ```
  pam_service_name[,group_name=sql_user_name]...
  ```

- group_name=sql_user_name の各ペアの前には、カンマを付ける必要があります。
- 二重引用符で囲まれていない先頭および末尾の空白文字は、無視されます。
- 引用符で囲まれていない pam_service_name、group_name、および sql_user_name の値には、等号、カンマ、または空白文字を除く、任意の文字を含めることができます。
- pam_service_name、group_name、または sql_user_name の値を二重引用符で囲むと、引用符間のすべての文字が値の一部となります。たとえば、値に空白文字が含まれている場合は、これが必要です。二重引用符およびバックスラッシュ ('\') を除く、すべての文字が有効です。どちらかの文字を含めるには、バックスラッシュを使用してエスケープします。

このプラグインは、各ログインチェック時に認証文字列を解析します。オーバーヘッドを最小限に抑えるには、できるだけ文字列を短く保ちます。

プラグインが正常にログイン名を認証すると、認証文字列でグループマッピングリストを検索し、そのリストを使用して外部ユーザーがメンバーになっているグループに基づいて MySQL サーバーに別のユーザー名を返します。

- 認証文字列にグループマッピングリストが含まれていない場合は、プラグインはログイン名を返します。
- 認証文字列にグループマッピングリストが含まれている場合は、プラグインはリスト内の各 group_name=sql_user_name ペアを左から右へと調査し、認証済みのユーザーに割り当てられたグループの MySQL 以外のディレクトリ内で、group_name 値の一致を見つけようとし、最初に見つかった一致に対応する sql_user_name を返します。プラグインが任意のグループの一致を見つけた場合は、ログイン名を返します。

プラグインがディレクトリ内のグループを検索できない場合は、グループマッピングリストを無視し、ログイン名を返します。

次のセクションでは、PAM 認証プラグインを使用するいくつかの認証シナリオを設定する方法について説明します。

- プロキシユーザーなし。ここでは、ログイン名とパスワードをチェックする際にのみ PAM が使用されます。MySQL サーバーへの接続が許可されている外部ユーザーはすべて、外部 PAM 認証を使用するように定義されている一致する MySQL アカウントを持つべきです。PAM でサポートされているさまざまな方式で、認証を実行できます。この説明では、従来の Unix パスワードおよび LDAP を使用する方法を示します。

  PAM 認証では、プロキシユーザーまたはグループを介して実行されない場合に、MySQL アカウントは Unix アカウントと同じユーザー名を持つ必要があります。MySQL ユーザー名は 16 文字に制限されているため (セクション6.2.2「権限システム付与テーブル」を参照してください)、PAM の非プロキシ認証は、最長で 16 文字の名前を持つ Unix アカウントに制限されます。

- プロキシログインのみ、およびグループマッピング。このシナリオでは、さまざまな権限セットを定義する数個の MySQL アカウントを作成します。(理想は、これらを使用して直接ログインしないようにすることです。) そのあと、権限セットを保持している数個の MySQL アカウントに、すべての外部ログインをマップする一部のマッピングスキームが (通常は、ユーザーが属する外部グループによって) 使用される PAM を使用して、デフォルトのユーザー認証を定義します。ログインする任意のユーザーは、MySQL アカウントのいずれかにマップされ、その権限を使用します。この説明では、Unix パスワードを使用してこれを設定する方法を示しますが、代わりに LDAP などのその他の PAM 方式が使用される可能性もあります。

これらのシナリオでは、ばらつきがある可能性があります。たとえば、一部のユーザーに直接ログインすることを許可するが、その他のユーザーにはプロキシユーザーを介して接続するように要求できます。

この例は、次のことが前提となっています。システムが異なる方法で設定されている場合は、多少の調整が必要になることもあります。

- PAM 構成ディレクトリは /etc/pam.d です。
- PAM サービス名は mysql です。これは、PAM 構成ディレクトリ内に mysql という名前の PAM ファイル (存在しない場合はファイルを作成します) を設定する必要があることを意味します。別のサービス名を使用する場合は、ファイル名が異なるため、CREATE USER および GRANT ステートメントの AS 句で別の名前を使用する必要があります。
- この例では、antonio というログイン名と verysecret というパスワードが使用されています。認証するユーザーに対応するように、これらを変更します。

PAM 認証プラグインは、初期化時に AUTHENTICATION_PAM_LOG 環境の値が設定されているかどうかをチェックします。その場合、プラグインを使用すると、標準出力への診断メッセージの口ギングが有効になります。これらのメッセージは、プラグインが認証を実行するときに発生する PAM 関連の問題をデバッグする際に役立つことがあります。詳細については、PAM 認証プラグインのデバッグを参照してください。

プロキシユーザーを使用しない Unix パスワード認証

この認証シナリオでは、Unix ユーザーのログイン名とパスワードをチェックする際にのみ PAM が使用されます。MySQL サーバーへの接続が許可されている外部ユーザーはすべて、外部 PAM 認証を使用するように定義されている一致する MySQL アカウントを持つべきです。

1. PAM の Unix 認証で、パスワード verysecret を使用して antonio としてログインすることが許可されていることを確認します。
2. mysql サービスを認証するように PAM を設定します。/etc/pam.d/mysql に次の行を挿入します。

```
#%PAM-1.0
auth       include      password-auth
account    include      password-auth
```

3. Unix ログイン名と同じユーザー名を持つ MySQL アカウントを作成し、PAM プラグインを使用して認証するように定義します。

```
CREATE USER 'antonio'@'localhost'
  IDENTIFIED WITH authentication_pam AS 'mysql';
GRANT ALL PRIVILEGES ON mydb.* TO 'antonio'@'localhost';
```

4. mysql コマンド行クライアントを使用して、MySQL サーバーへの接続を試みます。例:

```
mysql --user=antonio --password=verysecret --enable-cleartext-plugin mydb
```

サーバーは接続を許可し、以降のクエリーは次に示すような出力を返すはずです。

```
mysql> SELECT USER(), CURRENT_USER(), @@proxy_user;
+------------------+------------------+--------------+
| USER()           | CURRENT_USER()   | @@proxy_user |
+------------------+------------------+--------------+
| antonio@localhost | antonio@localhost | NULL         |
+------------------+------------------+--------------+
```

これは、antonio が antonio MySQL アカウントに付与された権限を使用し、プロキシ処理が発生していないことを示します。

プロキシユーザーを使用しない LDAP 認証

この認証シナリオでは、LDAP ユーザーのログイン名とパスワードをチェックする際にのみ PAM が使用されます。MySQL サーバーへの接続が許可されている外部ユーザーはすべて、外部 PAM 認証を使用するように定義されている一致する MySQL アカウントを持つべきです。

1. PAM の LDAP 認証で、パスワード verysecret を使用して antonio としてログインすることが許可されていることを確認します。
2. LDAP を使用して mysql サービスを認証するように PAM を設定します。/etc/pam.d/mysql に次の行を挿入します。

   ```
   #%PAM-1.0
   auth       required    pam_ldap.so
   account    required    pam_ldap.so
   ```

   PAM オブジェクトファイルのサフィクスがシステム上の .so と異なる場合は、正しいサフィクスに置き換えてください。
3. MySQL アカウントの作成およびサーバーへの接続は、プロキシユーザーを使用しない Unix パスワード認証ですでに説明したものと同じです。

プロキシユーザーとグループマッピングを使用した Unix パスワード認証

この認証スキームでは、さまざまな権限セットを定義する数個の MySQL アカウント上に、PAM を使用して MySQL サーバーに接続するユーザーをマップするグループマッピング、およびプロキシ処理が使用されます。ユーザーは、権限を定義するアカウントを使用して直接接続しません。代わりに、権限を保持している数個の MySQL アカウントに、すべての外部ログインをマップするマッピングスキームが使用される PAM を使用して、デフォルトのプロキシユーザー認証を使用して接続します。接続する任意のユーザーは、MySQL アカウントのいずれかにマップされ、その権限を使用します。

ここに示す手順では、Unix パスワード認証が使用されます。代わりに LDAP を使用するには、前半で示したプロキシユーザーを使用しない LDAP 認証の手順を参照してください。

1. PAM の Unix 認証で、パスワード verysecret を使用して antonio としてログインすることが許可されること、および antonio が root または users のメンバーであることを確認します。
2. mysql サービスを認証するように PAM を設定します。/etc/pam.d/mysql に次の行を挿入します。

   ```
   #%PAM-1.0
   auth          include       password-auth
   account       include       password-auth
   ```
3. プロキシ対象アカウントに外部 PAM ユーザーをマップするデフォルトのプロキシユーザーを作成します。root PAM グループから developer MySQL アカウントに外部ユーザーをマップし、users PAM グループから data_entry MySQL アカウントに外部ユーザーをマップします。

   ```
   CREATE USER ''@''
     IDENTIFIED WITH authentication_pam
     AS 'mysql, root=developer, users=data_entry';
   ```

   サービス名のあとのマッピングリストは、プロキシユーザーを設定する際に必要です。それ以外の場合、プラグインは適切なプロキシ対象ユーザー名に PAM グループの名前をマップする方法を指示できません。
4. データベースにアクセスする際に使用されるプロキシ対象アカウントを作成します。

   ```
   CREATE USER 'developer'@'localhost' IDENTIFIED BY 'very secret password';
   GRANT ALL PRIVILEGES ON mydevdb.* TO 'developer'@'localhost';
   CREATE USER 'data_entry'@'localhost' IDENTIFIED BY 'very secret password';
   ```

```
GRANT ALL PRIVILEGES ON mydb.* TO 'data_entry'@'localhost';
```

これらのアカウント用のパスワードをだれにも通知できない場合、ほかのユーザーはこれらのパスワードを使用して、MySQL サーバーに直接接続できません。代わりに、ユーザーは PAM を使用して認証し、自分の PAM グループに基づいてプロキシによる developer または data_entry アカウントを使用することが予想されます。

5. プロキシ対象アカウント用のプロキシアカウントに、PROXY 権限を付与します。

```
GRANT PROXY ON 'developer'@'localhost' TO ''@'';
GRANT PROXY ON 'data_entry'@'localhost' TO ''@'';
```

6. mysql コマンド行クライアントを使用して、MySQL サーバーへの接続を試みます。例:

```
mysql --user=antonio --password=verysecret --enable-cleartext-plugin mydb
```

サーバーは ''@'' アカウントを使用して、接続を認証します。権限 antonio は、メンバーとして属する PAM グループによって異なります。antonio が root PAM グループのメンバーである場合、PAM プラグインは、developer MySQL ユーザー名に root をマップし、その名前をサーバーに返します。サーバーは、''@'' が developer に対する PROXY 権限を持っていることを確認し、接続を許可します。以降のクエリーは、次に示すような出力を返すはずです。

```
mysql> SELECT USER(), CURRENT_USER(), @@proxy_user;
+------------------+---------------------+--------------+
| USER()           | CURRENT_USER()      | @@proxy_user |
+------------------+---------------------+--------------+
| antonio@localhost | developer@localhost | ''@''        |
+------------------+---------------------+--------------+
```

これは、antonio が developer MySQL アカウントに付与された権限を使用し、デフォルトのプロキシユーザーアカウントからプロキシ処理が発生していないことを示します。

antonio が root PAM グループのメンバーではないが、users グループのメンバーである場合にも、同様のプロセスが発生しますが、プラグインは data_entry MySQL ユーザー名に user グループメンバーシップをマップし、その名前をサーバーに返します。この場合、antonio は data_entry MySQL アカウントの権限を使用します。

```
mysql> SELECT USER(), CURRENT_USER(), @@proxy_user;
+------------------+---------------------+--------------+
| USER()           | CURRENT_USER()      | @@proxy_user |
+------------------+---------------------+--------------+
| antonio@localhost | data_entry@localhost | ''@''        |
+------------------+---------------------+--------------+
```

PAM 認証プラグインのデバッグ

PAM 認証プラグインは、初期化時に AUTHENTICATION_PAM_LOG 環境の値が設定されているかどうかをチェックします (値は問題ありません)。その場合、プラグインを使用すると、標準出力への診断メッセージの口ギングが有効になります。これらのメッセージは、プラグインが認証を実行するときに発生する PAM 関連の問題をデバッグする際に役立つことがあります。

一部のメッセージには、PAM プラグインソースファイルと行番号への参照が含まれています。これを使用すると、プラグインアクションをそれが発生するコード内の場所に、より緊密に関連付けることができます。

次のトランスクリプトでは、ロギングを有効にすると生成される情報の種類を示します。これは、正常なプロキシ認証の試行による結果です。

```
entering auth_pam_server
entering auth_pam_next_token
auth_pam_next_token:reading at [cups,admin=writer,everyone=reader], sep=[,]
auth_pam_next_token:state=PRESPACE, ptr=[cups,admin=writer,everyone=reader],
out=[]
auth_pam_next_token:state=IDENT, ptr=[cups,admin=writer,everyone=reader],
out=[]
auth_pam_next_token:state=AFTERSPACE, ptr=[,admin=writer,everyone=reader],
out=[cups]
auth_pam_next_token:state=DELIMITER, ptr=[,admin=writer,everyone=reader],
out=[cups]
auth_pam_next_token:state=DONE, ptr=[,admin=writer,everyone=reader],
out=[cups]
leaving auth_pam_next_token on
/Users/gkodinov/mysql/work/x-5.5.16-release-basket/release/plugin/pam-authentication-plugin/src/parser.c:191
auth_pam_server:password 12345qq received
```

```
auth_pam_server:pam_start rc=0
auth_pam_server:pam_set_item(PAM_RUSER,gkodinov) rc=0
auth_pam_server:pam_set_item(PAM_RHOST,localhost) rc=0
entering auth_pam_server_conv
auth_pam_server_conv:PAM_PROMPT_ECHO_OFF [Password:] received
leaving auth_pam_server_conv on
/Users/gkodinov/mysql/work/x-5.5.16-release-basket/release/plugin/pam-authentication-plugin/src/authentication_pam.c:257
auth_pam_server:pam_authenticate rc=0
auth_pam_server:pam_acct_mgmt rc=0
auth_pam_server:pam_setcred(PAM_ESTABLISH_CRED) rc=0
auth_pam_server:pam_get_item rc=0
auth_pam_server:pam_setcred(PAM_DELETE_CRED) rc=0
entering auth_pam_map_groups
entering auth_pam_walk_namevalue_list
auth_pam_walk_namevalue_list:reading at: [admin=writer,everyone=reader]
entering auth_pam_next_token
auth_pam_next_token:reading at [admin=writer,everyone=reader], sep=[=]
auth_pam_next_token:state=PRESPACE, ptr=[admin=writer,everyone=reader], out=[]
auth_pam_next_token:state=IDENT, ptr=[admin=writer,everyone=reader], out=[]
auth_pam_next_token:state=AFTERSPACE, ptr=[=writer,everyone=reader],
out=[admin]
auth_pam_next_token:state=DELIMITER, ptr=[=writer,everyone=reader],
out=[admin]
auth_pam_next_token:state=DONE, ptr=[=writer,everyone=reader], out=[admin]
leaving auth_pam_next_token on
/Users/gkodinov/mysql/work/x-5.5.16-release-basket/release/plugin/pam-authentication-plugin/src/parser.c:191
auth_pam_walk_namevalue_list:name=[admin]
entering auth_pam_next_token
auth_pam_next_token:reading at [writer,everyone=reader], sep=[,]
auth_pam_next_token:state=PRESPACE, ptr=[writer,everyone=reader], out=[]
auth_pam_next_token:state=IDENT, ptr=[writer,everyone=reader], out=[]
auth_pam_next_token:state=AFTERSPACE, ptr=[,everyone=reader], out=[writer]
auth_pam_next_token:state=DELIMITER, ptr=[,everyone=reader], out=[writer]
auth_pam_next_token:state=DONE, ptr=[,everyone=reader], out=[writer]
leaving auth_pam_next_token on
/Users/gkodinov/mysql/work/x-5.5.16-release-basket/release/plugin/pam-authentication-plugin/src/parser.c:191
walk, &error_namevalue_list:value=[writer]
entering auth_pam_map_group_to_user
auth_pam_map_group_to_user:pam_user=gkodinov, name=admin, value=writer
examining member root
examining member gkodinov
substitution was made to mysql user writer
leaving auth_pam_map_group_to_user on
/Users/gkodinov/mysql/work/x-5.5.16-release-basket/release/plugin/pam-authentication-plugin/src/authentication_pam.c:118
auth_pam_walk_namevalue_list:found mapping
leaving auth_pam_walk_namevalue_list on
/Users/gkodinov/mysql/work/x-5.5.16-release-basket/release/plugin/pam-authentication-plugin/src/parser.c:270
auth_pam_walk_namevalue_list returned 0
leaving auth_pam_map_groups on
/Users/gkodinov/mysql/work/x-5.5.16-release-basket/release/plugin/pam-authentication-plugin/src/authentication_pam.c:171
auth_pam_server:authenticated_as=writer
auth_pam_server: rc=0
leaving auth_pam_server on
/Users/gkodinov/mysql/work/x-5.5.16-release-basket/release/plugin/pam-authentication-plugin/src/authentication_pam.c:429
```

6.3.8.6 Windows ネイティブ認証プラグイン

> **注記**
>
> Windows 認証プラグインは、商用の拡張機能です。商用の製品 (MySQL Enterprise Edition) についてさらに学習するには、https://www.mysql.com/products/ を参照してください。

MySQL 5.6.10 の時点で、Windows 向けの MySQL の商用配布には、Windows 上で外部認証を実行する認証プラグインが含まれています。これを使用すると、MySQL サーバーがネイティブの Windows サービスを使用して、クライアント接続を認証できます。Windows にログインしたユーザーは、追加のパスワードを指定せずに、自分の環境内の情報に基づいて MySQL クライアントプログラムからサーバーに接続できます。

クライアントとサーバーは、認証ハンドシェイクでデータパケットを交換します。この交換の結果として、サーバーは Windows OS 内のクライアントのアイデンティティーを表すセキュリティーコンテキストオブジェクトを作成します。このアイデンティティーには、クライアントアカウントの名前が含まれています。Windows 認証プラグインはクライアントのアイデンティティーを使用して、特定のアカウントまたはグループのメンバーであるのかどうかをチェックします。デフォルトでは、認証のネゴシエーションに Kerberos が使用されます。Kerberos が使用できない場合は、NTLM が使用されます。

Windows 認証プラグインには、次のような機能が備わっています。

- 外部認証: このプラグインを使用すると、MySQL サーバーは、MySQL 付与テーブルの外部で定義されたユーザーからの接続を受け入れることができます。
- プロキシユーザーのサポート: このプラグインは、クライアントユーザーとは異なるユーザー名を MySQL に返すことができます。つまり、このプラグインは、外部の Windows で認証されたユーザーが持つべき権限を定義する MySQL ユーザーを返すことができます。たとえば、joe という名前の Windows ユーザーは接続して、developer という名前の MySQL ユーザーの権限を持つことができます。

次の表には、プラグインおよびライブラリファイルの名前を示します。ファイルの場所は、plugin_dir システム変数で指定されたディレクトリである必要があります。インストールに関する情報については、Windows 認証プラグインのインストールを参照してください。

表 6.12 MySQL Windows 認証プラグイン

サーバー側のプラグイン名	authentication_windows
クライアント側のプラグイン名	authentication_windows_client
ライブラリオブジェクトファイル名	authentication_windows.dll

ライブラリファイルには、サーバー側のプラグインのみが含まれています。クライアント側のプラグインは、libmysqlclient クライアントライブラリに組み込まれています。

サーバー側の Windows 認証プラグインは、商用配布にのみ含まれています。MySQL コミュニティー配布には含まれていません。クライアント側のプラグインは、コミュニティー配布を含むすべての配布に含まれています。これにより、任意の配布から、サーバー側のプラグインがロードされたサーバーに接続することがクライアントに許可されます。

Windows 認証プラグインは、MySQL 5.6 でサポートされている任意のバージョンの Windows でサポートされています (https://www.mysql.com/support/supportedplatforms/database.htmlを参照してください)。これには、MySQL Server 5.6.10 以降が必要です。

MySQL のプラガブル認証に関する一般的な情報については、セクション6.3.7「プラガブル認証」を参照してください。プロキシユーザーについては、セクション6.3.9「プロキシユーザー」を参照してください。

Windows 認証プラグインのインストール

Windows 認証プラグインは、MySQL プラグインディレクトリ (plugin_dir システム変数で指定されたディレクトリ) にインストールする必要があります。必要に応じて、サーバーの起動時に、プラグインディレクトリの場所をサーバーに指示する plugin_dir の値を設定します。

プラグインを有効にするには、--plugin-load オプションを付けてサーバーを起動します。たとえば、my.ini ファイルに次の行を挿入します。

```
[mysqld]
plugin-load=authentication_windows.dll
```

このプラグインを使用して認証される MySQL アカウントには、CREATE USER または GRANT ステートメントの IDENTIFIED WITH 句のプラグイン名 authentication_windows を使用します。

プラグインのインストールを確認するには、INFORMATION_SCHEMA.PLUGINS テーブルを調査するか、または SHOW PLUGINS ステートメントを使用します。セクション5.1.8.2「サーバープラグイン情報の取得」を参照してください。

Windows 認証プラグインの使用

Windows 認証プラグインでは、Windows にログインしたユーザーが追加のパスワードを指定しなくても、MySQL サーバーに接続できるように、MySQL アカウントの使用がサポートされています。サーバー側のプラグインが有効になっていて、クライアントプログラムが libmysqlclient に組み込まれたクライアント側のプラグインを追加するために十分新しいバージョンであることが前提となっています。DBA がサーバー側のプラグインを有効にして、それを使用するようにアカウントを設定すると、クライアントは自分の側でその他の設定を行う必要なしで、これらのアカウントを使用して接続できます。

CREATE USER または GRANT ステートメントの IDENTIFIED WITH 句で Windows 認証プラグインを参照するには、authentication_windows という名前を使用します。Rafal と Tasha という Windows ユーザー、および Administrators または Power Users グループ内の任意のユーザーが MySQL への接続が許可されるべきであ

ると仮定します。このように設定するには、Windows プラグインを使用して認証する sql_admin という名前の MySQL アカウントを作成します。

```
CREATE USER sql_admin
  IDENTIFIED WITH authentication_windows
  AS 'Rafal, Tasha, Administrators, "Power Users"';
```

プラグイン名は authentication_windows です。AS キーワードに続く文字列は、認証文字列です。Rafal または Tasha という名前の Windows ユーザー、および Administrators または Power Users グループ内の任意の Windows ユーザーが MySQL ユーザー sql_admin として、サーバーへの認証が許可されるように指定されます。後者のグループ名には空白文字が含まれているため、二重引用符で囲む必要があります。

sql_admin アカウントを作成したあとは、Windows にログインしたユーザーはそのアカウントを使用して、サーバーへの接続を試みることができます。

```
C:\> mysql --user=sql_admin
```

ここでは、パスワードは必要ありません。authentication_windows プラグインは Windows のセキュリティー API を使用して、接続中の Windows ユーザーをチェックします。そのユーザーが Rafal または Tasha という名前であるか、Administrators または Power Users グループに属する場合、サーバーはアクセス権を付与し、クライアントは sql_admin として認証され、sql_admin アカウントに付与される任意の権限を持ちます。それ以外の場合、サーバーはアクセスを拒否します。

Windows 認証プラグインでの認証文字列の構文は、次のようなルールに従っています。

- 文字列は、カンマで区切られた 1 つ以上のユーザーマッピングで構成されます。
- 各ユーザーマッピングによって、Windows ユーザー名またはグループ名が MySQL ユーザー名に関連付けられます。

  ```
  win_user_or_group_name=sql_user_name
  win_user_or_group_name
  ```

  後者の構文に sql_user_name 値を指定しない場合、値は暗黙的に、CREATE USER ステートメントで作成された MySQL ユーザーとなります。したがって、次のステートメントは同等です。

  ```
  CREATE USER sql_admin
    IDENTIFIED WITH authentication_windows
    AS 'Rafal, Tasha, Administrators, "Power Users"';

  CREATE USER sql_admin
    IDENTIFIED WITH authentication_windows
    AS 'Rafal=sql_admin, Tasha=sql_admin, Administrators=sql_admin,
       "Power Users"=sql_admin';
  ```

- MySQL の文字列ではバックスラッシュ ('\') はエスケープ文字であるため、値内の各バックスラッシュは二重に入力する必要があります。
- 二重引用符で囲まれていない先頭および末尾の空白文字は、無視されます。
- 引用符で囲まれていない win_user_or_group_name および sql_user_name の値には、等号、カンマ、または空白文字を除く、任意の文字を含めることができます。
- win_user_or_group_name または sql_user_name の値を二重引用符で囲むと、引用符間のすべての文字が値の一部となります。たとえば、名前に空白文字が含まれている場合は、これが必要です。二重引用符およびバックスラッシュを除く、二重引用符内のすべての文字が有効です。どちらかの文字を含めるには、バックスラッシュを使用してエスケープします。
- win_user_or_group_name 値では、Windows 主体 (ローカルまたはドメイン内) 用の従来の構文が使用されます。例 (バックスラッシュを二重にすることに注意してください):

  ```
  domain\\user
  .\\user
  domain\\group
  .\\group
  BUILTIN\\WellKnownGroup
  ```

クライアントを認証するためにサーバーから呼び出されると、プラグインは認証文字列を左から右へとスキャンして、Windows ユーザーとのユーザーまたはグループの一致があるかどうかを確認します。一致がある場合、このプラグインは対応する sql_user_name を MySQL サーバーに返します。一致がない場合は、認証に失敗します。

ユーザー名の一致は、グループ名の一致よりも優先されます。win_user という名前の Windows ユーザーが win_group のメンバーであり、認証文字列が次のとおりであると仮定します。

```
'win_group = sql_user1, win_user = sql_user2'
```

win_user が MySQL サーバーに接続すると、win_group と win_user の両方への一致があります。グループが認証文字列の最初に一覧表示されますが、より具体的なユーザーの一致がグループの一致よりも優先されるため、プラグインは sql_user2 としてユーザーを認証します。

サーバーが実行されているものと同じコンピュータからの接続では、Windows 認証は常に機能します。コンピュータ間の接続では、両方のコンピュータを Windows Active Directory に登録する必要があります。同じ Windows ドメイン内にある場合は、ドメイン名を指定する必要はありません。次の例に示すように、別のドメインからの接続を許可することもできます。

```
CREATE USER sql_accounting
  IDENTIFIED WITH authentication_windows
  AS 'SomeDomain\\Accounting';
```

ここで、SomeDomain は別のドメインの名前です。バックスラッシュ文字は文字列内の MySQL エスケープ文字であるため、二重に入力されています。

MySQL では、クライアントは 1 つのアカウントを使用して MySQL サーバーに接続して認証できるが、接続されると別のアカウントの権限を持つというプロキシユーザーの概念がサポートされています (セクション6.3.9「プロキシユーザー」を参照してください)。次のように、Windows ユーザーは単一のユーザー名を使用して接続するが、Windows ユーザー名およびグループ名に基づいて特定の MySQL アカウント上にマップされると仮定します。

- local_user および MyDomain\domain_user というローカルおよびドメインの Windows ユーザーは、local_wlad MySQL アカウントにマップするべきです。
- MyDomain\Developers ドメイングループ内のユーザーは、local_dev MySQL アカウントにマップするべきです。
- ローカルマシンの管理者は、local_admin MySQL アカウントにマップするべきです。

このように設定するには、接続先の Windows ユーザーのプロキシアカウントを作成し、ユーザーとグループが適切な MySQL アカウント (local_wlad、local_dev、local_admin) にマップされるように、このアカウントを構成します。さらに、実行する必要がある操作に適した権限を MySQL アカウントに付与します。次の手順では、プロキシアカウントとして win_proxy が使用され、プロキシ対象アカウントとして local_wlad、local_dev、および local_admin が使用されています。

1. プロキシ MySQL アカウントを作成します。

   ```
   CREATE USER win_proxy
     IDENTIFIED WITH  authentication_windows
     AS 'local_user = local_wlad,
         MyDomain\\domain_user = local_wlad,
         MyDomain\\Developers = local_dev,
         BUILTIN\\Administrators = local_admin';
   ```

2. プロキシ処理が動作するには、プロキシ対象アカウントが存在する必要があるため、次のように作成します。

   ```
   CREATE USER local_wlad IDENTIFIED BY 'wlad_pass';
   CREATE USER local_dev IDENTIFIED BY 'dev_pass';
   CREATE USER local_admin IDENTIFIED BY  'admin_pass';
   ```

   これらのアカウント用のパスワードをだれにも通知できない場合、ほかのユーザーはこれらのパスワードを使用して、MySQL サーバーに直接接続できません。

   また、必要な権限を各プロキシ対象アカウントに付与する GRANT ステートメント (非表示) も発行するようにしてください。

3. プロキシアカウントは、プロキシ対象アカウントごとに PROXY 権限を持つ必要があります。

   ```
   GRANT PROXY ON local_wlad TO win_proxy;
   GRANT PROXY ON local_dev TO win_proxy;
   GRANT PROXY ON local_admin TO win_proxy;
   ```

この時点で、local_user および MyDomain\domain_user という Windows ユーザーは、win_proxy として MySQL サーバーに接続でき、認証されると、認証文字列 (この場合は local_wlad) に指定されたアカウントの権限を持ち

ます。win_proxy として接続する MyDomain\Developers グループ内のユーザーは、local_dev アカウントの権限を持っています。BUILTIN\Administrators グループ内のユーザーは、local_admin アカウントの権限を持っています。

自分の MySQL アカウントを持っていないのすべての Windows ユーザーがプロキシアカウントを通過するように認証を構成するには、前述の手順で、win_proxy をデフォルトのプロキシユーザー ("@") に置き換えてください。デフォルトのプロキシユーザーについては、セクション6.3.9「プロキシユーザー」を参照してください。

Connection/Net 6.4.4 以上で、Connector/Net 接続文字列とともに Windows 認証プラグインを使用する方法については、Using the Windows Native Authentication Pluginを参照してください。

authentication_windows_use_principal_name および authentication_windows_log_level システム変数では、Windows 認証プラグインへの追加の制御が提供されています。セクション5.1.4「サーバーシステム変数」を参照してください。

6.3.8.7 クライアント側のクリアテキスト認証プラグイン

MySQL 5.6.2 の時点では、ハッシュ化または暗号化なしでサーバーにパスワードを送信するクライアント側の認証プラグインが使用可能です。このプラグインは、MySQL クライアントライブラリに組み込まれています。

次の表に、プラグイン名を示します。

表 6.13 MySQL クリアテキスト認証プラグイン

サーバー側のプラグイン名	なし (説明を参照してください)
クライアント側のプラグイン名	mysql_clear_password
ライブラリオブジェクトファイル名	なし (プラグインは組み込み済みです)

ネイティブの MySQL 認証では、クライアントはパスワードに一方向のハッシュ化を実行してから、それをサーバーに送信します。これにより、クライアントがパスワードをクリアテキストで送信することを回避できます。セクション6.1.2.4「MySQL でのパスワードハッシュ」を参照してください。ただし、ハッシュアルゴリズムは一方向であるため、サーバー側で元のパスワードをリカバリできません。

クライアント側で入力されたパスワードを受信するようにサーバーに要求する認証スキームの場合、一方向のハッシュ化を実行できません。このような場合、クライアント側のプラグイン mysql_clear_password を使用すると、パスワードをクリアテキストでサーバーに送信できます。対応するサーバー側のプラグインはありません。その上、クライアント側のプラグインは、クリアテキストのパスワードを必要とする任意のサーバー側のプラグインで使用できます。(PAM 認証プラグインは、このような例の 1 つです。セクション6.3.8.5「PAM 認証プラグイン」を参照してください。)

MySQL のプラガブル認証に関する一般的な情報については、セクション6.3.7「プラガブル認証」を参照してください。

> **注記**
>
> クリアテキストでパスワードを送信すると、一部の構成でセキュリティーの問題が発生する可能性があります。パスワードが傍受される可能性がある場合に問題を回避するには、クライアントはパスワードが保護される方式を使用して、MySQL サーバーに接続するようにしてください。SSL (セクション6.3.10「セキュアな接続のための SSL の使用」を参照)、IPsec、またはプライベートネットワークでも発生する可能性があります。

MySQL 5.6.7 の時点で、このプラグインが不注意に使用される可能性を低くするには、クライアントが明示的に有効にする必要があります。これは、複数の方法で実行できます。

- LIBMYSQL_ENABLE_CLEARTEXT_PLUGIN 環境変数を 1、Y、または y で始まる値に設定します。これにより、すべてのクライアント接続でプラグインが有効になります。
- mysql、mysqladmin、および mysqlslap クライアントプログラムでは、呼び出すたびにプラグインを有効にする --enable-cleartext-plugin オプションがサポートされています。
- mysql_options() C API 関数では、接続するたびにプラグインを有効にする MYSQL_ENABLE_CLEARTEXT_PLUGIN オプションがサポートされています。また、クライアントライブラ

リによって読み取られるオプショングループ内に enable-cleartext-plugin を含めると、libmysqlclient を使用し、オプションファイルを読み取る任意のプログラムでプラグインを有効にすることができます。

6.3.8.8 ソケットピア証明書認証プラグイン

MySQL 5.6.2 の時点では、Unix ソケットファイル経由でローカルホストから接続するクライアントを認証するサーバー側の認証プラグインが使用可能です。

このプラグインのソースコードは、ロード可能な認証プラグインを記述する方法を示す比較的単純な例として調査できます。

次の表には、プラグインおよびライブラリファイルの名前を示します。ファイル名のサフィクスは、システムによって異なる場合があります。ファイルの場所は、plugin_dir システム変数で指定されたディレクトリです。インストールに関する情報については、セクション6.3.7「プラガブル認証」を参照してください。

表 6.14 MySQL ソケットピア証明書認証プラグイン

サーバー側のプラグイン名	auth_socket
クライアント側のプラグイン名	なし (説明を参照してください)
ライブラリオブジェクトファイル名	auth_socket.so

auth_socket 認証プラグインは、Unix ソケットファイル経由でローカルホストから接続するクライアントを認証します。このプラグインは SO_PEERCRED ソケットオプションを使用して、クライアントプログラムを実行しているユーザーに関する情報を取得します。このプラグインは、ユーザー名がサーバーへのクライアントプログラムで指定された MySQL ユーザー名と一致するかどうかをチェックし、その名前が一致する場合にのみ接続を許可します。SO_PEERCRED オプションがサポートされているシステム (Linux など) 上でのみ、このプラグインを構築できます。

ソケットファイル経由でローカルホストから接続する際に、auth_socket プラグインで認証される valerie という名前のユーザー用に、MySQL アカウントが作成されると仮定します。

```
CREATE USER 'valerie'@'localhost' IDENTIFIED WITH auth_socket;
```

stefanie というログイン名を持つローカルホスト上のユーザーが --user=valerie オプションを付けて mysql を呼び出して、ソケットファイル経由で接続する場合、サーバーは auth_socket を使用してクライアントを認証します。このプラグインは、--user オプションの値 (valerie) がクライアントユーザー名 (stephanie) とは異なると判断し、接続を拒否します。valerie という名前のユーザーが同じことを試みた場合、プラグインはユーザー名と MySQL ユーザー名が両方とも valerie であると判断し、接続を許可します。ただし、TCP/IP などの別のプロトコルを使用して接続されると、valerie の場合でもプラグインは接続を拒否します。

MySQL のプラガブル認証に関する一般的な情報については、セクション6.3.7「プラガブル認証」を参照してください。

6.3.8.9 テスト認証プラグイン

MySQL には、MySQL ネイティブ認証を使用して認証するテストプラグインが含まれていますが、(組み込まれていない) ロード可能なプラグインであるため、使用する前にインストールする必要があります。標準または古い (短い) パスワードのハッシュ値と照合して認証できます。

このプラグインは、テストおよび開発のために使用され、本番環境では使用されません。テストプラグインのソースコードは組み込みのネイティブプラグインとは異なり、サーバーソースとは別々のものであるため、ロード可能な認証プラグインを記述する方法を示す比較的単純な例として調査できます。

次の表には、プラグインおよびライブラリファイルの名前を示します。ファイル名のサフィクスは、システムによって異なる場合があります。ファイルの場所は、plugin_dir システム変数で指定されたディレクトリです。インストールに関する情報については、セクション6.3.7「プラガブル認証」を参照してください。

表 6.15 MySQL テスト認証プラグイン

サーバー側のプラグイン名	test_plugin_server
クライアント側のプラグイン名	auth_test_plugin

ライブラリオブジェクトファイル名	auth_test_plugin.so

テストプラグインでは、ネイティブの MySQL 認証と同じ方法で認証されるため、サーバーに接続する際に、ネイティブ認証を使用するアカウント用に一般的に使用される通常の --user および --password オプションを指定します。例:

```
shell> mysql --user=your_name --password=your_pass
```

MySQL のプラガブル認証に関する一般的な情報については、セクション6.3.7「プラガブル認証」を参照してください。

6.3.9 プロキシユーザー

認証プラグインを使用した MySQL サーバーへの認証が発生すると、接続中の (外部) ユーザーを権限チェックのために別のユーザーとして処理するようにプラグインから要求されることがあります。これにより、外部ユーザーを 2 番目のユーザーのプロキシにすることができます。つまり、2 番目のユーザーの権限を持つことができます。言い換えると、外部ユーザーは「プロキシユーザー」(偽装できるユーザーまたは別のユーザーと呼ばれるようになるユーザー) であり、2 番目のユーザーは「プロキシ対象ユーザー」(プロキシユーザーが実行できるアイデンティティーを持つユーザー) です。

このセクションでは、プロキシユーザー機能の動作について説明します。認証プラグインに関する一般的な情報については、セクション6.3.7「プラガブル認証」を参照してください。プロキシユーザーがサポートされている独自の認証プラグインを作成することに関心がある場合は、認証プラグインでのプロキシユーザーサポートの実装を参照してください。

プロキシ処理を発生させるには、次の条件を満たす必要があります。

- 接続中のクライアントがプロキシユーザーとして処理されるときに、プラグインはプロキシ対象ユーザーの名前を示すために、別の名前を返す必要があります。
- プロキシユーザーのアカウントがプラグインで認証されるように設定する必要があります。CREATE USER または GRANT ステートメントを使用して、アカウントをプラグインに関連付けます。
- プロキシユーザーのアカウントは、プロキシ対象アカウントの PROXY 権限を持つ必要があります。これを行うには、GRANT ステートメントを使用します。

次のような定義について検討します。

```
CREATE USER 'empl_external'@'localhost'
  IDENTIFIED WITH auth_plugin AS 'auth_string';
CREATE USER 'employee'@'localhost'
  IDENTIFIED BY 'employee_pass';
GRANT PROXY
  ON 'employee'@'localhost'
  TO 'empl_external'@'localhost';
```

クライアントがローカルホストから empl_external として接続すると、MySQL は auth_plugin を使用して認証を実行します。auth_plugin が ('auth_string' の内容に基づいて、おそらく一部の外部認証システムを参照することで) サーバーに employee というユーザー名を返す場合は、このクライアントを権限チェックのために、employee ローカルユーザーとして処理するように求めるサーバーへのリクエストとして機能します。

この場合、empl_external はプロキシユーザー、employee はプロキシ対象ユーザーです。

サーバーは empl_external が employee に対する PROXY 権限を持っているかどうかをチェックすることで、empl_external ユーザーに対して employee のプロキシ認証を実行できることを確認します。(この権限が付与されていない場合は、エラーが発生します。)

プロキシ処理が発生したときに、USER() および CURRENT_USER() 関数を使用すると、接続中のユーザーと現在のセッション中に権限が適用されるアカウントとの相違点を確認できます。先ほど説明した例では、これらの関数は次の値を返します。

```
mysql> SELECT USER(), CURRENT_USER();
+------------------------+--------------------+
| USER()                 | CURRENT_USER()     |
+------------------------+--------------------+
| empl_external@localhost | employee@localhost |
```

```
+-----------------------+-------------------+
```

認証プラグインの名前を指定する IDENTIFIED WITH 句のあとに、ユーザーの接続時にサーバーがプラグインに渡す文字列を指定する AS 句が続く場合があります。AS 句が必要であるかどうかは、各プラグインに依存します。必要な場合、認証文字列の形式は、プラグインを使用する目的によって異なります。許可される認証文字列の値については、特定のプラグインに関するドキュメントを参照してください。

プロキシ権限の付与

外部ユーザーが別のユーザーとして接続し、その権限を持つことができるようにするには、特別な PROXY 権限が必要です。この権限を付与するには、GRANT ステートメントを使用します。例:

```
GRANT PROXY ON 'proxied_user' TO 'proxy_user';
```

proxy_user は、接続時に外部で認証された有効な MySQL ユーザーを表している必要があります。そうでない場合は、接続の試行に失敗します。proxied_user は、接続時にローカルで認証された有効なユーザーを表している必要があります。そうでない場合は、接続の試行に失敗します。

対応する REVOKE 構文は次のとおりです。

```
REVOKE PROXY ON 'proxied_user' FROM 'proxy_user';
```

MySQL GRANT および REVOKE 構文の拡張機能は、通常どおりに動作します。例:

```
GRANT PROXY ON 'a' TO 'b', 'c', 'd';
GRANT PROXY ON 'a' TO 'd' IDENTIFIED BY ...;
GRANT PROXY ON 'a' TO 'd' WITH GRANT OPTION;
GRANT PROXY ON 'a' TO ''@'';
REVOKE PROXY ON 'a' FROM 'b', 'c', 'd';
```

前述の例では、''@'' はデフォルトのプロキシユーザーであり、「任意のユーザー」を意味します。デフォルトのプロキシユーザーについては、このセクションの後半で説明します。

次のような場合に、PROXY 権限を付与できます。

- 自分で proxied_user による: アカウント名のユーザー名とホスト名の両方の部分で、USER() の値が CURRENT_USER() および proxied_user と完全に一致する必要があります。
- proxied_user に対する GRANT PROXY ... WITH GRANT OPTION を持つユーザーによる。

MySQL のインストール時にデフォルトで作成された root アカウントは、''@'' (つまり、すべてのユーザー) に対する PROXY ... WITH GRANT OPTION 権限を持っています。これにより、root はプロキシユーザーを設定したり、プロキシユーザーを設定するための権限をほかのアカウントに委任したりできます。たとえば、root は次の操作を実行できます。

```
CREATE USER 'admin'@'localhost' IDENTIFIED BY 'test';
GRANT PROXY ON ''@'' TO 'admin'@'localhost' WITH GRANT OPTION;
```

この時点で、admin ユーザーは特定のすべての GRANT PROXY マッピングを管理できます。たとえば、admin は次の操作を実行できます。

```
GRANT PROXY ON sally TO joe;
```

デフォルトのプロキシユーザー

一部またはすべてのユーザーが特定の外部プラグインを使用して接続するように指定するには、「空白の」MySQL ユーザーを作成し、そのプラグインを使用して認証するように設定し、(空白のユーザーとは異なる場合に) プラグインが認証された実際のユーザー名を返すことができるようにします。たとえば、LDAP 認証を実装する ldap_auth という名前の仮のプラグインが存在すると仮定します。

```
CREATE USER ''@'' IDENTIFIED WITH ldap_auth AS 'O=Oracle, OU=MySQL';
CREATE USER 'developer'@'localhost' IDENTIFIED BY 'developer_pass';
CREATE USER 'manager'@'localhost' IDENTIFIED BY 'manager_pass';
GRANT PROXY ON 'manager'@'localhost' TO ''@'';
GRANT PROXY ON 'developer'@'localhost' TO ''@'';
```

ここで、クライアントが次のように接続を試みるとします。

```
mysql --user=myuser --password='myuser_pass' ...
```

サーバーでは、MySQL ユーザーとして定義された myuser が見つかりません。ただし、クライアントのユーザー名およびホスト名と一致する空白のユーザーアカウント ("@") があるため、サーバーはそのアカウントと照合してクライアントを認証します。サーバーは ldap_auth を呼び出して、それをユーザー名とパスワードとして myuser および myuser_pass に渡します。

ldap_auth プラグインによって、myuser_pass が myuser の正しいパスワードでないことが LDAP ディレクトリで検出された場合は、認証に失敗し、サーバーは接続を拒否します。

パスワードが正しく、ldap_auth によって myuser が開発者であることが検出された場合は、MySQL サーバーに myuser ではなく、developer というユーザー名が返されます。サーバーは、(実行するための PROXY 権限を持っているため) "@" が developer として認証できることを確認し、接続を受け入れます。セッションは、developer の権限を持っている myuser で続行されます。(これらの権限は、DBA が GRANT ステートメントを使用して設定するべきですが、表示されません。)USER() および CURRENT_USER() 関数は、次の値を返します。

```
mysql> SELECT USER(), CURRENT_USER();
+----------------+--------------------+
| USER()         | CURRENT_USER()     |
+----------------+--------------------+
| myuser@localhost | developer@localhost |
+----------------+--------------------+
```

代わりに、プラグインによって myuser がマネージャーであることが LDAP ディレクトリで検出された場合は、ユーザー名として manager が返され、セッションは manager の権限を持つ myuser で続行されます。

```
mysql> SELECT USER(), CURRENT_USER();
+----------------+------------------+
| USER()         | CURRENT_USER()   |
+----------------+------------------+
| myuser@localhost | manager@localhost |
+----------------+------------------+
```

単純にするために、外部認証はマルチレベルで実行できません。前述の例では、developer の証明書も、manager の証明書も考慮されません。ただし、クライアントが developer または manager アカウントと照合して直接認証を試みる場合は、引き続き使用されます (そのため、これらのアカウントにはパスワードを割り当てられるはずです)。

デフォルトのプロキシアカウントは、任意のホストと一致する '' をホスト部分で使用します。デフォルトのプロキシユーザーを設定する場合は、ホスト部分で '%' を含むアカウントもチェックするように注意してください。その理由は、これらのアカウントは任意のホストにも一致しますが、サーバー内部でアカウント行をソートする際に使用されるルールによって、'' よりも優先されるためです (セクション6.2.4「アクセス制御、ステージ 1: 接続の検証」を参照してください)。

MySQL のインストールに、次の 2 つのアカウントが含まれていると仮定します。

```
CREATE USER "@" IDENTIFIED WITH some_plugin;
CREATE USER "@'%' IDENTIFIED BY 'some_password';
```

1 番目のアカウントの目的は、それ以外の場合は、より具体的なアカウントと一致しないユーザーの接続を認証する際に使用されるデフォルトのプロキシユーザーとして機能することです。2 番目のアカウントは、たとえば、匿名ユーザーとして独自のアカウントを持っていないユーザーを有効にするために、作成されることがあります。

ただし、この構成では、一致のルールによって "@" よりも先に "@'%' がソートされるため、1 番目のアカウントは使用されません。より具体的などのアカウントにも一致しないアカウントの場合、サーバーは "@" ではなく、"@'%' と照合して認証を試みます。

デフォルトのプロキシユーザーを作成する場合は、デフォルトのプロキシユーザーよりも優先されるために、そのユーザーが目的どおりに動作することを妨げるその他の既存の「任意のユーザーに一致する」アカウントがないかどうかチェックします。このようなアカウントをすべて削除する必要がある場合もあります。

プロキシユーザーのシステム変数

次の 2 つのシステム変数は、プロキシのログインプロセスをトレースする際に役立ちます。

- proxy_user: プロキシ処理が使用されていない場合、この値は NULL です。それ以外の場合は、プロキシユーザーのアカウントを示します。たとえば、クライアントがデフォルトのプロキシアカウントを使用して認証する場合、この変数は次のように設定されます。

  ```
  mysql> SELECT @@proxy_user;
  ```

```
+--------------+
| @@proxy_user |
+--------------+
| "@"          |
+--------------+
```

- external_user: 認証プラグインは外部ユーザーを使用して、MySQL サーバーへの認証を行うことがあります。たとえば、Windows のネイティブ認証を使用するときは、Windows の API を使用して認証するプラグインに、ログイン ID を渡す必要がありません。ただし、認証には引き続き Windows ユーザー ID が使用されます。このプラグインは、読み取り専用のセッション変数 external_user を使用して、この外部ユーザー ID (または最初の 512 UTF-8 バイト) をサーバーに返すことがあります。プラグインがこの変数を設定しない場合、その値は NULL です。

6.3.10 セキュアな接続のための SSL の使用

MySQL では、Secure Sockets Layer (SSL) プロトコルを使用した、MySQL クライアントとサーバー間のセキュアな (暗号化された) 接続がサポートされています。このセクションでは、SSL 接続を使用する方法について説明します。SSL 接続を使用するようにユーザーに要求する方法については、セクション13.7.1.4「GRANT 構文」で、GRANT ステートメントの REQUIRE 句の説明を参照してください。

MySQL の標準構成の目的は、できるかぎり高速にすることであるため、デフォルトでは暗号化された接続が使用されません。暗号化された接続で提供されるセキュリティーがアプリケーションで必要である場合は、データを暗号化する際に追加の計算を行なってみる価値があります。

MySQL では、接続ごとに暗号化できます。各アプリケーションの要件に従って、暗号化されていない接続または暗号化されたセキュアな SSL 接続を選択できます。

セキュアな接続は OpenSSL API に基づいており、MySQL C API を介して使用できます。レプリケーションでは C API が使用されるため、マスターサーバーとスレーブサーバー間でセキュアな接続を使用できます。セクション17.3.7「SSL を使用してレプリケーションをセットアップする」を参照してください。

セキュアに接続するためのもう 1 つの方法は、SSH 接続内から MySQL サーバーホストに接続することです。例については、セクション6.3.11「SSH を使用した Windows から MySQL へのリモート接続」を参照してください。

6.3.10.1 基本的な SSL の概念

どのように MySQL で SSL が使用されるのかを理解するには、SSL と X509 の基本概念についていくつか説明する必要があります。これらの概念に精通しているユーザーは、説明のこの部分をスキップできます。

MySQL のデフォルトでは、クライアントとサーバー間で暗号化されていない接続が使用されます。つまり、ネットワークへのアクセス権を持っているユーザーは、すべてのトラフィックを監視し、送受信されるデータを調査できます。クライアントとサーバー間で送受信中のデータでも変更できます。

セキュアな方法でネットワーク経由で情報を移動する必要がある場合は、暗号化されていない接続が許可されません。暗号化は、すべての種類のデータを読み取り不可にする方法です。暗号化されたメッセージの順序を変更したり、データを 2 回再生したりするなどの、多くの種類の既知の攻撃に対抗するために、暗号化アルゴリズムには、セキュリティー要素を含める必要があります。

SSL は、パブリックネットワーク経由で受信したデータを信頼できるようにするために、さまざまな暗号化アルゴリズムが使用されたプロトコルです。すべてのデータの変更、損失、または再生を検出するためのメカニズムが備わっています。SSL には、X509 標準を使用してアイデンティティーを検証するアルゴリズムも組み込まれています。

X509 を使用すると、インターネット上の人物を識別できます。電子商取引アプリケーションで、もっとも一般的に使用されます。基本的な用語には、必要とするすべてのユーザーに電子証明書を割り当てる「認証局」 (CA) と呼ばれるエンティティーがいくつか存在するはずです。証明書は、2 つの暗号化鍵 (公開鍵と秘密鍵) を持つ非対称の暗号化アルゴリズムに依存しています。証明書の所有者は、アイデンティティーの証明として別のパーティーに証明書を表示できます。証明書は、所有者の公開鍵で構成されます。この公開鍵を使用して暗号化されたデータはすべて、対応する秘密鍵 (証明書の所有者が保持しています) を使用しなければ復号化できません。

SSL、X509、暗号化、または公開鍵暗号についての詳細は、インターネット検索を実行して関心のあるキーワードを調べてください。

6.3.10.2 SSL を使用するための MySQL の構成

MySQL サーバーとクライアントプログラム間で SSL 接続を使用するには、システムで OpenSSL または yaSSL がサポートされている必要があります。MySQL 5.6.6 の時点では、SSL サポートがデフォルトで含まれています。

> **重要**
>
> MySQL Community Edition は yaSSL にバンドルされていますが、ソース配布から構築する際に、OpenSSL を使用するように構成できます。MySQL Enterprise Edition は OpenSSL にバンドルされています。MySQL Enterprise Edition では yaSSL を使用できません。

OpenSSL サポートを使用して MySQL サーバーを構築するには、次を実行する必要があります。

1. システムに openSSL 1.0.1 以上がインストールされていることを確認します。OpenSSL を入手するには、http://www.openssl.org にアクセスします。
2. 次の方法で CMake を呼び出して、openSSL が使用されるように MySQL ソース配布を構成します。

   ```
   shell> cmake . -DWITH_SSL=system
   ```

   このコマンドは、インストールされた OpenSSL ライブラリが使用されるように配布を構成します。セクション2.9.4「MySQL ソース構成オプション」を参照してください。-DWITH_SSL を指定しない場合は、yaSSL がデフォルトで使用されます。

   > **重要**
   >
   > インストールされた OpenSSL のバージョンが 1.0.1 よりも小さい場合は、CMake でエラーが生成されます。

3. 配布をコンパイルし、インストールします。

mysqld サーバーで SSL がサポートされているかどうかチェックするには、have_ssl システム変数の値を調査します。

```
mysql> SHOW VARIABLES LIKE 'have_ssl';
+---------------+-------+
| Variable_name | Value |
+---------------+-------+
| have_ssl      | YES   |
+---------------+-------+
```

値が YES の場合は、サーバーで SSL 接続がサポートされています。値が DISABLED の場合、サーバーは SSL 接続をサポートできますが、SSL 接続の使用が有効になるように、適切な --ssl-xxx オプションを付けて起動されませんでした。セクション6.3.10.3「SSL 接続の使用」を参照してください。

6.3.10.3 SSL 接続の使用

SSL 接続を有効にするには、セクション6.3.10.2「SSL を使用するための MySQL の構成」で説明するように、SSL サポートを使用して MySQL 配布を構築します。さらに、適切な証明書および鍵ファイルを指定するには、適切な SSL 関連のオプションを使用する必要があります。SSL オプションの完全なリストについては、セクション6.3.10.4「SSL コマンドのオプション」を参照してください。

SSL を使用した接続がクライアントに許可されるように MySQL サーバーを起動するには、セキュアな接続を確立する際に、サーバーで使用される証明書および鍵ファイルを識別するオプションを使用します。

- --ssl-ca は、認証局 (CA) 証明書を識別します。
- --ssl-cert は、サーバーの公開鍵証明書を識別します。これをクライアントに送信し、それに含まれる CA 証明書と照合して認証できます。
- --ssl-key は、サーバーの秘密鍵を識別します。

たとえば、次のようにサーバーを起動します。

```
shell> mysqld --ssl-ca=ca-cert.pem \
       --ssl-cert=server-cert.pem \
       --ssl-key=server-key.pem
```

各オプションには、PEM 形式のファイル名が指定されます。必要な SSL 証明書および鍵ファイルを生成する手順については、セクション6.3.10.5「MySQL での SSL 証明書および鍵の設定」を参照してください。MySQL

ソース配布を持っている場合は、配布の mysql-test/std_data ディレクトリで、デモ用の証明書および鍵ファイルを使用して設定をテストすることもできます。

クライアント側でも同様オプションが使用されますが、--ssl-cert および --ssl-key は、クライアントの公開鍵と秘密鍵を識別します。認証局の証明書を指定する場合は、サーバーで使用されるときと同じ形式で指定する必要があります。

SSL サポートを使用して MySQL サーバーへのセキュアな接続を確立するために、クライアントが指定する必要があるオプションは、クライアントで使用される MySQL アカウントの SSL 要件によって異なります。(セクション13.7.1.4「GRANT 構文」で、REQUIRE 句の説明を参照してください。)

特別な SSL 要件を持っていないアカウント、または REQUIRE SSL オプションを含む GRANT ステートメントを使用して作成されたアカウントを使用して接続すると仮定します。推奨される SSL オプションセットとして、少なくとも --ssl-cert と --ssl-key を付けてサーバーを起動し、--ssl-ca を使用してクライアントを呼び出します。次のように、クライアントはセキュアに接続できます。

```
shell> mysql --ssl-ca=ca-cert.pem
```

クライアント証明書も指定されるように要求するには、REQUIRE X509 オプションを使用してアカウントを作成します。そのあと、クライアントは適切なクライアント鍵および証明書ファイルを指定する必要があります。そうしない場合、サーバーは接続を拒否します。

```
shell> mysql --ssl-ca=ca-cert.pem \
      --ssl-cert=client-cert.pem \
      --ssl-key=client-key.pem
```

SSL の使用を回避し、その他の SSL オプションをオーバーライドするには、--ssl=0 またはシノニム (--skip-ssl、--disable-ssl) を使用してクライアントプログラムを呼び出します。

```
shell> mysql --ssl=0
```

クライアントは Ssl_cipher ステータス変数の値をチェックすることで、サーバーとの現在の接続で SSL が使用されているかどうかを確認できます。SSL が使用されている場合は、値が空以外になります。それ以外の場合は、空になります。例:

```
mysql> SHOW STATUS LIKE 'Ssl_cipher';
+---------------+--------------------+
| Variable_name | Value              |
+---------------+--------------------+
| Ssl_cipher    | DHE-RSA-AES256-SHA |
+---------------+--------------------+
```

mysql クライアントの場合は、代わりに STATUS または \s コマンドを使用して、SSL 行をチェックすることもできます。

```
mysql> \s
...
SSL: Cipher in use is DHE-RSA-AES256-SHA
...
```

または、

```
mysql> \s
...
SSL: Not in use
...
```

C API を使用すると、アプリケーションプログラムで SSL を使用できます。

- セキュアな接続を確立するには、mysql_real_connect() を呼び出す前に、mysql_ssl_set() C API 関数を使用して、適切な証明書オプションを設定します。セクション23.7.7.68「mysql_ssl_set()」を参照してください。
- 接続が確立されたあとに、SSL が使用中であるかどうかを判断するには、mysql_get_ssl_cipher() を使用します。NULL 以外の戻り値は、セキュアな接続であることを示し、暗号化に使用される SSL 暗号の名前を示します。NULL の戻り値は、SSL が使用されていないことを示します。セクション23.7.7.33「mysql_get_ssl_cipher()」を参照してください。

レプリケーションでは C API が使用されるため、マスターサーバーとスレーブサーバー間でセキュアな接続を使用できます。セクション17.3.7「SSL を使用してレプリケーションをセットアップする」を参照してください。

6.3.10.4 SSL コマンドのオプション

このセクションでは、SSL を使用するかどうかを指定するオプションと、SSL 証明書および鍵ファイルの名前を指定するオプションについて説明します。これらのオプションは、コマンド行またはオプションファイルで指定できます。MySQL が SSL サポートを使用して構築されていない場合は、これらを使用できません。セクション6.3.10.2「SSL を使用するための MySQL の構成」を参照してください。推奨される使用例および接続がセキュアであるかどうかをチェックする方法については、セクション6.3.10.3「SSL 接続の使用」を参照してください。

表 6.16 SSL オプション/変数のサマリー

名前	コマンド行	オプションファイル	システム変数	ステータス変数	変数スコープ	動的
have_openssl			はい		グローバル	いいえ
have_ssl			はい		グローバル	いいえ
skip-ssl	はい	はい				
ssl	はい	はい				
ssl_ca	はい	はい	はい		グローバル	いいえ
ssl_capath	はい	はい	はい		グローバル	いいえ
ssl_cert	はい	はい	はい		グローバル	いいえ
ssl_cipher	はい	はい	はい		グローバル	いいえ
ssl_crl	はい	はい	はい		グローバル	いいえ
ssl_crlpath	はい	はい	はい		グローバル	いいえ
ssl_key	はい	はい	はい		グローバル	いいえ

- --ssl

  サーバーに対して、このオプションを指定すると、サーバーで SSL 接続が許可されますが、必須にはなりません。

  クライアントプログラムに対して、このオプションを指定すると、SSL を使用してサーバーに接続することがクライアントに許可されますが、必須にはなりません。したがって、このオプションだけでは、SSL 接続を使用させるのに十分ではありません。たとえば、このオプションをクライアントプログラムに対して指定するが、サーバーは SSL 接続が許可されるよう構成されていない場合は、暗号化されていない接続が使用されます。

  SSL 接続を有効にするための推奨される SSL オプションセットとして、サーバー側で少なくとも --ssl-cert と --ssl-key を使用し、クライアント側で --ssl-ca を使用します。セクション6.3.10.3「SSL 接続の使用」を参照してください。

  その他の --ssl-xxx オプションに関する説明で示したように、これらのオプションによって、--ssl が暗黙的に指定されている可能性があります。

  --ssl オプションをネストされた形式で指定すると、その他の SSL をオーバーライドし、SSL が使用されないように指定できます。これを行うには、--ssl=0 またはシノニム (--skip-ssl、--disable-ssl) としてオプションを指定します。たとえば、MySQL クライアントプログラムを呼び出し時にデフォルトで SSL 接続が使用されるように、SSL オプションをオプションファイルの [client] グループに指定することがあります。代わりに、暗号化されていない接続を使用するには、コマンド行で --skip-ssl を指定してクライアントプログラムを呼び出して、オプションファイルのオプションをオーバーライドします。

  MySQL アカウントに対して SSL 接続の使用を要求にするには、そのアカウントに対して、少なくとも REQUIRE SSL 句を含む GRANT ステートメントを発行します。MySQL で SSL 接続がサポートされ、サーバーとクライアントが適切な SSL オプションを付けて起動されていない場合は、アカウントの接続が拒否されます。

  REQUIRE 句は、その他の SSL 関連のオプションを許可します。これを使用すれば、REQUIRE SSL よりも厳密な要件を強制的に適用できます。さまざまな REQUIRE オプションを使用して構成されているアカウントを使用して接続するクライアントが、どの SSL コマンドオプションを指定する可能性があるのか、またはどのオプションを指定する必要があるのかについての詳細は、セクション13.7.1.4「GRANT 構文」で REQUIRE の説明を参照してください。

- --ssl-ca=file_name

  信頼できる SSL 証明書認証局のリストを含む PEM 形式のファイルへのパス。このオプションは、暗黙的に --ssl を示します。

クライアント接続を確立する際に SSL を使用して、サーバー証明書を認証しないようにクライアントに指示する場合は、--ssl-ca も --ssl-capath も指定しません。サーバーは引き続き、クライアントアカウントに対して GRANT ステートメントを使用して確立された適用可能な要件に応じてクライアントを検証し、サーバーの起動時に指定された --ssl-ca または --ssl-capath オプションの値を使用します。

- --ssl-capath=dir_name

  PEM 形式の信頼できる SSL 認証機関証明書を含むディレクトリへのパス。このオプションは、暗黙的に --ssl を示します。

  クライアント接続を確立する際に SSL を使用して、サーバー証明書を認証しないようにクライアントに指示する場合は、--ssl-ca も --ssl-capath も指定しません。サーバーは引き続き、クライアントアカウントに対して GRANT ステートメントを使用して確立された適用可能な要件に応じてクライアントを検証し、サーバーの起動時に指定された --ssl-ca または --ssl-capath オプションの値を使用します。

  OpenSSL サポートを使用して構築された MySQL 配布では、--ssl-capath オプションがサポートされています。yaSSL は、どのディレクトリにも表示されず、チェーン証明書ツリーをたどらないため、yaSSL を使用して構築された配布ではサポートされません。yaSSL では、CA 証明書ツリーのコンポーネントをすべて単一の CA 証明書ツリー内に含め、そのファイル内の各証明書は一意の SubjectName 値を持つ必要があります。このような yaSSL の制限を回避するには、証明書ツリーを構成する個々の証明書ファイルを新しいファイルに連結し、そのファイルを --ssl-ca オプションの値として指定します。

- --ssl-cert=file_name

  セキュアな接続を確立する際に使用される PEM 形式の SSL 証明書ファイルの名前。このオプションは、暗黙的に --ssl を示します。

- --ssl-cipher=cipher_list

  SSL 暗号化に使用する許可されている暗号のリスト。リスト内の暗号がサポートされていない場合は、SSL 接続が機能しません。このオプションは、暗黙的に --ssl を示します。

  移植性を最大にするには、cipher_list をコロンで区切った 1 つ以上の暗号名のリストで指定するようにしてください。この形式は、OpenSSL と yaSSL の両方で認識されます。例:

  ```
  --ssl-cipher=AES128-SHA
  --ssl-cipher=DHE-RSA-AES256-SHA:AES128-SHA
  ```

  http://www.openssl.org/docs/apps/ciphers.html にある OpenSSL のドキュメントで説明されているように、OpenSSL では、暗号を指定するためのより柔軟な構文がサポートされています。ただし、yaSSL ではサポートされてないため、yaSSL を使用して構築された MySQL 配布で拡張構文の使用を試みると失敗します。

  OpenSSL では、サーバーがリンクされているバージョンによって、サポートされている暗号が異なる場合があります。たとえば、リストには次の暗号が含まれている可能性があります。

  ```
  AES256-GCM-SHA384
  AES256-SHA
  AES256-SHA256
  CAMELLIA256-SHA
  DES-CBC3-SHA
  DHE-DSS-AES256-GCM-SHA384
  DHE-DSS-AES256-SHA
  DHE-DSS-AES256-SHA256
  DHE-DSS-CAMELLIA256-SHA
  DHE-RSA-AES256-GCM-SHA384
  DHE-RSA-AES256-SHA
  DHE-RSA-AES256-SHA256
  DHE-RSA-CAMELLIA256-SHA
  ECDH-ECDSA-AES256-GCM-SHA384
  ECDH-ECDSA-AES256-SHA
  ECDH-ECDSA-AES256-SHA384
  ECDH-ECDSA-DES-CBC3-SHA
  ECDH-RSA-AES256-GCM-SHA384
  ECDH-RSA-AES256-SHA
  ECDH-RSA-AES256-SHA384
  ECDH-RSA-DES-CBC3-SHA
  ECDHE-ECDSA-AES128-GCM-SHA256
  ECDHE-ECDSA-AES128-SHA
  ECDHE-ECDSA-AES128-SHA256
  ECDHE-ECDSA-AES256-GCM-SHA384
  ECDHE-ECDSA-AES256-SHA
  ECDHE-ECDSA-AES256-SHA384
  ```

```
ECDHE-ECDSA-DES-CBC3-SHA
ECDHE-RSA-AES128-GCM-SHA256
ECDHE-RSA-AES128-SHA
ECDHE-RSA-AES128-SHA256
ECDHE-RSA-AES256-GCM-SHA384
ECDHE-RSA-AES256-SHA
ECDHE-RSA-AES256-SHA384
ECDHE-RSA-DES-CBC3-SHA
EDH-DSS-DES-CBC3-SHA
EDH-RSA-DES-CBC3-SHA
PSK-3DES-EDE-CBC-SHA
PSK-AES256-CBC-SHA
SRP-DSS-3DES-EDE-CBC-SHA
SRP-DSS-AES-128-CBC-SHA
SRP-DSS-AES-256-CBC-SHA
SRP-RSA-3DES-EDE-CBC-SHA
SRP-RSA-AES-128-CBC-S
SRP-RSA-AES-256-CBC-SHA
```

yaSSL では、次の暗号がサポートされています。

```
AES128-RMD
AES128-SHA
AES256-RMD
AES256-SHA
DES-CBC-SHA
DES-CBC3-RMD
DES-CBC3-SHA
DHE-RSA-AES128-RMD
DHE-RSA-AES128-SHA
DHE-RSA-AES256-RMD
DHE-RSA-AES256-SHA
DHE-RSA-DES-CBC3-RMD
EDH-RSA-DES-CBC-SHA
EDH-RSA-DES-CBC3-SHA
RC4-MD5
RC4-SHA
```

特定のサーバーでサポートされている暗号を正確に確認するには、次のクエリーを使用して Ssl_cipher_list ステータス変数の値をチェックします。

```
SHOW STATUS LIKE 'Ssl_cipher_list';
```

- --ssl-crl=file_name

  証明書失効リストを含む PEM 形式のファイルへのパス。このオプションは、暗黙的に --ssl を示します。

  --ssl-crl も --ssl-crlpath も指定されていない場合は、CA パスに証明書失効リストが含まれていても、CRL チェックが実行されません。

  OpenSSL を使用して構築された MySQL 配布では、--ssl-crl オプションがサポートされています。yaSSL では証明書失効リストが機能しないため、yaSSL を使用して構築された配布ではサポートされません。

  このオプションは MySQL 5.6.3 で追加されました。

- --ssl-crlpath=dir_name

  証明書失効リストを含む PEM 形式のファイルが含まれるディレクトリへのパス。このオプションは、暗黙的に --ssl を示します。

  --ssl-crl も --ssl-crlpath も指定されていない場合は、CA パスに証明書失効リストが含まれていても、CRL チェックが実行されません。

  OpenSSL を使用して構築された MySQL 配布では、--ssl-crlpath オプションがサポートされています。yaSSL では証明書失効リストが機能しないため、yaSSL を使用して構築された配布ではサポートされません。

  このオプションは MySQL 5.6.3 で追加されました。

- --ssl-key=file_name

  セキュアな接続を確立する際に使用される PEM 形式の SSL 鍵ファイルの名前。このオプションは、暗黙的に --ssl を示します。

  MySQL 配布が OpenSSL または (MySQL 5.6.3 時点の) yaSSL を使用して構築され、鍵ファイルがパスフレーズで保護されている場合は、プログラムからパスフレーズを求めるプロンプトがユーザーに表示されます。パスワードは対話形式で指定する必要があり、ファイルには格納できません。パスフレーズが正しくない場合は、鍵を読み取ることができない場合と同様に、プログラムが続行されます。MySQL 5.6.3 よりも前では、MySQL 配布が yaSSL を使用して構築され、鍵ファイルがパスフレーズで保護されている場合は、エラーが発生します。

- --ssl-verify-server-cert

  このオプションはクライアントプログラムでのみ使用でき、サーバーでは使用できません。これにより、サーバーがクライアントに送信する証明書内のサーバーの Common Name 値がクライアントによってチェックされます。クライアントは、サーバーへの接続時にクライアントで使用されるホスト名と照合してその名前を検証し、一致が見つからない場合は接続に失敗します。この機能は、中間者攻撃を防ぐために使用できます。検証はデフォルトで無効になっています。

6.3.10.5 MySQL での SSL 証明書および鍵の設定

このセクションでは、MySQL サーバーおよびクライアントで使用される SSL 証明書および鍵ファイルを設定する方法を示します。1 番目の例は、コマンド行から使用する場合などの単純化された手順を示しています。2 番目では、より詳細なものを含むスクリプトを示します。最初の 2 つの例は、Unix で使用するためのものであり、どちらの例でも OpenSSL の一部である openssl コマンドが使用されます。3 番目の例では、Windows 上で SSL ファイルを設定する方法について説明します。

> **重要**
>
> 証明書および鍵ファイルを生成する際にどの方法を使用するのかには関係なく、サーバーおよびクライアントの証明書と鍵で使用される Common Name の値はそれぞれ、CA 証明書で使用されている Common Name の値と異なる必要があります。それ以外の場合は、OpenSSL を使用してコンパイルされたサーバーで証明書および鍵ファイルが機能しません。この場合の一般的なエラーは、次のとおりです。
>
> ```
> ERROR 2026 (HY000): SSL connection error:
> error:00000001:lib(0):func(0):reason(1)
> ```

例 1: Unix でコマンド行から SSL ファイルを作成する

次の例には、MySQL サーバーおよびクライアントの証明書および鍵ファイルを作成するためのコマンドセットを示します。openssl コマンドで複数のプロンプトに回答する必要があります。すべてのプロンプトに対して Enter キーを押せば、テストファイルを生成できます。本番環境用のファイルを生成するには、空でない回答を提供するようにしてください。

```
# Create clean environment
shell> rm -rf newcerts
shell> mkdir newcerts && cd newcerts

# Create CA certificate
shell> openssl genrsa 2048 > ca-key.pem
shell> openssl req -new -x509 -nodes -days 3600 \
       -key ca-key.pem -out ca-cert.pem

# Create server certificate, remove passphrase, and sign it
# server-cert.pem = public key, server-key.pem = private key
shell> openssl req -newkey rsa:2048 -days 3600 \
       -nodes -keyout server-key.pem -out server-req.pem
shell> openssl rsa -in server-key.pem -out server-key.pem
shell> openssl x509 -req -in server-req.pem -days 3600 \
       -CA ca-cert.pem -CAkey ca-key.pem -set_serial 01 -out server-cert.pem

# Create client certificate, remove passphrase, and sign it
# client-cert.pem = public key, client-key.pem = private key
shell> openssl req -newkey rsa:2048 -days 3600 \
       -nodes -keyout client-key.pem -out client-req.pem
shell> openssl rsa -in client-key.pem -out client-key.pem
shell> openssl x509 -req -in client-req.pem -days 3600 \
       -CA ca-cert.pem -CAkey ca-key.pem -set_serial 01 -out client-cert.pem
```

証明書が生成されたら、それらを確認します。

```
shell> openssl verify -CAfile ca-cert.pem server-cert.pem client-cert.pem
server-cert.pem: OK
client-cert.pem: OK
```

この時点で、次のようなファイルセットを使用できます。

- ca-cert.pem: これは、サーバー側とクライアント側で --ssl-ca への引数として使用します。(CA 証明書を使用する場合は、両側で同じものを指定する必要があります。)
- server-cert.pem、server-key.pem: これらは、サーバー側で --ssl-cert および --ssl-key への引数として使用します。
- client-cert.pem、client-key.pem: これらは、クライアント側で --ssl-cert および --ssl-key への引数として使用します。

ファイルを使用して SSL 接続をテストする方法については、セクション6.3.10.3「SSL 接続の使用」を参照してください。

例 2: Unix でスクリプトを使用して SSL ファイルを作成する

次に、MySQL に SSL 証明書および鍵ファイルを設定する方法を示したサンプルスクリプトを示します。スクリプトを実行したあとに、セクション6.3.10.3「SSL 接続の使用」で説明するように、ファイルを使用して SSL 接続をテストします。

```
DIR=`pwd`/openssl
PRIV=$DIR/private

mkdir $DIR $PRIV $DIR/newcerts
cp /usr/share/ssl/openssl.cnf $DIR
replace ./demoCA $DIR -- $DIR/openssl.cnf

# Create necessary files: $database, $serial and $new_certs_dir
# directory (optional)

touch $DIR/index.txt
echo "01" > $DIR/serial

#
# Generation of Certificate Authority(CA)
#

openssl req -new -x509 -keyout $PRIV/cakey.pem -out $DIR/ca-cert.pem \
   -days 3600 -config $DIR/openssl.cnf

# Sample output:
# Using configuration from /home/monty/openssl/openssl.cnf
# Generating a 1024 bit RSA private key
# ................++++++
# .........++++++
# writing new private key to '/home/monty/openssl/private/cakey.pem'
# Enter PEM pass phrase:
# Verifying password - Enter PEM pass phrase:
# -----
# You are about to be asked to enter information that will be
# incorporated into your certificate request.
# What you are about to enter is what is called a Distinguished Name
# or a DN.
# There are quite a few fields but you can leave some blank
# For some fields there will be a default value,
# If you enter '.', the field will be left blank.
# -----
# Country Name (2 letter code) [AU]:FI
# State or Province Name (full name) [Some-State]:.
# Locality Name (eg, city) []:
# Organization Name (eg, company) [Internet Widgits Pty Ltd]:MySQL AB
# Organizational Unit Name (eg, section) []:
# Common Name (eg, YOUR name) []:MySQL admin
# Email Address []:

#
# Create server request and key
#
openssl req -new -keyout $DIR/server-key.pem -out \
   $DIR/server-req.pem -days 3600 -config $DIR/openssl.cnf

# Sample output:
# Using configuration from /home/monty/openssl/openssl.cnf
# Generating a 1024 bit RSA private key
```

```
# ..++++++
# .........++++++
# writing new private key to '/home/monty/openssl/server-key.pem'
# Enter PEM pass phrase:
# Verifying password - Enter PEM pass phrase:
# -----
# You are about to be asked to enter information that will be
# incorporated into your certificate request.
# What you are about to enter is what is called a Distinguished Name
# or a DN.
# There are quite a few fields but you can leave some blank
# For some fields there will be a default value,
# If you enter '.', the field will be left blank.
# -----
# Country Name (2 letter code) [AU]:FI
# State or Province Name (full name) [Some-State]:.
# Locality Name (eg, city) []:
# Organization Name (eg, company) [Internet Widgits Pty Ltd]:MySQL AB
# Organizational Unit Name (eg, section) []:
# Common Name (eg, YOUR name) []:MySQL server
# Email Address []:
#
# Please enter the following 'extra' attributes
# to be sent with your certificate request
# A challenge password []:
# An optional company name []:

#
# Remove the passphrase from the key
#
openssl rsa -in $DIR/server-key.pem -out $DIR/server-key.pem

#
# Sign server cert
#
openssl ca -cert $DIR/ca-cert.pem -policy policy_anything \
   -out $DIR/server-cert.pem -config $DIR/openssl.cnf \
   -infiles $DIR/server-req.pem

# Sample output:
# Using configuration from /home/monty/openssl/openssl.cnf
# Enter PEM pass phrase:
# Check that the request matches the signature
# Signature ok
# The Subjects Distinguished Name is as follows
# countryName           :PRINTABLE:'FI'
# organizationName      :PRINTABLE:'MySQL AB'
# commonName            :PRINTABLE:'MySQL admin'
# Certificate is to be certified until Sep 13 14:22:46 2003 GMT
# (365 days)
# Sign the certificate? [y/n]:y
#
#
# 1 out of 1 certificate requests certified, commit? [y/n]y
# Write out database with 1 new entries
# Data Base Updated

#
# Create client request and key
#
openssl req -new -keyout $DIR/client-key.pem -out \
   $DIR/client-req.pem -days 3600 -config $DIR/openssl.cnf

# Sample output:
# Using configuration from /home/monty/openssl/openssl.cnf
# Generating a 1024 bit RSA private key
# ..................................++++++
# ..............................................++++++
# writing new private key to '/home/monty/openssl/client-key.pem'
# Enter PEM pass phrase:
# Verifying password - Enter PEM pass phrase:
# -----
# You are about to be asked to enter information that will be
# incorporated into your certificate request.
# What you are about to enter is what is called a Distinguished Name
# or a DN.
# There are quite a few fields but you can leave some blank
# For some fields there will be a default value,
# If you enter '.', the field will be left blank.
# -----
```

```
# Country Name (2 letter code) [AU]:FI
# State or Province Name (full name) [Some-State]:.
# Locality Name (eg, city) []:
# Organization Name (eg, company) [Internet Widgits Pty Ltd]:MySQL AB
# Organizational Unit Name (eg, section) []:
# Common Name (eg, YOUR name) []:MySQL user
# Email Address []:
#
# Please enter the following 'extra' attributes
# to be sent with your certificate request
# A challenge password []:
# An optional company name []:

#
# Remove the passphrase from the key
#
openssl rsa -in $DIR/client-key.pem -out $DIR/client-key.pem

#
# Sign client cert
#

openssl ca -cert $DIR/ca-cert.pem -policy policy_anything \
    -out $DIR/client-cert.pem -config $DIR/openssl.cnf \
    -infiles $DIR/client-req.pem

# Sample output:
# Using configuration from /home/monty/openssl/openssl.cnf
# Enter PEM pass phrase:
# Check that the request matches the signature
# Signature ok
# The Subjects Distinguished Name is as follows
# countryName           :PRINTABLE:'FI'
# organizationName      :PRINTABLE:'MySQL AB'
# commonName            :PRINTABLE:'MySQL user'
# Certificate is to be certified until Sep 13 16:45:17 2003 GMT
# (365 days)
# Sign the certificate? [y/n]:y
#
#
# 1 out of 1 certificate requests certified, commit? [y/n]y
# Write out database with 1 new entries
# Data Base Updated

#
# Create a my.cnf file that you can use to test the certificates
#

cat <<EOF > $DIR/my.cnf
[client]
ssl-ca=$DIR/ca-cert.pem
ssl-cert=$DIR/client-cert.pem
ssl-key=$DIR/client-key.pem
[mysqld]
ssl-ca=$DIR/ca-cert.pem
ssl-cert=$DIR/server-cert.pem
ssl-key=$DIR/server-key.pem
EOF
```

例 3: Windows で SSL ファイルを作成する

Windows 用の OpenSSL がシステムにインストールされていない場合は、それをダウンロードします。次に、使用可能なパッケージの概要を示します。

```
http://www.slproweb.com/products/Win32OpenSSL.html
```

アーキテクチャー (32 ビットまたは 64 ビット) に応じて、Win32 OpenSSL Light または Win64 OpenSSL Light パッケージを選択します。デフォルトのインストール場所は、ダウンロードしたパッケージに応じて、C:\OpenSSL-Win32 または C:\OpenSSL-Win64 です。次の手順では、デフォルトの場所が C:\OpenSSL-Win32 であることが前提となっています。64 ビットのパッケージを使用している場合は、必要に応じて、これを変更します。

設定中に「...critical component is missing: Microsoft Visual C++ 2008 Redistributables」というメッセージが発生した場合は、設定を取り消し、アーキテクチャー (32 ビットまたは 64 ビット) に応じて、次のパッケージのいずれかをダウンロードします。

- Visual C++ 2008 Redistributables (x86) (次の場所で入手可能):

```
http://www.microsoft.com/downloads/details.aspx?familyid=9B2DA534-3E03-4391-8A4D-074B9F2BC1BF
```

- Visual C++ 2008 Redistributables (x64) (次の場所で入手可能):

```
http://www.microsoft.com/downloads/details.aspx?familyid=bd2a6171-e2d6-4230-b809-9a8d7548c1b6
```

追加のパッケージをインストールしたら、OpenSSL の設定手順を再開します。

インストール時に、インストールパスとしてデフォルトの C:\OpenSSL-Win32 のままにして、デフォルトの「Copy OpenSSL DLL files to the Windows system directory」オプションも選択されたままにします。

インストールが完了したら、サーバーの Windows システムパス変数に C:\OpenSSL-Win32\bin を追加します。

1. Windows デスクトップ上で「マイ コンピューター」アイコンを右クリックして、「プロパティ」を選択します。
2. 表示された「システムのプロパティ」メニューから「詳細設定」タブを選択し、「環境変数」ボタンをクリックします。
3. 「システム変数」で「パス」を選択してから、「編集」ボタンをクリックします。「システム変数の編集」ダイアログが表示されるはずです。
4. 末尾に「;C:\OpenSSL-Win32\bin」を追加します (セミコロンに注意してください)。
5. 「OK」を 3 回押します。
6. 新しいコマンドコンソール (Start>Run>cmd.exe) を開いて、OpenSSL が使用可能であることを確認することで、OpenSSL が Path 変数に正しく統合されたことをチェックします。

```
Microsoft Windows [Version ...]
Copyright (c) 2006 Microsoft Corporation. All rights reserved.

C:\Windows\system32>cd \

C:\>openssl
OpenSSL> exit <<< If you see the OpenSSL prompt, installation was successful.

C:\>
```

Windows のバージョンに応じて、前述のパス設定手順が多少異なる場合があります。

OpenSSL がインストールされたら、(このセクションの前半で示した) 例 1 と同様の手順を次のように変更して使用します。

- 次のように Unix コマンドを変更します。

```
# Create clean environment
shell> rm -rf newcerts
shell> mkdir newcerts && cd newcerts
```

Windows では、代わりに次のコマンドを使用します。

```
# Create clean environment
shell> md c:\newcerts
shell> cd c:\newcerts
```

- コマンド行の末尾に「\」文字が表示されたら、この「\」文字を削除し、コマンド行をすべて 1 行で入力する必要があります。

証明書および鍵ファイルが生成されたあとに、それらを使用して SSL 接続をテストする方法については、セクション6.3.10.3「SSL 接続の使用」を参照してください。

6.3.11 SSH を使用した Windows から MySQL へのリモート接続

このセクションでは、SSH を使用してリモートの MySQL サーバーへのセキュアな接続を確立する方法について説明します。元の情報は、David Carlson <dcarlson@mplcomm.com> によって提供されました。

1. Windows マシン上に SSH クライアントをインストールします。SSH クライアントの比較については、http://en.wikipedia.org/wiki/Comparison_of_SSH_clientsを参照してください。
2. Windows SSH クライアントを起動します。Host_Name = yourmysqlserver_URL_or_IP を設定します。サーバーにログインする userid=your_userid を設定します。この userid 値は、MySQL アカウントのユーザー名と同じでない可能性があります。

3. ポートフォワーディングを設定します。リモート転送 (local_port: 3306、remote_host: yourmysqlservername_or_ip、remote_port: 3306 を設定します) とローカル転送 (port: 3306、host: localhost、remote port: 3306 を設定します) のいずれかを実行します。
4. すべてを保存します。そうしない場合は、次回やり直す必要があります。
5. 作成した SSH セッションを使用して、サーバーにログインします。
6. Windows マシン上で、いくつかの ODBC アプリケーション (Access など) を起動します。
7. 通常と同じ方法で、Windows で新しいファイルを作成し、ODBC ドライバを使用して MySQL へのリンクを作成します。ただし、MySQL ホストサーバーでは、yourmysqlservername ではなく、localhost に入力します。

この時点で、MySQL への ODBC 接続が SSH を使用して暗号化されているはずです。

6.3.12 MySQL Enterprise Audit ログプラグイン

> **注記**
>
> MySQL Enterprise Audit は、商用の拡張機能です。商用の製品 (MySQL Enterprise Edition) についてさらに学習するには、https://www.mysql.com/products/ を参照してください。

MySQL 5.6.10 の時点で、MySQL Enterprise Edition には、audit_log という名前のサーバープラグインを使用して実装された MySQL Enterprise Audit が含まれています。MySQL Enterprise Audit では、特定の MySQL サーバーで実行される接続およびクエリーアクティビティーのポリシーベースの標準モニタリングおよびロギングを有効にする際に、MySQL Audit API が使用されます。MySQL Enterprise Audit は、オラクルの監査仕様を満たすように設計されており、内部および外部の規制ガイドラインの両方に管理されているアプリケーションに対して、そのままで簡単に使用できる監査およびコンプライアンスのソリューションを提供しています。

インストール時に監査プラグインを使用すると、MySQL サーバーはサーバーアクティビティーの監査レコードを含むログファイルを生成できます。ログの内容には、クライアントが接続および切断したとき、および接続中に実行されたアクション (アクセスされたデータベースやテーブルなど) が含まれます。

プラグインをインストールすると (セクション6.3.12.1「監査ログプラグインのインストール」を参照してください)、監査ログファイルが書き込まれます。デフォルトでは、そのファイルはサーバーのデータディレクトリ内の audit.log という名前です。ファイルの名前を変更するには、サーバーの起動時に audit_log_file システム変数を設定します。

監査ログファイルの内容は暗号化されません。セクション6.3.12.2「監査ログプラグインのセキュリティーに関する考慮事項」を参照してください。

監査ログファイルは、<AUDIT_RECORD> 要素としてエンコードされた監査可能なイベントとともに、XML 形式で書き込まれます。ファイル形式を選択するには、サーバーの起動時に audit_log_format システム変数を設定します。ファイルの形式および内容についての詳細は、セクション6.3.12.3「監査ログファイル」を参照してください。

audit_log によってログファイルに書き込まれる情報の内容を制御するには、audit_log_policy システム変数を設定します。この変数はデフォルトで、ALL (監査可能なイベントをすべて書き込む) に設定されていますが、ログインまたはクエリーイベントのログのみを記録する LOGINS または QUERIES や、ロギングを無効にする NONE の値も許可されています。

ロギングが発生する方法の制御についての詳細は、セクション6.3.12.4「監査ログプラグインのロギング制御」を参照してください。監査ログプラグインを構成する際に使用されるパラメータについては、セクション6.3.12.6「監査ログプラグインのオプションおよびシステム変数」を参照してください。

audit_log プラグインが有効になっている場合は、パフォーマンススキーマ (第22章「MySQL パフォーマンススキーマ」を参照してください) に監査ログプラグイン用のインストゥルメンテーションが追加されます。関連するインストゥルメントを識別するには、次のクエリーを使用します。

```
SELECT NAME FROM performance_schema.setup_instruments
WHERE NAME LIKE '%/alog/%';
```

古い監査ログプラグインのバージョンからの変更

MySQL 5.6.14 では、Oracle Audit Vault との互換性を高くするために、監査ログプラグインにいくつかの変更が行われました。

新しい監査ログファイルの形式が実装されました。audit_log_format システム変数を使用すれば、古い形式または新しい形式を選択できます。この変数では、OLD および NEW (デフォルトは OLD) の値が許可されています。2 つの形式の相違点は次のとおりです。

- 属性を使用した古い形式で書き込まれた <AUDIT_RECORD> 要素内の情報は、サブ要素を使用した新しい形式で書き込まれます。
- 新しい形式には、<AUDIT_RECORD> 要素の詳細な情報が含まれています。各要素には、一意の識別子を提供する RECORD_ID 値が含まれています。TIMESTAMP 値には、タイムゾーンの情報が含まれています。クエリーレコードには、HOST、IP、OS_LOGIN、USER の情報、および COMMAND_CLASS と STATUS_CODE の値が含まれています。

古い <AUDIT_RECORD> 形式の例:

```
<AUDIT_RECORD
 TIMESTAMP="2013-09-15T15:27:27"
 NAME="Query"
 CONNECTION_ID="3"
 STATUS="0"
 SQLTEXT="SELECT 1"
/>
```

新しい <AUDIT_RECORD> 形式の例:

```
<AUDIT_RECORD>
 <TIMESTAMP>2013-09-15T15:27:27 UTC</TIMESTAMP>
 <RECORD_ID>3998_2013-09-15T15:27:27</RECORD_ID>
 <NAME>Query</NAME>
 <CONNECTION_ID>3</CONNECTION_ID>
 <STATUS>0</STATUS>
 <STATUS_CODE>0</STATUS_CODE>
 <USER>root[root] @ localhost [127.0.0.1]</USER>
 <OS_LOGIN></OS_LOGIN>
 <HOST>localhost</HOST>
 <IP>127.0.0.1</IP>
 <COMMAND_CLASS>select</COMMAND_CLASS>
 <SQLTEXT>SELECT 1</SQLTEXT>
</AUDIT_RECORD>
```

監査ログプラグインによって監査ログファイルがローテーションされると、別のファイル名形式が使用されます。audit.log という名前のログファイルの場合、以前はプラグインによってファイル名が audit.log.TIMESTAMP に変更されました。現在は、プラグインによってファイル名は XML ファイルであることを示す audit.log.TIMESTAMP.xml に変更されます。

audit_log_format の値を変更する場合は、次の手順を使用して、ある形式のログエントリが別の形式のエントリを含む既存のログファイルに書き込まれることを回避します。

1. サーバーを停止します。
2. 現在の監査ログファイルの名前を手動で変更します。
3. 新しい audit_log_format の値でサーバーを再起動します。監査ログプラグインによって、選択した形式のログエントリを含む新しいログファイルが作成されます。

監査プラグインを記述するための API も変更されました。mysql_event_general の構造には、クライアントのホスト名と IP アドレス、コマンドのクラス、および外部ユーザーを表す新しいメンバーが含まれています。詳細については、セクション24.2.4.8「監査プラグインの作成」を参照してください。

6.3.12.1 監査ログプラグインのインストール

監査ログプラグインは、audit_log という名前です。サーバーで使用可能にするには、プラグインライブラリオブジェクトファイルを MySQL プラグインディレクトリ (plugin_dir システム変数で指定されたディレクトリ) に配置する必要があります。必要に応じて、サーバーの起動時に、プラグインディレクトリの場所をサーバーに指示する plugin_dir の値を設定します。

サーバーの起動時にプラグインをロードするには、--plugin-load オプションを使用して、プラグインを含むオブジェクトファイルの名前を指定します。このプラグインのロード方式では、サーバーを起動するたびにオプションを指定する必要があります。たとえば、my.cnf ファイルに次の行を挿入します。

```
[mysqld]
plugin-load=audit_log.so
```

システム上のオブジェクトファイルのサフィクスが .so とは異なる場合、正しいサフィクスに置き換えてください (たとえば Windows の場合は .dll)。

また、実行時にプラグインを登録するには、次のステートメントを使用します (必要に応じて、サフィクスを変更します)。

```
mysql> INSTALL PLUGIN audit_log SONAME 'audit_log.so';
```

INSTALL PLUGIN は、プラグインをロードします。また、後続の通常のサーバー起動のたびにプラグインがロードされるように、そのプラグインを mysql.plugins テーブルに登録します。

プラグインが --plugin-load を使用してロードされている場合、または INSTALL PLUGIN を使用して事前に登録されている場合は、サーバーの起動時に --audit-log オプションを使用すると、プラグインのアクティブ化を制御できます。たとえば、プラグインをロードして、実行時に削除されないようにするには、次のオプションを使用します。

```
[mysqld]
plugin-load=audit_log.so
audit-log=FORCE_PLUS_PERMANENT
```

監査プラグインを使用せずにサーバーが実行されることを回避する必要がある場合は、FORCE または FORCE_PLUS_PERMANENT の値とともに --audit-log を使用して、プラグインが正常に初期化されない場合にサーバーの起動を強制的に失敗させます。

プラグインのインストールについての一般的な情報は、セクション5.1.8「サーバープラグイン」を参照してください。プラグインのインストールを確認するには、INFORMATION_SCHEMA.PLUGINS テーブルを調査するか、または SHOW PLUGINS ステートメントを使用します。セクション5.1.8.2「サーバープラグイン情報の取得」を参照してください。

監査ログファイルの内容は暗号化されません。セクション6.3.12.2「監査ログプラグインのセキュリティーに関する考慮事項」を参照してください。

audit_log プラグインの操作を構成する際に使用されるパラメータに関する追加情報については、セクション6.3.12.6「監査ログプラグインのオプションおよびシステム変数」を参照してください。

6.3.12.2 監査ログプラグインのセキュリティーに関する考慮事項

audit_log 監査ログプラグインによって生成された監査ログファイルの内容は、暗号化されません。この内容には、SQL ステートメントのテキストなどの機密情報が含まれている可能性があります。セキュリティー上の理由から、このファイルは MySQL サーバーおよびログを表示する正当な理由を持つユーザーからのみアクセス可能なディレクトリに書き込むようにしてください。デフォルトのファイルは、データディレクトリ内の audit.log です。これは、サーバーの起動時に audit_log_file システム変数を設定することで変更できます。

6.3.12.3 監査ログファイル

監査ログファイルの内容は暗号化されません。セクション6.3.12.2「監査ログプラグインのセキュリティーに関する考慮事項」を参照してください。

監査ログファイルは、UTF-8 (1 文字当たり最大 4 バイト) を使用した XML として記述されています。ルート要素は、<AUDIT> です。監査プラグインログの終了時に、ルート要素の終了タグ </AUDIT> が書き込まれます。そのため、このタグは、プラグインがアクティブになっている間はファイル内に存在しません。

ルート要素には、<AUDIT_RECORD> 要素が含まれています。この要素のそれぞれは、監査対象のイベントに関する情報を提供します。

MySQL 5.6.14 では、Oracle Audit Vault との互換性を高くするために、新しい監査ログファイルの形式が導入されました。audit_log_format システム変数を使用すれば、古い形式または新しい形式を選択できます。この変数では、OLD および NEW (デフォルトは OLD) の値が許可されています。

audit_log_format の値を変更する場合は、次の手順を使用して、ある形式のログエントリが別の形式のエントリを含む既存のログファイルに書き込まれることを回避します。

1. サーバーを停止します。
2. 現在の監査ログファイルの名前を手動で変更します。
3. 新しい audit_log_format の値でサーバーを再起動します。監査ログプラグインによって、選択した形式のログエントリを含む新しいログファイルが作成されます。

次に、デフォルトの (古い) 形式のサンプルログファイルを示します。ただし、読みやすくするために形式が若干変更されています。

```
<?xml version="1.0" encoding="UTF-8"?>
<AUDIT>
```

```
 <AUDIT_RECORD
   TIMESTAMP="2012-08-02T14:52:12"
   NAME="Audit"
   SERVER_ID="1"
   VERSION="1"
   STARTUP_OPTIONS="--port=3306"
   OS_VERSION="i686-Linux"
   MYSQL_VERSION="5.6.10-log"/>
 <AUDIT_RECORD
   TIMESTAMP="2012-08-02T14:52:41"
   NAME="Connect"
   CONNECTION_ID="1"
   STATUS="0"
   USER="root"
   PRIV_USER="root"
   OS_LOGIN=""
   PROXY_USER=""
   HOST="localhost"
   IP="127.0.0.1"
   DB=""/>
 <AUDIT_RECORD
   TIMESTAMP="2012-08-02T14:53:45"
   NAME="Query"
   CONNECTION_ID="1"
   STATUS="0"
   SQLTEXT="INSERT INTO t1 () VALUES()"/>
 <AUDIT_RECORD
   TIMESTAMP="2012-08-02T14:53:51"
   NAME="Quit"
   CONNECTION_ID="1"
   STATUS="0"/>
 <AUDIT_RECORD
   TIMESTAMP="2012-08-06T14:21:03"
   NAME="NoAudit"
   SERVER_ID="1"/>
</AUDIT>
```

次に、新しい形式のサンプルログファイルを示します。ただし、読みやすくするために形式が若干変更されています。

```
<?xml version="1.0" encoding="UTF-8"?>
<AUDIT>
 <AUDIT_RECORD>
  <TIMESTAMP>2013-09-17T15:03:24 UTC</TIMESTAMP>
  <RECORD_ID>1_2013-09-17T15:03:24</RECORD_ID>
  <NAME>Audit</NAME>
  <SERVER_ID>1</SERVER_ID>
  <VERSION>1</VERSION>
  <STARTUP_OPTIONS>/usr/local/mysql/bin/mysqld
    --socket=/usr/local/mysql/mysql.sock
    --port=3306</STARTUP_OPTIONS>
  <OS_VERSION>x86_64-osx10.6</OS_VERSION>
  <MYSQL_VERSION>5.7.2-m12-log</MYSQL_VERSION>
 </AUDIT_RECORD>
 <AUDIT_RECORD>
  <TIMESTAMP>2013-09-17T15:03:40 UTC</TIMESTAMP>
  <RECORD_ID>2_2013-09-17T15:03:24</RECORD_ID>
  <NAME>Connect</NAME>
  <CONNECTION_ID>2</CONNECTION_ID>
  <STATUS>0</STATUS>
  <STATUS_CODE>0</STATUS_CODE>
  <USER>root</USER>
  <OS_LOGIN></OS_LOGIN>
  <HOST>localhost</HOST>
  <IP>127.0.0.1</IP>
  <COMMAND_CLASS>connect</COMMAND_CLASS>
  <PRIV_USER>root</PRIV_USER>
  <PROXY_USER></PROXY_USER>
  <DB>test</DB>
 </AUDIT_RECORD>

...

 <AUDIT_RECORD>
  <TIMESTAMP>2013-09-17T15:03:41 UTC</TIMESTAMP>
  <RECORD_ID>4_2013-09-17T15:03:24</RECORD_ID>
  <NAME>Query</NAME>
  <CONNECTION_ID>2</CONNECTION_ID>
  <STATUS>0</STATUS>
  <STATUS_CODE>0</STATUS_CODE>
```

```
 <USER>root[root] @ localhost [127.0.0.1]</USER>
 <OS_LOGIN></OS_LOGIN>
 <HOST>localhost</HOST>
 <IP>127.0.0.1</IP>
 <COMMAND_CLASS>drop_table</COMMAND_CLASS>
 <SQLTEXT>DROP TABLE IF EXISTS t</SQLTEXT>
</AUDIT_RECORD>
<AUDIT_RECORD>
 <TIMESTAMP>2013-09-17T15:03:41 UTC</TIMESTAMP>
 <RECORD_ID>5_2013-09-17T15:03:24</RECORD_ID>
 <NAME>Query</NAME>
 <CONNECTION_ID>2</CONNECTION_ID>
 <STATUS>0</STATUS>
 <STATUS_CODE>0</STATUS_CODE>
 <USER>root[root] @ localhost [127.0.0.1]</USER>
 <OS_LOGIN></OS_LOGIN>
 <HOST>localhost</HOST>
 <IP>127.0.0.1</IP>
 <COMMAND_CLASS>create_table</COMMAND_CLASS>
 <SQLTEXT>CREATE TABLE t (i INT)</SQLTEXT>
</AUDIT_RECORD>

...

 <AUDIT_RECORD>
 <TIMESTAMP>2013-09-17T15:03:41 UTC</TIMESTAMP>
 <RECORD_ID>7_2013-09-17T15:03:24</RECORD_ID>
 <NAME>Quit</NAME>
 <CONNECTION_ID>2</CONNECTION_ID>
 <STATUS>0</STATUS>
 <STATUS_CODE>0</STATUS_CODE>
 <USER></USER>
 <OS_LOGIN></OS_LOGIN>
 <HOST></HOST>
 <IP></IP>
 <COMMAND_CLASS>connect</COMMAND_CLASS>
</AUDIT_RECORD>

...

 <AUDIT_RECORD>
 <TIMESTAMP>2013-09-17T15:03:47 UTC</TIMESTAMP>
 <RECORD_ID>9_2013-09-17T15:03:24</RECORD_ID>
 <NAME>Shutdown</NAME>
 <CONNECTION_ID>3</CONNECTION_ID>
 <STATUS>0</STATUS>
 <STATUS_CODE>0</STATUS_CODE>
 <USER>root[root] @ localhost [127.0.0.1]</USER>
 <OS_LOGIN></OS_LOGIN>
 <HOST>localhost</HOST>
 <IP>127.0.0.1</IP>
 <COMMAND_CLASS></COMMAND_CLASS>
</AUDIT_RECORD>
<AUDIT_RECORD>
 <TIMESTAMP>2013-09-17T15:03:47 UTC</TIMESTAMP>
 <RECORD_ID>10_2013-09-17T15:03:24</RECORD_ID>
 <NAME>Quit</NAME>
 <CONNECTION_ID>3</CONNECTION_ID>
 <STATUS>0</STATUS>
 <STATUS_CODE>0</STATUS_CODE>
 <USER></USER>
 <OS_LOGIN></OS_LOGIN>
 <HOST></HOST>
 <IP></IP>
 <COMMAND_CLASS>connect</COMMAND_CLASS>
</AUDIT_RECORD>
<AUDIT_RECORD>
 <TIMESTAMP>2013-09-17T15:03:49 UTC</TIMESTAMP>
 <RECORD_ID>11_2013-09-17T15:03:24</RECORD_ID>
 <NAME>NoAudit</NAME>
 <SERVER_ID>1</SERVER_ID>
</AUDIT_RECORD>
</AUDIT>
```

<AUDIT_RECORD> 要素の属性には、次のような特性があります。

- 一部の属性はすべての要素に表示されますが、大部分はオプションであり、必ずしもすべての要素に表示されるとはかぎりません。
- 要素内の属性の順序は保証されません。

- 属性値は固定長ではありません。あとで属性の説明で示すように、長い値は切り捨てられる可能性があります。
- <、>、"、および & 文字は、それぞれ <、>、"、および & としてエンコードされます。NUL バイト (U+00) は、? 文字としてエンコードされます。
- XML 文字として有効でない文字は、数値の文字参照を使用してエンコードされます。有効な XML 文字は次のとおりです。

```
#x9 | #xA | #xD | [#x20-#xD7FF] | [#xE000-#xFFFD] | [#x10000-#x10FFFF]
```

新しい監査ログファイル形式

すべての <AUDIT_RECORD> 要素には、必須の要素セットが含まれています。監査レコードのタイプに応じて、その他のオプションの要素が表示される場合もあります。

次の要素は、すべての <AUDIT_RECORD> 要素で必須です。

- <NAME>

  監査イベントを生成した命令 (サーバーがクライアントから受信したコマンドなど) のタイプを表す文字列。

  例:

  ```
  <NAME>Query</NAME>
  ```

  一部の一般的な <NAME> 値:

  ```
  Audit    When auditing starts, which may be server startup time
  Connect  When a client connects, also known as logging in
  Query    An SQL statement (executed directly)
  Prepare  Preparation of an SQL statement; usually followed by Execute
  Execute  Execution of an SQL statement; usually follows Prepare
  Shutdown Server shutdown
  Quit     When a client disconnects
  NoAudit  Auditing has been turned off
  ```

  指定可能な値は Audit、Binlog Dump、Change user、Close stmt、Connect Out、Connect、Create DB、Daemon、Debug、Delayed insert、Drop DB、Execute、Fetch、Field List、Init DB、Kill、Long Data、NoAudit、Ping、Prepare、Processlist、Query、Quit、Refresh、Register Slave、Reset stmt、Set option、Shutdown、Sleep、Statistics、Table Dump、Time です。

  Audit と NoAudit を除いて、これらの値は mysql_com.h ヘッダーファイルに一覧表示された COM_xxx コマンドの値に対応します。たとえば、Create DB と Shutdown は、それぞれ COM_CREATE_DB と COM_SHUTDOWN に対応します。

- <RECORD_ID>

  監査レコードを表す一意の識別子。この値はシーケンス番号とタイムスタンプで構成され、形式は SEQ_TIMESTAMP です。シーケンス番号は、監査ログプラグインによって開かれ、レコードのログが記録されるたびに 1 ずつ増分されると、監査ログファイルのサイズに初期化されます。タイムスタンプは、監査ログプラグインによってファイルが開かれた時間を示す yyyy-mm-ddThh:mm:ss 形式の UTC 値です。

  例:

  ```
  <RECORD_ID>28743_2013-09-18T21:03:24</RECORD_ID>
  ```

- <TIMESTAMP>

  監査イベントが生成された日付と時間。たとえば、クライアントから受信された SQL ステートメントの実行に対応するイベントでは、<TIMESTAMP> 値はステートメントが受信されたときではなく、終了したあとに生成されます。この値の形式は、yyyy-mm-ddThh:mm:ss UTC (T の場合は小数点不可) です。この形式の末尾には、タイムゾーン指定子が含まれています。現在、タイムゾーンは常に UTC です。

  例:

  ```
  <TIMESTAMP>2013-09-17T15:03:49 UTC</TIMESTAMP>
  ```

次の要素は、<AUDIT_RECORD> 要素ではオプションです。これらの多くは、特定の <NAME> 値でのみ発生します。

- <COMMAND_CLASS>

実行されたアクションのタイプを示す文字列。

例:

```
<COMMAND_CLASS>drop_table</COMMAND_CLASS>
```

この値は、MySQL ソース配布の sql/mysqld.cc ファイル内の com_status_vars 配列から取得されます。これらは、このステートメントで表示されるステータス変数に対応します。

```
SHOW STATUS LIKE 'Com%';
```

- <CONNECTION_ID>

  クライアント接続識別子を表す符号なし整数。これは、セッション内の CONNECTION_ID() 関数の値と同じです。

  例:

  ```
  <CONNECTION_ID>127</CONNECTION_ID>
  ```

- <DB>

  デフォルトのデータベース名を表す文字列。この要素は、<NAME> 値が Connect または Change user である場合にのみ表示されます。

- <HOST>

  クライアントのホスト名を表す文字列。この要素は、<NAME> 値が Connect、Change user、または Query である場合にのみ表示されます。

  例:

  ```
  <HOST>localhost</HOST>
  ```

- <IP>

  クライアントの IP アドレスを表す文字列。この要素は、<NAME> 値が Connect、Change user、または Query である場合にのみ表示されます。

  例:

  ```
  <IP>127.0.0.1</IP>
  ```

- <MYSQL_VERSION>

  MySQL サーバーのバージョンを表す文字列。これは、セッション内の VERSION() 関数または version システム変数の値と同じです。この要素は、<NAME> 値が Audit である場合にのみ表示されます。

  例:

  ```
  <MYSQL_VERSION>5.7.1-m11-log</MYSQL_VERSION>
  ```

- <OS_LOGIN>

  外部ユーザーを表す文字列 (none の場合は空です)。たとえば、サーバーが外部の認証方式を使用してクライアントを認証する場合は、この値が <USER> の値と異なる可能性があります。この要素は、<NAME> 値が Connect、Change user、または Query である場合にのみ表示されます。

- <OS_VERSION>

  サーバーが構築された、または実行されているオペレーティングシステムを表す文字列。この要素は、<NAME> 値が Audit である場合にのみ表示されます。

  例:

  ```
  <OS_VERSION>x86_64-Linux</OS_VERSION>
  ```

- <PRIV_USER>

  サーバーがクライアントを認証する際に使用したユーザーを表す文字列。これは、サーバーが権限チェックを行う際に使用するユーザー名であり、<USER> の値とは異なる可能性があります。この要素は、<NAME> 値が Connect または Change user である場合にのみ表示されます。

- <PROXY_USER>

  プロキシユーザーを表す文字列。ユーザーのプロキシ処理が有効になっていない場合は、値が空です。この要素は、<NAME> 値が Connect または Change user である場合にのみ表示されます。

- <SERVER_ID>

  サーバー ID を表す符号なし整数。これは、server_id システム変数の値と同じです。この要素は、<NAME> 値が Audit または NoAudit である場合にのみ表示されます。

  例:

  ```
  <SERVER_ID>1</SERVER_ID>
  ```

- <SQLTEXT>

  SQL ステートメントのテキストを表す文字列。この値は、空にすることができます。長い値は、切り捨てられる可能性があります。この要素は、<NAME> 値が Query または Execute である場合にのみ表示されます。

  監査ログファイル自体などの文字列は、UTF-8 (1 文字当たり最大 4 バイト) を使用して記述されるため、この値が変換の結果となる場合があります。たとえば、元のステートメントは、SJIS 文字列としてクライアントから受信された可能性があります。

  例:

  ```
  <SQLTEXT>DELETE FROM t1</SQLTEXT>
  ```

- <STARTUP_OPTIONS>

  MySQL サーバーの起動時に、コマンド行またはオプションファイルで指定されたオプションを表す文字列。この要素は、<NAME> 値が Audit である場合にのみ表示されます。

  例:

  ```
  <STARTUP_OPTIONS>/usr/local/mysql/bin/mysqld
    --port=3306 --log-output=FILE</STARTUP_OPTIONS>
  ```

- <STATUS>

  コマンドのステータスを表す符号なし整数 (成功した場合は 0、エラーが発生した場合はゼロ以外)。これは、mysql_errno() C API 関数の値と同じです。

  監査ログには、SQLSTATE 値またはエラーメッセージが含まれていません。エラーコード、SQLSTATE 値、およびメッセージ間の関連性を確認する方法については、セクションB.3「サーバーのエラーコードおよびメッセージ」を参照してください。

  警告のログは記録されません。

  <STATUS> と異なる点については、<STATUS_CODE> の説明を参照してください。

  例:

  ```
  <STATUS>1051</STATUS>
  ```

- <STATUS_CODE>

  コマンドのステータスを表す符号なし整数 (成功した場合は 0、エラーが発生した場合は 1)。

  STATUS_CODE の値は、STATUS の値とは異なります。STATUS_CODE は、成功した場合は 0、エラーが発生した場合は 1 であり、Audit Vault の EZ_collector コンシューマとの互換性があります。STATUS

は、mysql_errno() C API 関数の値です。これは、成功した場合は 0、エラーが発生した場合はゼロ以外です。そのため、エラーが発生した場合、必ずしも 1 であるとはかぎりません。

例:

```
<STATUS_CODE>0</STATUS_CODE>
```

- <USER>

  クライアントによって送信されたユーザー名を表す文字列。これは、<PRIV_USER> の値とは異なる可能性があります。この要素は、<NAME> 値が Connect、Change user、または Query である場合にのみ表示されます。

  例:

```
<USER>root[root] @ localhost [127.0.0.1]</USER>
```

- <VERSION>

  監査ログファイル形式のバージョンを表す符号なし整数。この要素は、<NAME> 値が Audit である場合にのみ表示されます。

  例:

```
<VERSION>1</VERSION>
```

古い監査ログファイル形式

すべての <AUDIT_RECORD> 要素には、必須の属性セットが含まれています。監査レコードのタイプに応じて、その他のオプションの属性が表示される場合もあります。

次の属性は、すべての <AUDIT_RECORD> 要素で必須です。

- NAME

  監査イベントを生成した命令 (サーバーがクライアントから受信したコマンドなど) のタイプを表す文字列。

  例: NAME="Query"

  一部の一般的な NAME 値:

```
"Audit"    When auditing starts, which may be server startup time
"Connect"  When a client connects, also known as logging in
"Query"    An SQL statement (executed directly)
"Prepare"  Preparation of an SQL statement; usually followed by Execute
"Execute"  Execution of an SQL statement; usually follows Prepare
"Shutdown" Server shutdown
"Quit"     When a client disconnects
"NoAudit"  Auditing has been turned off
```

  指定可能な値は "Audit"、"Binlog Dump"、"Change user"、"Close stmt"、"Connect Out"、"Connect"、"Create DB"、"Daemon"、"Debug"、"Delayed insert"、"Drop DB"、"Execute"、"Fetch"、"Field List"、"Init DB"、"Kill"、"Long Data"、"NoAudit"、"Ping"、"Prepare"、"Processlist"、"Query"、"Quit"、"Refresh"、"Register Slave"、"Reset stmt"、"Set option"、"Shutdown"、"Sleep"、"Statistics"、"Table Dump"、"Time" です。

  "Audit" と "NoAudit" を除いて、これらの値は mysql_com.h ヘッダーファイルに一覧表示された COM_xxx コマンドの値に対応します。たとえば、"Create DB" と "Shutdown" は、それぞれ COM_CREATE_DB と COM_SHUTDOWN に対応します。

- TIMESTAMP

  監査イベントが生成された日付と時間。たとえば、クライアントから受信された SQL ステートメントの実行に対応するイベントでは、TIMESTAMP 値はステートメントが受信されたときではなく、終了したあとに生成されます。この値は、yyyy-mm-ddThh:mm:ss 形式 (T の場合は小数点不可) の UTC です。

  例: TIMESTAMP="2012-08-09T12:55:16"

次の属性は、<AUDIT_RECORD> 要素ではオプションです。これらの多くは、NAME 属性の特定の値を含む要素でのみ発生します。

- CONNECTION_ID

クライアント接続識別子を表す符号なし整数。これは、セッション内の CONNECTION_ID() 関数の値と同じです。

例: CONNECTION_ID="127"

- DB

  デフォルトのデータベース名を表す文字列。この属性は、NAME 値が "Connect" または "Change user" である場合にのみ表示されます。

- HOST

  クライアントのホスト名を表す文字列。この属性は、NAME 値が "Connect" または "Change user" である場合にのみ表示されます。

  例: HOST="localhost"

- IP

  クライアントの IP アドレスを表す文字列。この属性は、NAME 値が "Connect" または "Change user" である場合にのみ表示されます。

  例: IP="127.0.0.1"

- MYSQL_VERSION

  MySQL サーバーのバージョンを表す文字列。これは、セッション内の VERSION() 関数または version システム変数の値と同じです。この属性は、NAME 値が "Audit" である場合にのみ表示されます。

  例: MYSQL_VERSION="5.6.11-log"

- OS_LOGIN

  外部ユーザーを表す文字列 (none の場合は空です)。たとえば、サーバーが外部の認証方式を使用してクライアントを認証する場合は、この値が USER と異なる可能性があります。この属性は、NAME 値が "Connect" または "Change user" である場合にのみ表示されます。

- OS_VERSION

  サーバーが構築された、または実行されているオペレーティングシステムを表す文字列。この属性は、NAME 値が "Audit" である場合にのみ表示されます。

  例: OS_VERSION="x86_64-Linux"

- PRIV_USER

  サーバーがクライアントを認証する際に使用したユーザーを表す文字列。これは、サーバーが権限チェックを行う際に使用するユーザー名であり、USER の値とは異なる可能性があります。この属性は、NAME 値が "Connect" または "Change user" である場合にのみ表示されます。

- PROXY_USER

  プロキシユーザーを表す文字列。ユーザーのプロキシ処理が有効になっていない場合は、値が空です。この属性は、NAME 値が "Connect" または "Change user" である場合にのみ表示されます。

- SERVER_ID

  サーバー ID を表す符号なし整数。これは、server_id システム変数の値と同じです。この属性は、NAME 値が "Audit" または "NoAudit" である場合にのみ表示されます。

  例: SERVER_ID="1"

- SQLTEXT

  SQL ステートメントのテキストを表す文字列。この値は、空にすることができます。長い値は、切り捨てられる可能性があります。この属性は、NAME 値が "Query" または "Execute" である場合にのみ表示されます。

  監査ログファイル自体などの文字列は、UTF-8 (1 文字当たり最大 4 バイト) を使用して記述されるため、この値が変換の結果となる場合があります。たとえば、元のステートメントは、SJIS 文字列としてクライアントから受信された可能性があります。

  例: SQLTEXT="DELETE FROM t1"

- STARTUP_OPTIONS

  MySQL サーバーの起動時に、コマンド行またはオプションファイルで指定されたオプションを表す文字列。この属性は、NAME 値が "Audit" である場合にのみ表示されます。

  例: STARTUP_OPTIONS="--port=3306 --log-output=FILE"

- STATUS

  コマンドのステータスを表す符号なし整数 (成功した場合は 0、エラーが発生した場合はゼロ以外)。これは、mysql_errno() C API 関数の値と同じです。

  監査ログには、SQLSTATE 値またはエラーメッセージが含まれていません。エラーコード、SQLSTATE 値、およびメッセージ間の関連性を確認する方法については、セクションB.3「サーバーのエラーコードおよびメッセージ」を参照してください。

  警告のログは記録されません。

  例: STATUS="1051"

- USER

  クライアントによって送信されたユーザー名を表す文字列。これは、PRIV_USER の値とは異なる可能性があります。この属性は、NAME 値が "Connect" または "Change user" である場合にのみ表示されます。

- VERSION

  監査ログファイル形式のバージョンを表す符号なし整数。この属性は、NAME 値が "Audit" である場合にのみ表示されます。

  例: VERSION="1"

6.3.12.4 監査ログプラグインのロギング制御

このセクションでは、audit_log プラグインでロギングを実行する方法、およびロギングの発生方法を制御するシステム変数について説明します。セクション6.3.12.3「監査ログファイル」で説明したログファイル形式に精通していることが前提となっています。

監査ログプラグインはそのログファイルを開くと、XML 宣言および開始ルート要素タグ <AUDIT> を書き込む必要があるかどうかをチェックし、その場合はそれらのタグを書き込みます。監査ログプラグインが終了すると、終了タグ </AUDIT> をファイルに書き込みます。

ログファイルを開くときに存在すれば、プラグインはファイルが </AUDIT> タグで終了しているかどうかをチェックし、その場合はそれを切り捨ててから、任意の <AUDIT_RECORD> 要素を書き込みます。ログファイルが存在するが、</AUDIT> タグで終了していない場合や、</AUDIT> タグを切り捨てることができない場合、プラグインではファイル形式が不正であり、初期化に失敗したとみなされます。これは、サーバーがクラッシュした場合や、監査ログプラグインの実行中に強制終了された場合に発生する可能性があります。ロギングは、問題が修正されるまで発生しません。エラーログをチェックして、診断情報を確認してください。

```
[ERROR] Plugin 'audit_log' init function returned error.
```

この問題に対処するには、不正な形式のログファイルを削除するか、ファイル名を変更してから、サーバーを再起動する必要があります。

MySQL サーバーは、クライアントから受信された SQL ステートメントの実行が完了するときなど、監査可能なイベントが発生するたびに監査ログプラグインを呼び出して、<AUDIT_RECORD> 要素を書き込みます。一般に、サーバーの起動後に最初に書き込まれた <AUDIT_RECORD> 要素には、サーバーの説明および起動オプ

ションが含まれています。そのあとの要素は、クライアントの接続および切断イベント、SQL ステートメントの実行などのイベントを表します。最上位のステートメントのみのログが記録され、トリガーやストアドプロシージャーなどのストアドプログラム内のステートメントのログは記録されません。LOAD DATA INFILE などのステートメントから参照されるファイルの内容は、記録されません。

ロギングの発生方法に対する制御を許可するために、audit_log プラグインには、次に説明するような複数のシステム変数が用意されています。詳細については、セクション6.3.12.6「監査ログプラグインのオプションおよびシステム変数」を参照してください。

監査ログファイル名の指定

監査ログファイル名を制御するには、サーバーの起動時に audit_log_file システム変数を設定します。デフォルトでは、サーバーのデータディレクトリ内の audit.log という名前です。セキュリティー上の理由から、監査ログファイルは、MySQL サーバーおよびログを表示する正当な理由を持つユーザーにのみアクセス可能なディレクトリに書き込まれるべきです。

監査ロギングの戦略

監査プラグインは、ログの書き込みに関する複数の戦略のいずれかを使用できます。戦略を指定するには、サーバーの起動時に audit_log_strategy システム変数を設定します。デフォルトでは、戦略の値は ASYNCHRONOUS であり、プラグインは非同期的にログをバッファーに記録し、バッファーがいっぱいの場合は待機します。ファイルシステムのキャッシュ処理を使用するか (SEMISYNCHRONOUS)、各書き込みリクエストのあとに sync() を呼び出して出力を強制すれば (SYNCHRONOUS)、待機しないように (PERFORMANCE)、または同期的にログを記録するようにプラグインに指示できます。

非同期ロギングの戦略には、次のような特性があります。

- サーバーのパフォーマンスと拡張性への影響が最小限です。
- できるかぎり最短の時間 (つまり、バッファーを割り当てる時間とそのバッファーにイベントをコピーする時間を足した時間) で、監査イベントを生成するスレッドをブロックします。
- 出力はバッファーに書き込まれます。個別のスレッドがバッファーからログファイルへの書き込みに対処します。

PERFORMANCE 戦略のデメリットは、バッファーがいっぱいの場合にイベントが破棄される点です。負荷の高いサーバーでは、監査ログでイベントが欠落する可能性が高くなります。

非同期ロギングを使用すると、ファイルへの書き込み中に問題が発生した場合や、プラグインが正常にシャットダウンされない場合 (たとえば、サーバーホストがクラッシュした場合) に、ログファイルの完全性が危険にさらされる可能性があります。このリスクを減らすには、同期ロギングが使用されるように audit_log_strategy を設定します。戦略に関係なく、ロギングはベストエフォートベーシスで発生するため、一貫性は保証されません。

監査ログ領域の管理

監査ログプラグインには、そのログファイルで使用される領域を管理できる複数のシステム変数が用意されています。

- audit_log_buffer_size: 非同期ロギング用のバッファーサイズを設定するには、この変数をサーバーの起動時に設定します。このプラグインでは、初期化時に割り当てられ、終了時に削除される単一のバッファーが使用されます。このプラグインは、ロギングが非同期の場合にのみ、このバッファーを割り当てます。
- audit_log_rotate_on_size、audit_log_flush: これらの変数を使用すると、監査ログファイルのローテーションおよびフラッシュが許可されます。監査ログファイルが非常に大きくなり、大量のディスク領域が消費される可能性があります。使用される領域を管理するには、自動的なログのローテーションを有効にするか、手動で監査ファイルの名前を変更し、ログをフラッシュして新しいファイルを開きます。必要に応じて、名前が変更されたファイルを削除したり、バックアップしたりできます。

  デフォルトでは、audit_log_rotate_on_size=0 であり、ログのローテーションは発生しません。この場合、audit_log_flush の値が無効から有効に変更されると、監査ログプラグインはログファイルを閉じてから再度開きます。ログファイル名の変更は、サーバーの外部で実行される必要があります。名前を audit.log.1 から audit.log.3 で循環させる 3 つの最近のログファイルを保持すると仮定します。Unix 上で、次のように手動でローテーションを実行します。

  1. コマンド行から、現在のログファイル名を変更します。

```
shell> mv audit.log.2 audit.log.3
shell> mv audit.log.1 audit.log.2
```

```
shell> mv audit.log audit.log.1
```

この時点で、プラグインは引き続き、audit.log.1 に名前が変更された現在のログファイルに書き込みます。

2. サーバーに接続し、ログファイルをフラッシュします。これにより、プラグインはログファイルを閉じて、新しい audit.log ファイルログを再度開きます。

```
mysql> SET GLOBAL audit_log_flush = ON;
```

audit_log_rotate_on_size が 0 よりも大きい場合は、audit_log_flush を設定しても効果がありません。この場合、ファイルへの書き込みによってそのサイズが audit_log_rotate_on_size の値を超えるたびに、監査ログプラグインはそのログファイルを閉じてから再度開きます。このプラグインは、タイムスタンプ拡張子が追加されるように元のファイル名を変更します。たとえば、audit.log は audit.log.13440033615657730 という名前に変更される可能性があります。最後の 7 桁は小数部です。最初の 10 桁は、FROM_UNIXTIME() 関数を使用して解釈できる Unix タイムスタンプ値です。

```
mysql> SELECT FROM_UNIXTIME(1344003361);
+--------------------------+
| FROM_UNIXTIME(1344003361) |
+--------------------------+
| 2012-08-03 09:16:01      |
+--------------------------+
```

監査ログのフィルタリング

監査ログプラグインは、監査対象イベントをフィルタリングできます。これにより、イベントの発生元のアカウントやイベントのステータスに基づいて、監査ログファイルにイベントを書き込むかどうかを制御できます。ステータスのフィルタリングは、接続イベントおよびステートメントイベントごとに個別に発生します。

アカウント別のイベントフィルタリング

MySQL 5.6.20 の時点で、発生元のアカウントに基づいて監査対象イベントをフィルタリングするには、サーバーの起動時または実行時に、次のシステム変数のいずれかを設定します。

- audit_log_include_accounts: 監査ロギングに含めるアカウント。この変数が設定されている場合は、これらのアカウントのみが監査されます。
- audit_log_exclude_accounts: 監査ロギングから除外するアカウント。この変数が設定されている場合は、これらのアカウント以外がすべて監査されます。

いずれかの変数の値には、NULL またはカンマで区切った 1 つ以上のアカウント名を含む文字列を指定できます。それぞれの形式は user_name@host_name です。デフォルトでは、両方の変数が NULL になっています。この場合、アカウントのフィルタリングは実行されず、すべてのアカウントで監査が発生します。

例: user1 および user2 ローカルホストのアカウントでのみ監査ロギングを有効にするには、次のように audit_log_include_accounts システム変数を設定します。

```
SET GLOBAL audit_log_include_accounts = 'user1@localhost,user2@localhost';
```

同時に NULL 以外に設定できるのは、audit_log_include_accounts と audit_log_exclude_accounts のいずれかのみです。

- audit_log_include_accounts を設定すると、サーバーは audit_log_exclude_accounts を NULL に設定します。
- audit_log_include_accounts が NULL でない場合を除いて、audit_log_exclude_accounts を設定しようとするとエラーが発生します。この場合は、まず audit_log_include_accounts を NULL に設定することでクリアする必要があります。

```
-- This sets audit_log_exclude_accounts to NULL
SET GLOBAL audit_log_include_accounts = value;

-- This fails because audit_log_include_accounts is not NULL
SET GLOBAL audit_log_exclude_accounts = value;

-- To set audit_log_exclude_accounts, first set
-- audit_log_include_accounts to NULL
SET GLOBAL audit_log_include_accounts = NULL;
SET GLOBAL audit_log_exclude_accounts = value;
```

いずれかの変数の値を調査する場合は、SHOW VARIABLES で NULL が空の文字列として表示されることに注意してください。これを回避するには、代わりに SELECT を使用してください。

```
mysql> SHOW VARIABLES LIKE 'audit_log_include_accounts';
+---------------------------+-------+
| Variable_name             | Value |
+---------------------------+-------+
| audit_log_include_accounts |       |
+---------------------------+-------+
mysql> SELECT @@audit_log_include_accounts;
+-----------------------------+
| @@audit_log_include_accounts |
+-----------------------------+
| NULL                        |
+-----------------------------+
```

カンマ、スペース、またはその他の特殊文字が含まれているために、ユーザー名やホスト名を引用符で囲む必要がある場合は、一重引用符を使用して囲みます。変数の値自体が一重引用符で囲まれている場合は、内側の各一重引用符を二重に入力するか、バックスラッシュを使用してエスケープします。次のステートメントはそれぞれ、ローカルの root アカウントの監査ロギングを有効にします。引用符のスタイルが異なりますが、いずれも同等です。

```
SET GLOBAL audit_log_include_accounts = 'root@localhost';
SET GLOBAL audit_log_include_accounts = '"root"@"localhost"';
SET GLOBAL audit_log_include_accounts = '\'root\'@\'localhost\'';
SET GLOBAL audit_log_include_accounts = "'root'@'localhost'";
```

ANSI_QUOTES SQL モードでは、二重引用符は文字列の引用ではなく、識別子の引用を示すため、このモードが有効になっている場合は、最後のステートメントが機能しません。

ステータス別のイベントフィルタリング

MySQL 5.6.20 の時点で、ステータスに基づいて監査対象イベントをフィルタリングするには、サーバーの起動時または実行時に、次のシステム変数を設定します。

- audit_log_connection_policy: 接続イベントのロギングポリシーです
- audit_log_statement_policy: ステートメントイベントのロギングポリシーです

各変数には、ALL (関連付けられたすべてのイベントのログを記録します。これがデフォルトです)、ERRORS (失敗したイベントのログのみを記録します)、または NONE (イベントのログを記録しません) の値が指定されます。たとえば、ステートメントイベントのログはすべて記録するが、接続イベントのログは失敗したもののみを記録する場合は、次の設定を使用します。

```
SET GLOBAL audit_log_statement_policy = ALL;
SET GLOBAL audit_log_connection_policy = ERRORS;
```

MySQL 5.6.20 よりも前では、audit_log_connection_policy および audit_log_statement_policy を使用できません。代わりに、サーバーの起動時または実行時に audit_log_policy を使用します。これには、ALL (すべてのイベントのログを記録します。これがデフォルトです)、LOGINS (接続イベントのログを記録します)、QUERIES (ステートメントイベントのログを記録します)、または NONE (イベントのログを記録しません) の値が指定されます。これらの値のいずれを指定しても、監査ログプラグインは成功と失敗を区別せずに、選択したイベントのログをすべて記録します。

MySQL 5.6.20 の時点で、audit_log_policy は引き続き使用可能ですが、サーバーの起動時にしか設定できません。実行時は、読み取り専用の変数です。起動時に使用すると、次のように動作します。

- audit_log_policy を設定しない場合や、デフォルト値の ALL に設定した場合でも、audit_log_connection_policy または audit_log_statement_policy を明示的に設定すれば、指定どおりに適用されます。指定しない場合は、デフォルトが ALL に設定されます。
- audit_log_policy を ALL 以外の値に設定した場合は、次の表に示すように、その値が優先され、audit_log_connection_policy および audit_log_statement_policy を設定する際に使用されます。また、これらの変数のいずれかをデフォルトの ALL 以外の値に設定する場合、サーバーはそれらの値がオーバーライドされることを示すメッセージをエラーログに書き込みます。

起動時の audit_log_policy 値	結果として返される audit_log_connection_policy 値	結果として返される audit_log_statement_policy 値
LOGINS	ALL	NONE
QUERIES	NONE	ALL
NONE	NONE	NONE

イベントフィルタリングのレポート

次のステータス変数の値を調査すると、フィルタリングの結果を確認できます。

- Audit_log_events: フィルタリングのポリシーに基づいてログに書き込まれたかどうかに関係なく、監査ログプラグインによって処理されたイベントの数。
- Audit_log_events_filtered: フィルタリングのポリシーに基づいて、監査ログプラグインによってフィルタリングされた (ログに書き込まれなかった) イベントの数。
- Audit_log_events_written: 監査ログに書き込まれたイベントの数。

MySQL 5.6.20 の時点では、これらの変数を使用できます。

6.3.12.5 監査ログプラグインのオプションおよび変数のリファレンス

表 6.17 監査ログプラグインのオプション/変数のリファレンス

名前	コマンド行	オプションファイル	システム変数	ステータス変数	変数スコープ	動的
audit-log	はい	はい				
audit_log_buffer_size	はい	はい	はい		グローバル	いいえ
audit_log_connection_policy	はい	はい	はい		グローバル	はい
audit_log_current_session			はい		両方	いいえ
Audit_log_current_size				はい	グローバル	いいえ
Audit_log_event_max_drop_size				はい	グローバル	いいえ
Audit_log_events				はい	グローバル	いいえ
Audit_log_events_filtered				はい	グローバル	いいえ
Audit_log_events_lost				はい	グローバル	いいえ
Audit_log_events_written				はい	グローバル	いいえ
audit_log_exclude_accounts	はい	はい	はい		グローバル	はい
audit_log_file	はい	はい	はい		グローバル	いいえ
audit_log_flush			はい		グローバル	はい
audit_log_format	はい	はい	はい		グローバル	いいえ
audit_log_include_accounts	はい	はい	はい		グローバル	はい
audit_log_policy	はい	はい	はい		グローバル	異なる
audit_log_rotate_on_size	はい	はい	はい		グローバル	はい
audit_log_statement_policy	はい	はい	はい		グローバル	はい
audit_log_strategy	はい	はい	はい		グローバル	いいえ
Audit_log_total_size				はい	グローバル	いいえ
Audit_log_write_waits				はい	グローバル	いいえ

6.3.12.6 監査ログプラグインのオプションおよびシステム変数

このセクションでは、監査ログプラグインの操作を制御するコマンドオプションおよびシステム変数について説明します。起動時に指定された値が正しくない場合は、プラグインが正常に初期化できない可能性があり、サーバーでロードされません。この場合、サーバーでその他の監査ログ設定が認識されないため、それに関するエラーメッセージが生成される可能性もあります。

audit_log プラグインのアクティブ化を制御するには、次のオプションを使用します。

- --audit-log[=value]

コマンド行形式	--audit-log[=value]
導入	5.6.10
型	列挙

デフォルト	ON
有効な値	ON OFF FORCE FORCE_PLUS_PERMANENT

このオプションは、サーバーの起動時に audit_log プラグインをロードする方法を制御します。これは、監査ログプラグインが事前に INSTALL PLUGIN に登録されている場合や、--plugin-load を使用してロードされている場合にのみ指定可能です。セクション6.3.12.1「監査ログプラグインのインストール」を参照してください。

セクション5.1.8.1「プラグインのインストールおよびアンインストール」で説明したように、オプションの値は、プラグインのロードオプションに指定可能な値のいずれかである必要があります。たとえば、--audit-log=FORCE_PLUS_PERMANENT は、プラグインをロードし、それがサーバーの実行時に削除されることを回避するようにサーバーに指示します。

このオプションは MySQL 5.6.10 で追加されました。

audit_log プラグインがインストールされている場合は、ロギングに対する制御を許可する複数のシステム変数が表示されます。これらの変数は、audit_log プラグインが有効になっている場合にのみ指定可能です。

```
mysql> SHOW VARIABLES LIKE 'audit_log%';
+-------------------------------+-------------+
| Variable_name                 | Value       |
+-------------------------------+-------------+
| audit_log_buffer_size         | 1048576     |
| audit_log_connection_policy | ALL         |
| audit_log_current_session   | ON          |
| audit_log_exclude_accounts  |             |
| audit_log_file              | audit.log   |
| audit_log_flush             | OFF         |
| audit_log_format            | OLD         |
| audit_log_include_accounts  |             |
| audit_log_policy            | ALL         |
| audit_log_rotate_on_size    | 0           |
| audit_log_statement_policy  | ALL         |
| audit_log_strategy          | ASYNCHRONOUS |
+-------------------------------+-------------+
```

これらの変数のいずれも、サーバーの起動時 (一部は実行時) に設定できます。

- audit_log_buffer_size

コマンド行形式	--audit-log-buffer-size=value
導入	5.6.10
システム変数	audit_log_buffer_size
スコープ	グローバル
動的	いいえ
型	数値
デフォルト	1048576
最小値	4096
最大値 (64 ビットプラットフォーム)	18446744073709547520
最大値 (32 ビットプラットフォーム)	4294967295

監査ログプラグインが非同期的にイベントをログに書き込むと、イベントの内容を書き込む前に、バッファーを使用してそれらを格納します。この変数は、そのバッファーのサイズ (バイト単位) を制御します。サーバーは、この値を 4096 の倍数に調整します。このプラグインでは、初期化時に割り当てられ、終了時に削除される単一のバッファーが使用されます。このプラグインは、ロギングが非同期の場合にのみ、このバッファーを割り当てます。

この変数は MySQL 5.6.10 で追加されました。

- audit_log_connection_policy

コマンド行形式	--audit-log-connection-policy=value
導入	5.6.20
システム変数	audit_log_connection_policy
スコープ	グローバル
動的	はい
型	列挙
デフォルト	ALL
有効な値	ALL ERRORS NONE

監査ログプラグインが接続イベントをそのログファイルに書き込む方法を制御するポリシーです。次の表は、許可される値を示しています。

値	説明
ALL	接続イベントのログをすべて記録します
ERRORS	失敗した接続イベントのログのみを記録します
NONE	接続イベントのログを記録しません

> 注記
>
> セクション6.3.12.4「監査ログプラグインのロギング制御」で説明したように、audit_log_policy も指定されている場合は、サーバーの起動時に、audit_log_connection_policy に明示的に指定された値がオーバーライドされる可能性があります。

この変数は、MySQL 5.6.20 で追加されました。

- audit_log_current_session

導入	5.6.20
システム変数	audit_log_current_session
スコープ	グローバル、セッション
動的	いいえ
型	ブール
デフォルト	depends on filtering policy

現在のセッションで監査ロギングが有効になっているかどうかを示します。この変数のセッションの値は、読み取り専用です。セッションの開始時に、audit_log_include_accounts および audit_log_exclude_accounts システム変数の値に基づいて設定されます。監査ログプラグインはこのセッション値を使用して、そのセッションでイベントを監査するかどうかを決定します。(グローバル値もありますが、このプラグインでは使用されません。)

この変数は、MySQL 5.6.20 で追加されました。

- audit_log_exclude_accounts

コマンド行形式	--audit-log-exclude-accounts=value
導入	5.6.20
システム変数	audit_log_exclude_accounts
スコープ	グローバル
動的	はい
型	文字列

デフォルト	NULL

イベントのログが記録されないアカウント。この値には、NULL またはカンマで区切った 1 つ以上のアカウント名のリストを含む文字列を指定するようにしてください。詳細については、セクション6.3.12.4「監査ログプラグインのロギング制御」を参照してください。

この変数は、MySQL 5.6.20 で追加されました。

- audit_log_file

コマンド行形式	--audit-log-file=file_name
導入	5.6.10
システム変数	audit_log_file
スコープ	グローバル
動的	いいえ
型	ファイル名
デフォルト	audit.log

監査ログプラグインがイベントを書き込むファイルの名前。デフォルト値は audit.log です。ファイル名が相対パスの場合、サーバーはデータディレクトリに相対的なパスであると解釈します。セキュリティー上の理由から、監査ログファイルは、MySQL サーバーおよびログを表示する正当な理由を持つユーザーにのみアクセス可能なディレクトリに書き込まれるべきです。

この変数は MySQL 5.6.10 で追加されました。

- audit_log_flush

導入	5.6.10
システム変数	audit_log_flush
スコープ	グローバル
動的	はい
型	ブール
デフォルト	OFF

この変数が有効になるよう (1 または ON) に設定されている場合、監査ログプラグインはそのログファイルを閉じてから再度開いて、フラッシュします。(この値は、別のフラッシュを実行するために再度有効にする前に、明示的に無効にする必要がないように、OFF のままになっています。)audit_log_rotate_on_size が 0 である場合を除いて、この変数を有効にしても効果はありません。

この変数は MySQL 5.6.10 で追加されました。

- audit_log_format

コマンド行形式	--audit-log-format=value
導入	5.6.14
システム変数	audit_log_format
スコープ	グローバル
動的	いいえ
型	列挙
デフォルト	OLD
有効な値	OLD

	NEW

監査ログファイルの形式。許可されている値は、OLD および NEW (デフォルトは OLD) です。各形式についての詳細は、セクション6.3.12.3「監査ログファイル」を参照してください。

audit_log_format の値を変更する場合は、次の手順を使用して、ある形式のログエントリが別の形式のエントリを含む既存のログファイルに書き込まれることを回避します。

1. サーバーを停止します。
2. 現在の監査ログファイルの名前を手動で変更します。
3. 新しい audit_log_format の値でサーバーを再起動します。監査ログプラグインによって、選択した形式のログエントリを含む新しいログファイルが作成されます。

この変数は、MySQL 5.6.14 で追加されました。

- audit_log_include_accounts

コマンド行形式	--audit-log-include-accounts=value
導入	5.6.20
システム変数	audit_log_include_accounts
スコープ	グローバル
動的	はい
型	文字列
デフォルト	NULL

イベントのログが記録されるアカウント。この値には、NULL またはカンマで区切った 1 つ以上のアカウント名のリストを含む文字列を指定するようにしてください。詳細については、セクション6.3.12.4「監査ログプラグインのロギング制御」を参照してください。

この変数は、MySQL 5.6.20 で追加されました。

- audit_log_policy

コマンド行形式	--audit-log-policy=value
導入	5.6.10
システム変数	audit_log_policy
スコープ	グローバル
動的 (≥ 5.6.20)	いいえ
動的 (≤ 5.6.19)	はい
型	列挙
デフォルト	ALL
有効な値	ALL LOGINS QUERIES NONE

監査ログプラグインがイベントをそのログファイルに書き込む方法を制御するポリシーです。次の表は、許可される値を示しています。

値	説明
ALL	イベントのログをすべて記録します
LOGINS	ログインイベントのログのみを記録します
QUERIES	クエリーイベントのログのみを記録します

値	説明
NONE	ログを何も記録しません (監査ストリームを無効にします)

MySQL 5.6.20 の時点で、audit_log_policy はサーバーの起動時にしか設定できません。実行時は、読み取り専用の変数です。これは、より詳細にロギングポリシーを制御し、起動時または実行時に設定できるその他の 2 つのシステム変数 (audit_log_connection_policy と audit_log_statement_policy) が導入されたためです。その他の 2 つの変数の代わりに、起動時に audit_log_policy を使用し続けると、サーバーはその値を使用して、これらの変数を設定します。ポリシーの変数とそれらの相互作用についての詳細は、セクション6.3.12.4「監査ログプラグインのロギング制御」を参照してください。

MySQL 5.6.20 よりも前では、audit_log_connection_policy および audit_log_statement_policy システム変数が存在しません。audit_log_policy は、唯一のポリシー制御の変数であり、サーバーの起動時または実行時に設定できます。

この変数は MySQL 5.6.10 で追加されました。

- audit_log_rotate_on_size

コマンド行形式	--audit-log-rotate-on-size=N
導入	5.6.10
システム変数	audit_log_rotate_on_size
スコープ	グローバル
動的	はい
型	数値
デフォルト	0

audit_log_rotate_on_size 値を 0 よりも大きくした場合は、ファイルへの書き込みによってそのサイズがこの値を超えると、監査ログプラグインはそのログファイルを閉じてから再度開きます。元のファイル名は、タイムスタンプ拡張子が含まれるように変更されます。

audit_log_rotate_on_size 値が 0 の場合、プラグインはサイズに基づいて、そのログを閉じてから再度開きません。代わりに、audit_log_flush 使用して、要求に応じてログを閉じてから再度開きます。この場合、ファイルをフラッシュする前に、サーバーの外部でファイル名を変更します。

監査ログファイルのローテーションおよびタイムスタンプの解釈についての詳細は、セクション6.3.12.4「監査ログプラグインのロギング制御」を参照してください。

この変数を 4096 の倍数でない値に設定すると、もっとも近い倍数に切り捨てられます。(したがって、4096 未満の値に設定すると、0 (ゼロ) に設定した結果となり、ローテーションは発生しません。)

この変数は MySQL 5.6.10 で追加されました。

- audit_log_statement_policy

コマンド行形式	--audit-log-statement-policy=value
導入	5.6.20
システム変数	audit_log_statement_policy
スコープ	グローバル
動的	はい
型	列挙
デフォルト	ALL
有効な値	ALL ERRORS

	NONE

監査ログプラグインがステートメントイベントをそのログファイルに書き込む方法を制御するポリシーです。次の表は、許可される値を示しています。

値	説明
ALL	ステートメントイベントのログをすべて記録します
ERRORS	失敗したステートメントイベントのログのみを記録します
NONE	ステートメントイベントのログを記録しません

> **注記**
>
> セクション6.3.12.4「監査ログプラグインのロギング制御」で説明したように、audit_log_policy も指定されている場合は、サーバーの起動時に、audit_log_statement_policy に明示的に指定された値がオーバーライドされる可能性があります。

この変数は、MySQL 5.6.20 で追加されました。

- audit_log_strategy

コマンド行形式	--audit-log-strategy=value
導入	5.6.10
システム変数	audit_log_strategy
スコープ	グローバル
動的	いいえ
型	列挙
デフォルト	ASYNCHRONOUS
有効な値	ASYNCHRONOUS PERFORMANCE SEMISYNCHRONOUS SYNCHRONOUS

監査ログプラグインで使用されるロギング方法。次の表では、許可されている値について説明します。

表 6.18 監査ログの戦略

値	意味
ASYNCHRONOUS	非同期的にログを記録し、出力バッファー内の領域を待機します
PERFORMANCE	非同期的にログを記録し、出力バッファー内の領域が十分でない場合はリクエストを破棄します
SEMISYNCHRONOUS	同期的にログを記録し、オペレーティングシステムによるキャッシュ処理を許可します
SYNCHRONOUS	同期的にログを記録し、各リクエスト後に sync() を呼び出します

この変数は MySQL 5.6.10 で追加されました。

6.3.12.7 監査ログプラグインのステータス変数

監査ログプラグインでは、次のステータス変数がサポートされています。これらは、audit_log プラグインが有効になっている場合にのみ指定可能です。

- Audit_log_current_size

現在の監査ログファイルのサイズ。この値は、イベントがログに書き込まれると増加し、ログがローテーションされると 0 にリセットされます。

この変数は、MySQL 5.6.20 で追加されました。

- Audit_log_event_max_drop_size

  パフォーマンスロギングモードで破棄された最大イベントのサイズ。ロギングモードについては、セクション 6.3.12.4「監査ログプラグインのロギング制御」を参照してください。

  この変数は、MySQL 5.6.20 で追加されました。

- Audit_log_events

  フィルタリングのポリシーに基づいてログに書き込まれたかどうかに関係なく、監査ログプラグインによって処理されたイベントの数 (セクション6.3.12.4「監査ログプラグインのロギング制御」を参照してください)。

  この変数は、MySQL 5.6.20 で追加されました。

- Audit_log_events_filtered

  フィルタリングのポリシーに基づいて、監査ログプラグインによってフィルタリングされた (ログに書き込まれなかった) イベントの数 (セクション6.3.12.4「監査ログプラグインのロギング制御」を参照してください)。

  この変数は、MySQL 5.6.20 で追加されました。

- Audit_log_events_lost

  イベントが使用可能な監査ログバッファー領域よりも大きかったために、パフォーマンスロギングモードで失われたイベントの数。この値は、audit_log_buffer_size を設定して、パフォーマンスモード用にバッファーのサイズを調整する方法を評価する際に役立つことがあります。ロギングモードについては、セクション 6.3.12.4「監査ログプラグインのロギング制御」を参照してください。

  この変数は、MySQL 5.6.20 で追加されました。

- Audit_log_events_written

  監査ログに書き込まれたイベントの数。

  この変数は、MySQL 5.6.20 で追加されました。

- Audit_log_total_size

  すべての監査ログファイルに書き込まれたイベントの合計サイズ。Audit_log_current_size とは異なり、Audit_log_total_size の値は、ログがローテーションされたときにも増加します。

  この変数は、MySQL 5.6.20 で追加されました。

- Audit_log_write_waits

  非同期ロギングモードでイベントが監査ログバッファー内の領域を待機する必要があった回数。ロギングモードについては、セクション6.3.12.4「監査ログプラグインのロギング制御」を参照してください。

  この変数は、MySQL 5.6.20 で追加されました。

6.3.12.8 監査ログプラグインの制限事項

監査ログプラグインは、次の制限事項の対象になっています。

- 最上位のステートメントのみのログが記録され、トリガーやストアドプロシージャーなどのストアドプログラム内のステートメントのログは記録されません。
- LOAD DATA INFILE などのステートメントから参照されるファイルの内容は、記録されません。

6.3.13 SQL ベースの MySQL アカウントアクティビティーの監査

アプリケーションは次のガイドラインを使用することで、データベースアクティビティーを MySQL アカウントに関連付ける SQL ベースの監査を実行できます。

MySQL アカウントは、mysql.user テーブルの行に対応します。クライアントが正常に接続すると、サーバーはこのテーブル内の特定の行にアクセスするクライアントを認証します。この行の User および Host カラムの値は、

アカウントを一意に識別し、アカウント名が SQL ステートメントに書き込まれる 'user_name'@'host_name' 形式に対応します。

クライアントを認証する際に使用されるアカウントによって、クライアントが持っている権限が特定されます。通常、CURRENT_USER() 関数を呼び出すと、このアカウントがどのクライアントユーザー用であるかを特定できます。その値は、アカウントの user テーブル行の User および Host カラムで構成されています。

ただし、CURRENT_USER() 値がクライアントユーザーではなく、別のアカウントに対応するという状況もあります。これは、権限チェックがクライアントのアカウントに基づいて実行されないコンテキストで発生します。

- SQL SECURITY DEFINER 特性を使用して定義されたストアドルーチン (プロシージャーおよび関数)
- SQL SECURITY DEFINER 特性を使用して定義されたビュー
- トリガーとイベント

このようなコンテキストでは、権限チェックは DEFINER アカウントと照合して実行され、CURRENT_USER() はそのアカウントを参照し、ストアドルーチンまたはビューを呼び出したクライアント、またはトリガーをアクティブにしたクライアントのアカウントは参照しません。クライアントおよびクライアントの接続元ホストによって指定された実際のユーザー名を示す値を返す USER() 関数を呼び出すと、呼び出し元のユーザーを特定できます。ただし、USER() の値にはワイルドカードが含まれない一方で、(CURRENT_USER() によって返される) アカウントの値にはユーザー名およびホスト名のワイルドカードが含まれる可能性があるため、この値は必ずしも、user テーブル内のアカウントに直接対応するとはかぎりません。

たとえば、空白のユーザー名は任意のユーザーに一致するため、''@'localhost' のアカウントを使用すると、クライアントは任意のユーザー名を持つローカルホストから匿名ユーザーとして接続できます。この場合、クライアントがローカルホストから user1 として接続している場合、USER() と CURRENT_USER() は別々の値を返します。

```
mysql> SELECT USER(), CURRENT_USER();
+----------------+----------------+
| USER()         | CURRENT_USER() |
+----------------+----------------+
| user1@localhost | @localhost     |
+----------------+----------------+
```

アカウントのホスト名部分にも、ワイルドカードを含めることができます。ホスト名に '%' または '_' パターン文字が含まれている場合や、ネットマスク表記が使用されている場合は、複数のホストから接続しているクライアントにそのアカウントを使用できますが、CURRENT_USER() の値には、どのホストであるのかが示されません。たとえば、アカウント 'user2'@'%.example.com' を使用すると、user2 が example.com ドメイン内の任意のホストから接続できます。user2 が remote.example.com から接続すると、USER() と CURRENT_USER() は別々の値を返します。

```
mysql> SELECT USER(), CURRENT_USER();
+------------------------+--------------------+
| USER()                 | CURRENT_USER()     |
+------------------------+--------------------+
| user2@remote.example.com | user2@%.example.com |
+------------------------+--------------------+
```

アプリケーションがユーザーを監査するために USER() を呼び出す必要があるが (たとえば、トリガー内から監査を実行する場合)、USER() の値を user テーブル内のアカウントに関連付けることができる必要もある場合は、アカウントの User または Host カラムにワイルドカードが含まれることを回避する必要があります。特に、User を (匿名のユーザーアカウントが作成される) 空にすることは許可しないでください。また、Host の値に、パターン文字またはネットマスク表記を使用することも許可しないでください。すべてのアカウントには、空でない User 値とリテラルの Host 値を含める必要があります。

前述の例に関しては、ワイルドカードが使用されないように ''@'localhost' および 'user2'@'%.example.com' アカウントを変更するようにしてください。

```
RENAME USER ''@'localhost' TO 'user1'@'localhost';
RENAME USER 'user2'@'%.example.com' TO 'user2'@'remote.example.com';
```

user2 が example.com ドメイン内の複数のホストから接続できる必要がある場合は、ホストごとに個別のアカウントにするべきです。

CURRENT_USER() または USER() の値からユーザー名またはホスト名の部分を抽出するには、SUBSTRING_INDEX() 関数を使用します。

```
mysql> SELECT SUBSTRING_INDEX(CURRENT_USER(),'@',1);
```

```
+----------------------------------------+
| SUBSTRING_INDEX(CURRENT_USER(),'@',1) |
+----------------------------------------+
| user1                                  |
+----------------------------------------+

mysql> SELECT SUBSTRING_INDEX(CURRENT_USER(),'@',-1);
+----------------------------------------+
| SUBSTRING_INDEX(CURRENT_USER(),'@',-1) |
+----------------------------------------+
| localhost                              |
+----------------------------------------+
```

第 7 章 バックアップとリカバリ

目次

7.1 バックアップとリカバリの種類 752
7.2 データベースバックアップ方法 754
7.3 バックアップおよびリカバリ戦略の例 756
7.3.1 バックアップポリシーの確立 757
7.3.2 リカバリへのバックアップの使用 759
7.3.3 バックアップ戦略サマリー 759
7.4 バックアップへの mysqldump の使用 759
7.4.1 mysqldump による SQL フォーマットでのデータのダンプ 760
7.4.2 SQL フォーマットバックアップのリロード 760
7.4.3 mysqldump による区切りテキストフォーマットでのデータのダンプ 761
7.4.4 区切りテキストフォーマットバックアップのリロード 762
7.4.5 mysqldump のヒント 762
7.5 バイナリログを使用したポイントインタイム (増分) リカバリ 764
7.5.1 イベント時間を使用したポイントインタイムリカバリ 765
7.5.2 イベントの位置を使用したポイントインタイムリカバリ 766
7.6 MyISAM テーブルの保守とクラッシュリカバリ 766
7.6.1 クラッシュリカバリへの myisamchk の使用 767
7.6.2 MyISAM テーブルのエラーのチェック方法 767
7.6.3 MyISAM テーブルの修復方法 768
7.6.4 MyISAM テーブルの最適化 770
7.6.5 MyISAM テーブル保守スケジュールのセットアップ 770

システムクラッシュ、ハードウェアの障害、またはユーザーが誤ってデータを削除するなどの問題が発生した場合に、データをリカバリし、再度起動して、実行できるように、データベースをバックアップすることが重要です。バックアップは、MySQL インストールのアップグレード前の保護手段としても不可欠であり、それらを使用して、MySQL インストールを別のシステムに転送したり、レプリケーションスレーブサーバーをセットアップしたりできます。

MySQL では、多様なバックアップ戦略を提供しており、それらからインストールの要件にもっとも適合する方法を選択できます。この章では、熟知しておくべきであるいくつかのバックアップとリカバリのトピックについて説明します。

- バックアップの種類: 論理と物理、完全と増分など。
- バックアップの作成の方法。
- ポイントインタイムリカバリを含むリカバリ方法。
- バックアップのスケジューリング、圧縮、および暗号化。
- 破損したテーブルのリカバリを可能にするためのテーブルの保守。

追加のリソース

バックアップまたはデータの可用性の維持に関連するリソースには次のものが含まれます。

- MySQL Enterprise Edition のお客様はバックアップに MySQL Enterprise Backup 製品を使用できます。MySQL Enterprise Backup 製品の概要については、セクション25.2「MySQL Enterprise Backup」を参照してください。
- バックアップの問題に特化したフォーラムは https://forums.mysql.com/list.php?28 にあります。
- mysqldump、mysqlhotcopy、およびほかの MySQL バックアッププログラムの詳細は第4章「MySQL プログラム」で見つかります。
- ここで説明している SQL ステートメントの構文は、第13章「SQL ステートメントの構文」にあります。
- InnoDB バックアップ手順の追加情報については、セクション14.16「InnoDB のバックアップとリカバリ」を参照してください。

- レプリケーションにより、複数のサーバーで同一のデータを保持できます。これには、クライアントクエリーの負荷をサーバー全体に分散できること、特定のサーバーがオフラインにされるか障害が発生した場合でもデータを使用できること、およびスレーブサーバーを使用して、マスターに影響を与えずにバックアップを作成する機能など、いくつかのメリットがあります。第17章「レプリケーション」を参照してください。
- MySQL Cluster は、分散コンピューティング環境に適した高可用性、高冗長性バージョンの MySQL を提供します。第18章「MySQL Cluster NDB 7.3 および MySQL Cluster NDB 7.4」を参照してください。これは MySQL Cluster NDB 7.3 (MySQL 5.6 に基づいていますが、NDBCLUSTER ストレージエンジンの最新の改善と修正が含まれます) に関する情報を提供しています。

  > **注記**
  >
  > NDBCLUSTER ストレージエンジンは現在 MySQL 5.6 でサポートされていません。

- Distributed Replicated Block Device (DRBD) は別の高可用性ソリューションです。これは、プライマリサーバーからセカンダリサーバーにブロックレベルでブロック型デバイスをレプリケートすることによって機能します。第16章「高可用性と拡張性」を参照してください

7.1 バックアップとリカバリの種類

このセクションでは、さまざまな種類のバックアップの特性について説明します。

物理 (raw) バックアップと論理バックアップ

物理バックアップは、データベースの内容を格納するディレクトリとファイルの raw コピーから構成されます。この種類のバックアップは、問題の発生時に早急にリカバリさせる必要がある大規模で重要なデータベースに適しています。

論理バックアップは、論理データベース構造として表される情報 (CREATE DATABASE、CREATE TABLE ステートメント) と内容 (INSERT ステートメントまたは区切りテキストファイル) を保存します。この種類のバックアップは、ユーザーがデータ値やテーブル構造を編集したり、別のマシンアーキテクチャーにデータを再作成したりできる少量のデータに適しています。

物理バックアップ方法にはこれらの特性があります。

- バックアップはデータベースディレクトリおよびファイルの正確なコピーから構成されます。一般的に、これは MySQL データディレクトリのすべてまたは一部のコピーです。
- 物理バックアップ方法は、変換しないファイルコピーのみが含まれるため、論理より高速です。
- 出力は論理バックアップの場合よりコンパクトです。
- ビジーで、重要なデータベースには、バックアップの速度やコンパクトさが重要であるため、MySQL Enterprise Backup 製品は物理バックアップを実行します。MySQL Enterprise Backup 製品の概要については、セクション25.2「MySQL Enterprise Backup」を参照してください。
- バックアップとリストアの粒度は、データディレクトリ全体のレベルから個々のファイルのレベルまでの範囲になります。これは、ストレージエンジンに応じて、テーブルレベルの粒度を提供する場合としない場合があります。たとえば、InnoDB テーブルは、それぞれ個別のファイルにしたり、ほかの InnoDB テーブルとファイルストレージを共有したりできます。各 MyISAM テーブルはファイルのセットに一意に対応します。
- データベースに加えて、バックアップにはログファイルや構成ファイルなどの関連ファイルを含めることができます。
- MEMORY テーブルの内容はディスクに格納されないため、それらのデータをこの方法でバックアップすることは困難です。(MySQL Enterprise Backup 製品には、バックアップ中に MEMORY テーブルからデータを取得できる機能があります。)
- バックアップは、同一か類似のハードウェア特性を持つほかのマシンにのみ移植可能です。
- バックアップは MySQL サーバーが実行していない間に実行できます。サーバーが実行中の場合は、バックアップ中にサーバーがデータベースの内容を変更しないように、適切なロックを実行する必要があります。MySQL Enterprise Backup は、このロックが必要なテーブルに対して、自動的にロックを実行します。
- 物理バックアップツールには、InnoDB またはその他のテーブル用の MySQL Enterprise Backup の mysqlbackup、ファイルシステムレベルのコマンド (cp、scp、tar、rsync など)、または MyISAM テーブル用の mysqlhotcopy が含まれます。
- リストアの場合:

- MySQL Enterprise Backup はバックアップした InnoDB およびその他のテーブルをリストアします。
- ndb_restore は NDB テーブルをリストアします。
- ファイルシステムレベルまたは mysqlhotcopy によってコピーされたファイルは、ファイルシステムコマンドによってそれらの元の場所にコピーできます。

論理バックアップ方法にはこれらの特性があります。

- バックアップは、MySQL サーバーをクエリーし、データベース構造と内容情報を取得して実行されます。
- サーバーがデータベース情報にアクセスし、それを論理フォーマットに変換する必要があるため、バックアップは物理方法より遅くなります。クライアント側で出力が書き込まれた場合、サーバーはそれをバックアッププログラムに送信する必要もあります。
- 出力は特にテキストフォーマットで保存された場合に物理バックアップより大きくなります。
- バックアップとリストアの粒度は、サーバーレベル (すべてのデータベース)、データベースレベル (特定のデータベースのすべてのテーブル)、またはテーブルレベルで利用できます。これはストレージエンジンに関係なく当てはまります。
- バックアップには、ログファイルや構成ファイル、またはデータベースの一部ではないその他のデータベース関連ファイルは含まれません。
- 論理フォーマットで格納されているバックアップはマシンに依存せず、高度に移植可能です。
- 論理バックアップは MySQL サーバーの実行中に実行されます。サーバーはオフラインにされません。
- 論理バックアップツールには、mysqldump プログラムと SELECT ... INTO OUTFILE ステートメントが含まれます。これらは MEMORY でも、すべてのストレージエンジンで機能します。
- 論理バックアップをリストアするには、mysql クライアントを使用して、SQL フォーマットダンプファイルを処理できます。区切りテキストファイルをロードするには、LOAD DATA INFILE ステートメントまたは mysqlimport クライアントを使用します。

オンラインバックアップとオフラインバックアップ

オンラインバックアップは、データベース情報をサーバーから取得できるように、MySQL サーバーが実行中に行われます。オフラインバックアップは、サーバーが停止中に行われます。この区別は、「ホット」バックアップと「コールド」バックアップとして表すこともできます。「ウォーム」バックアップは、サーバーが実行したままですが、外部からデータベースファイルにアクセスしている間のデータの変更に対してロックされます。

オンラインバックアップ方法にはこれらの特性があります。

- このバックアップはほかのクライアントの邪魔になりにくく、クライアントはバックアップ中に MySQL サーバーに接続でき、実行する必要がある操作に応じて、データにアクセスできます。
- バックアップの完全性を損なう可能性のあるデータの変更が行われないように、適切なロックを適用する場合は、注意を払う必要があります。MySQL Enterprise Backup 製品はそのようなロックを自動的に実行します。

オフラインバックアップ方法にはこれらの特性があります。

- バックアップ中にサーバーを使用できないため、クライアントは影響を受ける可能性があります。そのため、そのようなバックアップは、多くの場合、可用性を損なうことなくオフラインにできるレプリケーションスレーブサーバーから行われます。
- バックアップ手順は、クライアントのアクティビティーからの干渉の可能性がないため単純になります。

オンラインとオフラインの同様の違いは、リカバリ操作にも当てはまり、同様の特性が当てはまります。ただし、リカバリにはより強力なロックが必要であるため、オンラインリカバリではオンラインバックアップより、クライアントが影響を受ける可能性が高くなります。バックアップ時、クライアントはバックアップ中にデータを読み取ることができます。リカバリはデータを変更し、読み取るだけでないため、データのリストア中は、クライアントのデータへのアクセスを妨げる必要があります。

ローカルバックアップとリモートバックアップ

ローカルバックアップは MySQL サーバーが実行している同じホストで実行され、リモートバックアップは別のホストから実行されます。特定の種類のバックアップでは、出力がサーバーホストにローカルで書きこまれる場合でも、バックアップをリモートホストから開始できます。

- mysqldump はローカルまたはリモートサーバーに接続できます。SQL 出力 (CREATE および INSERT ステートメント) の場合、ローカルまたはリモートダンプを実行でき、クライアント上に出力が生成されます。区切りテキスト出力 (--tab オプションを使用して) の場合、サーバーホスト上にデータファイルが作成されます。
- mysqlhotcopy はローカルバックアップのみを実行します。それはサーバーに接続し、データ変更に対してそれをロックしてから、ローカルテーブルファイルをコピーします。
- SELECT ... INTO OUTFILE はローカルまたはリモートクライアントホストから起動できますが、出力ファイルはサーバーホスト上に作成されます。
- 物理バックアップ方法は一般に、サーバーをオフラインにできるように、MySQL サーバーホスト上でローカルに開始されますが、コピーされるファイルの宛先はリモートにすることができます。

スナップショットバックアップ

一部のファイルシステム実装では、「スナップショット」を取得できます。これらは、ファイルシステム全体の物理コピーを必要とせずに、特定の時点のファイルシステムの論理コピーを提供します。(たとえば、実装では、スナップショット取得時間後に変更されたファイルシステムの部分のみがコピーされるように、コピーオンライト (copy-on-write) 技法を使用することがあります。)MySQL 自体はファイルシステムスナップショットを取得するための機能を提供していません。これは Veritas、LVM、または ZFS などのサードパーティーソリューションから使用できます。

完全バックアップと増分バックアップ

完全バックアップには、特定の時点の MySQL サーバーによって管理されるすべてのデータが含まれます。増分バックアップは、特定の期間 (ある時点から別の時点まで) 中にデータに行われた変更から構成されます。MySQL では、このセクションで先述したものなど、完全バックアップを実行するためのさまざまな方法があります。増分バックアップは、サーバーのバイナリログを有効にすることによって可能になります。サーバーはそれをデータの変更を記録するために使用します。

完全リカバリとポイントインタイム (増分) リカバリ

完全リカバリでは、完全バックアップからすべてのデータをリストアします。これは、サーバーインスタンスをバックアップが行われたときのその状態にリストアします。その状態が十分に最新でない場合、完全リカバリのあとに、完全バックアップ以降に行われた増分バックアップのリカバリを行なって、サーバーをより新しい状態にすることができます。

増分リカバリは、特定の期間中に行われた変更のリカバリです。これは、サーバーの状態を特定の時点の最新にするため、ポイントインタイムリカバリとも呼ばれます。ポイントインタイムリカバリは、バイナリログに基づき、一般にバックアップが行われたときの状態にサーバーをリストアするバックアップファイルからの完全リカバリのあとに行われます。バイナリログファイルに書き込まれたデータの変更が増分リカバリとして適用され、データの変更が元に戻され、サーバーが目的の時点の状態になります。

テーブルの保守

テーブルが破損した場合、データの完全性が損なわれる可能性があります。InnoDB テーブルの場合、これはよくある問題ではありません。MyISAM テーブルをチェックし、問題が見つかった場合にそれらを修復するプログラムについては、セクション7.6「MyISAM テーブルの保守とクラッシュリカバリ」を参照してください。

バックアップのスケジューリング、圧縮、および暗号化

バックアップスケジューリングはバックアップ手順の自動化に役立ちます。バックアップ出力の圧縮によって、領域要件が縮小し、出力の暗号化により、バックアップされたデータの権限のないアクセスに対するセキュリティーが強化されます。MySQL 自体はこれらの機能を提供していません。MySQL Enterprise Backup 製品によって InnoDB バックアップを圧縮し、バックアップ出力の圧縮や暗号化は、ファイルシステムユーティリティーを使用して実現できます。その他のサードパーティーソリューションも利用できます。

7.2 データベースバックアップ方法

このセクションでは、バックアップを作成する場合の一般的な方法をまとめています。

MySQL Enterprise Backup によるホットバックアップの作成

MySQL Enterprise Edition のお客様は MySQL Enterprise Backup 製品を使用して、インスタンス全体または選択されたデータベース、テーブル、またはその両方の物理バックアップを実行できます。この製品には、増分およ

び圧縮バックアップの機能が含まれます。物理データベースファイルのバックアップは、リストアが mysqldump コマンドなどの論理技法よりはるかに高速になります。InnoDB テーブルはホットバックアップメカニズムを使用してコピーされます。(理想的には InnoDB テーブルでデータの大部分を表しているべきです。)ほかのストレージエンジンのテーブルは、ウォームバックアップメカニズムを使用してコピーされます。MySQL Enterprise Backup 製品の概要については、セクション25.2「MySQL Enterprise Backup」を参照してください。

mysqldump または mysqlhotcopy によるバックアップの作成

mysqldump プログラムと mysqlhotcopy スクリプトでバックアップを作成できます。mysqldump はあらゆる種類のテーブルをバックアップできるため、より汎用的です。mysqlhotcopy は一部のストレージエンジンでのみ機能します。(セクション7.4「バックアップへの mysqldump の使用」およびセクション4.6.10「mysqlhotcopy — データベースバックアッププログラム」を参照してください。)

InnoDB テーブルの場合、mysqldump に --single-transaction オプションを使用して、テーブルをロックしないオンラインバックアップを実行できます。セクション7.3.1「バックアップポリシーの確立」を参照してください。

テーブルファイルのコピーによるバックアップの作成

独自のファイルを使用して、各テーブルを表すストレージエンジンの場合、テーブルはそれらのファイルをコピーしてバックアップできます。たとえば、MyISAM テーブルはファイルとして格納されるため、ファイル (*.frm、*.MYD、および *.MYI ファイル) をコピーして、簡単にバックアップを行うことができます。一貫したバックアップを取得するには、サーバーを停止するか、関連するテーブルをロックしてフラッシュします。

```
FLUSH TABLES tbl_list WITH READ LOCK;
```

読み取りロックのみが必要です。これにより、データベースディレクトリ内のファイルのコピー中に、ほかのクライアントが引き続きテーブルをクエリーすることができます。バックアップを開始する前に、すべてのアクティブインデックスページがディスクに書き込まれるようにするため、フラッシュが必要です。セクション13.3.5「LOCK TABLES および UNLOCK TABLES 構文」およびセクション13.7.6.3「FLUSH 構文」を参照してください。

サーバーが何も更新していないかぎり、すべてのテーブルファイルをコピーすることによって、バイナリバックアップを簡単に作成することもできます。mysqlhotcopy スクリプトはこの方法を使用します。(ただし、テーブルファイルコピー方法は、データベースに InnoDB テーブルが含まれている場合、機能しません。InnoDB では必ずしもデータベースディレクトリにテーブルの内容を格納しないため、mysqlhotcopy は InnoDB テーブルで機能しません。さらに、サーバーがアクティブにデータを更新していない場合、InnoDB は変更されたデータをまだメモリー内にキャッシュしており、ディスクにフラッシュしていないことがあります。)

区切りテキストファイルバックアップの作成

テーブルのデータを含むテキストファイルを作成するには、SELECT * INTO OUTFILE 'file_name' FROM tbl_name を使用できます。このファイルはクライアントホストではなく、MySQL サーバーホスト上に作成されます。このステートメントの場合、ファイルの上書きを許可すると、セキュリティーリスクになるため、出力ファイルがすでに存在していてはなりません。セクション13.2.9「SELECT 構文」を参照してください。この方法はあらゆる種類のデータファイルに機能しますが、テーブルデータのみ保存し、テーブル構造は保存しません。

テキストデータファイル (バックアップされたテーブルの CREATE TABLE ステートメントを含むファイルに加えて) を作成する別の方法は、mysqldump と --tab オプションを使用することです。セクション7.4.3「mysqldump による区切りテキストフォーマットでのデータのダンプ」を参照してください。

区切りテキストデータファイルをリロードするには、LOAD DATA INFILE または mysqlimport を使用します。

バイナリログを有効にすることによる増分バックアップの作成

MySQL は増分バックアップをサポートします。--log-bin オプションでサーバーを起動し、バイナリロギングを有効にする必要があります。セクション5.2.4「バイナリログ」を参照してください。バイナリログファイルは、バックアップを実行した時点のあとに行われたデータベースへの変更のレプリケートする必要がある情報を提供します。増分バックアップ (最後の完全バックアップまたは増分バックアップ以降に発生したすべての変更を含む) を作成しようとするときは、FLUSH LOGS を使用して、バイナリログをローテーションしてください。これが完了したら、最後の完全または増分バックアップの瞬間から最後の 1 つ前の範囲のすべてのバイナリログをバックアップの場所にコピーする必要があります。これらのバイナリログは増分バックアップで、リストア時に、セクション7.5「バイナリログを使用したポイントインタイム (増分) リカバリ」に説明するように、それらを適用します。次回に完全バックアップを行うときも、FLUSH LOGS、mysqldump --flush-logs、または mysqlhotcopy --flushlog を使用してバイナリログをローテーションしてください。セクション4.5.4「mysqldump

— データベースバックアッププログラム」およびセクション4.6.10「mysqlhotcopy — データベースバックアッププログラム」を参照してください。

レプリケーションスレーブを使用したバックアップの作成

バックアップの作成中に、マスターサーバーにパフォーマンスの問題がある場合、役に立つ可能性のある 1 つの戦略は、マスターではなくスレーブでレプリケーションをセットアップし、バックアップを実行することです。セクション17.3.1「バックアップ用にレプリケーションを使用する」を参照してください。

スレーブレプリケーションサーバーをバックアップする場合、選択したバックアップ方法に関係なく、スレーブのデータベースをバックアップする際に、そのマスター情報とリレーログ情報のリポジトリをバックアップしてください (セクション17.2.2「レプリケーションリレーおよびステータスログ」を参照してください)。これらの情報ファイルは、スレーブのデータをリストアしたあとに、レプリケーションを再開するために常に必要です。スレーブが LOAD DATA INFILE ステートメントをレプリケートする場合、スレーブがこのために使用するディレクトリ内に存在する SQL_LOAD-* ファイルもバックアップしてください。スレーブは、中断した LOAD DATA INFILE 操作のレプリケーションを再開するためにこれらのファイルを必要とします。このディレクトリの場所は --slave-load-tmpdir オプションの値です。そのオプションでサーバーを起動しなかった場合、ディレクトリの場所は tmpdir システム変数の値になります。

破損したテーブルのリカバリ

破損した MyISAM テーブルをリストアする必要がある場合、まず REPAIR TABLE または myisamchk -r を使用して、それらのリカバリを試みます。それは、すべてのケースの 99.9% で機能するはずです。myisamchk が失敗した場合は、セクション7.6「MyISAM テーブルの保守とクラッシュリカバリ」を参照してください。

ファイルシステムスナップショットを使用したバックアップの作成

Veritas ファイルシステムを使用している場合、次のようにバックアップを作成できます。

1. クライアントプログラムから、FLUSH TABLES WITH READ LOCK を実行します。
2. 別のシェルから、mount vxfs snapshot を実行します。
3. 最初のクライアントから、UNLOCK TABLES を実行します。
4. スナップショットからファイルをコピーします。
5. スナップショットをアンマウントします。

同様のスナップショット機能は、LVM や ZFS などのほかのファイルシステムでも利用できます。

7.3 バックアップおよびリカバリ戦略の例

このセクションでは、いくつかの種類のクラッシュ後にデータをリカバリできるようにするバックアップを実行するための手順について説明します。

- オペレーティングシステムのクラッシュ
- 停電
- ファイルシステムのクラッシュ
- ハードウェアの問題 (ハードドライブ、マザーボードなど)

コマンド例には、mysqldump および mysql クライアントプログラム用の --user や --password などのオプションは含まれていません。クライアントプログラムが MySQL サーバーに接続できるようにする必要に応じて、それらのオプションを含めてください。

データは、トランザクションと自動クラッシュリカバリをサポートする InnoDB ストレージエンジンに格納されているとします。さらに、MySQL サーバーはクラッシュ時に負荷がかかっているとします。そうでなければ、リカバリは必要ないことがあります。

オペレーティングシステムのクラッシュや停電の場合、再起動後、MySQL のディスクデータを使用できるものと考えることができます。InnoDB データファイルにはクラッシュのために一貫したデータが格納されていない可能性がありますが、InnoDB はその口グを読み取り、データファイルにフラッシュされていないコミット保留中のト

ランザクションやコミットされていないトランザクションのリストを見つけます。InnoDB はまだコミットされていないトランザクションを自動的にロールバックし、コミットされたものはデータファイルにフラッシュします。このリカバリプロセスに関する情報は、MySQL エラーログによってユーザーに伝えられます。次はログの抜粋の例です。

```
InnoDB: Database was not shut down normally.
InnoDB: Starting recovery from log files...
InnoDB: Starting log scan based on checkpoint at
InnoDB: log sequence number 0 13674004
InnoDB: Doing recovery: scanned up to log sequence number 0 13739520
InnoDB: Doing recovery: scanned up to log sequence number 0 13805056
InnoDB: Doing recovery: scanned up to log sequence number 0 13870592
InnoDB: Doing recovery: scanned up to log sequence number 0 13936128
...
InnoDB: Doing recovery: scanned up to log sequence number 0 20555264
InnoDB: Doing recovery: scanned up to log sequence number 0 20620800
InnoDB: Doing recovery: scanned up to log sequence number 0 20664692
InnoDB: 1 uncommitted transaction(s) which must be rolled back
InnoDB: Starting rollback of uncommitted transactions
InnoDB: Rolling back trx no 16745
InnoDB: Rolling back of trx no 16745 completed
InnoDB: Rollback of uncommitted transactions completed
InnoDB: Starting an apply batch of log records to the database...
InnoDB: Apply batch completed
InnoDB: Started
mysqld: ready for connections
```

ファイルシステムのクラッシュやハードウェアの問題の場合、再起動後、MySQL ディスクデータを使用できないものと考えることができます。これは、ディスクデータの一部のブロックが読み取り不可能になったため、MySQL が正常な起動に失敗することを意味します。この場合、ディスクを再フォーマットするか、新しいディスクをインストールするか、または根本的な問題を修正する必要があります。さらに、バックアップから MySQL データをリカバリする必要があります。これはバックアップがすでに行われていることを意味します。それが確実に当てはまるようにするには、バックアップポリシーを設計し、実装します。

7.3.1 バックアップポリシーの確立

役に立つように、バックアップは定期的にスケジュールする必要があります。完全バックアップ (特定の時点でのデータのスナップショット) は、MySQL でいくつかのツールを使用して実行できます。たとえば、MySQL Enterprise Backup は、InnoDB データファイルのバックアップ時にオーバーヘッドを最小にし、中断を防ぐ最適化を伴うインスタンス全体の物理バックアップを実行できます。mysqldump はオンライン論理バックアップを提供します。この説明では mysqldump を使用します。

負荷が少ない日曜日の午後 1 時に、次のコマンドを使用して、すべてのデータベースのすべての InnoDB テーブルの完全バックアップを作成するとします。

```
shell> mysqldump --single-transaction --all-databases > backup_sunday_1_PM.sql
```

mysqldump によって生成される結果の .sql ファイルには、あとでダンプしたテーブルのリロードに使用できる SQL INSERT ステートメントのセットが含まれます。

このバックアップ操作では、ダンプの最初ですべてのテーブルに対するグローバル読み取りロックを取得します (FLUSH TABLES WITH READ LOCK を使用して)。このロックが取得されるとすぐに、バイナリログの座標が読み取られ、ロックが解除されます。FLUSH ステートメントが発行されたときに長い更新ステートメントが実行中の場合、バックアップ操作はそれらのステートメントが終了するまで停止する可能性があります。その後、ダンプはロックフリーとなり、テーブルの読み取りと書き込みを妨げません。

先に、バックアップするテーブルは InnoDB テーブルであるとしたため、--single-transaction は、一貫性読み取りを使用し、mysqldump によって表示されたデータが変更されないことを保証します。(ほかのクライアントによる InnoDB テーブルへの変更は、mysqldump プロセスによって表示されません)。バックアップ操作に非トランザクションテーブルが含まれる場合、一貫性には、バックアップ中にそれらが変更されない必要があります。たとえば、mysql データベース内の MyISAM テーブルの場合、バックアップ中に、MySQL アカウントへの管理上の変更があってはなりません。

完全バックアップが必要ですが、それらを作成するために常に都合がよいとは限りません。それらは大きなバックアップファイルを生成し、生成に時間がかかります。それらは、連続した各完全バックアップに、前回の完全バックアップから変更されていない部分でもすべてのデータが含まれるという点で、最適ではありません。初期完全バックアップを作成し、次に増分バックアップを作成するほうが効率的です。増分バックアップは小さく、生成にかかる時間が少なくなります。このトレードオフは、リカバリ時に、完全バックアップをリロードするだけではデータをリストアできないことです。増分バックアップを処理して、増分の変更もリカバリする必要があります。

増分バックアップを作成するには、増分の変更を保存する必要があります。MySQL では、これらの変更はバイナリログで表されるため、MySQL サーバーを常に --log-bin オプションで起動して、そのログを有効にしてください。バイナリロギングが有効にされていると、サーバーはデータの更新中に、各データの変更をファイルに書き込みます。--log-bin オプションで起動し、数日間実行していた MySQL サーバーのデータディレクトリを調べると、これらの MySQL バイナリログファイルが見つかります。

```
-rw-rw---- 1 guilhem  guilhem   1277324 Nov 10 23:59 gbichot2-bin.000001
-rw-rw---- 1 guilhem  guilhem         4 Nov 10 23:59 gbichot2-bin.000002
-rw-rw---- 1 guilhem  guilhem        79 Nov 11 11:06 gbichot2-bin.000003
-rw-rw---- 1 guilhem  guilhem       508 Nov 11 11:08 gbichot2-bin.000004
-rw-rw---- 1 guilhem  guilhem 220047446 Nov 12 16:47 gbichot2-bin.000005
-rw-rw---- 1 guilhem  guilhem    998412 Nov 14 10:08 gbichot2-bin.000006
-rw-rw---- 1 guilhem  guilhem       361 Nov 14 10:07 gbichot2-bin.index
```

MySQL サーバーは再起動するたびに、シーケンスの次の番号を使用して、新しいバイナリログファイルを作成します。サーバーが実行している間、FLUSH LOGS SQL ステートメントを発行するか、mysqladmin flush-logs コマンドによって、手動で、それに現在のバイナリログファイルをクローズし、新しいファイルを開始するように伝えることもできます。mysqldump にはログをフラッシュするオプションもあります。データディレクトリ内の .index ファイルには、ディレクトリ内のすべての MySQL バイナリログのリストが含まれます。

MySQL バイナリログは、増分バックアップのセットを形成するため、リカバリに重要です。完全バックアップの作成時にログをフラッシュさせる場合、その後作成されるバイナリログファイルには、バックアップ以降に行われたすべてのデータの変更が含まれます。ここで、前述の mysqldump コマンドを少し修正して、完全バックアップの時点で MySQL バイナリログをフラッシュするようにし、ダンプファイルに新しい現在のバイナリログの名前が含まれるようにします。

```
shell> mysqldump --single-transaction --flush-logs --master-data=2 \
       --all-databases > backup_sunday_1_PM.sql
```

このコマンドの実行後、--flush-logs オプションによって、サーバーにそのログをフラッシュさせるため、データディレクトリには新しいバイナリログファイル gbichot2-bin.000007 が格納されます。--master-data オプションは mysqldump でその出力にバイナリログ情報を書き込ませるため、結果の .sql ダンプファイルにはこれらの行が含まれます。

```
-- Position to start replication or point-in-time recovery from
-- CHANGE MASTER TO MASTER_LOG_FILE='gbichot2-bin.000007',MASTER_LOG_POS=4;
```

mysqldump コマンドで完全バックアップを作成しているため、これらの行は 2 つのことを意味します。

- ダンプファイルには、gbichot2-bin.000007 以降のバイナリログファイルに書き込まれた変更の前に行われたすべての変更が含まれます。
- バックアップ後にログに記録されたすべてのデータの変更は、ダンプファイルに存在しませんが、gbichot2-bin.000007 以降のバイナリログファイルに存在します。

月曜日の午後 1 時に、ログをフラッシュし、新しいバイナリログファイルを開始することによって、増分バックアップを作成できます。たとえば、mysqladmin flush-logs コマンドを実行すると、gbichot2-bin.000008 が作成されます。日曜日の午後 1 時の完全バックアップから月曜日の午後 1 時までのすべての変更は、gbichot2-bin.000007 ファイル内にあります。この増分バックアップは重要であるため、それを安全な場所にコピーすることをお勧めします。(たとえば、それをテープや DVD にバックアップするか、別のマシンにコピーします。)火曜日の午後 1 時に、さらに mysqladmin flush-logs コマンドを実行します。月曜日の午後 1 時から火曜日の午後 1 時までのすべての変更が、gbichot2-bin.000008 ファイル内にあります (これもどこか安全な場所にコピーするべきです)。

MySQL バイナリログはディスク領域を占有します。領域を解放するため、ときどきそれらをパージします。これを実行する 1 つの方法は、完全バックアップを作成したときなど、必要なくなったバイナリログを削除することです。

```
shell> mysqldump --single-transaction --flush-logs --master-data=2 \
       --all-databases --delete-master-logs > backup_sunday_1_PM.sql
```

> 注記
>
> サーバーがレプリケーションマスターサーバーである場合、スレーブサーバーでまだバイナリログの内容を完全に処理していない可能性があるため、mysqldump --delete-master-logs で MySQL バイナリログを削除することは危険な場合があります。PURGE BINARY LOGS ステートメントの説明では、MySQL バイナリログを削除する前に確認すべきことを説明しています。セクション13.4.1.1「PURGE BINARY LOGS 構文」を参照してください。

7.3.2 リカバリへのバックアップの使用

ここで、水曜日の午前 8 時に、バックアップからのリカバリを必要とする致命的なクラッシュがあったとします。リカバリするには、まず存在する最後の完全バックアップ (日曜日の午後 1 時のもの) をリストアします。完全バックアップファイルは一連の SQL ステートメントにすぎないため、そのリストアはきわめて簡単です。

```
shell> mysql < backup_sunday_1_PM.sql
```

この時点で、データは日曜日の午後 1 時現在の状態にリストアされます。それ以降に行われた変更をリストアするには、増分バックアップを使用する必要があります。つまり、gbichot2-bin.000007 と gbichot2-bin.000008 バイナリログファイルです。必要に応じて、バックアップされた場所からファイルをフェッチして、次のようにそれらの内容を処理します。

```
shell> mysqlbinlog gbichot2-bin.000007 gbichot2-bin.000008 | mysql
```

これで、データを火曜日の午後 1 時現在の状態にリカバリしましたが、まだその日からクラッシュの日までの変更が不足しています。それらを失わないために、MySQL サーバーにその MySQL バイナリログを、そのデータファイルを格納している場所と異なる安全な場所 (RAID ディスク、SAN など) に保存させ、これらのログが破損したディスク上にないようにする必要がありました。(つまり、データディレクトリが存在する場所と別の物理デバイス上の場所を指定する --log-bin オプションでサーバーを起動できます。このようにすると、ディレクトリを格納するデバイスが失われてもログは安全です。)これを実行していた場合、gbichot2-bin.000009 ファイル (および任意の後続のファイル) が手元にあるため、mysqlbinlog と mysql を使用して、それらを適用し、クラッシュの瞬間まで損失なく、最新のデータの変更をリストアできます。

```
shell> mysqlbinlog gbichot2-bin.000009 ... | mysql
```

mysqlbinlog を使用して、バイナリログファイルを処理する詳細については、セクション7.5「バイナリログを使用したポイントインタイム (増分) リカバリ」を参照してください。

7.3.3 バックアップ戦略サマリー

オペレーティングシステムのクラッシュまたは停電の場合、InnoDB 自体がデータのリカバリのすべてのジョブを実行します。ただし、安心のため、次のガイドラインを参照してください。

- 常に MySQL サーバーを --log-bin オプションまたは --log-bin=log_name で実行します。ここでログファイル名は、データディレクトリが存在するドライブと異なる安全なメディア上に配置します。そのような安全なメディアがある場合、この技法は、ディスクの負荷分散にも役立ちます (その結果パフォーマンスも向上します)。
- セクション7.3.1「バックアップポリシーの確立」で先に示した、オンラインのブロックしないバックアップを作成する mysqldump コマンドを使用して、定期的な完全バックアップを作成します。
- FLUSH LOGS または mysqladmin flush-logs を使用して、ログをフラッシュして、定期的な増分バックアップを作成します。

7.4 バックアップへの mysqldump の使用

このセクションでは、mysqldump を使用して、ダンプファイルを生成する方法およびダンプファイルをリロードする方法について説明します。ダンプファイルはいくつかの方法で使用できます。

- データ損失の場合にデータリカバリを可能にするためのバックアップとして。
- レプリケーションスレーブをセットアップするためのデータのソースとして。
- 実験用のデータのソースとして。
  - 元のデータを変更せずに使用できるデータベースのコピーを作成する場合。
  - アップグレードの非互換性の可能性をテストする場合。

mysqldump は --tab オプションを指定するかどうかに応じて、2 種類の出力を生成します。

- --tab がないと、mysqldump は SQL ステートメントを標準出力に書き込みます。この出力は、ダンプされるオブジェクト (データベース、テーブル、ストアドルーチンなど) を作成する CREATE ステートメントとデータをテーブルにロードする INSERT ステートメントから構成されます。出力はファイルに保存して、あとで mysql を使用してリロードし、ダンプされたオブジェクトを再作成できます。SQL ステートメントのフォーマットを変更し、ダンプされるオブジェクトを制御するためにオプションを使用できます。

- --tab を付けると、mysqldump はダンプされるテーブルごとに 2 つの出力ファイルを生成します。サーバーは、テーブル行ごとに 1 行ずつ、タブ区切りテキストとして 1 つのファイルを書き込みます。このファイルは出力ディレクトリ内で tbl_name.txt という名前が付けられます。サーバーはテーブルの CREATE TABLE ステートメントも mysqldump に送信し、それは tbl_name.sql という名前のファイルとしてそれを出力ディレクトリに書き込みます。

7.4.1 mysqldump による SQL フォーマットでのデータのダンプ

このセクションでは、mysqldump を使用して、SQL フォーマットのダンプファイルを作成する方法について説明します。そのようなダンプファイルのリロードについては、セクション7.4.2「SQL フォーマットバックアップのリロード」を参照してください。

デフォルトで、mysqldump は情報を SQL ステートメントとして標準出力に書き込みます。出力をファイルに保存できます。

```
shell> mysqldump [arguments] > file_name
```

すべてのデータベースをダンプするには、--all-databases オプションを付けて mysqldump を呼び出します。

```
shell> mysqldump --all-databases > dump.sql
```

特定のデータベースのみをダンプするには、コマンド行でそれらを指定し、--databases オプションを使用します。

```
shell> mysqldump --databases db1 db2 db3 > dump.sql
```

--databases オプションによって、コマンド行上のすべての名前がデータベース名として扱われます。このオプションを使用しないと、mysqldump は最初の名前をデータベース名として、そのあとに続く名前をテーブル名として扱います。

--all-databases または --databases を使用すると、mysqldump は、各データベースのダンプ出力の前に、CREATE DATABASE および USE ステートメントを書き込みます。これにより、ダンプファイルがリロードされると、それが各データベースが存在しなければ作成して、デフォルトのデータベースにするため、データベースの内容がそれらの作成元の同じデータベースにロードされます。ダンプファイルに、各データベースを再作成する前にその削除を強制する場合、--add-drop-database オプションも使用します。この場合、mysqldump は各 CREATE DATABASE ステートメントの前に、DROP DATABASE ステートメントを書き込みます。

単一のデータベースをダンプするには、コマンド行でそれを指定します。

```
shell> mysqldump --databases test > dump.sql
```

単一のデータベースの場合、--databases オプションを省略できます。

```
shell> mysqldump test > dump.sql
```

2 つの先述のコマンドの違いは、--databases を付けないと、ダンプの出力に CREATE DATABASE または USE ステートメントが含まれません。これにはいくつかの問題があります。

- ダンプファイルをリロードする場合、サーバーがリロードするデータベースを認識するように、デフォルトのデータベース名を指定する必要があります。
- リロードする場合、元の名前と異なるデータベース名を指定でき、これにより、データを別のデータベースにリロードできます。
- リロードするデータベースが存在しない場合、まずそれを作成する必要があります。
- 出力には CREATE DATABASE ステートメントが含まれないため、--add-drop-database オプションは無効です。それを使用しても DROP DATABASE ステートメントは生成されません。

データベースから特定のテーブルのみをダンプするには、コマンド行でデータベース名に続いてそれらを指定します。

```
shell> mysqldump test t1 t3 t7 > dump.sql
```

7.4.2 SQL フォーマットバックアップのリロード

SQL ステートメントから構成される mysqldump によって書き込まれたダンプファイルをリロードするには、それを mysql クライアントへの入力として使用します。--all-databases または --databases オプションを使用し

て、mysqldump によってダンプファイルが作成された場合、それには CREATE DATABASE および USE ステートメントが含まれ、データをロードするデフォルトのデータベースを指定する必要がありません。

```
shell> mysql < dump.sql
```

または、mysql 内から、source コマンドを使用します。

```
mysql> source dump.sql
```

ファイルが CREATE DATABASE および USE ステートメントを含まない単一データベースダンプである場合、まずデータベースを作成します (必要に応じて)。

```
shell> mysqladmin create db1
```

次に、ダンプファイルをロードする場合、データベース名を指定します。

```
shell> mysql db1 < dump.sql
```

または mysql 内から、データベースを作成し、それをデフォルトのデータベースとして選択し、ダンプファイルをロードします。

```
mysql> CREATE DATABASE IF NOT EXISTS db1;
mysql> USE db1;
mysql> source dump.sql
```

7.4.3 mysqldump による区切りテキストフォーマットでのデータのダンプ

このセクションでは、mysqldump を使用して、区切りテキストのダンプファイルを作成する方法について説明します。そのようなダンプファイルのリロードについては、セクション7.4.4「区切りテキストフォーマットバックアップのリロード」を参照してください。

--tab=dir_name オプションを付けて、mysqldump を呼び出した場合、それは dir_name を出力ディレクトリとして使用し、テーブルごとに 2 つのファイルを使用して、そのディレクトリに個別にテーブルをダンプします。テーブル名はこれらのファイルのベース名です。t1 という名前のテーブルの場合、ファイルには t1.sql および t1.txt という名前が付けられます。.sql ファイルにはテーブルの CREATE TABLE ステートメントが含まれます。.txt ファイルにはテーブル行ごとに 1 行のテーブルデータが含まれます。

次のコマンドは db1 データベースの内容を /tmp データベース内のファイルにダンプします。

```
shell> mysqldump --tab=/tmp db1
```

テーブルデータを格納する .txt ファイルはサーバーによって書き込まれるため、それらはサーバーの実行に使用されるシステムアカウントによって所有されます。サーバーは SELECT ... INTO OUTFILE を使用して、ファイルを書き込むため、この操作を実行するために FILE 権限が必要であり、指定した .txt ファイルがすでに存在する場合はエラーが発生します。

サーバーはダンプされるテーブルの CREATE 定義を mysqldump に送信し、それはそれらを .sql ファイルに書き込みます。そのためこれらのファイルは、mysqldump を実行するユーザーによって所有されます。

--tab はローカルサーバーのダンプにのみ使用することをお勧めします。それをリモートサーバーに使用する場合、--tab ディレクトリがローカルホストとリモートホストの両方に存在する必要があり、.txt ファイルはサーバーによってリモートディレクトリ (サーバーホスト上) に書き込まれ、.sql ファイルは mysqldump によってローカルディレクトリ (クライアントホスト上) に書き込まれます。

mysqldump --tab の場合、サーバーはデフォルトでテーブルデータを、カラム値間にタブを、カラム値を引用符で囲まず、行ターミネータとして改行を使用して、1 行あたり 1 行で .txt ファイルに書き込みます。(これらは、SELECT ... INTO OUTFILE の場合と同じデフォルトです。)

別のフォーマットを使用して、データファイルを書き込めるようにするため、mysqldump はこれらのオプションをサポートしています。

- --fields-terminated-by=str

  カラム値を区切るための文字列 (デフォルト: タブ)。

- --fields-enclosed-by=char

  カラム値を囲む文字 (デフォルト: 文字なし)。

- --fields-optionally-enclosed-by=char

数値以外のカラム値を囲む文字 (デフォルト: 文字なし)。

- --fields-escaped-by=char

  特殊文字をエスケープするための文字 (デフォルト: エスケープなし)。

- --lines-terminated-by=str

  行終了文字列 (デフォルト: 改行)。

これらのオプションに指定する値に応じて、コマンド行で、コマンドインタプリタに合わせて値を引用符で囲むかエスケープする必要がある場合があります。または、16 進表記を使用して、値を指定します。mysqldump にカラム値を二重引用符で囲ませたいとします。そうするには、--fields-enclosed-by オプションの値として、二重引用符を指定します。ただし、この文字は多くの場合コマンドインタプリタに特有であるため、特別に扱う必要があります。たとえば、Unix ではこのように二重引用符を表すことができます。

```
--fields-enclosed-by=""
```

どのプラットフォームでも 16 進で値を指定できます。

```
--fields-enclosed-by=0x22
```

複数のデータフォーマットオプションを一緒に使用することもよくあります。たとえば、行を改行文字/復帰改行ペア (\r\n) で終了させたカンマ区切り値フォーマットでテーブルをダンプするには、このコマンドを使用します (1 行で入力します)。

```
shell> mysqldump --tab=/tmp --fields-terminated-by=,
       --fields-enclosed-by="" --lines-terminated-by=0x0d0a db1
```

テーブルデータをダンプするために、いずれかのデータフォーマットオプションを使用する場合、あとでデータファイルをリロードするときに、ファイルの内容が正しく解釈されるように、同じフォーマットを指定する必要があります。

7.4.4 区切りテキストフォーマットバックアップのリロード

mysqldump --tab によって生成されるバックアップの場合、各テーブルは出力ディレクトリに、テーブルの CREATE TABLE ステートメントを含む .sql ファイルと、テーブルデータを含む .txt ファイルで表されます。テーブルをリロードするには、まず場所を出力ディレクトリに変更します。次に、mysql で .sql ファイルを処理し、空のテーブルを作成し、.txt ファイルを処理して、データをテーブルにロードします。

```
shell> mysql db1 < t1.sql
shell> mysqlimport db1 t1.txt
```

mysqlimport を使用してデータファイルをロードする代替の方法は、mysql クライアント内から LOAD DATA INFILE ステートメントを使用することです。

```
mysql> USE db1;
mysql> LOAD DATA INFILE 't1.txt' INTO TABLE t1;
```

テーブルを最初にダンプしたときに、mysqldump で何らかのデータフォーマットオプションを使用した場合、mysqlimport または LOAD DATA INFILE で同じオプションを使用して、データファイルの内容が正しく解釈されるようにする必要があります。

```
shell> mysqlimport --fields-terminated-by=,
       --fields-enclosed-by="" --lines-terminated-by=0x0d0a db1 t1.txt
```

または:

```
mysql> USE db1;
mysql> LOAD DATA INFILE 't1.txt' INTO TABLE t1
    -> FIELDS TERMINATED BY ',' FIELDS ENCLOSED BY ""
    -> LINES TERMINATED BY '\r\n';
```

7.4.5 mysqldump のヒント

このセクションでは、mysqldump を使用して特定の問題を解決できる技法を調査します。

- データベースのコピーの作成方法
- サーバー間でデータベースをコピーする方法

- ストアドプログラム (ストアドプロシージャーおよび関数、トリガー、およびイベント) をダンプする方法
- 定義とデータを個別にダンプする方法

7.4.5.1 データベースのコピーの作成

```
shell> mysqldump db1 > dump.sql
shell> mysqladmin create db2
shell> mysql db2 < dump.sql
```

mysqldump コマンド行に --databases を使用すると、ダンプファイルに USE db1 が含まれ、それによって mysql コマンド行の db2 の指定の効果がオーバーライドされるため、使用しないでください。

7.4.5.2 サーバー間でのデータベースのコピー

サーバー 1 で:

```
shell> mysqldump --databases db1 > dump.sql
```

サーバー 1 からサーバー 2 にダンプファイルをコピーします。

サーバー 2 で:

```
shell> mysql < dump.sql
```

mysqldump コマンド行で --databases を使用すると、それが存在する場合にデータベースを作成し、それをリロードされるデータのデフォルトのデータベースにする CREATE DATABASE および USE ステートメントがダンプファイルに含まれます。

または、mysqldump コマンドから --databases を省略できます。次に、必要に応じて、サーバー 2 にデータベースを作成し、それをダンプファイルのリロード時のデフォルトのデータベースとして指定する必要があります。

サーバー 1 で:

```
shell> mysqldump db1 > dump.sql
```

サーバー 2 で:

```
shell> mysqladmin create db1
shell> mysql db1 < dump.sql
```

この場合、別のデータベース名を指定できるため、mysqldump コマンドから --databases を省略すると、あるデータベースからデータをダンプし、別のデータベースにそれをロードすることができます。

7.4.5.3 ストアドプログラムのダンプ

いくつかのオプションは、mysqldump がストアドプログラム (ストアドプロシージャーおよび関数、トリガー、およびイベント) を処理する方法を制御します。

- --events: イベントスケジューラのイベントのダンプ
- --routines: ストアドプロシージャーおよびストアドファンクションのダンプ
- --triggers: テーブルのトリガーのダンプ

テーブルがダンプされるときに、それらにそれらが持ついずれかのトリガーが伴うように、--triggers オプションはデフォルトで有効にされています。ほかのオプションはデフォルトで無効にされ、対応するオブジェクトをダンプするために明示的に指定する必要があります。これらのオプションのいずれかを明示的に無効にするには、そのスキップフォーム --skip-events、--skip-routines、または --skip-triggers を使用します。

7.4.5.4 テーブル定義と内容の個別のダンプ

--no-data オプションは mysqldump にテーブルデータをダンプしないように伝えるため、ダンプファイルにはテーブルを作成するステートメントのみが含まれます。逆に、--no-create-info オプションは、ダンプファイルにテーブルデータのみが含まれるように、mysqldump に出力から CREATE ステートメントを抑制するように伝えます。

たとえば、test データベースのテーブル定義とデータを別々にダンプするには、これらのコマンドを使用します。

```
shell> mysqldump --no-data test > dump-defs.sql
shell> mysqldump --no-create-info test > dump-data.sql
```

定義のみのダンプの場合、ストアドルーチンとイベントの定義も含めるには、--routines および --events オプションを追加します。

```
shell> mysqldump --no-data --routines --events test > dump-defs.sql
```

7.4.5.5 mysqldump を使用したアップグレードの非互換性のテスト

MySQL のアップグレードを検討する場合、新しいバージョンを現在の本番バージョンとべつにインストールすることが賢明です。これによって、本番サーバーからデータベースとデータベースオブジェクト定義をダンプし、新しいサーバーにロードして、それらが正しく処理されることを確認できます。(これはダウングレードのテストの場合にも役立ちます。)

本番サーバーで:

```
shell> mysqldump --all-databases --no-data --routines --events > dump-defs.sql
```

アップグレードされたサーバーで:

```
shell> mysql < dump-defs.sql
```

ダンプファイルにはテーブルデータが含まれないため、すばやく処理できます。これにより、長いデータロード操作を待つことなく、可能性のある非互換性を見分けることができます。ダンプファイルの処理中の警告やエラーを探します。

定義が正しく処理されていることを確認したら、データをダンプし、それをアップグレードしたサーバーにロードしてみます。

本番サーバーで:

```
shell> mysqldump --all-databases --no-create-info > dump-data.sql
```

アップグレードされたサーバーで:

```
shell> mysql < dump-data.sql
```

ここでテーブルの内容を確認し、いくつかのテストクエリーを実行します。

7.5 バイナリログを使用したポイントインタイム (増分) リカバリ

ポイントインタイムリカバリは、特定の時点以降に行われたデータ変更のリカバリを表します。一般に、この種類のリカバリは、サーバーをバックアップが行われた時点の状態にする完全バックアップのリストア後に実行されます。(完全バックアップは、セクション7.2「データベースバックアップ方法」に示すものなど、いくつかの方法で行うことができます。)さらに、ポイントインタイムリカバリは、完全バックアップの時点からより最近の時点まで、増分的にサーバーを最新にします。

ポイントインタイムリカバリは、これらの原則に基づきます。

- ポイントインタイムリカバリの情報のソースは、完全バックアップ操作のあとに生成されたバイナリログファイルによって表される一連の増分バックアップです。そのため、サーバーを --log-bin オプションで起動して、バイナリロギングを有効にする必要があります (セクション5.2.4「バイナリログ」を参照してください)。

  バイナリログからデータをリストアするには、現在のバイナリログファイルの名前と場所を知っている必要があります。デフォルトで、サーバーはデータディレクトリにバイナリログファイルを作成しますが、--log-bin オプションでパス名を指定して、別の場所にファイルを配置できます。セクション5.2.4「バイナリログ」。

  すべてのバイナリログファイルのリストを表示するには、次のステートメントを使用します。

  ```
  mysql> SHOW BINARY LOGS;
  ```

  現在のバイナリログファイルの名前を判断するには、次のステートメントを発行します。

  ```
  mysql> SHOW MASTER STATUS;
  ```

- mysqlbinlog ユーティリティーは、バイナリログファイル内のイベントを、実行したり、表示したりできるように、バイナリフォーマットからテキストに変換します。mysqlbinlog には、イベント時間や口グ内のイベントの位置に基づいて、バイナリログのセクションを選択するためのオプションがあります。セクション4.6.8「mysqlbinlog — バイナリログファイルを処理するためのユーティリティー」を参照してください。

- バイナリログからイベントを実行すると、それらが表すデータの変更が再実行されます。これにより、特定の期間のデータの変更のリカバリが可能です。バイナリログからイベントを実行するには、mysql クライアントを使用して、mysqlbinlog 出力を処理します。

```
shell> mysqlbinlog binlog_files | mysql -u root -p
```

- ログの内容を表示すると、イベントを実行する前に、イベントの時間や位置を特定して、ログの内容の一部を選択する必要がある場合に役立つことがあります。ログからイベントを表示するには、mysqlbinlog 出力をページングプログラムに送信します。

```
shell> mysqlbinlog binlog_files | more
```

  または、出力をファイルに保存し、テキストエディタでファイルを表示します。

```
shell> mysqlbinlog binlog_files > tmpfile
shell> ... edit tmpfile ...
```

- ファイルに出力を保存することは、予期しない DROP DATABASE など、特定のイベントが削除されたログの内容を実行する場合の予備として役立ちます。ファイルの内容を実行する前に、実行されないステートメントをファイルから削除できます。ファイルの編集後、次のように内容を実行します。

```
shell> mysql -u root -p < tmpfile
```

MySQL サーバーに実行する複数のバイナリログがある場合、安全な方法は、サーバーへの 1 つの接続を使用して、それらすべてを処理することです。これは、安全でない可能性があることを示す例です。

```
shell> mysqlbinlog binlog.000001 | mysql -u root -p # DANGER!!
shell> mysqlbinlog binlog.000002 | mysql -u root -p # DANGER!!
```

最初のログファイルに CREATE TEMPORARY TABLE ステートメントが含まれており、2 番目のログに一時テーブルを使用するステートメントが含まれている場合、サーバーへの異なる接続を使用して、このようにバイナリログを処理すると問題が発生します。最初の mysql プロセスが終了すると、サーバーは一時テーブルを削除します。2 番目の mysql プロセスでテーブルの使用を試みると、サーバーは「不明なテーブル」と報告します。

このような問題を回避するには、1 つの接続を使用して、処理するすべてのバイナリログの内容を実行します。これはそれを実行する 1 つの方法です。

```
shell> mysqlbinlog binlog.000001 binlog.000002 | mysql -u root -p
```

もう 1 つのアプローチは、すべての口グを 1 つのファイルに書き込み、次にそのファイルを処理することです。

```
shell> mysqlbinlog binlog.000001 >  /tmp/statements.sql
shell> mysqlbinlog binlog.000002 >> /tmp/statements.sql
shell> mysql -u root -p -e "source /tmp/statements.sql"
```

GTID (セクション17.1.3「グローバルトランザクション識別子を使用したレプリケーション」を参照) を含むバイナリログから読み取りながら、ダンプファイルに書き込む場合、次のように、mysqlbinlog で --skip-gtids オプションを使用します。

```
shell> mysqlbinlog --skip-gtids binlog.000001 >  /tmp/dump.sql
shell> mysqlbinlog --skip-gtids binlog.000002 >> /tmp/dump.sql
shell> mysql -u root -p -e "source /tmp/dump.sql"
```

7.5.1 イベント時間を使用したポイントインタイムリカバリ

リカバリの開始時間と終了時間を指示するには、mysqlbinlog に --start-datetime および --stop-datetime オプションを DATETIME フォーマットで指定します。例として、2005 年 4 月 20 日午前 10 時ちょうどに、大きなテーブルを削除する SQL ステートメントが実行されたとします。テーブルとデータをリストアするには、前夜のバックアップをリストアして、次のコマンドを実行します。

```
shell> mysqlbinlog --stop-datetime="2005-04-20 9:59:59" \
    /var/log/mysql/bin.123456 | mysql -u root -p
```

このコマンドは、すべてのデータを --stop-datetime オプションで指定された日時までリカバリします。数時間たって、入力された誤った SQL ステートメントを検出しなかった場合、その後発生したアクティビティーもリカバリする必要がある可能性があります。これに基づいて、次のように開始日時で mysqlbinlog を再度実行できます。

```
shell> mysqlbinlog --start-datetime="2005-04-20 10:01:00" \
    /var/log/mysql/bin.123456 | mysql -u root -p
```

このコマンドでは、午前 10:01 から記録された SQL ステートメントが再実行されます。前夜のダンプファイルのリストアと 2 つの mysqlbinlog コマンドの組み合わせでは、午前 10:00 の 1 秒前までのすべてのものと午前 10:01 からのすべてのものをリストアします。

ポイントインタイムリカバリのこの方法を使用するには、ログを調べて、コマンドに指定する正確な時間を確認してください。ログファイルの内容を実行せずに表示するには、次のコマンドを使用します。

```
shell> mysqlbinlog /var/log/mysql/bin.123456 > /tmp/mysql_restore.sql
```

次に、テキストエディタで /tmp/mysql_restore.sql ファイルを開き、それを調べます。

mysqlbinlog に時間を指定して、特定の変更を除外することは、除外されるステートメントと同時に複数のステートメントが実行された場合、うまく機能しません。

7.5.2 イベントの位置を使用したポイントインタイムリカバリ

日時を指定する代わりに、ログ位置を指定するために、mysqlbinlog で --start-position および --stop-position オプションを使用できます。それらは、日付ではなく、ログの位置番号を指定することを除いて、開始日および停止日オプションと同じように機能します。位置を使用することで、特に損害を与える SQL ステートメントと同じころに多くのトランザクションが発生している場合に、リカバリするログの部分をより正確に把握できます。位置番号を確認するには、予期していないトランザクションが実行された時間付近の期間で mysqlbinlog を実行しますが、結果を調査のためにテキストファイルにリダイレクトします。これは次のように実行できます。

```
shell> mysqlbinlog --start-datetime="2005-04-20 9:55:00" \
       --stop-datetime="2005-04-20 10:05:00" \
       /var/log/mysql/bin.123456 > /tmp/mysql_restore.sql
```

このコマンドは、/tmp ディレクトリに、有害な SQL ステートメントが実行された時間付近の SQL ステートメントを含む小さなテキストファイルを作成します。テキストエディタでこのファイルを開き、繰り返したくないステートメントを探します。リカバリを停止し、再開するバイナリログの位置を特定し、それらを書きとめます。位置は、うしろに番号が付いた log_pos とラベル付けされます。前のバックアップファイルのリストア後、位置番号を使用して、バイナリログファイルを処理します。たとえば、これらのようなコマンドを使用します。

```
shell> mysqlbinlog --stop-position=368312 /var/log/mysql/bin.123456 \
       | mysql -u root -p

shell> mysqlbinlog --start-position=368315 /var/log/mysql/bin.123456 \
       | mysql -u root -p
```

最初のコマンドは、指定した停止位置まで、すべてのトランザクションをリカバリします。2 番目のコマンドは、バイナリログの最後まで、指定された開始位置からのすべてのトランザクションをリカバリします。mysqlbinlog の出力には、各 SQL ステートメントが記録される前に、SET TIMESTAMP ステートメントが含まれるため、リカバリされたデータおよび関連する MySQL ログは、トランザクションが実行された元の時間を反映します。

7.6 MyISAM テーブルの保守とクラッシュリカバリ

このセクションでは、myisamchk を使用して、MyISAM テーブル (データとインデックスを格納するための .MYD ファイルおよび .MYI ファイルのあるテーブル) をチェックまたは修復する方法について説明します。myisamchk の一般的な背景に関しては、セクション4.6.3「myisamchk — MyISAM テーブルメンテナンスユーティリティー」を参照してください。その他のテーブル修復情報については、セクション2.11.4「テーブルまたはインデックスの再作成または修復」にあります。

myisamchk を使用して、データベーステーブルをチェック、修復、または最適化できます。次のセクションでは、これらの操作を実行する方法と、テーブル保守スケジュールをセットアップする方法について説明します。myisamchk を使用して、テーブルに関する情報を取得することについては、セクション4.6.3.5「myisamchk によるテーブル情報の取得」を参照してください。

myisamchk によるテーブルの修復はきわめてセキュアですが、テーブルに対して多くの変更を行う可能性のある修復または保守操作を実行する前に、常にバックアップを作成することをお勧めします。

インデックスに影響する myisamchk 操作により、MyISAM FULLTEXT インデックスが、MySQL サーバーで使用されている値と互換性がない全文パラメータで再構築される可能性があります。この問題を回避するには、セクション4.6.3.1「myisamchk の一般オプション」のガイドラインに従ってください。

MyISAM テーブルの保守は、myisamchk が実行するものと似た操作を実行する SQL ステートメントを使用しても実行できます。

- MyISAM テーブルをチェックするには、CHECK TABLE を使用します。
- MyISAM テーブルを修復するには、REPAIR TABLE を使用します。
- MyISAM テーブルを最適化するには、OPTIMIZE TABLE を使用します。
- MyISAM テーブルを分析するには、ANALYZE TABLE を使用します。

これらのステートメントの詳細については、セクション13.7.2「テーブル保守ステートメント」を参照してください。

これらのステートメントは、直接または mysqlcheck クライアントプログラムを利用して使用できます。myisamchk に勝るこれらのステートメントの利点の 1 つは、サーバーがすべての作業を行うということです。myisamchk では、myisamchk とサーバー間で不要なやり取りがないように、サーバーが同時にテーブルを使用しないことを確認する必要があります。

7.6.1 クラッシュリカバリへの myisamchk の使用

このセクションでは、MySQL データベースのデータの破損をチェックし、処理する方法について説明します。テーブルが頻繁に破損する場合は、その理由を見つけるようにしてください。セクションB.5.4.2「MySQL が繰り返しクラッシュする場合の対処方法」を参照してください。

MyISAM テーブルがどのように破損する可能性があるかについての説明は、セクション15.2.4「MyISAM テーブルの問題点」を参照してください。

外部ロックを無効にして mysqld を実行した (これはデフォルトです) 場合、mysqld が同じテーブルを使用中に、myisamchk を使用して、テーブルを確実にチェックすることはできません。myisamchk を実行している間、mysqld 経由でテーブルにだれもアクセスしないことを確信できる場合は、テーブルのチェックを開始する前に、mysqladmin flush-tables を実行する必要があるだけです。これを保証できない場合は、テーブルのチェック中に、mysqld を停止する必要があります。myisamchk を実行して、mysqld が同時に更新しているテーブルをチェックすると、テーブルが破損していなくても、破損しているという警告を受け取ることがあります。

外部ロックを有効にしてサーバーを実行する場合は、myisamchk を使用していつでもテーブルをチェックできます。この場合に、サーバーが myisamchk で使用しているテーブルを更新しようとする場合、サーバーは myisamchk が終了するまで待ってから、続行します。

myisamchk を使用して、テーブルを修復または最適化する場合は、mysqld サーバーがそのテーブルを使用していないことを常に確認する必要があります (これは外部ロックが無効にされている場合にも適用されます)。mysqld を停止しない場合、myisamchk を実行する前に、少なくとも mysqladmin flush-tables を実行してください。サーバーと myisamchk が同時にテーブルにアクセスすると、テーブルが破損する可能性があります。

クラッシュリカバリを実行する場合、データベース内の各 MyISAM テーブル tbl_name が次の表に示すデータベースディレクトリ内の 3 つのファイルに対応することを理解しておくことが重要です。

ファイル	目的
tbl_name.frm	定義 (フォーマット) ファイル
tbl_name.MYD	データファイル
tbl_name.MYI	インデックスファイル

これらの 3 つのファイルの種類はそれぞれさまざまに破損することがありますが、ほとんどの場合に問題はデータファイルとインデックスファイルで発生します。

myisamchk は、.MYD データファイルのコピーを行ごとに作成することによって機能します。これは、古い .MYD ファイルを削除し、新しいファイルを元のファイル名に変更して、修復ステージを終了します。--quick を使用した場合、myisamchk は一時 .MYD ファイルを作成せず、代わりに .MYD ファイルが正しいとみなし、.MYD ファイルに手を加えずに新しいインデックスファイルだけを生成します。myisamchk は .MYD ファイルが破損しているかどうかを自動的に検出し、破損している場合は修復を中止するため、これは安全です。myisamchk に --quick オプションを 2 回指定することもできます。この場合、myisamchk は一部のエラー (重複キーエラーなど) で中止せず、.MYD ファイルを修正して、それらを解決しようとします。通常、2 つの --quick オプションの使用は、通常の修復を実行するためにディスクの空き容量が少なすぎる場合にのみ役立ちます。その場合、少なくとも myisamchk を実行する前に、テーブルのバックアップを作成してください。

7.6.2 MyISAM テーブルのエラーのチェック方法

MyISAM テーブルをチェックするには、次のコマンドを使用します。

- myisamchk tbl_name

  これはすべてのエラーの 99.99% を発見します。これで発見できないエラーは、データファイルのみに関連する破損です (きわめてまれです)。テーブルをチェックする場合、通常、myisamchk をオプションなし、または -s (サイレント) オプションで実行してください。

- myisamchk -m tbl_name

  これはすべてのエラーの 99.999% を発見します。それは最初にすべてのインデックスエントリでエラーをチェックし、次にすべての行を読み取ります。それは行内のすべてのキー値のチェックサムを計算し、チェックサムがインデックスツリー内のキーのチェックサムと一致することを確認します。

- myisamchk -e tbl_name

  これはすべてのデータの完全で徹底的なチェックを実行します (-e は「拡張チェック」を意味します)。それは各行のすべてのキーのチェック読み取りを実行し、それらが実際に正しい行を指していることを確認します。これは、多数のインデックスを持つ大きなテーブルの場合に長い時間がかかることがあります。通常、myisamchk は見つかった最初のエラーのあとで停止します。詳細情報を取得する場合は、-v (verbose) オプションを追加できます。これにより、myisamchk は最大 20 のエラーまで続行します。

- myisamchk -e -i tbl_name

  これは、前述のコマンドと同様ですが、-i オプションは myisamchk に追加の統計情報を出力するように伝えます。

ほとんどの場合、テーブルをチェックするためには、テーブル名以外の引数なしの単純な myisamchk コマンドで十分です。

7.6.3 MyISAM テーブルの修復方法

このセクションの説明では、MyISAM テーブル (拡張子 .MYI および .MYD) に対し myisamchk を使用する方法について説明します。

さらに、CHECK TABLE および REPAIR TABLE ステートメントを使用して、MyISAM テーブルをチェックして修復することもできます。セクション13.7.2.2「CHECK TABLE 構文」およびセクション13.7.2.5「REPAIR TABLE 構文」を参照してください。

破損したテーブルの兆候として、予期せずに中止するクエリーや次のような観察可能なエラーが含まれます。

- tbl_name.frm が変更に対してロックされます
- ファイル tbl_name.MYI が見つかりません (エラーコード: nnn)。
- 予期しないファイルの終わり
- レコードファイルがクラッシュしました
- テーブルハンドラからエラー nnn を取得します

エラーの詳細を取得するには、perror nnn を実行します。ここで、nnn はエラー番号です。次の例は、perror を使用して、テーブルの問題を示すもっとも一般的なエラー番号の意味を見つける方法を示しています。

```
shell> perror 126 127 132 134 135 136 141 144 145
MySQL error code 126 = Index file is crashed
MySQL error code 127 = Record-file is crashed
MySQL error code 132 = Old database file
MySQL error code 134 = Record was already deleted (or record file crashed)
MySQL error code 135 = No more room in record file
MySQL error code 136 = No more room in index file
MySQL error code 141 = Duplicate unique key or constraint on write or update
MySQL error code 144 = Table is crashed and last repair failed
MySQL error code 145 = Table was marked as crashed and should be repaired
```

エラー 135 (レコードファイルに空きがない) およびエラー 136 (インデックスファイルに空きがない) は、単純な修復で修正できるエラーではありません。この場合、ALTER TABLE を使用して、MAX_ROWS および AVG_ROW_LENGTH テーブルオプションの値を増やす必要があります。

```
ALTER TABLE tbl_name MAX_ROWS=xxx AVG_ROW_LENGTH=yyy;
```

現在のテーブルオプション値が不明な場合は、SHOW CREATE TABLE を使用します。

その他のエラーの場合は、テーブルを修復する必要があります。myisamchk は通常発生するほとんどの問題を検出し、修正できます。

修復プロセスには、ここに示す最大 4 つのステージがあります。始める前に、場所をデータベースディレクトリに変更し、テーブルファイルの権限をチェックしてください。Unix では、mysqld を実行するユーザーによって (およびチェックするファイルにアクセスする必要があるため、チェックするユーザーにも)、それらが読み取り可能であることを確認します。ファイルを変更する必要があることが分かったら、それらに書き込みできる必要もあります。

このセクションでは、テーブルチェックが失敗した (セクション7.6.2「MyISAM テーブルのエラーのチェック方法」 で説明しているものなど) 場合、または myisamchk が提供する拡張機能を使用する場合について説明します。

テーブル保守に使用される myisamchk オプションについては、セクション4.6.3「myisamchk — MyISAM テーブルメンテナンスユーティリティー」で説明しています。myisamchk には、パフォーマンスを向上できるメモリー割り当てを制御するために設定できる変数もあります。セクション4.6.3.6「myisamchk メモリー使用量」を参照してください。

コマンド行からテーブルを修復する場合は、まず mysqld サーバーを停止する必要があります。リモートサーバーで mysqladmin shutdown を実行すると、mysqladmin が戻ったあとに、すべてのステートメント処理が停止し、すべてのインデックス変更がディスクにフラッシュされるまで、しばらくの間 mysqld サーバーがまだ使用できることに注意してください。

ステージ 1: テーブルのチェック

myisamchk *.MYI または時間があれば myisamchk -e *.MYI を実行します。-s (サイレント) オプションを使用すると、不要な情報を抑制します。

mysqld サーバーが停止している場合は、--update-state オプションを使用して、myisamchk にテーブルを「チェック済み」とマークするように指示してください。

myisamchk がエラーを報告しているテーブルだけを修復する必要があります。そのようなテーブルの場合、ステージ 2 に進みます。

チェック時に、予期しないエラー (out of memory エラーなど) を受け取った場合、または myisamchk がクラッシュした場合、ステージ 3 へ進みます。

ステージ 2: 簡単で安全な修復

まず myisamchk -r -q tbl_name を試します (-r -q は「クイックリカバリモード」を意味します)。これは、データファイルにアクセスせずに、インデックスファイルを修復しようとします。データファイルに、必要なすべてのものが含まれ、削除リンクがデータファイル内の正しい場所を指している場合、これは機能するはずであり、テーブルが修正されます。次のテーブルの修復を開始します。そうでない場合は、次の手順を使用します。

1. 続行する前に、データファイルのバックアップを作成します。
2. myisamchk -r tbl_name を使用します (-r は「リカバリモード」を意味します)。これによって、正しくない行と削除された行がデータファイルから削除され、インデックスファイルが再構築されます。
3. 先述のステップが失敗した場合、myisamchk --safe-recover tbl_name を使用します。安全なリカバリモードでは、通常のリカバリモードで扱われないわずかなケースを処理する古いリカバリ方法を使用します (ただし遅くなります)。

> **注記**
>
> 修復操作を大幅に高速化する場合、sort_buffer_size および key_buffer_size 変数の値をそれぞれ、myisamchk の実行時に使用可能なメモリーの約 25% に設定してください。

修復時に、予期しないエラー (out of memory エラーなど) を受け取った場合、または myisamchk がクラッシュした場合、ステージ 3 へ進みます。

ステージ 3: 困難な修復

このステージに到達するのは、インデックスファイル内の最初の 16K バイトのブロックが破損しているか、誤った情報が含まれている場合、またはインデックスファイルが失われている場合に限られるはずです。この場合、新しいインデックスファイルを作成する必要があります。次のように実行します。

1. データファイルを安全な場所に移動します。

2. テーブル記述ファイルを使用して、新しい (空の) データファイルとインデックスファイルを作成します。

```
shell> mysql db_name
mysql> SET autocommit=1;
mysql> TRUNCATE TABLE tbl_name;
mysql> quit
```

3. 古いデータファイルを新しく作成したデータファイルにコピーします。(古いファイルを新しいファイルに単に移動しないでください。何か異常があった場合に備えて、コピーを保持する必要があります。)

> **重要**
>
> レプリケーションを使用している場合、それにはファイルシステム操作が含まれ、これらは MySQL によって記録されないため、上記の手順を実行する前に、それを停止してください。

ステージ 2 に戻ります。myisamchk -r -q が機能するはずです。(これは無限ループにならないはずです。)

手順全体を自動的に実行する REPAIR TABLE tbl_name USE_FRM SQL ステートメントを使用することもできます。REPAIR TABLE を使用すると、サーバーがすべての作業を実行するため、ユーティリティーとサーバー間の不要なやり取りの可能性もなくなります。セクション13.7.2.5「REPAIR TABLE 構文」を参照してください。

ステージ 4: きわめて困難な修復

.frm 記述ファイルもクラッシュしている場合のみ、このステージに到達するはずです。記述ファイルはテーブルが作成されたあとに変更されないため、これが発生することはないはずです。

1. バックアップから記述ファイルをリストアし、ステージ 3 に戻ります。インデックスファイルをリストアし、ステージ 2 に戻ることもできます。後者の場合、myisamchk -r で起動してください。

2. バックアップがないが、テーブルの作成方法が正確にわかっている場合は、別のデータベースにテーブルのコピーを作成します。新しいデータファイルを削除して、ほかのデータベースから .frm 記述ファイルと .MYI インデックスファイルを、クラッシュしたデータベースへ移動します。これにより、新しい記述ファイルとインデックスファイルが得られますが、.MYD データファイルはそのまま残ります。ステージ 2 に戻り、インデックスファイルの再構築を試みます。

7.6.4 MyISAM テーブルの最適化

断片化した行を結合し、行の削除または更新の結果発生した無駄な領域を削除するには、myisamchk をリカバリモードで実行します。

```
shell> myisamchk -r tbl_name
```

OPTIMIZE TABLE SQL ステートメントを使用して、同様にテーブルを最適化することができます。OPTIMIZE TABLE はテーブルの修復とキー分析を行い、さらに、キーのルックアップが速くなるように、インデックスツリーをソートします。OPTIMIZE TABLE を使用すると、サーバーがすべての作業を実行するため、ユーティリティーとサーバー間の不要なやり取りの可能性もなくなります。セクション13.7.2.4「OPTIMIZE TABLE 構文」を参照してください。

myisamchk には、テーブルのパフォーマンスを向上させるために使用できる多数のその他オプションがあります。

- --analyze または -a: キー分布分析を実行します。これは、結合オプティマイザが、テーブルを結合する順番と、それが使用するインデックスをより適切に選択できるようにすることで、結合パフォーマンスを向上させます。
- --sort-index または -S: インデックスブロックをソートします。これはシークを最適化し、インデックスを使用するテーブルスキャンを高速化します。
- --sort-records=index_num または -R index_num: 特定のインデックスに従って、データ行をソートします。これにより、データが大幅に局所に集中化されるため、このインデックスを使用する、範囲に基づいた SELECT または ORDER BY 操作が高速化する可能性があります。

利用可能なすべてのオプションの完全な説明については、セクション4.6.3「myisamchk — MyISAM テーブルメンテナンスユーティリティー」を参照してください。

7.6.5 MyISAM テーブル保守スケジュールのセットアップ

問題が発生するのを待つより、テーブルチェックを定期的に実行することをお勧めします。MyISAM テーブルをチェックまたは修復する 1 つの方法は、CHECK TABLE および REPAIR TABLE ステートメントを使用することです。セクション13.7.2「テーブル保守ステートメント」を参照してください。

テーブルをチェックする別の方法は、myisamchk を使用することです。保守の目的には、myisamchk -s を使用できます。-s オプション (--silent の短縮形) により、サイレントモードで myisamchk が実行され、エラーが発生した場合のみ、メッセージが出力されます。

自動 MyISAM テーブルチェックを有効にすることをお勧めします。たとえば、マシンが更新の途中で再起動を実行した場合、通常、影響を受けた可能性のある各テーブルが使用される前に、それをチェックする必要があります。(これらは「予期されるクラッシュしたテーブル」です。) サーバーにMyISAM テーブルを自動的にチェックさせるには、それを --myisam-recover-options オプションを付けて起動します。セクション5.1.3「サーバーコマンドオプション」を参照してください。

通常のシステム操作時にも定期的にテーブルをチェックしてください。たとえば、crontab ファイル内の次のような行を使用して、週 1 回 cron ジョブを実行し、重要なテーブルをチェックします。

```
35 0 * * 0 /path/to/myisamchk --fast --silent /path/to/datadir/*/*.MYI
```

これはクラッシュしたテーブルに関する情報を出力するため、テーブルを調査し、必要に応じて修復できます。

はじめに、過去 24 時間中に更新されたすべてのテーブルに対して、毎晩 myisamchk -s を実行します。その問題がまれにしか発生しないことがわかったら、チェックの頻度を週 1 回などに減らすことができます。

通常、MySQL テーブルはほとんど保守が必要ありません。動的サイズの行のある MyISAM テーブル (VARCHAR、BLOB、または TEXT カラムのあるテーブル) に何回も更新を実行するか、または多くの削除された行のあるテーブルがある場合、ときどきテーブルの領域をデフラグ/再利用する必要がある場合があります。これは、問題のテーブルに OPTIMIZE TABLE を使用して実行できます。または、しばらくの間、mysqld サーバーを停止できる場合は、サーバーの停止中に、場所をデータディレクトリ内に変更し、次のコマンドを使用します。

```
shell> myisamchk -r -s --sort-index --myisam_sort_buffer_size=16M */*.MYI
```

第 8 章 最適化

目次

8.1 最適化の概要 774
8.2 SQL ステートメントの最適化 775
8.2.1 SELECT ステートメントの最適化 775
8.2.2 DML ステートメントの最適化 816
8.2.3 データベース権限の最適化 817
8.2.4 INFORMATION_SCHEMA クエリーの最適化 817
8.2.5 その他の最適化のヒント 822
8.3 最適化とインデックス 824
8.3.1 MySQL のインデックスの使用の仕組み 824
8.3.2 主キーの使用 825
8.3.3 外部キーの使用 825
8.3.4 カラムインデックス 826
8.3.5 マルチカラムインデックス 827
8.3.6 インデックスの使用の確認 828
8.3.7 InnoDB および MyISAM インデックス統計コレクション 828
8.3.8 B ツリーインデックスとハッシュインデックスの比較 829
8.4 データベース構造の最適化 830
8.4.1 データサイズの最適化 830
8.4.2 MySQL データ型の最適化 832
8.4.3 多数のテーブルの最適化 833
8.4.4 MySQL が内部一時テーブルを使用する仕組み 835
8.5 InnoDB テーブルの最適化 836
8.5.1 InnoDB テーブルのストレージレイアウトの最適化 836
8.5.2 InnoDB トランザクション管理の最適化 836
8.5.3 InnoDB ロギングの最適化 837
8.5.4 InnoDB テーブルの一括データロード 838
8.5.5 InnoDB クエリーの最適化 839
8.5.6 InnoDB DDL 操作の最適化 839
8.5.7 InnoDB ディスク I/O の最適化 839
8.5.8 InnoDB 構成変数の最適化 840
8.5.9 多くのテーブルのあるシステムに対する InnoDB の最適化 842
8.6 MyISAM テーブルの最適化 842
8.6.1 MyISAM クエリーの最適化 842
8.6.2 MyISAM テーブルの一括データロード 843
8.6.3 REPAIR TABLE ステートメントの速度 844
8.7 MEMORY テーブルの最適化 845
8.8 クエリー実行プランの理解 846
8.8.1 EXPLAIN によるクエリーの最適化 846
8.8.2 EXPLAIN 出力フォーマット 846
8.8.3 EXPLAIN EXTENDED 出力フォーマット 857
8.8.4 クエリーパフォーマンスの推定 858
8.8.5 クエリーオプティマイザの制御 859
8.9 バッファリングとキャッシュ 861
8.9.1 InnoDB バッファープール 861
8.9.2 MyISAM キーキャッシュ 864
8.9.3 MySQL クエリーキャッシュ 868
8.9.4 プリペアドステートメントおよびストアドプログラムのキャッシュ 873
8.10 ロック操作の最適化 874
8.10.1 内部ロック方法 874
8.10.2 テーブルロックの問題 876
8.10.3 同時挿入 877
8.10.4 メタデータのロック 878
8.10.5 外部ロック 879
8.11 MySQL サーバーの最適化 880
8.11.1 システム要素およびスタートアップパラメータのチューニング 880
8.11.2 サーバーパラメータのチューニング 880
8.11.3 ディスク I/O の最適化 887

8.11.4 メモリーの使用の最適化 891
8.11.5 ネットワークの使用の最適化 894
8.11.6 スレッドプールプラグイン 896
8.12 パフォーマンスの測定 (ベンチマーク) 901
8.12.1 式と関数の速度の測定 901
8.12.2 MySQL ベンチマークスイート 901
8.12.3 独自のベンチマークの使用 902
8.12.4 performance_schema によるパフォーマンスの測定 903
8.12.5 スレッド情報の検査 903

この章では、MySQL のパフォーマンスを最適化する方法について説明し、例を示します。最適化には、いくつかのレベルでの構成、チューニング、およびパフォーマンスの測定が含まれます。業務の役割 (開発者、データベース管理者、または両方の組み合わせ) に応じて、個々の SQL ステートメント、アプリケーション全体、単一のデータベースサーバー、または複数のネットワーク接続されたデータベースサーバーのレベルで最適化できます。プロアクティブにパフォーマンスを事前に計画する場合や、または問題の発生後に、構成やコードの問題のトラブルシューティングを行う場合があります。CPU やメモリーの使用を最適化することで、スケーラビリティーを向上し、データベースを低下させず、より多くの負荷を処理させることもできます。

8.1 最適化の概要

データベースのパフォーマンスは、テーブル、クエリー、構成設定など、データベースレベルの複数の要因に依存します。これらのソフトウェア構造は、ハードウェアレベルでの CPU および I/O 操作につながり、それらを最小限にし、可能なかぎり効率的にする必要があります。データベースのパフォーマンスを行う際は、ソフトウェア側の高レベルのルールとガイドラインについて学び、時計を使ってパフォーマンスを測定することから始めます。熟練するにつれ、内部で起こっていることについて詳しく学習し、CPU サイクルや I/O 操作などの測定を開始します。

一般的なユーザーの目標は、既存のソフトウェアやハードウェア構成から、最高のデータベースパフォーマンスを得ることです。上級ユーザーは、MySQL ソフトウェア自体を改善する機会を見つけたり、独自のストレージエンジンやハードウェアアプライアンスを開発して、MySQL エコシステムを拡張したりします。

データベースレベルでの最適化

データベースアプリケーションを高速にすることにおいてもっとも重要な要素は、その基本設計です。

- テーブルは適切に構築されていますか。特に、カラムに適切なデータ型があり、各テーブルに、作業の種類に適切なカラムがありますか。たとえば、頻繁な更新を実行するアプリケーションでは、多くの場合に少数のカラムのある多数のテーブルを使用し、大量のデータを解析するアプリケーションでは、多くの場合に多数のカラムのある少数のテーブルを使用します。
- クエリーを効率的にするため、適切なインデックスが設定されていますか。
- テーブルごとに適切なストレージエンジンを使用しており、使用している各ストレージエンジンの長所と機能を生かしていますか。特に、InnoDB などのトランザクションストレージエンジンまたは MyISAM などの非トランザクションストレージエンジンの選択は、パフォーマンスとスケーラビリティーにきわめて重要な場合があります。

  > **注記**
  >
  > MySQL 5.5 以上では、InnoDB は新しいテーブルのデフォルトのストレージエンジンです。実際に、高度な InnoDB パフォーマンス機能は、InnoDB テーブルが、特にビジーなデータベースに対して、多くの場合に単純な MyISAM テーブルよりパフォーマンスが優れていることを意味します。

- 各テーブルは適切な行フォーマットを使用していますか。この選択は、テーブルに使用されるストレージエンジンによっても異なります。特に、圧縮テーブルは使用するディスク領域が減るため、データの読み取りと書き込みに必要なディスク I/O も少なくなります。圧縮は、InnoDB テーブルでのあらゆる種類のワークロードと、読み取り専用 MyISAM テーブルに使用できます。
- アプリケーションでは、適切なロック戦略を使用していますか。たとえば、データベース操作を同時に実行できるように、可能なかぎり共有アクセスを許可したり、重要な操作が最優先されるように、適切な場合に排他的アクセスを要求したりするなどです。ここでも、ストレージエンジンの選択が重要です。InnoDB ストレージエンジンは、ユーザーが関与せずに、ほとんどのロックの問題を処理するため、データベースの同時実行性を向上し、コードの実験やチューニングの量を削減できます。

- キャッシュに使用されるメモリー領域がすべて正しくサイズ設定されていますか。つまり、頻繁にアクセスされるデータを保持するために十分な大きさがありながらも、物理メモリーをオーバーロードし、ページングを発生させるほど大きくしません。構成する主なメモリー領域は、InnoDB バッファープール、MyISAM キーキャッシュ、MySQL クエリーキャッシュです。

ハードウェアレベルでの最適化

データベースがビジーになるほど、どんなデータベースアプリケーションも最終的にハードウェアの制限に達します。データベース管理者は、アプリケーションをチューニングするか、サーバーを再構成してこれらのボトルネックを回避できるかどうか、または追加のハードウェアリソースが必要かどうかを評価する必要があります。システムボトルネックは一般に次の原因から発生します。

- ディスクシーク。ディスクがデータを検索するには時間がかかります。最新のディスクでは、通常この平均時間が 10 ms 未満であるため、理論的には 1 秒間に約 100 シーク実行できることになります。この時間は、新しいディスクでは徐々に改善されますが、1 つのテーブルに対して最適化することはきわめて困難です。シーク時間を最適化する方法は、複数のディスクにデータを分散することです。

- ディスクの読み取りと書き込み。ディスクが正しい位置にある場合に、データを読み取りまたは書き込みする必要があります。最新のディスクでは、1 つのディスクで少なくとも 10 - 20M バイト/秒のスループットを実現します。これは、複数のディスクから並列で読み取ることができるため、シークより最適化が簡単です。

- CPU サイクル。データがメインメモリー内にある場合、結果を得るために、それを処理する必要があります。メモリーの量と比較して大きなテーブルを使用することは、もっとも一般的な制限要因になります。しかし、小さいテーブルでは、通常速度は問題になりません。

- メモリー帯域幅。CPU で、CPU キャッシュに収められるより多くのデータを必要とする場合、メインメモリーの帯域幅がボトルネックになります。これは、ほとんどのシステムでまれなボトルネックですが、認識しておくべきです。

移植性とパフォーマンスのバランス

ポータブル MySQL プログラムで、パフォーマンス指向の SQL 拡張を使用するには、ステートメント内の MySQL 固有のキーワードを /*! */ コメント区切り文字で囲むことができます。ほかの SQL サーバーはコメントにされたキーワードを無視します。コメントの作成については、セクション9.6「コメントの構文」を参照してください。

8.2 SQL ステートメントの最適化

インタプリタから直接発行されるか、API によって内部で送信されるかに関係なく、データベースアプリケーションのコアロジックは SQL ステートメントによって実行されます。このセクションのチューニングのガイドラインは、あらゆる種類の MySQL アプリケーションの高速化に役立ちます。このガイドラインでは、データを読み取りおよび書き込みする SQL 操作、一般的な SQL 操作の内部オーバーヘッド、およびデータベースモニタリングなどの特定のシナリオで使われる操作について説明します。

8.2.1 SELECT ステートメントの最適化

SELECT ステートメントの形式のクエリーは、データベースのすべてのルックアップ操作を実行します。動的 Web ページの 1 秒未満の応答時間を達成するためでも、または巨大な夜間のレポートを生成するための時間から数時間を取り除くためでも、これらのステートメントのチューニングは最優先です。

8.2.1.1 SELECT ステートメントの速度

クエリーの最適化の主な考慮事項は次のとおりです。

- 遅い SELECT ... WHERE クエリーを高速化するため、最初に確認することは、インデックスを追加できるかどうかです。WHERE 句で使用するカラムにインデックスをセットアップし、評価、フィルタリング、および最終的な結果の取得を高速化します。無駄なディスク領域を避けるため、アプリケーションで使用される多くの関連クエリーを高速化する少数のインデックスのセットを構築します。

  インデックスは、結合や外部キーなどの機能を使用して、さまざまなテーブルを参照するクエリーに特に重要です。EXPLAIN ステートメントを使用して、SELECT に使用するインデックスを判断できます。セクション8.3.1「MySQL のインデックスの使用の仕組み」およびセクション8.8.1「EXPLAIN によるクエリーの最適化」を参照してください。

- 過度な時間がかかる関数呼び出しなどのクエリーの部分を特定し、チューニングします。クエリーの構築の仕方によっては、関数が結果セットのすべての行に対して 1 回ずつ、さらにはテーブル内のすべての行に対して 1 回ずつ呼び出されるなど、大幅に非効率性を拡大させていることがあります。
- 特に大きなテーブルの場合に、クエリーでの完全テーブルスキャンの回数を最小にします。
- ANALYZE TABLE ステートメントを定期的に使用して、テーブル統計を最新に維持し、オプティマイザが、効率的な実行プランを立てるために必要な情報が得られるようにします。
- チューニング技法、インデックス作成技法、および各テーブルのストレージエンジンに固有の構成パラメータについて学習します。InnoDB と MyISAM のどちらでも、クエリーの高いパフォーマンスを可能にし、維持するための一連のガイドラインがあります。詳細については、セクション8.5.5「InnoDB クエリーの最適化」およびセクション8.6.1「MyISAM クエリーの最適化」を参照してください。
- 特に、MySQL 5.6.4 以上では、セクション14.13.14「InnoDB の読み取り専用トランザクションの最適化」の技法を使用して、InnoDB テーブルの単一クエリートランザクションを最適化できます。
- 特にオプティマイザで同じ変換の一部を自動的に実行する場合、理解が困難になるようなクエリーの変換を避けます。
- いずれかの基本ガイドラインによって、パフォーマンスの問題が簡単に解決されない場合、EXPLAIN プランを読み、インデックス、WHERE 句、結合句などを調整して、特定のクエリーの内部の詳細を調査します。(ある程度の専門技術に達している場合は、EXPLAIN プランを読むことがすべてのクエリーの最初の手順になると考えられます。)
- MySQL がキャッシュに使用するメモリー領域のサイズとプロパティーを調整します。InnoDB バッファープール、MyISAM キーキャッシュ、および MySQL クエリーキャッシュの効率的な使用によって、2 回目以降、メモリーから結果が取得されるため、繰り返しのクエリーの実行が高速化します。
- キャッシュメモリー領域を使用して高速に実行するクエリーでも、必要なキャッシュメモリーを減らして、アプリケーションがよりスケーラブルになるように、さらに最適化できます。スケーラビリティーは、パフォーマンスを大幅に低下させずに、アプリケーションでより多くの同時ユーザー、大きなリクエストなどを処理できることを意味します。
- クエリーの速度が、テーブルに同時にアクセスしているほかのセッションによって影響を受ける可能性があるロックの問題を処理します。

8.2.1.2 MySQL の WHERE 句の最適化の方法

このセクションでは、WHERE 句の処理で実行可能な最適化について説明します。例では SELECT ステートメントを使用していますが、DELETE および UPDATE ステートメント内の WHERE 句にも同じ最適化を適用します。

> **注記**
>
> MySQL オプティマイザへの取り組みは継続中であるため、MySQL が実行する最適化のすべてをここで説明しているわけではありません。

読みやすさを犠牲にしても、算術演算を高速化するように、クエリーを書き換えたいと考えがちです。MySQL では同様の最適化を自動的に実行するため、多くの場合にこの作業を回避でき、クエリーを理解しやすく、保守しやすい形式のままにしておくことができます。MySQL によって実行される最適化の一部を次に示します。

- 不要なかっこの削除:

```
  ((a AND b) AND c OR (((a AND b) AND (c AND d))))
-> (a AND b AND c) OR (a AND b AND c AND d)
```

- 定数畳み込み:

```
  (a<b AND b=c) AND a=5
-> b>5 AND b=c AND a=5
```

- 定数条件の削除 (定数畳み込みのために必要です):

```
  (B>=5 AND B=5) OR (B=6 AND 5=5) OR (B=7 AND 5=6)
-> B=5 OR B=6
```

- インデックスによって使用される定数式は 1 回だけ評価されます。
- WHERE を使用しない単一テーブルの COUNT(*) は、MyISAM テーブルと MEMORY テーブルのテーブル情報から直接取得されます。これは、1 つだけのテーブルで使用された場合に、NOT NULL 式にも実行されます。

- 無効な定数式の早期の検出。MySQL は一部の SELECT ステートメントが実行不可能であることをすみやかに検出し、行を返しません。
- GROUP BY または集約関数 (COUNT()、MIN() など) を使用しない場合、HAVING は WHERE とマージされます。
- 結合内の各テーブルについて、テーブルの高速の WHERE 評価を取得し、可能なかぎり早く行をスキップするために、より単純な WHERE が構築されます。
- クエリー内のほかのすべてのテーブルの前に、まず、すべての定数テーブルが読み取られます。定数テーブルは次のいずれかです。
  - 空白のテーブルまたは 1 行のテーブル。
  - PRIMARY KEY または UNIQUE インデックスでの WHERE 句で使用されるテーブル。ここではすべてのインデックス部分が定数式と比較され、NOT NULL として定義されます。

  次のテーブルはすべて定数テーブルとして使用されます。

```
SELECT * FROM t WHERE primary_key=1;
SELECT * FROM t1,t2
  WHERE t1.primary_key=1 AND t2.primary_key=t1.id;
```

- テーブルを結合するための最適な結合の組み合わせは、すべての可能性を試してみることで見つかります。ORDER BY および GROUP BY 句内のすべてのカラムが同じテーブルにある場合、結合する際に最初にそのテーブルが選ばれます。
- ORDER BY 句と別の GROUP BY 句がある場合、または、ORDER BY または GROUP BY に結合キュー内の最初のテーブルと異なるテーブルのカラムが含まれている場合は、一時テーブルが作成されます。
- SQL_SMALL_RESULT オプションを使用すると、MySQL ではインメモリー一時テーブルが使用されます。
- オプティマイザがテーブルスキャンを使用する方が効率的であると判断しないかぎり、各テーブルインデックスがクエリーされ、最適なインデックスが使用されます。かつて、スキャンは、最適なインデックスがテーブルの 30% 超にまたがっているかどうかに基づいて使用されていましたが、固定のパーセンテージによって、インデックスを使用するか、スキャンを使用するかの選択が決定されなくなりました。現在のオプティマイザは複雑になり、テーブルサイズ、行数、I/O ブロックサイズなどの追加の要因に基づいて推定します。
- 場合によって、MySQL はデータファイルを参照しなくてもインデックスから行を読み取ることができます。インデックスから使用されるすべてのカラムが数値の場合、クエリーの解決にインデックスツリーのみが使用されます。
- 各行が出力される前に、HAVING 句に一致しないものはスキップされます。

きわめて高速なクエリーのいくつかの例:

```
SELECT COUNT(*) FROM tbl_name;

SELECT MIN(key_part1),MAX(key_part1) FROM tbl_name;

SELECT MAX(key_part2) FROM tbl_name
  WHERE key_part1=constant;

SELECT ... FROM tbl_name
  ORDER BY key_part1,key_part2,... LIMIT 10;

SELECT ... FROM tbl_name
  ORDER BY key_part1 DESC, key_part2 DESC, ... LIMIT 10;
```

MySQL は、インデックス設定されたカラムが数値であるとして、インデックスツリーのみを使用して、次のクエリーを解決します。

```
SELECT key_part1,key_part2 FROM tbl_name WHERE key_part1=val;

SELECT COUNT(*) FROM tbl_name
  WHERE key_part1=val1 AND key_part2=val2;

SELECT key_part2 FROM tbl_name GROUP BY key_part1;
```

次のクエリーは、個別のソーティングパスを使用せずに、インデックスを使用して、ソート順で行を取得します。

```
SELECT ... FROM tbl_name
```

```
 ORDER BY key_part1,key_part2,... ;

SELECT ... FROM tbl_name
 ORDER BY key_part1 DESC, key_part2 DESC, ... ;
```

8.2.1.3 range の最適化

range アクセスメソッドは単一のインデックスを使用して、1 つまたは複数のインデックス値間隔の中に含まれるテーブル行のサブセットを取得します。これは、シングルパートまたはマルチパートインデックスに使用できます。次のセクションでは、WHERE 句から間隔を抽出する方法について詳しく説明します。

シングルパートインデックスの range アクセスメソッド

シングルパートインデックスでは、インデックス値間隔は WHERE 句内の対応する条件によって便利に表すことができるため、「間隔」よりも範囲条件について説明します。

シングルパートインデックスの範囲条件の定義は次のとおりです。

- BTREE と HASH の両方のインデックスで、=、<=>、IN()、IS NULL、または IS NOT NULL 演算子を使用した場合、キーパートと定数値の比較は範囲条件です。
- さらに、BTREE インデックスでは、>、<、>=、<=、BETWEEN、!=、または <> 演算子、または LIKE への引数が、ワイルドカード文字で始まっていない定数文字列である場合の LIKE 比較を使用した場合に、キーパートと定数値の比較は範囲条件です。
- すべての種類のインデックスで、OR または AND で組み合わされた複数の範囲条件は、1 つの範囲条件を形成します。

先述の「定数値」とは次のいずれかを意味します。

- クエリー文字列からの定数
- 同じ結合からの const または system テーブルのカラム
- 非相関サブクエリーの結果
- 以前の型の部分式からのみ構成された式

以下に WHERE 句内で範囲条件を使用したクエリーのいくつかの例を示します。

```
SELECT * FROM t1
 WHERE key_col > 1
 AND key_col < 10;

SELECT * FROM t1
 WHERE key_col = 1
 OR key_col IN (15,18,20);

SELECT * FROM t1
 WHERE key_col LIKE 'ab%'
 OR key_col BETWEEN 'bar' AND 'foo';
```

定数伝播フェーズ中に、一部の非定数値が定数に変換されることがあります。

MySQL は可能なインデックスごとに、WHERE 句から範囲条件を抽出しようとします。抽出プロセス時に、範囲条件の構築に使用できない条件はドロップされ、重複する範囲を生成する条件は組み合わされて、空の範囲を生成する条件は削除されます。

key1 がインデックス設定されたカラムで nonkey がインデックス設定されていない、次のステートメントを考慮します。

```
SELECT * FROM t1 WHERE
 (key1 < 'abc' AND (key1 LIKE 'abcde%' OR key1 LIKE '%b')) OR
 (key1 < 'bar' AND nonkey = 4) OR
 (key1 < 'uux' AND key1 > 'z');
```

キー key1 の抽出プロセスは次のとおりです。

1. 元の WHERE 句から始めます。

   ```
   (key1 < 'abc' AND (key1 LIKE 'abcde%' OR key1 LIKE '%b')) OR
   ```

```
(key1 < 'bar' AND nonkey = 4) OR
(key1 < 'uux' AND key1 > 'z')
```

2. nonkey = 4 と key1 LIKE '%b' は、範囲スキャンに使用できないため、削除します。それらを削除する正しい方法は、範囲スキャンの実行時に一致する行を見落とさないように、それらを TRUE で置き換えることです。TRUE で置き換えると、次のようになります。

```
(key1 < 'abc' AND (key1 LIKE 'abcde%' OR TRUE)) OR
(key1 < 'bar' AND TRUE) OR
(key1 < 'uux' AND key1 > 'z')
```

3. 常に true または false である条件を縮小します。

   - (key1 LIKE 'abcde%' OR TRUE) は常に true です
   - (key1 < 'uux' AND key1 > 'z') は常に false です

   これらの条件を定数で置き換えると、次のようになります。

```
(key1 < 'abc' AND TRUE) OR (key1 < 'bar' AND TRUE) OR (FALSE)
```

   不要な TRUE および FALSE 定数を削除すると、次のようになります。

```
(key1 < 'abc') OR (key1 < 'bar')
```

4. 重複する間隔を 1 つに組み合わせて、範囲スキャンに使用される最終的な条件が生成されます。

```
(key1 < 'bar')
```

一般に (前の例で示したように)、範囲スキャンに使用される条件は、WHERE 句より制限がゆるくなります。MySQL は、範囲条件を満たすが、完全な WHERE 句でない行をフィルタ処理する追加のチェックを実行します。

範囲条件抽出アルゴリズムは、任意の深さのネストの AND/OR 構造を処理でき、その出力は WHERE 句内の条件が存在する順番に依存しません。

現在、MySQL では、空間インデックスに対して、range アクセスメソッドの複数の範囲のマージをサポートしていません。この制限を回避するには、同じ SELECT ステートメントで UNION を使用できますが、ただし、各空間述語は、別の SELECT に入れます。

マルチパートインデックスの range アクセスメソッド

マルチパートインデックスの範囲条件は、シングルパートインデックスの範囲条件の拡張です。マルチパートインデックスの範囲条件は、インデックス行を 1 つまたは複数のキータプル間隔内に入るように制限します。キータプル間隔は、インデックスからの順序付けを使用して、キータプルのセットに定義されます。

たとえば、key1(key_part1、key_part2、key_part3) として定義されたマルチパートインデックスと、キー順で示された次のキータプルのセットを考慮します。

```
key_part1  key_part2  key_part3
 NULL      1         'abc'
 NULL      1         'xyz'
 NULL      2         'foo'
  1        1         'abc'
  1        1         'xyz'
  1        2         'abc'
  2        1         'aaa'
```

条件 key_part1 = 1 は次の間隔を定義します。

```
(1,-inf,-inf) <= (key_part1,key_part2,key_part3) < (1,+inf,+inf)
```

間隔は前のデータセットの 4、5、6 番目のタプルをカバーし、range アクセスメソッドで使用できます。

対照的に、条件 key_part3 = 'abc' は単一の間隔を定義せず、range アクセスメソッドで使用できません。

次の説明では、マルチパートインデックスに対して、範囲条件がどのように作用するかを詳しく示します。

- HASH インデックスでは、同一の値を含む各間隔を使用できます。これは次の形式の条件に対してのみ、間隔を生成できることを意味します。

```
   key_part1 cmp const1
AND key_part2 cmp const2
AND ...
AND key_partN cmp constN;
```

ここで、const1、const2、... は定数で、cmp は、=、<=>、または IS NULL 比較演算子のいずれかで、条件はすべてのインデックスパートをカバーします。(つまり、N パートインデックスの各パートに 1 つずつ N 条件があります。)たとえば、次は 3 パート HASH インデックスの範囲条件です。

```
key_part1 = 1 AND key_part2 IS NULL AND key_part3 = 'foo'
```

何を定数とみなすかの定義については、「シングルパートインデックスの range アクセスメソッド」を参照してください。

- BTREE インデックスでは、各条件で =、<=>、IS NULL、>、<、>=、<=、!=、<>、BETWEEN、または LIKE 'pattern' (ここで 'pattern' はワイルドカードで始まらない) を使用して、キーパートと定数値を比較する、AND で組み合わされた条件に、間隔を使用できます。条件に一致するすべての行を含む単一のキータプルを判断できる場合にかぎり、1 つの間隔を使用できます (または <> または != を使用する場合は 2 つの間隔)。

  オプティマイザは、比較演算子が =、<=>、または IS NULL である場合にかぎり、追加のキーパートを使用して、間隔を判断しようとします。演算子が >、<、>=、<=、!=、<>、BETWEEN、または LIKE の場合、オプティマイザはそれを使用しますが、追加のキーパートは考慮しません。次の式では、オプティマイザは最初の比較からの = を使用します。さらに 2 番目の比較からの >= も使用しますが、それ以上のキーパートを考慮せず、間隔の構築に 3 番目の比較を使用しません。

  ```
  key_part1 = 'foo' AND key_part2 >= 10 AND key_part3 > 10
  ```

  単一の間隔は次のとおりです。

  ```
  ('foo',10,-inf) < (key_part1,key_part2,key_part3) < ('foo',+inf,+inf)
  ```

  作成された間隔に初期条件よりも多い行が含まれる可能性があります。たとえば、前の間隔は値 ('foo', 11, 0) を含みますが、これは元の条件を満たしません。

- 間隔内に含まれる行セットをカバーする条件が OR で組み合わされている場合、それらは、それらの間隔の和集合内に含まれる行セットをカバーする条件を形成します。条件が AND で組み合わされている場合、それらは間隔の共通集合内に含まれる行セットを対象とする条件を形成します。たとえば、2 パートインデックスでのこの条件の場合:

  ```
  (key_part1 = 1 AND key_part2 < 2) OR (key_part1 > 5)
  ```

  間隔は次のとおりです。

  ```
  (1,-inf) < (key_part1,key_part2) < (1,2)
  (5,-inf) < (key_part1,key_part2)
  ```

  この例で、1 行目の間隔は、左境界に 1 つのキーパートを使用し、右境界に 2 つのキーパートを使用しています。2 行目の間隔は 1 つのキーパートのみを使用しています。EXPLAIN 出力の key_len カラムは、使用されたキープリフィクスの最大長を示しています。

  場合によって、key_len はキーパートが使用されたことを示しますが、それが予期したものではないことがあります。key_part1 と key_part2 が NULL になることがあるとします。次に、key_len カラムに、次の条件の 2 つのキーパート長が表示されます。

  ```
  key_part1 >= 1 AND key_part2 < 2
  ```

  しかし、実際は条件が次に変換されます。

  ```
  key_part1 >= 1 AND key_part2 IS NOT NULL
  ```

「シングルパートインデックスの range アクセスメソッド」では、単一パートインデックスで、範囲条件の間隔を組み合わせたり、削除したりするために、どのように最適化が実行されるかを説明しています。マルチパートインデックスでの範囲条件にも類似の手順が実行されます。

複数値比較の等価範囲の最適化

col_name がインデックス設定されたカラムである次の式を考慮します。

```
col_name IN(val1, ..., valN)
col_name = val1 OR ... OR col_name = valN
```

col_name が複数の値のいずれかと等しい場合に、各式は true になります。これらの比較は等価範囲比較です (ここで「範囲」は単一の値です)。オプティマイザは、次のように等価範囲比較の対象とする行の読み取りのコストを推定します。

- col_name に一意のインデックスがある場合、指定した値を持つことができる行は多くても 1 つであるため、各範囲の行の見積もりは 1 です。
- そうでない場合は、オプティマイザは、インデックスのダイブまたはインデックス統計を使用して、各範囲の行数を推定できます。

インデックスダイブでは、オプティマイザは範囲の両端でダイブを作成し、範囲内の行数を見積もりとして使用します。たとえば、式 col_name IN (10, 20, 30) には 3 つの等価範囲があり、オプティマイザは範囲あたり 2 つのダイブを作成して、行の見積もりを生成します。ダイブのペアごとに、指定した値を持つ行数の見積もりを生成します。

インデックスダイブは、正確な行見積もりを提供しますが、式内の比較値の数が増えるほど、オプティマイザの行見積もりの生成に時間がかかるようになります。インデックス統計の使用は、インデックスダイブより正確ではありませんが、大きな値リストの場合に、行見積もりが高速になります。

eq_range_index_dive_limit システム変数を使用して、オプティマイザが行の見積もり戦略を別の戦略に切り替える値の数を構成できます。統計の使用を無効にして、常にインデックスダイブを使用するには、eq_range_index_dive_limit を 0 に設定します。最大 N 個の等価範囲の比較にインデックスダイブの使用を許可するには、eq_range_index_dive_limit を N + 1 に設定します。

eq_range_index_dive_limit は MySQL 5.6.5 以降で使用できます。5.6.5 より前では、オプティマイザは eq_range_index_dive_limit=0 と同等のインデックスダイブを使用します。

最適な推定を行うためにテーブルインデックス統計を更新するには、ANALYZE TABLE を使用します。

8.2.1.4 インデックスマージの最適化

インデックスマージメソッドは、複数の range スキャンによって、行を取得しそれらの結果を 1 つにマージするために使用されます。このマージによって、その基盤となるスキャンの和集合、共通集合、または共通集合の和集合を生成できます。このアクセスメソッドは、1 つのテーブルからのインデックススキャンをマージします。複数のテーブルにわたるスキャンはマージしません。

EXPLAIN 出力では、インデックスマージメソッドは type カラムに index_merge と表示されます。この場合、key カラムには使用されたインデックスのリストが含まれ、key_len にはそれらのインデックスの最長のキーパートのリストが含まれます。

例:

```
SELECT * FROM tbl_name WHERE key1 = 10 OR key2 = 20;

SELECT * FROM tbl_name
  WHERE (key1 = 10 OR key2 = 20) AND non_key=30;

SELECT * FROM t1, t2
  WHERE (t1.key1 IN (1,2) OR t1.key2 LIKE 'value%')
  AND t2.key1=t1.some_col;

SELECT * FROM t1, t2
  WHERE t1.key1=1
  AND (t2.key1=t1.some_col OR t2.key2=t1.some_col2);
```

インデックスマージメソッドにはいくつかのアクセスアルゴリズムがあります (EXPLAIN 出力の Extra フィールドで確認されます)。

- Using intersect(...)
- Using union(...)
- Using sort_union(...)

次のセクションでは、これらのメソッドについて詳しく説明します。

> **注記**
>
> インデックスマージ最適化アルゴリズムには次の既知の不具合があります。

- クエリーに AND/OR の深いネストのある複雑な WHERE 句があり、MySQL が最適なプランを選択しない場合、次の同一律を使用して、項を分配してみてください。

```
(x AND y) OR z = (x OR z) AND (y OR z)
(x OR y) AND z = (x AND z) OR (y AND z)
```

- インデックスマージは全文インデックスには適用できません。将来の MySQL リリースでこれらを扱うように、それを拡張する予定です。
- MySQL 5.6.6 より前では、一部のキーに対して範囲スキャンが使用可能な場合、オプティマイザはインデックスマージ和集合またはインデックスマージソート和集合アルゴリズムを使用することを考慮しません。たとえば、次のクエリーを考慮します。

```
SELECT * FROM t1 WHERE (goodkey1 < 10 OR goodkey2 < 20) AND badkey < 30;
```

このクエリーでは、2 つのプランが使用可能です。

- (goodkey1 < 10 OR goodkey2 < 20) 条件を使用したインデックスマージスキャン。
- badkey < 30 条件を使用した範囲スキャン。

ただし、オプティマイザは 2 つめのプランしか考慮しません。

インデックスマージアクセスメソッドの可能性のあるさまざまなバリアントとその他のアクセスメソッドとの選択は、使用可能な各種オプションのコスト見積もりに基づきます。

インデックスマージ共通集合アクセスアルゴリズム

このアクセスアルゴリズムは、WHERE 句が、AND で結合されたさまざまなキーに対する複数の範囲条件に変換され、各条件が次のいずれかである場合に採用できます。

- この形式では、インデックスには正確に N 個のパートがあります (つまり、すべてのインデックスパートがカバーされます)。

```
key_part1=const1 AND key_part2=const2 ... AND key_partN=constN
```

- InnoDB テーブルの主キーに対する範囲条件。

例:

```
SELECT * FROM innodb_table WHERE primary_key < 10 AND key_col1=20;

SELECT * FROM tbl_name
  WHERE (key1_part1=1 AND key1_part2=2) AND key2=2;
```

インデックスマージ共通集合アルゴリズムは、使用されたすべてのインデックスの同時スキャンを実行し、マージされたインデックススキャンから受け取る行シーケンスの共通集合を生成します。

クエリーに使用されているすべてのカラムが、使用されるインデックスによってカバーされている場合、完全なテーブル行は取得されません (この場合、EXPLAIN 出力の Extra フィールドに Using index が含まれます)。次はそのようなクエリーの例です。

```
SELECT COUNT(*) FROM t1 WHERE key1=1 AND key2=1;
```

使用されるインデックスで、クエリーに使用されているすべてのカラムがカバーされない場合、使用されているすべてのキーの範囲条件が満たされている場合にのみ、完全な行が取得されます。

マージされた条件のいずれかが InnoDB テーブルの主キーに対する条件である場合、それは行の取得には使用されませんが、ほかの条件を使用して取得された行をフィルタ処理するために使用されます。

インデックスマージ和集合アクセスアルゴリズム

このアルゴリズムの適用基準はインデックスマージメソッド共通集合アルゴリズムの場合と似ています。このアルゴリズムは、テーブルの WHERE 句が、OR で組み合わされたさまざまなキーに対する複数の範囲条件に変換されており、各条件が次のいずれかである場合に採用できます。

- この形式では、インデックスには正確に N 個のパートがあります (つまり、すべてのインデックスパートがカバーされます)。

```
key_part1=const1 AND key_part2=const2 ... AND key_partN=constN
```

- InnoDB テーブルの主キーに対する範囲条件。
- インデックスマージメソッド共通集合アルゴリズムを適用できる条件。

例:

```
SELECT * FROM t1 WHERE key1=1 OR key2=2 OR key3=3;

SELECT * FROM innodb_table WHERE (key1=1 AND key2=2) OR
  (key3='foo' AND key4='bar') AND key5=5;
```

インデックスマージソート和集合アクセスアルゴリズム

このアクセスアルゴリズムは、WHERE 句が、OR で組み合わされた複数の範囲条件に変換されているが、インデックスマージメソッド和集合アルゴリズムを適用できない場合に採用されます。

例:

```
SELECT * FROM tbl_name WHERE key_col1 < 10 OR key_col2 < 20;

SELECT * FROM tbl_name
  WHERE (key_col1 > 10 OR key_col2 = 20) AND nonkey_col=30;
```

ソート和集合アルゴリズムと和集合アルゴリズムの違いは、ソート和集合アルゴリズムでは、行を返す前にまずすべての行の行 ID をフェッチし、それらをソートする必要があることです。

8.2.1.5 エンジンコンディションプッシュダウンの最適化

この最適化は、インデックスが設定されていないカラムと定数との直接比較の効率性を向上します。このような場合、条件が評価のためにストレージエンジンに「プッシュダウン」されます。この最適化は、NDB ストレージエンジンでのみ使用できます。

MySQL Cluster では、この最適化によって、クラスタのデータノードとクエリーを発行した MySQL Server 間で、ネットワーク経由で一致しない行を送る必要性をなくすことができ、それを使用した場合のクエリーを、コンディションプッシュダウンが可能であっても使用しない場合より、5 - 10 倍高速化できます。

MySQL Cluster テーブルが次のように定義されているとします。

```
CREATE TABLE t1 (
    a INT,
    b INT,
    KEY(a)
) ENGINE=NDB;
```

コンディションプッシュダウンは、インデックスが設定されていないカラムと定数との比較を含む、ここに示すようなクエリーで使用できます。

```
SELECT a, b FROM t1 WHERE b = 10;
```

コンディションプッシュダウンの使用は、EXPLAIN の出力で確認できます。

```
mysql> EXPLAIN SELECT a,b FROM t1 WHERE b = 10\G
*************************** 1. row ***************************
           id: 1
  select_type: SIMPLE
        table: t1
         type: ALL
possible_keys: NULL
          key: NULL
      key_len: NULL
          ref: NULL
         rows: 10
        Extra: Using where with pushed condition
```

ただし、コンディションプッシュダウンは、これらの 2 つのクエリーのいずれかと一緒に使用することはできません。

```
SELECT a,b FROM t1 WHERE a = 10;
SELECT a,b FROM t1 WHERE b + 1 = 10;
```

カラム a にインデックスが存在するため、コンディションプッシュダウンは最初のクエリーには適用できません。(インデックスアクセスメソッドの方が効率的であるため、コンディションプッシュダウンよりも優先して選択されます。) インデックスが設定されていないカラム b を含む比較は間接的であるため、2 つめのクエリーにコンディションプッシュダウンを採用することはできません。(ただし、WHERE 句内で b + 1 = 10 を b = 9 にまとめる場合はコンディションプッシュダウンを適用できます。)

> または < 演算子を使用して、インデックス設定されたカラムを定数と比較する場合にもコンディションプッシュダウンを採用できます。

```
mysql> EXPLAIN SELECT a, b FROM t1 WHERE a < 2\G
*************************** 1. row ***************************
           id: 1
  select_type: SIMPLE
        table: t1
         type: range
possible_keys: a
          key: a
      key_len: 5
          ref: NULL
         rows: 2
        Extra: Using where with pushed condition
```

コンディションプッシュダウンでサポートされるその他の比較には、次が含まれます。

- column [NOT] LIKE pattern

  pattern は、照合するパターンを含む文字列リテラルである必要があります。構文については、セクション 12.5.1「文字列比較関数」を参照してください。

- column IS [NOT] NULL

- column IN (value_list)

  value_list の各項目は定数のリテラル値である必要があります。

- column BETWEEN constant1 AND constant2

  constant1 と constant2 はそれぞれ、定数のリテラル値である必要があります。

前のリストのすべての場合で、条件をカラムと定数との 1 つ以上の直接比較の形式に変換できます。

エンジンコンディションプッシュダウンはデフォルトで有効です。サーバーの起動時にそれを無効にするには、optimizer_switch システム変数を設定します。たとえば、my.cnf ファイルで、次の行を使用します。

```
[mysqld]
optimizer_switch=engine_condition_pushdown=off
```

実行時に、次のようにコンディションプッシュダウンを有効にします。

```
SET optimizer_switch='engine_condition_pushdown=off';
```

制限　エンジンコンディションプッシュダウンには次の制限があります。

- コンディションプッシュダウンは、NDB ストレージエンジンによってのみサポートされます。
- カラムは定数とのみ比較できますが、これには、定数値に評価される式も含まれます。
- 比較に使用されるカラムは、BLOB 型または TEXT 型のいずれかであってはいけません。
- カラムと比較される文字列値は、カラムと同じ照合順序を使用する必要があります。
- 結合は直接サポートされていません。複数のテーブルを含む条件は、可能な場合に個別にプッシュされます。実際にプッシュダウンされる条件を判断するには、EXPLAIN EXTENDED を使用します。

8.2.1.6 インデックスコンディションプッシュダウンの最適化

インデックスコンディションプッシュダウン (ICP) は、MySQL がインデックスを使用してテーブルから行を取得する場合の最適化です。ICP を使用しない場合、ストレージエンジンはインデックスをトラバースして、ベーステーブル内で行を検索し、MySQL Server に返し、MySQL Server が行に対して WHERE 条件を評価します。ICP を有効にすると、インデックスからのフィールドだけを使用して WHERE 条件の部分を評価できる場合

は、MySQL Server はこの WHERE 条件の部分をストレージエンジンにプッシュダウンします。ストレージエンジンは、インデックスエントリを使用して、プッシュされたインデックス条件を評価し、これが満たされている場合にのみ、テーブルから行を読み取ります。ICP は、ストレージエンジンがベーステーブルにアクセスする必要がある回数と、MySQL サーバーがストレージエンジンにアクセスする必要がある回数を削減できます。

インデックスコンディションプッシュダウン最適化は、完全なテーブル行にアクセスする必要がある場合に、range、ref、eq_ref、および ref_or_null アクセスメソッドで使用されます。この戦略は、InnoDB テーブルと MyISAM テーブルに使用できます。(インデックスコンディションプッシュダウンは、MySQL 5.6 ではパーティション化されたテーブルでサポートされていません。この問題は MySQL 5.7 で解決されています。)ただし、InnoDB テーブルの場合、ICP はセカンダリインデックスにのみ使用されます。ICP の目標は、完全なレコードの読み取りの回数を減らし、それによって IO 操作を減らすことです。InnoDB のクラスタ化されたインデックスの場合、完全なレコードはすでに InnoDB バッファーに読み込まれています。この場合に ICP を使用しても IO は削減されません。

この最適化の仕組みを確認するには、まずインデックスコンディションプッシュダウンが使用されない場合に、インデックススキャンがどのように進められるかを考察します。

1. まず、インデックスタプルを読み取り、次にそのインデックスタプルを使用して、完全なテーブル行を見つけて読み取ることで、次の行を取得します。
2. このテーブルに適用される WHERE 条件の部分をテストします。テスト結果に基づいて行を受け入れるか、拒否します。

インデックスコンディションプッシュダウンが使用される場合、代わりにスキャンは次のように進められます。

1. 次の行のインデックスタプルを取得します (ただし完全なテーブル行ではありません)。
2. このテーブルに適用され、インデックスカラムのみを使用してチェックできる WHERE 条件の部分をテストします。条件が満たされている場合、次の行のインデックスタプルに進みます。
3. 条件が満たされている場合、インデックスタプルを使用して、完全なテーブル行を見つけて読み取ります。
4. このテーブルに適用される WHERE 条件の残りの部分をテストします。テスト結果に基づいて行を受け入れるか、拒否します。

インデックスコンディションプッシュダウンが使用されると、EXPLAIN 出力の Extra カラムに Using index condition と表示されます。完全なテーブル行を読み取る必要がある場合に適用されないため、Index only は表示されません。

人とその住所に関する情報を格納するテーブルがあり、そのテーブルに、INDEX (zipcode, lastname, firstname) と定義されたインデックスがあるとします。ある個人の zipcode 値を知っているが、名前が確かでない場合に、次のように検索できます。

```
SELECT * FROM people
  WHERE zipcode='95054'
  AND lastname LIKE '%etrunia%'
  AND address LIKE '%Main Street%';
```

MySQL はインデックスを使用して、zipcode='95054' を持つ人をスキャンします。2 番目の部分 (lastname LIKE '%etrunia%') は、スキャンする必要がある行数を制限するために使用できないため、インデックスコンディションプッシュダウンを使用しない場合に、このクエリーでは zipcode='95054' を持つすべての人の完全なテーブル行を取得する必要があります。

インデックスコンディションプッシュダウンを使用すると、MySQL は完全なテーブル行を読み取る前に、lastname LIKE '%etrunia%' 部分をチェックします。これにより、lastname 条件に一致しないすべてのインデックスタプルに対応する完全な行の読み取りが避けられます。

インデックスコンディションプッシュダウンはデフォルトで有効です。これは、optimizer_switch システム変数で index_condition_pushdown フラグを設定することで制御できます。セクション8.8.5.2「切り替え可能な最適化の制御」を参照してください。

8.2.1.7 インデックス拡張の使用

InnoDB は、自動的に各セカンダリインデックスに主キーカラムを追加して、それを拡張します。このテーブル定義について考えます。

```
CREATE TABLE t1 (
```

```
 i1 INT NOT NULL DEFAULT 0,
 i2 INT NOT NULL DEFAULT 0,
 d DATE DEFAULT NULL,
 PRIMARY KEY (i1, i2),
 INDEX k_d (d)
) ENGINE = InnoDB;
```

このテーブルでは、カラム (i1, i2) に主キーを定義しています。さらに、カラム (d) にセカンダリインデックス k_d を定義していますが、内部で InnoDB はこのインデックスを拡張し、それをカラム (d, i1, i2) として処理します。

MySQL 5.6.9 より前では、オプティマイザは拡張セカンダリインデックスの使用方法や使用するかどうかを判断する際に、その主キーカラムを考慮しません。5.6.9 以降、オプティマイザは主キーカラムを考慮するようになったため、より効率的なクエリー実行プランやパフォーマンスの向上につながる可能性があります。

オプティマイザは、ref、range、および index_merge インデックスアクセス、ルースインデックススキャン、結合とソートの最適化、および MIN()/MAX() 最適化に拡張セカンダリインデックスを使用できます。

次の例に、オプティマイザが拡張セカンダリインデックスを使用するかどうかによって、実行プランにどのような影響を与えるか示します。これらの行に t1 が移入されているとします。

```
INSERT INTO t1 VALUES
(1, 1, '1998-01-01'), (1, 2, '1999-01-01'),
(1, 3, '2000-01-01'), (1, 4, '2001-01-01'),
(1, 5, '2002-01-01'), (2, 1, '1998-01-01'),
(2, 2, '1999-01-01'), (2, 3, '2000-01-01'),
(2, 4, '2001-01-01'), (2, 5, '2002-01-01'),
(3, 1, '1998-01-01'), (3, 2, '1999-01-01'),
(3, 3, '2000-01-01'), (3, 4, '2001-01-01'),
(3, 5, '2002-01-01'), (4, 1, '1998-01-01'),
(4, 2, '1999-01-01'), (4, 3, '2000-01-01'),
(4, 4, '2001-01-01'), (4, 5, '2002-01-01'),
(5, 1, '1998-01-01'), (5, 2, '1999-01-01'),
(5, 3, '2000-01-01'), (5, 4, '2001-01-01'),
(5, 5, '2002-01-01');
```

ここで次のクエリーを考慮します。

```
EXPLAIN SELECT COUNT(*) FROM t1 WHERE i1 = 3 AND d = '2000-01-01'
```

この例では、主キーがカラム (i1, i2) で構成され、クエリーで i2 を参照していないため、オプティマイザは主キーを使用できません。代わりに、オプティマイザは (d) に対してセカンダリインデックス k_d を使用でき、実行プランは拡張インデックスを使用するかどうかによって異なります。

オプティマイザがインデックス拡張を考慮しない場合、それはインデックス k_d を (d) のみとして扱います。クエリーの EXPLAIN では次の結果が生成されます。

```
mysql> EXPLAIN SELECT COUNT(*) FROM t1 WHERE i1 = 3 AND d = '2000-01-01'\G
*************************** 1. row ***************************
           id: 1
  select_type: SIMPLE
        table: t1
         type: ref
possible_keys: PRIMARY,k_d
          key: k_d
      key_len: 4
          ref: const
         rows: 5
        Extra: Using where; Using index
```

オプティマイザがインデックス拡張を考慮する場合、それはインデックス k_d を (d, i1, i2) として扱います。この場合、それは左端のインデックスプリフィクス (d, i1) を使用して、より適切な実行プランを生成できます。

```
mysql> EXPLAIN SELECT COUNT(*) FROM t1 WHERE i1 = 3 AND d = '2000-01-01'\G
*************************** 1. row ***************************
           id: 1
  select_type: SIMPLE
        table: t1
         type: ref
possible_keys: PRIMARY,k_d
          key: k_d
      key_len: 8
          ref: const,const
         rows: 1
```

```
    Extra: Using index
```

どちらの場合も key は、オプティマイザがセカンダリインデックス k_d を使用することを示しますが、EXPLAIN 出力には、拡張インデックスの使用による次のような改善が示されます。

- key_len は 4 バイトから 8 バイトになり、キールックアップでカラム d だけでなく、d と i1 も使用されていることを示しています。
- キールックアップで 1 つではなく 2 つのキーパートが使用されるため、ref 値が const から const,const に変更されています。
- rows 数は 5 から 1 に減少し、InnoDB が結果を生成するために調査する必要がある行数が少なくなることを示しています。
- Extra 値が Using where; Using index から Using index に変更されています。このことは、データ行のカラムを参照せずに、インデックスのみを使用して、行を読み取れることを意味します。

拡張インデックスの使用のオプティマイザの動作の違いは、SHOW STATUS でも確認できます。

```
FLUSH TABLE t1;
FLUSH STATUS;
SELECT COUNT(*) FROM t1 WHERE i1 = 3 AND d = '2000-01-01';
SHOW STATUS LIKE 'handler_read%'
```

前のステートメントには FLUSH TABLE と FLUSH STATUS が含まれ、テーブルキャッシュをフラッシュし、ステータスカウンタをクリアします。

インデックス拡張を使用しないと、SHOW STATUS は次の結果を生成します。

```
+-----------------------+-------+
| Variable_name         | Value |
+-----------------------+-------+
| Handler_read_first    | 0     |
| Handler_read_key      | 1     |
| Handler_read_last     | 0     |
| Handler_read_next     | 5     |
| Handler_read_prev     | 0     |
| Handler_read_rnd      | 0     |
| Handler_read_rnd_next | 0     |
+-----------------------+-------+
```

インデックス拡張を使用すると、SHOW STATUS は次の結果を生成します。Handler_read_next 値が 5 から 1 に減少し、インデックスをより効率的に使用していることを示しています。

```
+-----------------------+-------+
| Variable_name         | Value |
+-----------------------+-------+
| Handler_read_first    | 0     |
| Handler_read_key      | 1     |
| Handler_read_last     | 0     |
| Handler_read_next     | 1     |
| Handler_read_prev     | 0     |
| Handler_read_rnd      | 0     |
| Handler_read_rnd_next | 0     |
+-----------------------+-------+
```

optimizer_switch システム変数の use_index_extensions フラグにより、InnoDB テーブルのセカンダリインデックスの使用方法を判断する際に、オプティマイザが主キーカラムを考慮するかどうかを制御できます。デフォルトで、use_index_extensions は有効です。インデックス拡張の使用を無効にするとパフォーマンスが向上するかどうかを確認するには、次のステートメントを使用します。

```
SET optimizer_switch = 'use_index_extensions=off';
```

オプティマイザによるインデックス拡張の使用は、インデックス (16) のキーパートの数と最大キー長 (3072 バイト) への通常の制限によります。

8.2.1.8 IS NULL の最適化

MySQL は、col_name = constant_value に対して使用できる同じ最適化を col_name IS NULL に対しても実行できます。たとえば、MySQL は、インデックスと範囲を使用して、IS NULL を含む NULL を検索できます。

例:

```
SELECT * FROM tbl_name WHERE key_col IS NULL;

SELECT * FROM tbl_name WHERE key_col <=> NULL;

SELECT * FROM tbl_name
  WHERE key_col=const1 OR key_col=const2 OR key_col IS NULL;
```

WHERE 句に、NOT NULL として宣言されているカラムの col_name IS NULL 条件が含まれている場合、その式は最適化により除去されます。この最適化は、とにかくカラムで NULL が生成される可能性がある場合には行われません。たとえば、LEFT JOIN の右側のテーブルから取得されている場合です。

MySQL は、解決済みのサブクエリーで一般的な形式である col_name = expr OR col_name IS NULL の組み合わせを最適化することもできます。この最適化が使用された場合、EXPLAIN で ref_or_null と示されます。

この最適化は、任意のキーパートに対して 1 つの IS NULL を処理できます。

テーブル t2 のカラム a および b にインデックスがあるとして、最適化されるクエリーのいくつかの例:

```
SELECT * FROM t1 WHERE t1.a=expr OR t1.a IS NULL;

SELECT * FROM t1, t2 WHERE t1.a=t2.a OR t2.a IS NULL;

SELECT * FROM t1, t2
  WHERE (t1.a=t2.a OR t2.a IS NULL) AND t2.b=t1.b;

SELECT * FROM t1, t2
  WHERE t1.a=t2.a AND (t2.b=t1.b OR t2.b IS NULL);

SELECT * FROM t1, t2
  WHERE (t1.a=t2.a AND t2.a IS NULL AND ...)
  OR (t1.a=t2.a AND t2.a IS NULL AND ...);
```

ref_or_null はまずリファレンスキーの読み取りを行い、次に NULL キー値のある行の個別の検索を実行します。

この最適化では、1 つの IS NULL レベルしか処理できません。次のクエリーでは、MySQL は式 (t1.a=t2.a AND t2.a IS NULL) に対してのみキールックアップを使用し、b に対してはキーパートを使用できません。

```
SELECT * FROM t1, t2
  WHERE (t1.a=t2.a AND t2.a IS NULL)
  OR (t1.b=t2.b AND t2.b IS NULL);
```

8.2.1.9 LEFT JOIN および RIGHT JOIN の最適化

MySQL は次のように A LEFT JOIN B join_condition を実装します。

- テーブル B は、テーブル A と A が依存するすべてのテーブルに依存して設定されます。
- テーブル A は、LEFT JOIN 条件で使用されるすべてのテーブル (B を除く) に依存して設定されます。
- LEFT JOIN 条件は、テーブル B からの行の取得方法を決定するために使用されます。(言い換えると、WHERE 句内のすべての条件が使用されません)。
- テーブルは常にそれが依存するすべてのテーブルのあとに読み取られることを除き、すべての標準の結合最適化が実行されます。循環依存関係がある場合、MySQL はエラーを発行します。
- すべての標準 WHERE 最適化が実行されます。
- A に WHERE 句に一致する行があるが、B に ON 条件に一致する行がない場合、すべてのカラムが NULL に設定された追加の B 行が生成されます。
- LEFT JOIN を使用して、一部のテーブルに存在しない行を検索し、WHERE 部分の col_name IS NULL のテストを実行した場合 (ここで col_name は NOT NULL と宣言されているカラム)、MySQL は LEFT JOIN 条件に一致する 1 つの行が見つかったあとに、それ以上の行 (の特定のキーの組み合わせ) の検索を停止します。

RIGHT JOIN の実装は、テーブルの役割が逆の LEFT JOIN の場合と類似しています。

結合オプティマイザは、テーブルを結合すべき順序を計算します。LEFT JOIN または STRAIGHT_JOIN によって強制されるテーブル読み取り順序は、確認するテーブル配列が少なくなるため、結合オプティマイザがはるかに高速にその作業を実行するのに役立ちます。これは、次のような種類のクエリーを実行する場合、LEFT JOIN によって d の前に b を読み取るように強制されるため、MySQL がその完全スキャンを実行することを意味します。

```
SELECT *
 FROM a JOIN b LEFT JOIN c ON (c.key=a.key)
 LEFT JOIN d ON (d.key=a.key)
 WHERE b.key=d.key;
```

この例の修正は、a と b が FROM 句内に示される順序を逆にすることです。

```
SELECT *
 FROM b JOIN a LEFT JOIN c ON (c.key=a.key)
 LEFT JOIN d ON (d.key=a.key)
 WHERE b.key=d.key;
```

LEFT JOIN では、生成された NULL 行に対して、WHERE 条件が常に false である場合、LEFT JOIN は通常の結合に変更されます。たとえば、t2.column1 が NULL であった場合、次のクエリーの WHERE 句は false になります。

```
SELECT * FROM t1 LEFT JOIN t2 ON (column1) WHERE t2.column2=5;
```

そのため、クエリーを通常の結合に変換しても問題ありません。

```
SELECT * FROM t1, t2 WHERE t2.column2=5 AND t1.column1=t2.column1;
```

これにより、テーブル t1 の前にテーブル t2 を使用することがより適切なクエリー計画になる場合に、MySQL はそれを実行できるため、高速化できます。テーブルの結合順序についてヒントを提供するには、STRAIGHT_JOIN を使用します。(セクション13.2.9「SELECT 構文」を参照してください。)

8.2.1.10 Nested Loop 結合アルゴリズム

MySQL は、Nested Loop アルゴリズムまたはそのバリエーションを使用してテーブル間の結合を実行します。

Nested Loop 結合アルゴリズム

単純な Nested Loop Join (NLJ) アルゴリズムは、ループ内の最初のテーブルから行を一度に 1 つずつ読み取り、各行を、結合の次のテーブルを処理するネストしたループに渡します。このプロセスは、結合するテーブルが残っている回数だけ繰り返されます。

3 つのテーブル t1、t2、および t3 間の結合が、次の結合型を使用して実行されるとします。

```
Table   Join Type
t1      range
t2      ref
t3      ALL
```

単純な NLJ アルゴリズムを使用した場合、結合は次のように処理されます。

```
for each row in t1 matching range {
 for each row in t2 matching reference key {
  for each row in t3 {
   if row satisfies join conditions,
   send to client
  }
 }
}
```

NLJ アルゴリズムでは、外側のループから内側のループに、一度に 1 つずつ行を渡すため、一般に内側のループで処理されるテーブルを何回も読み取ります。

Block Nested Loop 結合アルゴリズム

Block Nested-Loop (BNL) 結合アルゴリズムは、外側のループで読み取られた行のバッファリングを使用して、内側のループでテーブルを読み取る必要がある回数が削減されます。たとえば、バッファーに 10 行が読み込まれ、このバッファーが次の内側のループに渡される場合、内側のループで読み取られる各行をバッファー内のすべての 10 行と比較できます。これにより、内部テーブルを読み取る必要のある回数が大幅に減少します。

MySQL は、次の条件下で結合バッファリングを使用します。

- join_buffer_size システム変数によって各結合バッファーのサイズが決まります。
- 結合バッファリングは、結合の型が ALL または index である (つまり、使用できるキーがなく、データ行またはインデックス行の完全スキャンがそれぞれ実行される場合) か、または range である場合に使用できま

す。MySQL 5.6 では、セクション8.2.1.14「Block Nested Loop 結合と Batched Key Access 結合」に説明するように、バッファリングの使用が外部結合に適用できるように拡張されています。

- バッファリング可能な結合ごとに 1 つのバッファーが割り当てられるため、特定のクエリーが、複数の結合バッファーを使用して処理されることがあります。
- ALL 型または index 型であっても、最初の非定数テーブルには結合バッファーが割り当てられません。
- 結合バッファーは、結合の実行前に割り当てられ、クエリーの完了後に解放されます。
- 結合バッファーには、行全体ではなく、結合に関連するカラムだけが格納されます。

NLJ アルゴリズム (バッファリングなし) で先述した結合の例では、結合は結合バッファリングを使用すると、次のように実行されます。

```
for each row in t1 matching range {
 for each row in t2 matching reference key {
   store used columns from t1, t2 in join buffer
   if buffer is full {
     for each row in t3 {
       for each t1, t2 combination in join buffer {
         if row satisfies join conditions,
         send to client
       }
     }
     empty buffer
   }
 }
}

if buffer is not empty {
 for each row in t3 {
   for each t1, t2 combination in join buffer {
     if row satisfies join conditions,
     send to client
   }
 }
}
```

S が結合バッファー内の格納される各 t1、t2 の組み合わせのサイズであり、C がバッファー内の組み合わせの数である場合、テーブル t3 がスキャンされる回数は:

```
(S * C)/join_buffer_size + 1
```

join_buffer_size が前のすべての行の組み合わせを保持できるだけの大きさになる時点まで、join_buffer_size の値が大きくなるほど、t3 スキャンの回数は減少します。その時点では、さらに大きくしても速度は向上しなくなります。

8.2.1.11 ネストした結合の最適化

結合を表す構文では、ネストした結合を使用できます。次の説明は、セクション13.2.9.2「JOIN 構文」に説明する結合構文について言及しています。

table_factor の構文は SQL 標準と比較して拡張されています。後者は table_reference のみを受け付け、かっこ内のそれらのリストは受け付けません。これは、table_reference 項目のリストの各カンマを内部結合と同等とみなす場合、保守的な拡張です。例:

```
SELECT * FROM t1 LEFT JOIN (t2, t3, t4)
               ON (t2.a=t1.a AND t3.b=t1.b AND t4.c=t1.c)
```

次と同等です。

```
SELECT * FROM t1 LEFT JOIN (t2 CROSS JOIN t3 CROSS JOIN t4)
               ON (t2.a=t1.a AND t3.b=t1.b AND t4.c=t1.c)
```

MySQL では、CROSS JOIN は INNER JOIN と構文上同等です (それらは相互に置き換え可能です)。標準 SQL では、それらは同等ではありません。INNER JOIN は ON 句と一緒に使用します。CROSS JOIN はそうでない場合でも使用できます。

一般に、内部結合操作のみを含む結合式内のかっこは無視できます。かっこを削除し、操作を左側にグループ化すると、結合式は:

```
t1 LEFT JOIN (t2 LEFT JOIN t3 ON t2.b=t3.b OR t2.b IS NULL)
```

```
  ON t1.a=t2.a
```

次の式に変換されます。

```
(t1 LEFT JOIN t2 ON t1.a=t2.a) LEFT JOIN t3
  ON t2.b=t3.b OR t2.b IS NULL
```

まだ、2 つの式は同等ではありません。これを確認するには、テーブル t1、t2、t3 が次の状態であるとします。

- テーブル t1 には行 (1)、(2) が含まれます
- テーブル t2 には行 (1,101) が含まれます
- テーブル t3 には行 (101) が含まれます

この場合、最初の式は行 (1,1,101,101)、(2,NULL,NULL,NULL) を含む結果セットを返し、2 番目の式は行 (1,1,101,101)、(2,NULL,NULL,101) を返します。

```
mysql> SELECT *
    -> FROM t1
    ->      LEFT JOIN
    ->      (t2 LEFT JOIN t3 ON t2.b=t3.b OR t2.b IS NULL)
    ->      ON t1.a=t2.a;
+------+------+------+------+
| a    | a    | b    | b    |
+------+------+------+------+
|    1 |    1 |  101 |  101 |
|    2 | NULL | NULL | NULL |
+------+------+------+------+

mysql> SELECT *
    -> FROM (t1 LEFT JOIN t2 ON t1.a=t2.a)
    ->      LEFT JOIN t3
    ->      ON t2.b=t3.b OR t2.b IS NULL;
+------+------+------+------+
| a    | a    | b    | b    |
+------+------+------+------+
|    1 |    1 |  101 |  101 |
|    2 | NULL | NULL |  101 |
+------+------+------+------+
```

次の例では、外部結合操作が内部結合操作と一緒に使用されています。

```
t1 LEFT JOIN (t2, t3) ON t1.a=t2.a
```

その式は次の式に変換できません。

```
t1 LEFT JOIN t2 ON t1.a=t2.a, t3.
```

指定されたテーブル状態では、次の 2 つの式は異なる行セットを返します。

```
mysql> SELECT *
    -> FROM t1 LEFT JOIN (t2, t3) ON t1.a=t2.a;
+------+------+------+------+
| a    | a    | b    | b    |
+------+------+------+------+
|    1 |    1 |  101 |  101 |
|    2 | NULL | NULL | NULL |
+------+------+------+------+

mysql> SELECT *
    -> FROM t1 LEFT JOIN t2 ON t1.a=t2.a, t3;
+------+------+------+------+
| a    | a    | b    | b    |
+------+------+------+------+
|    1 |    1 |  101 |  101 |
|    2 | NULL | NULL |  101 |
+------+------+------+------+
```

したがって、外部結合演算子を含む結合式のかっこを省略すると、元の式の結果セットが変わることがあります。

正確に言えば、左外部結合操作の右オペランドと右結合操作の左オペランドのかっこを無視することはできません。言い換えれば、外部結合操作の内部テーブル式のかっこを無視することはできません。ほかのオペランド (外部テーブルのオペランド) のかっこは無視できます。

次の式:

```
(t1,t2) LEFT JOIN t3 ON P(t2.b,t3.b)
```

は次の式と同等です。

```
t1, t2 LEFT JOIN t3 ON P(t2.b,t3.b)
```

任意のテーブル t1,t2,t3 と属性 t2.b および t3.b に対する任意の条件 P の場合。

結合式 (join_table) の結合操作の実行順序が左から右でない場合は常に、ネストした結合と呼びます。次のクエリーを考慮します。

```
SELECT * FROM t1 LEFT JOIN (t2 LEFT JOIN t3 ON t2.b=t3.b) ON t1.a=t2.a
  WHERE t1.a > 1

SELECT * FROM t1 LEFT JOIN (t2, t3) ON t1.a=t2.a
  WHERE (t2.b=t3.b OR t2.b IS NULL) AND t1.a > 1
```

それらのクエリーは次のネストした結合が含まれるとみなされます。

```
t2 LEFT JOIN t3 ON t2.b=t3.b
t2, t3
```

最初のクエリーでは、左結合操作によってネストした結合が形成され、2 番目のクエリーでは、内部結合操作によってそれが形成されます。

最初のクエリーでは、かっこを省略できます。結合式の文法構造によって結合操作の実行の同じ順序が決定されます。2 番目のクエリーでは、かっこを省略できますが、それらがなくてもここの結合式は一義的に解釈できます。(ここの拡張構文では、2 番目のクエリーの (t2, t3) のかっこは必要ですが、理論上はなくても解析できます。LEFT JOIN と ON が式 (t2,t3) の左と右の区切り文字の役割を果たすため、クエリーの構文構造が一義的になります。)

前の例でこれらの点を説明します。

- 内部結合のみを含む (外部結合を含まない) 結合式の場合、かっこは削除できます。かっこを削除して、左から右に評価できます (実際には、任意の順序でテーブルを評価できます)。
- 一般に、外部結合、または内部結合と混在した外部結合の場合には、同じことが当てはまりません。かっこの削除によって、結果が変わることがあります。

ネストした外部結合を含むクエリーは内部結合を含むクエリーと同じパイプライン方式で実行されます。正確には、Nested Loop 結合アルゴリズムのバリエーションが利用されます。Nested Loop 結合がクエリーを実行する際に利用するアルゴリズムスキーマを思い出してください。たとえば、次の形式の 3 つのテーブル T1,T2,T3 に対する結合クエリーがあるとします。

```
SELECT * FROM T1 INNER JOIN T2 ON P1(T1,T2)
              INNER JOIN T3 ON P2(T2,T3)
  WHERE P(T1,T2,T3).
```

ここでは、P1(T1,T2) と P2(T3,T3) が何らかの結合条件 (式での) で、P(T1,T2,T3) はテーブル T1,T2,T3 のカラムに対する条件です。

Nested Loop 結合アルゴリズムでは、このクエリーを次のように実行します。

```
FOR each row t1 in T1 {
  FOR each row t2 in T2 such that P1(t1,t2) {
    FOR each row t3 in T3 such that P2(t2,t3) {
      IF P(t1,t2,t3) {
        t:=t1||t2||t3; OUTPUT t;
      }
    }
  }
}
```

表記 t1||t2||t3 は、「行 t1、t2、および t3 のカラムを連結させて行が構築される」ことを意味します。次のいくつかの例では、行名が表示される場所の NULL は、その行の各カラムに NULL が使用されることを意味します。たとえば、t1||t2||NULL は、行 「t1 と t2 のカラムと、t3 の各カラムの NULL を連結させて行が構築される」ことを意味します。

ここで、ネストした外部結合のあるクエリーを考慮しましょう。

```
SELECT * FROM T1 LEFT JOIN
          (T2 LEFT JOIN T3 ON P2(T2,T3))
          ON P1(T1,T2)
 WHERE P(T1,T2,T3).
```

このクエリーでは、Nested Loop パターンを変更して、次を取得します。

```
FOR each row t1 in T1 {
 BOOL f1:=FALSE;
 FOR each row t2 in T2 such that P1(t1,t2) {
  BOOL f2:=FALSE;
  FOR each row t3 in T3 such that P2(t2,t3) {
   IF P(t1,t2,t3) {
    t:=t1||t2||t3; OUTPUT t;
   }
   f2=TRUE;
   f1=TRUE;
  }
  IF (!f2) {
   IF P(t1,t2,NULL) {
    t:=t1||t2||NULL; OUTPUT t;
   }
   f1=TRUE;
  }
 }
 IF (!f1) {
  IF P(t1,NULL,NULL) {
   t:=t1||NULL||NULL; OUTPUT t;
  }
 }
}
```

一般に、外部結合操作の最初の内部テーブルのネストしたループでは、ループの前にオフにされ、ループのあとにチェックされるフラグが導入されます。フラグは、外部テーブルの現在行で、内側オペランドを表すテーブルからの一致が見つかったときにオンにされます。ループサイクルの最後でフラグがまだオフの場合は、外部テーブルの現在行で一致が見つかりませんでした。この例では、行が内部テーブルのカラムの NULL 値で補完されます。結果の行は、出力の最終チェックまたは次のネストしたループに渡されますが、行が、埋め込まれたすべての外部結合の結合条件を満たしている場合に限られます。

ここでの例では、次の式で表された外部結合テーブルが埋め込まれています。

```
(T2 LEFT JOIN T3 ON P2(T2,T3))
```

内部結合を含むクエリーでは、オプティマイザは次のようなネストしたループの異なる順序を選択することがあります。

```
FOR each row t3 in T3 {
 FOR each row t2 in T2 such that P2(t2,t3) {
  FOR each row t1 in T1 such that P1(t1,t2) {
   IF P(t1,t2,t3) {
    t:=t1||t2||t3; OUTPUT t;
   }
  }
 }
}
```

外部結合を含むクエリーでは、オプティマイザは外部テーブルのループが内部テーブルのループの前に実行される順序のみを選択できます。つまり、外部結合を含むクエリーでは、1 つだけのネスト順序しか使用できません。次のクエリーでは、オプティマイザは 2 つの異なるネストを評価します。

```
SELECT * T1 LEFT JOIN (T2,T3) ON P1(T1,T2) AND P2(T1,T3)
 WHERE P(T1,T2,T3)
```

ネストは次のようになります。

```
FOR each row t1 in T1 {
 BOOL f1:=FALSE;
 FOR each row t2 in T2 such that P1(t1,t2) {
  FOR each row t3 in T3 such that P2(t1,t3) {
   IF P(t1,t2,t3) {
    t:=t1||t2||t3; OUTPUT t;
   }
   f1:=TRUE
```

```
   }
  }
  IF (!f1) {
    IF P(t1,NULL,NULL) {
      t:=t1||NULL||NULL; OUTPUT t;
    }
  }
}
```

および:

```
FOR each row t1 in T1 {
  BOOL f1:=FALSE;
  FOR each row t3 in T3 such that P2(t1,t3) {
    FOR each row t2 in T2 such that P1(t1,t2) {
      IF P(t1,t2,t3) {
        t:=t1||t2||t3; OUTPUT t;
      }
      f1:=TRUE
    }
  }
  IF (!f1) {
    IF P(t1,NULL,NULL) {
      t:=t1||NULL||NULL; OUTPUT t;
    }
  }
}
```

両方のネストで、T1 は外部結合で使用されているため、外側のループで処理される必要があります。T2 と T3 は内部結合で使用されているため、その結合は内側のループで処理される必要があります。ただし、結合は内部結合であるため、T2 と T3 はどちらの順序でも処理できます。

内部結合の Nested Loop アルゴリズムについて説明した際に、クエリー実行のパフォーマンスに与える影響が大きい場合があるという詳細については省きました。いわゆる「プッシュダウン」 条件については説明しませんでした。たとえば、WHERE 条件 P(T1,T2,T3) を論理積標準形によって表現できるとします。

```
P(T1,T2,T2) = C1(T1) AND C2(T2) AND C3(T3).
```

この場合、MySQL は実際に内部結合を含むクエリーの実行に、次の Nested Loop スキーマを使用します。

```
FOR each row t1 in T1 such that C1(t1) {
  FOR each row t2 in T2 such that P1(t1,t2) AND C2(t2)  {
    FOR each row t3 in T3 such that P2(t2,t3) AND C3(t3) {
      IF P(t1,t2,t3) {
        t:=t1||t2||t3; OUTPUT t;
      }
    }
  }
}
```

等位項 C1(T1)、C2(T2)、C3(T3) がそれぞれ、もっとも内側のループから、評価可能なもっとも外側のループまで押し出されることがわかります。C1(T1) がきわめて制限の強い条件である場合、このコンディションプッシュダウンによって、テーブル T1 から内側ループに渡される行数が大幅に少なくなることがあります。結果として、クエリーの実行時間が大幅に短縮される可能性があります。

外部結合を含むクエリーでは、外部テーブルの現在行で内部テーブルに一致があることが見つかったあとにのみ、WHERE 条件がチェックされます。そのため、内側のネストしたループからのプッシュ条件の最適化は、外部結合を含むクエリーには直接適用できません。ここでは、一致が検出されたときにオンにされるフラグによって保護された、条件付きプッシュダウン述語を導入する必要があります。

次の外部結合のある例の場合:

```
P(T1,T2,T3)=C1(T1) AND C(T2) AND C3(T3)
```

保護されたプッシュダウン条件を使用した Nested Loop スキーマは次のようになります。

```
FOR each row t1 in T1 such that C1(t1) {
  BOOL f1:=FALSE;
  FOR each row t2 in T2
      such that P1(t1,t2) AND (f1?C2(t2):TRUE) {
    BOOL f2:=FALSE;
    FOR each row t3 in T3
        such that P2(t2,t3) AND (f1&&f2?C3(t3):TRUE) {
      IF (f1&&f2?TRUE:(C2(t2) AND C3(t3))) {
```

```
      t:=t1||t2||t3; OUTPUT t;
    }
    f2=TRUE;
    f1=TRUE;
  }
  IF (!f2) {
    IF (f1?TRUE:C2(t2) && P(t1,t2,NULL)) {
      t:=t1||t2||NULL; OUTPUT t;
    }
    f1=TRUE;
  }
 }
 IF (!f1 && P(t1,NULL,NULL)) {
    t:=t1||NULL||NULL; OUTPUT t;
 }
}
```

一般に、プッシュダウン述語は P1(T1,T2) や P(T2,T3) などの結合条件から抽出できます。この場合、プッシュダウン述語は、対応する外部結合操作によって生成される NULL が補完された行の述語のチェックを妨げるフラグによっても保護されます。

ここで、ある内部テーブルから、同じネストした結合内の別の内部テーブルへのキーによるアクセスは、それが WHERE 条件からの述語によって引き起こされている場合に、禁止されます。(この例では、条件付きキーアクセスを使用できますが、この手法はまだ MySQL 5.6 に採用されていません。)

8.2.1.12 外部結合の単純化

クエリーの FROM 句内のテーブル式は、多くの場合単純化されます。

パーサー段階で、右外部結合操作を含むクエリーは、左結合操作のみを含む同等のクエリーに変換されます。一般的な場合、変換は次のルールに従って実行されます。

```
(T1, ...) RIGHT JOIN (T2,...) ON P(T1,...,T2,...) =
(T2, ...) LEFT JOIN (T1,...) ON P(T1,...,T2,...)
```

形式 T1 INNER JOIN T2 ON P(T1,T2) のすべての内部結合式は、WHERE 条件に (または埋め込まれる結合の結合条件が存在する場合は、それに) 等位項として結合されるリスト T1,T2、P(T1,T2) によって、置き換えられます。

オプティマイザは、外部結合操作を含む結合クエリーのプランを評価する際、そのような各操作で、外部テーブルが内部テーブルより前にアクセスされるプランのみを考慮に入れます。そのようなプランのみ、Nested Loop スキーマによって、外部結合操作を含むクエリーを実行できるため、オプティマイザのオプションが制限されます。

次の形式のクエリーがあるとします。

```
SELECT * T1 LEFT JOIN T2 ON P1(T1,T2)
  WHERE P(T1,T2) AND R(T2)
```

テーブル T2 の一致する行数を大幅に狭める R(T2) を使用しています。クエリーをそのまま実行した場合、オプティマイザは、テーブル T2 の前にテーブル T1 にアクセスする以外に選択肢がなく、きわめて非効率的な実行プランにつながる可能性があります。

さいわい、MySQL では、WHERE 条件が NULL を受け付けない場合に、それらのクエリーを外部結合操作を含まないクエリーに変換します。条件は、操作のために構築された NULL で補完された行に対し、FALSE または UNKNOWN に評価する場合に、外部結合操作に対して NULL を受け付けないと呼ばれます。

したがって、この外部結合の場合:

```
T1 LEFT JOIN T2 ON T1.A=T2.A
```

次のような条件は NULL を受け付けません。

```
T2.B IS NOT NULL,
T2.B > 3,
T2.C <= T1.C,
T2.B < 2 OR T2.C > 1
```

次のような条件は NULL を受け付けます。

```
T2.B IS NULL,
T1.B < 3 OR T2.B IS NOT NULL,
```

```
T1.B < 3 OR T2.B > 3
```

条件が外部結合操作に対して NULL を受け付けるかどうかをチェックする一般的なルールは単純です。条件は次の場合に NULL を受け付けます。

- その形式が A IS NOT NULL で、A がいずれかの内部テーブルの属性である場合
- いずれかの引数が NULL である場合に、UNKNOWN に評価する内部テーブルへの参照を含む述語である場合
- NULL を受け付けない条件を等位項として含む論理積である場合
- NULL を受け付けない条件の論理和である場合。

条件は、クエリー内で、ある外部結合操作に対しては NULL を受け付けないが、ほかの外部結合操作に対しては NULL を受け付ける場合があります。次のクエリーで:

```
SELECT * FROM T1 LEFT JOIN T2 ON T2.A=T1.A
            LEFT JOIN T3 ON T3.B=T1.B
 WHERE T3.C > 0
```

WHERE 条件は、2 番目の外部結合操作に対しては NULL を受け付けませんが、最初の外部結合操作に対しては NULL を受け付けます。

WHERE 条件がクエリーの外部結合操作に対して NULL を受け付けない場合、外部結合操作は内部結合操作に置き換えられます。

たとえば、前のクエリーは次のクエリーに置き換えられます。

```
SELECT * FROM T1 LEFT JOIN T2 ON T2.A=T1.A
            INNER JOIN T3 ON T3.B=T1.B
 WHERE T3.C > 0
```

元のクエリーでは、オプティマイザは、1 つのアクセス順序 T1,T2,T3 のみと互換性のあるプランを評価します。置換先のクエリーでは、さらにアクセスシーケンス T3,T1,T2 も考慮します。

ある外部結合操作の変換によって、別の操作の変換がトリガーされることがあります。そのため、次のクエリー:

```
SELECT * FROM T1 LEFT JOIN T2 ON T2.A=T1.A
            LEFT JOIN T3 ON T3.B=T2.B
 WHERE T3.C > 0
```

は、まず次のクエリーに変換されます。

```
SELECT * FROM T1 LEFT JOIN T2 ON T2.A=T1.A
            INNER JOIN T3 ON T3.B=T2.B
 WHERE T3.C > 0
```

これは次のクエリーと同等です。

```
SELECT * FROM (T1 LEFT JOIN T2 ON T2.A=T1.A), T3
 WHERE T3.C > 0 AND T3.B=T2.B
```

ここで、条件 T3.B=T2.B は NULL を受け付けず、外部結合をまったく含まないクエリーを取得するため、残りの外部結合操作を内部結合に置き換えることができます。

```
SELECT * FROM (T1 INNER JOIN T2 ON T2.A=T1.A), T3
 WHERE T3.C > 0 AND T3.B=T2.B
```

場合によっては、埋め込まれた外部結合操作を置き換えることに成功しても、埋め込む外部結合を変換できない場合があります。次のクエリー:

```
SELECT * FROM T1 LEFT JOIN
          (T2 LEFT JOIN T3 ON T3.B=T2.B)
          ON T2.A=T1.A
 WHERE T3.C > 0
```

は次に変換されます。

```
SELECT * FROM T1 LEFT JOIN
          (T2 INNER JOIN T3 ON T3.B=T2.B)
          ON T2.A=T1.A
```

```
 WHERE T3.C > 0,
```

それは埋め込む外部結合操作を含む形式にのみ書き換えることができます。

```
SELECT * FROM T1 LEFT JOIN
        (T2,T3)
        ON (T2.A=T1.A AND T3.B=T2.B)
 WHERE T3.C > 0.
```

クエリーに埋め込まれた外部結合操作を変換しようとする場合、WHERE 条件と一緒に埋め込む外部結合に対して、結合条件を考慮する必要があります。次のクエリーで:

```
SELECT * FROM T1 LEFT JOIN
        (T2 LEFT JOIN T3 ON T3.B=T2.B)
        ON T2.A=T1.A AND T3.C=T1.C
 WHERE T3.D > 0 OR T1.D > 0
```

WHERE 条件は埋め込まれた外部結合に対しては NULL を受け付けますが、埋め込む外部結合 T2.A=T1.A AND T3.C=T1.C の結合条件は NULL を受け付けません。そのため、クエリーは次に変換できます。

```
SELECT * FROM T1 LEFT JOIN
        (T2, T3)
        ON T2.A=T1.A AND T3.C=T1.C AND T3.B=T2.B
 WHERE T3.D > 0 OR T1.D > 0
```

8.2.1.13 Multi-Range Read の最適化

セカンダリインデックスでの範囲スキャンを使用して行を読み取ると、テーブルが大きく、ストレージエンジンのキャッシュに格納されていない場合、ベーステーブルへのランダムディスクアクセスが多発する結果になることがあります。Disk-Sweep Multi-Range Read (MRR) 最適化を使用すると、MySQL は、最初にインデックスだけをスキャンし、該当する行のキーを収集することによって、範囲スキャンのランダムディスクアクセスの回数を軽減しようとします。続いてキーがソートされ、最後に主キーの順序を使用してベーステーブルから行が取得されます。Disk-Sweep MRR の目的は、ランダムディスクアクセスの回数を減らし、その代わりに、ベーステーブルデータの順次スキャンを増やすことです。

Multi-Range Read の最適化には、次のメリットがあります。

- MRR により、データ行はインデックスタプルに基づいて、ランダムな順序ではなく、順次アクセスできます。サーバーはクエリー条件を満たすインデックスタプルセットを取得し、それらをデータ行 ID 順に従ってソートし、ソートされたタプルを使用して、データ行を順番に取得します。これにより、データアクセスの効率が向上し、負荷が軽減されます。
- MRR により、範囲インデックススキャンや結合属性にインデックスを使用する等価結合などの、インデックスタプル経由でのデータ行へのアクセスを必要とする操作のキーアクセスのリクエストのバッチ処理が可能になります。MRR はインデックス範囲のシーケンスを反復処理して、対象のインデックスタプルを取得します。これらの結果が累積されると、それらは対応するデータ行にアクセスするために使用されます。データ行の読み取りを開始する前に、すべてのインデックスタプルを取得する必要はありません。

次のシナリオでは、MRR の最適化に利益がある場合について説明しています。

シナリオ A: インデックス範囲スキャンと等価結合操作で、InnoDB テーブルと MyISAM テーブルに対して MRR を使用できます。

1. インデックスタプルの一部はバッファーに累積されます。
2. バッファー内のタプルはそれらのデータ行 ID によってソートされます。
3. データ行には、ソートされたインデックスタプルシーケンスに従ってアクセスされます。

シナリオ B: 複数範囲インデックススキャンで、または属性によって等価結合を実行する際に、NDB テーブルに対して、MRR を使用できます。

1. 単一キー範囲の可能性のある範囲の一部は、クエリーが送信される中央ノード上のバッファーに累積されます。
2. 範囲はデータ行にアクセスする実行ノードに送信されます。
3. アクセスされた行はパッケージに格納され、中央ノードに返送されます。
4. 受け取ったデータ行を含むパッケージはバッファーに入れられます。

5. データ行がバッファーから読み取られます。

MRR が使用された場合は、EXPLAIN 出力の Extra カラムに Using MRR と示されます。

InnoDB と MyISAM は、クエリー結果を生成するために完全なテーブル行にアクセスする必要がない場合、MRR を使用しません。これは、(カバーするインデックス経由で) インデックスタプル内の情報に完全に基づいて結果を生成できる場合であり、MRR にメリットはありません。

MRR を使用でき、(key_part1, key_part2) にインデックスがあると想定するクエリーの例:

```
SELECT * FROM t
  WHERE key_part1 >= 1000 AND key_part1 < 2000
  AND key_part2 = 10000;
```

インデックスは (key_part1, key_part2) 値のタプルから構成され、最初に key_part1 によって、次に key_part2 によって順序付けされます。

MRR を使用しないと、インデックススキャンでは、インデックスタプル内の key_part2 値に関係なく、1000 から最大 2000 の key_part1 範囲のすべてのインデックスタプルがカバーされます。スキャンは範囲内のタプルに 10000 以外の key_part2 値が含まれるかぎり、追加の作業を実行します。

MRR を使用すると、スキャンが key_part1 の 1 つの値 (1000、1001、...、1999) に 1 つずつ、複数の範囲に分割されます。これらの各スキャンは、key_part2 = 10000 のタプルのみを検索する必要があります。インデックスに key_part2 が 10000 でない多数のタプルが含まれる場合、MRR により、読み取られるインデックスタプルが大幅に少なくなります。

これを間隔表記を使用して表すには、非 MRR スキャンで、key_part2 = 10000 のタプルでない多数のタプルが含まれる可能性のあるインデックス範囲 [{1000, 10000}, {2000, MIN_INT}) を調査する必要があります。MRR スキャンでは、key_part2 = 10000 のタプルのみを含む複数の単一ポイント間隔 [{1000, 10000}]、...、[{1999, 10000}] を調査します。

2 つの optimizer_switch システム変数フラグは、MRR 最適化の使用へのインタフェースを提供します。mrr フラグは MRR を有効にするかどうかを制御します。mrr が有効である (on) 場合、mrr_cost_based フラグは、オプティマイザが MRR を使用するか使用しないかをコストベースで選択しようと試みる (on) か、または可能なかぎり MRR を使用する (off) かどうかを制御します。デフォルトで、mrr は on で mrr_cost_based は on です。セクション8.8.5.2「切り替え可能な最適化の制御」を参照してください。

MRR では、ストレージエンジンが、そのバッファーに割り当てることができるメモリーの量のガイドラインとして、read_rnd_buffer_size システム変数の値を使用します。エンジンは最大 read_rnd_buffer_size バイトを使用して、単一のパスで処理する範囲の数を判断します。

8.2.1.14 Block Nested Loop 結合と Batched Key Access 結合

MySQL 5.6 では、結合したテーブルと結合バッファーの両方へのインデックスアクセスを使用する、Batched Key Access (BKA) 結合アルゴリズムが使用できるようになりました。BKA アルゴリズムは、ネストした外部結合を含む、内部結合、外部結合、および準結合操作をサポートします。BKA には、テーブルスキャンの効率性の向上による結合パフォーマンスの改善というメリットもあります。さらに、以前内部結合にのみ使用されていた Block Nested Loop (BNL) 結合アルゴリズムが拡張され、ネストした外部結合を含む、外部結合と準結合操作にも採用できます。

次のセクションでは、元の BNL アルゴリズムの拡張の基礎にある結合バッファー管理、拡張 BNL アルゴリズム、および BKA アルゴリズムについて説明します。準結合戦略については、「準結合変換によるサブクエリーの最適化」を参照してください。

Block Nested Loop および Batched Key Access アルゴリズムの結合バッファー管理

MySQL 5.6 では、MySQL Server は、内部テーブルへのインデックスアクセスなしの内部結合だけでなく、外部結合と、サブクエリーのフラット化のあとに見られる準結合も実行するために結合バッファーを使用できます。さらに、内部テーブルへのインデックスアクセスがある場合、結合バッファーを効率的に使用できます。

結合バッファー管理コードは、目的の行カラムの値を格納する際に、結合バッファー領域を少し効率的に利用します。行カラムの値が NULL の場合に行カラムにバッファー内の追加バイトを割り当てず、VARCHAR 型の値には最小数のバイトが割り当てられます。

コードでは、標準と増分の 2 つの種類のバッファーをサポートします。結合テーブル t1 と t2 に結合バッファー B1 が使用されており、この操作の結果が結合バッファー B2 を使用して、テーブル t3 と結合されるとします。

- 標準結合バッファーには、各結合オペランドからのカラムが格納されます。B2 が標準結合バッファーである場合、B2 に入れられる各行 r は、B1 からの行 r1 のカラムと、テーブル t2 からの一致する行 r2 の対象のカラムから構成されます。
- 増分結合バッファーには、2 つめの結合オペランドによって生成されるテーブルの行からのカラムのみが格納されます。つまり、それは 1 つめのオペランドバッファーからの行の増分になります。B2 が増分結合バッファーである場合、それには、B1 からの行 r1 へのリンクとともに、行 r2 の対象のカラムが格納されます。

増分結合バッファーは常に、前の結合操作からの結合バッファーに相対的な増分になるため、最初の結合操作からのバッファーは常に標準バッファーになります。直前の例では、テーブル t1 および t2 を結合するために使用されるバッファー B1 は標準バッファーである必要があります。

結合操作に使用される増分バッファーの各行には、結合されるテーブルからの行の対象カラムのみが格納されます。これらのカラムには、最初の結合オペランドによって生成されたテーブルからの一致する行の対象カラムへの参照が追加されます。増分バッファー内の複数の行から、カラムが前の結合バッファーに格納されている同じ行 r を参照できます。ただし、これらのすべての行が行 r に一致する場合にかぎります。

増分バッファーにより、前の結合操作で使用されたバッファーからのカラムのコピーの頻度を少なくできます。これにより、一般に、最初の結合オペランドによって生成された行が 2 つめの結合オペランドによって生成される複数の行に一致する可能性があるため、バッファー領域が節約されます。最初のオペランドからの行のコピーを何回も行う必要がありません。さらに、増分バッファーにより、コピー時間の短縮のため、処理時間も節約されます。

MySQL 5.6.3 現在、optimizer_switch システム変数の block_nested_loop および batched_key_access フラグによって、オプティマイザがどのように Block Nested Loop 結合アルゴリズムと Batched Key Access 結合アルゴリズムを使用するかを制御します。デフォルトで、block_nested_loop は on で batched_key_access は off です。セクション8.8.5.2「切り替え可能な最適化の制御」を参照してください。

MySQL 5.6.3 より前では、optimizer_join_cache_level システム変数によって、結合バッファー管理を制御します。この変数の指定可能な値とそれらの意味については、セクション5.1.4「サーバーシステム変数」の説明を参照してください。

準結合戦略については、「準結合変換によるサブクエリーの最適化」を参照してください。

外部結合と準結合の Block Nested Loop アルゴリズム

MySQL 5.6 では、BNL アルゴリズムの元の実装が、外部結合および準結合操作をサポートするように拡張されています。

結合バッファーを使用して、これらの操作が実行されると、バッファーに入れられた各行に一致フラグが付加されます。

結合バッファーを使用して、外部結合操作が実行された場合、2 つめのオペランドによって生成されたテーブルの各行で、結合バッファー内の各行に対する一致がチェックされます。一致が見つかると、新しく拡張された行が形成され (元の行に 2 つめのオペランドからのカラムを追加)、残りの結合操作によるさらなる拡張のために送られます。さらに、バッファー内の一致した行の一致フラグが有効にされます。結合されるテーブル内のすべての行が調査されたあとに、結合バッファーがスキャンされます。有効にされた一致フラグがないバッファーからの各行は、NULL の補完 (2 つめのオペランドの各カラムの NULL 値) によって拡張され、残りの結合操作によるさらなる拡張のために送られます。

MySQL 5.6.3 現在、optimizer_switch システム変数の block_nested_loop フラグによって、オプティマイザが Block Nested Loop アルゴリズムを使用する方法を制御します。デフォルトで、block_nested_loop は on です。セクション8.8.5.2「切り替え可能な最適化の制御」を参照してください。

MySQL 5.6.3 より前では、optimizer_join_cache_level システム変数によって、結合バッファー管理を制御します。この変数の指定可能な値とそれらの意味については、セクション5.1.4「サーバーシステム変数」の説明を参照してください。

EXPLAIN 出力で、Extra 値に Using join buffer (Block Nested Loop) が含まれ、type 値が ALL、index、または range の場合に、テーブルへの BNL の使用が示されます。

準結合戦略については、「準結合変換によるサブクエリーの最適化」を参照してください。

Batched Key Access 結合

MySQL 5.6.3 では Batched Key Access (BKA) 結合アルゴリズムと呼ばれるテーブルの結合の方法を実装しています。BKA は、2 つめの結合オペランドによって生成されるテーブルへのインデックスアクセスがある場合に適

用できます。BNL 結合アルゴリズムと同様、BKA 結合アルゴリズムでは、結合バッファーを使用して、結合操作の最初のオペランドによって生成された行の対象カラムを累積します。次に、BKA アルゴリズムは、バッファー内のすべての行に対し、結合されるテーブルにアクセスするためのキーを構築し、これらのキーをインデックスルックアップのために、データベースエンジンに一括で送信します。キーは、Multi-Range Read (MRR) インタフェース経由で、エンジンに送信されます (セクション8.2.1.13「Multi-Range Read の最適化」を参照してください)。キーの送信後、MRR エンジン関数は最適な方法で、インデックス内のルックアップを実行し、これらのキーによって見つかった結合されたテーブルの行をフェッチし、BKA 結合アルゴリズムに一致する行の提供を開始します。一致する各行は結合バッファー内の行への参照が組み合わされます。

BKA が使用される場合、join_buffer_size の値によって、ストレージエンジンへの個々のリクエストでのキーのバッチの大きさが定義されます。バッファーが大きいほど、結合操作の右側テーブルへの順次アクセスが増え、パフォーマンスを著しく向上させることができます。

BKA を使用するには、optimizer_switch システム変数の batched_key_access フラグが on に設定されている必要があります。BKA では MRR を使用するため、mrr フラグも on に設定されている必要があります。現在、MRR のコスト見積もりはきわめて悲観的です。したがって、BKA を使用するには、mrr_cost_based を off にする必要もあります。次の設定によって、BKA が有効になります。

```
mysql> SET optimizer_switch='mrr=on,mrr_cost_based=off,batched_key_access=on';
```

MRR 関数が実行される 2 つのシナリオがあります。

- 最初のシナリオは、InnoDB や MyISAM などの従来のディスクベースのストレージエンジンで使用されます。これらのエンジンでは通常、結合バッファーからのすべての行のキーが一度に MRR インタフェースに送信されます。エンジン固有の MRR 関数は、送信されたキーのインデックスルックアップを実行し、それらから行 ID (または主キー) を取得して、BKA アルゴリズムからのリクエストによって、これらの選択されたすべての行 ID の行を 1 つずつフェッチします。各行は、結合バッファー内の一致した行へのアクセスを可能にするアソシエーション参照とともに返されます。行は MRR 関数によって最適な方法でフェッチされます。それらは、行 ID (主キー) 順でフェッチされます。これにより、読み取りがランダムな順序ではなく、ディスク順になるため、パフォーマンスが向上します。

- 2 つめのシナリオは、NDB などのリモートストレージエンジンで使用されます。結合バッファーからの行の一部のキーのパッケージが、それらのアソシエーションとともに、MySQL Server (SQL ノード) によって、MySQL Cluster データノードに送信されます。返信で、SQL ノードは、対応するアソシエーションが組み合わされた一致する行のパッケージ (または複数のパッケージ) を受け取ります。BKA 結合アルゴリズムでは、これらの行を取得し、新しく結合された行を構築します。次に、新しいキーセットがデータノードに送信され、返されたパッケージからの行が新しい結合された行の構築に使用されます。このプロセスは、結合バッファーからの最後のキーがデータノードに送信され、SQL ノードがこれらのキーに一致するすべての行を受け取り、結合するまで、続行されます。これにより、SQL ノードによってデータノードに送信されるキーを含むパッケージが少なくなることは、結合操作を実行するために、それとデータノード間のラウンドトリップが少なくなることを意味するため、パフォーマンスが向上します。

最初のシナリオでは、結合バッファーの一部がインデックスルックアップによって選択され、MRR 関数へのパラメータとして渡される行 ID (主キー) を格納するために予約されます。

結合バッファーからの行に対して構築されるキーを格納するための特別なバッファーはありません。代わりに、バッファー内の次の行のキーを構築する関数が、MRR 関数へのパラメータとして渡されます。

EXPLAIN 出力で、Extra 値に Using join buffer (Batched Key Access) が含まれ、type 値が ref または eq_ref の場合に、テーブルへの BKA の使用が示されます。

8.2.1.15 ORDER BY の最適化

場合によって、MySQL は、インデックスを使用して、特別なソートを行わずに ORDER BY 句を満たすことができます。

インデックスのすべての未使用の部分とすべての特別な ORDER BY カラムが WHERE 句内で定数であるかぎり、ORDER BY がインデックスに完全に一致しない場合でもインデックスを使用できます。次のクエリーではインデックスを使用して ORDER BY 部分を解決します。

```
SELECT * FROM t1
  ORDER BY key_part1,key_part2,... ;

SELECT * FROM t1
  WHERE key_part1 = constant
  ORDER BY key_part2;

SELECT * FROM t1
```

```
 ORDER BY key_part1 DESC, key_part2 DESC;

SELECT * FROM t1
 WHERE key_part1 = 1
 ORDER BY key_part1 DESC, key_part2 DESC;

SELECT * FROM t1
 WHERE key_part1 > constant
 ORDER BY key_part1 ASC;

SELECT * FROM t1
 WHERE key_part1 < constant
 ORDER BY key_part1 DESC;

SELECT * FROM t1
 WHERE key_part1 = constant1 AND key_part2 > constant2
 ORDER BY key_part2;
```

場合によって、MySQL は WHERE 句に一致する行を見つけるためにインデックスを使用しても、ORDER BY を解決するために、インデックスを使用できないことがあります。これらの例には、次のようなものが含まれます。

- さまざまなキーに対して ORDER BY を使用します。

  ```
  SELECT * FROM t1 ORDER BY key1, key2;
  ```

- キーの連続しない部分に対して ORDER BY を使用します。

  ```
  SELECT * FROM t1 WHERE key2=constant ORDER BY key_part2;
  ```

- ASC と DESC を混在させます。

  ```
  SELECT * FROM t1 ORDER BY key_part1 DESC, key_part2 ASC;
  ```

- 行をフェッチするために使用されるキーが ORDER BY で使用されるキーと同じでありません。

  ```
  SELECT * FROM t1 WHERE key2=constant ORDER BY key1;
  ```

- キーカラム名以外の項を含む式で ORDER BY を使用します。

  ```
  SELECT * FROM t1 ORDER BY ABS(key);
  SELECT * FROM t1 ORDER BY -key;
  ```

- 多数のテーブルを結合しようとしており、ORDER BY 内のカラムがすべて、行の取得に使用される最初の非定数テーブルからのものとはかぎりません。(これは EXPLAIN 出力で、const 結合型を持たない最初のテーブルです。)

- ORDER BY 式と GROUP BY 式が異なります。

- ORDER BY 句に指定されたカラムのプリフィクスのみにインデックスを設定しています。この場合、インデックスを使用してソート順序を完全には解決できません。たとえば、CHAR(20) カラムがあり、その先頭の 10 バイトだけにインデックスを設定している場合、インデックスで、10 バイト目を越える値を区別できないため、filesort が必要になります。

- 使用されたテーブルインデックスの種類が、行を順番に格納しません。たとえば、これは、MEMORY テーブルの HASH インデックスに当てはまります。

インデックスをソートに使用できるかどうかは、カラム工イリアスの使用によって影響を受けることがあります。カラム t1.a にインデックスが設定されているとします。次のステートメントでは、選択リスト内のカラム名は a です。これは t1.a を指しているため、ORDER BY 内の a への参照にはインデックスを使用できます。

```
SELECT a FROM t1 ORDER BY a;
```

次のステートメントでも、選択リスト内のカラム名は a ですが、これは工イリアス名です。これは ABS(a) を指しているため、ORDER BY 内の a への参照にはインデックスを使用できません。

```
SELECT ABS(a) AS a FROM t1 ORDER BY a;
```

次のステートメントでは、ORDER BY は、選択リスト内のカラムの名前でない名前を参照しています。ただし、t1 には a というカラムがあるため、ORDER BY はそれを使用し、インデックスを使用できます。(当然ながら、結果のソート順序は、ABS(a) の順序とはまったく異なる可能性があります。)

```
SELECT ABS(a) AS b FROM t1 ORDER BY a;
```

デフォルトで、MySQL はすべての GROUP BY col1, col2, ... クエリーを、ORDER BY col1, col2, ... とクエリーに指定したかのように、ソートします。同じカラムリストを含む明示的な ORDER BY 句が含まれている場合、ソート処理は引き続き行われますが、速度の低下なく、MySQL が最適化によってそれを除去します。クエリーに GROUP BY が含まれているが、結果のソートのオーバーヘッドを避けたい場合は、ORDER BY NULL を指定することでソートを抑止できます。例:

```
INSERT INTO foo
SELECT a, COUNT(*) FROM bar GROUP BY a ORDER BY NULL;
```

> 注記
>
> MySQL 5.6 における暗黙の GROUP BY ソートへの依存は、非推奨になっています。グループ化された結果の特定のソート順序を実現するには、明示的な ORDER BY 句を使用することをお勧めします。GROUP BY ソートは、たとえば、オプティマイザがもっとも効率的であると考えるどのような方法でも、グループ化を指示できるようにしたり、ソートオーバーヘッドを回避したりするためなどに、今後のリリースで変更される可能性のある MySQL 拡張機能です。

EXPLAIN SELECT ... ORDER BY を使用すると、MySQL がインデックスを使用してクエリーを解決できるかどうかを確認できます。Extra カラムに Using filesort と表示された場合、それはできません。「セクション 8.8.1「EXPLAIN によるクエリーの最適化」」を参照してください。filesort は、MEMORY ストレージエンジンによって使用されるものと似た固定長の行ストレージフォーマットを使用します。VARCHAR などの可変長型は、固定長を使用して格納されます。

MySQL には、結果をソートして取得するために 2 つの filesort アルゴリズムがあります。元のメソッドは ORDER BY カラムだけを使用します。変更されたメソッドは ORDER BY カラムだけでなく、クエリーによって参照されるすべてのカラムを使用します。

どちらの filesort アルゴリズムを使用するかはオプティマイザが選択します。通常は変更されたアルゴリズムが使用されますが、BLOB カラムや TEXT カラムが含まれる場合を除きます。その場合には元のアルゴリズムが使用されます。どちらのアルゴリズムでも、ソートバッファーサイズは sort_buffer_size システム変数値です。

元の filesort アルゴリズムは次のように動作します。

1. キーに従って、またはテーブルスキャンによって、すべての行を読み取ります。WHERE 句に一致しない行をスキップします。
2. 行ごとに、ソートバッファーに値のペア (ソートキー値と行 ID) を格納します。
3. すべてのペアがソートバッファーに収まる場合は、一時ファイルが作成されません。そうでない場合は、ソートバッファーがいっぱいになると、メモリー内でそれに対して qsort (quicksort) が実行され、それが一時ファイルに書き込まれます。ソートされたブロックへのポインタを保存します。
4. すべての行が読み取られるまで、前の手順を繰り返します。
5. 別の一時ファイルで、最大 MERGEBUFF (7) 領域の 1 つのブロックへのマルチマージを実行します。最初のファイルのすべてのブロックが 2 番目のファイルに格納されるまで、この処理を繰り返します。
6. 残りが MERGEBUFF2 (15) ブロックより少なくなるまで、次を繰り返します。
7. 最後のマルチマージで、行 ID (値のペアの最後の部分) のみが結果ファイルに書き込まれます。
8. 結果ファイルで、行 ID を使用して、ソートされた順序で行を読み取ります。これを最適化するには、行 ID の大きなブロックを読み取り、それらをソートして、それらを使用して、ソートされた順序で行を行バッファーに読み込みます。行バッファーサイズは read_rnd_buffer_size システム変数値です。この手順のコードは sql/records.cc ソースファイルにあります。

このアプローチの問題の 1 つは、WHERE 句の評価時に 1 回と値のペアのソート後にもう 1 回と、2 回行を読み取ることです。さらに、1 回目は行が連続してアクセスされても (テーブルスキャンを実行する場合など)、2 回目はそれらがランダムにアクセスされます。(ソートキーは順序付けされますが、行の位置は順序付けされません。)

変更された filesort アルゴリズムには、行を 2 回読み取ることを回避する最適化が組み込まれています。それは、ソートキー値を記録しますが、行 ID の代わりに、クエリーで参照されているカラムを記録します。変更された filesort アルゴリズムは次のように動作します。

1. WHERE 句に一致する行を読み取ります。

2. 行ごとに、ソートキー値とクエリーで参照されるカラムから構成される値のタプルを記録します。
3. ソートバッファーがいっぱいになると、メモリー内でソートキー値によってタプルをソートし、それを一時ファイルに書き込みます。
4. 一時ファイルのマージソート後、ソートされた順序で行を取得しますが、2 回目はテーブルにアクセスするのではなく、ソートされたタプルから直接必要なカラムを読み取ります。

変更された filesort アルゴリズムを使用すると、タプルが元のメソッドで使用されるペアより長くなり、ソートバッファーに収まるそれらの数が少なくなります。その結果、追加の I/O によって、変更されたアプローチの方が速くなるのではなく、遅くなる可能性があります。速度の低下を防ぐため、オプティマイザはソートタプル内の追加のカラムの合計サイズが max_length_for_sort_data システム変数の値を超えない場合にのみ、変更されたアルゴリズムを使用します。(この変数の値を著しく高く設定すると、高いディスクアクティビティーと低い CPU アクティビティーの組み合わせが見られます。)

filesort が実行されると、EXPLAIN 出力で、Extra カラムに Using filesort が含まれます。さらに、オプティマイザトレース出力には filesort_summary ブロックが含まれます。例:

```
"filesort_summary": {
  "rows": 100,
  "examined_rows": 100,
  "number_of_tmp_files": 0,
  "sort_buffer_size": 25192,
  "sort_mode": "<sort_key, additional_fields>"
}
```

sort_mode 値は、使用されたアルゴリズムとソートバッファー内のタプルの内容に関する情報を提供します。

- <sort_key, rowid>: ソートバッファータプルには、ソートキー値と元のテーブル行の行 ID が含まれます。タプルはソートキー値でソートされ、行 ID は、テーブルからの行の読み取りに使用されます。
- <sort_key, additional_fields>: ソートバッファータプルには、ソートキー値とクエリーで参照されているカラムが含まれます。タプルはソートキー値でソートされ、カラム値は、タプルから直接読み取られます。

オプティマイザのトレースについては、「MySQL Internals: Tracing the Optimizer」を参照してください。

テーブル t1 に、4 つの VARCHAR カラム a、b、c、および d があり、オプティマイザはこのクエリーに filesort を使用するとします。

```
SELECT * FROM t1 ORDER BY a, b;
```

クエリーは a と b でソートしますが、すべてのカラムを返すため、クエリーによって参照されているカラムは a、b、c、および d です。オプティマイザがどの filesort アルゴリズムを選択するかによって、クエリーは次のように実行されます。

元のアルゴリズムの場合、ソートバッファータプルの内容は次のようになります。

```
(fixed size a value, fixed size b value,
row ID into t1)
```

オプティマイザは固定サイズ値でソートします。ソート後、オプティマイザは、順番にタプルを読み取り、各タプル内の行 ID を使用して、t1 から行を読み取り、選択リストカラム値を取得します。

変更されたアルゴリズムの場合、ソートバッファータプルの内容は次のようになります。

```
(fixed size a value, fixed size b value,
a value, b value, c value, d value)
```

オプティマイザは固定サイズ値でソートします。ソート後、オプティマイザは、順番にタプルを読み取り、a、b、c、および d の値を使用して、t1 を再度読み取ることなく、選択リストカラム値を取得します。

filesort が使用されない遅いクエリーでは、max_length_for_sort_data を filesort がトリガーされる適切な値まで小さくしてみてください。

ORDER BY 速度を向上するには、MySQL で、追加のソートフェーズではなく、インデックスを使用させることができるかどうかをチェックします。これが不可能な場合は、次の戦略を試すことができます。

- sort_buffer_size 変数値を増やします。
- read_rnd_buffer_size 変数値を増やします。

- カラムに格納された値を保持するために必要なだけの大きさでカラムを宣言することにより、行あたりに使用する RAM を減らします。たとえば、値が 16 文字を超えることがない場合は、CHAR(16) の方が CHAR(200) よりも適切です。

- tmpdir システム変数を変更して、大量の空き領域のある専用ファイルシステムを指すようにします。変数値には、ラウンドロビン方式で使用される複数のパスをリストできます。この機能を使用して、複数のディレクトリに負荷を分散できます。パスは UNIX ではコロン文字 (「:」)、Windows ではセミコロン文字 (「;」) で区切るようにしてください。パスには、同じディスク上の異なるパーティションではなく、異なる物理ディスクにあるファイルシステム内のディレクトリを指定してください。

ORDER BY にインデックスが使用されないが、LIMIT 句も存在する場合、オプティマイザはマージファイルの使用を避け、メモリー内で行をソートできます。詳細は、セクション8.2.1.19「LIMIT クエリーの最適化」を参照してください。

8.2.1.16 GROUP BY の最適化

GROUP BY 句を満たすもっとも一般的な方法は、テーブル全体をスキャンし、各グループのすべての行が連続する新しい一時テーブルを作成することであり、それにより、この一時テーブルを使用してグループを見つけて、集約関数 (ある場合) を適用できます。場合によって、MySQL はインデックスアクセスを使用することで、それよりはるかに適切に実行し、一時テーブルの作成を回避できます。

GROUP BY にインデックスを使用するためのもっとも重要な前提条件は、すべての GROUP BY カラムが同じインデックスから属性を参照することと、インデックスがそのキーを正しい順序で格納する (たとえば、これは BTREE インデックスで、HASH インデックスではありません) ことです。一時テーブルの使用をインデックスアクセスに置き換えられるかどうかは、クエリー内でインデックスのどの部分が使用されているか、その部分に指定された条件、および選択された集約関数にもよります。

次のセクションで詳しく説明するように、インデックスアクセスによって GROUP BY クエリーを実行する方法は 2 つあります。最初の方法では、グループ化操作はすべての範囲述語 (ある場合) とともに適用されます。2 つめの方法では、まず範囲スキャンを実行し、次に結果タプルをグループ化します。

MySQL では、GROUP BY はソートに使用されるため、サーバーはグループ化に ORDER BY 最適化を適用することもあります。セクション8.2.1.15「ORDER BY の最適化」を参照してください。

ルースインデックススキャン

GROUP BY を処理するもっとも効率的な方法は、インデックスを使用してグループ化するカラムを直接取得することです。このアクセスメソッドでは、MySQL はキーが順序付けられている、インデックス型のプロパティーを使用します。(たとえば、BTREE)。このプロパティーにより、インデックス内のすべての WHERE 条件を満たすキーを考慮する必要なく、インデックス内のルックアップグループを使用できます。このアクセスメソッドはインデックス内のキーの一部だけを考慮するため、ルースインデックススキャンと呼ばれています。WHERE 句がない場合、ルースインデックススキャンでは、グループの数だけキーを読み取りますが、これはすべてのキーの数よりもはるかに少ないことがあります。WHERE 句に範囲述語が含まれる場合 (セクション8.8.1「EXPLAIN によるクエリーの最適化」の range 結合型の説明を参照してください)、ルースインデックススキャンでは範囲条件を満たす各グループの最初のキーをルックアップし、再度最小限の数のキーを読み取ります。これは次の条件の下で可能です。

- クエリーが単一テーブルに対するものです。

- GROUP BY はインデックスの左端のプリフィクスを形成するカラムのみを指定し、ほかのカラムは指定しません。(GROUP BY の代わりに、クエリーに DISTINCT 句がある場合、個々のすべての属性がインデックスの左端のプリフィクスを形成するカラムを参照します。)たとえば、テーブル t1 の (c1,c2,c3) にインデックスがある場合、クエリーに GROUP BY c1, c2, がある場合に、ルースインデックススキャンを適用できます。クエリーに GROUP BY c2, c3 (カラムは左端のプリフィクスでない) または GROUP BY c1, c2, c4 (c4 はインデックス内にない) がある場合は適用できません。

- 選択リスト (ある場合) で使用されている集約関数が、MIN() と MAX() だけであり、それらはすべて同じカラムを参照します。カラムはインデックス内にある必要があり、GROUP BY にあるカラムを追跡する必要があります。

- クエリーで参照された GROUP BY からの部分以外のインデックスの部分は、定数である必要があります (つまり、定数と同等のもので参照されている必要があります) が、MIN() または MAX() 関数の引数を除きます。

- インデックス内のカラムの場合、プリフィクスだけでなく、完全なカラム値にインデックスが設定されている必要があります。たとえば、c1 VARCHAR(20), INDEX (c1(10)) では、インデックスはルースインデックススキャンに使用できません。

ルースインデックススキャンをクエリーに適用できる場合、EXPLAIN 出力で、Extra カラムに Using index for group-by と示されます。

テーブル t1(c1,c2,c3,c4) にインデックス idx(c1,c2,c3) があると仮定します。ルースインデックススキャンアクセスメソッドは、次のクエリーに使用できます。

```
SELECT c1, c2 FROM t1 GROUP BY c1, c2;
SELECT DISTINCT c1, c2 FROM t1;
SELECT c1, MIN(c2) FROM t1 GROUP BY c1;
SELECT c1, c2 FROM t1 WHERE c1 < const GROUP BY c1, c2;
SELECT MAX(c3), MIN(c3), c1, c2 FROM t1 WHERE c2 > const GROUP BY c1, c2;
SELECT c2 FROM t1 WHERE c1 < const GROUP BY c1, c2;
SELECT c1, c2 FROM t1 WHERE c3 = const GROUP BY c1, c2;
```

次に示す理由により、以下のクエリーはこのクイック選択メソッドで実行できません。

- MIN() または MAX() 以外の集約関数があります。

  ```
  SELECT c1, SUM(c2) FROM t1 GROUP BY c1;
  ```

- GROUP BY 句内のカラムがインデックスの左端のプリフィクスを形成していません。

  ```
  SELECT c1, c2 FROM t1 GROUP BY c2, c3;
  ```

- クエリーは GROUP BY 部分のあとに続くキーの部分を参照し、そこに定数と同等のものがありません。

  ```
  SELECT c1, c3 FROM t1 GROUP BY c1, c2;
  ```

  クエリーに WHERE c3 = const が含まれる場合、ルースインデックススキャンを使用できます。

ルースインデックススキャンアクセスメソッドは、選択リスト内で、すでにサポートされている MIN() および MAX() 参照に加えて、ほかの形式の集約関数参照にも適用できます。

- AVG(DISTINCT)、SUM(DISTINCT)、および COUNT(DISTINCT) がサポートされています。AVG(DISTINCT) と SUM(DISTINCT) は 1 つの引数をとります。COUNT(DISTINCT) には複数のカラム引数を指定できます。
- クエリーに GROUP BY または DISTINCT 句があってはいけません。
- ここでも先述したルーススキャンの制限が適用されます。

テーブル t1(c1,c2,c3,c4) にインデックス idx(c1,c2,c3) があると仮定します。ルースインデックススキャンアクセスメソッドは、次のクエリーに使用できます。

```
SELECT COUNT(DISTINCT c1), SUM(DISTINCT c1) FROM t1;

SELECT COUNT(DISTINCT c1, c2), COUNT(DISTINCT c2, c1) FROM t1;
```

ルースインデックススキャンは次のクエリーに適用できません。

```
SELECT DISTINCT COUNT(DISTINCT c1) FROM t1;

SELECT COUNT(DISTINCT c1) FROM t1 GROUP BY c1;
```

タイトインデックススキャン

タイトインデックススキャンは、クエリー条件によって、フルインデックススキャンまたは範囲インデックススキャンのいずれかになります。

ルースインデックススキャンの条件が満たされていなくても、GROUP BY クエリーの一時テーブルの作成を回避できる場合があります。WHERE 句に範囲条件がある場合、このメソッドはこれらの条件を満たすキーだけを読み取ります。そうでない場合は、インデックススキャンを実行します。このメソッドは WHERE 句によって定義された各範囲内のすべてのキーを読み取るか、または範囲条件がなければインデックス全体をスキャンするため、タイトインデックススキャンと呼んでいます。タイトインデックススキャンでは、範囲条件を満たすすべてのキーが見つかったあとにのみ、グループ化操作が実行されます。

このメソッドが機能するためには、クエリー内のすべてのカラムに、GROUP BY キーの前にくるか、または間の部分にあるキーの部分を参照する定数同等条件があれば十分です。同等条件からの定数は、インデックスの完全なプリフィクスを形成できるように、検索キーの「ギャップ」を埋めます。これらのインデックスのプリフィクスは、インデックスルックアップに使用できます。GROUP BY 結果のソートが必要で、インデックスのプリフィクスである検索キーを形成できる場合、順序付けされたインデックス内のプリフィクスによる検索で、すでにすべてのキーが順番に取得されているため、MySQL は余分なソート操作も避けられます。

テーブル t1(c1,c2,c3,c4) にインデックス idx(c1,c2,c3) があると仮定します。次のクエリーは、前述のルースインデックススキャンアクセスメソッドでは機能しませんが、タイトインデックススキャンアクセスメソッドでは機能します。

- GROUP BY にはギャップがありますが、条件 c2 = 'a' によってカバーされます。

```
SELECT c1, c2, c3 FROM t1 WHERE c2 = 'a' GROUP BY c1, c3;
```

- GROUP BY は、キーの最初の部分から開始されませんが、その部分に対して定数を与える条件があります。

```
SELECT c1, c2, c3 FROM t1 WHERE c1 = 'a' GROUP BY c2, c3;
```

8.2.1.17 DISTINCT の最適化

ORDER BY と組み合わされた DISTINCT では多くの場合に一時テーブルが必要です。

DISTINCT では GROUP BY を使用できるため、MySQL が ORDER BY または HAVING 句内の選択したカラムの部分でないカラムをどのように処理するかを学んでください。セクション12.19.3「MySQL での GROUP BY の処理」を参照してください。

ほとんどの場合、DISTINCT 句は GROUP BY の特殊な例と考えることができます。たとえば、次の 2 つのクエリーは同等です。

```
SELECT DISTINCT c1, c2, c3 FROM t1
WHERE c1 > const;

SELECT c1, c2, c3 FROM t1
WHERE c1 > const GROUP BY c1, c2, c3;
```

この同等性のため、GROUP BY クエリーに適用できる最適化は DISTINCT 句のあるクエリーにも適用できます。そのため、DISTINCT クエリー最適化の可能性の詳細については、セクション8.2.1.16「GROUP BY の最適化」を参照してください。

LIMIT row_count を DISTINCT と組み合わせた場合、MySQL は row_count 固有の行が見つかるとただちに停止します。

クエリーに指定されたすべてのテーブルのカラムを使用しない場合、MySQL は最初の一致が見つかるとただちに未使用テーブルのスキャンを停止します。次の例では、t1 が t2 の前に使用され (これは、EXPLAIN で確認できます)、MySQL は t2 (t1 内の特定の行の) で、最初の行を見つけると、t2 からの読み取りを停止します。

```
SELECT DISTINCT t1.a FROM t1, t2 where t1.a=t2.a;
```

8.2.1.18 サブクエリーの最適化

MySQL クエリーオプティマイザには、サブクエリーの評価に使用できるさまざまな戦略があります。IN (または =ANY) サブクエリーの場合、オプティマイザには次の選択肢があります。

- 準結合
- 実体化
- EXISTS 戦略

NOT IN (または <>ALL) サブクエリーの場合、オプティマイザには次の選択肢があります。

- 実体化
- EXISTS 戦略

次のセクションでは、これらの最適化戦略について詳しく説明します。

準結合変換によるサブクエリーの最適化

MySQL 5.6.5 現在、オプティマイザは、このセクションで説明するように、準結合戦略を使用して、サブクエリーの実行を改善します。

2 つのテーブル間の内部結合の場合、結合は、他方のテーブルに一致がある回数だけ、一方のテーブルから 1 行を返します。ただし、問題によっては、重要な情報は一致の数ではなく、一致があるかどうかだけの場合があり

ます。コースカリキュラムのクラスとクラス名簿 (各クラスに登録されている生徒) をそれぞれ一覧表示する class と roster というテーブルがあるとします。実際に生徒が登録されているクラスを一覧表示するには、次の結合を使用できます。

```
SELECT class.class_num, class.class_name
FROM class INNER JOIN roster
WHERE class.class_num = roster.class_num;
```

ただし、結果には、登録された生徒ごとに、各クラスが 1 回ずつ一覧表示されます。ここでの問題では、これは不要な情報の重複です。

class_num が class テーブル内の主キーとすると、重複の抑制は、SELECT DISTINCT を使用して実現できますが、あとで重複を除去するためにだけ、まずすべての一致する行を生成することは非効率的です。

同じ重複のない結果は、次のサブクエリーを使用して取得できます。

```
SELECT class_num, class_name
FROM class
WHERE class_num IN (SELECT class_num FROM roster);
```

ここで、オプティマイザは IN 句に roster テーブルから各クラス番号のインスタンスを 1 つだけ返すサブクエリーが必要であることを認識できます。この場合、クエリーは準結合として実行できます。つまり、roster 内の行に一致する class 内の各行のインスタンスを 1 つだけ返す操作です。

MySQL 5.6.6 より前では、外部クエリーの指定は、単純なテーブルスキャンやカンマ構文を使用した内部結合に制限されており、ビュー参照は不可能でした。5.6.6 現在、外部クエリー指定で外部結合および内部結合構文を使用でき、テーブル参照がベーステーブルでなければいけないという制限はなくなりました。

MySQL では、サブクエリーは準結合として扱われるために、次の条件を満たしている必要があります。

- それは、おそらく AND 式内の項として、WHERE 句または ON 句のトップレベルに表示される IN (または =ANY) サブクエリーである必要があります。例:

  ```
  SELECT ...
  FROM ot1, ...
  WHERE (oe1, ...) IN (SELECT ie1, ... FROM it1, ... WHERE ...);
  ```

  ここで、ot_i と it_i は、クエリーの外側部分と内側部分のテーブルを表し、oe_i と ie_i は、外部テーブルと内部テーブル内のカラムを参照する式を表します。

- それは UNION コンストラクトのない単一の SELECT である必要があります。
- それには GROUP BY または HAVING 句または集約関数が含まれていてはなりません。
- それには、LIMIT を使用した ORDER BY があってはなりません。
- 外部テーブルとおよび内部テーブルの合計数が結合で許可されている最大テーブル数より少なくなければなりません。

サブクエリーは相関する場合と相関しない場合があります。LIMIT と同様に、ORDER BY も使用しなければ、DISTINCT を使用できます。

サブクエリーが先述の条件を満たしている場合、MySQL はそれを準結合に変換し、次の戦略からコストに基づいた選択を行います。

- サブクエリーを結合に変換するか、テーブルプルアウトを使用して、クエリーをサブクエリーテーブルと外部テーブル間の内部結合として実行します。テーブルプルアウトは、テーブルをサブクエリーから外部クエリーに引き出します。
- 重複の除去: 準結合を結合のように実行し、一時テーブルを使用して、重複レコードを削除します。
- FirstMatch: 行の組み合わせの内部テーブルをスキャンし、指定した値グループの複数のインスタンスがある場合、それらすべてを返すのではなく、1 つを選択します。これはスキャンを「ショートカット」し、不要な行の生成をなくします。
- LooseScan: 各サブクエリーの値グループから単一の値を選択できるようにするインデックスを使用して、サブクエリーテーブルをスキャンします。
- サブクエリーをインデックス付きの一時テーブルに実体化し、その一時テーブルを使用して、結合を実行します。インデックスは重複の削除に使用されます。さらに、インデックスはあとで一時テーブルと外部テーブ

ルを結合する際のルックアップにも使用されることがあります。そうでない場合はテーブルがスキャンされます。

重複の除去を除いて、これらの各戦略を有効または無効にするには、optimizer_switch システム変数を使用します。semijoin フラグは準結合を使用するかどうかを制御します。これが on に設定されている場合、firstmatch、loosescan、および materialization フラグによって、使用可能な準結合戦略を詳細に制御できます。これらのフラグはデフォルトで on です。セクション8.8.5.2「切り替え可能な最適化の制御」を参照してください。

準結合戦略の使用は、EXPLAIN 出力で次のように示されます。

- 準結合されたテーブルは、外側の選択に表示されます。EXPLAIN EXTENDED と SHOW WARNINGS には書き換えられたクエリーが示され、準結合構造が表示されます。ここから、準結合から引き出されたテーブルについての情報を得ることができます。サブクエリーが準結合に変換された場合、サブクエリー述語がなくなっており、そのテーブルと WHERE 句が外部クエリー結合リストと WHERE 句にマージされたことがわかります。
- 重複の除去のための一時テーブルの使用は、Extra カラムの Start temporary と End temporary によって示されます。引き出されておらず、Start temporary と End temporary によってカバーされる EXPLAIN 出力行の範囲内にあるテーブルは、一時テーブルにそれらの rowid が格納されます。
- Extra カラムの FirstMatch(tbl_name) は結合のショートカットを示します。
- Extra カラムの LooseScan(m..n) は LooseScan 戦略の使用を示します。m と n はキーパート番号です。
- MySQL 5.6.7 現在、実体化のための一時テーブルの使用は、MATERIALIZED の select_type 値のある行と、<subqueryN> の table 値のある行によって示されます。

  MySQL 5.6.7 より前では、実体化のための一時テーブルの使用は、Extra カラムに、単一のテーブルが使用された場合 Materialize によって示され、複数のテーブルが使用された場合 Start materialize と End materialize によって示されます。Scan が存在する場合、テーブルの読み取りに一時テーブルインデックスは使用されません。そうでない場合は、インデックスルックアップが使用されます。

サブクエリー実体化によるサブクエリーの最適化

MySQL 5.6.5 現在、オプティマイザは、サブクエリー処理の効率向上を可能にする戦略として、サブクエリー実体化を使用します。

実体化を使用しない場合、オプティマイザは、非相関サブクエリーを相関サブクエリーとして書き換えることがあります。たとえば、次の IN サブクエリーは非相関です (where_condition には t2 からのカラムのみが含まれ、t1 からは含まれません)。

```
SELECT * FROM t1
WHERE t1.a IN (SELECT t2.b FROM t2 WHERE where_condition);
```

オプティマイザはこれを EXISTS 相関サブクエリーとして書き換えることがあります。

```
SELECT * FROM t1
WHERE EXISTS (SELECT t2.b FROM t2 WHERE where_condition AND t1.a=t2.b);
```

一時テーブルを使用したサブクエリー実体化により、そのような書き換えを回避し、外部クエリーの行ごとに 1 回ではなく、1 回だけサブクエリーを実行させることができます。実体化は、通常メモリー内に一時テーブルとしてサブクエリー結果を生成することによって、クエリー実行を高速化します。MySQL ははじめてサブクエリー結果を必要としたときに、その結果を一時テーブルに実体化します。あとで結果が必要になったときに、MySQL は再度一時テーブルを参照します。テーブルはルックアップを高速にし、負荷を軽減するため、ハッシュインデックスによってインデックス設定されます。このインデックスは一意で、重複がないため、テーブルを小さくします。

サブクエリー実体化では、可能なかぎりインメモリー一時テーブルを使用しようとし、テーブルが大きくなりすぎた場合、ディスク上のストレージに戻ります。セクション8.4.4「MySQL が内部一時テーブルを使用する仕組み」を参照してください。

MySQL で使用されるサブクエリー実体化では、optimizer_switch システム変数の materialization フラグが on である必要があります。その後実体化は、任意の場所 (選択リスト、WHERE、ON、GROUP BY、HAVING、または ORDER BY 内) に存在する、次のいずれかのユースケースに分類される述語のサブクエリー述語に適用されます。

- 外側の式 oe_i または内側の式 ie_i が NULL 可能でない場合に、述語はこの形式になります。N には 1 以上を指定できます。

```
(oe_1, oe_2, ..., oe_N) [NOT] IN (SELECT ie_1, i_2, ..., ie_N ...)
```

- 単一の外側の式 oe と内側の式 ie がある場合に、述語はこの形式になります。式は NULL 可能にできます。

```
oe [NOT] IN (SELECT ie ...)
```

- 述語は IN または NOT IN で UNKNOWN (NULL) の結果は FALSE の結果と同じ意味になります。

次の例に、UNKNOWN および FALSE 述語評価の同等性の要件が、サブクエリー実体化を使用できるかどうかにどのように影響するかを示します。サブクエリーが非相関になるように、where_condition に t2 からのカラムのみが含まれ、t1 からは含まれないとします。

このクエリーは実体化の対象になります。

```
SELECT * FROM t1
WHERE t1.a IN (SELECT t2.b FROM t2 WHERE where_condition);
```

ここでは、IN 述語が UNKNOWN を返すか、FALSE を返すかは問題ではありません。どちらも t1 からの行はクエリー結果に含まれません。

サブクエリー実体化が使用されない例は、t2.b が NULL 可能カラムである次のクエリーです。

```
SELECT * FROM t1
WHERE (t1.a,t1.b) NOT IN (SELECT t2.a,t2.b FROM t2
                  WHERE where_condition);
```

クエリーで EXPLAIN を使用すると、オプティマイザがサブクエリー実体化を使用しているかどうかの何らかの指示が得られます。実体化を使用しないクエリー実行と比較して、select_type が DEPENDENT SUBQUERY から SUBQUERY に変更されることがあります。これは、外部行ごとに 1 回実行されるサブクエリーの場合、実体化によってサブクエリーが 1 回だけ実行されるようにできることを示します。さらに、EXPLAIN EXTENDED の場合、次の SHOW WARNINGS によって表示されるテキストには materialize materialize および materialized-subquery (MySQL 5.6.6 より前では materialized subselect) が含まれます。

FROM 句内のサブクエリー (派生テーブル) の最適化

MySQL 5.6.3 現在、オプティマイザは FROM 句内のサブクエリー (つまり派生テーブル) をより効率的に処理します。

- FROM 句内のサブクエリーの実体化は、クエリー実行中にその内容が必要になるまで延期されるので、パフォーマンスが向上します。
  - これまで、FROM 句内のサブクエリーは EXPLAIN SELECT ステートメントに対して実体化されていました。これにより、EXPLAIN の目的がクエリーを実行することではなく、クエリー計画情報を取得することであっても、SELECT が部分的に実行されました。この実体化は行われなくなったため、EXPLAIN はそのようなクエリーに対して高速化しています。
  - EXPLAIN 以外のクエリーでは、実体化の遅延によって、それをまったく実行する必要がなくなることがあります。FROM 句内のサブクエリーの結果を別のテーブルに結合するクエリーを考慮します。オプティマイザはその他方のテーブルを最初に処理し、それが行を返さないことがわかると、それ以上結合を実行する必要はないため、オプティマイザはサブクエリーの実体化を完全にスキップできます。
- クエリー実行中に、オプティマイザは派生テーブルにインデックスを追加して、そこからの行の取得を高速化できます。

SELECT クエリーの FROM 句にサブクエリーが表示される、次の EXPLAIN ステートメントを考慮します。

```
EXPLAIN SELECT * FROM (SELECT * FROM t1);
```

オプティマイザは、SELECT 実行中に結果が必要になるまで、サブクエリーの実体化を遅延させて、それを回避します。この例では、クエリーが実行されないため、結果が必要になることはありません。

実行されるクエリーの場合でも、サブクエリー実体化の遅延によって、オプティマイザは実体化を完全に避けられることがあります。FROM 句内のサブクエリーの結果を別のテーブルに結合する次のクエリーを考慮します。

```
SELECT * FROM t1
 JOIN (SELECT t2.f1 FROM t2) AS derived_t2 ON t1.f2=derived_t2.f1
 WHERE t1.f1 > 0;
```

最適化によって t1 が最初に処理され、WHERE 句で空の結果が生成された場合、結合は空である必要があり、サブクエリーは実体化される必要はありません。

最悪の場合 (派生テーブルが実体化される)、追加の作業が実行されないため、クエリーの実行に MySQL 5.6.3 より前と同じ時間がかかります。最善の場合 (派生テーブルが実体化されない)、実体化の実行に必要な時間の分だけ、クエリーの実行が速くなります。

FROM 句内のサブクエリーに実体化が必要な場合、オプティマイザは、実体化されたテーブルにインデックスを追加して、結果へのアクセスを高速化できます。そのようなインデックスによって、テーブルに ref アクセスできる場合、クエリー実行中に読み取る必要があるデータの量を大幅に削減できます。次のクエリーを考慮してください。

```
SELECT * FROM t1
  JOIN (SELECT * FROM t2) AS derived_t2 ON t1.f1=derived_t2.f1;
```

オプティマイザは、derived_t2 のカラム f1 に対してインデックスを構築することで、最低コストの実行プランでの ref アクセスの使用が可能になる場合に、そのようにします。インデックスの追加後、オプティマイザは、実体化された派生テーブルをインデックス付きの通常のテーブルと同じように扱うことができ、生成されたインデックスから同様の利点が得られます。インデックス作成のオーバーヘッドは、インデックスを使用しないクエリー実行のコストと比較して無視できます。ref アクセスが、ほかのアクセスメソッドよりコストが高くなる場合は、インデックスが作成されず、オプティマイザは何も失いません。

EXISTS 戦略によるサブクエリーの最適化

IN 演算子を使用して (または、同等の =ANY を使用して) サブクエリーの結果をテストする比較に、特定の最適化を適用できます。このセクションでは、これらの最適化について、特に NULL 値が存在する課題に関して説明します。説明の最後の部分では、オプティマイザを支援するためにユーザーが実行できることに関する提案も紹介します。

次のようなサブクエリーの比較を考慮します。

```
outer_expr IN (SELECT inner_expr FROM ... WHERE subquery_where)
```

MySQL は「外側から内側に」クエリーを評価します。つまり、まず外側の式 outer_expr の値を取得してから、サブクエリーを実行し、それによって生成される行を取得します。

内側の式 inner_expr が outer_expr と等しい行だけが目的の行であることをサブクエリーに「通知する」ことは、かなり役に立つ最適化です。これを実行するには、適切な等式をサブクエリーの WHERE 句にプッシュダウンします。つまり、この比較は次のように変換されます。

```
EXISTS (SELECT 1 FROM ... WHERE subquery_where AND outer_expr=inner_expr)
```

変換後、MySQL はプッシュダウンされた等式を使用して、サブクエリーの評価時に検査する必要のある行数を制限できます。

より一般的には、N 個の値と N 値の行を返すサブクエリーとの比較は、同じ変換の対象になります。oe_i と ie_i が対応する外側と内側の式の値を表す場合、次のサブクエリー比較は:

```
(oe_1, ..., oe_N) IN
  (SELECT ie_1, ..., ie_N FROM ... WHERE subquery_where)
```

次のようになります。

```
EXISTS (SELECT 1 FROM ... WHERE subquery_where
                    AND oe_1 = ie_1
                    AND ...
                    AND oe_N = ie_N)
```

以下の説明では、簡単にするために、1 組の外側と内側の式の値があると仮定します。

先述の変換には制限があります。これは、可能性のある NULL 値を無視する場合にかぎり有効です。つまり、「プッシュダウン」戦略は、次の 2 つの条件が両方とも true であるかぎり機能します。

- outer_expr と inner_expr は NULL にできません。
- FALSE サブクエリー結果と NULL を区別する必要はありません。(サブクエリーが WHERE 句内の OR または AND 式の一部である場合に、MySQL はユーザーが気にしないものと想定します。)

これらの条件の一方または両方が成立しない場合、最適化は複雑になります。

outer_expr は NULL 以外の値であることがわかっているが、サブクエリーは outer_expr = inner_expr となるような行を生成しないものとします。その場合、outer_expr IN (SELECT ...) は次のように評価されます。

- inner_expr が NULL である行を SELECT が生成する場合は NULL
- SELECT が NULL 以外の値のみを生成するかまたは何も生成しない場合は FALSE

この状況では、outer_expr = inner_expr である行を探すアプローチは有効でなくなります。そのような行を探すことは必要ですが、何も見つからない場合には、inner_expr が NULL となる行も探します。大ざっぱに言うと、サブクエリーは次のように変換できます。

```
EXISTS (SELECT 1 FROM ... WHERE subquery_where AND
      (outer_expr=inner_expr OR inner_expr IS NULL))
```

追加の IS NULL 条件を評価する必要性は、MySQL に ref_or_null アクセスメソッドがある理由です。

```
mysql> EXPLAIN
    -> SELECT outer_expr IN (SELECT t2.maybe_null_key
    ->                       FROM t2, t3 WHERE ...)
    -> FROM t1;
*************************** 1. row ***************************
           id: 1
  select_type: PRIMARY
        table: t1
...
*************************** 2. row ***************************
           id: 2
  select_type: DEPENDENT SUBQUERY
        table: t2
         type: ref_or_null
possible_keys: maybe_null_key
          key: maybe_null_key
      key_len: 5
          ref: func
         rows: 2
        Extra: Using where; Using index
...
```

unique_subquery および index_subquery サブクエリー固有のアクセスメソッドには 「or NULL」 バリアントもあります。ただし、それらは EXPLAIN の出力に表示されないため、EXPLAIN EXTENDED のあとに SHOW WARNINGS を付けて使用する必要があります (警告メッセージの checking NULL に注意してください)。

```
mysql> EXPLAIN EXTENDED
    -> SELECT outer_expr IN (SELECT maybe_null_key FROM t2) FROM t1\G
*************************** 1. row ***************************
           id: 1
  select_type: PRIMARY
        table: t1
...
*************************** 2. row ***************************
           id: 2
  select_type: DEPENDENT SUBQUERY
        table: t2
         type: index_subquery
possible_keys: maybe_null_key
          key: maybe_null_key
      key_len: 5
          ref: func
         rows: 2
        Extra: Using index

mysql> SHOW WARNINGS\G
*************************** 1. row ***************************
  Level: Note
   Code: 1003
Message: select (`test`.`t1`.`outer_expr`,
        (((`test`.`t1`.`outer_expr`) in t2 on
        maybe_null_key checking NULL))) AS `outer_expr IN (SELECT
        maybe_null_key FROM t2)` from `test`.`t1`
```

追加の OR ... IS NULL 条件によってクエリーの実行は多少複雑になり、サブクエリー内の最適化の一部も適用できなくなりますが、通常これは許容できます。

outer_expr が NULL になる可能性がある場合、状況ははるかに悪くなります。「不明な値」としての NULL の SQL の解釈によると、NULL IN (SELECT inner_expr ...) は次のように評価されるはずです。

- SELECT が何らかの行を生成する場合は NULL
- SELECT が行を生成しない場合は FALSE

正しい評価には、SELECT がとにかく何らかの行を生成したかどうかを確認できるようにする必要があるため、outer_expr = inner_expr をサブクエリーにプッシュダウンすることはできません。等式をプッシュダウンできないと、多くの実際のサブクエリーが非常に遅くなるため、これは問題になります。

基本的に、outer_expr の値に応じて、サブクエリーを実行するさまざまな方法が存在する必要があります。

オプティマイザは速度よりも SQL 準拠を選択するため、outer_expr が NULL になる可能性を考慮します。

outer_expr が NULL の場合、次の式を評価するには、SELECT を実行して何らかの行が生成されるかどうかを判断する必要があります。

```
NULL IN (SELECT inner_expr FROM ... WHERE subquery_where)
```

ここで、先述の種類のプッシュダウンされた等式を使用せずに、元の SELECT を実行する必要があります。

一方、outer_expr が NULL でない場合、次の比較が絶対に必要です:

```
outer_expr IN (SELECT inner_expr FROM ... WHERE subquery_where)
```

この比較をプッシュダウンされた条件を使用する式に変換

```
EXISTS (SELECT 1 FROM ... WHERE subquery_where AND outer_expr=inner_expr)
```

この変換を行わないと、サブクエリーが遅くなります。条件をサブクエリーにプッシュダウンするかどうかのジレンマを解決するには、条件を「トリガー」関数にラップします。したがって、次の形式の式は:

```
outer_expr IN (SELECT inner_expr FROM ... WHERE subquery_where)
```

次のように変換されます。

```
EXISTS (SELECT 1 FROM ... WHERE subquery_where
                AND trigcond(outer_expr=inner_expr))
```

より一般的には、サブクエリーの比較が外側の式と内側の式の複数のペアに基づく場合、変換は次の比較をします。

```
(oe_1, ..., oe_N) IN (SELECT ie_1, ..., ie_N FROM ... WHERE subquery_where)
```

さらに、それを次の式に変換します。

```
EXISTS (SELECT 1 FROM ... WHERE subquery_where
                AND trigcond(oe_1=ie_1)
                AND ...
                AND trigcond(oe_N=ie_N)
    )
```

各 trigcond(X) は、次の値に評価される特殊な関数です。

- 「リンクされた」外側の式 oe_i が NULL でない場合は X
- 「リンクされた」外側の式 oe_i が NULL の場合は TRUE

トリガー関数は、CREATE TRIGGER で作成する種類のトリガーではありません。

trigcond() 関数にラップされた等式は、クエリーオプティマイザにとって最高の述語ではありません。ほとんどの最適化では、クエリーの実行時にオンまたはオフになる可能性のある述語を処理できないため、trigcond(X) をすべて不明な関数であるとみなし、無視します。現時点では、トリガー等式は次の最適化で使用できます。

- 参照の最適化: trigcond(X=Y [OR Y IS NULL]) を使用して、ref、eq_ref、または ref_or_null テーブルアクセスを構築できます。
- インデックスルックアップベースのサブクエリー実行エンジン: trigcond(X=Y) を使用して、unique_subquery または index_subquery アクセスを構築できます。
- テーブル条件ジェネレータ: サブクエリーが複数のテーブルの結合である場合、トリガー条件は可能なかぎり早く確認されます。

オプティマイザがトリガー条件を使用して、何らかの種類のインデックスルックアップベースのアクセスを作成する場合 (上記リストの最初の 2 項目に関して)、条件がオフである場合のフォールバック戦略が必要です。このフォールバック戦略は常に同じで、フルテーブルスキャンを実行します。EXPLAIN の出力で、フォールバックは Extra カラムに Full scan on NULL key と表示されます。

```
mysql> EXPLAIN SELECT t1.col1,
    -> t1.col1 IN (SELECT t2.key1 FROM t2 WHERE t2.col2=t1.col2) FROM t1\G
*************************** 1. row ***************************
           id: 1
  select_type: PRIMARY
        table: t1
        ...
*************************** 2. row ***************************
           id: 2
  select_type: DEPENDENT SUBQUERY
        table: t2
         type: index_subquery
possible_keys: key1
          key: key1
      key_len: 5
          ref: func
         rows: 2
        Extra: Using where; Full scan on NULL key
```

EXPLAIN EXTENDED に続いて SHOW WARNINGS を実行すると、トリガー条件を確認できます。

```
*************************** 1. row ***************************
  Level: Note
   Code: 1003
Message: select `test`.`t1`.`col1` AS `col1`,
        <in_optimizer>(`test`.`t1`.`col1`,
        <exists>(<index_lookup>(<cache>(`test`.`t1`.`col1`) in t2
        on key1 checking NULL
        where (`test`.`t2`.`col2` = `test`.`t1`.`col2`) having
        trigcond(<is_not_null_test>(`test`.`t2`.`key1`))))) AS
        `t1.col1 IN (select t2.key1 from t2 where t2.col2=t1.col2)`
        from `test`.`t1`
```

トリガー条件を使用すると、パフォーマンスに多少の影響があります。現在 NULL IN (SELECT ...) 式では、以前に実行されなかった (遅い) フルテーブルスキャンが行われる可能性があります。これは正しい結果を得るための代価です (トリガー条件戦略の目的は、速度ではなく適合性を向上させることでした)。

複数テーブルサブクエリーでは、外側の式が NULL である場合に、結合オプティマイザが最適化を行わないため、NULL IN (SELECT ...) の実行が特に遅くなります。それは、左辺が NULL の場合のサブクエリーの評価はめったにないものと想定しています (そうでないことを示す統計があっても)。一方、外側の式が NULL になる可能性があっても実際にそうなることがない場合、パフォーマンスの低下はありません。

クエリーオプティマイザでクエリーがより適切に実行されるようにするには、次のヒントを使用してください。

- カラムが実際に NOT NULL である場合は、そのように宣言します。(これにより、カラムの条件テストを簡単にすることで、オプティマイザのほかの側面にも役立ちます。)
- NULL と FALSE サブクエリー結果を区別する必要がない場合、遅い実行パスを簡単に回避できます。次のような比較を置き換えます。

  ```
  outer_expr IN (SELECT inner_expr FROM ...)
  ```

  次の式で:

  ```
  (outer_expr IS NOT NULL) AND (outer_expr IN (SELECT inner_expr FROM ...))
  ```

  これにより、MySQL は式の結果が明確になるとただちに AND 部分の評価を停止するため、NULL IN (SELECT ...) が評価されることはなくなります。

subquery_materialization_cost_based により、サブクエリー実体化と IN -> EXISTS サブクエリー変換の選択を制御できます。セクション8.8.5.2「切り替え可能な最適化の制御」を参照してください。

8.2.1.19 LIMIT クエリーの最適化

結果セットから指定した数の行のみが必要な場合、結果セット全体をフェッチして、余分なデータを破棄するのではなく、クエリーで LIMIT 句を使用します。

MySQL は LIMIT row_count 句があり HAVING 句のないクエリーを最適化することがあります。

- LIMIT で少数の行のみを選択すると、MySQL では、通常フルテーブルスキャンを実行するより望ましい特定の場合に、インデックスが使用されます。
- ORDER BY とともに LIMIT row_count を使用した場合、MySQL では、結果全体をソートするのではなく、ソートされた結果の最初の row_count 行が見つかるとすぐにソートを終了します。インデックスを使用して順序付けが行われている場合、これはきわめて高速になります。filesort を実行する必要がある場合、最初の row_count を見つける前に、LIMIT 句を使用しないクエリーに一致するすべての行が選択され、それらのほとんどまたはすべてがソートされます。初期の行が見つかったら、MySQL は結果セットの残りをすべてソートしません。

  この動作をはっきり示している現象の 1 つは、このセクションで後述するように、LIMIT を付けるか付けないかで ORDER BY クエリーは異なる順序で行を返す場合があることです。
- LIMIT row_count を DISTINCT と組み合わせた場合、MySQL は row_count 固有の行が見つかるとただちに停止します。
- 場合によって、GROUP BY はキーを順番に読み取り (またはキーのソートを実行し)、次にキー値が変わるまでサマリーを計算して解決できます。この場合、LIMIT row_count は不要な GROUP BY 値を計算しません。
- MySQL は必要な数の行をクライアントに送信するとただちに、SQL_CALC_FOUND_ROWS が使用されていないかぎり、クエリーを中止します。
- LIMIT 0 は迅速に空のセットを返します。これは、クエリーの妥当性のチェックに役立つことがあります。いずれかの MySQL API を使用している場合、それは結果カラムの型の取得にも使用できます。この技法は mysql クライアントプログラムでは機能せず、そのような場合には、単に Empty set を表示します。代わりに、この目的では SHOW COLUMNS または DESCRIBE を使用します。
- サーバーは、クエリーを解決するために一時テーブルを使用する場合、LIMIT row_count 句を使用して、必要な領域の量を計算します。

複数の行の ORDER BY カラムに同一の値がある場合、サーバーは自由にそれらの行を任意の順序で返しますが、その実行は実行プラン全体によって異なることがあります。言い換えると、それらの行のソート順序は、順序付けされていないカラムに関して決定的ではありません。

実行プランに影響する 1 つの要素は LIMIT であるため、LIMIT を付けるか付けないかで ORDER BY クエリーは異なる順序で行を返すことがあります。category カラムによってソートされるが、id および rating カラムに関して非決定的である次のクエリーを考慮します。

```
mysql> SELECT * FROM ratings ORDER BY category;
+----+----------+--------+
| id | category | rating |
+----+----------+--------+
| 1 |        1 |   4.5 |
| 5 |        1 |   3.2 |
| 3 |        2 |   3.7 |
| 4 |        2 |   3.5 |
| 6 |        2 |   3.5 |
| 2 |        3 |   5.0 |
| 7 |        3 |   2.7 |
+----+----------+--------+
```

LIMIT を含めると、各 category 値内の行の順序に影響することがあります。たとえば、これは有効なクエリー結果です。

```
mysql> SELECT * FROM ratings ORDER BY category LIMIT 5;
+----+----------+--------+
| id | category | rating |
+----+----------+--------+
| 1 |        1 |   4.5 |
| 5 |        1 |   3.2 |
| 4 |        2 |   3.5 |
| 3 |        2 |   3.7 |
| 6 |        2 |   3.5 |
+----+----------+--------+
```

各ケースで、行は ORDER BY カラムによってソートされますが、SQL 標準で必要とされるのはこれだけです。

LIMIT を使用してもしなくても同じ行順序を確保することが重要な場合は、ORDER BY 句に順序を決定的にする追加カラムを含めます。たとえば、id 値が一意である場合、指定した category 値の行を id 順で表示させるようにソートできます。

```
mysql> SELECT * FROM ratings ORDER BY category, id;
```

```
+----+----------+--------+
| id | category | rating |
+----+----------+--------+
|  1 |        1 |    4.5 |
|  5 |        1 |    3.2 |
|  3 |        2 |    3.7 |
|  4 |        2 |    3.5 |
|  6 |        2 |    3.5 |
|  2 |        3 |    5.0 |
|  7 |        3 |    2.7 |
+----+----------+--------+

mysql> SELECT * FROM ratings ORDER BY category, id LIMIT 5;
+----+----------+--------+
| id | category | rating |
+----+----------+--------+
|  1 |        1 |    4.5 |
|  5 |        1 |    3.2 |
|  3 |        2 |    3.7 |
|  4 |        2 |    3.5 |
|  6 |        2 |    3.5 |
+----+----------+--------+
```

MySQL 5.6.2 時点で、オプティマイザは次の形式のクエリー (およびサブクエリー) をより効率的に処理します。

```
SELECT ... FROM single_table ... ORDER BY non_index_column [DESC] LIMIT [M,]N;
```

この種のクエリーは、大きな結果セットの数行だけを表示する Web アプリケーションで一般的なものです。例:

```
SELECT col1, ... FROM t1 ... ORDER BY name LIMIT 10;
SELECT col1, ... FROM t1 ... ORDER BY RAND() LIMIT 15;
```

ソートバッファーには、sort_buffer_size のサイズが入ります。N 行 (M が指定された場合は M+N 行) のソート要素が、ソートバッファーに収まるほど小さい場合、サーバーはマージファイルの使用を回避し、ソートバッファーを優先度キューとして扱うことでメモリー内で完全にソートを実行できます。

- テーブルをスキャンし、キュー内のソート順で選択された各行から選択リストカラムを挿入します。キューがいっぱいになった場合、ソート順で最後の行を押し出します。
- キューから最初の N 行を返します。(M が指定されている場合、最初の M 行をスキップし、次の N 行を返します。)

以前、サーバーはソートにマージファイルを使用して、この操作を実行していました。

- テーブルをスキャンし、テーブルの最後まで次の手順を繰り返します。
  - ソートバッファーがいっぱいになるまで、行を選択します。
  - バッファー内の最初の N 行 (M が指定された場合は M+N 行) をマージファイルに書き込みます。
- マージファイルをソートして、最初の N 行を返します。(M が指定されている場合、最初の M 行をスキップし、次の N 行を返します。)

テーブルスキャンのコストは、キュー方法でもマージファイル方法でも同じであるため、オプティマイザはその他のコストに基づいて、方法を選択します。

- キュー方法では、キューに行を順番に挿入するために多くの CPU を必要とします
- マージファイル方法では、ファイルの書き込みと読み取りの I/O コストとそれをソートするための CPU コストがあります

オプティマイザは、N の特定の値と行サイズのこれらの要素のバランスを考慮します。

8.2.1.20 フルテーブルスキャンを回避する方法

MySQL がフルテーブルスキャンを使用してクエリーを解決する場合、EXPLAIN からの出力には type カラムに ALL と示されます。これは通常は次の条件で発生します。

- テーブルがきわめて小さいため、キールックアップで煩わされるよりもテーブルスキャンを実行する方が速くなります。これは、10 行未満の行や短い行長のテーブルによくあります。
- インデックスが設定されたカラムに対して、ON または WHERE 句に使用可能な制限がありません。

- インデックスが設定されたカラムと定数値を比較していて、MySQL が (インデックスツリーに基づいて) その定数がテーブルのきわめて大きい部分をカバーしており、テーブルスキャンが高速に行われると計算しました。セクション8.2.1.2「MySQL の WHERE 句の最適化の方法」を参照してください。
- 別のカラム経由で、カーディナリティーが低い (多数の行がキー値に一致する) キーを使用しています。この場合、MySQL は、キーを使用して、多数のキールックアップが実行され、テーブルスキャンが高速であると想定します。

小さいテーブルでは、テーブルスキャンは多くの場合に適切であり、実行の影響は無視できます。大きいテーブルでは、オプティマイザがテーブルスキャンを誤って選択しないように、次の技法を試してください。

- ANALYZE TABLE tbl_name を使用して、スキャンされるテーブルのキー分布を更新します。セクション13.7.2.1「ANALYZE TABLE 構文」を参照してください。
- スキャンされるテーブルに FORCE INDEX を使用して、MySQL に、テーブルスキャンは指定したインデックスを使用するのと比較して著しく負荷が大きいことを伝えます。

```
SELECT * FROM t1, t2 FORCE INDEX (index_for_column)
  WHERE t1.col_name=t2.col_name;
```

セクション13.2.9.3「インデックスヒントの構文」を参照してください。

- --max-seeks-for-key=1000 オプションを使用して mysqld を開始するか、または SET max_seeks_for_key=1000 を使用して、オプティマイザに、キースキャンでは 1,000 より多くのキーシークは発生しないと想定するように伝えます。セクション5.1.4「サーバーシステム変数」を参照してください。

8.2.2 DML ステートメントの最適化

このセクションでは、データ操作言語 (DML) ステートメントの INSERT、UPDATE、および DELETE を高速化する方法について説明します。従来の OLTP アプリケーションと最近の Web アプリケーションは一般に多くの小さな DML 操作を実行し、そこでは並列性が不可欠です。データ分析およびレポートアプリケーションは一般に、同時に多くの行に影響する DML 操作を実行しますが、ここでの主な考慮事項は大量のデータを書き込み、インデックスを最新に維持するための I/O です。大量のデータの挿入と更新 (業界では ETL (「extract-transform-load」) と呼ばれる) では、INSERT、UPDATE、および DELETE ステートメントの効果を模倣する、その他の SQL ステートメントや外部コマンドを使用することがあります。

8.2.2.1 INSERT ステートメントの速度

挿入の速度を最適化するには、多くの小さな操作を 1 つの大きな操作に組み合わせます。理想的には、単一の接続を作成し、多くの新しい行のデータを一度に送信し、すべてのインデックスの更新と一貫性チェックを最後まで延期します。

行の挿入に必要な時間は、次の要因によって決まります。ここでの数はおよその割合を示しています。

- 接続: (3)
- サーバーへのクエリーの送信: (2)
- クエリーの解析: (2)
- 行の挿入: (1 ×行サイズ)
- インデックスの挿入: (1 ×インデックス数)
- クローズ: (1)

これには、テーブルを開く初期オーバーヘッドを考慮に入れていません。これは同時実行クエリーごとに 1 回実行されます。

テーブルのサイズによって、log N だけインデックスの挿入が遅くなります (B ツリーインデックスであるとして)。

次の方法を使用して、挿入を高速化できます。

- 同じクライアントから同時に多数の行を挿入する場合は、複数の VALUES リストで INSERT ステートメントを使用して、同時に複数の行を挿入します。これは、個別の単一行の INSERT ステートメントを使用するより、大幅に (場合によっては数倍) 速くなります。空ではないテーブルにデータを追加する場合は、データの挿入を

さらに速くするために、bulk_insert_buffer_size 変数を調整できます。セクション5.1.4「サーバーシステム変数」を参照してください。

- テキストファイルからテーブルをロードする場合は LOAD DATA INFILE を使用します。通常、これは INSERT ステートメントを使用する場合より、20 倍速くなります。セクション13.2.6「LOAD DATA INFILE 構文」を参照してください。
- カラムにデフォルト値があることを利用します。挿入する値がデフォルト値と異なる場合にのみ、明示的に値を挿入します。これにより、MySQL が実行する必要がある解析が減り、挿入速度が向上します。
- InnoDB テーブルに固有のヒントについては、セクション8.5.4「InnoDB テーブルの一括データロード」を参照してください。
- MyISAM テーブルに固有のヒントについては、セクション8.6.2「MyISAM テーブルの一括データロード」を参照してください。

8.2.2.2 UPDATE ステートメントの速度

更新ステートメントは、SELECT クエリーと同様に最適化されますが、書き込みの追加のオーバーヘッドがあります。書き込みの速度は更新されるデータの量と更新されるインデックス数によって異なります。変更がないインデックスは更新されません。

更新を速くするもう 1 つの方法は、更新を遅延して、あとで 1 行で多くの更新を実行することです。複数の更新をまとめて実行することで、テーブルをロックした場合に、一度に 1 つずつ実行するよりはるかに高速になります。

動的な行フォーマットを使用する MyISAM テーブルの場合、行を長い合計長に更新すると、行が分割されることがあります。頻繁にこれを実行する場合は、ときどき OPTIMIZE TABLE を使用することがきわめて重要になります。セクション13.7.2.4「OPTIMIZE TABLE 構文」を参照してください。

8.2.2.3 DELETE ステートメントの速度

MyISAM テーブル内の個々の行を削除するために必要な時間は、インデックスの数に正確に比例します。行をもっと速く削除するには、key_buffer_size システム変数を増やして、キーキャッシュのサイズを大きくできます。セクション8.11.2「サーバーパラメータのチューニング」を参照してください。

MyISAM テーブルからすべての行を削除するには、TRUNCATE TABLE tbl_name の方が DELETE FROM tbl_name より速くなります。切り捨て操作はトランザクションセーフではありません。アクティブなトランザクションやアクティブなテーブルロックの途中で試みるとエラーが発生します。セクション13.1.33「TRUNCATE TABLE 構文」を参照してください。

8.2.3 データベース権限の最適化

権限のセットアップが複雑であるほど、すべての SQL ステートメントに適用されるオーバーヘッドが大きくなります。GRANT ステートメントによって確立された権限を簡単にすることで、クライアントがステートメントを実行するときの MySQL の権限チェックのオーバーヘッドを軽減できます。たとえば、テーブルレベルやカラムレベルの権限を付与しない場合、サーバーは tables_priv と columns_priv テーブルの内容をチェックする必要はなくなります。同じように、どのアカウントにもリソース制限を設けない場合、サーバーはリソースのカウントを実行する必要がありません。ステートメント処理の負荷が著しく高い場合は、簡略化した付与構造を使用して、権限チェックのオーバーヘッドを軽減することを考慮してください。

8.2.4 INFORMATION_SCHEMA クエリーの最適化

データベースをモニターするアプリケーションでは、INFORMATION_SCHEMA テーブルを頻繁に使用することがあります。INFORMATION_SCHEMA テーブルに対する特定の種類のクエリーは、高速に実行するように最適化できます。この目標は、ファイル操作 (ディレクトリのスキャンやテーブルファイルを開くなど) を最小限にし、これらの動的テーブルを構成する情報を収集することです。これらの最適化は、INFORMATION_SCHEMA テーブルの検索にどのような照合順序が使われるかに影響します。詳細は、セクション10.1.7.9「照合順序と INFORMATION_SCHEMA 検索」を参照してください。

1) WHERE 句のデータベース名とテーブル名には定数のルックアップ値を使用してみます

この原則は次のように活用できます。

- データベースやテーブルをルックアップするには、リテラル値、定数を返す関数、スカラーサブクエリーなど、定数に評価される式を使用します。

- 一致するデータベースディレクトリ名を見つけるためにデータディレクトリのスキャンが必要になるため、非定数のデータベース名ルックアップ値を使用する (またはルックアップ値を使用しない) クエリーを避けます。
- データベース内では、一致するテーブルファイルを見つけるためにデータベースディレクトリのスキャンが必要になるため、非定数のテーブル名ルックアップ値を使用する (またはルックアップ値を使用しない) クエリーを避けます。

この原則は、定数のルックアップ値によって、サーバーがディレクトリスキャンを回避できるカラムを示している次の表で示されている INFORMATION_SCHEMA テーブルに適用されます。たとえば、TABLES から選択する場合は、WHERE 句で TABLE_SCHEMA に定数のルックアップ値を使用すると、データディレクトリのスキャンを回避できます。

テーブル	データディレクトリスキャンを避けるために指定するカラム	データベースディレクトリスキャンを避けるために指定するカラム
COLUMNS	TABLE_SCHEMA	TABLE_NAME
KEY_COLUMN_USAGE	TABLE_SCHEMA	TABLE_NAME
PARTITIONS	TABLE_SCHEMA	TABLE_NAME
REFERENTIAL_CONSTRAINTS	CONSTRAINT_SCHEMA	TABLE_NAME
STATISTICS	TABLE_SCHEMA	TABLE_NAME
TABLES	TABLE_SCHEMA	TABLE_NAME
TABLE_CONSTRAINTS	TABLE_SCHEMA	TABLE_NAME
TRIGGERS	EVENT_OBJECT_SCHEMA	EVENT_OBJECT_TABLE
VIEWS	TABLE_SCHEMA	TABLE_NAME

特定の定数のデータベース名に制限されたクエリーの利点は、指定したデータベースディレクトリのみをチェックするだけで済むことです。例:

```
SELECT TABLE_NAME FROM INFORMATION_SCHEMA.TABLES
WHERE TABLE_SCHEMA = 'test';
```

リテラルのデータベース名 test を使用すると、データベースがいくつあるかに関係なく、サーバーは test データベースディレクトリだけをチェックできます。対照的に、次のクエリーでは、パターン 'test%' に一致するデータベース名を特定するために、データディレクトリのスキャンが必要であるため、効率が低下します。

```
SELECT TABLE_NAME FROM INFORMATION_SCHEMA.TABLES
WHERE TABLE_SCHEMA LIKE 'test%';
```

特定の定数のテーブル名に制限されたクエリーの場合、対応するデータベースディレクトリ内の指定したテーブルのみをチェックするだけで済みます。例:

```
SELECT TABLE_NAME FROM INFORMATION_SCHEMA.TABLES
WHERE TABLE_SCHEMA = 'test' AND TABLE_NAME = 't1';
```

リテラルのテーブル名 t1 を使用すると、test データベースにテーブルがいくつあるかに関係なく、サーバーは t1 テーブルのファイルだけをチェックできます。対照的に、次のクエリーでは、パターン 't%' に一致するテーブル名を特定するために、test データベースディレクトリのスキャンが必要です。

```
SELECT TABLE_NAME FROM INFORMATION_SCHEMA.TABLES
WHERE TABLE_SCHEMA = 'test' AND TABLE_NAME LIKE 't%';
```

次のクエリーでは、パターン 'test%' に一致するデータベース名を特定するためにデータディレクトリをスキャンする必要があり、一致するデータベースごとに、パターン 't%' に一致するテーブル名を特定するためにデータベースディレクトリをスキャンする必要があります。

```
SELECT TABLE_NAME FROM INFORMATION_SCHEMA.TABLES
WHERE TABLE_SCHEMA = 'test%' AND TABLE_NAME LIKE 't%';
```

2) 開く必要のあるテーブルファイルの数が最小になるクエリーを書きます

特定の INFORMATION_SCHEMA テーブルカラムを参照するクエリーでは、開く必要のあるテーブルファイルの数を最小にするいくつかの最適化を使用できます。例:

```
SELECT TABLE_NAME, ENGINE FROM INFORMATION_SCHEMA.TABLES
WHERE TABLE_SCHEMA = 'test';
```

この場合、サーバーがデータベースディレクトリをスキャンしてデータベース内のテーブルの名前を特定したら、さらにファイルシステムをルックアップしなくても、それらの名前を使用できるようになります。したがって、TABLE_NAME はファイルを開く必要はありません。ENGINE (ストレージエンジン) の値は、テーブルの .frm ファイルを開くことで特定でき、.MYD や .MYI などのほかのテーブルファイルにアクセスすることはありません。

MyISAM テーブルの INDEX_LENGTH など、一部の値では .MYD または .MYI ファイルも開く必要があります。

ファイルオープンの最適化の種類は、次のように表されます。

- SKIP_OPEN_TABLE: テーブルファイルを開く必要はありません。データベースディレクトリをスキャンすることによって、クエリー内ですでに情報を使用できるようになっています。
- OPEN_FRM_ONLY: テーブルの .frm ファイルのみを開く必要があります。
- OPEN_TRIGGER_ONLY: テーブルの .TRG ファイルのみを開く必要があります。
- OPEN_FULL_TABLE: 最適化されていない情報のルックアップ。.frm、.MYD、および .MYI ファイルを開く必要があります。

次のリストに、上記の最適化の種類がどのように INFORMATION_SCHEMA テーブルカラムに適用されるかを示します。指定されていないテーブルとカラムには、最適化が適用されません。

- COLUMNS: OPEN_FRM_ONLY がすべてのカラムに適用されます
- KEY_COLUMN_USAGE: OPEN_FULL_TABLE がすべてのカラムに適用されます
- PARTITIONS: OPEN_FULL_TABLE がすべてのカラムに適用されます
- REFERENTIAL_CONSTRAINTS: OPEN_FULL_TABLE がすべてのカラムに適用されます
- STATISTICS:

カラム	最適化の種類
TABLE_CATALOG	OPEN_FRM_ONLY
TABLE_SCHEMA	OPEN_FRM_ONLY
TABLE_NAME	OPEN_FRM_ONLY
NON_UNIQUE	OPEN_FRM_ONLY
INDEX_SCHEMA	OPEN_FRM_ONLY
INDEX_NAME	OPEN_FRM_ONLY
SEQ_IN_INDEX	OPEN_FRM_ONLY
COLUMN_NAME	OPEN_FRM_ONLY
COLLATION	OPEN_FRM_ONLY
CARDINALITY	OPEN_FULL_TABLE
SUB_PART	OPEN_FRM_ONLY
PACKED	OPEN_FRM_ONLY
NULLABLE	OPEN_FRM_ONLY
INDEX_TYPE	OPEN_FULL_TABLE
COMMENT	OPEN_FRM_ONLY

- TABLES:

カラム	最適化の種類
TABLE_CATALOG	SKIP_OPEN_TABLE
TABLE_SCHEMA	SKIP_OPEN_TABLE
TABLE_NAME	SKIP_OPEN_TABLE
TABLE_TYPE	OPEN_FRM_ONLY
ENGINE	OPEN_FRM_ONLY

カラム	最適化の種類
VERSION	OPEN_FRM_ONLY
ROW_FORMAT	OPEN_FULL_TABLE
TABLE_ROWS	OPEN_FULL_TABLE
AVG_ROW_LENGTH	OPEN_FULL_TABLE
DATA_LENGTH	OPEN_FULL_TABLE
MAX_DATA_LENGTH	OPEN_FULL_TABLE
INDEX_LENGTH	OPEN_FULL_TABLE
DATA_FREE	OPEN_FULL_TABLE
AUTO_INCREMENT	OPEN_FULL_TABLE
CREATE_TIME	OPEN_FULL_TABLE
UPDATE_TIME	OPEN_FULL_TABLE
CHECK_TIME	OPEN_FULL_TABLE
TABLE_COLLATION	OPEN_FRM_ONLY
CHECKSUM	OPEN_FULL_TABLE
CREATE_OPTIONS	OPEN_FRM_ONLY
TABLE_COMMENT	OPEN_FRM_ONLY

- TABLE_CONSTRAINTS: OPEN_FULL_TABLE がすべてのカラムに適用されます
- TRIGGERS: OPEN_TRIGGER_ONLY がすべてのカラムに適用されます
- VIEWS:

カラム	最適化の種類
TABLE_CATALOG	OPEN_FRM_ONLY
TABLE_SCHEMA	OPEN_FRM_ONLY
TABLE_NAME	OPEN_FRM_ONLY
VIEW_DEFINITION	OPEN_FRM_ONLY
CHECK_OPTION	OPEN_FRM_ONLY
IS_UPDATABLE	OPEN_FULL_TABLE
DEFINER	OPEN_FRM_ONLY
SECURITY_TYPE	OPEN_FRM_ONLY
CHARACTER_SET_CLIENT	OPEN_FRM_ONLY
COLLATION_CONNECTION	OPEN_FRM_ONLY

3) EXPLAIN を使用して、サーバーがクエリーに INFORMATION_SCHEMA 最適化を使用できるかどうかを判断します

これは特に、複数のデータベースの情報を検索し、長時間かかり、パフォーマンスに影響を与える可能性のある INFORMATION_SCHEMA クエリーに適用されます。先述の最適化のうち、サーバーが INFORMATION_SCHEMA クエリーの評価に使用できるものがあれば、EXPLAIN の出力の Extra 値に示されます。次の例は、Extra 値に表示されることが予想される情報の種類を示しています。

```
mysql> EXPLAIN SELECT TABLE_NAME FROM INFORMATION_SCHEMA.VIEWS WHERE
    -> TABLE_SCHEMA = 'test' AND TABLE_NAME = 'v1'\G
*************************** 1. row ***************************
           id: 1
  select_type: SIMPLE
        table: VIEWS
         type: ALL
possible_keys: NULL
          key: TABLE_SCHEMA,TABLE_NAME
      key_len: NULL
          ref: NULL
         rows: NULL
        Extra: Using where; Open_frm_only; Scanned 0 databases
```

定数のデータベースルックアップ値およびテーブルルックアップ値を使用すると、サーバーはディレクトリスキャンを回避できます。VIEWS.TABLE_NAME の参照では、.frm ファイルのみを開く必要があります。

```
mysql> EXPLAIN SELECT TABLE_NAME, ROW_FORMAT FROM INFORMATION_SCHEMA.TABLES\G
*************************** 1. row ***************************
           id: 1
  select_type: SIMPLE
        table: TABLES
         type: ALL
possible_keys: NULL
          key: NULL
      key_len: NULL
          ref: NULL
         rows: NULL
        Extra: Open_full_table; Scanned all databases
```

ルックアップ値が指定されていない (WHERE 句がない) ため、サーバーはデータディレクトリと各データベースディレクトリをスキャンする必要があります。このようにして特定された各テーブルについて、テーブル名と行フォーマットが選択されます。TABLE_NAME では、さらにテーブルファイルを開く必要はありません (SKIP_OPEN_TABLE 最適化が適用されます)。ROW_FORMAT では、すべてのテーブルファイルを開く必要があります (OPEN_FULL_TABLE が適用されます)。EXPLAIN は OPEN_FULL_TABLE (SKIP_OPEN_TABLE より負荷が大きいため) をレポートします。

```
mysql> EXPLAIN SELECT TABLE_NAME, TABLE_TYPE FROM INFORMATION_SCHEMA.TABLES
    -> WHERE TABLE_SCHEMA = 'test'\G
*************************** 1. row ***************************
           id: 1
  select_type: SIMPLE
        table: TABLES
         type: ALL
possible_keys: NULL
          key: TABLE_SCHEMA
      key_len: NULL
          ref: NULL
         rows: NULL
        Extra: Using where; Open_frm_only; Scanned 1 database
```

テーブル名のルックアップ値が指定されていないため、サーバーは test データベースディレクトリをスキャンする必要があります。TABLE_NAME カラムと TABLE_TYPE カラムには、それぞれ SKIP_OPEN_TABLE 最適化と OPEN_FRM_ONLY 最適化が適用されます。EXPLAIN は OPEN_FRM_ONLY (これの方が負荷が大きいため) をレポートします。

```
mysql> EXPLAIN SELECT B.TABLE_NAME
    -> FROM INFORMATION_SCHEMA.TABLES AS A, INFORMATION_SCHEMA.COLUMNS AS B
    -> WHERE A.TABLE_SCHEMA = 'test'
    -> AND A.TABLE_NAME = 't1'
    -> AND B.TABLE_NAME = A.TABLE_NAME\G
*************************** 1. row ***************************
           id: 1
  select_type: SIMPLE
        table: A
         type: ALL
possible_keys: NULL
          key: TABLE_SCHEMA,TABLE_NAME
      key_len: NULL
          ref: NULL
         rows: NULL
        Extra: Using where; Skip_open_table; Scanned 0 databases
*************************** 2. row ***************************
           id: 1
  select_type: SIMPLE
        table: B
         type: ALL
possible_keys: NULL
          key: NULL
      key_len: NULL
          ref: NULL
         rows: NULL
        Extra: Using where; Open_frm_only; Scanned all databases;
               Using join buffer
```

最初の EXPLAIN 出力行の場合: 定数のデータベースルックアップ値およびテーブルルックアップ値により、サーバーは TABLES の値のディレクトリスキャンを回避できます。TABLES.TABLE_NAME の参照には、さらにテーブルファイルは必要ありません。

2 つめの EXPLAIN 出力行の場合 : COLUMNS テーブルのすべての値が OPEN_FRM_ONLY ルックアップであるため、COLUMNS.TABLE_NAME では、.frm ファイルを開く必要があります。

```
mysql> EXPLAIN SELECT * FROM INFORMATION_SCHEMA.COLLATIONS\G
*************************** 1. row ***************************
           id: 1
  select_type: SIMPLE
        table: COLLATIONS
         type: ALL
possible_keys: NULL
          key: NULL
      key_len: NULL
          ref: NULL
         rows: NULL
        Extra:
```

この場合、COLLATIONS は最適化を使用できる INFORMATION_SCHEMA テーブルのいずれでもないため、最適化は適用されません。

8.2.5 その他の最適化のヒント

このセクションでは、クエリー処理速度を向上するためのさまざまな多くのヒントを示します。

- 接続のオーバーヘッドを回避するには、データベースに対して永続的な接続を使用します。永続的な接続を使用できないため、データベースに対して多くの新しい接続を開始する場合、thread_cache_size 変数の値の変更が必要になることがあります。セクション8.11.2「サーバーパラメータのチューニング」を参照してください。

- すべてのクエリーがテーブル内に作成したインデックスを実際に使用していることを常に確認します。MySQL では、EXPLAIN ステートメントでこれを実行できます。「セクション8.8.1「EXPLAIN によるクエリーの最適化」」を参照してください。

- 頻繁に更新される MyISAM テーブルに対する複雑な SELECT クエリーを避け、リーダーとライターの競合のために発生するテーブルロックの問題を回避するようにしてください。

- MyISAM は同時挿入をサポートしています。テーブルのデータファイルの途中に空きブロックがなければ、ほかのスレッドがテーブルから読み取るのと同時に新しい行をそれに INSERT できます。これを実行できることが重要な場合、行の削除を避けるようにテーブルを使用することを考慮してください。別の可能性は、テーブルの大量の行を削除したあとに OPTIMIZE TABLE を実行して、テーブルをデフラグすることです。この動作は concurrent_insert 変数の設定によって変更されます。行を削除したテーブルにも新しい行を強制的に追加 (したがって同時挿入を許可) できます。セクション8.10.3「同時挿入」を参照してください。

- ARCHIVE テーブルで発生したデータ圧縮問題を修正するには、OPTIMIZE TABLE を使用できます。セクション15.5「ARCHIVE ストレージエンジン」を参照してください。

- 通常 expr1、expr2、... の順で行を取得する場合は、ALTER TABLE ... ORDER BY expr1, expr2, ... を使用します。テーブルを大幅に変更したあとにこのオプションを使用することで、パフォーマンスを向上できることがあります。

- 場合によって、ほかのカラムの情報に基づいて「ハッシュされた」カラムを導入することが役立つ場合があります。このカラムが短く、十分に一意で、インデックスが設定されている場合は、多数のカラムに「広範な」インデックスを使用するより大幅に高速化できる可能性があります。MySQL では、この追加カラムをきわめて簡単に使用できます。

  ```
  SELECT * FROM tbl_name
    WHERE hash_col=MD5(CONCAT(col1,col2))
    AND col1='constant' AND col2='constant';
  ```

- 頻繁に変更される MyISAM テーブルでは、すべての可変長カラム (VARCHAR、TEXT、および BLOB) を避けるようにします。テーブルに 1 つしか可変長カラムが含まれていない場合でも、テーブルは動的行フォーマットを使用します。第15章「代替ストレージエンジン」を参照してください。

- 一般に、行が大きくなるためだけに、1 つのテーブルを異なるテーブルに分割することは有益ではありません。行へのアクセスで、もっとも大きくパフォーマンスに打撃を与えるものは、行の先頭バイトを見つけるために必要なディスクシークです。データが見つかったあとは、ほとんどの最新のディスクで、大多数のアプリケーションに十分な速度で行全体を読み取ることができます。テーブルを分割することがかなりの違いをもたらす状況は、固定の行サイズに変更できる動的行フォーマットを使用している MyISAM テーブルの場合か、またはテーブルを著しく頻繁にスキャンする必要があるが、ほとんどのカラムには必要でない場合だけです。第15章「代替ストレージエンジン」を参照してください。

- 多数の行の情報に基づいたカウントなど、結果を頻繁に計算する必要がある場合、新しいテーブルを導入し、リアルタイムでカウンタを更新する方が望ましいことがあります。次のような形式の更新はきわめて高速です。

```
UPDATE tbl_name SET count_col=count_col+1 WHERE key_col=constant;
```

これは、テーブルレベルのロック (単一ライターと複数リーダー) しかない MyISAM のような MySQL ストレージエンジンを使用する場合に、きわめて重要です。また、この場合に行ロックマネージャーが実行する必要があることは少ないため、ほとんどのデータベースシステムでパフォーマンスが向上します。

- 大きなログテーブルから統計を収集する必要がある場合は、ログテーブル全体をスキャンするのではなく、サマリーテーブルを使用します。サマリーの管理は、「ライブ」で統計を計算しようとする場合よりはるかに高速になるはずです。状況が変わった (ビジネス上の決定に応じて) 場合、ログから新しいサマリーテーブルを再生成する方が、実行中のアプリケーションを変更するより早いです。
- 可能であれば、統計レポートに必要なデータが、ライブデータから定期的に生成されるサマリーテーブルからのみ作成される「ライブ」または「統計」として、レポートを分類します。
- カラムにデフォルト値があることを利用します。挿入する値がデフォルト値と異なる場合にのみ、明示的に値を挿入します。これにより、MySQL が実行する必要がある解析が減り、挿入速度が向上します。
- 状況によっては、データを BLOB カラムにパックし、格納すると便利です。この場合、情報をパックおよびアンパックするコードをアプリケーションに追加する必要がありますが、これにより、特定の段階で大量のアクセスを省略できます。これは、行とカラムのテーブル構造にうまく準拠していないデータがある場合に実用的です。
- 通常、すべてのデータを非冗長に維持しようとしてください (データベース理論で第 3 正規形と呼ばれるものを順守します)。ただし、高速化を図るために、情報を複製したり、サマリーテーブルを作成したりすることが有利になる状況もあります。
- ストアドルーチンや UDF (ユーザー定義関数) は特定のタスクでパフォーマンスの向上に適切な方法である場合があります。詳しくは、セクション20.2「ストアドルーチン (プロシージャーと関数) の使用」およびセクション24.3「MySQL への新しい関数の追加」を参照してください。
- アプリケーションでクエリーや応答をキャッシュしてから、多くの挿入や更新をまとめて実行することによって、パフォーマンスを向上できます。データベースシステムで MySQL のようにテーブルロックをサポートしている場合、これはすべての更新後にインデックスキャッシュが 1 回だけフラッシュされるようにするために役立つはずです。同様の結果を得るために、MySQL のクエリーキャッシュを利用することもできます。セクション8.9.3「MySQL クエリーキャッシュ」を参照してください。
- 1 つの SQL ステートメントで多数の行を格納するには、複数行の INSERT ステートメントを使用します。(これは比較的移植可能な技法です。)
- 大量のデータをロードするには LOAD DATA INFILE を使用します。これは INSERT ステートメントを使用するより高速になります。
- テーブルの各行を 1 つの一意の値で識別できるように、AUTO_INCREMENT カラムを使用します。
- ときどき OPTIMIZE TABLE を使用して、動的形式の MyISAM テーブルによる断片化を回避します。セクション15.2.3「MyISAM テーブルのストレージフォーマット」を参照してください。
- 可能であれば、MEMORY テーブルを使用して、高速化を図ります。セクション15.3「MEMORY ストレージエンジン」を参照してください。Web ブラウザで Cookie が有効にされていないユーザーに対して最後に表示されたバナーに関する情報など、頻繁にアクセスされる非クリティカルデータには MEMORY テーブルが役立ちます。ユーザーセッションも、揮発状態データを処理するために、多くの Web アプリケーション環境で使用できるもう 1 つの代替方法です。
- Web サーバーでは、イメージとその他のバイナリアセットが通常、ファイルとして格納されているはずです。つまり、データベース内にはファイル自体ではなく、ファイルへの参照のみを格納します。ほとんどの Web サーバーは、データベースコンテンツよりファイルのキャッシュに優れているため、ファイルの使用は一般に高速です。
- 対応するカラムに基づいた結合が速くなるように、異なるテーブル内の同一の情報を持つカラムは同一のデータ型を持つように宣言するべきです。
- カラム名が簡単になるようにします。たとえば、customer というテーブルでは customer_name ではなく name のカラム名を使用します。名前をほかの SQL サーバーに移植できるようにするため、18 文字より短くすることを考慮します。
- 実際に高速化が必要である場合、別の SQL サーバーがサポートするデータストレージの低レベルインタフェースを調べます。たとえば、MySQL MyISAM ストレージエンジンに直接アクセスすることによって、SQL イン

タフェースを使用する場合と比較して 2 倍から 5 倍の速度の向上が得られる可能性があります。これを実行可能にするには、データがアプリケーションと同じサーバー上にある必要があり、通常 1 プロセスのみからアクセスするようにしてください (外部ファイルロックは非常に遅いため)。これらの問題は、MySQL サーバーに低レベルの MyISAM コマンドを導入することで解消できます (これが、必要に応じてパフォーマンスを向上させる 1 つの簡単な方法になります)。データベースインタフェースを慎重に設計することで、この種類の最適化をきわめて簡単にサポートできるはずです。

- 数値データを使用している場合、多くの場合にテキストファイルにアクセスするより、ライブ接続を使用して、データベースから情報にアクセスする方が高速です。データベース内の情報はテキストファイルよりコンパクトなフォーマットで格納される可能性が高いため、それへのアクセスにかかわるディスクアクセスが少なくなります。さらに、テキストファイルを解析して、行とカラムの境界を見つける必要がないため、アプリケーション内のコードも節約できます。

- レプリケーションは、特定の操作でパフォーマンスの向上を実現できます。クライアントの取得をレプリケーションサーバー間で分散して、負荷を分割できます。バックアップを作成する間のマスターの速度低下を避けるため、スレーブサーバーを使用して、バックアップを作成できます。第17章「レプリケーション」を参照してください。

- DELAY_KEY_WRITE=1 テーブルオプションを使用して MyISAM テーブルを宣言すると、テーブルが閉じられるまで、ディスクにフラッシュされないため、インデックスの更新が速くなります。短所は、そのようなテーブルが開いている間に、何かによってサーバーが強制終了させられた場合に、--myisam-recover-options オプションを使用してサーバーを実行するか、サーバーを再起動する前に myisamchk を実行して、テーブルが問題ないことを確認する必要があることです。(ただし、この場合でも、キー情報は常にデータ行から生成できるため、DELAY_KEY_WRITE を使用しても何も失われないはずです。)

- SELECT ステートメントの優先度を挿入より高くしたい場合、サポートされる非トランザクションテーブルに、INSERT LOW_PRIORITY を使用します。

- キューに割り込んで先に取得されるようにするには、サポートされる非トランザクションテーブルに SELECT HIGH_PRIORITY を使用します。つまり、書き込みの実行を待機している別のクライアントがある場合でも、SELECT が実行されます。

  LOW_PRIORITY と HIGH_PRIORITY はテーブルレベルのロックのみを使用する非トランザクションストレージエンジンにのみ効果があります。

8.3 最適化とインデックス

SELECT 操作のパフォーマンスを向上する最善の方法は、クエリーでテストされる 1 つ以上のカラムにインデックスを作成することです。インデックスエントリは、テーブル行へのポインタのように動作し、クエリーが WHERE 句の条件に一致する行を迅速に特定し、それらの行のほかのカラム値を取得できます。すべての MySQL データ型にインデックスを設定できます。

クエリーで使用されている可能なすべてのカラムにインデックスを作成しようとしがちですが、不要なインデックスは領域を無駄にし、MySQL が使用するインデックスを判断するための時間を無駄にします。各インデックスを更新する必要があるため、インデックスは挿入、更新、削除のコストも追加します。最適なインデックスのセットを使用して、高速のクエリーを実現するために、適切なバランスを見つける必要があります。

8.3.1 MySQL のインデックスの使用の仕組み

インデックスは特定のカラム値のある行をすばやく見つけるために使用されます。インデックスがないと、MySQL は関連する行を見つけるために、先頭行から始めてテーブル全体を読み取る必要があります。テーブルが大きいほど、このコストが大きくなります。テーブルに問題のカラムのインデックスが含まれている場合、MySQL はすべてのデータを調べる必要なく、データファイルの途中のシークする位置をすばやく特定できます。これはすべての行を順次読み取るよりはるかに高速です。

ほとんどの MySQL インデックス (PRIMARY KEY、UNIQUE、INDEX、および FULLTEXT) は B ツリーに格納されます。例外: 空間データ型のインデックスは R ツリーを使用します。MEMORY テーブルはハッシュインデックスもサポートします。InnoDB は FULLTEXT インデックスの逆のリストを使用します。

一般に、インデックスは次の説明に示すように使われます。ハッシュインデックス (MEMORY テーブルで使用されているような) に固有の特性については、セクション8.3.8「B ツリーインデックスとハッシュインデックスの比較」で説明しています。

MySQL はこれらの操作にインデックスを使用します。

- WHERE 句に一致する行をすばやく見つけるため。

- 行を考慮に入れないようにするため。複数のインデックスから選択する場合、MySQL は通常最小数の行を見つけるインデックス (もっとも選択的なインデックス) を使用します。

- テーブルにマルチカラムインデックスがある場合、オプティマイザは、インデックスの左端のプリフィクスを使用して行をルックアップできます。たとえば、(col1, col2, col3) に 3 カラムのインデックスがある場合、(col1)、(col1, col2)、および (col1, col2, col3) に対して、インデックス検索機能を使用できます。詳細については、セクション8.3.5「マルチカラムインデックス」を参照してください。

- 結合の実行時に、ほかのテーブルから行を取得するため。カラムが同じ型とサイズで宣言されていると、MySQL はカラムのインデックスをより効率的に使用できます。このコンテキストでは、VARCHAR と CHAR は同じサイズで宣言されていれば同じとみなされます。たとえば、VARCHAR(10) と CHAR(10) は同じサイズですが、VARCHAR(10) と CHAR(15) は異なります。

  非バイナリ文字列カラム間での比較の場合、両方のカラムで同じ文字セットを使用しているべきです。たとえば、utf8 カラムと latin1 カラムの比較はインデックスの使用の可能性を否定します。

  異種のカラムの比較 (文字列カラムを時間または数値カラムと比較するなど) では、値を変換せずに直接比較できない場合、インデックスの使用が妨げられることがあります。数値カラム内の 1 などの特定の値の場合、'1'、' 1'、'00001'、または '01.e1' などの文字列カラム内の任意の数の値と等しくなる可能性があります。これは、文字列カラムのインデックスの使用を除外します。

- 特定のインデックス設定されたカラム key_col に対して、MIN() あるいは MAX() 値を見つけるため。これはインデックス内の key_col より前に発生するすべてのキーパートで、WHERE key_part_N = constant が使用されているかどうかをチェックするプリプロセッサによって最適化されます。この場合、MySQL は各 MIN() または MAX() 式に対して単一キールックアップを行い、それを定数で置き換えます。すべての式が定数で置き換えられた場合、クエリーは同時に返されます。例:

```
SELECT MIN(key_part2),MAX(key_part2)
  FROM tbl_name WHERE key_part1=10;
```

- 使用可能なインデックスの左端のプリフィクスに対してソートまたはグループ化が行われている場合 (たとえば、ORDER BY key_part1, key_part2) に、テーブルをソートまたはグループ化するため。すべてのキーパートのあとに DESC が付けられている場合、キーは逆の順序で読み取られます。セクション8.2.1.15「ORDER BY の最適化」およびセクション8.2.1.16「GROUP BY の最適化」を参照してください。

- 場合によって、データ行を参照しないで値を取得するように、クエリーを最適化できます。(クエリーの必要なすべての結果を提供するインデックスは、カバーするインデックスと呼ばれます。)クエリーがテーブルから特定のインデックスに含まれるカラムのみを使用している場合、きわめて高速に、選択した値をインデックスツリーから取得できます。

```
SELECT key_part3 FROM tbl_name
  WHERE key_part1=1
```

小さなテーブルまたは、レポートクエリーが行の大半またはすべてを処理する大きなテーブルに対するクエリーでは、インデックスはあまり重要ではありません。クエリーで行の大半にアクセスする必要がある場合は、順次読み取る方が、インデックスを処理するより高速です。クエリーですべての行が必要でない場合でも、順次読み取りは、ディスクシークを最小にします。詳細は、セクション8.2.1.20「フルテーブルスキャンを回避する方法」を参照してください。

8.3.2 主キーの使用

テーブルの主キーは、もっとも重要なクエリーで使用するカラムやカラムのセットを表します。それには、高速のクエリーパフォーマンスのため、インデックスが関連付けられます。それには NULL 値を含めることができないため、クエリーパフォーマンスは NOT NULL 最適化からメリットが得られます。InnoDB ストレージエンジンによって、テーブルデータが、主キーカラムに基づいて、超高速ルックアップおよびソートを実行するように物理的に編成されます。

テーブルが大きく、重要でも、主キーとして使用する明確なカラムやカラムのセットがない場合は、自動インクリメント値で個別のカラムを作成して、主キーとして使用できます。これらの一意の ID は、外部キーを使用してテーブルを結合する場合に、ほかのテーブル内の対応する行へのポインタとして使用できます。

8.3.3 外部キーの使用

テーブルに多くのカラムがあり、多くのさまざまなカラムの組み合わせをクエリーする場合、あまり頻繁に使用されないデータをそれぞれ少数のカラムを持つ個別のテーブルに分割し、それらを、メインテーブルの数値 ID カラムを複製してメインテーブルに関連付けると、効率的なことがあります。そのようにして、小さな各テーブル

に、そのデータの高速ルックアップのための主キーを設定でき、結合操作を使用して必要とするカラムのセットだけをクエリーできます。データの分散状況に応じて、関連カラムがディスク上にまとめてパックされるため、クエリーで実行する I/O が少なくなり、使用するキャッシュメモリーが減る可能性があります。(パフォーマンスを最大にするため、クエリーはディスクから可能なかぎり少ないデータブロックを読み取ろうとします。数個のカラムしかないテーブルでは各データブロックにより多くのデータを収めることができます。)

8.3.4 カラムインデックス

もっとも一般的なインデックスの種類には、単一カラムがあり、データ構造にそのカラムの値のコピーを格納し、対応するカラム値のある行を高速にルックアップできます。B ツリーデータ構造により、インデックスは、WHERE 句内の =、>、≤、BETWEEN、IN などの演算子に対応する特定の値、値のセット、または値の範囲をすばやく見つけることができます。

テーブルあたりの最大インデックス数とインデックスの最大長は、ストレージエンジンごとに定義されます。第15章「代替ストレージエンジン」を参照してください。すべてのストレージエンジンは、1 テーブルあたり 16 個以上のインデックスと 256 バイト以上の合計インデックス長をサポートします。ほとんどのストレージエンジンでは、制限が高く設定されています。

プリフィクスインデックス

インデックス指定でcol_name(N) 構文を使用して、文字列カラムの先頭の N 文字のみを使用するインデックスを作成できます。このようにカラム値のプリフィクスのみのインデックスを作成すると、インデックスファイルをかなり小さくできます。BLOB または TEXT カラムにインデックス設定する場合、インデックスのプリフィクス長を指定する必要があります。例:

```
CREATE TABLE test (blob_col BLOB, INDEX(blob_col(10)));
```

プリフィクスは最大 1000 バイト長 (innodb_large_prefix を設定していないかぎり、InnoDB テーブルの場合は 767 バイト) です。

> **注記**
>
> プリフィクスの制限はバイト単位で測定されますが、CREATE TABLE ステートメントでのプリフィクス長は文字数で解釈されます。複数バイト文字セットを使用するカラムのプリフィクス長を指定する場合はこれを考慮してください。

FULLTEXT インデックス

FULLTEXT インデックスの作成も可能です。これらは全文検索に使用されます。InnoDB および MyISAM ストレージエンジンのみが、CHAR、VARCHAR、および TEXT カラムに対してのみ、FULLTEXT インデックスをサポートしています。インデックス設定は常にカラム全体に対して行われ、カラムプリフィクスインデックス設定はサポートされていません。詳細については、セクション12.9「全文検索関数」を参照してください。

最適化は、単一の InnoDB テーブルに対する特定の種類の FULLTEXT クエリーに適用されます。これらの特性を持つクエリーは特に効率的です。

- ドキュメント ID またはドキュメント ID と検索ランクのみを返す FULLTEXT クエリー。
- 一致する行をスコアの降順でソートし、LIMIT 句を適用して、上位 N 個の一致する行を取得する FULLTEXT クエリー。この最適化を適用するには、WHERE 句がなく、降順の単一の ORDER BY 句のみがある必要があります。
- 検索語に一致する行の COUNT(*) 値のみを取得し、追加の WHERE 句がない FULLTEXT クエリー。WHERE 句を > 0 比較演算子を使用せずに、WHERE MATCH(text) AGAINST ('other_text') とコーディングします。

空間インデックス

空間データ型にインデックスを作成することもできます。現在、MyISAM のみが空間型への R ツリーインデックスをサポートしています。ほかのストレージエンジンは、空間型のインデックス設定に B ツリーを使用します (ARCHIVE を除きます。これは空間型のインデックス設定をサポートしていません)。

MEMORY ストレージエンジンでのインデックス

MEMORY ストレージエンジンはデフォルトで HASH インデックスを使用しますが、BTREE インデックスもサポートしています。

8.3.5 マルチカラムインデックス

MySQL は複合インデックス (つまり、複数のカラムに対するインデックス) を作成できます。インデックスは最大 16 カラムで構成できます。特定のデータ型では、カラムのプリフィクスにインデックスを設定できます (セクション8.3.4「カラムインデックス」を参照してください)。

MySQL では、インデックスで、すべてのカラムをテストするクエリーまたは最初のカラム、最初の 2 つのカラム、最初の 3 つのカラムというようにテストするクエリーにマルチカラムインデックスを使用できます。インデックス定義の正しい順序でカラムを指定する場合、単一の複合インデックスにより、同じテーブルへの複数の種類のクエリーを高速化できます。

マルチカラムインデックスは、インデックス設定されたカラムの値を連結して作成された値を格納する行である、ソート済みの配列とみなすことができます。

> **注記**
>
> 複合インデックスの代わりに、ほかのカラムの情報に基づいて「ハッシュされた」カラムを導入できます。このカラムが短く、十分に一意で、インデックスが設定されている場合は、多数のカラムへの「広範な」インデックスより速くなる可能性があります。MySQL では、この追加カラムをきわめて簡単に使用できます。
>
> ```
> SELECT * FROM tbl_name
>   WHERE hash_col=MD5(CONCAT(val1,val2))
>   AND col1=val1 AND col2=val2;
> ```

テーブルが次のような仕様であるとします。

```
CREATE TABLE test (
    id         INT NOT NULL,
    last_name  CHAR(30) NOT NULL,
    first_name CHAR(30) NOT NULL,
    PRIMARY KEY (id),
    INDEX name (last_name,first_name)
);
```

name インデックスは、last_name カラムと first_name カラムに対するインデックスです。このインデックスは、last_name 値と first_name 値の組み合わせに既知の範囲の値を指定するクエリーで、ルックアップに使用できます。そのカラムはインデックスの左端のプリフィクスであるため、last_name 値だけを指定するクエリーにも使用できます (このセクションで後述するように)。そのため、name インデックスは、次のクエリーでのルックアップに使用されます。

```
SELECT * FROM test WHERE last_name='Widenius';

SELECT * FROM test
  WHERE last_name='Widenius' AND first_name='Michael';

SELECT * FROM test
  WHERE last_name='Widenius'
  AND (first_name='Michael' OR first_name='Monty');

SELECT * FROM test
  WHERE last_name='Widenius'
  AND first_name >='M' AND first_name < 'N';
```

ただし、name インデックスは次のクエリーでのルックアップには使用されません。

```
SELECT * FROM test WHERE first_name='Michael';

SELECT * FROM test
  WHERE last_name='Widenius' OR first_name='Michael';
```

次の SELECT ステートメントを発行するとします。

```
SELECT * FROM tbl_name
  WHERE col1=val1 AND col2=val2;
```

col1 と col2 に対するマルチカラムインデックスが存在する場合、該当する行を直接フェッチできます。col1 および col2 に対して個別の単一カラムのインデックスが存在する場合、オプティマイザは、インデックスマージ最適化 (セクション8.2.1.4「インデックスマージの最適化」を参照してください) の使用を試みるか、またはより多くの行を除外するインデックスを判断して、そのインデックスを使用して行をフェッチすることで、もっとも制限の厳しいインデックスを見つけようとします。

テーブルにマルチカラムインデックスがある場合、オプティマイザは、インデックスの左端のプリフィクスを使用して行をルックアップできます。たとえば、(col1, col2, col3) に 3 カラムのインデックスがある場合、(col1)、(col1, col2)、および (col1, col2, col3) に対して、インデックス検索機能を使用できます。

カラムがインデックスの左端のプリフィクスを形成していない場合、MySQL はこのインデックスを使用してルックアップを実行できません。ここに示す SELECT ステートメントがあるとします。

```
SELECT * FROM tbl_name WHERE col1=val1;
SELECT * FROM tbl_name WHERE col1=val1 AND col2=val2;

SELECT * FROM tbl_name WHERE col2=val2;
SELECT * FROM tbl_name WHERE col2=val2 AND col3=val3;
```

(col1, col2, col3) にインデックスが存在する場合、最初の 2 つのクエリーだけがインデックスを使用します。3 つめと 4 つめのクエリーには、インデックス設定されたカラムがかかわりますが、(col2) と (col2, col3) は (col1, col2, col3) の左端のプリフィクスではありません。

8.3.6 インデックスの使用の確認

すべてのクエリーがテーブル内に作成したインデックスを実際に使用していることを常に確認します。セクション8.8.1「EXPLAIN によるクエリーの最適化」に説明するように、EXPLAIN ステートメントを使用します。

8.3.7 InnoDB および MyISAM インデックス統計コレクション

ストレージエンジンはオプティマイザによって使用されるテーブルに関する統計を収集します。テーブル統計は値グループに基づきますが、ここで値グループは同じキープリフィクス値を持つ行のセットです。オプティマイザの目的で、重要な統計は平均値グループサイズです。

MySQL は平均値グループサイズを次のように使用します。

- 各 ref アクセスごとに読み取る必要がある行数を見積もるため
- 部分結合で生成される行数、つまりこの形式の操作で生成される行数を見積もるため:

  ```
  (...) JOIN tbl_name ON tbl_name.key = expr
  ```

インデックスの平均値グループサイズが増えるほど、ルックアップあたりの平均行数が増えるため、それらの 2 つの目的でインデックスが役立たなくなります。インデックスが最適化の目的に役立つようにするには、各インデックス値でターゲットとするテーブル内の行を少なくすることがもっとも適切です。指定したインデックス値が多数の行を生成する場合、そのインデックスはあまり役に立たず、MySQL がそれを使用する可能性は少なくなります。

平均値グループサイズは、値グループの数であるテーブルカーディナリティーと関連しています。SHOW INDEX ステートメントは、N/S に基づいて、カーディナリティー値を表示します。ここで N はテーブル内の行数で、S は平均値グループサイズです。その比率から、テーブル内の値グループの概数がわかります。

<=> 比較演算子に基づいた結合では、NULL の扱いはほかの値と異なりません。ほかのどの N に対しても N <=> N とまったく同じように、NULL <=> NULL です。

ただし、= 演算子に基づく結合では、NULL は NULL 以外の値と異なります。expr1 または expr2 (または両方) が NULL である場合、expr1 = expr2 は true になりません。これは、形式 tbl_name.key = expr の比較の ref アクセスに影響を与えます。expr の現在値が NULL の場合、比較は true にならないため、MySQL はテーブルにアクセスしません。

= 比較では、テーブルにある NULL 値の数は問題になりません。最適化の目的で、関連のある値は NULL 以外の値グループの平均サイズです。ただし、MySQL では現在その平均サイズを収集したり、使用したりできません。

InnoDB および MyISAM テーブルでは、innodb_stats_method および myisam_stats_method システム変数をそれぞれ使用して、テーブル統計のコレクションに対していくらかの制御ができます。これらの変数には、3 つの可能性のある値を使用でき、次のように異なります。

- 変数が nulls_equal に設定されている場合、すべての NULL 値が同一として扱われます (つまり、それらすべてが単一の値グループを形成します)。

  NULL 値グループサイズが、NULL 以外の値グループサイズよりはるかに大きい場合、このメソッドは平均値グループサイズを上方に歪めます。これにより、オプティマイザには、NULL 以外の値を検索する結合に対して、インデックスが実際以上に役に立たないかのように見えます。結果として、nulls_equal メソッドにより、オプティマイザに ref アクセスに対してインデックスを使用すべきときでも使用させないようにすることがあります。

- 変数が nulls_unequal に設定されている場合、NULL 値は同じとみなされません。代わりに、各 NULL 値はサイズ 1 の個別の値グループを形成します。

  多くの NULL 値がある場合、このメソッドは平均値グループサイズを下方に歪めます。NULL 以外の平均値グループサイズが大きい場合、NULL 値をサイズ 1 のグループとしてカウントすると、オプティマイザは NULL 以外の値を検索する結合に対して、インデックスの値を多く見積もりすぎます。結果として、nulls_unequal メソッドによって、ほかのメソッドの方が適している可能性がある場合に、オプティマイザに ref ルックアップに対してこのインデックスを使用させることがあります。

- 変数が nulls_ignored に設定されている場合、NULL 値は無視されます。

= より <=> を使用する多くの結合を使用する傾向がある場合、比較で NULL 値は特別ではなく、NULL は互いに等しくなります。この場合、nulls_equal は適切な統計メソッドです。

innodb_stats_method システム変数にはグローバル値があります。myisam_stats_method システム変数にはグローバル値とセッション値の両方があります。グローバル値を設定すると、対応するストレージエンジンからのテーブルの統計収集に影響します。セッション値を設定すると、現在のクライアント接続のみに対する統計収集に影響します。これは、myisam_stats_method のセッション値を設定することで、ほかのクライアントに影響を与えずに、指定したメソッドで、テーブルの統計を強制的に再生成させることができることを意味します。

テーブル統計を再生成するには、次のいずれかのメソッドを使用できます。

- myisamchk --stats_method=method_name --analyze を実行します
- テーブルを変更して、統計を古くさせ (たとえば、行を挿入してから削除します)、次に myisam_stats_method を設定して、ANALYZE TABLE ステートメントを発行します。

innodb_stats_method と myisam_stats_method の使用に関するいくつかの警告は次のとおりです。

- 先述したように、テーブル統計を明示的に収集させることができます。ただし、MySQL は統計を自動的に収集することもあります。たとえば、テーブルへのステートメントの実行の途中で、そうしたステートメントの中にはテーブルを変更するものもあり、MySQL は統計を収集する場合があります。(たとえば、これは一括挿入や削除、または一部の ALTER TABLE ステートメントで行われることがあります。)これが行われた場合、その時点での innodb_stats_method または myisam_stats_method の値を使用して、統計が収集されます。そのため、あるメソッドを使用して統計を収集しても、あとでテーブルの統計が自動的に収集されたときに、システム変数にほかのメソッドが設定されていると、そのほかのメソッドが使われます。
- 特定のテーブルの統計の生成に使用されたメソッドを伝える方法はありません。
- これらの変数は InnoDB および MyISAM テーブルにのみ適用されます。ほかのストレージエンジンはテーブル統計を収集するメソッドが 1 つしかありません。通常、それは nulls_equal メソッドに近いものになります。

8.3.8 B ツリーインデックスとハッシュインデックスの比較

B ツリーおよびハッシュデータ構造を理解することは、インデックスにこれらのデータ構造を使用するさまざまなストレージエンジンで (特に B ツリーインデックスを使用するか、ハッシュインデックスを使用するかを選択できる MEMORY ストレージエンジンの場合に)、さまざまなクエリーがどのように実行されるかを予測するのに役立つ可能性があります。

B ツリーインデックスの特性

B ツリーインデックスは =、>、>=、<、<=、または BETWEEN 演算子を使用する式で、カラム比較に使用できます。このインデックスは、LIKE への引数がワイルドカード文字で始まらない定数文字列の場合の LIKE 比較にも使用できます。たとえば、次の SELECT ステートメントはインデックスを使用します。

```
SELECT * FROM tbl_name WHERE key_col LIKE 'Patrick%';
SELECT * FROM tbl_name WHERE key_col LIKE 'Pat%_ck%';
```

最初のステートメントでは、'Patrick' <= key_col < 'Patricl' の行のみが考慮されます。2 つめのステートメントでは、'Pat' <= key_col < 'Pau' の行のみが考慮されます。

次の SELECT ステートメントはインデックスを使用しません。

```
SELECT * FROM tbl_name WHERE key_col LIKE '%Patrick%';
SELECT * FROM tbl_name WHERE key_col LIKE other_col;
```

最初のステートメントでは、LIKE 値はワイルドカード文字で始まります。2 つめのステートメントでは、LIKE 値は定数ではありません。

... LIKE '%string%' を使用し、string が 3 文字より長い場合、MySQL は Turbo Boyer-Moore アルゴリズムを使用して、文字列のパターンを初期化してから、このパターンを使用して検索をより迅速に実行します。

col_name IS NULL を使用した検索では、col_name にインデックスが設定されている場合にインデックスが使用されます。

WHERE 句内のすべての AND レベルにまたがっていないインデックスは、クエリーの最適化に使用されません。言い換えると、インデックスの使用を可能にするには、インデックスのプリフィクスがすべての AND グループで使用されている必要があります。

次の WHERE 句ではインデックスが使用されます。

```
... WHERE index_part1=1 AND index_part2=2 AND other_column=3

   /* index = 1 OR index = 2 */
... WHERE index=1 OR A=10 AND index=2

   /* optimized like "index_part1='hello'" */
... WHERE index_part1='hello' AND index_part3=5

   /* Can use index on index1 but not on index2 or index3 */
... WHERE index1=1 AND index2=2 OR index1=3 AND index3=3;
```

これらの WHERE 句ではインデックスが使用されません。

```
   /* index_part1 is not used */
... WHERE index_part2=1 AND index_part3=2

   /*  Index is not used in both parts of the WHERE clause  */
... WHERE index=1 OR A=10

   /* No index spans all rows  */
... WHERE index_part1=1 OR index_part2=10
```

MySQL ではインデックスが使用できる場合でも使用しないことがあります。これが発生する 1 つの状況は、オプティマイザが、インデックスを使用することによって MySQL がテーブルの大部分の行にアクセスする必要があると推定した場合です。(この場合、必要なシークが少ないため、テーブルスキャンの方がはるかに高速になる可能性があります。) ただし、そのようなクエリーで、行の一部のみを取得する LIMIT を使用している場合、結果で返す少数の行をはるかにすばやく見つけることができるため、MySQL はとにかくインデックスを使用します。

ハッシュインデックスの特性

ハッシュインデックスは先述したものといくらか異なる特性を持ちます。

- それらは、= または <=> 演算子を使用する等価比較にのみ使用されます (ただしきわめて高速です)。それらは、値の範囲を見つける < などの比較演算子には使用されません。この種類の単一値ルックアップに依存するシステムは、「キー値ストア」として知られています。そのようなアプリケーションで MySQL を使用するには、可能なかぎりハッシュインデックスを使用します。
- オプティマイザはハッシュインデックスを使用して、ORDER BY 操作を高速化することはできません。(この種類のインデックスは順番に次のエントリを検索するために使用できません。)
- MySQL は 2 つの値の間におよそどのくらいの行数があるかを判断できません (これは範囲オプティマイザによって使用するインデックスを特定するために使用されます)。これは、MyISAM または InnoDB テーブルをハッシュインデックス設定された MEMORY テーブルに変更した場合に、一部のクエリーに影響することがあります。
- 行の検索にはキー全体のみを使用できます。(B ツリーインデックスでは、キーの任意の左端のプリフィクスを使用して行を検索できます。)

8.4 データベース構造の最適化

データベース設計者としての役割では、スキーマ、テーブル、およびカラムを編成するもっとも効率的な方法を探します。アプリケーションコードをチューニングする場合、I/O を最小にし、関連項目をまとめて、データボリュームが増加してもパフォーマンスを高く維持するように、事前に計画します。効率的なデータベース設計から始めることで、チームメンバーは高性能のアプリケーションコードを簡単に書けるようになり、アプリケーションが発展して、書き換えられても、データベースを持ちこたえさせる可能性が高くなります。

8.4.1 データサイズの最適化

ディスク上の領域を最小にするようにテーブルを設計します。これにより、ディスクに対して読み取りおよび書き込みされるデータの量が減ることで、大幅な改善が見られます。内容がクエリー実行中にアクティブに処理される間、テーブルが小さいほど、通常必要なメインメモリーの量は少なくなります。テーブルデータの領域の削減により、インデックスも小さくなり、高速に処理できます。

MySQL は多数のさまざまなストレージエンジン (テーブル型) と行フォーマットをサポートしています。テーブルごとに、使用するストレージとインデックス設定方法を決定できます。アプリケーションに適切なテーブル形式を選択することで、大幅なパフォーマンスの向上が得られることがあります。第15章「代替ストレージエンジン」を参照してください。

ここで挙げられた技法を使用して、テーブルのパフォーマンス改善とストレージ領域の最小化を図ることができます。

テーブルカラム

- 可能なかぎりもっとも効率的 (最小) のデータ型を使用します。MySQL にはディスク領域とメモリーを節約する多くの専用の型があります。たとえば、可能な場合は、小さなテーブルを取得するために、小さな整数型を使用します。MEDIUMINT カラムが使用する領域は 25% 少ないため、MEDIUMINT は多くの場合に INT より適切な選択肢です。
- 可能な場合は、カラムを NOT NULL として宣言します。それにより、インデックスを適切に使用し、各値が NULL であるかどうかをテストするためのオーバーヘッドがなくなることで、SQL の操作が速くなります。カラムあたり 1 ビットでいくらかのストレージ領域も節約します。テーブルで実際に NULL 値が必要な場合、それらを使用します。単にすべてのカラムで NULL 値を許可するデフォルトの設定を避けます。

行フォーマット

- InnoDB テーブルはコンパクトストレージフォーマットを使用します。5.0.3 より前の MySQL のバージョンでは、InnoDB の行に、固定サイズのカラムでも、カラム数や各カラムの長さなど、いくつかの冗長な情報が含まれます。デフォルトで、テーブルはコンパクト形式 (ROW_FORMAT=COMPACT) で作成されます。MySQL の古いバージョンにダウングレードしたい場合、ROW_FORMAT=REDUNDANT で古い形式を要求できます。

  コンパクト行フォーマットの存在により、行のストレージ領域が約 20% 減少しますが、一部の操作で CPU の使用が増加する犠牲を伴います。ワークロードが、キャッシュヒット率とディスク速度によって制限される通常のワークロードであれば、速くなる可能性があります。CPU 速度によって制限されるまれな例では、遅くなることがあります。

  コンパクト InnoDB 形式では UTF-8 データを含む CHAR カラムが格納される方法も変わります。UTF-8 エンコード文字の最大長が 3 バイトであるとして、ROW_FORMAT=REDUNDANT では、UTF-8 CHAR(N) は 3 × N バイトを占有します。多くの言語は主にシングルバイト UTF-8 文字を使用して書くことができるため、固定のストレージ長は多くの場合に領域を無駄にします。ROW_FORMAT=COMPACT 形式では、InnoDB は N から 3 × N バイトの範囲のストレージの可変容量を、必要に応じて末尾のスペースを取り除いて、これらのカラムに割り当てます。最小のストレージ長は、一般的な場合にインプレース更新を容易にする N バイトとして保持されます。
- テーブルデータを圧縮形式で保存することで、さらに領域を最小にするには、InnoDB テーブルを作成する際に ROW_FORMAT=COMPRESSED を指定するか、既存の MyISAM テーブルに対して、myisampack コマンドを実行します。(InnoDB 圧縮テーブルは読み取り可能で書き込み可能ですが、MyISAM 圧縮テーブルは読み取り専用です。)
- MyISAM テーブルで、可変長カラム (VARCHAR、TEXT、あるいは BLOB など) がない場合は、固定サイズ行フォーマットが使用されます。これは高速ですが、いくらか領域を無駄にすることがあります。セクション15.2.3「MyISAM テーブルのストレージフォーマット」を参照してください。CREATE TABLE オプション ROW_FORMAT=FIXED によって、VARCHAR カラムがある場合でも、固定長の行を必要としていることを伝えることができます。

インデックス

- テーブルのプライマリインデックスは可能なかぎり短くしてください。これにより、各行の識別が容易になり効率的になります。InnoDB テーブルの場合、主キーカラムは、各セカンダリインデックスエントリに複製されるため、多数のセカンダリインデックスがある場合に、短い主キーによって、かなりの領域が節約されます。
- クエリーパフォーマンスを向上するために必要なインデックスのみを作成します。インデックスは取得には有効ですが、挿入および更新操作を遅くします。ほとんどカラムの組み合わせに対して検索することによって、テーブルにアクセスする場合、カラムごとに個別のインデックスを作成するのではなく、それらに対して単一の複合インデックスを作成します。インデックスの最初の部分は、もっとも使用されるカラムにするべきで

す。テーブルから選択する場合に、常に多くのカラムを使用する場合、適切なインデックスの圧縮を取得するため、インデックスの最初のカラムは、もっとも重複の多いカラムにするべきです。

- 長い文字列カラムで、最初の数文字に一意のプリフィクスがある可能性が高い場合、MySQL のカラムの左端の部分へのインデックスの作成のサポート (セクション13.1.13「CREATE INDEX 構文」を参照してください) を使用して、このプリフィクスのみにインデックスを設定することをお勧めします。短いインデックスほど速くなるのは、必要なディスク領域が少ないだけでなく、インデックスキャッシュでのヒットが多くなり、そのためにディスクシークが少なくなるためでもあります。セクション8.11.2「サーバーパラメータのチューニング」を参照してください。

結合

- 状況によって、頻繁にスキャンされるテーブルを 2 つに分割することで、メリットがある場合があります。これは特に、それが動的形式テーブルで、テーブルのスキャン時に、関連行を見つけるために使用できる小さな静的形式テーブルを使用できる場合に当てはまります。
- 対応するカラムに基づいた結合を高速化するには、異なるテーブル内の同一の情報を持つカラムを同一のデータ型で宣言します。
- 異なるテーブルで同じ名前を使用し、結合クエリーを簡略化できるように、カラム名を簡単にします。たとえば、customer というテーブルでは customer_name ではなく name のカラム名を使用します。名前をほかの SQL サーバーに移植できるようにするため、18 文字より短くすることを考慮します。

正規化

- 通常、すべてのデータを非冗長に維持しようとしてください (データベース理論で第 3 正規形と呼ばれるものを順守します)。名前や住所などの長い値を繰り返す代わりに、それらに一意の ID を割り当て、複数の小さなテーブルで必要なだけこれらの ID を繰り返し、結合句で ID を参照して、クエリーでテーブルを結合します。
- たとえば、大きなテーブルからすべてのデータを解析するビジネスインテリジェンスシナリオなどで、ディスク領域やデータの複数のコピーを維持する保守コストより、速度の方が重要である場合、正規化ルールを緩和して、情報を複製したり、サマリーテーブルを作成したりして、速度を向上させることができます。

8.4.2 MySQL データ型の最適化

8.4.2.1 数値データの最適化

- 文字列または数値として表すことができる一意の ID やその他の値の場合、文字列カラムより数値カラムをお勧めします。大きな数値は、対応する文字列より少ないバイト数で格納できるため、それらの転送と比較が高速になり、使用するメモリーが少なくなります。
- 数値データを使用している場合、多くの場合にテキストファイルにアクセスするより、ライブ接続を使用して、データベースから情報にアクセスする方が高速です。データベース内の情報はテキストファイルよりコンパクトなフォーマットで格納される可能性が高いため、それへのアクセスにかかわるディスクアクセスが少なくなります。また、テキストファイルを解析して、行とカラムの境界を見つけることを回避できるため、アプリケーションのコードも節約できます。

8.4.2.2 文字および文字列型の最適化

文字および文字列カラムの場合、次のガイドラインに従います。

- 言語固有の照合機能が必要でない場合は、比較およびソート操作を速くするため、バイナリ照合順序を使用します。特定のクエリー内でバイナリ照合順序を使用するには、BINARY 演算子を使用できます。
- さまざまなカラムの値を比較する場合、可能なかぎり、それらのカラムを同じ文字セットと照合順序で宣言し、クエリー実行中の文字列変換を避けます。
- サイズが 8K バイト未満のカラム値では、BLOB の代わりにバイナリ VARCHAR を使用します。GROUP BY および ORDER BY 句は一時テーブルを生成する可能性があり、これらの一時テーブルでは、元のテーブルに BLOB カラムが含まれない場合に、MEMORY ストレージエンジンを使用することがあります。
- テーブルに名前や住所などの文字列カラムが含まれるが、多くのクエリーでそれらのカラムを取得しない場合、文字列カラムを個別のテーブルに分割し、必要に応じて、外部キーで結合クエリーを使用することを考慮します。MySQL で行から何らかの値を取得する場合、その行 (およびおそらくその他の隣接する行) のすべて

のカラムを含むデータブロックを読み取ります。もっとも頻繁に使用するカラムのみで、各行を小さくすることで、より多くの行を各データブロックに収めることができます。そのようなコンパクトなテーブルは、一般的なクエリーのディスク I/O やメモリーの使用量を削減します。

- InnoDB テーブルで主キーとして、ランダムに生成された値を使用する場合、可能であれば、現在の日時などの降順の値でプリフィクスを付けます。連続したプライマリ値が、相互に物理的に近くに保存されていれば、InnoDB はそれらを高速に挿入し、取得できます。
- 数値カラムの方が通常同等の文字列カラムより推奨される理由については、セクション8.4.2.1「数値データの最適化」を参照してください。

8.4.2.3 BLOB 型の最適化

- テキストデータを格納する大きな BLOB を保存する場合、まずそれを圧縮することを考慮します。テーブル全体が InnoDB または MyISAM によって圧縮されている場合は、この技法を使用しないでください。
- 複数のカラムのあるテーブルで、BLOB カラムを使用しないクエリーのメモリー要件を削減するには、BLOB カラムを個別のテーブルに分割し、必要に応じて、結合クエリーでそれを参照することを考慮します。
- BLOB 値を取得し、表示するためのパフォーマンス要件は、ほかのデータ型と大きく異なることがあるため、BLOB 固有テーブルを別のストレージデバイスまたは個別のデータベースインスタンスに置くことができます。たとえば、BLOB を取得するには、大量の順次ディスク読み取りが必要で、SSD デバイスより、従来のハードドライブの方が適しています。
- バイナリ VARCHAR カラムの方が同等の BLOB カラムより推奨されることがある理由については、セクション8.4.2.2「文字および文字列型の最適化」を参照してください。
- きわめて長いテキスト文字列に対して、同等性をテストする代わりに、個別のカラムにカラムのハッシュを格納し、そのカラムにインデックスを設定して、クエリー内のハッシュ値をテストします。(MD5() または CRC32() 関数を使用して、ハッシュ値を生成します。)ハッシュ関数は、異なる入力で重複した結果を生成することがあるため、引き続きクエリーに句 AND blob_column = long_string_value を含めて、誤った一致に対して保護します。パフォーマンスは、ハッシュ値の小さく、簡単にスキャンされるインデックスからメリットが得られます。

8.4.2.4 PROCEDURE ANALYSE の使用

ANALYSE([max_elements[,max_memory]])

ANALYSE() はクエリーからの結果を調査し、テーブルサイズの削減に役立つ可能性がある各カラムの最適なデータ型を提案する結果の分析を返します。この分析を取得するには、SELECT ステートメントの末尾に PROCEDURE ANALYSE を追加します。

```
SELECT ... FROM ... WHERE ... PROCEDURE ANALYSE([max_elements,[max_memory]])
```

例:

```
SELECT col1, col2 FROM table1 PROCEDURE ANALYSE(10, 2000);
```

結果には、クエリーによって返された値のいくつかの統計が表示され、カラムの最適なデータ型が提案されます。これは、既存のテーブルのチェックや新しいデータのインポート後に役立つことがあります。ENUM データ型が適切でない場合に、PROCEDURE ANALYSE() がそれを提案しないように、引数の異なる設定を試してみる必要がある場合があります。

引数はオプションで次のように使用します。

- max_elements (デフォルト 256) は、ANALYSE() がカラムあたりに認識する個々の値の最大数です。これは、ANALYSE() によって、最適なデータ型が型 ENUM であるかどうかをチェックするために使用されます。max_elements 個を超える個別の値がある場合、ENUM は提案される型ではありません。
- max_memory (デフォルト 8192) は ANALYSE() がすべての個別の値を見つけようとする間に、カラムごとに割り当てるべき最大メモリー量です。

8.4.3 多数のテーブルの最適化

各クエリーを高速にするいくつかの技法には、多数のテーブルへのデータの分割が含まれます。テーブルの数が数千または数百万にもなる場合、これらすべてのテーブルの処理のオーバーヘッドは新たなパフォーマンスの考慮事項になります。

8.4.3.1 MySQL でのテーブルのオープンとクローズの方法

mysqladmin status コマンドを実行すると、次のように表示されるはずです。

```
Uptime: 426 Running threads: 1 Questions: 11082
Reloads: 1 Open tables: 12
```

テーブルが 6 つしかない場合に、12 の Open tables 値はいくぶん不可解に思うことがあります。

MySQL はマルチスレッド対応であるため、特定のテーブルに対して多くのクライアントが同時にクエリーを発行している場合があります。同じテーブルに対して、複数のクライアントセッションが異なる状態を持つ問題を最小にするため、テーブルは各同時セッションに独立して開かれます。これは追加メモリーを使用しますが、一般にパフォーマンスは向上します。MyISAM テーブルでは、テーブルを開いているクライアントごとに、データファイルに 1 つの追加のファイルディスクリプタが必要になります。(対照的に、インデックスファイルディスクリプタはすべてのセッションで共有されます。)

table_open_cache および max_connections システム変数は、サーバーが開いたままにするファイルの最大数に影響します。これらの値のいずれかまたは両方を増やすと、オープンファイルディスクリプタのプロセスあたりの数に関して、オペレーティングシステムによって適用されている制限に達する可能性があります。多くのオペレーティングシステムでは、オープンファイル制限を増やすことができますが、方法はシステムによって大きく異なります。制限値を増やすことができるかどうか、およびその実行方法については、使用するオペレーティングシステムのドキュメントを参照してください。

table_open_cache は max_connections に関連します。たとえば、200 の同時実行接続の場合、少なくとも 200 * N のテーブルキャッシュサイズを指定します。ここで N は実行するクエリーの結合あたりのテーブルの最大数です。また、一時テーブルとファイル用のいくつかの追加のファイルディスクリプタを予約する必要もあります。

オペレーティングシステムで、table_open_cache の設定に示されたオープンファイルディスクリプタの数を処理できることを確認してください。table_open_cache の設定が大きすぎると、MySQL がファイルディスクリプタを使い果たして接続を拒否し、クエリーの実行に失敗して、信頼性が大幅に低下します。また、MyISAM ストレージエンジンでは一意のオープンテーブルごとに 2 つのファイルディスクリプタが必要であることも考慮に入れる必要があります。mysqld に --open-files-limit スタートアップオプションを使用すると、MySQL で使用可能なファイルディスクリプタの数を増やすことができます。セクションB.5.2.18「'File' が見つかりません、および同様のエラー」を参照してください。

オープンテーブルのキャッシュは、table_open_cache エントリのレベルで保持されます。サーバーはスタートアップ時にキャッシュサイズを自動サイズ設定します。サイズを明示的に設定するには、スタートアップ時に table_open_cache システム変数を設定します。MySQL は、クエリーを実行するために、一時的にこれより多くのテーブルを開くことがあります。

次の状況では、MySQL は未使用のテーブルを閉じ、それをテーブルキャッシュから削除します。

- キャッシュがいっぱいで、スレッドがキャッシュにないテーブルを開こうとした場合。
- キャッシュに table_open_cache を超えるエントリがあり、キャッシュ内のテーブルがどのスレッドによっても使用されなくなった場合。
- テーブルフラッシュ操作が行われた場合。これは、だれかが FLUSH TABLES ステートメントを発行するか、または mysqladmin flush-tables または mysqladmin refresh コマンドを実行した場合に行われます。

テーブルキャッシュがいっぱいになると、サーバーは次の手順に従って使用するキャッシュエントリを見つけます。

- 現在使用中でないテーブルは、もっとも長く使用されていないテーブルから、解放されます。
- 新しいテーブルを開く必要があるが、キャッシュがいっぱいで、解放できるテーブルがない場合、必要に応じてキャッシュが一時的に拡張されます。キャッシュが一時的に拡張された状況で、テーブルが使用中から未使用状態になったときは、そのテーブルが閉じられ、キャッシュから解放されます。

MyISAM テーブルは同時アクセスごとに開かれます。つまり、2 つのスレッドで同じテーブルにアクセスする場合、または 1 つのスレッドが同一クエリーでテーブルに 2 回アクセスする場合 (テーブルをそれ自体に結合することによってなど) は、テーブルを 2 回開く必要があることを意味します。同時オープンは、それぞれテーブルキャッシュにエントリが必要になります。いずれかの MyISAM テーブルを最初に開くと、データファイルに 1 つとインデックスファイルに 1 つの 2 つのファイルディスクリプタが必要になります。テーブルの追加の使用では、それぞれデータファイルに 1 つだけのファイルディスクリプタが必要です。インデックスファイルディスクリプタはすべてのスレッドで共有されます。

HANDLER tbl_name OPEN ステートメントを使用してテーブルを開く場合、専用のテーブルオブジェクトがスレッドに割り当てられます。このテーブルオブジェクトはほかのスレッドと共有されず、スレッドが HANDLER tbl_name CLOSE を呼び出すか、スレッドが終了するまでクローズされません。これが発生すると、テーブルがテーブルキャッシュに戻されます (キャッシュがいっぱいでない場合)。セクション13.2.4「HANDLER 構文」を参照してください。

テーブルキャッシュが小さすぎるかどうかは、mysqld のステータス変数 Opened_tables をチェックして判断できます。これは、サーバーの起動以降のテーブルを開く操作の数を示します。

```
mysql> SHOW GLOBAL STATUS LIKE 'Opened_tables';
+---------------+-------+
| Variable_name | Value |
+---------------+-------+
| Opened_tables | 2741  |
+---------------+-------+
```

多くの FLUSH TABLES ステートメントを発行していない場合でも、値がきわめて大きいか、急増した場合、テーブルキャッシュサイズを増やします。セクション5.1.4「サーバーシステム変数」およびセクション5.1.6「サーバーステータス変数」を参照してください。

8.4.3.2 同じデータベースに大量のテーブルを作成することの短所

同じデータベースディレクトリに多数の MyISAM テーブルがある場合、オープン、クローズ、および作成操作が遅くなります。多数のテーブルに対して SELECT ステートメントを実行した場合、開く必要があるテーブルごとに、別のテーブルを閉じる必要があるため、テーブルキャッシュがいっぱいの場合にオーバーヘッドが少し発生します。テーブルキャッシュで許可されるエントリ数を増やすことによって、このオーバーヘッドを減らすことができます。

8.4.4 MySQL が内部一時テーブルを使用する仕組み

場合により、サーバーはクエリーの処理中に内部一時テーブルを作成します。それらのテーブルは、メモリー内に保持して、MEMORY ストレージエンジンによって処理したり、ディスク上に格納して、MyISAM ストレージエンジンによって処理したりできます。サーバーは最初にインメモリーテーブルとして内部で一時テーブルを作成し、それが大きくなりすぎた場合に、それをディスク上テーブルに変換することがあります。サーバーが内部一時テーブルを作成するタイミングや、サーバーがそれを管理するためにどのストレージエンジンを使用するかに関して、ユーザーは直接制御できません。

一時テーブルは、次のような条件で作成される可能性があります。

- UNION クエリーが一時テーブルを使用します。
- TEMPTABLE アルゴリズムを使用して評価されるものや、UNION またはアグリゲーションを使用するものなど、一部のビューで一時テーブルを必要とします。
- ORDER BY 句と別の GROUP BY 句がある場合、または、ORDER BY または GROUP BY に結合キュー内の最初のテーブルと異なるテーブルのカラムが含まれている場合は、一時テーブルが作成されます。
- DISTINCT と ORDER BY の組み合わせで、一時テーブルが必要になることがあります。
- SQL_SMALL_RESULT オプションを使用すると、MySQL では、クエリーにディスク上ストレージを必要とする要素 (後述) も含まれていないかぎり、インメモリー一時テーブルが使用されます。
- 複数テーブル UPDATE ステートメント。
- GROUP_CONCAT() または COUNT(DISTINCT) 評価。
- 派生テーブル (FROM 句内のサブクエリー)。
- サブクエリーまたは準結合実体化のために作成されるテーブル。

クエリーで一時テーブルを必要とするかどうかを判断するには、EXPLAIN を使用し、Extra カラムをチェックして、そこに Using temporary と示されているかどうかを確認します (セクション8.8.1「EXPLAIN によるクエリーの最適化」を参照してください)。EXPLAIN では、派生されるか、実体化された一時テーブルに対して、必ずしも Using temporary と表示しないことがあります。

内部一時テーブルが最初にインメモリーテーブルとして作成されたが、これが大きくなりすぎた場合、MySQL はこれを自動的にディスク上のテーブルに変換します。インメモリー一時テーブルの最大サイズは、tmp_table_size と max_heap_table_size の最小値です。これは、CREATE TABLE によって明示的に作成された MEMORY テー

ブルと異なります。そのようなテーブルの場合、max_heap_table_size システム変数でのみ、テーブルがどのくらい拡大でき、ディスク上フォーマットへの変換がないことが判断されます。

サーバーは内部一時テーブル (メモリー内またはディスク上のいずれか) を作成すると、Created_tmp_tables ステータス変数を増分します。サーバーはディスク上にテーブルを作成する (内部で、またはインメモリーテーブルを変換して) 場合、Created_tmp_disk_tables ステータス変数を増分します。

状況によっては、インメモリー一時テーブルの使用が妨げられる場合があり、その場合サーバーは代わりにディスク上テーブルを使用します。

- テーブル内の BLOB または TEXT カラムの存在
- GROUP BY または DISTINCT 句内の、バイナリ文字列の場合に 512 バイトまたは非バイナリ文字列の場合に 512 文字より大きい文字列カラムの存在。(MySQL 5.6.15 より前のこの制限は、文字列の型に関係なく 512 バイトです。)
- UNION または UNION ALL が使用された場合に、SELECT リスト内の 512 (バイナリ文字列の場合はバイト数、非バイナリ文字列の場合は文字数) より大きい最大長を持つ文字列カラムの存在。

8.5 InnoDB テーブルの最適化

InnoDB は、MySQL のお客様が一般に、信頼性と並列性が重要である本番環境のデータベースで使用するストレージエンジンです。InnoDB は MySQL 5.5 以上のデフォルトのストレージエンジンであるため、以前より頻繁に InnoDB テーブルを目にすることが予想されます。このセクションでは、InnoDB テーブルに対するデータベース操作を最適化する方法について説明します。

8.5.1 InnoDB テーブルのストレージレイアウトの最適化

- データが安定したサイズに達するか、拡大しているテーブルが数十または数百メガバイト単位で増大した場合、OPTIMIZE TABLE ステートメントを使用して、テーブルを再編成し、無駄なスペースを圧縮することを考慮します。再編成されたテーブルでは、フルテーブルスキャンを実行するために必要なディスク I/O が減ります。これは、インデックスの使用の改善やアプリケーションコードのチューニングなどのほかの技法が現実的でない場合に、パフォーマンスを向上できる直接的な技法です。

  OPTIMIZE TABLE はテーブルのデータ部分をコピーし、インデックスを再構築します。インデックス内へのデータのパックの改善とテーブルスペース内およびディスク上の断片化の削減からメリットが得られます。このメリットは各テーブル内のデータによって異なります。利点が大きいものとそうでないものがあること、またはテーブルの次回の最適化まで、時間の経過とともに利点が減っていくことに気付く場合があります。この操作は、テーブルが大きい場合や再構築されるインデックスがバッファープールに収まらない場合に、遅くなることがあります。大量のデータをテーブルに追加したあとの最初の実行では、多くの場合にその後の実行よりかなり遅くなります。

- InnoDB では、長い PRIMARY KEY (長い値を持つ単一カラムまたは長い複合値を形成する複数のカラムのいずれか) があると、大量のディスク領域を無駄にします。行の主キー値は、同じ行を指すすべてのセカンダリインデックスレコードに複製されます。(セクション14.2.13「InnoDB テーブルおよびインデックスの構造」を参照してください。)主キーが長い場合、AUTO_INCREMENT カラムを主キーとして作成するか、カラム全体ではなく、長い VARCHAR カラムのプリフィクスをインデックス設定します。

- 可変長の文字列を格納するために、または多くの NULL 値を持つカラムに対して、CHAR の代わりに VARCHAR データ型を使用します。CHAR(N) カラムは、文字列が短いか、その値が NULL だとしても、データを格納するために常に N 文字を必要とします。テーブルが小さいほどバッファープールに収まりやすく、ディスク I/O が減ります。

  COMPACT 行フォーマット (MySQL 5.6 でのデフォルトの InnoDB フォーマット) と utf8 や sjis などの可変長文字セットを使用する場合、CHAR(N) カラムは可変量でも、やはり N バイト以上の領域を占有します。

- 大きいか、繰り返しの多い大量のテキストや数値データを格納するテーブルでは、COMPRESSED 行フォーマットを使用することを考慮します。データをバッファープールに入れたり、フルテーブルスキャンを実行したりするために必要なディスク I/O が減ります。永続的な決断を下す前に、COMPRESSED と COMPACT の行フォーマットを使用して達成できる圧縮の量を測定してください。

8.5.2 InnoDB トランザクション管理の最適化

InnoDB トランザクション処理を最適化するには、トランザクション機能のパフォーマンスオーバーヘッドとサーバーのワークロードの理想的なバランスを見つけます。たとえば、アプリケーションで、秒あたり数千回コミットする場合にパフォーマンスの問題が発生し、2、3 時間に 1 回だけコミットする場合に別のパフォーマンスの問題が発生することがあります。

- デフォルトの MySQL 設定 AUTOCOMMIT=1 は、ビジーなデータベースサーバーにパフォーマンスの制限を課すことがあります。現実的であれば、SET AUTOCOMMIT=0 または START TRANSACTION ステートメントを発行し、すべての変更を行なったあとに、COMMIT ステートメントを発行することで、複数の関連 DML 操作を単一のトランザクションにまとめます。

  InnoDB は、トランザクションによってデータベースが変更された場合、そのトランザクションのコミットのたびにディスクにログをフラッシュする必要があります。変更のたびにあとでコミットされる場合 (デフォルトの自動コミット設定のように)、ストレージデバイスの I/O スループットによって、秒あたりに可能な操作数が制限されます。

- または、単一の SELECT ステートメントのみから構成されるトランザクションの場合、AUTOCOMMIT をオンにすると、InnoDB が読み取り専用トランザクションを認識し、それらを最適化するのに役立ちます。要件については、セクション14.13.14「InnoDB の読み取り専用トランザクションの最適化」を参照してください。

- 大量の行の挿入、更新、または削除後のロールバックの実行は避けます。大きなトランザクションによってサーバーのパフォーマンスが低下する場合、それをロールバックすると、問題が悪化し、元の DML 操作の数倍の実行時間がかかる可能性があります。ロールバックはサーバーの起動時に再度開始されるため、データベースプロセスを強制終了しても役立ちません。

  この問題の発生の可能性を最小にするには: すべての DML の変更をただちにディスクに書き込むのではなく、キャッシュできるように、バッファープールのサイズを増やします。挿入に加えて、更新および削除操作がバッファリングされるように、innodb_change_buffering=all を設定します。大きな DML 操作中に、COMMIT ステートメントを定期的に発行し、可能であれば単一の削除または更新を少数の行に対して操作する複数のステートメントに分割することを考慮します。

  ロールバックの暴走が発生した場合にそれを解消するには、ロールバックが CPU に依存して高速に実行するように、バッファープールを増加するか、セクション14.16.1「InnoDB のリカバリプロセス」に説明するように、サーバーを強制終了し、innodb_force_recovery=3 で再起動します。

  この問題は、MySQL 5.5 以上または InnoDB プラグイン付きの MySQL 5.1 では、あまり目立たなくなっていると予想されます。デフォルトの設定 innodb_change_buffering=all により、更新および削除操作がメモリー内にキャッシュされ、それらがそもそも高速に実行されるようになり、必要な場合にロールバックも高速になったためです。多くの挿入、更新、または削除を伴う長時間実行トランザクションを処理するサーバーでこのパラメータ設定を使うようにしてください。

- クラッシュが発生した場合に、最新のコミットされたトランザクションの一部の損失を許容できる場合は、innodb_flush_log_at_trx_commit パラメータを 0 に設定できます。フラッシュが保証されていなくても、InnoDB はとにかく 1 秒に 1 回ログをフラッシュしようとします。さらに、innodb_support_xa の値を 0 に設定し、これにより、ディスク上データとバイナリログの同期によるディスクフラッシュの数を減らします。

- 行が変更されるか削除される場合、行と関連付けられたUndo ログはただちに、またはトランザクションのコミットの直後でも、物理的に削除されません。以前または同時に開始したトランザクションが終了するまで古いデータは保持されるため、それらのトランザクションは変更または削除された行の以前の状態にアクセスできます。そのため、長時間実行トランザクションは、InnoDB が別のトランザクションによって変更されたデータをパージすることを妨げることがあります。

- 長時間実行トランザクション内で行が変更されるか、削除された場合、READ COMMITTED および REPEATABLE READ 分離レベルを使用するほかのトランザクションは、古いデータを再構築するために、それらの同じ行を読み取る場合、多くの作業を実行する必要があります。

- 長時間実行トランザクションでテーブルが変更された場合、ほかのトランザクションからのそのテーブルに対するクエリーは、カバリングインデックス技法を利用しません。通常、セカンダリインデックスからすべての結果カラムを取得できるクエリーは、代わりにテーブルデータから該当する値をルックアップします。

  セカンダリインデックスページに、新しすぎる PAGE_MAX_TRX_ID があることが検出された場合、またはセカンダリインデックス内のレコードに削除がマークされている場合、InnoDB はクラスタ化されたインデックスを使用してレコードをルックアップする必要がある可能性があります。

8.5.3 InnoDB ロギングの最適化

- バッファープールと同じくらいの大きさまでログファイルを大きくします。InnoDB がログファイルにいっぱいまで書き込んだ場合、チェックポイントでバッファープールの変更された内容をディスクに書き込む必要があります。小さいログファイルは多くの不必要なディスク書き込みを発生させます。従来、大きなログファイルは長いリカバリ時間の原因になっていましたが、現在リカバリは大幅に速くなったため、確信を持って大きなログファイルを使うことができます。

- ログバッファーのサイズも十分に大きく (約 8M バイト) してください。

8.5.4 InnoDB テーブルの一括データロード

これらのパフォーマンスのヒントは、セクション8.2.2.1「INSERT ステートメントの速度」の高速挿入の一般的なガイドラインを補足するものです。

- InnoDB にデータをインポートする場合、自動コミットモードでは挿入のたびに、ディスクへのログのフラッシュを実行するため、それをオフにします。インポート操作時に自動コミットを無効にするには、それを、SET autocommit ステートメントと COMMIT ステートメントで囲みます。

```
SET autocommit=0;
... SQL import statements ...
COMMIT;
```

  mysqldump オプション --opt は、それらを SET autocommit ステートメントと COMMIT ステートメントで囲まなくても、InnoDB テーブルに高速にインポートするダンプファイルを作成します。

- 副キーに UNIQUE 制約がある場合、インポートセッション中に一意性チェックを一時的にオフにすることで、テーブルインポートを高速化できます。

```
SET unique_checks=0;
... SQL import statements ...
SET unique_checks=1;
```

  大きいテーブルの場合、InnoDB はその挿入バッファーを使用して、セカンダリインデックスレコードを一括して書き込むことができるため、これにより、大量のディスク I/O が節約されます。データに重複キーが含まれていないことを確認してください。

- テーブルに FOREIGN KEY 制約がある場合、インポートセッションの間の外部キーチェックをオフにすることで、テーブルインポートを高速化できます。

```
SET foreign_key_checks=0;
... SQL import statements ...
SET foreign_key_checks=1;
```

  大きいテーブルの場合、これにより、大量のディスク I/O を節約できます。

- 多くの行を挿入する必要がある場合、複数行 INSERT 構文を使用して、クライアントとサーバー間の通信オーバーヘッドを軽減します。

```
INSERT INTO yourtable VALUES (1,2), (5,5), ...;
```

  このヒントは、InnoDB テーブルだけではなく、任意のテーブルへの挿入に有効です。

- 自動インクリメントカラムのあるテーブルへの一括挿入を実行する場合、innodb_autoinc_lock_mode をデフォルト値の 1 の代わりに 2 に設定します。詳細は、セクション14.6.5.2「構成可能な InnoDB の自動インクリメントロック」を参照してください。

- InnoDB FULLTEXT インデックスにデータをロードする場合の最高のパフォーマンスのため、次の一連のステップに従います。

  - テーブル作成時に、FTS_DOC_ID_INDEX という一意のインデックスで、型 BIGINT UNSIGNED NOT NULL のカラム FTS_DOC_ID を定義します。例:

```
CREATE TABLE t1 (
FTS_DOC_ID BIGINT unsigned NOT NULL AUTO_INCREMENT,
title varchar(255) NOT NULL DEFAULT '',
text mediumtext NOT NULL,
PRIMARY KEY (`FTS_DOC_ID`)
) ENGINE=InnoDB DEFAULT CHARSET=latin1;
CREATE UNIQUE INDEX FTS_DOC_ID_INDEX on t1(FTS_DOC_ID);
```

  - テーブルにデータをロードします。

  - データがロードされたら、FULLTEXT インデックスを作成します。

> **注記**
>
> テーブル作成時に FTS_DOC_ID カラムを追加する場合、FTS_DOC_ID は各 INSERT または UPDATE によって単調に増分される必要があるため、FULLTEXT インデックス設定されたカラムが更新されたときに、FTS_DOC_ID カラムが更新されることを確

> 認します。テーブル作成時に FTS_DOC_ID を追加せず、InnoDB に自動的に DOC ID を管理させるようにした場合、InnoDB は次の CREATE FULLTEXT INDEX 呼び出しで、FTS_DOC_ID を非表示カラムとして追加します。ただし、このアプローチでは、パフォーマンスに影響するテーブルの再構築が必要になります。

8.5.5 InnoDB クエリーの最適化

InnoDB テーブルのクエリーをチューニングするには、各テーブルに適切なインデックスのセットを作成します。詳細は、セクション8.3.1「MySQL のインデックスの使用の仕組み」を参照してください。InnoDB インデックスに関する次のガイドラインに従います。

- 各 InnoDB テーブルには主キー があ る (それをリクエストするかしないかに関係なく) ため、もっとも重要で緊急を要するクエリーで使用されるカラムとして、テーブルごとに主キーカラムのセットを指定します。
- 主キーのカラム値は各セカンダリインデックスに複製されるため、あまり多く、長すぎるカラムを指定しないでください。インデックスに不要なデータが含まれていると、このデータを読み取る I/O とそれをキャッシュするメモリーによって、サーバーのパフォーマンスとスケーラビリティーが低下します。
- 各クエリーで使用できるインデックスは 1 つだけであるため、カラムごとに個別のセカンダリインデックスを作成しないでください。ほんの少数の異なる値を持ち、めったにテストされないカラムへのインデックスは、どのクエリーにも役立たない可能性があります。同じテーブルに対して多くのクエリーがあり、カラムのさまざまな組み合わせをテストする場合、多数の単一カラムインデックスよりも、少数の連結されたインデックスを作成してみてください。インデックスに結果セットに必要なすべてのカラムが含まれている (カバリングインデックスと呼ばれます) 場合、クエリーはテーブルデータをまったく読み取らなくても済む可能性があります。
- インデックス設定されたカラムに NULL 値が含まれることがない場合は、テーブルの作成時に、それを NOT NULL として宣言します。オプティマイザは、各カラムに NULL 値が含まれているかどうかを知っていれば、クエリーに使用するためにもっとも効率的なインデックスをより適切に判断できます。
- MySQL 5.6.4 以上では、セクション14.13.14「InnoDB の読み取り専用トランザクションの最適化」の技法を使用して、InnoDB テーブルの単一クエリートランザクションを最適化できます。
- 頻繁に更新されないテーブルに対して、たびたび繰り返しのクエリーを行う場合、クエリーキャッシュを有効にします。

```
[mysqld]
query_cache_type = 1
query_cache_size = 10M
```

8.5.6 InnoDB DDL 操作の最適化

- テーブルとインデックスに対する DDL 操作 (CREATE、ALTER、および DROP ステートメント) で、InnoDB テーブルのもっとも重要な側面は、MySQL 5.5 以上でのセカンダリインデックスの作成と削除が MySQL 5.1 以前よりはるかに速くなっていることです。詳細は、セクション14.11「InnoDB とオンライン DDL」を参照してください。
- 「高速のインデックス作成」により、特定の場合で、データをテーブルにロードする前にインデックスを削除し、次にデータのロード後にインデックスを再作成することが高速になります。
- テーブルを空にするには DELETE FROM tbl_name ではなく、TRUNCATE TABLE を使用します。外部キー制約により、TRUNCATE ステートメントを通常の DELETE ステートメントのように動作させることができます。その場合、DROP TABLE や CREATE TABLE のようなコマンドのシーケンスがもっとも速くなる可能性があります。
- 主キーは、各 InnoDB テーブルのストレージレイアウトに不可欠であり、主キーの定義の変更には、テーブル全体の再編成が必要であるため、常に主キーを CREATE TABLE ステートメントの一部としてセットアップし、あとで主キーを ALTER または DROP する必要がないように、事前に計画してください。

8.5.7 InnoDB ディスク I/O の最適化

データベース設計と SQL 操作のチューニング技法のベストプラクティスに従っても、データベースが大量のディスク I/O アクティビティーによってまだ遅い場合は、ディスク I/O に関するこれらの低レベル技法を調査してください。Unix top ツールまたは Windows タスクマネージャーに、ワークロードの CPU 使用率が 70% 未満であることが示されている場合、ワークロードはディスクに依存している可能性があります。

- テーブルデータを InnoDB バッファープールにキャッシュすると、ディスク I/O を必要とせずに、クエリーでそれを繰り返し処理できます。バッファープールサイズは、innodb_buffer_pool_size オプションで指定します。

このメモリー領域はきわめて重要であるため、ビジーなデータベースでは多くの場合、サイズを物理メモリーの量の約 80% に指定します。詳細については、セクション8.9.1「InnoDB バッファープール」を参照してください。

- GNU/Linux および Unix の一部のバージョンでは、Unix fsync() 呼び出し (これは InnoDB がデフォルトで使用します) および類似のメソッドによるファイルのディスクへのフラッシュが驚くほど低速です。データベースの書き込みパフォーマンスが問題である場合、innodb_flush_method パラメータを O_DSYNC に設定してベンチマークを実行します。
- x86_64 アーキテクチャー (AMD Opteron) の Solaris 10 で InnoDB ストレージエンジンを使用する場合、InnoDB 関連ファイルにダイレクト I/O を使用して、InnoDB のパフォーマンスの低下を回避します。InnoDB 関連ファイルを格納するために使用される UFS ファイルシステム全体にダイレクト I/O を使用するには、それを forcedirectio オプションでマウントします。mount_ufs(1M) を参照してください。(Solaris 10/x86_64 のデフォルトではこのオプションを使用しません)。ダイレクト I/O をファイルシステム全体ではなく、InnoDB ファイル操作にのみ適用するには、innodb_flush_method = O_DIRECT を設定します。この設定では、InnoDB はデータファイルへの I/O (ログファイルへの I/O ではなく) に fcntl() ではなく、directio() を呼び出します。
- Solaris 2.6 以上の任意のリリースおよび任意のプラットフォーム (sparc/x86/x64/amd64) で、大きな innodb_buffer_pool_size 値を使用して、InnoDB ストレージエンジンを使用する場合、先述の forcedirectio マウントオプションを使用して、raw デバイスまたは個別のダイレクト I/O UFS ファイルシステムで、InnoDB データファイルおよびログファイルのベンチマークを実行します。(ログファイルのダイレクト I/O が必要な場合、innodb_flush_method を設定する代わりに、マウントオプションを使用する必要があります。)Veritas ファイルシステム VxFS のユーザーは、convosync=direct マウントオプションを使用してください。

  ダイレクト I/O ファイルシステムに、MyISAM テーブルのファイルなど、ほかの MySQL データファイルを配置しないでください。実行ファイルやライブラリは、ディレクト I/O ファイルシステムに配置しないでください。
- RAID 構成または別のディスクへのシンボリックリンクをセットアップするために追加のストレージデバイスを使用できるようにする場合、追加の低レベル I/O のヒントについては、セクション8.11.3「ディスク I/O の最適化」を参照してください。
- InnoDB チェックポイント操作のため、スループットが周期的に低下する場合、innodb_io_capacity 構成オプションの値を増加することを考慮します。値を大きくすると、フラッシュが頻繁になり、スループットを低下させる可能性のある作業のバックログが避けられます。
- InnoDB フラッシュ操作によって、システムが遅くならない場合は、innodb_io_capacity 構成オプションの値を小さくすることを考慮します。一般に、このオプション値はできるかぎり小さくしますが、前の箇条書きで示したように、スループットに周期的な低下が発生するほど小さくしないでください。オプション値を小さくすることができる一般的なシナリオでは、SHOW ENGINE INNODB STATUS からの出力に、次のような組み合わせが示されることがあります。
  - 履歴リストの長さが短く、数千未満です。
  - 挿入バッファーマージ数が挿入された行数に近いです。
  - バッファープール内の変更されたページが、一貫してバッファープールの innodb_max_dirty_pages_pct をはるかに下回っています。(サーバーが一括挿入を実行していないときに測定します。変更されたページの一括挿入時に、パーセンテージが大幅に高くなるのは正常です。)
  - Log sequence number - Last checkpoint が、InnoDB ログファイルの合計サイズの 7/8 未満か、理想的には 6/8 未満です。
- I/O に依存したワークロードのチューニング時に考慮するその他の InnoDB 構成オプションには次が含まれます: innodb_adaptive_flushing、innodb_change_buffer_max_size、innodb_change_buffering、innodb_flush_neighbors、innodb_log_buffer_size、innodb_log_file_size、innodb_lru_scan_depth、innodb_max_dirty_pages_pct、innodb_max_purge_la および sync_binlog。

8.5.8 InnoDB 構成変数の最適化

軽量の予測可能な負荷のあるサーバーと、常時ほぼいっぱいの容量で実行していたり、高アクティビティーの急増が発生したりするサーバーとでは、もっとも適切に機能する設定が異なります。

InnoDB ストレージエンジンは、多くの最適化を自動的に実行するため、多くのパフォーマンスチューニングタスクには、データベースが適切に実行していることを確認するためのモニタリングと、パフォーマンスの低下時の構成オプションの変更が含まれます。詳細な InnoDB のパフォーマンスモニタリングについては、セクション14.13.11「InnoDB の MySQL パフォーマンススキーマとの統合」を参照してください。

もっとも重要で、最新の InnoDB パフォーマンス機能については、セクション14.13「InnoDB のパフォーマンス」を参照してください。以前のバージョンで InnoDB テーブルを使用していた場合でも、これらの機能は「InnoDB プラグイン」からのものであるため、なじみがないと思われます。プラグインは MySQL 5.1 の組み込みの InnoDB と共存でき、MySQL 5.5 以上でのデフォルトのストレージエンジンになります。

実行できる主な構成ステップは次のようになります。

- 高性能メモリーアロケータを装備するシステムで、InnoDB がそれらを使用できるようにします。セクション14.13.3「InnoDB のためのメモリーアロケータの構成」を参照してください。
- 頻繁な小さなディスク書き込みを避けるため、InnoDB が変更されたデータをバッファーする DML 操作の種類を制御します。セクション14.13.4「InnoDB 変更バッファリングの構成」を参照してください。デフォルトはすべての種類の DML 操作をバッファーすることであるため、バッファリングの量を減らす必要がある場合にのみ、この設定を変更してください。
- innodb_adaptive_hash_index オプションを使用して、アダプティブハッシュインデックス機能をオンまたはオフにします。詳しくはセクション14.2.13.6「適応型ハッシュインデックス」をご覧ください。異常なアクティビティーの間にこの設定を変更し、その後、その元の設定にリストアできます。
- コンテキストスイッチングがボトルネックである場合に、InnoDB が処理する同時スレッドの数に制限を設定します。セクション14.13.5「InnoDB のスレッド並列性の構成」を参照してください。
- InnoDB がその先読み操作で実行するプリフェッチの量を制御します。システムに未使用の I/O 容量がある場合、先読みによってクエリーのパフォーマンスが向上することがあります。先読みが多すぎると、負荷の大きいシステムで、パフォーマンスが周期的に低下する可能性があります。セクション14.13.1.1「InnoDB バッファープールのプリフェッチ (先読み) の構成」を参照してください。
- デフォルト値で十分に活用されていないハイエンド I/O サブシステムがある場合、読み取りまたは書き込み操作のバックグラウンドスレッドの数を増やします。セクション14.13.6「InnoDB バックグラウンド I/O スレッドの数の構成」を参照してください。
- バックグラウンドで InnoDB が実行する I/O の量を制御します。セクション14.13.8「InnoDB マスタースレッドの I/O レートの構成」を参照してください。バックグラウンド I/O の量は MySQL 5.1 より大きいため、パフォーマンスに周期的な低下が観察された場合、この設定を縮小した方がよいことがあります。
- InnoDB が特定の種類のバックグラウンドの書き込みを実行するタイミングを判断するアルゴリズムを制御します。セクション14.13.1.2「InnoDB バッファープールのフラッシュの頻度の構成」を参照してください。アルゴリズムはワークロードの種類によって機能する場合と機能しない場合があるため、パフォーマンスに周期的な低下が観察された場合は、この設定をオフした方がよいことがあります。
- コンテキストスイッチングの遅延を最小にするため、マルチコアプロセッサとそれらのキャッシュメモリー構成を利用します。セクション14.13.10「スピンロックのポーリングの構成」を参照してください。
- テーブルスキャンなどの一度だけの操作が、InnoDB バッファーキャッシュに格納された頻繁にアクセスされるデータを妨げることを防ぎます。セクション14.13.1.3「バッファープールをスキャンに耐えられるようにする」を参照してください。
- 信頼性とクラッシュリカバリに適切なサイズにログファイルを調整します。InnoDB ログファイルは、多くの場合にクラッシュ後の長い起動時間を避けるため、小さく維持されてきました。MySQL 5.5.4 で導入された最適化によって、クラッシュリカバリプロセスの特定のステップが高速化します。特に、Redo ログのスキャンと Redo ログの適用は、メモリー管理のアルゴリズムの改善のため、高速化します。長い起動時間を避けるため、ログファイルを人為的に小さく維持していた場合、ログファイルサイズを拡大し、Redo ログレコードのリサイクルのために発生する I/O を削減することを考慮できるようになりました。
- InnoDB バッファープールのインスタンスのサイズと数を構成します。特に数ギガバイトのバッファープールのあるシステムに重要です。セクション14.13.1.4「複数のバッファープールインスタンスの使用」を参照してください。
- 同時トランザクションの最大数を増やします。これはきわめてビジーなデータベースのスケーラビリティーを劇的に向上します。セクション14.13.12「複数のロールバックセグメントによるスケーラビリティーの向上」を参照してください。この機能は、日常の操作中のアクションを必要としませんが、データベースの MySQL 5.5 へのアップグレード中またはその後に、低速シャットダウンを実行して、制限を大きくできるようにする必要があります。
- パージ操作 (ガベージコレクションの一種) をバックグラウンドスレッドに移動します。セクション14.13.13「InnoDB のパージスケジューリングの構成」を参照してください。この設定の結果を効率的に測定するには、ほかの I/O 関連およびスレッド関連の構成設定を先にチューニングします。

- ビジーなサーバーで SQL 操作が列を成し、「渋滞」が発生しないように、InnoDB が同時スレッド間で実行するスイッチングの量を削減します。innodb_thread_concurrency オプションの値を設定します (強力な最新のシステムで最大約 32)。innodb_concurrency_tickets オプションの値を一般に 5000 程度に増やします。このオプションの組み合わせにより、InnoDB が一度に処理するスレッド数に制限を設定し、各スレッドがスワップアウトされるまでに大量の作業を実行できるようにするため、待機スレッドの数が少なくなり、過度なコンテキストスイッチングが発生せずに、操作を完了できます。

8.5.9 多くのテーブルのあるシステムに対する InnoDB の最適化

- InnoDB は、起動後テーブルにはじめてアクセスされたときに、そのテーブルのインデックスカーディナリティー値を計算し、そのような値をテーブルに保存しません。データを多くのテーブルに分割しているシステムでは、このステップに大量の時間がかかることがあります。このオーバーヘッドは最初のテーブルオープン操作にのみ適用されるため、テーブルをあとで使用するために「ウォームアップ」するには、SELECT 1 FROM tbl_name LIMIT 1 などのステートメントを発行して、起動後すぐにそれにアクセスします。

8.6 MyISAM テーブルの最適化

MyISAM ストレージエンジンは、テーブルロックによって同時更新を実行する機能を制限するため、読み取りが大半のデータや並列性の低い操作で最適に実行します。MySQL 5.6 では、MyISAM ではなく、InnoDB がデフォルトのストレージエンジンです。

8.6.1 MyISAM クエリーの最適化

MyISAM テーブルのクエリーを高速化するためのいくつかの一般的なヒント:

- MySQL がクエリーをより適切に最適化できるようにするには、テーブルにデータがロードされたあとに、それに対して ANALYZE TABLE を使用するか、または myisamchk --analyze を実行します。これにより、同じ値がある平均行数を示す各インデックスパートの値を更新します。(一意のインデックスの場合、これは常に 1 です。)MySQL はこれを使用して、非定数式に基づいて、2 つのテーブルを結合する際に選択するインデックスを決定します。SHOW INDEX FROM tbl_name を使用し、Cardinality 値を調べることで、テーブル分析の結果を確認できます。myisamchk --description --verbose はインデックスの分布情報を示します。

- インデックスに従ってインデックスとデータをソートするには、myisamchk --sort-index --sort-records=1 を使用します (インデックス 1 でソートすると仮定して)。インデックスに従って順番にすべての行を読み取りたいと考える一意のインデックスがある場合、これはクエリーを高速にする適切な方法です。この方法で大きなテーブルをはじめてソートするときは、長い時間がかかることがあります。

- 頻繁に更新される MyISAM テーブルに対する複雑な SELECT クエリーを避け、リーダーとライターの競合のために発生するテーブルロックの問題を回避するようにしてください。

- MyISAM は同時挿入をサポートしています。テーブルのデータファイルの途中に空きブロックがなければ、ほかのスレッドがテーブルから読み取るのと同時に新しい行をそれに INSERT できます。これを実行できることが重要な場合、行の削除を避けるようにテーブルを使用することを考慮してください。別の可能性は、テーブルの大量の行を削除したあとに OPTIMIZE TABLE を実行して、テーブルをデフラグすることです。この動作は concurrent_insert 変数の設定によって変更されます。行を削除したテーブルにも新しい行を強制的に追加 (したがって同時挿入を許可) できます。セクション8.10.3「同時挿入」を参照してください。

- 頻繁に変更される MyISAM テーブルでは、すべての可変長カラム (VARCHAR、TEXT、および BLOB) を避けるようにします。テーブルに 1 つしか可変長カラムが含まれていない場合でも、テーブルは動的行フォーマットを使用します。第15章「代替ストレージエンジン」を参照してください。

- 一般に、行が大きくなるためだけに、1 つのテーブルを異なるテーブルに分割することは有益ではありません。行へのアクセスで、もっとも大きくパフォーマンスに打撃を与えるものは、行の先頭バイトを見つけるために必要なディスクシークです。データが見つかったあとは、ほとんどの最新のディスクで、大多数のアプリケーションに十分な速度で行全体を読み取ることができます。テーブルを分割することがかなりの違いをもたらす状況は、固定の行サイズに変更できる動的行フォーマットを使用している MyISAM テーブルの場合か、またはテーブルを著しく頻繁にスキャンする必要があるが、ほとんどのカラムには必要でない場合だけです。第15章「代替ストレージエンジン」を参照してください。

- 通常 expr1、expr2、... の順で行を取得する場合は、ALTER TABLE ... ORDER BY expr1, expr2, ... を使用します。テーブルを大幅に変更したあとにこのオプションを使用することで、パフォーマンスを向上できることがあります。

- 多数の行の情報に基づいたカウントなど、結果を頻繁に計算する必要がある場合、新しいテーブルを導入し、リアルタイムでカウンタを更新する方が望ましいことがあります。次のような形式の更新はきわめて高速です。

```
UPDATE tbl_name SET count_col=count_col+1 WHERE key_col=constant;
```

これは、テーブルレベルのロック (単一ライターと複数リーダー) しかない MyISAM のような MySQL ストレージエンジンを使用する場合に、きわめて重要です。また、この場合に行ロックマネージャーが実行する必要があることは少ないため、ほとんどのデータベースシステムでパフォーマンスが向上します。

- データが書き込まれるタイミングを知る必要がない場合は、MyISAM (またはその他のサポートされる非トランザクションテーブル) に INSERT DELAYED を使用します。多くの行を 1 回のディスク書き込みで書き込むことができるため、これにより、挿入の全体の影響が少なくなります。

  > **注記**
  >
  > MySQL 5.6.6 現在、INSERT DELAYED は非推奨であり、将来のリリースで削除されます。代わりに INSERT (DELAYED を付けない) を使用してください。

- 定期的に OPTIMIZE TABLE を使用して、動的フォーマット MyISAM テーブルの断片化を防ぎます。セクション15.2.3「MyISAM テーブルのストレージフォーマット」を参照してください。
- DELAY_KEY_WRITE=1 テーブルオプションを使用して MyISAM テーブルを宣言すると、テーブルが閉じられるまで、ディスクにフラッシュされないため、インデックスの更新が速くなります。短所は、そのようなテーブルが開いている間に、何かによってサーバーが強制終了させられた場合に、--myisam-recover-options オプションを使用してサーバーを実行するか、サーバーを再起動する前に myisamchk を実行して、テーブルが問題ないことを確認する必要があることです。(ただし、この場合でも、キー情報は常にデータ行から生成できるため、DELAY_KEY_WRITE を使用しても何も失われないはずです。)
- MyISAM インデックスでは、文字列の前後のスペースが自動的に圧縮されます。セクション13.1.13「CREATE INDEX 構文」を参照してください。
- アプリケーションでクエリーや応答をキャッシュしてから、多くの挿入や更新をまとめて実行することによって、パフォーマンスを向上できます。この操作中にテーブルをロックすることで、すべての更新後にインデックスキャッシュが 1 回だけフラッシュされます。同様の結果を得るために、MySQL のクエリーキャッシュを利用することもできます。セクション8.9.3「MySQL クエリーキャッシュ」を参照してください。

8.6.2 MyISAM テーブルの一括データロード

これらのパフォーマンスのヒントは、セクション8.2.2.1「INSERT ステートメントの速度」の高速挿入の一般的なガイドラインを補足するものです。

- 複数のクライアントが大量の行を挿入する場合のパフォーマンスを向上するには、INSERT DELAYED ステートメントを使用します。セクション13.2.5.2「INSERT DELAYED 構文」を参照してください。この技法は、MyISAM およびその他の一部のストレージエンジンには有効ですが、InnoDB には機能しません。

  > **注記**
  >
  > MySQL 5.6.6 現在、INSERT DELAYED は非推奨であり、将来のリリースで削除されます。代わりに INSERT (DELAYED を付けない) を使用してください。

- MyISAM テーブルでは、データファイルの途中に削除された行がない場合、SELECT ステートメントの実行中に同時に、同時挿入を使用して行を追加できます。セクション8.10.3「同時挿入」を参照してください。
- 少しの追加作業で、MyISAM テーブルに多数のインデックスがある場合に、テーブルの LOAD DATA INFILE の実行をさらに高速化できます。次の手順を使用します。

  1. FLUSH TABLES ステートメントまたは mysqladmin flush-tables コマンドを実行します。
  2. テーブルのインデックスのすべての使用を削除するには、myisamchk --keys-used=0 -rq /path/to/db/tbl_name を使用します。
  3. LOAD DATA INFILE を使用して、テーブルにデータを挿入します。これはインデックスを更新しないため、非常に高速です。
  4. 今後、テーブルから読み取りだけをする場合は、myisampack を使用してそれを圧縮します。セクション15.2.3.3「圧縮テーブルの特徴」を参照してください。
  5. myisamchk -rq /path/to/db/tbl_name を使用してインデックスを再作成します。これにより、ディスクに書き込む前にメモリー内にインデックスツリーを作成し、大量のディスクシークを回避するため、LOAD DATA INFILE 時のインデックスの更新よりかなり高速になります。結果のインデックスツリーは完全にバランスも取れています。

6. FLUSH TABLES ステートメントまたは mysqladmin flush-tables コマンドを実行します。

データを挿入する MyISAM テーブルが空の場合は、LOAD DATA INFILE は先述の最適化を自動的に実行します。自動の最適化と明示的に手順を使用することの主な違いは、サーバーに LOAD DATA INFILE ステートメントの実行時に、インデックスの再作成で割り当てさせることができる量より、myisamchk ではインデックスの作成のためにはるかに多くの一時メモリーを割り当てることができることです。

myisamchk の代わりに次のステートメントを使用して、MyISAM テーブルの一意でないインデックスを無効または有効にすることもできます。これらのステートメントを使用すると、FLUSH TABLE 操作をスキップできます。

```
ALTER TABLE tbl_name DISABLE KEYS;
ALTER TABLE tbl_name ENABLE KEYS;
```

- 非トランザクションテーブルに対して、複数ステートメントで実行される INSERT 操作を高速化するには、テーブルをロックします。

```
LOCK TABLES a WRITE;
INSERT INTO a VALUES (1,23),(2,34),(4,33);
INSERT INTO a VALUES (8,26),(6,29);
...
UNLOCK TABLES;
```

これは、すべての INSERT ステートメントの完了後に、インデックスバッファーが 1 回だけディスクにフラッシュされるため、パフォーマンスにメリットがあります。通常は、INSERT ステートメントの数と同じだけ、インデックスバッファーのフラッシュが行われます。すべての行を 1 つの INSERT で挿入できる場合は、明示的なロックステートメントは必要ありません。

ロックは複数接続テストの合計時間も短縮しますが、個々の接続がロックを待機するため、それらの最大待機時間は長くなることがあります。次のように 5 台のクライアントが同時に挿入の実行を試みるとします。

- 接続 1 は 1000 回の挿入を実行します
- 接続 2、3、および 4 は 1 回の挿入を実行します
- 接続 5 は 1000 回の挿入を実行します

ロックを使用しない場合、接続 2、3、および 4 は 1 と 5 の前に終了します。ロックを使用した場合、接続 2、3、および 4 は 1 または 5 の前に終了しない可能性がありますが、合計時間は約 40% 高速化するはずです。

MySQL では、INSERT、UPDATE、および DELETE 操作はきわめて高速ですが、約 5 回超の連続した挿入や更新を実行するすべての操作の周囲にロックを追加することによって、全体のパフォーマンスを向上できます。著しく多くの連続した挿入を実行する場合、LOCK TABLES のあとにときどき (1,000 行程度ごとに) UNLOCK TABLES を実行して、ほかのスレッドのテーブルへのアクセスを許可できます。これによってもパフォーマンスの向上が得られます。

上記の戦略を使用した場合でも、データのロードには LOAD DATA INFILE より INSERT の方がはるかに遅くなります。

- MyISAM テーブルの LOAD DATA INFILE と INSERT の両方に対してパフォーマンスを向上するには、key_buffer_size システム変数を増やして、キーキャッシュを拡張します。セクション8.11.2「サーバーパラメータのチューニング」を参照してください。

8.6.3 REPAIR TABLE ステートメントの速度

MyISAM テーブルの REPAIR TABLE は、修復操作に myisamchk を使用することと似ており、同じパフォーマンス最適化の一部が適用されます。

- myisamchk にはメモリー割り当てを制御する変数があります。セクション4.6.3.6「myisamchk メモリー使用量」に説明するように、これらの変数を設定してパフォーマンスを向上できることがあります。
- REPAIR TABLE では、同じ原則が適用されますが、修復はサーバーによって実行されるため、myisamchk 変数の代わりに、サーバーシステム変数を設定します。さらに、メモリー割り当て変数の設定に加えて、myisam_max_sort_file_size システム変数を増やすと、修復で高速の filesort 方法が使用され、キーキャッシュ方法による遅い修復が避けられる可能性が高くなります。テーブルファイルのコピーを保持できるだけの十分な空き領域があることを確認したら、システムの最大ファイルサイズに変数を設定します。元のテーブルファイルを格納しているファイルシステムで、空き領域が使用できる必要があります。

次のオプションを使用して、そのメモリー割り当て変数を設定して、myisamchk テーブル修復操作が実行されたとします。

```
--key_buffer_size=128M --myisam_sort_buffer_size=256M
--read_buffer_size=64M --write_buffer_size=64M
```

それらの myisamchk 変数の一部はサーバーシステム変数に対応します。

myisamchk 変数	システム変数
key_buffer_size	key_buffer_size
myisam_sort_buffer_size	myisam_sort_buffer_size
read_buffer_size	read_buffer_size
write_buffer_size	none

各サーバーシステム変数は実行時に設定でき、それらの一部 (myisam_sort_buffer_size、read_buffer_size) にはグローバル値に加えてセッション値もあります。セッション値を設定することで、現在のセッションへの変更の影響を制限し、ほかのユーザーに影響しません。グローバルのみの変数 (key_buffer_size、myisam_max_sort_file_size) を変更すると、ほかのユーザーにも影響します。key_buffer_size の場合、バッファーがそれらのユーザーと共有されることを考慮しておく必要があります。たとえば、myisamchk key_buffer_size 変数を 128M バイトに設定した場合、対応する key_buffer_size システム変数をそれより大きく設定し (それがすでに大きく設定されていない場合)、ほかのセッションのアクティビティーによるキーバッファーの使用を許可できます。ただし、グローバルキーバッファーサイズを変更すると、バッファーが無効になり、ディスク I/O が増加して、ほかのセッションが遅くなります。この問題を回避する代替策は、個別のキーキャッシュを使用し、それを修復対象のテーブルのインデックスに割り当て、修復が完了したら、その割り当てを解除することです。セクション8.9.2.2「複合キーキャッシュ」を参照してください。

先述の説明に基づいて、REPAIR TABLE 操作は、次のように実行して、myisamchk コマンドに似た設定を使用できます。ここでは、個別の 128M バイトのキーバッファーが割り当てられ、ファイルシステムは 100G バイト以上のファイルサイズを許可するものとします。

```
SET SESSION myisam_sort_buffer_size = 256*1024*1024;
SET SESSION read_buffer_size = 64*1024*1024;
SET GLOBAL myisam_max_sort_file_size = 100*1024*1024*1024;
SET GLOBAL repair_cache.key_buffer_size = 128*1024*1024;
CACHE INDEX tbl_name IN repair_cache;
LOAD INDEX INTO CACHE tbl_name;
REPAIR TABLE tbl_name ;
SET GLOBAL repair_cache.key_buffer_size = 0;
```

グローバル変数を変更するが、ほかのユーザーへの影響を最小にするため、REPAIR TABLE 操作の間にのみ実行するようにしたい場合、その値をユーザー変数に保存して、あとでそれをリストアします。例:

```
SET @old_myisam_sort_buffer_size = @@GLOBAL.myisam_max_sort_file_size;
SET GLOBAL myisam_max_sort_file_size = 100*1024*1024*1024;
REPAIR TABLE tbl_name ;
SET GLOBAL myisam_max_sort_file_size = @old_myisam_max_sort_file_size;
```

REPAIR TABLE に影響するシステム変数は、変数をデフォルトで有効にしたい場合、サーバーの起動時にグローバルに設定できます。たとえば、次の行をサーバーの my.cnf ファイルに追加します。

```
[mysqld]
myisam_sort_buffer_size=256M
key_buffer_size=1G
myisam_max_sort_file_size=100G
```

これらの設定には read_buffer_size は含まれません。read_buffer_size をグローバルに大きな値に設定すると、すべてのセッションに対してそれが実行され、多くの同時セッションのあるサーバーに過剰なメモリーが割り当てられるため、パフォーマンスが低下する可能性があります。

8.7 MEMORY テーブルの最適化

頻繁にアクセスされ、読み取り専用かめったに更新されない非クリティカルデータに MEMORY テーブルを使用することを考慮します。現実的なワークロードで、同等の InnoDB または MyISAM テーブルに対してアプリケーションのベンチマークを実行し、追加のパフォーマンスが、データの損失のリスクやアプリケーションの起動時にディスクベースのテーブルからデータをコピーすることのオーバーヘッドに値するかを確認します。

MEMORY テーブルで最高のパフォーマンスを得るには、各テーブルに対するクエリーの種類を調査し、関連付けられた各インデックスに使用する B ツリーインデックスまたはハッシュインデックスのいずれかの種類を指定し

ます。CREATE INDEX ステートメントで、句 USING BTREE または USING HASH を使用します。B ツリーインデックスは、> や BETWEEN などの操作によって、greater-than または less-than の比較を実行するクエリーで高速です。ハッシュインデックスは、= 演算子によって単一の値、または IN 演算子によって制限された値のセットをルックアップするクエリーでのみ高速です。USING BTREE が多くの場合にデフォルトの USING HASH より適切な選択である理由については、セクション8.2.1.20「フルテーブルスキャンを回避する方法」を参照してください。さまざまな種類の MEMORY インデックスの実装の詳細については、セクション8.3.8「B ツリーインデックスとハッシュインデックスの比較」を参照してください。

8.8 クエリー実行プランの理解

WHERE 句内のテーブル、カラム、インデックス、および条件の詳細に応じて、MySQL オプティマイザは SQL クエリーに含まれるルックアップを効率的に実行するための多くの技法を考慮します。巨大なテーブルに対するクエリーは、すべての行を読み取らなくても実行でき、複数のテーブルを含む結合は、行のすべての組み合わせを比較しなくても実行できます。オプティマイザがもっとも効率的なクエリーを実行するために選択する操作のセットは、「クエリー実行プラン」と呼ばれ、EXPLAIN プランとも呼ばれます。目的は、クエリーが適切に最適化されていることを示す EXPLAIN プランの側面を認識し、非効率的な操作が見られた場合に、プランを改善するための SQL 構文とインデックス設定技法を学ぶことです。

8.8.1 EXPLAIN によるクエリーの最適化

EXPLAIN ステートメントを使用して、MySQL がステートメントを実行する方法に関する情報を取得できます。

- MySQL 5.6.3 現在、EXPLAIN に使用できる説明可能なステートメントは、SELECT、DELETE、INSERT、REPLACE、および UPDATE です。MySQL 5.6.3 より前では、SELECT が唯一の説明可能なステートメントです。
- 説明可能なステートメントで EXPLAIN を使用すると、MySQL は、オプティマイザからのステートメント実行プランに関する情報を表示します。つまり、MySQL はテーブルがどのように、どんな順番で結合されているかに関する情報を含む、ステートメントを処理する方法を説明します。EXPLAIN を使用して、実行プラン情報を取得することについては、セクション8.8.2「EXPLAIN 出力フォーマット」を参照してください。
- EXPLAIN EXTENDED を使用して、追加の実行プラン情報を取得できます。セクション8.8.3「EXPLAIN EXTENDED 出力フォーマット」を参照してください。
- EXPLAIN PARTITIONS は、パーティション化されたテーブルを含むクエリーの調査に役立ちます。セクション19.3.5「パーティションに関する情報を取得する」を参照してください。
- MySQL 5.6.3 現在、FORMAT オプションを使用して、出力フォーマットを選択できます。TRADITIONAL は表形式で出力を表示します。FORMAT オプションが存在しない場合、これはデフォルトです。JSON フォーマットは JSON フォーマットで情報を表示します。FORMAT = JSON を使用すると、出力には拡張されたパーティション情報が含まれます。

EXPLAIN によって、インデックスを使用して行を見つけることで、ステートメントが高速に実行されるように、テーブルにインデックスを追加するべき場所がわかります。また、EXPLAIN を使用して、オプティマイザがテーブルを最適な順序で結合しているかどうかを確認することもできます。SELECT ステートメントでテーブルが指定されている順序に対応する結合順序を使用するように、オプティマイザにヒントを提供するには、ステートメントを SELECT だけでなく、SELECT STRAIGHT_JOIN で始めます。(セクション13.2.9「SELECT 構文」を参照してください。)

インデックスが使われるはずであると思うタイミングでそれらが使われていない問題がある場合、ANALYZE TABLE を実行して、オプティマイザが行う選択に影響する可能性があるキーのカーディナリティーなどのテーブル統計を更新します。セクション13.7.2.1「ANALYZE TABLE 構文」を参照してください。

> 注記
>
> EXPLAIN はテーブル内のカラムに関する情報を取得するためにも使用できます。EXPLAIN tbl_name は DESCRIBE tbl_name および SHOW COLUMNS FROM tbl_name と同義です。詳細については、セクション13.8.1「DESCRIBE 構文」およびセクション13.7.5.6「SHOW COLUMNS 構文」を参照してください。

8.8.2 EXPLAIN 出力フォーマット

EXPLAIN ステートメントは SELECT ステートメントの実行プランに関する情報を提供します。

EXPLAIN は SELECT ステートメントで使用される各テーブルに関する情報の行を返します。これは、MySQL がステートメントの処理中にテーブルを読み取る順番で、出力にテーブルを一覧表示します。MySQL は Nested

Loop 結合メソッドを使用して、すべての結合を解決します。これは、MySQL が最初のテーブルから行を読み取り、次に 2 つめのテーブル、3 つめのテーブルというように、一致する行を見つけることを意味します。すべてのテーブルが処理されると、MySQL は選択したカラムを出力し、さらに一致する行があるテーブルが見つかるまで、テーブルリストを逆戻りします。次の行がテーブルから読み取られ、プロセスは次のテーブルに進みます。

EXTENDED キーワードを使用すると、EXPLAIN は、EXPLAIN ステートメントに続けて SHOW WARNINGS ステートメントを発行することで表示できる追加の情報を生成します。EXPLAIN EXTENDED はフィルタ処理されたカラムも表示します。セクション8.8.3「EXPLAIN EXTENDED 出力フォーマット」を参照してください。

> **注記**
>
> EXTENDED キーワードと PARTITIONS キーワードを、同じ EXPLAIN ステートメントで一緒に使用することはできません。

- EXPLAIN 出力カラム
- EXPLAIN 結合型
- EXPLAIN 追加情報
- EXPLAIN 出力の解釈

EXPLAIN 出力カラム

このセクションでは、EXPLAIN によって生成される出力カラムについて説明します。あとのセクションで、type と Extra カラムに関する追加情報を提供します。

EXPLAIN からの各出力行は 1 つのテーブルに関する情報を提供します。各行には、表8.1「EXPLAIN 出力カラム」で要約し、次の表に詳しく説明している値が格納されます。

表 8.1 EXPLAIN 出力カラム

カラム	意味
id	SELECT 識別子。
select_type	SELECT 型
table	出力行のテーブル
partitions	一致するパーティション
type	結合型
possible_keys	選択可能なインデックス
key	実際に選択されたインデックス
key_len	選択されたキーの長さ
ref	インデックスと比較されるカラム
rows	調査される行の見積もり
filtered	テーブル条件によってフィルタ処理される行の割合
Extra	追加情報

- id

  SELECT 識別子。これはクエリー内の SELECT の連番です。行がほかの行の和集合結果を参照する場合に、値は NULL になることがあります。この場合、table カラムには、<unionM,N> などの値が表示され、行が M および N の id 値のある行の和集合を参照していることが示されます。

- select_type

  SELECT の種類で、次の表に示すもののいずれかになります。

select_type 値	意味
SIMPLE	単純な SELECT (UNION やサブクエリーを使用しません)
PRIMARY	もっとも外側の SELECT

select_type 値	意味
UNION	UNION 内の 2 つめ以降の SELECT ステートメント
DEPENDENT UNION	UNION 内の 2 つめ以降の SELECT ステートメントで、外側のクエリーに依存します
UNION RESULT	UNION の結果。
SUBQUERY	サブクエリー内の最初の SELECT
DEPENDENT SUBQUERY	サブクエリー内の最初の SELECT で、外側のクエリーに依存します
DERIVED	派生テーブル SELECT (FROM 句内のサブクエリー)
MATERIALIZED	実体化されたサブクエリー
UNCACHEABLE SUBQUERY	結果をキャッシュできず、外側のクエリーの行ごとに再評価される必要があるサブクエリー
UNCACHEABLE UNION	キャッシュ不可能なサブクエリー (UNCACHEABLE SUBQUERY を参照してください) に属する UNION 内の 2 つめ以降の SELECT

DEPENDENT は一般に、相関サブクエリーの使用を示します。セクション13.2.10.7「相関サブクエリー」を参照してください。

DEPENDENT SUBQUERY の評価は UNCACHEABLE SUBQUERY の評価とは異なります。DEPENDENT SUBQUERY の場合、その外部コンテキストの変数の異なる値の各セットにつき、一回だけサブクエリーが再評価されます。UNCACHEABLE SUBQUERY の場合、外部コンテキストの行ごとにサブクエリーが再評価されます。

サブクエリーのキャッシュ可能性は、クエリーキャッシュへのクエリー結果のキャッシュ (これについてはセクション8.9.3.1「クエリーキャッシュの動作」で説明しています) と異なります。サブクエリーのキャッシュは、クエリー実行中に行われ、クエリーキャッシュは、クエリーの実行が終了したあとにのみ、結果を格納するために使用されます。

- table

  出力の行で参照しているテーブルの名前。これも次のいずれかの値になることがあります。

  - <unionM,N>: 行は M および N の id 値のある行の和集合を参照しています。
  - <derivedN>: 行は N の id 値のある行の派生テーブル結果を参照しています。派生テーブルは、たとえば FROM 句内のサブクエリーの結果などになります。
  - <subqueryN>: 行は N の id 値のある行の実体化されたサブクエリーの結果を参照しています。サブクエリー実体化によるサブクエリーの最適化を参照してください。

- partitions

  クエリーでレコードが照合されるパーティション。このカラムは、PARTITIONS キーワードが使用されている場合にのみ表示されます。パーティション化されていないテーブルの場合、この値は NULL です。セクション19.3.5「パーティションに関する情報を取得する」を参照してください。

- type

  結合型。さまざまな型の説明については、「EXPLAIN 結合型」を参照してください。

- possible_keys

  possible_keys カラムは、MySQL がこのテーブル内の行の検索に使用するために選択できるインデックスを示します。このカラムは EXPLAIN の出力に表示されたテーブルの順序にまったく依存しません。つまり、possible_keys のキーの一部は、生成されたテーブルの順序で実際に使用できないことがあります。

  このカラムが NULL の場合は、関連するインデックスがありません。この場合、WHERE 句を調査して、それがインデックス設定に適したカラムを参照しているかどうかをチェックすることで、クエリーのパフォーマンスを向上できることがあります。その場合は、適切なインデックスを作成し、再度 EXPLAIN でクエリーをチェックします。セクション13.1.7「ALTER TABLE 構文」を参照してください。

  テーブルにあるインデックスを確認するには、SHOW INDEX FROM tbl_name を使用します。

- key

key カラムは、MySQL が実際に使用することを決定したキー (インデックス) を示します。MySQL が行をルックアップするために、いずれかの possible_keys インデックスを使用することを決定した場合、キー値としてそのインデックスが一覧表示されます。

key は possible_keys 値に存在しないインデックスを指定している可能性があります。これは possible_keys インデックスのどれも行のルックアップに適していない場合に発生する可能性がありますが、クエリーによって選択されるすべてのカラムはほかのインデックスのカラムになります。つまり、指定されたインデックスは選択されたカラムをカバーするため、取得する行を決定するために使用されませんが、インデックススキャンはデータ行スキャンよりも効率的です。

InnoDB は各セカンダリインデックスとともに主キー値を保存するため、InnoDB では、クエリーで主キーも選択している場合でも、セカンダリインデックスで選択されたカラムをカバーしている可能性があります。key が NULL の場合、MySQL はクエリーをより効率的に実行するために使用するインデックスを見つけられませんでした。

MySQL で possible_keys カラムに示されたインデックスを強制的に使用させるか、無視させるには、クエリーで FORCE INDEX、USE INDEX、または IGNORE INDEX を使用します。セクション13.2.9.3「インデックスヒントの構文」を参照してください。

MyISAM テーブルと NDB テーブルの場合、ANALYZE TABLE を実行することで、オプティマイザがより適切なインデックスを選択するために役立ちます。NDB テーブルの場合、これにより、分散されたプッシュダウン結合のパフォーマンスも向上します。MyISAM テーブルの場合、myisamchk --analyze は ANALYZE TABLE と同じことを実行します。セクション7.6「MyISAM テーブルの保守とクラッシュリカバリ」を参照してください。

- key_len

  key_len カラムは、MySQL が使用することを決定したキーの長さを示します。key カラムに NULL と示されている場合、この長さは NULL になります。key_len の値によって、MySQL が実際に使用するマルチパートキーのパート数を判断できます。

- ref

  ref カラムは、テーブルから行を選択するために、key カラムに指定されたインデックスに対して比較されるカラムまたは定数を示します。

  値が func の場合、使用される値は、特定の関数の結果です。どの関数か確認するには、EXPLAIN EXTENDED のあとに SHOW WARNINGS を付けて使用します。関数は、実際には算術演算子などの演算子である場合があります。

- rows

  rows カラムは、MySQL がクエリーを実行するために調査する必要があると考える行数を示します。

  InnoDB テーブルの場合、これは推定値であり、常に正確ではないことがあります。

- filtered

  filtered カラムは、テーブル条件によってフィルタ処理されるテーブル行の推定の割合を示します。つまり、rows は調査される推定の行数を示し、rows × filtered / 100 が前のテーブルと結合される行数を示します。EXPLAIN EXTENDED を使用すると、このカラムが表示されます。

- Extra

  このカラムには、MySQL がクエリーを解決する方法に関する追加情報が含まれます。さまざまな値の説明については、「EXPLAIN の追加情報」を参照してください。

EXPLAIN 結合型

EXPLAIN 出力の type カラムには、テーブルの結合方法が示されます。次のリストに、もっとも適切な型からもっとも不適切な型の順番で並べた結合型を示します。

- system

  テーブルには行が 1 つしかありません (= system テーブル)。これは、const 結合型の特殊なケースです。

- const

テーブルには、一致するレコードが最大で 1 つあり、クエリーの開始時に読み取られます。行が 1 つしかないため、この行のカラムの値は、オプティマイザの残りによって定数とみなされることがあります。const テーブルは、1 回しか読み取られないため、非常に高速です。

const は PRIMARY KEY または UNIQUE インデックスのすべてのパートを定数値と比較する場合に使用されます。次のクエリーでは、tbl_name は const テーブルとして使用できます。

```
SELECT * FROM tbl_name WHERE primary_key=1;

SELECT * FROM tbl_name
 WHERE primary_key_part1=1 AND primary_key_part2=2;
```

- eq_ref

前のテーブルの行の組み合わせごとに、このテーブルから 1 行ずつ読み取られます。system と const 型以外で、これは最適な結合型です。これは、結合でインデックスのすべてのパートが使用されており、インデックスが PRIMARY KEY または UNIQUE NOT NULL インデックスである場合に使用されます。

eq_ref は、= 演算子を使用して比較されるインデックス設定されたカラムに使用できます。比較値は、定数またはこのテーブルより前に読み取られたテーブルのカラムを使用する式を指定できます。次の例では、MySQL は eq_ref 結合を使用して、ref_table を処理できます。

```
SELECT * FROM ref_table,other_table
 WHERE ref_table.key_column=other_table.column;

SELECT * FROM ref_table,other_table
 WHERE ref_table.key_column_part1=other_table.column
 AND ref_table.key_column_part2=1;
```

- ref

前のテーブルの行の組み合わせごとに、一致するインデックス値を持つすべての行がこのテーブルから読み取られます。ref は、結合でキーの左端のプリフィクスのみが使用される場合、またはキーが PRIMARY KEY や UNIQUE インデックスではない場合 (つまり、結合で、キー値に基づいて単一の行を選択できない場合) に使用されます。使用されているキーがほんの数行にしか一致しない場合、これは適切な結合型です。

ref は、= または <=> 演算子を使用して比較されるインデックス設定されたカラムに使用できます。次の例では、MySQL は ref 結合を使用して、ref_table を処理できます。

```
SELECT * FROM ref_table WHERE key_column=expr;

SELECT * FROM ref_table,other_table
 WHERE ref_table.key_column=other_table.column;

SELECT * FROM ref_table,other_table
 WHERE ref_table.key_column_part1=other_table.column
 AND ref_table.key_column_part2=1;
```

- fulltext

結合は FULLTEXT インデックスを使用して実行されます。

- ref_or_null

この結合型は、ref と似ていますが、MySQL が NULL 値を含む行の追加検索を実行することが追加されます。この結合型の最適化は、ほとんどの場合に、サブクエリーの解決で使用されます。次の例では、MySQL は ref_or_null 結合を使用して、ref_table を処理できます。

```
SELECT * FROM ref_table
 WHERE key_column=expr OR key_column IS NULL;
```

セクション8.2.1.8「IS NULL の最適化」を参照してください。

- index_merge

この結合型はインデックスマージ最適化が使用されたことを示します。この場合、出力行の key カラムには使用されたインデックスのリストが含まれ、key_len には使用されたインデックスの最長キーパートのリストが含まれます。詳細については、セクション8.2.1.4「インデックスマージの最適化」を参照してください。

- unique_subquery

この型は、次の形式の IN サブクエリーの ref を置き換えます。

```
value IN (SELECT primary_key FROM single_table WHERE some_expr)
```

unique_subquery は、効率化のため、サブクエリーを完全に置き換える単なるインデックスルックアップ関数です。

- index_subquery

  この結合型は unique_subquery に似ています。IN サブクエリーを置き換えますが、次の形式のサブクエリー内の一意でないインデックスに対して機能します。

```
value IN (SELECT key_column FROM single_table WHERE some_expr)
```

- range

  行を選択するためのインデックスを使用して、特定の範囲にある行のみが取得されます。出力行の key カラムは、使用されるインデックスを示します。key_len には使用された最長のインデックスパートが格納されます。この型の ref カラムは NULL です。

  range は、=、<>、>、>=、<、<=、IS NULL、<=>、BETWEEN、または IN() 演算子のいずれかを使用して、キーカラムを定数と比較する場合に使用できます。

```
SELECT * FROM tbl_name
  WHERE key_column = 10;

SELECT * FROM tbl_name
  WHERE key_column BETWEEN 10 and 20;

SELECT * FROM tbl_name
  WHERE key_column IN (10,20,30);

SELECT * FROM tbl_name
  WHERE key_part1 = 10 AND key_part2 IN (10,20,30);
```

- index

  index 結合型は、インデックスツリーがスキャンされることを除いて、ALL と同じです。これは 2 つの方法で行われます。

  - インデックスがクエリーのカバリングインデックスで、使用すると、テーブルから必要なすべてのデータを満たすことができる場合、インデックスツリーのみがスキャンされます。この場合、Extra カラムには Using index と示されます。インデックスのサイズは通常テーブルデータより小さいため、インデックスのみのスキャンは通常、ALL より高速です。
  - フルテーブルスキャンは、インデックスからの読み取りを使用して、インデックス順でデータ行をルックアップして実行されます。Extra カラムに Uses index が表示されません。

  MySQL は、クエリーで単一のインデックスの一部であるカラムのみが使用されている場合に、この結合型を使用できます。

- ALL

  フルテーブルスキャンは、前のテーブルの行の組み合わせごとに実行されます。これは、通常テーブルが const とマークされていない最初のテーブルである場合には適しておらず、通常ほかのすべてのケースで著しく不適切です。通常、定数値または以前のテーブルからのカラム値に基づいて、テーブルからの行の取得を可能にするインデックスを追加することで、ALL を回避できます。

EXPLAIN の追加情報

EXPLAIN 出力の Extra カラムには、MySQL がクエリーを解決する方法に関する追加情報が含まれます。次のリストに、このカラムに表示される可能性のある値について説明します。クエリーを可能なかぎり高速にしたい場合は、Using filesort および Using temporary の Extra 値に注意します。

- Child of 'table' pushed join@1

  このテーブルは、NDB カーネルにプッシュダウンできる結合内の table の子として参照されます。MySQL Cluster で、プッシュダウンされた結合が有効な場合にのみ適用されます。詳細と例については、ndb_join_pushdown サーバーシステム変数の説明を参照してください。

- const row not found

SELECT ... FROM tbl_name などのクエリーの場合、テーブルは空でした。

- Deleting all rows

  DELETE に対し、一部のストレージエンジン (MyISAM など) は簡単で高速にすべての行テーブルを削除するハンドラメソッドをサポートしています。この Extra 値は、エンジンでこの最適化が使用された場合に表示されます。

- Distinct

  MySQL は個別の値を検索するため、最初に一致する行が見つかったら、現在の行の組み合わせについてのそれ以上の行の検索を停止します。

- FirstMatch(tbl_name)

  tbl_name には、準結合 FirstMatch 結合ショートカット戦略が使用されます。

- Full scan on NULL key

  これは、オプティマイザがインデックスルックアップアクセスメソッドを使用できない場合の代替の戦略として、サブクエリーの最適化で行われます。

- Impossible HAVING

  HAVING 句は常に false で、どの行も選択できません。

- Impossible WHERE

  WHERE 句は常に false で、どの行も選択できません。

- Impossible WHERE noticed after reading const tables

  MySQL はすべての const (および system) テーブルを読み取り、WHERE 句が常に false であることを通知します。

- LooseScan(m..n)

  準結合 LooseScan 戦略が使用されます。m と n はキーパート番号です。

- Materialize、Scan

  MySQL 5.6.7 より前では、これは単一の実体化された一時テーブルの使用を示します。Scan が存在する場合、テーブルの読み取りに一時テーブルインデックスは使用されません。そうでない場合は、インデックスルックアップが使用されます。さらに、Start materialize エントリも参照してください。

  MySQL 5.6.7 現在、実体化は、MATERIALIZED の select_type 値のある行と、<subqueryN> の table 値のある行によって示されます。

- No matching min/max row

  SELECT MIN(...) FROM ... WHERE condition などのクエリーの条件を満たす行がありません。

- no matching row in const table

  結合のあるクエリーで、空のテーブルまたは一意のインデックス条件を満足する行がないテーブルがありました。

- No matching rows after partition pruning

  DELETE または UPDATE に対し、オプティマイザはパーティションのプルーニング後に削除または更新するものが何も見つかりませんでした。それは、SELECT ステートメントの Impossible WHERE に意味が似ています。

- No tables used

  クエリーに FROM 句がないか、FROM DUAL 句があります。

  INSERT または REPLACE ステートメントで、SELECT パートがない場合に、EXPLAIN にこの値が表示されます。たとえば、EXPLAIN INSERT INTO t VALUES(10) に対して、それは EXPLAIN INSERT INTO t SELECT 10 FROM DUAL と同等であるために表示されます。

- Not exists

  MySQL はクエリーに対する LEFT JOIN 最適化を実行でき、LEFT JOIN 条件に一致する 1 つの行が見つかったら、前の行の組み合わせについて、このテーブルでそれ以上の行を調査しません。これは、このように最適化できるクエリーの種類の例です。

  ```
  SELECT * FROM t1 LEFT JOIN t2 ON t1.id=t2.id
    WHERE t2.id IS NULL;
  ```

  t2.id が NOT NULL で定義されているとします。この場合、MySQL は t1 をスキャンし、t1.id の値を使用して t2 内の行をルックアップします。MySQL が t2 内に一致する行を見つけた場合、t2.id は NULL にならないことがわかっているため、同じ id 値を持つ t2 内の残りの行をスキャンしません。つまり、t1 の各行について、MySQL は、t2 内の実際に一致する行数にかかわらず、t2 内の単一のルックアップのみを実行する必要があります。

- Range checked for each record (index map: N)

  MySQL は使用に適したインデックスを見つけられませんでしたが、前のテーブルからのカラム値がわかったあとに、いくつかのインデックスが使用できることがわかりました。以前のテーブルの行の組み合わせごとに、MySQL は range または index_merge アクセスメソッドを使用して、行を取得できるかどうかをチェックします。これは、非常に高速ではありませんが、インデックスがまったくない結合の実行より高速です。前のテーブルのすべてのカラム値がわかっており、定数とみなされることを除き、適用基準は、セクション 8.2.1.3「range の最適化」とセクション8.2.1.4「インデックスマージの最適化」で説明されているとおりです。

  インデックスは、テーブルの SHOW INDEX に示される同じ順序で 1 から番号付けされます。インデックスマップ値 N は、候補となるインデックスを示すビットマスク値です。たとえば、0x19 (2 進数の 11001) の値は、インデックス 1、4、および 5 が考慮されることを示します。

- Scanned N databases

  これは、セクション8.2.4「INFORMATION_SCHEMA クエリーの最適化」に説明するように、サーバーが INFORMATION_SCHEMA テーブルのクエリーを処理する際に実行するディレクトリスキャンの数を示します。N の値は 0、1、または all です。

- Select tables optimized away

  クエリーにはすべてインデックスを使用して解決された集約関数 (MIN()、MAX())、または COUNT(*) のみが含まれていますが、GROUP BY 句は含まれていませんでした。オプティマイザは 1 行のみを返すべきであると判断しました。

- Skip_open_table、Open_frm_only、Open_trigger_only、Open_full_table

  これらの値は、セクション8.2.4「INFORMATION_SCHEMA クエリーの最適化」に説明するように、INFORMATION_SCHEMA テーブルに対するクエリーに適用するファイルオープン最適化を示します。

  - Skip_open_table: テーブルファイルを開く必要はありません。データベースディレクトリをスキャンすることによって、クエリー内ですでに情報を使用できるようになっています。
  - Open_frm_only: テーブルの .frm ファイルのみを開く必要があります。
  - Open_trigger_only: テーブルの .TRG ファイルのみを開く必要があります。
  - Open_full_table: 最適化されていない情報のルックアップ。.frm、.MYD、および .MYI ファイルを開く必要があります。

- Start materialize、End materialize、Scan

  MySQL 5.6.7 より前では、これは複数の実体化された一時テーブルの使用を示します。Scan が存在する場合、テーブルの読み取りに一時テーブルインデックスは使用されません。そうでない場合は、インデックスルックアップが使用されます。さらに、Materialize エントリも参照してください。

  MySQL 5.6.7 現在、実体化は、MATERIALIZED の select_type 値のある行と、<subqueryN> の table 値のある行によって示されます。

- Start temporary、End temporary

  これは、準結合重複除去戦略の一時テーブルの使用を示します。

- unique row not found

  SELECT ... FROM tbl_name などのクエリーの場合に、テーブルに UNIQUE インデックスまたは PRIMARY KEY の条件を満たす行がありません。

- Using filesort

  MySQL はソート順で行を取得する方法を見つけるために、追加のパスを実行する必要があります。ソートは、結合型に従ってすべての行を進み、ソートキーと WHERE 句に一致するすべての行について行へのポインタを格納して実行されます。次にキーがソートされ、ソート順で行が取得されます。セクション8.2.1.15「ORDER BY の最適化」を参照してください。

- Using index

  実際の行を読み取るための追加のシークを実行する必要がなく、インデックスツリーの情報のみを使用して、テーブルからカラム情報が取得されます。この戦略は、クエリーで単一のインデックスの一部であるカラムのみを使用している場合に使用できます。

  Extra カラムに Using where とも示されている場合、キー値のルックアップを実行するためにインデックスが使用されていることを意味します。Using where がない場合、オプティマイザはインデックスを読み取って、データ行の読み取りを回避できますが、それをルックアップに使用していません。たとえば、インデックスがクエリーのカバリングインデックスである場合、オプティマイザはそれをルックアップに使用せずにそれをスキャンできます。

  ユーザー定義のクラスタ化されたインデックスを持つ InnoDB テーブルの場合、そのインデックスは Extra カラムに Using index がない場合でも使用できます。これは、type が index で key が PRIMARY の場合です。

- Using index condition

  インデックスタプルにアクセスし、まずそれらをテストして、すべてのテーブル行を読み取るかどうかを判断することによって、テーブルが読み取られます。このように、必要でないかぎり、すべてのテーブル行の読み取りを遅延 (「プッシュダウン」) するためにインデックス情報が使用されます。セクション8.2.1.6「インデックスコンディションプッシュダウンの最適化」を参照してください。

- Using index for group-by

  Using index テーブルアクセスメソッドと同様に、Using index for group-by は MySQL が、実際のテーブルへの追加のディスクアクセスをせずに、GROUP BY または DISTINCT クエリーのすべてのカラムを取得するために使用できるインデックスを見つけたことを示します。さらに、各グループに対して、少数のインデックスエントリだけが読み取られるように、インデックスがもっとも効率的に使われます。詳細については、セクション8.2.1.16「GROUP BY の最適化」を参照してください。

- Using join buffer (Block Nested Loop)、Using join buffer (Batched Key Access)

  初期の結合からのテーブルは、部分ごとに結合バッファーに読み込まれ、それらの行がバッファーから使用されて、現在のテーブルとの結合が実行されます。(Block Nested Loop) は Block Nested Loop アルゴリズムの使用を示し、(Batched Key Access) は Batched Key Access アルゴリズムの使用を示します。つまり、EXPLAIN 出力の前の行のテーブルからのキーがバッファリングされ、Using join buffer が表示された行によって表されるテーブルから、一致する行が一括してフェッチされます。

- Using MRR

  テーブルは Multi-Range Read 最適化戦略を使用して読み取られます。セクション8.2.1.13「Multi-Range Read の最適化」を参照してください。

- Using sort_union(...)、Using union(...)、Using intersect(...)

  これらは index_merge 結合型でインデックススキャンがどのようにマージされるかを示しています。セクション8.2.1.4「インデックスマージの最適化」を参照してください。

- Using temporary

  クエリーを解決するために、MySQL は結果を保持する一時テーブルを作成する必要があります。これは一般に、クエリーに、カラムを異なって一覧表示する GROUP BY 句と ORDER BY 句が含まれる場合に発生します。

- Using where

  WHERE 句は、次のテーブルに対して照合されるか、またはクライアントに送信される行を制限するために使用されます。具体的にテーブルからすべての行をフェッチするか、調査する意図がないかぎり、Extra 値が Using where でなく、テーブル結合型が ALL または index である場合、クエリーに何らかの誤りがある可能性があります。

- Using where with pushed condition

  この項目は NDB テーブルのみに適用されます。つまり、MySQL Cluster がコンディションプッシュダウン最適化を使用して、インデックス設定されていないカラムと定数の直接比較の効率を向上します。そのような場合、条件がクラスタのデータノードに「プッシュダウン」され、すべてのデータノードで同時に評価されます。これにより、一致しない行をネットワーク経由で送る必要がなくなり、コンディションプッシュダウンを使用できるが使用しない場合より、そのようなクエリーを 5 - 10 倍高速化できます。詳細については、セクション8.2.1.5「エンジンコンディションプッシュダウンの最適化」を参照してください。

EXPLAIN 出力の解釈

EXPLAIN 出力の rows カラムの値の積を取得することで、結合がどの程度適しているかを示す適切な目安を得ることができます。これは、クエリーを実行するために MySQL が調査する必要がある行数を大ざっぱに示すはずです。max_join_size システム変数によってクエリーを制限する場合、この行の積は、どの複数テーブル SELECT ステートメントを実行し、どれを中止するかを判断するためにも使用されます。セクション8.11.2「サーバーパラメータのチューニング」を参照してください。

次の例は、EXPLAIN によって得られた情報に基づいて、複数テーブル結合を段階的に最適化する方法を示しています。

ここに示す SELECT ステートメントがあり、EXPLAIN を使用して調査するつもりであるとします。

```
EXPLAIN SELECT tt.TicketNumber, tt.TimeIn,
            tt.ProjectReference, tt.EstimatedShipDate,
            tt.ActualShipDate, tt.ClientID,
            tt.ServiceCodes, tt.RepetitiveID,
            tt.CurrentProcess, tt.CurrentDPPerson,
            tt.RecordVolume, tt.DPPrinted, et.COUNTRY,
            et_1.COUNTRY, do.CUSTNAME
        FROM tt, et, et AS et_1, do
        WHERE tt.SubmitTime IS NULL
          AND tt.ActualPC = et.EMPLOYID
          AND tt.AssignedPC = et_1.EMPLOYID
          AND tt.ClientID = do.CUSTNMBR;
```

この例では次のように想定しています。

- 比較対象のカラムは次のように宣言されています。

テーブル	カラム	データ型
tt	ActualPC	CHAR(10)
tt	AssignedPC	CHAR(10)
tt	ClientID	CHAR(10)
et	EMPLOYID	CHAR(15)
do	CUSTNMBR	CHAR(15)

- テーブルには次のインデックスがあります。

テーブル	インデックス
tt	ActualPC
tt	AssignedPC
tt	ClientID
et	EMPLOYID (主キー)
do	CUSTNMBR (主キー)

- tt.ActualPC 値は均一に分布されていません。

最初、最適化が実行される前は、EXPLAIN ステートメントで次の情報が生成されました。

```
table type possible_keys key  key_len ref  rows  Extra
et    ALL  PRIMARY       NULL NULL    NULL 74
do    ALL  PRIMARY       NULL NULL    NULL 2135
et_1  ALL  PRIMARY       NULL NULL    NULL 74
tt    ALL  AssignedPC,   NULL NULL    NULL 3872
           ClientID,
           ActualPC
     Range checked for each record (index map: 0x23)
```

各テーブルの type は ALL であるため、この出力は MySQL がすべてのテーブル、つまりすべての行の組み合わせのデカルト積を生成することを示しています。これは、各テーブルの行数の積を調査する必要があるため、著しく時間がかかります。このケースの場合は、この積が 74 × 2135 × 74 × 3872 = 45,268,558,720 行になります。テーブルがもっと大きければ、どのくらい時間がかかっていたか簡単に想像がつきます。

ここでの問題の 1 つは、カラムが同じ型とサイズで宣言されている場合に、MySQL はカラムに対してインデックスをより効率的に使用できることです。このコンテキストでは、VARCHAR と CHAR は同じサイズとして宣言されている場合、それらは同じとみなされます。tt.ActualPC は CHAR(10) として宣言されており、et.EMPLOYID は CHAR(15) であるため、長さの不一致があります。

このカラム長の不一致を修正するには、ALTER TABLE を使用して ActualPC を 10 文字から 15 文字に長くします。

```
mysql> ALTER TABLE tt MODIFY ActualPC VARCHAR(15);
```

これで tt.ActualPC と et.EMPLOYID はいずれも VARCHAR(15) になります。EXPLAIN ステートメントを再度実行すると、次の結果が生成されます。

```
table type   possible_keys key     key_len ref        rows    Extra
tt    ALL    AssignedPC,   NULL    NULL    NULL       3872    Using
             ClientID,                                 where
             ActualPC
do    ALL    PRIMARY       NULL    NULL    NULL       2135
      Range checked for each record (index map: 0x1)
et_1  ALL    PRIMARY       NULL    NULL    NULL       74
      Range checked for each record (index map: 0x1)
et    eq_ref PRIMARY       PRIMARY 15      tt.ActualPC 1
```

これは完全ではありませんが、はるかに改善されています。rows 値の積は 74 の係数分だけ少なくなります。このバージョンは、数秒で実行します。

2 つめの変更を実行して、tt.AssignedPC = et_1.EMPLOYID と tt.ClientID = do.CUSTNMBR の比較でのカラム長の不一致を解消できます。

```
mysql> ALTER TABLE tt MODIFY AssignedPC VARCHAR(15),
    ->                MODIFY ClientID   VARCHAR(15);
```

その変更後、EXPLAIN は次に示す出力を生成します。

```
table type   possible_keys key     key_len ref          rows Extra
et    ALL    PRIMARY       NULL    NULL    NULL         74
tt    ref    AssignedPC,   ActualPC 15     et.EMPLOYID   52   Using
             ClientID,                                  where
             ActualPC
et_1  eq_ref PRIMARY       PRIMARY  15     tt.AssignedPC 1
do    eq_ref PRIMARY       PRIMARY  15     tt.ClientID   1
```

この時点で、クエリーはほぼ可能なかぎり十分に最適化されています。残りの問題は、MySQL はデフォルトで tt.ActualPC カラムの値が均一に分布しているものと想定しますが、tt テーブルにはそれが当てはまらないことです。さいわい、MySQL にキー分布を分析するように伝えることは簡単です。

```
mysql> ANALYZE TABLE tt;
```

追加のインデックス情報によって、結合が完全になり、EXPLAIN が次の結果を生成します。

```
table type   possible_keys key     key_len ref          rows Extra
tt    ALL    AssignedPC    NULL    NULL    NULL         3872 Using
             ClientID,                                  where
             ActualPC
et    eq_ref PRIMARY       PRIMARY 15      tt.ActualPC   1
et_1  eq_ref PRIMARY       PRIMARY 15      tt.AssignedPC 1
do    eq_ref PRIMARY       PRIMARY 15      tt.ClientID   1
```

EXPLAIN の出力の rows カラムは、MySQL 結合オプティマイザの学習による推測です。rows の積とクエリーが返す実際の行数を比較して、数値が実際と近いかどうかをチェックしてください。数値がかなり異なる場合は、SELECT ステートメントで STRAIGHT_JOIN を使用し、FROM 句で異なる順序でテーブルを一覧表示してみるとパフォーマンスを改善できる可能性があります。

場合によっては、サブクエリーで EXPLAIN SELECT を使用するときに、データを変更するステートメントを実行できることもあります。詳細については、セクション13.2.10.8「FROM 句内のサブクエリー」を参照してください。

8.8.3 EXPLAIN EXTENDED 出力フォーマット

EXPLAIN を EXTENDED キーワードを付けて使用すると、出力に、ほかの場合に表示されない filtered カラムが含まれます。このカラムは、テーブル条件によってフィルタ処理されるテーブル行の推定の割合を示します。さらに、ステートメントは、EXPLAIN ステートメントに続けて SHOW WARNINGS ステートメントを発行することで表示できる追加の情報を生成します。SHOW WARNINGS 出力の Message 値には、オプティマイザが SELECT ステートメント内のテーブルおよびカラム名をどのように修飾するか、書き換えおよび最適化ルールの適用後に SELECT がどのように見えるか、および場合によって最適化プロセスに関するその他のメモが表示されます。

これは拡張された出力の例です。

```
mysql> EXPLAIN EXTENDED
    -> SELECT t1.a, t1.a IN (SELECT t2.a FROM t2) FROM t1\G
*************************** 1. row ***************************
           id: 1
  select_type: PRIMARY
        table: t1
         type: index
possible_keys: NULL
          key: PRIMARY
      key_len: 4
          ref: NULL
         rows: 4
     filtered: 100.00
        Extra: Using index
*************************** 2. row ***************************
           id: 2
  select_type: SUBQUERY
        table: t2
         type: index
possible_keys: a
          key: a
      key_len: 5
          ref: NULL
         rows: 3
     filtered: 100.00
        Extra: Using index
2 rows in set, 1 warning (0.00 sec)

mysql> SHOW WARNINGS\G
*************************** 1. row ***************************
  Level: Note
   Code: 1003
Message: /* select#1 */ select `test`.`t1`.`a` AS `a`,
         <in_optimizer>(`test`.`t1`.`a`,`test`.`t1`.`a` in
         ( <materialize> (/* select#2 */ select `test`.`t2`.`a`
         from `test`.`t2` where 1 having 1 ),
         <primary_index_lookup>(`test`.`t1`.`a` in
         <temporary table> on <auto_key>
         where ((`test`.`t1`.`a` = `materialized-subquery`.`a`)))) AS `t1.a
         IN (SELECT t2.a FROM t2)` from `test`.`t1`
1 row in set (0.00 sec)
```

MySQL 5.6.3 現在、EXPLAIN EXTENDED は SELECT、DELETE、INSERT、REPLACE、および UPDATE ステートメントで使用できます。ただし、次の SHOW WARNINGS ステートメントは、SELECT ステートメントに対してのみ、空でない結果を表示します。MySQL 5.6.3 より前では、EXPLAIN EXTENDED は SELECT ステートメントでのみ使用できます。

SHOW WARNINGS によって表示されるステートメントには、クエリーの書き換えやオプティマイザのアクションに関する情報を提供する特別なマーカーが含まれることがあるため、ステートメントは必ずしも有効な SQL ではなく、実行されることを目的としていません。出力には、オプティマイザによってとられたアクションに関する追加の SQL でない説明のメモを提供する Message 値のある行が含まれることもあります。

次のリストに、SHOW WARNINGS によって表示され、EXTENDED 出力に示される可能性がある特別なマーカーを説明します。

- <auto_key>

  一時テーブルの自動的に生成されるキー。

- <cache>(expr)

  式 (スカラーサブクエリーなど) が 1 回実行され、あとで使用するために、結果の値がメモリーに保存されます。複数の値から構成される結果の場合は、一時テーブルが作成されることがあり、代わりに <temporary table> が表示されます。

- <exists>(query fragment)

  サブクエリー述語は EXISTS 述語に変換され、サブクエリーは EXISTS 述語と一緒に使用できるように変換されます。

- <in_optimizer>(query fragment)

  これは、ユーザーにとっては意味がない内部オプティマイザオブジェクトです。

- <index_lookup>(query fragment)

  対象の行を見つけるためにインデックスルックアップを使用して、クエリーフラグメントが処理されます。

- <if>(condition, expr1, expr2)

  条件が true の場合は expr1、そうでない場合は expr2 に評価されます。

- <is_not_null_test>(expr)

  式が NULL に評価されないことを確認するためのテスト。

- <materialize>(query fragment)

  サブクエリーの実体化が使用されます。

- `materialized-subquery`.col_name, `materialized subselect`.col_name

  サブクエリーの評価の結果を保持するために実体化された内部一時テーブル内のカラム col_name への参照。

- <primary_index_lookup>(query fragment)

  対象の行を見つけるために主キールックアップを使用して、クエリーフラグメントが処理されます。

- <ref_null_helper>(expr)

  これは、ユーザーにとっては意味がない内部オプティマイザオブジェクトです。

- /* select#N */ select_stmt

  SELECT は、EXTENDED EXPLAIN 以外の出力で、N の id 値を持つ行に関連付けられます。

- outer_tables semi join (inner_tables)

  準結合操作。inner_tables は、取り出されなかったテーブルを示します。準結合変換によるサブクエリーの最適化を参照してください。

- <temporary table>

  これは、中間結果をキャッシュするために作成される内部一時テーブルを表します。

一部のテーブルが const または system 型である場合、これらのテーブルからのカラムを含む式は、オプティマイザによって早期に評価され、表示されるステートメントに含まれません。ただし、FORMAT=JSON では、一部の const テーブルアクセスが定数値を使用する ref アクセスとして表示されます。

8.8.4 クエリーパフォーマンスの推定

ほとんどの場合、ディスクシークをカウントしてクエリーパフォーマンスを推定できます。小さいテーブルの場合は一般に 1 回のディスクシークでレコードが見つかります (インデックスがキャッシュされている可能性が高いため)。大きなテーブルの場合、B ツリーインデックスを使用して、それを推定できますが、行を見つけるためにこのように多くのシークが必要です。log(row_count) / log(index_block_length / 3 * 2 / (index_length + data_pointer_length)) + 1。

MySQL では、インデックスブロックが通常 1,024 バイトで、データポインタは通常 4 バイトです。3 バイトのキー値長 (MEDIUMINT のサイズ) の 500,000 行のテーブルの場合、この公式は log(500,000)/log(1024/3*2/(3+4)) + 1 = 4 シークを示します。

このインデックスには、約 500,000 * 7 * 3/2 = 5.2M バイト (2/3 の一般的なインデックスバッファー充てん率と想定して) のストレージが必要であるため、インデックスの多くをメモリーに置く可能性が高く、データを読み取り、行を見つけるために 1 つか 2 つの呼び出しだけで済みます。

ただし、書き込みについては、新しいインデックス値の配置場所を見つけるために 4 つのシークリクエスト、およびインデックスの更新と行の書き込みに通常 2 回のシークが必要になります。

前の説明は、アプリケーションのパフォーマンスが log N ずつ徐々に低下することを意味しているわけではありません。OS または MySQL サーバーによってすべてがキャッシュされているかぎり、テーブルが大きくなってもほんの少し遅くなるだけです。データがキャッシュできないほど大きくなると、アプリケーションがディスクシーク (これは log NN ずつ増加する) によってのみ制限されるまで著しく遅くなり始めます。これを回避するには、データの増加に合わせてキーキャッシュを増やします。MyISAM テーブルでは、キーキャッシュサイズは key_buffer_size システム変数によって制御されます。セクション8.11.2「サーバーパラメータのチューニング」を参照してください。

8.8.5 クエリーオプティマイザの制御

MySQL では、クエリー計画の評価方法や有効にされている切り替え可能な最適化に影響するシステム変数によって、オプティマイザを制御します。

8.8.5.1 クエリー計画評価の制御

クエリーオプティマイザのタスクは SQL クエリーを実行するために最適なプランを見つけることです。「良い」プランと「悪い」プランのパフォーマンスの差は、桁違い (つまり、数秒に対して数時間や数日にまで) になる可能性があるため、MySQL のオプティマイザを含むほとんどのクエリーオプティマイザは、多かれ少なかれ、すべての可能なクエリー評価プランの中から最適なプランを徹底的に探します。結合クエリーに対して、MySQL オプティマイザによって調査される可能なプランの数は、クエリーで参照されるテーブル数とともに指数関数的に増大します。少数のテーブル (一般に 7 から 10 未満) の場合、これは問題になりません。ただし、大きなクエリーが送信された場合、クエリーの最適化に費やされる時間が、サーバーのパフォーマンスの大きなボトルネックになりやすいことがあります。

クエリー最適化のより柔軟な方法により、ユーザーはオプティマイザが最適なクエリー評価プランをどの程度徹底的に探すかを制御できます。一般的な考えは、オプティマイザによって調査されるプランが少ないほど、クエリーのコンパイルに費やす時間も少なくなるということです。一方、オプティマイザは一部のプランをスキップするため、最適なプランを見逃す可能性もあります。

評価するプランの数に関して、オプティマイザの動作を 2 つのシステム変数を使用して制御できます。

- optimizer_prune_level 変数は、オプティマイザに、テーブルごとにアクセスされる行数の見積もりに基づいて、特定のプランをスキップするように伝えます。経験上、この種類の「学習による推測」は最適なプランをめったに見逃すことはなく、クエリーのコンパイル時間を劇的に短縮できます。デフォルトでこのオプションがオン optimizer_prune_level=1 であるのはこのためです。ただし、オプティマイザがより適したクエリー計画を見逃したと思う場合は、クエリーのコンパイルにかなりの時間がかかるリスクを伴いますが、このオプションをオフにする (optimizer_prune_level=0) ことができます。この経験則を使用しても、オプティマイザはまだ指数関数的な数のプランを探索します。
- optimizer_search_depth 変数は、オプティマイザがそれ以上拡張すべきかどうかを評価するために、不完全な各プランの「将来」をどの程度見通すかを伝えます。optimizer_search_depth の値を小さくするほど、クエリーのコンパイル時間が桁違いに少なくなる可能性があります。たとえば、12、13、またはそれ以上のテーブルのクエリーは、optimizer_search_depth がクエリー内のテーブル数に近い場合、コンパイルに数時間または数日間も容易に必要になることがあります。同時に、3 か 4 に等しい optimizer_search_depth でコンパイルされた場合、オプティマイザは同じクエリーで 1 分以内にコンパイルできることがあります。optimizer_search_depth の適切な値が不明な場合、この変数を 0 に設定することで、オプティマイザに自動的に値を決定させることができます。

8.8.5.2 切り替え可能な最適化の制御

optimizer_switch システム変数を使用するとオプティマイザの動作を制御できます。その値はフラグのセットで、それぞれ対応するオプティマイザの動作を有効にするかまたは無効にするかを示す on または off の値を持ちます。この変数はグローバル値およびセッション値を持ち、実行時に変更できます。グローバル値のデフォルトはサーバーの起動時に設定できます。

オプティマイザの現在のフラグセットを表示するには、変数値を選択します。

```
mysql> SELECT @@optimizer_switch\G
*************************** 1. row ***************************
@@optimizer_switch: index_merge=on,index_merge_union=on,
                    index_merge_sort_union=on,
                    index_merge_intersection=on,
```

```
                engine_condition_pushdown=on,
                index_condition_pushdown=on,
                mrr=on,mrr_cost_based=on,
                block_nested_loop=on,batched_key_access=off,
                materialization=on,semijoin=on,loosescan=on,
                firstmatch=on,
                subquery_materialization_cost_based=on,
                use_index_extensions=on
```

optimizer_switch の値を変更するには、1 つ以上のコマンドのカンマ区切りのリストから構成される値を割り当てます。

```
SET [GLOBAL|SESSION] optimizer_switch='command[,command]...';
```

各 command 値は、次の表に示すいずれかの形式になるようにしてください。

コマンドの構文	意味
default	すべての最適化をそのデフォルト値にリセットします
opt_name=default	指定した最適化をそのデフォルト値に設定します
opt_name=off	指定した最適化を無効にします
opt_name=on	指定した最適化を有効にします

default コマンドが存在する場合最初に実行されますが、値の中のコマンドの順序は問題ではありません。opt_name フラグを default に設定すると、そのデフォルト値が on または off のどちらであってもそれに設定されます。値に特定の opt_name を複数回指定することは許可されず、エラーが発生します。値のエラーによって、割り当てがエラーを伴って失敗し、optimizer_switch の値が変更されないままになります。

次の表に、最適化戦略別にグループ化した、許可される opt_name フラグ名を一覧表示します。

最適化	フラグ名	意味
Batched Key Access	batched_key_access	BKA 結合アルゴリズムの使用を制御します
Block Nested Loop	block_nested_loop	BNL 結合アルゴリズムの使用を制御します
エンジンコンディションプッシュダウン	engine_condition_pushdown	エンジンコンディションプッシュダウンを制御します
インデックスコンディションプッシュダウン	index_condition_pushdown	インデックスコンディションプッシュダウンを制御します
インデックス拡張	use_index_extensions	インデックス拡張の使用を制御します
インデックスマージ	index_merge	すべてのインデックスマージ最適化を制御します
	index_merge_intersection	インデックスマージ共通集合アクセス最適化を制御します
	index_merge_sort_union	インデックスマージソート和集合アクセス最適化を制御します
	index_merge_union	インデックスマージ和集合アクセス最適化を制御します
Multi-Range Read	mrr	Multi-Range Read 戦略を制御します
	mrr_cost_based	mrr=on の場合にコストベースの MRR の使用を制御します
準結合	semijoin	すべての準結合戦略を制御します
	firstmatch	準結合 FirstMatch 戦略を制御します
	loosescan	準結合 LooseScan 戦略を制御します (GROUP BY の LooseScan と混同しないでください)
サブクエリー実体化	materialization	実体化を制御します (準結合実体化を含む)
	subquery_materialization_cost_based	使用されたコストベースの実体化の選択

block_nested_loop および batched_key_access フラグは MySQL 5.6.3 で追加されました。batched_key_access が on に設定されている場合に何らかの効果を持つためには、mrr フラグも on である必要があります。現在、MRR のコスト見積もりはきわめて悲観的です。したがって、BKA を使用するには、mrr_cost_based を off にする必要もあります。

semijoin、firstmatch、loosescan、および materialization フラグは MySQL 5.6.5 で、準結合およびサブクエリー実体化戦略を制御できるようにするために追加されました。semijoin フラグは準結合を使用するかどうかを制御します。これが on に設定されている場合、firstmatch および loosescan フラグによって、使用可能な準結合戦略を詳細に制御できます。materialization フラグはサブクエリー実体化を使用するかどうかを制御します。semijoin と materialization が両方とも on の場合、該当すれば準結合でも実体化が使用されます。これらのフラグはデフォルトで on です。

subquery_materialization_cost_based は、MySQL 5.6.7 で、サブクエリー実体化と IN -> EXISTS サブクエリー変換の選択を制御できるようにするために追加されました。フラグが on (デフォルト) の場合、オプティマイザは、サブクエリー実体化と IN -> EXISTS サブクエリー変換のどちらの方法も使用できる場合に、コストベースの選択を実行します。フラグが off の場合、オプティマイザは、MySQL 5.6.7 より前の動作だった IN -> EXISTS サブクエリー変換より、サブクエリー実体化を選択します。

個々の最適化戦略の詳細については、次のセクションを参照してください。

- セクション8.2.1.14「Block Nested Loop 結合と Batched Key Access 結合」
- セクション8.2.1.5「エンジンコンディションプッシュダウンの最適化」
- セクション8.2.1.7「インデックス拡張の使用」
- セクション8.2.1.6「インデックスコンディションプッシュダウンの最適化」
- セクション8.2.1.4「インデックスマージの最適化」
- セクション8.2.1.13「Multi-Range Read の最適化」
- セクション8.2.1.18「サブクエリーの最適化」

optimizer_switch に値を割り当てると、指定されていないフラグはそれらの現在の値を維持します。これにより、ほかの動作に影響を与えることなく、単一のステートメントで特定のオプティマイザの動作を有効または無効にできます。ステートメントは、ほかの存在するオプティマイザフラグやそれらの値に依存しません。すべてのインデックスマージ最適化が有効になっているとします。

```
mysql> SELECT @@optimizer_switch\G
*************************** 1. row ***************************
@@optimizer_switch: index_merge=on,index_merge_union=on,
                    index_merge_sort_union=on,
                    index_merge_intersection=on,
                    engine_condition_pushdown=on,
                    index_condition_pushdown=on,
                    mrr=on,mrr_cost_based=on,
                    block_nested_loop=on,batched_key_access=off,
                    materialization=on,semijoin=on,loosescan=on,
                    firstmatch=on,
                    subquery_materialization_cost_based=on,
                    use_index_extensions=on
```

サーバーが特定のクエリーに対して、インデックスマージ和集合アクセスメソッドとインデックスマージソート和集合アクセスメソッドを使用しており、それらがなければオプティマイザの実行が改善されるかどうかをチェックする場合は、変数値を次のように設定します。

```
mysql> SET optimizer_switch='index_merge_union=off,index_merge_sort_union=off';

mysql> SELECT @@optimizer_switch\G
*************************** 1. row ***************************
@@optimizer_switch: index_merge=on,index_merge_union=off,
                    index_merge_sort_union=off,
                    index_merge_intersection=on,
                    engine_condition_pushdown=on,
                    index_condition_pushdown=on,
                    mrr=on,mrr_cost_based=on,
                    block_nested_loop=on,batched_key_access=off,
                    materialization=on,semijoin=on,loosescan=on,
                    firstmatch=on,
                    subquery_materialization_cost_based=on,
                    use_index_extensions=on
```

8.9 バッファリングとキャッシュ

MySQL は、パフォーマンスを向上するため、メモリーバッファーに情報をキャッシュするいくつかの戦略を使用します。

8.9.1 InnoDB バッファープール

InnoDB は、データとインデックスをメモリーにキャッシュするためのバッファープールと呼ばれるストレージ領域を維持しています。InnoDB バッファープールの仕組みを知り、頻繁にアクセスされるデータをメモリーに維持するためにそれを利用することは、MySQL チューニングの重要な側面です。

ガイドライン

理想的には、バッファープールのサイズをできるだけ大きな値に設定して、サーバー上のほかのプロセスが過剰なページングなく実行するように、十分なメモリーを残します。バッファープールが大きいほど、InnoDB はさらにインメモリーデータベースのように動作し、ディスクから 1 回データを読み取り、後続の読み取り時に、メモリーからデータにアクセスします。パフォーマンス向上のため、ディスク書き込みをグループ化できるように、バッファープールは挿入および更新操作によって変更されたデータもキャッシュします。

システムの一般的なワークロードに応じて、バッファープール内の各パートの割合を調整した方がよい場合があります。バッファープールがいっぱいになったときにキャッシュするブロックを選択する方法をチューニングし、バックアップやレポートなどの操作のアクティビティーが急増しても頻繁にアクセスされるデータをメモリーに保持できます。

大きなメモリーサイズを備える 64 ビットシステムでは、バッファープールを複数のパートに分割することで、同時操作中のメモリー構造の競合を最小にできます。詳細については、セクション14.13.1.4「複数のバッファープールインスタンスの使用」を参照してください。

内部の詳細

InnoDB は、LRU (Least Recently Used) アルゴリズムのバリエーションを使用して、プールをリストとして管理します。プールに新しいブロックを追加するために、空きが必要な場合、InnoDB は最近もっとも使用されていないブロックを削除し、新しいブロックをリストの途中に追加します。この「ミッドポイント挿入戦略」はリストを 2 つのサブリストとして扱います。

- 先頭は、最近アクセスされた「新しい」 (または「若い」) ブロックのサブリストです。
- 末尾は、最近あまりアクセスされていない「古い」ブロックのサブリストです。

このアルゴリズムでは、クエリーによって頻繁に使用されるブロックを新しいサブリストに維持します。古いサブリストはあまり使用されないブロックを格納し、これらのブロックはエビクションの候補になります。

LRU アルゴリズムはデフォルトで次のように動作します。

- バッファープールの 3/8 が古いサブリストに割り振られます。
- リストのミッドポイントは、新しいサブリストの末尾と古いサブリストの先頭が接する境界です。
- InnoDB がブロックをバッファープールに読み込むと、最初にそれをミッドポイント (古いサブリストの先頭) に挿入します。ブロックを読み取ることができるのは、SQL クエリーなどのユーザー指定の操作や、InnoDB によって自動的に実行される先読み操作の一部として、必要であるためです。
- 古いサブリスト内のブロックにアクセスすると、それが「若く」なり、バッファープールの先頭 (新しいサブリストの先頭) に移動されます。ブロックが必要であるために読み取られた場合、最初のアクセスはただちに行われ、ブロックが若くなります。先読みのためにブロックが読み取られた場合、最初のアクセスはただちに行われません (ブロックが削除されるまでまったく行われないこともあります)。
- データベースが動作すると、アクセスされていないバッファープール内のブロックが、リストの末尾に移動されることによって、「古く」なります。新しいサブリストと古いサブリストの両方のブロックは、ほかのブロックが新しくなると、古くなります。古いサブリストのブロックは、ブロックがミッドポイントに挿入されたときも古くなります。最終的に、長い間使われないままのブロックは、古いサブリストの末尾に到達し、削除されます。

デフォルトで、クエリーによって読み取られたブロックは、ただちに新しいサブリストに移動され、それらが長時間バッファープールにとどまることを意味します。テーブルスキャン (mysqldump 操作または WHERE 句のない SELECT ステートメントで実行されるような) によって、大量のデータがバッファープールに取り込まれ、新しいデータが再度使われることがない場合でも、同等の量の古いデータが削除されることがあります。同様に、先読みバックグラウンドスレッドによってロードされ、その後 1 回だけアクセスされたブロックは新しいリストの先頭に移動されます。こうした状況では、頻繁に使用されるブロックが古いサブリストに押し出され、そこでそれらがエビクション対象になることがあります。

構成オプション

いくつかの InnoDB システム変数で、バッファープールのサイズを制御し、LRU アルゴリズムをチューニングできます。

- innodb_buffer_pool_size

  バッファープールのサイズを指定します。バッファープールが小さく、十分なメモリーがある場合、プールを大きくすると、クエリーが InnoDB テーブルにアクセスするときに必要なディスク I/O の量が減ることによってパフォーマンスが向上することがあります。

- innodb_buffer_pool_instances

  バッファープールを、それぞれ独自の LRU リストと関連データ構造を持つユーザー指定の数の個別の領域に分割して、同時メモリー読み取りおよび書き込み操作中の競合を削減します。このオプションは、innodb_buffer_pool_size を 1G バイト以上のサイズに設定した場合にのみ有効になります。指定した合計サイズは、すべてのバッファープール間で分割されます。最高の効率を得るには、innodb_buffer_pool_instances と innodb_buffer_pool_size の組み合わせを、各バッファープールインスタンスが少なくとも 1G バイトになるように指定します。

- innodb_old_blocks_pct

  InnoDB が古いブロックサブリストに使用するバッファープールのおおよその割合を指定します。値の範囲は 5 から 95 です。デフォルト値は 37 (つまり、プールの 3/8) です。

- innodb_old_blocks_time

  古いサブリストに挿入されたブロックが、その最初のアクセス後、新しいサブリストに移動するまでに、そこにとどまる必要のある時間をミリ秒 (ms) 単位で指定します。デフォルト値は 0 です。挿入後にどのくらいの期間でアクセスが発生するかに関係なく、古いサブリストに挿入されたブロックは、Innodb がバッファープールから、挿入されたブロックのページの 1/4 を削除したときに、新しいサブリストに移動されます。この値が 0 より大きい場合、ブロックは最初のアクセス後、少なくともそのミリ秒でアクセスが発生するまで、古いサブリストに残ります。たとえば、1000 の値では、ブロックは最初のアクセス後、それらが新しいサブリストに移動される資格を得るまで、1 秒間古いサブリストにとどまります。

innodb_old_blocks_time を 0 より大きく設定すると、1 回のテーブルスキャンで、スキャンだけに使用されたブロックによって新しいサブリストがいっぱいになることを防ぎます。スキャンで読み取られるブロック内の行は、すばやく連続して何回もアクセスされますが、その後ブロックは使用されません。innodb_old_blocks_time がブロックを処理するより長い時間の値に設定されていれば、ブロックは「古い」サブリストに残り、リストの末尾まで古くなり、すぐに削除されます。このようにすると、1 回のスキャンだけに使用されるブロックが、新しいサブリスト内の頻繁に使用されるブロックの損失を促進しません。

innodb_old_blocks_time は実行時に設定できるため、テーブルスキャンやダンプなどの操作の実行中にそれを一時的に変更できます。

```
SET GLOBAL innodb_old_blocks_time = 1000;
... perform queries that scan tables ...
SET GLOBAL innodb_old_blocks_time = 0;
```

目的が、テーブルの内容をバッファープールに入れることによって、バッファープールを「ウォームアップ」することである場合、この戦略は適用されません。たとえば、通常の使用期間後、データは通常バッファープール内にあるため、ベンチマークテストでは、多くの場合サーバーの起動時にテーブルまたはインデックススキャンを実行します。この場合、少なくともウォームアップフェーズが完了するまで、innodb_old_blocks_time を 0 に設定されたままにします。

バッファープールのモニタリング

InnoDB 標準モニターからの出力には、BUFFER POOL AND MEMORY セクションに、バッファープール LRU アルゴリズムの操作に属するいくつかのフィールドが含まれます。

- Old database pages: バッファープールの古いサブリスト内のページ数。
- Pages made young, not young: バッファープールの先頭 (新しいサブリスト) に移動された古いページ数と、新しくなることなく、古いサブリストに残されているページ数。
- youngs/s non-youngs/s: 若くされたか、またはされなかった古いページへのアクセスの数。このメトリックは、2 つの点で、前の項目のそれとは異なります。まず、これは古いページにのみ関連します。2 つめに、それはページへのアクセス数に基づき、ページ数に基づきません。(特定のページに複数のアクセスがあることがあり、そのすべてがカウントされます。)
- young-making rate: ブロックがバッファープールの先頭に移動されるヒット。
- not: ブロックがバッファープールの先頭に移動されないヒット (満たされていない遅延のため)。

young-making 率と not 率は通常バッファープール全体のヒット率まで達することはありません。古いサブリスト内のブロックのヒットによって、それらが新しいサブリストに移動されますが、新しいサブリスト内のブロックへのヒットでは、それらが先頭から特定の距離にある場合にのみ、リストの先頭に移動されます。

モニターからの先述の情報が、LRU のチューニングの決定に役立つことがあります。

- 大きなスキャンが実行中でないときに、youngs/s 値がきわめて低いことが確認された場合、遅延時間を減らすか、古いサブリストに使用されるバッファープールの割合を増やす必要がある可能性があることを示しています。割合を増やすと、古いサブリストが大きくなるため、そのサブリスト内のブロックが末尾に移動され、削除されるまで長くかかるようになります。これにより、再度アクセスされ、若くされる可能性が高くなります。
- 大きなテーブルスキャンの実行中 (および大量の youngs/s) に大量の non-youngs/s が確認されない場合、遅延値を大きくするようにチューニングします。

> **注記**
>
> InnoDB モニターの出力で示される 1 秒あたりの平均は、現在の時間と InnoDB モニターの出力が最後に出力された時間の間の経過時間に基づいています。

InnoDB モニターの詳細については、セクション14.15「InnoDB モニター」を参照してください。

INNODB_BUFFER_POOL_STATS テーブルと InnoDB バッファープールのサーバーステータス変数は、SHOW ENGINE INNODB STATUS 出力によって提供される、多くの同じバッファープール情報を提供します。

8.9.2 MyISAM キーキャッシュ

ディスク I/O を最小にするために、MyISAM ストレージエンジンは多くのデータベース管理システムで使用されている戦略を利用します。それは、もっとも頻繁にアクセスされるテーブルブロックをメモリー内で保持するキャッシュメカニズムを採用しています。

- インデックスブロックの場合、キーキャッシュ (またはキーバッファー) と呼ばれる特別な構造が維持されます。その構造には、もっとも多く使用されるインデックスブロックが置かれる多数のブロックバッファーが含まれます。
- データブロックに対しては、MySQL は特別なキャッシュを使用しません。代わりに、ネイティブオペレーティングシステムのファイルシステムキャッシュに依存します。

このセクションではまず MyISAM キーキャッシュの基本動作について説明します。次に、キーキャッシュパフォーマンスを向上させる機能と、キャッシュ操作をより適切に制御できるようにする機能について説明します。

- 複数のセッションが同時にキャッシュにアクセスできます。
- 複数のキーキャッシュをセットアップし、特定のキャッシュにテーブルインデックスを割り当てることができます。

キーキャッシュのサイズを制御するには、key_buffer_size システム変数を使用します。この変数がゼロに設定されている場合、キーキャッシュは使われません。キーキャッシュは、key_buffer_size 値が小さすぎて、最小数のブロックバッファー (8) を割り当てられない場合も使用されません。

キーキャッシュが動作していない場合、インデックスファイルはオペレーティングシステムによって提供されるネイティブファイルシステムバッファリングのみを使用してアクセスされます。(つまり、テーブルインデックスブロックは、テーブルデータブロックに採用されている同じ戦略を使用してアクセスされます。)

インデックスブロックは MyISAM インデックスファイルへの連続したアクセスの単位です。通常、インデックスブロックのサイズは、インデックス B ツリーのノードのサイズと等しくなります。(インデックスはディスク上で B ツリーデータ構造を使用して表されます。ツリーの下部にあるノードはリーフノードです。リーフノードの上にあるノードは非リーフノードです。)

キーキャッシュ構造内のすべてのブロックバッファーは同じサイズです。このサイズは、テーブルインデックスブロックのサイズと等しいか、大きいか、小さくできます。通常これら 2 つの値のうちの一方は、他方の倍数になります。

いずれかのテーブルインデックスブロックのデータにアクセスする必要がある場合、サーバーはまず、キーキャッシュの何らかのブロックバッファーでそれを使用できるかどうかを確認します。そうである場合、サーバーはディスク上ではなく、キーキャッシュ内のデータにアクセスします。つまり、ディスクから読み取ったり、それに書き込んだりするのではなく、キャッシュから読み取ったり、それに書き込んだりします。そうでない場合、サーバーは別のテーブルインデックスブロックを含むキャッシュブロックバッファーを選択し、その

データを必要なテーブルインデックスブロックのコピーで置き換えます。新しいインデックスブロックがキャッシュに入れられるとただちに、インデックスデータにアクセスできます。

置き換えのために選択されているブロックが変更されていた場合、ブロックは「ダーティー」とみなされます。この場合、置き換えられる前に、その内容が取得元のテーブルインデックスにフラッシュされます。

通常サーバーは LRU (Least Recently Used) 戦略に従います。置き換えるブロックを選択する場合、直近で使用されていないインデックスブロックを選択します。この選択を簡単にするため、キーキャッシュモジュールは、使用されたすべてのブロックを特別なリスト (LRU チェーン) に使用時間で順序付けして保持しています。ブロックがアクセスされると、それは直近で使用されたものになり、リストの末尾に置かれます。ブロックを置き換える必要がある場合、リストの先頭にあるブロックが、直近で使用されていないことになり、エビクションの最初の候補になります。

InnoDB ストレージエンジンは、そのバッファープールを管理するためにも LRU アルゴリズムを使用します。セクション8.9.1「InnoDB バッファープール」を参照してください。

8.9.2.1 共有キーキャッシュアクセス

スレッドはキーキャッシュバッファーに同時にアクセスでき、次の条件に従います。

- 更新中でないバッファーは複数のセッションによってアクセスできます。
- 更新中のバッファーは、更新が完了するまで、それを使用する必要があるセッションを待機させます。
- 複数のセッションは、互いに干渉しないかぎり (つまり、それらは異なるインデックスブロックを必要とし、そのため、異なるキャッシュブロックが置き換えられるかぎり)、キャッシュブロックの置換を引き起こすリクエストを開始できます。

キーキャッシュへの共有アクセスによって、サーバーのスループットを大幅に向上できます。

8.9.2.2 複合キーキャッシュ

キーキャッシュへの共有アクセスはパフォーマンスを向上させますが、セッション間の競合を完全には排除しません。それらはまだキーキャッシュバッファーへのアクセスを管理する制御構造を得るために争います。キーキャッシュアクセスの競合をもっと軽減するために、MySQL は複合キーキャッシュも提供しています。この機能により、異なるキーキャッシュにさまざまなテーブルインデックスを割り当てることができます。

複合キーキャッシュがある場合、サーバーは特定の MyISAM テーブルに対してクエリーを処理する際に、使用すべきキャッシュを知っている必要があります。デフォルトでは、すべての MyISAM テーブルインデックスはデフォルトのキーキャッシュにキャッシュされます。テーブルインデックスを特定のキーキャッシュに割り当てるには、CACHE INDEX ステートメントを使用します (セクション13.7.6.2「CACHE INDEX 構文」を参照してください)。たとえば、次のステートメントは t1、t2、および t3 テーブルから、hot_cache という名前のキーキャッシュにインデックスを割り当てます。

```
mysql> CACHE INDEX t1, t2, t3 IN hot_cache;
+---------+--------------------+----------+----------+
| Table   | Op                 | Msg_type | Msg_text |
+---------+--------------------+----------+----------+
| test.t1 | assign_to_keycache | status   | OK       |
| test.t2 | assign_to_keycache | status   | OK       |
| test.t3 | assign_to_keycache | status   | OK       |
+---------+--------------------+----------+----------+
```

CACHE INDEX ステートメントで参照されているキーキャッシュは、SET GLOBAL パラメータ設定ステートメントでそのサイズを設定するか、またはサーバー起動オプションを使用して作成できます。例:

```
mysql> SET GLOBAL keycache1.key_buffer_size=128*1024;
```

キーキャッシュを破棄するには、そのサイズをゼロに設定します。

```
mysql> SET GLOBAL keycache1.key_buffer_size=0;
```

デフォルトのキーキャッシュは破棄できません。これを実行するすべての試みは無視されます。

```
mysql> SET GLOBAL key_buffer_size = 0;

mysql> SHOW VARIABLES LIKE 'key_buffer_size';
+-----------------+---------+
| Variable_name   | Value   |
+-----------------+---------+
| key_buffer_size | 8384512 |
+-----------------+---------+
```

キーキャッシュ変数は名前とコンポーネントのある構造化システム変数です。keycache1.key_buffer_size の場合、keycache1 はキャッシュ変数名であり、key_buffer_size はキャッシュコンポーネントです。構造化キーキャッシュシステム変数を参照するために使用する構文の詳細については、セクション5.1.5.1「構造化システム変数」を参照してください。

デフォルトで、テーブルインデックスは、サーバー起動時に作成されるメイン (デフォルト) キーキャッシュに割り当てられます。キーキャッシュが破棄されると、それに割り当てられたすべてのインデックスはデフォルトのキーキャッシュに再割り当てされます。

ビジーなサーバーの場合、3 つのキーキャッシュを含む戦略を使用できます。

- すべてのキーキャッシュに割り当てられたスペースの 20% を占める「ホット」キーキャッシュ。これは、検索に頻繁に使用されるが、更新されないテーブルに使用します。
- すべてのキーキャッシュに割り当てられたスペースの 20% を占める「コールド」キーキャッシュ。このキャッシュは、一時テーブルなどの中規模の集中的に変更されるテーブルに使用します。
- キーキャッシュスペースの 60% を占める「ウォーム」キーキャッシュ。これは、デフォルトでほかのすべてのテーブルに使用されるように、デフォルトのキーキャッシュとして使用します。

3 つのキーキャッシュを使用することに利点がある理由の 1 つは、1 つのキーキャッシュ構造へのアクセスが、ほかへのアクセスをブロックしないことです。あるキャッシュに割り当てられたテーブルにアクセスするステートメントは、ほかのキャッシュに割り当てられたテーブルにアクセスするステートメントと競合しません。パフォーマンスの向上はほかの理由でも発生します。

- ホットキャッシュはクエリーの取得にのみ使用されるため、その内容が変更されることはありません。その結果、インデックスブロックをディスクから取り出す必要がある場合常に、置き換えのために選択されたキャッシュブロックの内容を最初にフラッシュする必要はありません。
- ホットキャッシュに割り当てられたインデックスの場合、インデックススキャンを必要とするクエリーがなければ、インデックス B ツリーの非リーフノードに対応するインデックスブロックがキャッシュに残っている可能性が高くなります。
- 一時テーブルに対するもっとも頻繁に実行される更新操作は、更新されるノードがキャッシュ内にあり、最初にディスクから読み取られる必要がない場合、はるかに高速に実行されます。一時テーブルのインデックスのサイズがコールドキーキャッシュのサイズと同程度である場合、更新されるノードがキャッシュ内にある可能性が高くなります。

CACHE INDEX ステートメントは、テーブルとキーキャッシュ間のアソシエーションをセットアップしますが、そのアソシエーションはサーバーが再起動されるたびに失われます。サーバーが起動するたびにアソシエーションを有効にしたい場合、これを実現する 1 つの方法はオプションファイルを使用することです。キーキャッシュを構成する変数設定と、実行される CACHE INDEX ステートメントを含むファイルを指定する init-file オプションを含めます。例:

```
key_buffer_size = 4G
hot_cache.key_buffer_size = 2G
cold_cache.key_buffer_size = 2G
init_file=/path/to/data-directory/mysqld_init.sql
```

サーバーが起動するたびに mysqld_init.sql 内のステートメントが実行されます。ファイルには 1 行に 1 つずつ SQL ステートメントを含めてください。次の例は hot_cache と cold_cache に複数のテーブルをそれぞれ割り当てます。

```
CACHE INDEX db1.t1, db1.t2, db2.t3 IN hot_cache
CACHE INDEX db1.t4, db2.t5, db2.t6 IN cold_cache
```

8.9.2.3 ミッドポイント挿入戦略

デフォルトで、キーキャッシュ管理システムは、削除されるキーキャッシュブロックの選択に、単純な LRU 戦略を使用しますが、ミッドポイント挿入戦略というさらに高度な方法もサポートしています。

ミッドポイント挿入戦略を使用すると、LRU チェーンがホットサブリストとウォームサブリストの 2 つのパートに分割されます。2 つのパート間の分割点は固定ではありませんが、キーキャッシュ管理システムでは、ウォームパートが「短くなりすぎ」ず、常にキーキャッシュブロックの少なくとも key_cache_division_limit パーセントを含むように配慮されます。key_cache_division_limit は構造化キーキャッシュ変数のコンポーネントであるため、その値はキャッシュごとに設定可能なパラメータです。

インデックスブロックがテーブルからキーキャッシュに読み込まれると、それはウォームサブリストの末尾に置かれます。特定の数のヒット (ブロックのアクセス) 後、それはホットサブリストに昇格されます。現在のところ、ブロックを昇格させるために必要なヒット数 (3) はすべてのインデックスブロックで同じです。

ホットサブリストに昇格されるブロックはリストの末尾に置かれます。ブロックはこのサブリスト内で循環されます。ブロックが十分な時間サブリストの先頭にとどまっている場合、それはウォームサブリストに降格されます。この時間はキーキャッシュの key_cache_age_threshold コンポーネントの値によって決定されます。

しきい値は、N ブロックを含むキーキャッシュの場合、最後の N * key_cache_age_threshold / 100 ヒット内にアクセスされないホットサブリストの先頭のブロックが、ウォームサブリストの先頭に移動されることを規定します。置き換えられるブロックは常にウォームサブリストの先頭から取得されるため、その後、それは削除の最初の候補になります。

ミッドポイント挿入戦略により、価値の高いブロックを常にキャッシュ内に保持できます。単純な LRU 戦略を使用したい場合は、key_cache_division_limit 値をそのデフォルトの 100 に設定したままにします。

ミッドポイント挿入戦略は、インデックススキャンを必要とするクエリーの実行で、価値の高い高レベル B ツリーノードに対応するすべてのインデックスブロックを、キャッシュから効率的に押し出す際のパフォーマンスの向上に役立ちます。これを回避するには、key_cache_division_limit を 100 よりかなり小さい値に設定して、ミッドポイント挿入戦略を使用する必要があります。これにより、インデックススキャン操作中でも、価値の高い頻繁にヒットされるノードがホットサブリストに保持されます。

8.9.2.4 インデックスプリロード

キーキャッシュ内に、インデックス全体のブロックを保持するために十分なブロックがあるか、または少なくともその非リーフノードに対応するブロックがある場合、使用を開始する前に、キーキャッシュにインデックスブロックをプリロードすることは役立ちます。プリロードにより、インデックスブロックをディスクから順番に読み取ることで、もっとも効率的にテーブルインデックスをキーキャッシュバッファーに挿入できます。

プリロードしない場合、ブロックは、引き続きクエリーによって必要とされるときに、キーキャッシュに置かれます。ブロックはキャッシュ内にとどまりますが、それらのすべてに対して十分なバッファーがあるため、ディスクからランダムな順序で、順番ではなくフェッチされます。

インデックスをキャッシュにプリロードするには LOAD INDEX INTO CACHE ステートメントを使用します。たとえば、次のステートメントはテーブル t1 および t2 のインデックスのノード (インデックスブロック) をプリロードします。

```
mysql> LOAD INDEX INTO CACHE t1, t2 IGNORE LEAVES;
+---------+--------------+----------+----------+
| Table   | Op           | Msg_type | Msg_text |
+---------+--------------+----------+----------+
| test.t1 | preload_keys | status   | OK       |
| test.t2 | preload_keys | status   | OK       |
+---------+--------------+----------+----------+
```

IGNORE LEAVES 修飾子によって、インデックスの非リーフノードのブロックのみがプリロードされます。したがって、上記のステートメントは t1 からすべてのインデックスブロックをプリロードしますが、t2 からは非リーフノードのブロックのみをプリロードします。

インデックスが CACHE INDEX ステートメントを使用してキーキャッシュに割り当てられている場合、プリロードによって、インデックスブロックがそのキャッシュに置かれます。そうでない場合は、インデックスはデフォルトのキーキャッシュにロードされます。

8.9.2.5 キーキャッシュブロックサイズ

key_cache_block_size 変数を使用して、個々のキーキャッシュのブロックバッファーのサイズを指定できます。これによって、インデックスファイルの I/O 操作のパフォーマンスをチューニングできます。

I/O 操作の最適なパフォーマンスは、読み取りバッファーのサイズがネイティブオペレーティングシステム I/O バッファーのサイズに等しい場合に達成されます。ただし、キーノードのサイズを I/O バッファーのサイズと等しく設定しても、常に全体の最適なパフォーマンスが確保されるわけではありません。大きなリーフノードを読み取る場合、サーバーは大量の不要なデータを取り出し、事実上ほかのリーフノードの読み取りを妨げます。

MyISAM テーブルの .MYI インデックスファイルのブロックのサイズを制御するには、サーバーの起動時に --myisam-block-size オプションを使用します。

8.9.2.6 キーキャッシュの再構築

キーキャッシュはそのパラメータ値を更新することで、いつでも再構築できます。例:

```
mysql> SET GLOBAL cold_cache.key_buffer_size=4*1024*1024;
```

key_buffer_size または key_cache_block_size キーキャッシュコンポーネントに、コンポーネントの現在値と異なる値を割り当てた場合、サーバーはキャッシュの古い構造を破棄し、新しい値に基づいた新しい構造を作成しま

す。キャッシュにダーティーブロックが含まれる場合、サーバーはキャッシュを破棄し、再作成する前にディスクにそれらを保存します。ほかのキーキャッシュパラメータを変更した場合は、再構築が行われません。

キーキャッシュを再構築する場合、サーバーはまずダーティーバッファーの内容をディスクにフラッシュします。その後、キャッシュの内容は使用できなくなります。しかし、再構築は、キャッシュに割り当てられたインデックスを使用する必要があるクエリーをブロックしません。代わりに、サーバーはネイティブファイルシステムキャッシュを使用して、テーブルインデックスに直接アクセスします。ファイルシステムキャッシュはキーキャッシュを使用するときほど効率的ではないため、クエリーが実行されても速度の低下が予想されます。キャッシュが再構築されると、それに割り当てられたインデックスをキャッシュするためにふたたび使用できるようになり、インデックスのファイルシステムキャッシュの使用は停止されます。

8.9.3 MySQL クエリーキャッシュ

クエリーキャッシュには、クライアントに送信された対応する結果とともに、SELECT ステートメントのテキストが格納されます。あとで同じステートメントを受け取った場合、サーバーはそのステートメントを再度解析して実行する代わりに、クエリーキャッシュから結果を取得します。クエリーキャッシュはセッション間で共有されるため、1 つのクライアントで生成された結果セットを、別のクライアントによって発行された同じクエリーへの応答で送信できます。

クエリーキャッシュは、あまり頻繁に変更されないテーブルがあり、それに対してサーバーが多くの同一のクエリーを受け取る環境で役立つことがあります。これは、データベースの内容に基づいて、多くの動的ページを生成する多くの Web サーバーに一般的な状況です。

クエリーキャッシュは古くなったデータを返しません。テーブルが変更されると、クエリーキャッシュ内の関連エントリがフラッシュされます。

> 注記
>
> 同じ MyISAM テーブルを更新する複数の mysqld サーバーがある環境では、クエリーキャッシュは機能しません。

クエリーキャッシュは、セクション8.9.3.1「クエリーキャッシュの動作」に説明された条件の下で、準備されたステートメントに使用されます。

> 注記
>
> MySQL 5.6.5 現在、クエリーキャッシュは、パーティション化されたテーブルに対してはサポートされておらず、パーティション化されたテーブルを含むクエリーには自動的に無効にされます。そのようなクエリーに対しては、クエリーキャッシュを有効にできません。(Bug #53775)

クエリーキャッシュの一部のパフォーマンスデータを次に示します。これらの結果は、2G バイトの RAM と 64M バイトのクエリーキャッシュを搭載する Linux Alpha 2 × 500MHz システムで、MySQL ベンチマークスイートを実行して生成されました。

- 実行しているすべてのクエリーは単純です (1 行のテーブルから行を選択するなど) が、それでも異なっているため、クエリーをキャッシュできない場合、クエリーキャッシュをアクティブにしておくためのオーバーヘッドは 13% です。これは最悪のケースのシナリオとみなすことができます。実際には、クエリーははるかに複雑になる傾向があるため、オーバーヘッドは通常かなり低くなります。
- 単一行テーブル内の単一行の検索は、クエリーキャッシュがあると、それがない場合より、238% 高速化します。これは、キャッシュされているクエリーに予想される最小の高速化に近いとみなすことができます。

サーバーの起動時にクエリーキャッシュを無効にするには、query_cache_size システム変数を 0 に設定します。クエリーキャッシュコードを無効にすることによって、目立ったオーバーヘッドはなくなります。

クエリーキャッシュは、かなりのパフォーマンスの改善の可能性を提供しますが、すべての環境でそうなるものと想定しないでください。クエリーキャッシュの構成やサーバーのワークロードによっては、実際にパフォーマンスの低下が見られることもあります。

- クエリーキャッシュのサイズを過度に大きくすると、キャッシュの保守に必要なオーバーヘッドが増え、それを有効にすることのメリットに勝る可能性があります。数十メガバイトのサイズが通常は有益です。数百メガバイトのサイズはそうでない可能性があります。
- サーバーのワークロードは、クエリーキャッシュの効率にかなりの影響を与えます。ほぼ全体が固定の SELECT ステートメントのセットで構成される複合クエリーでは、頻繁な INSERT ステートメントによってキャッシュ内の結果が絶えず無効にされるような複合クエリーよりも、キャッシュを有効にすることでメリットが得られる可能性がはるかに高くなります。場合によっては、回避方法として、SQL_NO_CACHE オプショ

ンを使用して、頻繁に変更されるテーブルを使用する SELECT ステートメントに対して、結果をキャッシュに入れないようにします。(セクション8.9.3.2「クエリーキャッシュ SELECT オプション」を参照してください。)

クエリーキャッシュを使用することでメリットがあるかどうかを確認するには、キャッシュを有効および無効にして MySQL サーバーの動作をテストします。サーバーのワークロードが変わるとクエリーキャッシュの効率も変わることがあるため、その後、定期的に再テストします。

8.9.3.1 クエリーキャッシュの動作

このセクションでは、クエリーキャッシュが動作可能な場合のその仕組みについて説明します。セクション8.9.3.3「クエリーキャッシュの構成」では、それを動作可能にするかどうかを制御する方法について説明しています。

受信したクエリーは、解析前にクエリーキャッシュにあるそれらと比較されるため、次の 2 つのクエリーは、クエリーキャッシュによって異なるものとみなされます。

```
SELECT * FROM tbl_name
Select * from tbl_name
```

クエリーは、同一とみなされるためには、正確に同じ (バイトごと) である必要があります。さらに、ほかの理由で、同一のクエリー文字列が異なるものとして扱われることもあります。異なるデータベース、異なるプロトコルバージョン、または異なるデフォルトの文字セットを使用するクエリーは、異なるクエリーとみなされ、別々にキャッシュされます。

次の種類のクエリーにはキャッシュが使用されません。

- 外部クエリーのサブクエリーであるクエリー
- ストアドファンクション、トリガー、またはイベントの本体内で実行されるクエリー

クエリー結果をクエリーキャッシュからフェッチする前に、MySQL は、関連するすべてのデータベースとテーブルに対して、ユーザーが SELECT 権限を持っているかどうかをチェックします。これが当てはまらない場合、キャッシュ結果は使用されません。

クエリー結果がクエリーキャッシュから返される場合、サーバーは Com_select ではなく Qcache_hits ステータス変数を増やします。セクション8.9.3.4「クエリーキャッシュのステータスと保守」を参照してください。

テーブルが変更された場合、そのテーブルを使用するキャッシュされたすべてのクエリーが無効になり、キャッシュから削除されます。これには、変更されたテーブルにマップされた MERGE テーブルを使用するクエリーも含まれます。テーブルは、INSERT、UPDATE、DELETE、TRUNCATE TABLE、ALTER TABLE、DROP TABLE、または DROP DATABASE などの多くの種類のステートメントによって変更できます。

InnoDB テーブルを使用する際のトランザクション内でもクエリーキャッシュは機能します。

MySQL 5.6 では、ビュー上の SELECT クエリーからの結果がキャッシュされます。

クエリーキャッシュは、SELECT SQL_CALC_FOUND_ROWS ... クエリーに対して機能し、後続の SELECT FOUND_ROWS() クエリーで返される値を格納します。FOUND_ROWS() は前のクエリーがキャッシュからフェッチされている場合でも、見つかった行の数もキャッシュに格納されているため、正確な値を返します。SELECT FOUND_ROWS() クエリー自体はキャッシュできません。

mysql_stmt_prepare() および mysql_stmt_execute() を使用し、バイナリプロトコルを使用して発行されたプリペアドステートメント (セクション23.7.8「C API プリペアドステートメント」を参照してください) はキャッシュが制限されます。クエリーキャッシュ内のステートメントとの比較は、? パラメータマーカーの拡張後のステートメントのテキストに基づきます。ステートメントは、バイナリプロトコルを使用して実行されたほかのキャッシュされたステートメントとのみ比較されます。つまり、クエリーキャッシュの目的で、バイナリプロトコルを使用して発行されたプリペアドステートメントは、テキストプロトコルを使用して発行されたプリペアドステートメント (セクション13.5「準備済みステートメントのための SQL 構文」を参照してください) と区別されます。

クエリーに次の表に示すいずれかの関数が含まれる場合、クエリーはキャッシュできません。

AES_DECRYPT() (5.7.4 現在)	AES_ENCRYPT() (5.7.4 現在)	BENCHMARK()
CONNECTION_ID()	CONVERT_TZ()	CURDATE()
CURRENT_DATE()	CURRENT_TIME()	CURRENT_TIMESTAMP()
CURTIME()	DATABASE()	1 つのパラメータを持つ ENCRYPT()
FOUND_ROWS()	GET_LOCK()	LAST_INSERT_ID()

LOAD_FILE()	MASTER_POS_WAIT()	NOW()
PASSWORD()	RAND()	RANDOM_BYTES()
RELEASE_LOCK()	SLEEP()	SYSDATE()
パラメータを持たない UNIX_TIMESTAMP()	USER()	UUID()
UUID_SHORT()		

次の条件下のクエリーもキャッシュされません。

- それがユーザー定義関数 (UDF) またはストアドファンクションを参照している。
- それがユーザー変数またはローカルに保存されたプログラム変数を参照している。
- それが mysql、INFORMATION_SCHEMA、または performance_schema データベース内のテーブルを参照している。
- (MySQL 5.6.5 以降:) それがパーティション化されたテーブルを参照している。
- それが次のいずれかの形式である。

```
SELECT ... LOCK IN SHARE MODE
SELECT ... FOR UPDATE
SELECT ... INTO OUTFILE ...
SELECT ... INTO DUMPFILE ...
SELECT * FROM ... WHERE autoincrement_col IS NULL
```

最後の形式は、最後の挿入 ID 値を取得するための ODBC の回避方法として使用されるため、キャッシュされません。第23章「Connector および API」の Connector/ODBC のセクションを参照してください。

SERIALIZABLE 分離レベルを使用するトランザクション内のステートメントは LOCK IN SHARE MODE ロックを使用するため、それらもキャッシュできません。

- それが TEMPORARY テーブルを使用している。
- それがどのテーブルも使用していない。
- それが警告を生成する。
- ユーザーが関連するすべてのテーブルのカラムレベルの権限を持っている。

8.9.3.2 クエリーキャッシュ SELECT オプション

クエリーキャッシュ関連の 2 つのオプションを SELECT ステートメントに指定できます。

- SQL_CACHE

  クエリー結果がキャッシュ可能で、query_cache_type システム変数の値が ON または DEMAND である場合、クエリー結果はキャッシュされます。

- SQL_NO_CACHE

  サーバーはクエリーキャッシュを使用しません。それは、結果がすでにキャッシュされているかどうかを確認するためにクエリーキャッシュをチェックせず、クエリー結果もキャッシュしません。(パーサーの制限のため、スペース文字の前後に SQL_NO_CACHE キーワードを付ける必要があります。改行などのスペース以外では、結果がすでにキャッシュされているかどうかを確認するために、サーバーにクエリーキャッシュをチェックさせます。)

例:

```
SELECT SQL_CACHE id, name FROM customer;
SELECT SQL_NO_CACHE id, name FROM customer;
```

8.9.3.3 クエリーキャッシュの構成

have_query_cache サーバーシステム変数は、クエリーキャッシュが使用できるかどうかを示します。

```
mysql> SHOW VARIABLES LIKE 'have_query_cache';
+------------------+-------+
| Variable_name    | Value |
```

```
+------------------+-------+
| have_query_cache | YES   |
+------------------+-------+
```

標準 MySQL バイナリを使用している場合、クエリーキャッシュが無効にされている場合でも、この値は常に YES です。

ほかのいくつかのシステム変数は、クエリーキャッシュ操作を制御します。これらは、mysqld の起動時に、オプションファイルやコマンド行で設定できます。クエリーキャッシュシステム変数はすべて、query_cache_ で始まる名前を持ちます。それらについては、ここで提供している追加の構成情報とともに、セクション5.1.4「サーバーシステム変数」で簡単に説明しています。

クエリーキャッシュのサイズを設定するには、query_cache_size システム変数を設定します。それを 0 に設定すると、query_cache_type=0 を設定するのと同様に、クエリーキャッシュが無効になります。デフォルトでは、クエリーキャッシュは無効化されます。これは 1M のデフォルトのサイズと、0 の query_cache_type のデフォルトを使用して実現されます。(MySQL 5.6.8 より前では、1 のデフォルトの query_cache_type で、デフォルトのサイズは 0 です。)

オーバーヘッドを大幅に削減するには、クエリーキャッシュを使用しない場合に query_cache_type=0 でサーバーも起動します。

> **注記**
>
> Windows Configuration Wizard を使用して、MySQL をインストールまたは構成する場合、query_cache_size のデフォルト値が、使用可能なさまざまな構成の種類に基づいて、自動的に構成されます。Windows Configuration Wizard を使用する場合、選択した構成のため、クエリーキャッシュが有効になる (つまり、ゼロではない値に設定される) ことがあります。クエリーキャッシュは、query_cache_type 変数の設定によっても制御されます。構成が行われたあとに、my.ini ファイルに設定されたこれらの変数の値をチェックしてください。

query_cache_size をゼロ以外の値に設定する場合は、その構造を割り当てるために、クエリーキャッシュに約 40KB の最小サイズが必要であることを覚えておいてください。(正確なサイズはシステムアーキテクチャーによります。)小さすぎる値を設定すると、この例のように警告を受け取ります。

```
mysql> SET GLOBAL query_cache_size = 40000;
Query OK, 0 rows affected, 1 warning (0.00 sec)

mysql> SHOW WARNINGS\G
*************************** 1. row ***************************
  Level: Warning
   Code: 1282
Message: Query cache failed to set size 39936;
         new query cache size is 0

mysql> SET GLOBAL query_cache_size = 41984;
Query OK, 0 rows affected (0.00 sec)

mysql> SHOW VARIABLES LIKE 'query_cache_size';
+------------------+-------+
| Variable_name    | Value |
+------------------+-------+
| query_cache_size | 41984 |
+------------------+-------+
```

クエリーキャッシュで実際にクエリー結果を保持できるようにするには、そのサイズを大きく設定する必要があります。

```
mysql> SET GLOBAL query_cache_size = 1000000;
Query OK, 0 rows affected (0.04 sec)

mysql> SHOW VARIABLES LIKE 'query_cache_size';
+------------------+--------+
| Variable_name    | Value  |
+------------------+--------+
| query_cache_size | 999424 |
+------------------+--------+
1 row in set (0.00 sec)
```

query_cache_size 値は、もっとも近い 1024 バイトブロックに調整されます。そのため、レポートされる値は、割り当てた値と異なることがあります。

クエリーキャッシュのサイズが 0 より大きい場合、query_cache_type 変数はその動作に影響します。この変数は次の値に設定できます。

- 0 または OFF の値は、キャッシュまたはキャッシュされた結果の取得を妨げます。
- 1 または ON の値は、SELECT SQL_NO_CACHE から始まるステートメントを除いて、キャッシュを有効にします。
- 2 または DEMAND の値は、SELECT SQL_CACHE で始まるステートメントのみをキャッシュさせます。

query_cache_size が 0 の場合、query_cache_type 変数も 0 に設定してください。この場合、サーバーはクエリーキャッシュ相互排他ロックをまったく獲得しません。これは、実行時にクエリーキャッシュを有効にできず、クエリー実行のオーバーヘッドが削減されることを意味します。

GLOBAL query_cache_type 値を設定すると、変更が行われたあとに接続するすべてのクライアントのクエリーキャッシュの動作が決定されます。SESSION query_cache_type 値を設定して、個々のクライアントでそれぞれ独自の接続のキャッシュ動作を制御できます。たとえば、クライアントは次のように独自のクエリーへのクエリーキャッシュの使用を無効にできます。

```
mysql> SET SESSION query_cache_type = OFF;
```

サーバーの起動時 (SET ステートメントによる実行時ではなく) に query_cache_type を設定する場合、数値のみが許可されます。

キャッシュ可能な個々のクエリー結果の最大サイズを制御するには、query_cache_limit システム変数を設定します。デフォルト値は 1M バイトです。

キャッシュを大きすぎるサイズに設定しないでください。更新時にキャッシュをロックするスレッドの必要性のため、きわめて大きいキャッシュでは ロックの競合問題が見られることがあります。

> **注記**
>
> コマンド行または構成ファイルに --maximum-query_cache_size=32M オプションを使用して、SET ステートメントで実行時にクエリーキャッシュに指定できる最大サイズを設定できます。

クエリーがキャッシュされるようにすると、その結果 (クライアントに送信されたデータ) が結果の取得時に、クエリーキャッシュに格納されます。そのため、データは通常 1 つの大きなまとまりで処理されません。クエリーキャッシュはオンデマンドでこのデータを格納するためのブロックを割り当てるため、1 つのブロックがいっぱいになると、新しいブロックが割り当てられます。メモリーの割り当て操作はコスト (時間的) がかかるため、クエリーキャッシュは query_cache_min_res_unit システム変数によって指定された最小サイズでブロックを割り当てます。クエリーが実行されると、未使用のメモリーが解放されるように、最後の結果ブロックが実際のデータサイズにトリミングされます。サーバーで実行するクエリーの種類によっては、query_cache_min_res_unit の値をチューニングすることが有効であるとわかる場合があります。

- query_cache_min_res_unit のデフォルト値は 4K バイトです。ほとんどの場合、これで十分であるはずです。
- 小さい結果の大量のクエリーがある場合、多数の空きブロックに示されるように、デフォルトのブロックサイズはメモリーの断片化につながることがあります。断片化は、メモリー不足のために、クエリーキャッシュにキャッシュからクエリーを強制的にプルーニング (削除) させる可能性があります。この場合、query_cache_min_res_unit の値を減らします。空きブロックと、プルーニングによって削除されたクエリーの数は Qcache_free_blocks および Qcache_lowmem_prunes ステータス変数の値によって得られます。
- ほとんどのクエリーの結果が大きい (Qcache_total_blocks および Qcache_queries_in_cache ステータス変数をチェックします) 場合、query_cache_min_res_unit を増やして、パフォーマンスを向上できます。ただし、大きくしすぎないようにしてください (前の項目を参照してください)。

8.9.3.4 クエリーキャッシュのステータスと保守

MySQL サーバーにクエリーキャッシュが存在するかどうかをチェックするには、次のステートメントを使用します。

```
mysql> SHOW VARIABLES LIKE 'have_query_cache';
+------------------+-------+
| Variable_name    | Value |
+------------------+-------+
| have_query_cache | YES   |
+------------------+-------+
```

FLUSH QUERY CACHE ステートメントによって、クエリーキャッシュをデフラグして、そのメモリーの利用を改善できます。このステートメントは、キャッシュからクエリーを削除しません。

RESET QUERY CACHE ステートメントは、クエリーキャッシュからすべてのクエリー結果を削除します。FLUSH TABLES ステートメントもこれを実行します。

クエリーキャッシュのパフォーマンスをモニターするには、SHOW STATUS を使用して、キャッシュのステータス変数を表示します。

```
mysql> SHOW STATUS LIKE 'Qcache%';
+-------------------------+--------+
| Variable_name           | Value  |
+-------------------------+--------+
| Qcache_free_blocks      | 36     |
| Qcache_free_memory      | 138488 |
| Qcache_hits             | 79570  |
| Qcache_inserts          | 27087  |
| Qcache_lowmem_prunes    | 3114   |
| Qcache_not_cached       | 22989  |
| Qcache_queries_in_cache | 415    |
| Qcache_total_blocks     | 912    |
+-------------------------+--------+
```

これらの各変数の説明は、セクション5.1.6「サーバーステータス変数」にあります。ここでは、それらのいくつかの使用方法について説明します。

SELECT クエリーの合計数は、次の式で得られます。

```
  Com_select
+ Qcache_hits
+ queries with errors found by parser
```

Com_select 値は次の式で得られます。

```
  Qcache_inserts
+ Qcache_not_cached
+ queries with errors found during the column-privileges check
```

クエリーキャッシュでは、可変長ブロックを使用するため、Qcache_total_blocks および Qcache_free_blocks に、クエリーキャッシュのメモリー断片化が示されることがあります。FLUSH QUERY CACHE 後には、空きブロックが 1 つだけ残ります。

キャッシュされるすべてのクエリーには、少なくとも 2 つのブロック (1 つはクエリーテキスト用で、1 つ以上はクエリー結果用) が必要です。さらに、クエリーで使用される各テーブルにも 1 つのブロックが必要です。ただし、複数のクエリーで同じテーブルを使用している場合、1 つのテーブルブロックだけを割り当てる必要があります。

Qcache_lowmem_prunes ステータス変数によって提供される情報は、クエリーキャッシュサイズをチューニングするのに役立つことがあります。それは、新しいクエリーのキャッシュのためにメモリーを解放するために、キャッシュから削除されたクエリーの数をカウントします。クエリーキャッシュは、LRU (Least Recently Used) 戦略を使用して、キャッシュから削除するクエリーを決定します。チューニング情報は、セクション8.9.3.3「クエリーキャッシュの構成」にあります。

8.9.4 プリペアドステートメントおよびストアドプログラムのキャッシュ

セッション中にクライアントが複数回実行する可能性がある特定のステートメントに対し、サーバーはステートメントを内部構造に変換し、実行時にその構造が使用されるようにキャッシュします。キャッシュによって、セッション中にそれが再度必要になった場合に、ステートメントを再変換するオーバーヘッドが避けられるため、サーバーはより効率的に実行できます。変換とキャッシュは、次のステートメントに対して行われます。

- SQL レベルで処理されるもの (PREPARE ステートメントを使用して) とバイナリクライアント/サーバープロトコルを使用して処理されるもの (mysql_stmt_prepare() C API 関数を使用して) の両方のプリペアドステートメント。max_prepared_stmt_count システム変数は、サーバーがキャッシュするステートメントの合計数を制御します。(すべてのセッションでのプリペアドステートメントの合計数。)
- ストアドプログラム (ストアドプロシージャーおよび関数、トリガー、およびイベント)。この場合、サーバーはプログラム本体全体を変換し、キャッシュします。stored_program_cache システム変数は、サーバーがセッションあたりにキャッシュするストアドプログラムのおおよその数を示します。

サーバーは、セッション単位でプリペアドステートメントおよびストアドプログラム用のキャッシュを保守します。1 つのセッションでキャッシュされたステートメントは、ほかのセッションからアクセスできません。セッションが終了すると、サーバーはそのためにキャッシュされたすべてのステートメントを破棄します。

サーバーがキャッシュされた内部ステートメント構造を使用する場合、構造が古くなっていないことに注意する必要があります。ステートメントによって使用されているオブジェクトにメタデータの変更があり、現在のオブジェクト定義と内部ステートメント構造で表されている定義に不一致が発生することがあります。メタデータの変更は、テーブルの作成、削除、変更、名前変更、切り捨てを行う DDL ステートメントや、テーブルの解析、最

適化、修復を行う DDL ステートメントなどに対して発生します。テーブルの内容の変更 (INSERT や UPDATE などによる) ではメタデータが変更されず、SELECT ステートメントも変更されません。

次にこの問題を説明します。クライアントがこのステートメントを準備するとします。

```
PREPARE s1 FROM 'SELECT * FROM t1';
```

SELECT * は内部構造からテーブル内のカラムのリストに展開します。テーブル内のカラムのセットが ALTER TABLE によって変更されている場合、プリペアドステートメントが古くなります。次回にクライアントが s1 を実行したときに、サーバーがこの変更を検出しない場合、プリペアドステートメントは正しくない結果を返します。

プリペアドステートメントによって参照されているテーブルやビューのメタデータの変更に原因がある問題を避けるため、サーバーはこれらの変更を検出し、次回の実行時にステートメントを自動的に再準備します。つまり、サーバーはステートメントを再解析し、内部構造を再構築します。再解析は、キャッシュ内に新しいエントリのための空きを作るために暗黙的に、または FLUSH TABLES によって明示的に、参照されているテーブルやビューがテーブル定義キャッシュからフラッシュされたあとにも行われます。

同様に、ストアドプログラムによって使用されているオブジェクトに変更が発生した場合、サーバーはプログラム内の影響のあるステートメントを再解析します。(MySQL 5.6.6 より前では、サーバーはストアドプログラムに影響するメタデータの変更を検出しないため、そのような変更によって、誤った結果やエラーが発生する可能性があります。)

サーバーは式内のオブジェクトのメタデータの変更も検出します。これらは、DECLARE CURSOR などのストアドプログラムに固有のステートメントや IF、CASE、および RETURN などのフロー制御ステートメントで使用できます。

ストアドプログラム全体の再解析を避けるため、サーバーは必要に応じて、プログラム内の影響のあるステートメントや式のみを再解析します。例:

- テーブルまたはビューのメタデータが変更されているとします。再解析は、テーブルやビューにアクセスするプログラム内の SELECT * に対して行われますが、テーブルやビューにアクセスしない SELECT * に対しては行われません。
- ステートメントが影響を受ける場合、サーバーは可能なかぎり部分的にのみそれを再解析します。この CASE ステートメントを考慮します。

  ```
  CASE case_expr
    WHEN when_expr1 ...
    WHEN when_expr2 ...
    WHEN when_expr3 ...
    ...
  END CASE
  ```

  メタデータの変更が WHEN when_expr3 にのみ影響する場合、その式が再解析されます。case_expr およびその他の WHEN 式は再解析されません。

再解析では、元の内部形式への変換に有効であったデフォルトのデータベースと SQL モードが使われます。

サーバーは最大 3 回再解析を試みます。すべての試みが失敗した場合、エラーが発生します。

再解析は自動ですが、それが行われた場合、プリペアドステートメントとストアドプログラムのパフォーマンスが低下します。

プリペアドステートメントの場合、Com_stmt_reprepare ステータス変数が再準備の数を追跡します。

8.10 ロック操作の最適化

MySQL はロックを使用して、テーブルの内容の競合を管理します。

- 内部ロックは、複数スレッドによるテーブルの内容の競合を管理するために、MySQL サーバー自体の内部で実行されます。この種類のロックは、完全にサーバーによって実行され、ほかのプログラムは関与しないため、内部です。セクション8.10.1「内部ロック方法」を参照してください。
- 外部ロックは、サーバーとほかのプログラム間で、どのプログラムがいつテーブルにアクセスできるかを調整するために、MyISAM テーブルファイルをロックする場合に発生します。セクション8.10.5「外部ロック」を参照してください。

8.10.1 内部ロック方法

このセクションでは、内部ロック、つまり複数のセッションによるテーブル内容の競合を管理するために、MySQL サーバー自体の内部で実行されるロックについて説明します。この種類のロックは、完全にサーバーによって実行され、ほかのプログラムは関与しないため、内部です。ほかのプログラムによって MySQL ファイルに対して実行されるロックについては、セクション8.10.5「外部ロック」を参照してください。

行レベルロック

MySQL は InnoDB テーブルに行レベルロックを使用して、複数のセッションによる同時書き込みアクセスをサポートし、それらを複数ユーザー、高度な並列性、および OLTP アプリケーションに適したものにします。

単一の InnoDB テーブルに対する複数の同時書き込み操作の実行時のデッドロックを避けるには、トランザクションのあとの方に DML ステートメントがある場合でも、変更が予想される行のグループごとに、SELECT ... FOR UPDATE ステートメントを発行して、トランザクションの開始時に必要なロックを獲得します。トランザクションで複数のテーブルを変更またはロックする場合、各トランザクション内で、該当するステートメントを同じ順序で発行します。InnoDB は自動的にデッドロック状況を検出し、影響のあるいずれかのトランザクションをロールバックするため、デッドロックは重大エラーを表すより、パフォーマンスに影響します。

行レベルロックの利点:

- 異なるセッションが異なる行にアクセスする場合、ロックの競合は少なくなります。
- ロールバックする変更が少なくなります。
- 1 つの行を長時間ロックできます。

テーブルレベルロック

MySQL は、MyISAM、MEMORY、および MERGE テーブルにテーブルレベルロックを使用して、一度に 1 つだけのセッションがそれらのテーブルを更新できるようにし、それらを読み取り専用、読み取りが大部分、または単一ユーザーのアプリケーションに適したものにします。

これらのストレージエンジンは、常にクエリーの最初に 1 回だけ必要なすべてのロックをリクエストし、常に同じ順序でテーブルをロックすることによって、デッドロックを回避します。このトレードオフは、この戦略では並列性が低くなることです。テーブルを変更したいほかのセッションは、現在の DML ステートメントが終了するまで待機する必要があります。

MySQL はテーブル書き込みロックを次のように許可します。

1. テーブルにロックがない場合、それを書き込みロックします。
2. そうでない場合、書き込みロックキューにロックリクエストを入れます。

MySQL はテーブル読み取りロックを次のように許可します。

1. テーブルに書き込みロックがない場合、それを読み取りロックします。
2. そうでない場合、読み取りロックキューにロックリクエストを入れます。

テーブルの更新は、テーブルの取得よりも高い優先度が与えられます。そのため、ロックが解放されると、ロックは書き込みロックキュー内のリクエストに使用できるようになり、次に読み取りロックキュー内のリクエストに使用できるようになります。これにより、テーブルに対して重い SELECT アクティビティーがある場合でも、テーブルに対する更新が「不足」することはありません。ただし、テーブルに対して多くの更新がある場合、SELECT ステートメントは更新がなくなるまで待機します。

読み取りと書き込みの優先度を変更する方法については、セクション8.10.2「テーブルロックの問題」を参照してください。

Table_locks_immediate および Table_locks_waited ステータス変数をチェックすることでシステム上のテーブルロック競合を分析できます。これらは、テーブルロックのリクエストがすぐに許可された回数と待機する必要があった回数を示します。

```
mysql> SHOW STATUS LIKE 'Table%';
+-----------------------+---------+
| Variable_name         | Value   |
+-----------------------+---------+
| Table_locks_immediate | 1151552 |
| Table_locks_waited    | 15324   |
+-----------------------+---------+
```

MyISAM ストレージエンジンでは、特定のテーブルのリーダーとライター間の競合を軽減するために、同時挿入をサポートしています。MyISAM テーブルでデータファイルの途中に空きブロックがない場合、行は常にデー

タファイルの末尾に挿入されます。この場合、ロックなしで MyISAM テーブルに対して同時 INSERT および SELECT ステートメントを自由に組み合わせることができます。つまり、ほかのクライアントが MyISAM テーブルから読み取ると同時に、それに行を挿入できます。テーブルの途中で行が削除されるか更新されると、隙間が発生します。隙間がある場合、同時挿入は無効にされますが、すべての隙間が新しいデータで埋められた場合は、自動的にふたたび有効にされます。この動作は concurrent_insert システム変数によって変更します。セクション8.10.3「同時挿入」を参照してください。

LOCK TABLES で明示的にテーブルロックを獲得する場合、READ ロックではなく READ LOCAL ロックをリクエストして、テーブルをロックしている間に、ほかのセッションが同時挿入を実行できるようにできます。

同時挿入が可能でない場合に、テーブル real_table に対して多くの INSERT および SELECT 操作を実行するには、一時テーブル temp_table に行を挿入し、定期的に一時テーブルからの行で実際のテーブルを更新します。これは次のコードで実行できます。

```
mysql> LOCK TABLES real_table WRITE, temp_table WRITE;
mysql> INSERT INTO real_table SELECT * FROM temp_table;
mysql> DELETE FROM temp_table;
mysql> UNLOCK TABLES;
```

テーブルレベルロックの利点:

- 必要なメモリーが比較的少なくなります。
- 単一のロックだけが必要であるため、テーブルの大部分に対して使用する場合に高速です。
- データの大部分に対して GROUP BY 操作を頻繁に実行する場合や、テーブル全体を頻繁にスキャンする必要がある場合に高速です。

一般にテーブルロックは次の場合に適しています。

- テーブルに対するほとんどのステートメントが読み取りです。
- テーブルに対するステートメントが読み取りと書き込みの組み合わせであり、そのうち書き込みは 1 つのキーの読み取りでフェッチできる単一の行に対する更新または削除です。

  ```
  UPDATE tbl_name SET column=value WHERE unique_key_col=key_value;
  DELETE FROM tbl_name WHERE unique_key_col=key_value;
  ```

- 同時 INSERT ステートメントとごく少数の UPDATE または DELETE ステートメントと組み合わされた SELECT。
- ライターを使用しない、テーブル全体への多くのスキャンまたは GROUP BY 操作。

8.10.2 テーブルロックの問題

InnoDB テーブルでは、複数のセッションとアプリケーションが互いに待機したり、不整合の結果を生成したりすることなく、同じテーブルに対して同時に読み取りや書き込みを実行できるように、行レベルロックを使用します。このストレージエンジンでは、LOCK TABLES ステートメントは特別な保護を提供せず、代わりに並列性が低くなるため、この使用を避けてください。自動の行レベルロックにより、これらのテーブルがもっとも重要なデータを格納するもっともビジーなデータベースに適合し、同時にテーブルのロックやロック解除が必要ないためアプリケーションロジックが簡単になります。その結果、InnoDB ストレージエンジンは MySQL 5.6 のデフォルトです。

MySQL は InnoDB を除く、すべてのストレージエンジンに対して、テーブルロック (ページ、行、またはカラムロックの代わりに) を使用します。ロック操作自体には、あまりオーバーヘッドがありません。ただし、一度に 1 つのセッションしかテーブルに書き込むことができないため、これらのほかのストレージエンジンでの最高のパフォーマンスのため、頻繁にクエリーされ、めったに挿入または更新されないテーブルに対して主にそれらを使用します。

InnoDB を優先するパフォーマンスの考慮事項

テーブルを作成するために、InnoDB を使用するか、別のストレージエンジンを使用するかを選択する場合、テーブルロックの次の短所を考慮してください。

- テーブルロックにより、多くのセッションを同時にテーブルから読み取ることができますが、セッションでテーブルに書き込む必要がある場合、まず排他的アクセスを取得する必要がありますが、これはまずほかのセッションがテーブルを処理し終わるのを待つ必要がある可能性があることを意味します。更新中、この特定のテーブルにアクセスしようとするほかのすべてのセッションは、更新が完了するまで待機する必要があります。

- ディスクがいっぱいで、セッションを続行するには空き領域が使用できるようになる必要があるため、セッションが待機している場合にテーブルロックによって問題が発生します。この場合、問題のテーブルにアクセスしようとするすべてのセッションが、より多くのディスク領域が使用できるようになるまで待機状態になります。
- 実行に長時間かかる SELECT ステートメントにより、その間ほかのセッションのテーブルの更新が妨げられ、ほかのセッションが遅くなり、応答していないように見えます。セッションが更新のためにテーブルへの排他アクセスを取得するのを待機している間、SELECT ステートメントを発行するほかのセッションはそのあとに列をなし、読み取りセッションでも並列性が低くなります。

ロックパフォーマンスの問題の回避

次の項目では、テーブルロックによって発生する競合を回避または軽減するいくつかの方法について説明します。

- セットアップ時に CREATE TABLE ... ENGINE=INNODB を使用するか、既存のテーブルに対して ALTER TABLE ... ENGINE=INNODB を使用して、テーブルを InnoDB ストレージエンジンに切り替えることを考慮します。このストレージエンジンの詳細については、第14章「InnoDB ストレージエンジン」を参照してください。
- テーブルをロックする時間が短くなるように、SELECT ステートメントを最適化して、実行を高速化します。これを実行するには、いくつかのサマリーテーブルを作成する必要がある場合があります。
- --low-priority-updates で mysqld を起動します。テーブルレベルロックのみを使用するストレージエンジン (MyISAM、MEMORY、および MERGE など) の場合、これにより、テーブルを更新 (変更) するすべてのステートメントに SELECT ステートメントより低い優先度を与えます。この場合、先述のシナリオの 2 つめの SELECT ステートメントは UPDATE ステートメントの前に実行され、最初の SELECT の終了を待機しません。
- 特定の接続で発行されたすべての更新を低い優先度で実行させるように指定するには、low_priority_updates サーバーシステム変数を 1 に等しく設定します。
- 特定の INSERT、UPDATE、または DELETE ステートメントに低い優先度を与えるには、LOW_PRIORITY 属性を使用します。
- 特定の SELECT ステートメントに高い優先度を与えるには、HIGH_PRIORITY 属性を使用します。セクション13.2.9「SELECT 構文」を参照してください。
- max_write_lock_count システム変数の値を低くして mysqld を開始し、テーブルに対する特定の数の挿入が行われたあとにテーブルを待機しているすべての SELECT ステートメントの優先度を一時的に強制的に高めます。これにより、特定の数の WRITE ロックのあとの READ ロックが許可されます。
- SELECT と組み合わせた INSERT に問題がある場合は、同時 SELECT ステートメントと INSERT ステートメントをサポートする MyISAM テーブルに切り替えることを考慮します。(セクション8.10.3「同時挿入」を参照してください。)
- 同じ非トランザクションテーブルに対して挿入と削除を組み合わせる場合、INSERT DELAYED が役立つことがあります。セクション13.2.5.2「INSERT DELAYED 構文」を参照してください。

  > **注記**
  >
  > MySQL 5.6.6 現在、INSERT DELAYED は非推奨であり、将来のリリースで削除されます。代わりに INSERT (DELAYED を付けない) を使用してください。

- 組み合わされた SELECT と DELETE ステートメントに問題がある場合、DELETE への LIMIT オプションが役立つことがあります。セクション13.2.2「DELETE 構文」を参照してください。
- SELECT ステートメントで SQL_BUFFER_RESULT を使用すると、テーブルロックの時間の短縮に役立つことがあります。セクション13.2.9「SELECT 構文」を参照してください。
- テーブルの内容を個別のテーブルに分割すると (クエリーを 1 つのテーブルのカラムに対して実行し、更新を別のテーブルのカラムに制限することによって)、役立つことがあります。
- 単一のキューを使用するように、mysys/thr_lock.c のロックコードを変更できます。この場合、書き込みロックと読み取りロックは同じ優先度を持ち、一部のアプリケーションに役立つことがあります。

8.10.3 同時挿入

MyISAM ストレージエンジンでは、特定のテーブルに対する読み取りと書き込みの競合を軽減するために、同時挿入をサポートしています。MyISAM テーブルのデータファイルに隙間 (途中の削除された行) がない場合、SELECT ステートメントがテーブルの行を読み取るのと同時に、INSERT ステートメントを実行してテーブルの末尾に行を追加できます。複数の INSERT ステートメントがある場合、それらはキューに入れられ、SELECT ステートメントと同時に順番に実行されます。同時 INSERT の結果はすぐに見られないことがあります。

concurrent_insert システム変数を設定して、同時挿入の処理を変更できます。デフォルトで、変数は AUTO (または 1) に設定され、同時挿入が先述のように処理されます。concurrent_insert が NEVER (または 0) に設定されている場合、同時挿入は無効にされます。変数が ALWAYS (または 2) に設定されている場合、行が削除されたテーブルに対してもテーブルの末尾での同時挿入が許可されます。concurrent_insert システム変数の説明も参照してください。

同時挿入を使用できる状況下では、INSERT ステートメントの DELAYED 修飾子を使用する必要はほとんどありません。セクション13.2.5.2「INSERT DELAYED 構文」を参照してください。

バイナリログを使用している場合、同時挿入は CREATE ... SELECT または INSERT ... SELECT ステートメントの通常の挿入に変換されます。これは、バックアップ操作中にログを適用することでテーブルの正確なコピーを確実に再作成できるようにするために行われます。セクション5.2.4「バイナリログ」を参照してください。また、これらのステートメントに対しては、選択元のテーブルへの挿入がブロックされるように、そのテーブルに読み取りロックが設定されます。その結果、そのテーブルに対する同時挿入も待機する必要があります。

LOAD DATA INFILE で同時挿入の条件 (つまり、途中に空きブロックを含まない) を満たす MyISAM テーブルを使用して CONCURRENT を指定する場合、ほかのセッションは LOAD DATA の実行中にテーブルからデータを取得できます。CONCURRENT オプションの使用は、同時にテーブルを使用しているほかのセッションがない場合でも、LOAD DATA のパフォーマンスに多少の影響があります。

HIGH_PRIORITY を指定すると、サーバーが --low-priority-updates オプションで起動されている場合に、その効果がオーバーライドされます。また、同時挿入も使用されなくなります。

LOCK TABLE の場合、READ LOCAL と READ の違いは READ LOCAL が、ロックが保持されている間に、競合していない INSERT ステートメント (同時挿入) の実行を許可することです。ただし、ロックを保持している間にサーバーの外部のプロセスを使用してデータベースを操作する場合、これを使用することはできません。

8.10.4 メタデータのロック

MySQL はメタデータのロックを使用して、オブジェクト (テーブル、トリガーなど) へのアクセスを管理します。メタデータのロックは、データの一貫性を確保するために使用されますが、いくらかのオーバーヘッドがあり、クエリーボリュームが増えるとそれも大きくなります。複数のクエリーが同じオブジェクトにアクセスを試みることが多くなるほど、メタデータの競合が増加します。

メタデータのロックは、テーブル定義キャッシュの代替ではなく、その相互排他ロックとロックは、LOCK_open 相互排他ロックと異なります。次の説明では、メタデータのロックの仕組みに関する情報を提供します。

トランザクションのシリアライザビリティーを確保するため、サーバーは、別のセッションで、未完了の明示的または暗黙的に開始されたトランザクションで使用されているテーブルに対して、セッションがデータ定義言語 (DDL) ステートメントを実行することを許可してはいけません。サーバーは、トランザクション内で使用されているテーブルに対してメタデータロックを獲得し、トランザクションが終了するまでそれらのロックの解放を延期させることによって、これを実現します。テーブルへのメタデータロックは、テーブルの構造への変更を妨げます。このロックアプローチには、あるセッション内のトランザクションによって使用されているテーブルは、トランザクションが終了するまで、ほかのセッションによって DDL ステートメントで使用できないという問題があります。

この原則は、トランザクションテーブルだけでなく、非トランザクションテーブルにも適用されます。あるセッションがトランザクションテーブル t と非トランザクションテーブル nt を次のように使用するトランザクションを開始するとします。

```
START TRANSACTION;
SELECT * FROM t;
SELECT * FROM nt;
```

サーバーはトランザクションが終了するまで、t と nt の両方に対するメタデータロックを保持します。別のセッションがいずれかのテーブルに対して、DDL または書き込みロック操作を試みると、それはトランザクションの終了時にメタデータロックが解放されるまでブロックされます。たとえば、2 つめのセッションはこれらのいずれかの操作を試みるとブロックされます。

```
DROP TABLE t;
ALTER TABLE t ...;
DROP TABLE nt;
```

```
ALTER TABLE nt ...;
LOCK TABLE t ... WRITE;
```

サーバーが構文上有効であるが、実行中に失敗するステートメントのメタデータロックを獲得した場合、そのロックを早期に解放しません。失敗したステートメントがバイナリログに書き込まれ、ロックによってログの一貫性が保護されるため、ロックの解放はまだトランザクションの終了まで延期されます。

自動コミットモードでは、各ステートメントが事実上完全なトランザクションであるため、そのステートメントに対して獲得されたメタデータロックは、ステートメントの終了までしか保持されません。

PREPARE ステートメント中に獲得されたメタデータロックは、準備が複数ステートメントトランザクション内で行われる場合でも、ステートメントが準備されると解放されます。

MySQL 5.5 より前では、トランザクションがステートメント内で使用されているテーブルのメタデータロックと同等のロックを獲得した場合、ステートメントの終了時にロックを解放していました。このアプローチには、アクティブなトランザクションで別のセッションによって使用されているテーブルに対して、DDL ステートメントが実行された場合、ステートメントは誤った順序でバイナリログに書き込まれる可能性があるという欠点がありました。

8.10.5 外部ロック

外部ロックは、複数のプロセスによる MyISAM データベーステーブルの競合を管理するためのファイルシステムロックの使用です。外部ロックは、MySQL サーバーなどの単一のプロセスが、テーブルへのアクセスを必要とする唯一のプロセスであると想定できない状況で使用されます。次にいくつかの例を示します。

- 同じデータベースディレクトリを使用する複数のサーバーを実行する場合 (推奨されません)、各サーバーで外部ロックが有効になっている必要があります。
- myisamchk を使用して MyISAM テーブルに対して保守操作を実行する場合、サーバーが実行中でないことを確認するか、サーバーが必要に応じてテーブルファイルをロックし、テーブルへのアクセスを myisamchk によって調整するように、サーバーで外部ロックが有効になっていることを確認する必要があります。同じことが、MyISAM テーブルをパックするために myisampack を使用する場合にも当てはまります。

  外部ロックを有効にしてサーバーを実行する場合、テーブルのチェックなどの読み取り操作のために、いつでも myisamchk を使用できます。この場合に、サーバーが myisamchk で使用しているテーブルを更新しようとする場合、サーバーは myisamchk が終了するまで待ってから、続行します。

  テーブルの修復や最適化などの書き込み操作のために myisamchk を使用する場合、または myisampack を使用してテーブルをパックする場合は、mysqld サーバーがそのテーブルを使用していないことを常に確認する必要があります。mysqld を停止しない場合、myisamchk を実行する前に、少なくとも mysqladmin flush-tables を実行してください。サーバーと myisamchk が同時にテーブルにアクセスすると、テーブルが破損する可能性があります。

外部ロックが有効になっていると、テーブルへのアクセスを必要とする各プロセスは、テーブルへのアクセスに進む前にテーブルファイルに対するファイルシステムロックを獲得します。必要なすべてのロックを獲得できない場合、(現在ロックを保持しているプロセスがそれらを解放したあとに) ロックを取得できるまで、プロセスはテーブルへのアクセスをブロックされます。

サーバーは場合によってテーブルにアクセスできるまでほかのプロセスを待機する必要があるため、外部ロックはサーバーのパフォーマンスに影響します。

単一のサーバーを実行して特定のデータディレクトリにアクセスする場合 (これは通常のケースです) およびサーバーの実行中に myisamchk などのほかのプログラムでテーブルを変更する必要がない場合、外部ロックは不要です。ほかのプログラムでテーブルを読み取るだけである場合、外部ロックは不要ですが、myisamchk がテーブルを読み取っている間にサーバーがテーブルを変更すると、myisamchk が警告をレポートすることがあります。

外部ロックが無効になっていて、myisamchk を使用するには、myisamchk の実行中にサーバーを停止するか、myisamchk を実行する前にテーブルをロックし、フラッシュする必要があります。(セクション8.11.1「システム要素およびスタートアップパラメータのチューニング」を参照してください。)この要件を回避するには、CHECK TABLE および REPAIR TABLE ステートメントを使用して、MyISAM テーブルをチェックし、修復します。

mysqld の場合、外部ロックは skip_external_locking システム変数の値で制御されます。この変数が有効にされている場合、外部ロックは無効になり、その逆も同じです。MySQL 4.0 以降、外部ロックはデフォルトで無効にされます。

外部ロックの使用は、サーバーの起動時に --external-locking または --skip-external-locking オプションを使用して制御できます。

多数の MySQL プロセスから MyISAM テーブルを更新できるようにするために外部ロックオプションを使用する場合、次の条件を満たしていることを確認する必要があります。

- 別のプロセスによって更新されるテーブルを使用するクエリーには、クエリーキャッシュを使用しないでください。
- サーバーを --delay-key-write=ALL オプションで起動したり、共有テーブルに対して DELAY_KEY_WRITE=1 テーブルオプションを使用したりしないでください。そうでないと、インデックスが破損する可能性があります。

これらの条件をもっとも簡単に満たす方法は、常に --external-locking を --delay-key-write=OFF および --query-cache-size=0 と一緒に使用することです。(これは、多くのセットアップで、前述のオプションを組み合わせることが有用であるため、デフォルトで実行されません。)

8.11 MySQL サーバーの最適化

このセクションでは、主に SQL ステートメントのチューニングではなくシステム構成を扱うデータベースサーバーの最適化技法について説明します。このセクションの情報は、管理しているサーバー全体のパフォーマンスとスケーラビリティーを確保したい DBA、データベースのセットアップを含むインストールスクリプトを構築する開発者、および生産性を最大にしたいと考え、開発、テストなどのために自分自身で MySQL を実行しているユーザーに適しています。

8.11.1 システム要素およびスタートアップパラメータのチューニング

大幅なパフォーマンスの向上を実現するためには、システムレベルの要素の一部を早急に決定する必要があるため、その要素から始めます。ほかの場合は、このセクションをざっと目を通すだけで十分かもしれません。ただし、このレベルで適用する要素を変更することで、どの程度改善できるかの感覚をつかんでおくことは常に望ましいと考えられます。

本番環境で MySQL を使用する前に、目的のプラットフォームでそれをテストすることをお勧めします。

ほかのヒント:

- RAM が十分にある場合は、すべてのスワップデバイスを取り外すことができます。オペレーティングシステムによっては、空きメモリーがある場合でも、特定のコンテキストで、スワップデバイスが使用されることがあります。
- MyISAM テーブルの外部ロックを避けます。MySQL 4.0 以降、すべてのシステムで外部ロックはデフォルトで無効にされています。--external-locking と --skip-external-locking オプションは外部ロックを明示的に有効および無効にします。

  外部ロックを無効にしても、1 台のサーバーしか実行していないかぎり、MySQL の機能に影響しません。myisamchk を実行する前にサーバーを停止する (または関連するテーブルをロックしてフラッシュする) ことを忘れないでください。一部のシステムでは外部ロックが機能しないため、無効にする必要があります。

  外部ロックを無効にできない唯一のケースは、同じデータに対して複数の MySQL サーバー (クライアントではない) を実行している場合、またはサーバーにまずテーブルをフラッシュしてロックするように伝えずに、myisamchk を実行してテーブルをチェックする (修復しない) 場合です。MySQL Cluster を使用している場合を除き、複数の MySQL サーバーを使用して、同じデータに同時にアクセスすることは一般に推奨されません。

  > **注記**
  >
  > MySQL Cluster は現在 MySQL 5.6 でサポートされていません。MySQL Cluster のアップグレードを希望するユーザーは、代わりに MySQL Cluster NDB 7.3 に移行してください。これは MySQL 5.6 に基づいていますが、最新の NDB の改善と修正が含まれています。

  LOCK TABLES および UNLOCK TABLES ステートメントは内部ロックを使用するため、外部ロックが無効にされている場合でもそれらを使用できます。

8.11.2 サーバーパラメータのチューニング

次のコマンドを使用して、mysqld サーバーで使用されるデフォルトのバッファーサイズを判断できます。

```
shell> mysqld --verbose --help
```

このコマンドによって、すべての mysqld オプションと構成可能なシステム変数のリストが生成されます。この出力には、デフォルトの変数値も含まれ、次のように見えます。

```
abort-slave-event-count                         0
allow-suspicious-udfs                           FALSE
archive                                         ON
auto-increment-increment                        1
auto-increment-offset                           1
autocommit                                      TRUE
automatic-sp-privileges                         TRUE
back-log                                        80
basedir                                         /home/jon/bin/mysql-5.6/
big-tables                                      FALSE
bind-address                                    *
binlog-cache-size                               32768
binlog-checksum                                 CRC32
binlog-direct-non-transactional-updates         FALSE
binlog-format                                   STATEMENT
binlog-max-flush-queue-time                     0
binlog-order-commits                            TRUE
binlog-row-event-max-size                       8192
binlog-row-image                                FULL
binlog-rows-query-log-events                    FALSE
binlog-stmt-cache-size                          32768
blackhole                                       ON
bulk-insert-buffer-size                         8388608
character-set-client-handshake                  TRUE
character-set-filesystem                        binary
character-set-server                            latin1
character-sets-dir                              /home/jon/bin/mysql-5.6/share/charsets/
chroot                                          (No default value)
collation-server                                latin1_swedish_ci
completion-type                                 NO_CHAIN
concurrent-insert                               AUTO
connect-timeout                                 10
console                                         FALSE
datadir                                         (No default value)
date-format                                     %Y-%m-%d
datetime-format                                 %Y-%m-%d %H:%i:%s
default-storage-engine                          InnoDB
default-time-zone                               (No default value)
default-tmp-storage-engine                      InnoDB
default-week-format                             0
delay-key-write                                 ON
delayed-insert-limit                            100
delayed-insert-timeout                          300
delayed-queue-size                              1000
des-key-file                                    (No default value)
disconnect-on-expired-password                  TRUE
disconnect-slave-event-count                    0
div-precision-increment                         4
end-markers-in-json                             FALSE
enforce-gtid-consistency                        FALSE
eq-range-index-dive-limit                       10
event-scheduler                                 OFF
expire-logs-days                                0
explicit-defaults-for-timestamp                 FALSE
external-locking                                FALSE
flush                                           FALSE
flush-time                                      0
ft-boolean-syntax                               + -><()~*:""&|
ft-max-word-len                                 84
ft-min-word-len                                 4
ft-query-expansion-limit                        20
ft-stopword-file                                (No default value)
gdb                                             FALSE
general-log                                     FALSE
general-log-file                                /home/jon/bin/mysql-5.6/data/havskatt.log
group-concat-max-len                            1024
gtid-mode                                       OFF
help                                            TRUE
host-cache-size                                 279
ignore-builtin-innodb                           FALSE
init-connect
init-file                                       (No default value)
init-slave
innodb                                          ON
innodb-adaptive-flushing                        TRUE
innodb-adaptive-flushing-lwm                    10
```

```
innodb-adaptive-hash-index                TRUE
innodb-adaptive-max-sleep-delay           150000
innodb-additional-mem-pool-size           8388608
innodb-api-bk-commit-interval             5
innodb-api-disable-rowlock                FALSE
innodb-api-enable-binlog                  FALSE
innodb-api-enable-mdl                     FALSE
innodb-api-trx-level                      0
innodb-autoextend-increment               64
innodb-autoinc-lock-mode                  1
innodb-buffer-page                        ON
innodb-buffer-page-lru                    ON
innodb-buffer-pool-dump-at-shutdown       FALSE
innodb-buffer-pool-dump-now               FALSE
innodb-buffer-pool-filename               ib_buffer_pool
innodb-buffer-pool-instances              0
innodb-buffer-pool-load-abort             FALSE
innodb-buffer-pool-load-at-startup        FALSE
innodb-buffer-pool-load-now               FALSE
innodb-buffer-pool-size                   134217728
innodb-buffer-pool-stats                  ON
innodb-change-buffer-max-size             25
innodb-change-buffering                   all
innodb-checksum-algorithm                 innodb
innodb-checksums                          TRUE
innodb-cmp                                ON
innodb-cmp-per-index                      ON
innodb-cmp-per-index-enabled              FALSE
innodb-cmp-per-index-reset                ON
innodb-cmp-reset                          ON
innodb-cmpmem                             ON
innodb-cmpmem-reset                       ON
innodb-commit-concurrency                 0
innodb-compression-failure-threshold-pct  5
innodb-compression-level                  6
innodb-compression-pad-pct-max            50
innodb-concurrency-tickets                5000
innodb-data-file-path                     (No default value)
innodb-data-home-dir                      (No default value)
innodb-disable-sort-file-cache            FALSE
innodb-doublewrite                        TRUE
innodb-fast-shutdown                      1
innodb-file-format                        Antelope
innodb-file-format-check                  TRUE
innodb-file-format-max                    Antelope
innodb-file-io-threads                    4
innodb-file-per-table                     TRUE
innodb-flush-log-at-timeout               1
innodb-flush-log-at-trx-commit            1
innodb-flush-method                       (No default value)
innodb-flush-neighbors                    1
innodb-flushing-avg-loops                 30
innodb-force-load-corrupted               FALSE
innodb-force-recovery                     0
innodb-ft-aux-table                       (No default value)
innodb-ft-being-deleted                   ON
innodb-ft-cache-size                      8000000
innodb-ft-config                          ON
innodb-ft-default-stopword                ON
innodb-ft-deleted                         ON
innodb-ft-enable-diag-print               FALSE
innodb-ft-enable-stopword                 TRUE
innodb-ft-index-cache                     ON
innodb-ft-index-table                     ON
innodb-ft-inserted                        ON
innodb-ft-max-token-size                  84
innodb-ft-min-token-size                  3
innodb-ft-num-word-optimize               2000
innodb-ft-server-stopword-table           (No default value)
innodb-ft-sort-pll-degree                 2
innodb-ft-user-stopword-table             (No default value)
innodb-io-capacity                        200
innodb-io-capacity-max                    18446744073709551615
innodb-large-prefix                       FALSE
innodb-lock-wait-timeout                  50
innodb-lock-waits                         ON
innodb-locks                              ON
innodb-locks-unsafe-for-binlog            FALSE
innodb-log-buffer-size                    8388608
innodb-log-compressed-pages               TRUE
```

```
innodb-log-file-size                  50331648
innodb-log-files-in-group             2
innodb-log-group-home-dir             (No default value)
innodb-lru-scan-depth                 1024
innodb-max-dirty-pages-pct            75
innodb-max-dirty-pages-pct-lwm        0
innodb-max-purge-lag                  0
innodb-max-purge-lag-delay            0
innodb-metrics                        ON
innodb-mirrored-log-groups            1
innodb-monitor-disable                (No default value)
innodb-monitor-enable                 (No default value)
innodb-monitor-reset                  (No default value)
innodb-monitor-reset-all              (No default value)
innodb-old-blocks-pct                 37
innodb-old-blocks-time                1000
innodb-online-alter-log-max-size      134217728
innodb-open-files                     0
innodb-optimize-fulltext-only         FALSE
innodb-page-size                      16384
innodb-print-all-deadlocks            FALSE
innodb-purge-batch-size               300
innodb-purge-threads                  1
innodb-random-read-ahead              FALSE
innodb-read-ahead-threshold           56
innodb-read-io-threads                4
innodb-read-only                      FALSE
innodb-replication-delay              0
innodb-rollback-on-timeout            FALSE
innodb-rollback-segments              128
innodb-sort-buffer-size               1048576
innodb-spin-wait-delay                6
innodb-stats-auto-recalc              TRUE
innodb-stats-method                   nulls_equal
innodb-stats-on-metadata              FALSE
innodb-stats-persistent               TRUE
innodb-stats-persistent-sample-pages  20
innodb-stats-sample-pages             8
innodb-stats-transient-sample-pages   8
innodb-status-file                    FALSE
innodb-strict-mode                    FALSE
innodb-support-xa                     TRUE
innodb-sync-array-size                1
innodb-sync-spin-loops                30
innodb-sys-columns                    ON
innodb-sys-datafiles                  ON
innodb-sys-fields                     ON
innodb-sys-foreign                    ON
innodb-sys-foreign-cols               ON
innodb-sys-indexes                    ON
innodb-sys-tables                     ON
innodb-sys-tablespaces                ON
innodb-sys-tablestats                 ON
innodb-table-locks                    TRUE
innodb-thread-concurrency             0
innodb-thread-sleep-delay             10000
innodb-trx                            ON
innodb-undo-directory                 .
innodb-undo-logs                      128
innodb-undo-tablespaces               0
innodb-use-native-aio                 TRUE
innodb-use-sys-malloc                 TRUE
innodb-write-io-threads               4
interactive-timeout                   28800
join-buffer-size                      262144
keep-files-on-create                  FALSE
key-buffer-size                       8388608
key-cache-age-threshold               300
key-cache-block-size                  1024
key-cache-division-limit              100
language                              /home/jon/bin/mysql-5.6/share/
large-pages                           FALSE
lc-messages                           en_US
lc-messages-dir                       /home/jon/bin/mysql-5.6/share/
lc-time-names                         en_US
local-infile                          TRUE
lock-wait-timeout                     31536000
log-bin                               (No default value)
log-bin-index                         (No default value)
log-bin-trust-function-creators       FALSE
```

```
log-bin-use-v1-row-events                       FALSE
log-error
log-isam                                        myisam.log
log-output                                      FILE
log-queries-not-using-indexes                   FALSE
log-raw                                         FALSE
log-short-format                                FALSE
log-slave-updates                               FALSE
log-slow-admin-statements                       FALSE
log-slow-slave-statements                       FALSE
log-tc                                          tc.log
log-tc-size                                     24576
log-throttle-queries-not-using-indexes          0
log-warnings                                    1
long-query-time                                 10
low-priority-updates                            FALSE
lower-case-table-names                          0
master-info-file                                master.info
master-info-repository                          FILE
master-retry-count                              86400
master-verify-checksum                          FALSE
max-allowed-packet                              4194304
max-binlog-cache-size                           18446744073709547520
max-binlog-dump-events                          0
max-binlog-size                                 1073741824
max-binlog-stmt-cache-size                      18446744073709547520
max-connect-errors                              100
max-connections                                 151
max-delayed-threads                             20
max-error-count                                 64
max-heap-table-size                             16777216
max-join-size                                   18446744073709551615
max-length-for-sort-data                        1024
max-prepared-stmt-count                         16382
max-relay-log-size                              0
max-seeks-for-key                               18446744073709551615
max-sort-length                                 1024
max-sp-recursion-depth                          0
max-tmp-tables                                  32
max-user-connections                            0
max-write-lock-count                            18446744073709551615
memlock                                         FALSE
metadata-locks-cache-size                       1024
metadata-locks-hash-instances                   8
min-examined-row-limit                          0
multi-range-count                               256
myisam-block-size                               1024
myisam-data-pointer-size                        6
myisam-max-sort-file-size                       9223372036853727232
myisam-mmap-size                                18446744073709551615
myisam-recover-options                          OFF
myisam-repair-threads                           1
myisam-sort-buffer-size                         8388608
myisam-stats-method                             nulls_unequal
myisam-use-mmap                                 FALSE
net-buffer-length                               16384
net-read-timeout                                30
net-retry-count                                 10
net-write-timeout                               60
new                                             FALSE
old                                             FALSE
old-alter-table                                 FALSE
old-passwords                                   0
old-style-user-limits                           FALSE
open-files-limit                                1024
optimizer-prune-level                           1
optimizer-search-depth                          62
optimizer-switch                                index_merge=on,index_merge_union=on,index_merge_sort_union=on,index_merge_intersection=on,engine_condition
optimizer-trace
optimizer-trace-features                        greedy_search=on,range_optimizer=on,dynamic_range=on,repeated_subselect=on
optimizer-trace-limit                           1
optimizer-trace-max-mem-size                    16384
optimizer-trace-offset                          -1
partition                                       ON
performance-schema                              TRUE
performance-schema-accounts-size                -1
performance-schema-consumer-events-stages-current       FALSE
performance-schema-consumer-events-stages-history       FALSE
performance-schema-consumer-events-stages-history-long  FALSE
performance-schema-consumer-events-statements-current   TRUE
```

```
performance-schema-consumer-events-statements-history      FALSE
performance-schema-consumer-events-statements-history-long FALSE
performance-schema-consumer-events-waits-current           FALSE
performance-schema-consumer-events-waits-history           FALSE
performance-schema-consumer-events-waits-history-long      FALSE
performance-schema-consumer-global-instrumentation         TRUE
performance-schema-consumer-statements-digest              TRUE
performance-schema-consumer-thread-instrumentation         TRUE
performance-schema-digests-size                            -1
performance-schema-events-stages-history-long-size         -1
performance-schema-events-stages-history-size              -1
performance-schema-events-statements-history-long-size     -1
performance-schema-events-statements-history-size          -1
performance-schema-events-waits-history-long-size          -1
performance-schema-events-waits-history-size               -1
performance-schema-hosts-size                              -1
performance-schema-instrument
performance-schema-max-cond-classes                        80
performance-schema-max-cond-instances                      -1
performance-schema-max-file-classes                        50
performance-schema-max-file-handles                        32768
performance-schema-max-file-instances                      -1
performance-schema-max-mutex-classes                       200
performance-schema-max-mutex-instances                     -1
performance-schema-max-rwlock-classes                      30
performance-schema-max-rwlock-instances                    -1
performance-schema-max-socket-classes                      10
performance-schema-max-socket-instances                    -1
performance-schema-max-stage-classes                       150
performance-schema-max-statement-classes                   167
performance-schema-max-table-handles                       -1
performance-schema-max-table-instances                     -1
performance-schema-max-thread-classes                      50
performance-schema-max-thread-instances                    -1
performance-schema-session-connect-attrs-size              -1
performance-schema-setup-actors-size                       100
performance-schema-setup-objects-size                      100
performance-schema-users-size                              -1
pid-file                                                   /home/jon/bin/mysql-5.6/data/havskatt.pid
plugin-dir                                                 /home/jon/bin/mysql-5.6/lib/plugin/
port                                                       3306
port-open-timeout                                          0
preload-buffer-size                                        32768
profiling-history-size                                     15
query-alloc-block-size                                     8192
query-cache-limit                                          1048576
query-cache-min-res-unit                                   4096
query-cache-size                                           1048576
query-cache-type                                           OFF
query-cache-wlock-invalidate                               FALSE
query-prealloc-size                                        8192
range-alloc-block-size                                     4096
read-buffer-size                                           131072
read-only                                                  FALSE
read-rnd-buffer-size                                       262144
relay-log                                                  (No default value)
relay-log-index                                            (No default value)
relay-log-info-file                                        relay-log.info
relay-log-info-repository                                  FILE
relay-log-purge                                            TRUE
relay-log-recovery                                         FALSE
relay-log-space-limit                                      0
replicate-same-server-id                                   FALSE
report-host                                                (No default value)
report-password                                            (No default value)
report-port                                                0
report-user                                                (No default value)
safe-user-create                                           FALSE
secure-auth                                                TRUE
secure-file-priv                                           (No default value)
server-id                                                  0
server-id-bits                                             32
sha256-password-private-key-path                           private_key.pem
sha256-password-public-key-path                            public_key.pem
show-slave-auth-info                                       FALSE
skip-grant-tables                                          FALSE
skip-name-resolve                                          FALSE
skip-networking                                            FALSE
skip-show-database                                         FALSE
skip-slave-start                                           FALSE
```

```
slave-allow-batching                    FALSE
slave-checkpoint-group                  512
slave-checkpoint-period                 300
slave-compressed-protocol                 FALSE
slave-exec-mode                        STRICT
slave-load-tmpdir                      /tmp
slave-max-allowed-packet                    1073741824
slave-net-timeout                      3600
slave-parallel-workers                  0
slave-pending-jobs-size-max                 16777216
slave-rows-search-algorithms                TABLE_SCAN,INDEX_SCAN
slave-skip-errors                     (No default value)
slave-sql-verify-checksum                TRUE
slave-transaction-retries               10
slave-type-conversions
slow-launch-time                        2
slow-query-log                         FALSE
slow-query-log-file                     /home/jon/bin/mysql-5.6/data/havskatt-slow.log
socket                                /tmp/mysql.sock
sort-buffer-size                       262144
sporadic-binlog-dump-fail                 FALSE
sql-mode                             NO_ENGINE_SUBSTITUTION
ssl                                 FALSE
ssl-ca                               (No default value)
ssl-capath                             (No default value)
ssl-cert                             (No default value)
ssl-cipher                             (No default value)
ssl-crl                             (No default value)
ssl-crlpath                            (No default value)
ssl-key                              (No default value)
stored-program-cache                        256
super-large-pages                      FALSE
symbolic-links                        TRUE
sync-binlog                           0
sync-frm                              TRUE
sync-master-info                       10000
sync-relay-log                        10000
sync-relay-log-info                    10000
sysdate-is-now                         FALSE
table-definition-cache                  615
table-open-cache                        431
table-open-cache-instances                 1
tc-heuristic-recover                    COMMIT
temp-pool                             TRUE
thread-cache-size                       9
thread-concurrency                      10
thread-handling                       one-thread-per-connection
thread-stack                          262144
time-format                           %H:%i:%s
timed-mutexes                          FALSE
tmp-table-size                        16777216
tmpdir                              /tmp
transaction-alloc-block-size                8192
transaction-isolation                 REPEATABLE-READ
transaction-prealloc-size               4096
transaction-read-only                  FALSE
updatable-views-with-limit               YES
verbose                               TRUE
wait-timeout
```

現在実行中の mysqld サーバーの場合、それに接続し、次のステートメントを発行することで、そのシステム変数の現在の値を確認できます。

```
mysql> SHOW VARIABLES;
```

また、次のステートメントを発行して、実行中のサーバーの統計やステータスインジケータの一部を表示することもできます。

```
mysql> SHOW STATUS;
```

システム変数とステータス情報は、mysqladmin を使用して取得することもできます。

```
shell> mysqladmin variables
shell> mysqladmin extended-status
```

すべてのシステムおよびステータス変数の完全な説明については、セクション5.1.4「サーバーシステム変数」およびセクション5.1.6「サーバーステータス変数」を参照してください。

MySQL はきわめてスケーラブルなアルゴリズムを使用しているため、通常ごくわずかなメモリーで実行できます。ただし、通常 MySQL に多くのメモリーを割り当てることによって、パフォーマンスが向上します。

MySQL サーバーをチューニングする場合、構成するもっとも重要な 2 つの変数は key_buffer_size と table_open_cache です。ほかの変数の変更を試みる前に、まずこれらの変数が適切に設定されていることを確信しておくべきです。

次の例に、さまざまな実行時構成の一般的な変数値を示します。

- 少なくとも 256M バイトのメモリーと多くのテーブルがあり、中程度のクライアント数で最大のパフォーマンスを必要とする場合、次のようなものを使用します。

```
shell> mysqld_safe --key_buffer_size=64M --table_open_cache=256 \
        --sort_buffer_size=4M --read_buffer_size=1M &
```

- メモリーが 128M バイトで、少数のテーブルしかないが、大量のソートを実行する場合、次のようなものを使用できます。

```
shell> mysqld_safe --key_buffer_size=16M --sort_buffer_size=1M
```

  著しく多くの同時接続がある場合、mysqld が接続ごとにごく少量のメモリーを使用するように構成されていないかぎり、スワップの問題が発生する可能性があります。すべての接続に十分なメモリーがある場合に、mysqld は効率的に実行します。

- メモリーがほとんどなく大量の接続がある場合は、次のようなものを使用します。

```
shell> mysqld_safe --key_buffer_size=512K --sort_buffer_size=100K \
        --read_buffer_size=100K &
```

  これでもかまいません。

```
shell> mysqld_safe --key_buffer_size=512K --sort_buffer_size=16K \
        --table_open_cache=32 --read_buffer_size=8K \
        --net_buffer_length=1K &
```

使用可能なメモリーよりはるかに大きいテーブルに対して、GROUP BY または ORDER BY 操作を実行する場合、read_rnd_buffer_size の値を増やして、行の読み取りとそれに続くソート操作を高速化します。

MySQL 配布に付属するサンプルオプションファイルを使用できます。セクション5.1.2「サーバー構成のデフォルト値」を参照してください。

コマンド行で mysqld または mysqld_safe のオプションを指定する場合、そのサーバーの呼び出しに対してのみ有効です。サーバーの実行のたびにオプションを使用するには、それをオプションファイルに入れます。

パラメータの変更の効果を確認するには、次のようなものを実行します。

```
shell> mysqld --key_buffer_size=32M --verbose --help
```

変数値は出力の最後近くに一覧表示されます。--verbose および --help オプションが最後になるようにしてください。そうでないと、コマンド行でそれらのあとに挙げられているすべてのオプションの効果が出力に反映されません。

InnoDB ストレージエンジンのチューニングについては、セクション8.5「InnoDB テーブルの最適化」を参照してください。

8.11.3 ディスク I/O の最適化

- ディスクシークはパフォーマンスの大きなボトルネックです。この問題は、データの量が、効果的なキャッシュが実行不能になるほど大きくなり始めると、明確になります。多かれ少なかれランダムにデータにアクセスする大きなデータベースの場合、読み取りには最低 1 回、書き込みには 2 回のディスクシークが確実に必要になります。この問題を最小にするには、少ないシーク回数でディスクを使用します。

- さまざまなディスクにファイルをシンボリックリンクするか、ディスクストライピングを行なって、使用可能なディスクスピンドル数を増やします (およびそれによってシークのオーバーヘッドを軽減します)。

  - シンボリックリンクの使用

    これは、MyISAM テーブルの場合、データディレクトリ内の通常の場所から別のディスクへのインデックスファイルやデータファイルのシンボリックリンクを作成する (ストライピングされることもある) ことを意味

します。ディスクがほかの目的にも使用されていないものとして、これによって、シークと読み取り時間がともに改善されます。セクション8.11.3.1「シンボリックリンクの使用」を参照してください。

- ストライピング

  ストライピングは、多数のディスクがあり、最初のブロックを最初のディスクに、2 番目のブロックを 2 番目のディスクに、N 番目のブロックを (N MOD number_of_disks) 番目のディスクにというように配置することを意味します。つまり、通常のデータサイズがストライプサイズより小さい (または完全に一致している) 場合に、パフォーマンスが大幅に向上します。ストライピングはオペレーティングシステムとストライプサイズに大きく依存するため、さまざまなストライプサイズでアプリケーションのベンチマークを行なってください。セクション8.12.3「独自のベンチマークの使用」を参照してください。

  ストライピングの速度の違いは、パラメータに大きく依存します。ストライピングパラメータの設定方法とディスク数によって、桁違いの差が測定されることがあります。ランダムアクセスに対する最適化か順次アクセスに対する最適化かを選択する必要があります。

- 信頼性のため、RAID 0+1 (ストライピングとミラーリング) を使用したいと考える場合がありますが、この場合、N 個のドライブのデータを保持するために 2 × N 個のドライブが必要です。これは、そのための資金がある場合に最適なオプションである可能性があります。ただし、それを効率的に処理するために、何らかのボリューム管理ソフトウェアに投資する必要がある場合もあります。

- 適切なオプションは、ある種類のデータがどのくらい重要であるかに応じて RAID レベルを変えることです。たとえば、再生成が可能なやや重要なデータは RAID 0 ディスクに格納しますが、ホスト情報や口グなどの本当に重要なデータは、RAID 0+1 または RAID N ディスクに格納します。RAID N は、パリティビットの更新に必要な時間のため、多くの書き込みがある場合に問題になる可能性があります。

- Linux では、hdparm を使用して、ディスクのインタフェースを構成することによって、パフォーマンスを大幅に向上できます。(負荷時に最大 100% も珍しくありません。)次の hdparm オプションは、MySQL、およびおそらくその他の多くのアプリケーションに非常に適しているはずです。

  ```
  hdparm -m 16 -d 1
  ```

  このコマンドを使用したときのパフォーマンスと信頼性はハードウェアに依存するため、hdparm の使用後はシステムを徹底的にテストすることを強くお勧めします。詳細については、hdparm のマニュアルページを参照してください。hdparm を適切に使用しないと、ファイルシステムの破損が発生することがあるため、実験する前に、すべてをバックアップしてください。

- データベースが使用するファイルシステムのパラメータを設定することもできます。

  ファイルに最後にアクセスされたタイミングを知る必要がない (実際にデータベースサーバーで役立たない) 場合、-o noatime オプションを使用してファイルシステムをマウントできます。それは、ファイルシステム上の i ノードの最終アクセス時間への更新をスキップするため、一部のディスクシークが避けられます。

  多くのオペレーティングシステムで、-o async オプションを使用してファイルシステムをマウントすることによって、ファイルシステムが非同期に更新されるように設定できます。コンピュータが適度に安定している場合、これにより、それほど信頼性を犠牲にすることなく、パフォーマンスが向上するはずです。(Linux ではこのフラグがデフォルトでオンにされています。)

8.11.3.1 シンボリックリンクの使用

データベースやテーブルをデータベースディレクトリからほかの場所に移動して、それらを新しい場所へのシンボリックリンクに置き換えることができます。これを実行したいと考える可能性があるのは、たとえば、データベースを空き領域の多いファイルシステムに移動する場合や、テーブルを別のディスクに分散させてシステムの速度を高める場合です。

InnoDB テーブルの場合、セクション14.5.4「テーブルスペースの位置の指定」に説明するように、シンボリックリンクの代わりに、CREATE TABLE ステートメントで DATA DIRECTORY 句を使用します。この新機能は、サポートされるクロスプラットフォーム技法です。

これを実行する推奨される方法は、データベースディレクトリ全体の別のディスクへのシンボリックリンクを作成することです。MyISAM テーブルのシンボルリンク作成は最後の手段として行います。

データディレクトリの場所を特定するには、次のステートメントを使用します。

```
SHOW VARIABLES LIKE 'datadir';
```

Unix 上のデータベースへのシンボリックリンクの使用

Unix で、データベースのシンボリックリンクを作成する方法は、まず空き領域のあるディスクにディレクトリを作成してから、MySQL データディレクトリからそれへのソフトリンクを作成することです。

```
shell> mkdir /dr1/databases/test
shell> ln -s /dr1/databases/test /path/to/datadir
```

MySQL は、1 つのディレクトリから複数のデータベースへのリンクをサポートしていません。データベースディレクトリとシンボリックリンクの置換は、データベース間のシンボリックリンクを作成しないかぎり、機能します。MySQL データディレクトリにデータベース db1 があり、db1 を指すシンボリックリンク db2 を作成するとします。

```
shell> cd /path/to/datadir
shell> ln -s db1 db2
```

その結果、db1 のすべてのテーブル tbl_a は、db2 にもテーブル tbl_a として表示されます。あるクライアントが db1.tbl_a を更新し、ほかのクライアントが db2.tbl_a を更新すると、問題が発生する可能性があります。

Unix 上の MyISAM へのシンボリックリンクの使用

シンボリックリンクは、MyISAM テーブルに対してのみ完全にサポートされています。ほかのストレージエンジンのテーブルで使用されているファイルの場合、シンボリックリンクを使用しようとすると、未知の問題が発生することがあります。InnoDB テーブルの場合は、代わりにセクション14.5.4「テーブルスペースの位置の指定」に説明する代替の技法を使用します。

完全に動作する realpath() 呼び出しがないシステムでは、テーブルのシンボリックリンクを作成しないでください。(Linux と Solaris では realpath() をサポートしています)。システムでシンボリックリンクをサポートしているかどうかを判断するには、次のステートメントを使用して、have_symlink システム変数の値をチェックします。

```
SHOW VARIABLES LIKE 'have_symlink';
```

MyISAM テーブルのシンボリックリンクの処理は次のように機能します。

- データディレクトリには、常にテーブルフォーマット (.frm) ファイル、データ (.MYD) ファイル、およびインデックス (.MYI) ファイルがあります。データファイルとインデックスファイルは、ほかの場所に移動し、データディレクトリ内でシンボリックリンクによって置き換えることができます。フォーマットファイルはできません。
- データファイルとインデックスファイルは、独立して別々のディレクトリへのシンボリックリンクを作成できます。
- 実行中の MySQL サーバーにシンボリックリンクの作成を実行するように指示するには、CREATE TABLE に DATA DIRECTORY および INDEX DIRECTORY オプションを使用します。セクション13.1.17「CREATE TABLE 構文」を参照してください。または、mysqld が実行中でない場合は、コマンド行から ln -s を使用して、シンボリックリンクの作成を手動で実行できます。

  > **注記**
  >
  > DATA DIRECTORY および INDEX DIRECTORY オプションのいずれか、または両方で使用されるパスには、MySQL data ディレクトリを含めることができません。(Bug #32167)

- myisamchk が、シンボリックリンクをデータファイルやインデックスファイルに置き換えません。それは、シンボリックリンクが指しているファイルに対して直接作用します。一時ファイルはすべてデータファイルやインデックスファイルが配置されているディレクトリに作成されます。同じことが ALTER TABLE、OPTIMIZE TABLE、および REPAIR TABLE ステートメントにも当てはまります。
-

  > **注記**
  >
  > シンボリックリンクを使用しているテーブルを削除すると、シンボリックリンクとシンボリックリンクが指しているファイルの両方が削除されます。これは、システム root として mysqld を実行したり、システムユーザーに MySQL データベースディレクトリへの書き込みアクセス権を許可したりしないきわめて正当な理由です。

- ALTER TABLE ... RENAME または RENAME TABLE を使用してテーブルの名前を変更し、テーブルを別のデータベースに移動しない場合、データベースディレクトリのシンボリックリンクの名前が新しい名前に変更され、データファイルとインデックスファイルもそれに従って名前が変更されます。
- ALTER TABLE ... RENAME または RENAME TABLE を使用してテーブルを別のデータベースに移動すると、テーブルが別のデータベースディレクトリに移動されます。テーブル名が変更された場合、新しいデータベー

スディレクトリ内のシンボリックリンクの名前が新しい名前に変更され、データファイルとインデックスファイルもそれに従って名前が変更されます。

- シンボリックリンクを使用していない場合、--skip-symbolic-links オプションを付けて mysqld を起動し、だれも mysqld を使用して、データディレクトリ外のファイルを削除したり名前を変更したりできないようにします。

これらのテーブルシンボリックリンクの操作はサポートされていません。

- ALTER TABLE は DATA DIRECTORY および INDEX DIRECTORY テーブルオプションを無視します。
- 前に示したように、データファイルとインデックスファイルにのみシンボリックリンクにできます。.frm ファイルはシンボリックリンクにできません。これを実行しようとすると (たとえば、1 つのテーブル名を別のテーブルのシノニムにするなど) 正しくない結果が生成されます。MySQL データディレクトリにデータベース db1、このデータベースにテーブル tbl1 があり、db1 ディレクトリに tbl1 を指すシンボリックリンク tbl2 を作成するとします。

```
shell> cd /path/to/datadir/db1
shell> ln -s tbl1.frm tbl2.frm
shell> ln -s tbl1.MYD tbl2.MYD
shell> ln -s tbl1.MYI tbl2.MYI
```

あるスレッドが db1.tbl1 を読み取り、別のスレッドで db1.tbl2 を更新すると、問題が発生します。

- クエリーキャッシュが「だまされます」 (tbl1 が更新されていないことを知る方法がないため、古くなっている結果を返します)。
- tbl2 に対する ALTER ステートメントが失敗します。

Windows 上のデータベースへのシンボリックリンクの使用

Windows では、データベースディレクトリにシンボリックリンクを使用できます。これにより、データベースディレクトリへのシンボリックリンクを設定して、それを別の場所 (別のディスク上など) に置くことができます。Windows でのデータベースシンボリックリンクの使用は、Unix でのそれらの使用に似ていますが、リンクのセットアップの手順は異なります。

mydb というデータベースのデータベースディレクトリを D:\data\mydb に配置したいとします。これを実行するには、MySQL データディレクトリ内に D:\data\mydb を指すシンボリックリンクを作成します。ただし、シンボリックリンクを作成する前に、必要に応じて D:\data\mydb ディレクトリを作成して、それが存在することを確認します。データディレクトリ内に mydb というデータベースディレクトリがすでにある場合は、それを D:\data に移動します。そうしないと、シンボリックリンクは無効になります。問題を避けるために、データベースディレクトリの移動時にサーバーが実行していないことを確認してください。

データベースシンボリックリンクを作成するための手順は Windows のバージョンによって異なります。

Windows Vista、Windows Server 2008 以降には、ネイティブのシンボリックリンクのサポートがあるため、mklink コマンドを使用して、シンボリックリンクを作成できます。このコマンドには管理者権限が必要です。

1. 場所をデータディレクトリ内に変更します。

   ```
   C:\> cd \path\to\datadir
   ```

2. データディレクトリで、データベースディレクトリの場所を指す mydb というシンボリックリンクを作成します。

   ```
   C:\> mklink /d mydb D:\data\mydb
   ```

このあと、データベース mydb に作成されるすべてのテーブルが D:\data\mydb に作成されます。

または、MySQL でサポートされる任意のバージョンの Windows で、データディレクトリに宛先ディレクトリのパスを格納する .sym ファイルを作成して、MySQL データベースへのシンボリックリンクを作成できます。ファイルは、db_name.sym という名前にしてください。ここで db_name はデータベース名です。

Windows で、.sym ファイルを使用したデータベースシンボリックリンクのサポートは、デフォルトで有効にされています。.sym ファイルシンボリックリンクが必要でない場合は、--skip-symbolic-links オプションで mysqld を起動して、それらのサポートを無効にできます。システムで .sym ファイルシンボリックリンクをサポートしているかどうかを判断するには、次のステートメントを使用して、have_symlink システム変数の値をチェックします。

```
SHOW VARIABLES LIKE 'have_symlink';
```

.sym ファイルシンボリックリンクを作成するには、次の手順を使用します。

1. 場所をデータディレクトリ内に変更します。

   ```
   C:\> cd \path\to\datadir
   ```

2. データディレクトリ内に、パス名 D:\data\mydb\ を含む mydb.sym というテキストファイルを作成します。

   > **注記**
   >
   > 新しいデータベースとテーブルのパス名は絶対パスにしてください。相対パスを指定する場合、場所は mydb.sym ファイルに相対的になります。

このあと、データベース mydb に作成されるすべてのテーブルが D:\data\mydb に作成されます。

> **注記**
>
> .sym ファイルのサポートは、mklink を使用して使用可能なネイティブシンボリックリンクのサポートと重複しているため、.sym ファイルの使用は、MySQL 5.6.9 現在非推奨にされ、それらのサポートは将来の MySQL リリースで削除されます。

Windows でのデータベースシンボリックリンクへの .sym ファイルの使用には、次の制限が適用されます。これらの制限は mklink を使用して作成されるシンボリックリンクには適用されません。

- MySQL データディレクトリにデータベースと同じ名前のディレクトリが存在する場合、シンボリックリンクは使用されません。
- --innodb_file_per_table オプションは使用できません。
- mysqld をサービスとして実行する場合、リモートサーバーにマップされたドライブをシンボリックリンクのリンク先として使用することはできません。回避方法として、フルパス (\\servername\path\) を使用できます。

8.11.4 メモリーの使用の最適化

8.11.4.1 MySQL のメモリーの使用方法

次のリストに、mysqld サーバーがメモリーを使用する方法のいくつかを示します。該当する場合、メモリー使用に関連するサーバー変数の名前も示しています。

- すべてのスレッドは MyISAM キーバッファーを共有し、そのサイズは key_buffer_size 変数によって決定されます。サーバーによって使用されるほかのバッファーは、必要に応じて割り当てられます。セクション 8.11.2「サーバーパラメータのチューニング」を参照してください。
- クライアント接続の管理に使用される各スレッドは、いくらかのスレッド固有の領域を使用します。次のリストに、これらとそれらのサイズを制御する変数を示します。
  - スタック (変数 thread_stack)
  - 接続バッファー (変数 net_buffer_length)
  - 結果バッファー (変数 net_buffer_length)

  接続バッファーと結果バッファーはそれぞれ net_buffer_length バイトに等しいサイズから開始されますが、必要に応じて max_allowed_packet バイトまで動的に拡大されます。結果バッファーは各 SQL ステートメントのあとに net_buffer_length バイトに縮小されます。ステートメントの実行中は現在のステートメント文字列のコピーも割り当てられます。
- すべてのスレッドで同じベースメモリーを共有します。
- スレッドが必要ない場合、それに割り当てられたメモリーが解放され、スレッドがスレッドキャッシュに戻らないかぎり、システムに返されます。その場合、メモリーは割り当てられた状態のままになります。
- myisam_use_mmap システム変数を 1 に設定して、すべての MyISAM テーブルのメモリーマッピングを有効にできます。
- テーブルの順次スキャンを実行する各リクエストは、read buffer (変数 read_buffer_size) を割り当てます。

- 行を任意の順序で読み取る場合 (たとえば、ソートに続いて)、random-read buffer (変数 read_rnd_buffer_size) を割り当てて、ディスクシークを避けることができます。
- すべての結合は単一のパスで実行され、ほとんどの結合は一時テーブルも使用せずに実行できます。ほとんどの一時テーブルはメモリーベースのハッシュテーブルです。大きな行長 (すべてのカラム長の合計として算出される) を持つか BLOB カラムを含む一時テーブルはディスク上に格納されます。

  内部インメモリー一時テーブルが大きくなりすぎると、MySQL は、テーブルをインメモリーから、MyISAM ストレージエンジンによって処理されるディスク上フォーマットに変更して、これを自動的に処理します。セクション8.4.4「MySQL が内部一時テーブルを使用する仕組み」に説明するように、許可される一時テーブルのサイズを増やすことができます。
- ソートを実行するほとんどのリクエストは、ソートバッファーおよび結果セットサイズに応じた 0 から 2 つの一時ファイルを割り当てます。セクションB.5.4.4「MySQL が一時ファイルを格納する場所」を参照してください。
- ほとんどすべての解析と計算は、スレッドローカルの再利用可能なメモリープールで実行されます。小さい項目にはメモリーオーバーヘッドが不要であるため、通常の低速メモリーの割り当てと解放が回避されます。メモリーは、予測外に大きな文字列にのみ割り当てられます。
- 開かれる MyISAM テーブルごとにインデックスファイルが 1 回開かれ、データファイルは同時実行中のスレッドごとに 1 回開かれます。同時スレッドごとに、テーブル構造、各カラムのカラム構造、およびサイズ 3 * N のバッファーが割り当てられます (ここで N は最大行長で、BLOB カラムをカウントしていません)。BLOB カラムには、5 から 8 バイト+ BLOB データの長さが必要です。MyISAM ストレージエンジンは、内部使用のため 1 つ余分な行バッファーを保持します。
- BLOB カラムがあるテーブルごとに、大きな BLOB 値を読み取るためにバッファーが動的に拡大されます。テーブルをスキャンする場合は、最大の BLOB 値と同じ大きさのバッファーが割り当てられます。
- 使用中のすべてのテーブルのハンドラ構造がキャッシュに保存され、FIFO として管理されます。初期キャッシュサイズは、table_open_cache システム変数の値から取得されます。テーブルが同時に 2 つの実行中のスレッドによって使用されている場合、キャッシュにはそのテーブルの 2 つのエントリが含まれます。セクション8.4.3.1「MySQL でのテーブルのオープンとクローズの方法」を参照してください。
- FLUSH TABLES ステートメントまたは mysqladmin flush-tables コマンドは、使用中でないすべてのテーブルを一度に閉じ、現在実行中のスレッドの終了時に閉じられるように使用中のすべてのテーブルをマークします。これにより、事実上ほとんどの使用中のメモリーが解放されます。FLUSH TABLES はすべてのテーブルが閉じられるまで戻りません。
- GRANT、CREATE USER、CREATE SERVER、および INSTALL PLUGIN ステートメントの結果として、サーバーは情報をメモリーにキャッシュします。このメモリーは、対応する REVOKE、DROP USER、DROP SERVER、および UNINSTALL PLUGIN ステートメントによって解放されないため、キャッシュを発生させるステートメントの多数のインスタンスを実行するサーバーでは、メモリー使用量が増加します。このキャッシュされたメモリーは FLUSH PRIVILEGES で解放できます。

ps およびその他のステータスプログラムが、mysqld が大量のメモリーを使用していることをレポートすることがあります。これは、さまざまなメモリーアドレス上のスレッドスタックによって発生する可能性があります。たとえば、Solaris バージョンの ps はスタック間の未使用のメモリーが使用されているメモリーとしてカウントされます。これを確認するには、swap -s で使用可能なスワップをチェックします。いくつかのメモリーリーク検出ツール (市販とオープンソースの両方の) で mysqld をテストしているため、メモリーリークはないはずです。

8.11.4.2 ラージページのサポートの有効化

ハードウェアまたはオペレーティングシステムのアーキテクチャーによっては、デフォルト (通常は 4K バイト) よりも大きいメモリーページをサポートしています。このサポートの実際の実装は、ベースとなるハードウェアやオペレーティングシステムに依存します。大量のメモリーアクセスがあるアプリケーションの場合、大きいページを使用して、トランスレーションルックアサイドバッファー (TLB; Translation Lookaside Buffer) のミスが減ることによってパフォーマンスが改善される可能性があります。

MySQL では、InnoDB でラージページを使用して、バッファープールと追加のメモリープールにメモリーを割り当てることができます。

MySQL での標準的な大規模ページの使用では、サポートされる最大サイズである 4M バイトまでの使用が試行されます。Solaris では「超大規模ページ」機能により 256M バイトまでのページの使用が可能です。この機能は最新の SPARC プラットフォームで使用できます。これは --super-large-pages または --skip-super-large-pages オプションを使用して有効または無効にできます。

MySQL は、ラージページのサポートの Linux 実装 (Linux では HugeTLB と呼ばれる) もサポートします。

Linux でラージページを使用する前に、カーネルで、それらをサポートできるようにする必要があり、HugeTLB メモリープールを構成する必要があります。参考のため、HugeTBL API は、Linux ソースの Documentation/vm/hugetlbpage.txt ファイルで説明されています。

Red Hat Enterprise Linux などの一部の最近のシステムのカーネルでは、ラージページ機能がデフォルトで有効にされているようです。使用しているカーネルにこれが当てはまるかどうかを確認するには、次のコマンドを使用し、「huge」 を含む出力行を探します。

```
shell> cat /proc/meminfo | grep -i huge
HugePages_Total:       0
HugePages_Free:        0
HugePages_Rsvd:        0
HugePages_Surp:        0
Hugepagesize:       4096 kB
```

空でないコマンド出力は、ラージページのサポートが存在することを示しますが、ゼロの値は、使用するように構成されたページがないことを示します。

ラージページをサポートするようにカーネルを再構成する必要がある場合、手順については hugetlbpage.txt ファイルを参照してください。

Linux カーネルでラージページのサポートが有効にされていると仮定し、それを次のコマンドを使用して、MySQL で使用するように構成します。通常、システムが起動するたびにコマンドが実行されるように、システムのブートシーケンス中に実行される rc ファイルまたは同等の起動ファイルにこれらを入れます。コマンドは、ブートシーケンスの早期の、MySQL サーバーが起動する前に実行されるべきです。システムに適切なように、割り当ての数値とグループ番号を変更してください。

```
# Set the number of pages to be used.
# Each page is normally 2MB, so a value of 20 = 40MB.
# This command actually allocates memory, so this much
# memory must be available.
echo 20 > /proc/sys/vm/nr_hugepages

# Set the group number that is permitted to access this
# memory (102 in this case). The mysql user must be a
# member of this group.
echo 102 > /proc/sys/vm/hugetlb_shm_group

# Increase the amount of shmem permitted per segment
# (12G in this case).
echo 1560281088 > /proc/sys/kernel/shmmax

# Increase total amount of shared memory.  The value
# is the number of pages. At 4KB/page, 4194304 = 16GB.
echo 4194304 > /proc/sys/kernel/shmall
```

MySQL で使用する場合、通常 shmmax の値を shmall の値に近くなるようにしたいと考えます。

ラージページの構成を確認するには、前述のとおりに再度 /proc/meminfo をチェックします。これで、0 以外の値が表示されるはずです。

```
shell> cat /proc/meminfo | grep -i huge
HugePages_Total:      20
HugePages_Free:       20
HugePages_Rsvd:        0
HugePages_Surp:        0
Hugepagesize:       4096 kB
```

hugetlb_shm_group を使用するための最後の手順は、mysql ユーザーに、memlock 制限として 「unlimited」 値を指定することです。これを実行するには、/etc/security/limits.conf を編集するか、mysqld_safe スクリプトに次のコマンドを追加します。

```
ulimit -l unlimited
```

ulimit コマンドを mysqld_safe に追加すると、mysql ユーザーに切り替える前に root ユーザーの memlock 制限が unlimited に設定されます。(これは、mysqld_safe が root によって起動されたものと仮定します。)

MySQL のラージページのサポートはデフォルトで無効にされています。それを有効にするには、サーバーを --large-pages オプションで起動します。たとえば、サーバーの my.cnf ファイルで次の行を使用できます。

```
[mysqld]
```

```
large-pages
```

このオプションを使用すると、InnoDB はそのバッファープールと追加のメモリープールに自動的にラージページを使用します。InnoDB がこれを実行できない場合、従来のメモリーの使用に戻り、エラーログに警告を書き込みます: Warning: Using conventional memory pool

ラージページが使用されていることを確認するには、再度 /proc/meminfo をチェックします。

```
shell> cat /proc/meminfo | grep -i huge
HugePages_Total:      20
HugePages_Free:       20
HugePages_Rsvd:        2
HugePages_Surp:        0
Hugepagesize:       4096 kB
```

8.11.5 ネットワークの使用の最適化

8.11.5.1 MySQL のクライアント接続のためのスレッドの使用方法

接続マネージャースレッドは、サーバーが待機しているネットワークインタフェース上でクライアントの接続要求を処理します。どのプラットフォームでも、1 つのマネージャースレッドが TCP/IP 接続要求を処理します。Unix では、このマネージャースレッドは Unix ソケットファイルの接続要求も処理します。Windows では、1 つのマネージャースレッドが共有メモリーの接続要求を処理し、別のマネージャースレッドが名前付きパイプの接続要求を処理します。サーバーは、待機していないインタフェースを処理するためのスレッドを作成しません。たとえば、Windows サーバーで名前付きパイプ接続のサポートが有効になっていない場合、これらの接続を処理するスレッドは作成されません。

接続マネージャースレッドは、各クライアント接続を、その接続の認証および要求を処理する専用スレッドに関連付けます。マネージャースレッドは、必要に応じて新しいスレッドを作成しますが、まずスレッドキャッシュを調べて接続に使用できるスレッドが含まれているかどうかを確認することによって、それを回避することを試みます。接続が終了すると、スレッドキャッシュが満杯でない場合は、そのスレッドがスレッドキャッシュに返されます。

この接続スレッドモデルでは、現在接続しているクライアントと同数のスレッドが存在し、多数の接続を処理するためにサーバーのワークロードを拡大する必要がある場合にはいくつか欠点があります。たとえば、スレッドの作成と破棄の負荷が大きくなります。また、各スレッドにスタック領域などのサーバーリソースとカーネルリソースが必要になります。多数の同時接続に対応するには、スレッドあたりのスタックサイズは小さく保つ必要があり、それが小さくなりすぎるか、またはサーバーで大量のメモリーを消費することになる状況につながります。ほかのリソースを使い果たす可能性もあり、スケジューリングのオーバーヘッドがかなり大きくなる可能性があります。

MySQL 5.6.10 現在、MySQL 5.6 の商用配布には、オーバーヘッドを軽減し、パフォーマンスを向上するように設計されている代替のスレッド処理モデルを提供するスレットプールプラグインが付属しています。これは、多数のクライアント接続のステートメント実行スレッドを効率的に管理して、サーバーのパフォーマンスを向上させるスレッドプールを実装します。セクション8.11.6「スレッドプールプラグイン」を参照してください。

クライアント接続を処理するスレッドをサーバーがどのように管理するかを制御し、モニターするには、いくつかのシステム変数とステータス変数が関連します。(セクション5.1.4「サーバーシステム変数」およびセクション5.1.6「サーバーステータス変数」を参照してください。)

スレッドキャッシュは、thread_cache_size システム変数によって決定されるサイズを持ちます。デフォルト値は 0 (キャッシュなし) で、この場合、スレッドは新しい接続ごとにセットアップされ、接続の終了時に破棄されます。thread_cache_size を N に設定し、N 個の非アクティブ接続スレッドをキャッシュできるようにします。thread_cache_size はサーバーの起動時に設定するか、サーバーの実行中に変更できます。関連付けられていたクライアント接続が終了すると、接続スレッドは非アクティブになります。

キャッシュ内のスレッド数、およびスレッドをキャッシュから取得できなかったため作成されたスレッドの数をモニターするには、Threads_cached および Threads_created ステータス変数をモニターします。

サーバーの起動時または実行時に max_connections を設定して、同時に接続できるクライアントの最大数を制御できます。

スレッドスタックが小さすぎると、これによって、サーバーが処理できる SQL ステートメントの複雑さ、ストアドプロシージャーの再帰の深さ、およびその他のメモリーを大量に消費するアクションが制限されます。各スレッドに N バイトのスタックサイズを設定するには、サーバーを --thread_stack=N で起動します。

8.11.5.2 DNS ルックアップの最適化とホストキャッシュ

MySQL サーバーはクライアントに関する情報 (IP アドレス、ホスト名、エラー情報) を格納するホストキャッシュをメモリーに保持します。サーバーはこのキャッシュを非ローカル TCP 接続に使用します。それは、ループバックインタフェースアドレス (127.0.0.1 または ::1) を使用して確立された TCP 接続、または Unix ソケットファイル、名前付きパイプ、または共有メモリーを使用して確立された接続には、キャッシュを使用しません。

新しいクライアント接続ごとに、サーバーはクライアント IP アドレスを使用して、クライアントホスト名がホストキャッシュ内にあるかどうかをチェックします。ない場合は、サーバーはホスト名の解決を試みます。まず、それは IP アドレスをホスト名に解決し、そのホスト名を再度 IP アドレスに解決します。次に、その結果と元の IP アドレスを比較して、それらが同じであることを確認します。サーバーはこの操作の結果に関する情報をホストキャッシュに格納します。キャッシュがいっぱいである場合、直近で使用されていないエントリが破棄されます。

host_cache パフォーマンススキーマテーブルは、SELECT ステートメントを使用して調査できるようにホストキャッシュの内容を公開します。これは、接続の問題の原因の診断に役立つことがあります。セクション 22.9.10.1「host_cache テーブル」を参照してください。

サーバーは次のようにホストキャッシュ内のエントリを処理します。

1. 最初の TCP クライアント接続が指定された IP アドレスからサーバーに到達すると、クライアント IP、ホスト名、およびクライアントルックアップ検証フラグを記録する新しいエントリが作成されます。最初に、ホスト名が NULL に設定され、フラグは false になります。このエントリは同じ発信元 IP からの後続のクライアント接続にも使用されます。

2. クライアント IP エントリの検証フラグが false の場合、サーバーは IP からホスト名への DNS の解決を試みます。それが成功した場合、ホスト名が解決されたホスト名で更新され、検証フラグが true に設定されます。解決が成功しない場合、とられるアクションは、エラーが永続的か一時的かによって異なります。永続的なエラーの場合、ホスト名は NULL のままになり、検証フラグは true に設定されます。一時的なエラーの場合、ホスト名と検証フラグは変更されないままになります。(次回にクライアントがこの IP から接続したときは、別の DNS 解決の試みが行われます。)

3. 特定の IP アドレスからの着信クライアント接続の処理中にエラーが発生した場合、サーバーはその IP のエントリ内の対応するエラーカウンタを更新します。記録されたエラーの説明については、セクション 22.9.10.1「host_cache テーブル」を参照してください。

オペレーティングシステムでスレッドセーフな gethostbyaddr_r() および gethostbyname_r() 呼び出しをサポートしている場合、サーバーはそれらを使用してホスト名解決を実行します。そうでない場合、ルックアップを実行するスレッドは相互排他ロックを実行し、代わりに、gethostbyaddr() および gethostbyname() を呼び出します。この場合、相互排他ロックを保持するスレッドがそれを解放するまで、ほかのスレッドはホストキャッシュ内にないホスト名を解決できません。

サーバーはいくつかの目的でホストキャッシュを使用します。

- IP からホスト名へのルックアップの結果をキャッシュすることによって、サーバーはクライアント接続ごとの DNS ルックアップの実行を回避します。代わりに、特定のホストに対して、そのホストからの最初の接続でのみルックアップを実行する必要があります。

- キャッシュには、接続プロセス中に発生したエラーに関する情報が格納されます。一部のエラーは「ブロッキング」とみなされます。成功した接続がない特定のホストから、これらの多くが連続して発生している場合、サーバーはそのホストからのその後の接続をブロックします。max_connect_errors システム変数は、ブロックが行われるまで許可されるエラーの数を指定します。セクションB.5.2.6「ホスト 'host_name' は拒否されました」を参照してください。

ブロックされたホストのブロックを解除するには、FLUSH HOSTS ステートメントを発行するか、mysqladmin flush-hosts コマンドを実行して、ホストキャッシュをフラッシュします。

ブロックされたホストからの最後の接続の試み以降に、ほかのホストからのアクティビティーが発生した場合、FLUSH HOSTS を使用しなくても、ブロックされたホストのブロックが解除される可能性があります。これは、キャッシュ内にないクライアント IP から接続が到着したときに、キャッシュがいっぱいである場合、サーバーが直近で使用されていないキャッシュエントリを破棄して、新しいエントリのための空きを作るために発生する可能性があります。破棄されたエントリがブロックされたホストのものである場合、そのホストのブロックが解除されます。

ホストキャッシュはデフォルトで有効になっています。それを無効にするには、サーバーの起動時や実行時に、host_cache_size システム変数を 0 に設定します。

DNSホスト名ルックアップを無効にするには、--skip-name-resolve オプションでサーバーを起動します。この場合、サーバーは IP アドレスのみを使用し、ホスト名を使用しないで、接続しているホストを MySQL 付与テーブル内の行と照合します。IP アドレスを使用してそれらのテーブルに指定されたアカウントのみを使用できます。

著しく遅い DNS と多くのホストがある場合、--skip-name-resolve で DNS ルックアップを無効にするか、または host_cache_size の値を増やしてホストキャッシュを大きくすることによって、パフォーマンスを向上できる可能性があります。

TCP/IP 接続を完全に禁止するには、--skip-networking オプションでサーバーを起動します。

一部の接続エラーは TCP 接続に関連付けられないか、接続プロセスのきわめて早期に (IP アドレスも判明する前に) 発生するか、または特定の IP アドレスに固有でありません (メモリー不足の状況など)。これらのエラーについては、Connection_errors_xxx ステータス変数をチェックしてください (セクション5.1.6「サーバーステータス変数」を参照してください)。

8.11.6 スレッドプールプラグイン

> **注記**
>
> MySQL スレッドプールは商用拡張機能です。商用製品 (MySQL Enterprise Edition) の詳細については、https://www.mysql.com/products/ を参照してください。

MySQL 5.6.10 現在、MySQL 5.6 の商用配布には、サーバープラグインを使用して実装される MySQL スレッドプールが付属しています。MySQL サーバーのデフォルトのスレッド処理モデルでは、クライアント接続ごとに 1 つのスレッドを使用してステートメントが実行されます。より多くのクライアントがサーバーに接続してステートメントを実行すると、全体的なパフォーマンスが低下します。スレットプールプラグインは、オーバーヘッドを軽減し、パフォーマンスを向上するように設計されている代替のスレッド処理モデルを提供します。このプラグインは、多数のクライアント接続に対してステートメント実行スレッドを効率的に管理することによってサーバーのパフォーマンスを向上させるスレッドプールを実装します。

スレッドプールは、接続モデルあたり 1 つのスレッドのいくつかの問題に対処します。

- スレッドが多すぎると、高度な並列実行ワークロードで CPU キャッシュがほとんど役に立たなくなります。スレッドプールはスレッドスタックの再利用を促進し、CPU キャッシュのフットプリントを最小にします。
- 並列で実行しているスレッド数が多すぎると、コンテキストスイッチングのオーバーヘッドが高くなります。これは、オペレーティングシステムスケジューラにも困難なタスクを与えます。スレッドプールは、アクティブスレッドの数を制御して、それが処理可能で、MySQL を実行しているサーバーホストに適切なレベルで MySQL サーバー内の並列性を維持します。
- 並列で実行するトランザクションが多すぎると、リソースの競合が増加します。InnoDB では、これにより中央の相互排他ロックの保持に費やされる時間が多くなります。スレッドプールは、あまり多く並列で実行しないように、トランザクションが開始するタイミングを制御します。

スレッドプールプラグインは商用機能です。MySQL コミュニティー配布には含まれていません。

Windows では、スレッドプールプラグインに Windows Vista 以降が必要です。Linux では、プラグインにカーネル 2.6.9 以降が必要です。

追加のリソース

セクションA.14「MySQL 5.6 FAQ: MySQL エンタープライズスケーラビリティースレッドプール」

8.11.6.1 スレッドプールコンポーネントとインストール

スレッドプール機能は次のコンポーネントで構成されます。

- プラグインライブラリオブジェクトファイルには、スレッドプールコード用のプラグインと、いくつかの INFORMATION_SCHEMA テーブル用のプラグインが含まれています。

  スレッドプールの仕組みの詳細については、セクション8.11.6.2「スレッドプール操作」を参照してください。

  INFORMATION_SCHEMA テーブルには、TP_THREAD_STATE、TP_THREAD_GROUP_STATE、および TP_THREAD_GROUP_STATS という名前が付けられています。これらのテーブルは、スレッドプール操作に関する情報を提供します。詳細については、セクション21.31「スレッドプールの INFORMATION_SCHEMA テーブル」を参照してください。

- いくつかのシステム変数がスレッドプールに関連しています。thread_handling システム変数は、サーバーがスレッドプールプラグインを正常にロードしたときに、loaded-dynamically の値になります。

  ほかの関連の変数はスレッドプールプラグインによって実装されます。それが有効にされていない場合、それらは使用できません。

  - thread_pool_algorithm: スケジューリングに使用する並列性アルゴリズム。
  - thread_pool_high_priority_connection: セッションのステートメント実行のスケジュール方法。
  - thread_pool_prio_kickup_timer: スレッドプールが、実行を待機しているステートメントを低優先度キューから高優先度キューに移動するまでの時間。
  - thread_pool_max_unused_threads: 許可するスリープ中のスレッド数。
  - thread_pool_size: スレッドプール内のスレッドグループの数。これはスレッドプールのパフォーマンスを制御するもっとも重要なパラメータです。
  - thread_pool_stall_limit: 実行中のステートメントが停滞しているとみなされるまでの時間。

  起動時に、プラグインによって実装されているいずれかの変数が不正な値に設定された場合、プラグインの初期化が失敗し、プラグインはロードされません。

  スレッドプールパラメータの設定については、セクション8.11.6.3「スレッドプールのチューニング」を参照してください。

- パフォーマンススキーマは、スレッドプールに関する情報を公開し、操作のパフォーマンスの調査に使用できます。詳細については、第22章「MySQL パフォーマンススキーマ」を参照してください。

サーバーが使用できるように、スレッドプールライブラリオブジェクトファイルは MySQL プラグインディレクトリ (plugin_dir システム変数によって指定されたディレクトリ) に存在する必要があります。スレッドプール機能を有効にするには、--plugin-load オプションでサーバーを起動することによって、使用されるプラグインをロードします。たとえば、プラグインオブジェクトファイルだけを指定した場合、サーバーはそれに含まれるすべてのプラグイン (つまり、スレッドプールプラグインとすべての INFORMATION_SCHEMA テーブル) をロードします。これを実行するには、これらの行を my.cnf ファイルに挿入します。

```
[mysqld]
plugin-load=thread_pool.so
```

それは、個別にスレッドプールプラグインを指定して、それらをすべてロードするのと同等です。

```
[mysqld]
plugin-load=thread_pool.so
plugin-load=thread_pool=thread_pool.so;tp_thread_state=thread_pool.so;tp_thread_group_state=thread_pool.so;tp_thread_group_stats=thread_pool.so
```

システム上のオブジェクトファイルのサフィクスが .so とは異なる場合、正しいサフィクスに置き換えてください (たとえば Windows の場合は .dll)。

必要に応じて、サーバーにプラグインディレクトリの場所を伝えるために、plugin_dir システム変数の値を設定します。

必要な場合、ライブラリファイルから個々のプラグインをロードできます。スレッドプールプラグインをロードするが、INFORMATION_SCHEMA テーブルはロードしない場合、次のようなオプションを使用します。

```
[mysqld]
plugin-load=thread_pool=thread_pool.so
```

スレッドプールプラグインと TP_THREAD_STATE INFORMATION_SCHEMA テーブルのみをロードするには、次のようなオプションを使用します。

```
[mysqld]
plugin-load=thread_pool=thread_pool.so;TP_THREAD_STATE=thread_pool.so
```

ただし、すべての INFORMATION_SCHEMA テーブルをロードしない場合、一部またはすべての MySQL Enterprise Monitor スレッドプールグラフが空になります。

プラグインのインストールを検証するには、INFORMATION_SCHEMA.PLUGINS テーブルを調査するか、SHOW PLUGINS ステートメントを使用します。セクション5.1.8.2「サーバープラグイン情報の取得」を参照してください。

サーバーはスレッドプラグインを正常にロードしたら、thread_handling システム変数を dynamically-loaded に設定します。プラグインのロードに失敗した場合、サーバーはエラーログにメッセージを書き込みます。

8.11.6.2 スレッドプール操作

スレッドプールは、それぞれクライアント接続のセットを管理するいくつかのスレッドグループから構成されます。接続が確立されると、スレッドプールはラウンドロビン方式でそれらをスレッドグループに割り当てます。

スレッドグループの数は、thread_pool_size システム変数を使用して構成できます。グループのデフォルトの数は 16 です。この変数の設定のガイドラインについては、セクション8.11.6.3「スレッドプールのチューニング」を参照してください。

グループあたりのスレッドの最大数は 4096 (または 1 つのスレッドが内部で使用される一部のシステムでは 4095) です。

スレッドプールは接続とスレッドを区別するため、接続と、それらの接続から受信したステートメントを実行するスレッド間に固定の関係はありません。これは、1 つのスレッドを 1 つの接続に関連付けて、そのスレッドがその接続からのすべてのステートメントを実行するようにするデフォルトのスレッド処理モデルとは異なります。

スレッドプールは、いつでも各グループで最大 1 つのスレッドが実行するように努めますが、ときによって、最高のパフォーマンスのため、一時的に多くのスレッドの実行を許可することがあります。このアルゴリズムは次のように機能します。

- 各スレッドグループには、グループに割り当てられた接続からの着信ステートメントを待機するリスナースレッドがあります。ステートメントが到着すると、スレッドグループはその実行をただちに開始するか、あとで実行するためにキューに入れます。
  - 即時の実行は、ステートメントが受信した唯一のもので、キューに入れられていたり、現在実行していたりするステートメントがない場合に行われます。
  - キューイングは、ステートメントの実行をすぐに開始できない場合に行われます。
- 即時の実行が行われる場合、実行はリスナースレッドによって行われます。(つまり、グループ内に一時的に待機しているスレッドがなくなります。)ステートメントがすぐに終了すると、実行中のスレッドがステートメントの待機に戻ります。そうでない場合、スレッドプールはステートメントを停滞中とみなし、別のスレッド (必要に応じて作成して) をリスナースレッドとして開始します。スレッドグループが停滞中のステートメントによってブロックされないように、スレッドプールには、スレッドグループ状態を定期的にモニターするバックグラウンドスレッドがあります。

  待機中のスレッドを使用して、ただちに開始できるステートメントを実行することによって、ステートメントがすぐに終了した場合、追加のスレッドを作成する必要はありません。これにより、同時スレッド数が少ない場合に、もっとも効率的な実行が可能になります。

  スレッドプールプラグインが開始すると、それはグループあたり 1 つのスレッド (リスナースレッド) に加えてバックグラウンドスレッドを作成します。ステートメントを実行するための必要に応じて、追加のスレッドが作成されます。
- thread_pool_stall_limit システム変数の値は、先述の項目の「すぐに終了する」の意味を決定します。スレッドが停滞中とみなされるまでのデフォルトの時間は 60 ミリ秒ですが、6 秒まで設定できます。このパラメータは、サーバーのワークロードに適切なバランスがとれるように構成できます。待機の値が短いと、スレッドはよりすみやかに開始できます。短い値はデッドロック状況を回避により適しています。長い待機の値は、長時間実行するステートメントを含むワークロードで有用で、現在のステートメントの実行時に多数の新しいステートメントが開始しないようにします。
- スレッドプールは、同時の短時間実行ステートメントの数を制限することに焦点を合わせています。実行中のステートメントが停滞時間に達する前に、ほかのステートメントの実行の開始を妨げます。ステートメントが停滞時間を過ぎて実行している場合、それは続行が許可されますが、ほかのステートメントの開始は妨げられなくなります。このように、スレッドプールは、各スレッドグループに、複数の長時間実行ステートメントがあっても、複数の短時間実行ステートメントがないように努めます。必要な待機時間に対する制限がないため、長時間実行ステートメントによって、ほかのステートメントの実行が妨げられることは望ましくありません。たとえば、レプリケーションマスターで、バイナリログイベントをスレーブに送信するスレッドは、事実上永久に実行します。
- ステートメントはディスク I/O 操作またはユーザーレベルロック (行ロックまたはテーブルロック) を検出するとブロックされます。ブロックによって、スレッドグループは使用されなくなることがあるため、スレッドプールがこのグループで新しいスレッドをただちに開始して、別のステートメントを実行できるようにするた

め、スレッドプールへのコールバックがあります。ブロックされたスレッドが返されると、スレッドプールはそれをすぐに再開することを許可します。

- 優先度が高いキューと優先度が低いキューの 2 つのキューがあります。トランザクションの最初のステートメントは優先度が低いキューに入ります。トランザクションの後続のステートメントは、トランザクションが進行中 (そのステートメントが実行を開始している) 場合、優先度が高いキューに入れられ、そうでない場合は優先度が低いキューに入れられます。キューの割り当ては、thread_pool_high_priority_connection システム変数を有効にすることによって影響を受けることがあります。これにより、セッションのすべてのキューに入れられているステートメントが優先度の高いキューに入れられます。

  非トランザクションストレージエンジンまたは autocommit が有効にされている場合のトランザクションエンジンのステートメントは、この場合に各ステートメントがトランザクションであるため、優先度の低いステートメントとして扱われます。そのため、InnoDB テーブルと MyISAM テーブルに対するステートメントを組み合わせると、autocommit が有効にされていないかぎり、スレッドプールは MyISAM に対するステートメントより、InnoDB に対するステートメントを優先します。autocommit が有効にされていると、すべてのステートメントの優先度が低くなります。

- スレッドグループが実行のためにキューに入れられているステートメントを選択する場合、まず優先度が高いキューを調べて、次に優先度が低いキューを調べます。ステートメントが見つかった場合、そのキューからそれが削除され、実行が開始されます。

- ステートメントが優先度の低いキューに長くとどまりすぎた場合、スレッドプールは優先度の高いキューに移動します。thread_pool_prio_kickup_timer システム変数の値は、移動までの時間を制御します。スレッドグループごとに、最大 10 ms あたり 1 つのステートメントまたは 1 秒あたり 100 個のステートメントが優先度の低いキューから優先度の高いキューに移動されます。

- スレッドプールは、CPU キャッシュの使用を大幅に効率化するために、もっともアクティブなスレッドを再利用します。これは、パフォーマンスに大きな影響を与える小さな調整です。

- スレッドがユーザー接続からステートメントを実行している間、パフォーマンススキーマインストゥルメンテーションは、ユーザー接続にスレッドアクティビティーを報告します。それ以外の場合、パフォーマンススキーマはアクティビティーをスレッドプールに報告します。

これは、スレッドグループがステートメントを実行するために複数のスレッドを開始している状況の例です。

- 1 つのスレッドがステートメントの実行を開始しますが、長時間実行しているため、停滞中とみなされます。スレッドグループは、最初のスレッドがまだ実行中であっても、別のスレッドに別のステートメントの実行の開始を許可します。

- 1 つのスレッドがステートメントの実行を開始し、その後ブロックされ、このことをスレッドプールにレポートします。スレッドグループは、別のスレッドに別のステートメントの実行の開始を許可します。

- 1 つのスレッドがステートメントの実行を開始し、ブロックされましたが、スレッドプールのコールバックによってインストゥルメントされたコードでブロックが発生していないため、ブロックされたことをレポートしません。この場合、スレッドはスレッドグループにまだ実行中であるように見えます。ステートメントが停滞中とみなされるほどブロックが長く継続した場合、グループは、別のスレッドに別のステートメントの実行の開始を許可します。

スレッドプールは、増加する接続全体に拡張できるように設計されています。さらに、アクティブに実行しているステートメントの数を制限することから発生する可能性のあるデッドロックを回避するようにも設計されています。スレッドプールにレポートしないスレッドは、ほかのステートメントの実行を妨げないため、スレッドプールのデッドロックを引き起こすことは重要です。そのようなステートメントの例を次に示します。

- 長時間実行ステートメント。これらによって、すべてのリソースがほんの少数のステートメントで使用されることになり、ほかのすべてのステートメントのサーバーへのアクセスを妨げる可能性があります。

- バイナリログを読み取り、それをスレーブに送信するバイナリログダンプスレッド。これは、きわめて長い時間実行する長時間実行「ステートメント」の一種であり、ほかのステートメントの実行を妨げないはずです。

- MySQL Server またはストレージエンジンによって、スレッドプールにレポートされていない、行ロック、テーブルロック、またはほかの何らかのブロックアクティビティーでブロックされたステートメント。

どの場合も、デッドロックを避けるため、スレッドグループが別のステートメントの実行の開始を許可できるように、ステートメントがすぐに完了しない場合、停滞中カテゴリに移動されます。この設計により、スレッドが長時間実行するか、ブロックされた場合に、スレッドプールはスレッドを停滞中カテゴリに移動し、ステートメントの実行の残りの間、ほかのステートメントの実行を妨げません。

発生する可能性のあるスレッドの最大数は、max_connections と thread_pool_size の合計です。これは、すべての接続が実行モードにあり、グループあたりに追加のステートメントを待機する 1 つの追加スレッドが作成され

る状況で発生する可能性があります。これは必ずしも頻繁に発生する状態ではありませんが、理論的には可能性があります。

8.11.6.3 スレッドプールのチューニング

このセクションでは、秒あたりのトランザクション数などのメトリックを使用して測定された、最高のパフォーマンスを得るためのスレッドプールシステム変数の設定に関するガイドラインを提供します。

thread_pool_size はスレッドプールのパフォーマンスを制御するもっとも重要なパラメータです。それはサーバーの起動時にのみ設定できます。スレッドプールのテストにおける経験では、次のように示されます。

- プライマリストレージエンジンが InnoDB である場合、最適な thread_pool_size 設定は、16 から 36 の間になる可能性があり、もっとも一般的な最適な値は 24 から 36 になる傾向があります。36 を超える設定が最適であった状況はありませんでした。16 未満の値が最適である特殊なケースがある場合もあります。

  DBT2 や Sysbench などのワークロードの場合、InnoDB の最適な値は通常 36 くらいであるようです。著しく書き込みの多いワークロードでは、最適な設定はもっと少ない可能性があります。

- プライマリストレージエンジンが MyISAM である場合、thread_pool_size 設定はかなり小さくするべきです。4 から 8 の値で最適なパフォーマンスが得られる傾向があります。値を大きくすると、パフォーマンスにややマイナスでも劇的な影響を与える傾向はありません。

もう 1 つのシステム変数 thread_pool_stall_limit はブロックされたステートメントと長時間実行ステートメントの処理に重要です。MySQL Server をブロックするすべての呼び出しがスレッドプールにレポートされる場合、実行スレッドがブロックされるといつでもわかります。ただし、これは常には当てはまらないことがあります。たとえば、ブロックはスレッドプールコールバックによってインストゥルメントされていないコードで発生する可能性があります。そのような場合、スレッドプールはブロックされているように見えるスレッドを識別できる必要があります。これは thread_pool_stall_limit システム変数を使用してチューニングできる長さであるタイムアウトを使用して実行されます。このパラメータにより、サーバーは完全にブロックされることはありません。thread_pool_stall_limit の値は、デッドロックされたサーバーのリスクを回避するため、6 秒の上限があります。

thread_pool_stall_limit により、スレッドプールは長時間実行ステートメントを処理することもできます。長期間実行するステートメントがスレッドグループをブロックすることを許可された場合、グループに割り当てられるその他のすべての接続はブロックされ、長期間実行するステートメントが完了するまで実行を開始できません。最悪の場合、これには数時間または数日かかることもあります。

thread_pool_stall_limit の値は、その値より長く実行するステートメントが停滞中とみなされるように選択するべきです。停滞中のステートメントは、追加のコンテキストスイッチと場合によっては追加のスレッド作成が必要であるため、大量の追加のオーバーヘッドを生成します。一方、thread_pool_stall_limit パラメータを高く設定しすぎることは、長時間実行ステートメントが必要以上に長い間、多数の短時間実行ステートメントをブロックすることを意味します。待機の値が短いと、スレッドはよりすみやかに開始できます。短い値はデッドロック状況を回避により適しています。長い待機の値は、長時間実行するステートメントを含むワークロードで有用で、現在のステートメントの実行時に多数の新しいステートメントが開始しないようにします。

サーバーに負荷がかかっている場合でも、サーバーはステートメントの 99.9% が 100 ミリ秒以内に完了するワークロードを実行しており、残りのステートメントが 100 ミリ秒から 2 時間の間でまったく均等に分散してかかるものとします。この場合、thread_pool_stall_limit を 10 (100 ミリ秒を示す) に設定すると有益であると考えられます。60 ミリ秒のデフォルト値は、主にきわめて簡単なステートメントを実行するサーバーには十分です。

thread_pool_stall_limit パラメータは、サーバーのワークロードに対して適切なバランスをとることができるように、実行時に変更できます。TP_THREAD_GROUP_STATS テーブルが有効にされているとすると、次のクエリーを使用して、実行されたステートメントの停滞した部分を特定できます。

```
SELECT SUM(STALLED_QUERIES_EXECUTED) / SUM(QUERIES_EXECUTED)
FROM information_schema.TP_THREAD_GROUP_STATS;
```

この数値は可能なかぎり小さくするべきです。ステートメントの停滞の可能性を削減するには、thread_pool_stall_limit の値を増やします。

ステートメントが到着したときに、それが実際に実行を開始するまで、遅延できる最大の時間はどれくらいですか。次の条件が当てはまるとします。

- 優先度が低いキューに 200 ステートメントが入れられています。
- 優先度が高いキューに 10 ステートメントが入れられています。
- thread_pool_prio_kickup_timer は 10000 (10 秒) に設定されています。

- thread_pool_stall_limit は 100 (1 秒) に設定されています。

最悪の場合、10 個の優先度の高いステートメントは長時間実行し続ける 10 個のトランザクションを表します。そのため、最悪の場合に、優先度の高いキューには常に実行を待機しているステートメントがすでに含まれるため、このキューにステートメントが移動されません。10 秒後、新しいステートメントは優先度の高いキューに移動される資格を得ます。ただし、それが移動される前に、その前のすべてのステートメントも移動される必要があります。優先度の高いキューに移動されるのは、1 秒あたり最大 100 ステートメントであるため、これはさらに 2 秒かかる可能性があります。ステートメントが優先度の高いキューに到達したときに、多くの長時間実行ステートメントがその前にある可能性があります。最悪の場合、それらのすべてが停滞中になり、次のステートメントが優先度の高いキューから取得されるまで、ステートメントごとに 1 秒かかります。そのため、このシナリオでは、新しいステートメントが実行を開始するまで 222 秒かかります。

この例では、アプリケーションの最悪のケースを示しています。その処理方法はアプリケーションに依存します。アプリケーションの応答時間に対する要件が高い場合、おそらくそれ自体で高いレベルでユーザーを制限するはずです。そうでない場合は、スレッドプール構成パラメータを使用して、何らかの最大待機時間を設定できます。

8.12 パフォーマンスの測定 (ベンチマーク)

パフォーマンスを測定するには、次の要因を考慮します。

- ビジーでないシステムで単一の操作の速度を測定するかどうか、一連の操作 (「ワークロード」) が一定の期間でどの程度機能するか。単純なテストでは、通常 1 つの側面 (構成設定、テーブルのインデックスのセット、クエリー内の SQL 句) の変化がパフォーマンスにどのように影響するかをテストします。ベンチマークは一般に長時間実行の複雑なパフォーマンステストであり、結果によって、ハードウェアやストレージ構成などの高レベルの選択や新しい MySQL バージョンにあとどのくらいでアップグレードするかが決まります。
- ベンチマークでは、正確な実態を得るために、重いデータベースワークロードをシミュレートする必要がある場合があります。
- パフォーマンスはきわめて多くのさまざまな要因によって異なる可能性があり、数パーセントの違いが決定的勝利にならないことがあります。結果は、別の環境でテストした場合に、逆の方向に転換することもあります。
- 特定の MySQL 機能は、ワークロードに応じて、パフォーマンスに役立つ場合と役立たない場合があります。完全性のため、常にそれらの機能をオンにした状態とオフにした状態でパフォーマンスをテストします。各ワークロードで試すべきもっとも重要な 2 つの機能は、MySQL クエリーキャッシュと InnoDB テーブルのアダプティブハッシュインデックスです。

このセクションでは、1 人の開発者が実行できる単純で直接的な測定技法から、実行と結果の解釈に追加の専門技術を必要とするもっと複雑な技法に進めていきます。

8.12.1 式と関数の速度の測定

特定の MySQL 式または関数の速度を測定するには、mysql クライアントプログラムを使用して、BENCHMARK() 関数を呼び出します。その構文は BENCHMARK(loop_count,expression) です。戻り値は常に 0 ですが、mysql はステートメントの実行にどのくらいの時間を要したかを表示する行を出力します。例:

```
mysql> SELECT BENCHMARK(1000000,1+1);
+------------------------+
| BENCHMARK(1000000,1+1) |
+------------------------+
|                      0 |
+------------------------+
1 row in set (0.32 sec)
```

この結果は Pentium II 400MHz システムで取得されました。これは、MySQL がそのシステムで 1,000,000 件の単純な加算式を 0.32 秒間で実行できることを示しています。

組み込みの MySQL 関数は一般に高度に最適化されますが、例外がある場合もあります。BENCHMARK() はクエリーで特定の関数が問題になっているかどうかを調べる場合に優れたツールです。

8.12.2 MySQL ベンチマークスイート

このベンチマークスイートは、特定の SQL 実装のパフォーマンスが向上または低下する操作をユーザーに示すことを目的としています。MySQL ソース配布の sql-bench ディレクトリにあるコードと結果を確認することで、ベンチマークの動作について十分に理解できます。

このベンチマークはシングルスレッドであるため、実行される操作の最短時間を測定します。将来はこのベンチマークスイートにマルチスレッドのテストも追加する予定です。

ベンチマークスイートを使用するには、次の要件を満たす必要があります。

- ベンチマークスイートは、MySQL ソース配布によって提供されます。https://dev.mysql.com/downloads/ からリリース済みの配布をダウンロードするか、現在の開発ソースツリーを使用します。(セクション2.9.3「開発ソースツリーを使用して MySQL をインストールする」を参照してください。)
- ベンチマークスクリプトは Perl で書かれ、データベースサーバーにアクセスするために Perl DBI モジュールを使用しているため、DBI をインストールする必要があります。さらに、テスト対象のサーバーのそれぞれにサーバー固有の DBD ドライバも必要です。たとえば、MySQL、PostgreSQL、および DB2 をテストするには、DBD::mysql、DBD::Pg、DBD::DB2 のモジュールがインストールされている必要があります。セクション2.13「Perl のインストールに関する注釈」 を参照してください。

MySQL ソース配布の入手後、その sql-bench ディレクトリにあるベンチマークスイートを見つけることができます。ベンチマークテストを実行するには、MySQL を構築し、場所を sql-bench ディレクトリに変更し、run-all-tests スクリプトを実行します。

```
shell> cd sql-bench
shell> perl run-all-tests --server=server_name
```

server_name はサポートされるいずれかのサーバーの名前にするべきです。すべてのオプションとサポート対象サーバーの一覧を取得するには、このコマンドを呼び出します。

```
shell> perl run-all-tests --help
```

crash-me スクリプトも sql-bench ディレクトリにあります。crash-me では、実際のクエリーを実行することによって、データベースシステムがサポートする機能と、その性能と制限を判断しようとします。たとえば、次を判断します。

- サポートされるデータ型
- サポートされるインデックス数
- サポートされる関数
- 使用可能なクエリーの大きさ
- 使用可能な VARCHAR カラムの大きさ

ベンチマーク結果の詳細については、http://www.mysql.com/why-mysql/benchmarks/ を参照してください。

8.12.3 独自のベンチマークの使用

アプリケーションとデータベースのベンチマークを行い、ボトルネックのある場所を見つけます。1 つのボトルネックを修正 (または、それを「ダミー」モジュールで置換) することによって、次のボトルネックの識別に進むことができます。現在のアプリケーションの全体的なパフォーマンスが許容できるものであっても、いつか実際にパフォーマンスの強化が必要になった場合に、少なくとも各ボトルネックの計画を立て、解決方法を決定しておくべきです。

移植可能なベンチマークプログラムの例については、MySQL ベンチマークスイートのそれらを参照してください。セクション8.12.2「MySQL ベンチマークスイート」を参照してください。このスイートから任意のプログラムを選び、独自のニーズに合わせて変更できます。これを実行することによって、それぞれの問題に対してさまざまな解決方法を試してみて、実際にもっとも高速であるのはどれかをテストできます。

もう 1 つの無料のベンチマークスイートは Open Source Database Benchmark であり、http://osdb.sourceforge.net/ で入手できます。

システムの負荷が非常に高い場合にのみ問題が発生することはよくあることです。(テスト済みの) システムを本稼働させて、負荷の問題が発生したときに、問い合わせてくる顧客が多数いました。ほとんどの場合、パフォーマンスの問題は、高負荷時のテーブルスキャンの不良などデータベースの基本的な設計の問題か、オペレーティングシステムやライブラリの問題によると判明しています。ほとんどの場合、システムがまだ本稼働に入っていない場合の方がこれらの問題の修正がはるかに容易です。

このような問題を回避するには、可能性のある最悪の負荷でアプリケーション全体のベンチマークを行います。

- 複数のクライアントが同時にクエリーを発行して生成される高い負荷をシミュレートするには、mysqlslap プログラムが役立つ可能性があります。セクション4.5.7「mysqlslap — 負荷エミュレーションクライアント」を参照してください。
- SysBench および DBT2 などのベンチマークパッケージを試してみることもできます。これは、http://sourceforge.net/projects/sysbench/ および http://osdldbt.sourceforge.net/#dbt2 で入手できます。

これらのプログラムやパッケージはシステムを破損させる可能性があるため、それらは開発システムでのみ使用するようにしてください。

8.12.4 performance_schema によるパフォーマンスの測定

performance_schema データベースのテーブルをクエリーし、それを実行しているサーバーとアプリケーションのパフォーマンス特性に関するリアルタイムの情報を確認できます。詳細は、第22章「MySQL パフォーマンススキーマ」を参照してください。

8.12.5 スレッド情報の検査

MySQL サーバーで何が実行されているかを確認しようとする場合、プロセスリストを調査すると役立つ場合があります。これは、サーバー内で現在実行されているスレッドのセットです。プロセスリストの情報はこれらのソースから入手できます。

- SHOW [FULL] PROCESSLIST ステートメント: セクション13.7.5.30「SHOW PROCESSLIST 構文」
- SHOW PROFILE ステートメント: セクション13.7.5.32「SHOW PROFILES 構文」
- INFORMATION_SCHEMA PROCESSLIST テーブル: セクション21.15「INFORMATION_SCHEMA PROCESSLIST テーブル」
- mysqladmin processlist コマンド: セクション4.5.2「mysqladmin — MySQL サーバーの管理を行うクライアント」
- performance_schema.threads テーブル: セクション22.9.10「パフォーマンススキーマのその他のテーブル」

threads へのアクセスには相互排他ロックは必要なく、サーバーパフォーマンスへの影響は最小です。INFORMATION_SCHEMA.PROCESSLIST および SHOW PROCESSLIST は、相互排他ロックを必要とするので、負のパフォーマンスの結果になります。threads はまた、バックグラウンドスレッドに関する情報も表示しますが、INFORMATION_SCHEMA.PROCESSLIST および SHOW PROCESSLIST は表示しません。これは、threads は、ほかのスレッド情報源では行えないアクティビティーのモニターに使用できることを意味します。

自分のスレッドに関する情報はいつでも表示できます。ほかのアカウントで実行されているスレッドに関する情報を表示するには、PROCESS 権限が必要です。

プロセスリストの各エントリには、いくつかの情報が含まれています。

- Id は、スレッドに関連付けられているクライアントの接続識別子です。
- User と Host は、スレッドに関連付けられているアカウントを示します。
- db は、スレッドのデフォルトのデータベースで、または何も選択されていない場合は NULL です。
- Command と State は、スレッドが何を実行しているかを示します。

  ほとんどの状態がきわめてすばやい操作に対応します。スレッドの状態が何秒間も特定の状態にとどまっている場合は、調査が必要な問題が発生している可能性があります。
- Time は、スレッドの現在の状態がどれだけ続いているかを示します。特定の場合に、スレッドの現在の時間の概念が変わることがあります。スレッドは、SET TIMESTAMP = value によって時間を変更することがあります。マスターからのイベントを処理しているスレーブで実行しているスレッドの場合、スレッドの時間はイベント内に見つかった時間に設定されるため、スレーブではなくマスターの現在の時間を反映します。
- Info には、スレッドで実行されているステートメントのテキストが含まれるか、または何も実行されていない場合は NULL です。デフォルトでは、この値にはステートメントの先頭の 100 文字だけが含まれます。完全なステートメントを表示するには、SHOW FULL PROCESSLIST を使用します。

以下のセクションでは、Command の可能な値と、カテゴリ別にグループ化した State の値を説明します。これらの一部の値の意味は自明です。その他については追加の説明を提供しています。

8.12.5.1 スレッドのコマンド値

スレッドの Command 値は次のいずれかになります。

- Binlog Dump

  これは、バイナリログの内容をスレーブサーバーに送信するためのマスターサーバー上のスレッドです。

- Change user

  スレッドはユーザー変更操作を実行しています。

- Close stmt

  スレッドはプリペアドステートメントをクローズしています。

- Connect

  レプリケーションスレーブはそのマスターに接続されています。

- Connect Out

  レプリケーションスレーブはそのマスターに接続しています。

- Create DB

  スレッドはデータベース作成操作を実行しています。

- Daemon

  このスレッドはサーバーの内部で使用され、クライアント接続をホストするスレッドではありません。

- Debug

  スレッドはデバッグ情報を生成しています。

- Delayed insert

  スレッドは遅延挿入ハンドラです。

- Drop DB

  スレッドはデータベース削除操作を実行しています。

- Error

- Execute

  スレッドはプリペアドステートメントを実行しています。

- Fetch

  スレッドはプリペアドステートメントの実行から結果をフェッチしています。

- Field List

  スレッドはテーブルカラムの情報を取得しています。

- Init DB

  スレッドはデフォルトのデータベースを選択しています。

- Kill

  スレッドは別のスレッドを強制終了しています。

- Long Data

  スレッドはプリペアドステートメントの実行の結果から長いデータを取得しています。

- Ping

スレッドはサーバー ping 要求を処理しています。

- Prepare

  スレッドはプリペアドステートメントを準備しています。

- Processlist

  スレッドはサーバースレッドに関する情報を生成しています。

- Query

  スレッドはステートメントを実行しています。

- Quit

  スレッドは終了しています。

- Refresh

  スレッドは、テーブル、ログ、またはキャッシュをフラッシュしているか、ステータス変数またはレプリケーションサーバーの情報をリセットしています。

- Register Slave

  スレッドはスレーブサーバーを登録しています。

- Reset stmt

  スレッドはプリペアドステートメントをリセットしています。

- Set option

  スレッドはクライアントのステートメント実行オプションを設定またはリセットしています。

- Shutdown

  スレッドはサーバーをシャットダウンしています。

- Sleep

  スレッドはクライアントが新しいステートメントをそれに送信するのを待機しています。

- Statistics

  スレッドはサーバーステータス情報を生成しています。

- Table Dump

  スレッドはテーブルの内容をスレーブサーバーに送信しています。

- Time

  使用されません。

8.12.5.2 一般的なスレッドの状態

次のリストは、レプリケーションなどの特殊なアクティビティーではなく、一般的なクエリーの処理に関連付けられた、スレッドの State 値を説明しています。これらの多くは、サーバーのバグを見つけるためにのみ役立ちます。

- After create

  これは、スレッドがテーブル (内部一時テーブルも含む) を作成する際の、テーブルを作成する関数の最後に発生します。何らかのエラーのためテーブルを作成できなかった場合でも、この状態が使われます。

- altering table

  サーバーはインプレース ALTER TABLE の実行中です。

- Analyzing

  スレッドは MyISAM テーブルのキー分布を計算しています (ANALYZE TABLE などで)。

- checking permissions

  スレッドは、サーバーがステートメントを実行するために必要な権限を持っているかどうかを確認しています。

- Checking table

  スレッドはテーブルチェック操作を実行しています。

- cleaning up

  スレッドは 1 つのコマンドを処理し、メモリーの解放と特定の状態変数のリセットを準備しています。

- closing tables

  スレッドは、変更されたテーブルデータをディスクにフラッシュし、使用されたテーブルをクローズしています。これは高速の操作であるはずです。そうでない場合は、ディスクがいっぱいでないか、ディスクが著しく頻繁に使用されていないかを確認してください。

- committing alter table to storage engine

  サーバーはインプレース ALTER TABLE を終了し、結果をコミットしています。

- converting HEAP to MyISAM

  スレッドは内部一時テーブルを MEMORY テーブルからディスク上の MyISAM テーブルに変換しています。

- copy to tmp table

  スレッドは ALTER TABLE ステートメントを処理しています。この状態は、新しい構造でテーブルが作成されたあと、ただし、それに行がコピーされる前に発生します。

- Copying to group table

  ステートメントの ORDER BY と GROUP BY の基準が異なる場合、行はグループによってソートされ、一時テーブルにコピーされます。

- Copying to tmp table

  サーバーはメモリー内の一時テーブルにコピーしています。

- Copying to tmp table on disk

  サーバーはディスク上の一時テーブルにコピーしています。一時結果セットが大きくなりすぎました (セクション8.4.4「MySQL が内部一時テーブルを使用する仕組み」を参照してください)。その結果、スレッドは一時テーブルをインメモリーからディスクベースのフォーマットに変更して、メモリーを節約します。

- Creating index

  スレッドは MyISAM テーブルに対する ALTER TABLE ... ENABLE KEYS を処理しています。

- Creating sort index

  スレッドは内部一時テーブルを使用して解決される SELECT を処理しています。

- creating table

  スレッドはテーブルを作成しています。これには一時テーブルの作成が含まれます。

- Creating tmp table

  スレッドはメモリー内またはディスク上に一時テーブルを作成しています。メモリー内に作成されたテーブルがあとでディスク上のテーブルに変換される場合、その操作中の状態は Copying to tmp table on disk になります。

- deleting from main table

サーバーは複数テーブル削除の最初の部分を実行しています。最初のテーブルからのみ削除し、別の (参照) テーブルからの削除に使用されるカラムとオフセットを保存しています。

- deleting from reference tables

  サーバーは複数テーブル削除の 2 番目の部分を実行しており、別のテーブルから一致した行を削除しています。

- discard_or_import_tablespace

  スレッドは ALTER TABLE ... DISCARD TABLESPACE または ALTER TABLE ... IMPORT TABLESPACE ステートメントを処理しています。

- end

  これは、ALTER TABLE、CREATE VIEW、DELETE、INSERT、SELECT、または UPDATE ステートメントの最後、ただしクリーンアップの前に発生します。

- executing

  スレッドはステートメントの実行を開始しました。

- Execution of init_command

  スレッドは init_command システム変数の値のステートメントを実行しています。

- freeing items

  スレッドはコマンドを実行しました。この状態中に実行される項目の解放の一部には、クエリーキャッシュが含まれます。通常、この状態のあとは cleaning up になります。

- Flushing tables

  スレッドは FLUSH TABLES を実行しており、すべてのスレッドがそれぞれのテーブルをクローズするのを待っています。

- FULLTEXT initialization

  サーバーは自然言語全文検索を実行する準備をしています。

- init

  これは、ALTER TABLE、DELETE、INSERT、SELECT、または UPDATE ステートメントの初期化の前に発生します。この状態のサーバーによってとられるアクションには、バイナリログ、InnoDB ログ、および一部のクエリーキャッシュクリーンアップ操作のフラッシュが含まれます。

  end 状態では、次の操作が行われることがあります。

  - テーブルのデータが変更されたあとのクエリーキャッシュエントリの削除
  - バイナリログへのイベントの書き込み
  - BLOB 用を含むメモリーバッファーの解放

- Killed

  だれかがスレッドに KILL ステートメントを送っており、スレッドは次に強制終了フラグをチェックしたときに中止するはずです。フラグは MySQL の各主要ループ内でチェックされますが、場合によってはスレッドが停止するまでに少し時間がかかる場合があります。スレッドがほかのスレッドにロックされている場合、強制終了はほかのスレッドがその ロックを解除するとすぐに有効になります。

- logging slow query

  スレッドはステートメントを低速クエリーログに書き込んでいます。

- NULL

  この状態は SHOW PROCESSLIST 状態に使用されます。

- login

クライアントが正常に認証されるまでの接続スレッドの初期状態です。

- manage keys

  サーバーはテーブルインデックスを有効または無効にしています。

- Opening tables、Opening table

  スレッドはテーブルをオープンしようと試みています。これは、何かにオープンを妨げられないかぎり、きわめて高速な手順であるはずです。たとえば、ALTER TABLE または LOCK TABLE ステートメントは、そのステートメントが終了するまでテーブルのオープンを妨げることがあります。table_open_cache 値が十分に大きいことをチェックすることも価値があります。

- optimizing

  サーバーはクエリーの初期最適化を実行しています。

- preparing

  この状態はクエリーの最適化中に発生します。

- preparing for alter table

  サーバーはインプレース ALTER TABLE の実行を準備しています。

- Purging old relay logs

  スレッドは不要なリレーログファイルを削除しています。

- query end

  この状態は、クエリーを処理したあと、ただし freeing items 状態の前に発生します。

- Reading from net

  サーバーはネットワークからパケットを読み取っています。

- Removing duplicates

  クエリーは、MySQL が早い段階で個別の操作を最適化できなくなるような方法で SELECT DISTINCT を使用していました。このため、MySQL は結果をクライアントに送る前にすべての重複した行を削除するための追加の段階を必要とします。

- removing tmp table

  スレッドは SELECT ステートメントを処理したあとに内部一時テーブルを削除しています。一時テーブルが作成されなかった場合、この状態は使用されません。

- rename

  スレッドはテーブルの名前を変更しています。

- rename result table

  スレッドは ALTER TABLE ステートメントを処理しており、新しいテーブルを作成し、元のテーブルを置き換えるためにその名前を変更しています。

- Reopen tables

  スレッドはテーブルのロックを取得しましたが、ロックの取得後、基盤となるテーブル構造が変更されたことを認識しました。それはロックを解除し、テーブルをクローズして、再度オープンしようとしています。

- Repair by sorting

  修復コードはインデックスを作成するためにソートを使用しています。

- Repair done

  スレッドは MyISAM テーブルに対するマルチスレッドの修復を完了しました。

- Repair with keycache

  修復コードはキーキャッシュ経由で、1 つずつキーの作成を使用しています。これは Repair by sorting よりはるかに遅くなります。

- Rolling back

  スレッドはトランザクションをロールバックしています。

- Saving state

  MyISAM テーブルの修復や分析などの操作で、スレッドは新しいテーブルの状態を .MYI ファイルヘッダーに保存しています。状態には、行の数、AUTO_INCREMENT カウンタ、キー分布などの情報が含まれています。

- Searching rows for update

  スレッドは、すべての一致する行を更新する前に、それらを見つけるための第 1 フェーズを実行しています。これは、UPDATE が、関連する行を見つけるために使用されるインデックスを変更している場合に、実行される必要があります。

- Sending data

  スレッドは SELECT ステートメントの行を読み取り、処理して、データをクライアントに送信しています。この状態で行われる操作は、大量のディスクアクセス (読み取り) を実行する傾向があるため、特定のクエリーの存続期間にわたる最長時間実行状態になることがあります。

- setup

  スレッドは ALTER TABLE 操作を開始しています。

- Sorting for group

  スレッドは GROUP BY を満たすためにソートを実行しています。

- Sorting for order

  スレッドは ORDER BY を満たすためにソートを実行しています。

- Sorting index

  スレッドは MyISAM テーブルの最適化操作中に、より効率的なアクセスのためにインデックスページをソートしています。

- Sorting result

  SELECT ステートメントの場合、これは Creating sort index と似ていますが、非一時テーブルに対するものです。

- statistics

  サーバーはクエリー実行プランを開発するための統計を計算しています。スレッドが長期間この状態にある場合、サーバーはディスクに依存してほかの作業を実行している可能性があります。

- System lock

  スレッドは、テーブルの内部または外部システムロックをリクエストしようとしているか待機しています。この状態が外部ロックへのリクエストによって発生しており、同じ MyISAM テーブルにアクセスしている複数の mysqld サーバーを使用していない場合、--skip-external-locking オプションによって外部システムロックを無効にできます。ただし、外部ロックはデフォルトで無効になるため、このオプションには効果がない可能性があります。SHOW PROFILE の場合、この状態はスレッドがロックをリクエストしている (待機しているのではなく) ことを意味します。

- update

  スレッドはテーブルの更新を開始する準備ができています。

- Updating

  スレッドは更新する行を探していて、それらを更新しています。

- updating main table

サーバーは複数テーブル更新の最初の部分を実行しています。最初のテーブルのみを更新しており、別の (参照) テーブルの更新に利用されるカラムとオフセットを保存しています。

- updating reference tables

  サーバーは複数テーブル更新の 2 番目の部分を実行しており、ほかのテーブルから一致した行を更新しています。

- User lock

  スレッドは GET_LOCK() 呼び出しによってリクエストされたアドバイザリロックを、リクエストしようとしているか待機しています。SHOW PROFILE の場合、この状態はスレッドがロックをリクエストしている (待機しているのではなく) ことを意味します。

- User sleep

  スレッドは SLEEP() 呼び出しを呼び出しました。

- Waiting for commit lock

  FLUSH TABLES WITH READ LOCK はコミットロックを待機しています。

- Waiting for global read lock

  FLUSH TABLES WITH READ LOCK はグローバル読み取りロックまたはグローバル read_only システム変数が設定されるのを待機しています。

- Waiting for tables, Waiting for table flush

  スレッドは、テーブルの基盤となる構造が変更され、その新しい構造を得るためにテーブルを再度オープンする必要があるという通知を受け取りました。ただし、テーブルを再度オープンするには、ほかのすべてのスレッドが問題のテーブルをクローズするまで待機する必要があります。

  この通知は、別のスレッドが FLUSH TABLES か、問題のテーブルに次のステートメントのいずれかを使用した場合に、この通知が行われます: FLUSH TABLES tbl_name、ALTER TABLE、RENAME TABLE、REPAIR TABLE、ANALYZE TABLE、または OPTIMIZE TABLE。

- Waiting for lock_type lock

  サーバーはロックの獲得を待機しています。ここで lock_type はロックの種類を示しています。

  - Waiting for event metadata lock
  - Waiting for global read lock
  - Waiting for schema metadata lock
  - Waiting for stored function metadata lock
  - Waiting for stored procedure metadata lock
  - Waiting for table level lock
  - Waiting for table metadata lock
  - Waiting for trigger metadata lock

- Waiting on cond

  スレッドが条件が true になるのを待機している一般的な状態です。特定の状態情報は使用できません。

- Writing to net

  サーバーはネットワークにパケットを書き込んでいます。

8.12.5.3 遅延挿入スレッドの状態

これらのスレッドの状態は、DELAYED 挿入の処理に関連付けられています (セクション13.2.5.2「INSERT DELAYED 構文」を参照してください)。一部の状態は、クライアントから INSERT DELAYED ステートメントを

処理する接続スレッドに関連付けられています。ほかの状態は、行を挿入する遅延挿入ハンドラスレッドに関連付けられています。INSERT DELAYED ステートメントが発行された各テーブルに、遅延挿入ハンドラスレッドが存在します。

クライアントから INSERT DELAYED ステートメントを処理する接続スレッドに関連付けられているスレッド:

- allocating local table

  スレッドは遅延挿入ハンドラスレッドに行を提供する準備をしています。

- Creating delayed handler

  スレッドは DELAYED 挿入のハンドラを作成しています。

- got handler lock

  これは、allocating local table 状態の前、かつ waiting for handler lock 状態のあとの、接続スレッドが遅延挿入ハンドラスレッドにアクセスするときに発生します。

- got old table

  これは waiting for handler open 状態のあとに発生します。遅延挿入ハンドラスレッドは、初期化フェーズを終了したことを通知しました。これには、遅延挿入のためのテーブルのオープンが含まれます。

- storing row into queue

  スレッドは、遅延挿入ハンドラスレッドで挿入する必要のある行のリストに、新しい行を追加しています。

- waiting for delay_list

  これは、初期化フェーズ中、スレッドがテーブルの遅延挿入ハンドラスレッドを見つけようとしているときで、遅延挿入スレッドのリストにアクセスを試みる前に発生します。

- waiting for handler insert

  INSERT DELAYED ハンドラはすべての未解決の挿入を処理し、新しい挿入を待機しています。

- waiting for handler lock

  これは、接続スレッドが遅延挿入ハンドラスレッドへのアクセスを待機しているときの allocating local table 状態の前に発生します。

- waiting for handler open

  これは Creating delayed handler 状態のあとで got old table 状態の前に発生します。遅延挿入ハンドラスレッドが開始したばかりで、接続スレッドはそれが初期化されるのを待機しています。

行を挿入する遅延挿入ハンドラスレッドに関連付けられている状態:

- insert

  テーブルに行を挿入する直前に発生する状態。

- reschedule

  いくつかの行の挿入後、遅延挿入スレッドはスリープし、ほかのスレッドが作業を実行できるようにします。

- upgrading lock

  遅延挿入ハンドラは行を挿入するために、テーブルのロックを取得しようとしています。

- Waiting for INSERT

  遅延挿入ハンドラは、接続スレッドがキューに行を追加するのを待機しています (storing row into queue を参照してください)。

8.12.5.4 クエリーキャッシュスレッドの状態

これらのスレッドの状態はクエリーキャッシュに関連付けられています (セクション8.9.3「MySQL クエリーキャッシュ」を参照してください)。

- checking privileges on cached query

  サーバーはユーザーがキャッシュされたクエリー結果にアクセスする権限を持っているかどうかをチェックしています。

- checking query cache for query

  サーバーは、現在のクエリーがクエリーキャッシュに存在するかどうかをチェックしています。

- invalidating query cache entries

  基盤となるテーブルが変更されているため、クエリーキャッシュエントリは無効とマークされています。

- sending cached result to client

  サーバーはクエリーキャッシュからクエリーの結果を取得し、それをクライアントに送信しています。

- storing result in query cache

  サーバーはクエリーの結果をクエリーキャッシュに保存しています。

- Waiting for query cache lock

  この状態は、セッションがクエリーキャッシュのロックを取得するのを待機している間に発生します。これは、クエリーキャッシュを無効にする INSERT や DELETE、キャッシュされたエントリを検索する SELECT、RESET QUERY CACHE などの一部のクエリーキャッシュ操作を実行するために必要なステートメントで発生する可能性があります。

8.12.5.5 レプリケーションマスタースレッドの状態

次のリストに、マスターの Binlog Dump スレッドの State カラムに示される可能性があるもっとも一般的な状態を示します。マスターサーバーで、Binlog Dump スレッドが見られない場合、これは、レプリケーションが実行中でない、つまりスレーブが現在接続されていないことを意味します。

- Sending binlog event to slave

  バイナリログはイベントで構成され、そこではイベントが通常更新と何らかのその他の情報が追加されたものになります。スレッドはバイナリログからイベントを読み取り、現在それをスレーブに送信しています。

- Finished reading one binlog; switching to next binlog

  スレッドはバイナリログファイルの読み取りを完了し、スレーブに送信するために次のファイルをオープンしています。

- Master has sent all binlog to slave; waiting for binlog to be updated

  スレッドはバイナリログからすべての未処理の更新を読み取り、それらをスレーブに送信しました。スレッドは現在アイドルで、マスターで行われている新しい更新の結果として、新しいイベントがバイナリログに表示されるのを待機しています。

- Waiting to finalize termination

  スレッド停止中に発生するきわめて短い状態。

8.12.5.6 レプリケーションスレーブの I/O スレッド状態

次のリストに、スレーブサーバーの I/O スレッドの State カラムに表示されるもっとも一般的な状態を示します。この状態は、SHOW SLAVE STATUS によって表示される Slave_IO_State カラムにも表示されるため、そのステートメントを使用して、何が起こっているかを十分に把握できます。

- Waiting for master update

  Connecting to master の前の初期状態。

- Connecting to master

  スレッドはマスターへの接続を試みています。

- Checking master version

マスターへの接続が確立されたあとに一時的に発生する状態。

- Registering slave on master

  マスターへの接続が確立されたあとに一時的に発生する状態。

- Requesting binlog dump

  マスターへの接続が確立されたあとに一時的に発生する状態。スレッドは、マスターにそのバイナリログの内容のリクエストを送信し、リクエストしたバイナリログファイル名と位置から開始します。

- Waiting to reconnect after a failed binlog dump request

  切断のため、バイナリログダンプリクエストに失敗した場合、スレッドはスリープ中にこの状態になり、定期的に再接続を試みます。再試行の間隔は、CHANGE MASTER TO ステートメントを使用して指定できます。

- Reconnecting after a failed binlog dump request

  スレッドはマスターへの再接続を試みています。

- Waiting for master to send event

  スレッドはマスターに接続し、バイナリログイベントの到着を待機しています。マスターがアイドル状態の場合、これは長時間続く可能性があります。待機が slave_net_timeout 秒継続した場合、タイムアウトになります。その時点で、スレッドは接続が切断されているとみなし、再接続を試みます。

- Queueing master event to the relay log

  スレッドはイベントを読み取っており、SQL スレッドがそれを処理できるように、それをリレーログにコピーしています。

- Waiting to reconnect after a failed master event read

  切断のため、読み取り中にエラーが発生しました。スレッドは CHANGE MASTER TO ステートメントに設定された秒数 (デフォルト 60) の間スリープしてから、再接続を試みます。

- Reconnecting after a failed master event read

  スレッドはマスターへの再接続を試みています。ふたたび接続が確立されると、状態は Waiting for master to send event になります。

- Waiting for the slave SQL thread to free enough relay log space

  0 以外の relay_log_space_limit 値を使用しており、リレーログの組み合わせたサイズがこの値を超えるまで拡大しています。I/O スレッドは、一部のリレーログファイルを削除できるように、リレーログ内容を処理することによって SQL スレッドが十分な領域を解放するまで、待機しています。

- Waiting for slave mutex on exit

  スレッドの停止中に一時的に発生する状態。

8.12.5.7 レプリケーションスレーブ SQL スレッドの状態

次のリストに、スレーブサーバー SQL スレッドの State カラムに表示される可能性のあるもっとも一般的な状態を示します。

- Waiting for the next event in relay log

  Reading event from the relay log の前の初期状態です。

- Reading event from the relay log

  スレッドはイベントを処理できるように、イベントをリレーログから読み取りました。

- Making temporary file (append) before replaying LOAD DATA INFILE

  スレッドは LOAD DATA INFILE ステートメントを実行しており、スレーブが行を読み取る、データを格納している一時ファイルにデータを追加しています。

- Making temporary file (create) before replaying LOAD DATA INFILE

  スレッドは LOAD DATA INFILE ステートメントを実行しており、スレーブが行を読み取る、データを格納している一時ファイルを作成しています。この状態は、元の LOAD DATA INFILE ステートメントが、バージョン 5.0.3 より前の MySQL のバージョンを実行しているマスターによって記録された場合にのみ検出される可能性があります。

- Slave has read all relay log; waiting for more updates

  スレッドはリレーログファイル内のすべてのイベントを処理しており、現在 I/O スレッドが新しいイベントをリレーログに書き込むのを待機しています。

- Waiting for slave mutex on exit

  スレッド停止中に発生するきわめて短い状態。

- Waiting until MASTER_DELAY seconds after master executed event

  SQL スレッドはイベントを読み取りましたが、スレーブの遅延の満了を待機しています。この遅延は、CHANGE MASTER TO の MASTER_DELAY オプションによって設定されます。

I/O スレッドの Info カラムには、ステートメントのテキストも表示されることがあります。これは、スレッドがリレーログからイベントを読み取り、それからステートメントを抽出して、それを実行している可能性があることを示しています。

8.12.5.8 レプリケーションスレーブ接続スレッドの状態

これらのスレッドの状態はレプリケーションスレーブで発生しますが、I/O スレッドや SQL スレッドではなく、接続スレッドに関連付けられています。

- Changing master

  スレッドは CHANGE MASTER TO ステートメントを処理しています。

- Killing slave

  スレッドは STOP SLAVE ステートメントを処理しています。

- Opening master dump table

  この状態は Creating table from master dump のあとに発生します。

- Reading master dump table data

  この状態は Opening master dump table のあとに発生します。

- Rebuilding the index on master dump table

  この状態は Reading master dump table data のあとに発生します。

8.12.5.9 MySQL Cluster スレッドの状態

- Committing events to binlog
- Opening mysql.ndb_apply_status
- Processing events

  スレッドはバイナリロギングのイベントを処理しています。

- Processing events from schema table

  スレッドはスキーマレプリケーションの作業を実行しています。

- Shutting down
- Syncing ndb table schema operation and binlog

  これは、NDB のスキーマ操作の正しいバイナリログを維持するために使用されます。

- Waiting for event from ndbcluster

  サーバーは MySQL クラスタ内の SQL ノードとして機能しており、クラスタ管理ノードに接続されています。

- Waiting for first event from ndbcluster
- Waiting for ndbcluster binlog update to reach current position
- Waiting for ndbcluster to start
- Waiting for schema epoch

  スレッドはスキーマエポック (つまり、グローバルチェックポイント) を待機しています。

- Waiting for allowed to take ndbcluster global schema lock

  スレッドはグローバルスキーマロックを取得する権限を待機しています。

- Waiting for ndbcluster global schema lock

  スレッドは、別のスレッドによって保持されているグローバルスキーマロックが解放されるのを待機しています。

8.12.5.10 イベントスケジューラスレッドの状態

これらの状態は、イベントスケジューラスレッド、スケジュールされたイベントを実行するために作成されるスレッド、またはスケジューラを終了するスレッドで発生します。

- Clearing

  スケジューラスレッドまたはイベントを実行していたスレッドは終了中で、まもなく終了します。

- Initialized

  スケジューラスレッドまたはイベントを実行するスレッドが初期化されました。

- Waiting for next activation

  スケジューラには空でないイベントキューがありますが、次のアクティブ化はあとで行われます。

- Waiting for scheduler to stop

  スレッドは SET GLOBAL event_scheduler=OFF を発行し、スケジューラが停止するのを待機しています。

- Waiting on empty queue

  スケジューラのイベントキューは空で、スリープ中です。

第 9 章 言語構造

目次

9.1 リテラル値 917
9.1.1 文字列リテラル 917
9.1.2 数値リテラル 919
9.1.3 日付リテラルと時間リテラル 920
9.1.4 16 進数リテラル 921
9.1.5 boolean リテラル 922
9.1.6 ビットフィールドリテラル 922
9.1.7 NULL 値 922
9.2 スキーマオブジェクト名 923
9.2.1 識別子の修飾子 924
9.2.2 識別子の大文字と小文字の区別 925
9.2.3 識別子とファイル名のマッピング 927
9.2.4 関数名の構文解析と解決 928
9.3 予約語 931
9.4 ユーザー定義変数 939
9.5 式の構文 942
9.6 コメントの構文 943

この章では、MySQL を使用するときに、SQL ステートメントの次の要素を書き込むためのルールについて説明します。

- リテラル値: 文字列や数値など
- 識別子: データベース名、テーブル名、カラム名など
- 予約語
- ユーザー定義変数とシステム変数
- コメント

9.1 リテラル値

このセクションでは、MySQL でリテラル値を記述する方法について説明します。これらには、文字列、数値、16 進値、ブール値、および NULL が含まれます。また、このセクションでは、MySQL におけるこれらの基本データ型の処理で経験する可能性のある、微妙な違いや「なるほど」についても扱います。

9.1.1 文字列リテラル

文字列は、単一引用符 (「'」) または二重引用符 (「"」) で囲まれたバイトまたは文字のシーケンスです。例:

```
'a string'
"another string"
```

隣同士にある引用符付きの文字列は、1 つの文字列に連結されます。次の行は同等です。

```
'a string'
'a' ' ' 'string'
```

ANSI_QUOTES SQLモードを有効にしている場合は、二重引用符で囲んだ文字列は識別子として解釈されるため、文字列リテラルを囲む引用符には単一引用符だけを使用できます。

バイナリ文字列とは、文字セットや照合順序を持たないバイト文字列のことです。非バイナリ文字列とは、文字セットや照合順序を持つ文字列のことです。これらの両方の文字列タイプは、文字列単位の数値に基づいて比較されます。バイナリ文字列の場合、単位はバイトです。非バイナリ文字列の場合、単位は文字であり、マルチバイト文字をサポートする文字セットもあります。文字値の順序は、文字列照合順序の関数です。

文字列リテラルでは、オプションとして文字セットイントロデューサと COLLATE 句を指定できます。

```
[_charset_name]'string' [COLLATE collation_name]
```

例:

```
SELECT _latin1'string';
SELECT _latin1'string' COLLATE latin1_danish_ci;
```

N'literal' (または n'literal') を使用すると、各国文字セットの文字列を作成できます。次のステートメントは同等です。

```
SELECT N'some text';
SELECT n'some text';
SELECT _utf8'some text';
```

これらの文字列構文の形式の詳細は、セクション10.1.3.5「文字列リテラルの文字セットおよび照合順序」およびセクション10.1.3.6「各国文字セット」を参照してください。

NO_BACKSLASH_ESCAPES SQL モードが有効になっている場合を除いて、一部のシーケンスが文字列内で特別な意味を持ちます。これらのシーケンスはいずれも、エスケープ文字として知られるバックスラッシュ (「\」) で始まります。MySQL は、表9.1「特殊文字エスケープシーケンス」に示すエスケープシーケンスを認識します。ほかのすべてのエスケープシーケンスでは、バックスラッシュは無視されます。つまり、エスケープされた文字がエスケープされていないと解釈されます。たとえば、「\x」は単なる「x」です。これらのシーケンスでは大文字と小文字が区別されます。たとえば、「\b」はバックスペースと解釈されますが、「\B」は「B」と解釈されます。エスケープ処理は character_set_connection システム変数で指定された文字セットに応じて実行されます。セクション10.1.3.5「文字列リテラルの文字セットおよび照合順序」で説明するとおり、ほかの文字セットを示すイントロデューサが前に置かれている文字列についても同じことがいえます。

表 9.1 特殊文字エスケープシーケンス

エスケープシーケンス	シーケンスが表す文字
\0	ASCII NUL (0x00) 文字。
\'	単一引用符 (「'」) 文字。
\"	二重引用符 (「"」) 文字。
\b	バックスペース文字。
\n	改行 (ラインフィード) 文字。
\r	復帰改行文字。
\t	タブ文字。
\Z	ASCII 26 (Ctrl+Z)。表に続いて記されている注釈を参照してください。
\\	バックスラッシュ (「\」) 文字。
\%	「%」文字。表に続いて記されている注釈を参照してください。
_	「_」文字。表に続いて記されている注釈を参照してください。

ASCII 26 文字を「\Z」としてエンコードすると、Windows で ASCII 26 が END-OF-FILE を表すという問題を回避できます。mysql db_name < file_name を使用しようとすると、ファイル内の ASCII 26 が問題を引き起こします。

「\%」および「_」シーケンスは、パターンマッチングコンテキストでリテラルインスタンス「%」および「_」を検索するために使用されます (パターンマッチングコンテキスト以外ではワイルドカード文字として解釈される)。セクション12.5.1「文字列比較関数」内の LIKE 演算子に関する記述を参照してください。パターンマッチングコンテキスト以外で「\%」または「_」を使用すると、「%」、「_」ではなく、文字列「\%」、「_」として評価されます。

文字列に引用符を含める方法は、いくつかあります。

- 「'」で囲んだ文字列内で「'」を使用する場合は、「''」と記述します。
- 「"」で囲んだ文字列内で「"」を使用する場合は、「""」と記述します。
- 引用符文字の直前にエスケープ文字 (「\」) を指定します。
- 「"」で囲んだ文字列内で「'」を使用する場合、引用符を 2 つ続けて入力したり、エスケープしたりなどの特別な処理は必要はありません。同様に、「'」で囲んだ文字列内で「"」を使用する場合も、特別な処理は必要ありません。

次の SELECT ステートメントは、引用符を使用した場合とエスケープを使用した場合にどのような効果があるかを示しています。

```
mysql> SELECT 'hello', '"hello"', '""hello""', 'hel"lo', '\'hello';
+-------+---------+-----------+--------+--------+
| hello | "hello" | ""hello"" | hel'lo | 'hello |
+-------+---------+-----------+--------+--------+

mysql> SELECT "hello", '"hello"', '""hello""', "hel""lo", "\"hello";
+-------+---------+-----------+--------+--------+
| hello | 'hello' | "hello" | hel"lo | "hello |
+-------+---------+-----------+--------+--------+

mysql> SELECT 'This\nIs\nFour\nLines';
+--------------------+
| This
Is
Four
Lines |
+--------------------+

mysql> SELECT 'disappearing\ backslash';
+------------------------+
| disappearing backslash |
+------------------------+
```

文字列カラム (BLOB カラムなど) にバイナリデータを挿入する場合、ある種の文字はエスケープシーケンスで表現してください。バックスラッシュ (「\」) と、文字列を囲む引用符は、エスケープする必要があります。ある種のクライアント環境では、NUL や Ctrl+Z もエスケープする必要があります。mysql クライアントは、NUL 文字がエスケープされていない場合、これを含む引用符付きの文字列を切り捨てます。Ctrl+Z は、エスケープされていない場合、Windows で END-OF-FILE を表すと見なされる可能性があります。これらのそれぞれの文字を表すエスケープシーケンスについては、表9.1「特殊文字エスケープシーケンス」を参照してください。

アプリケーションプログラムを書く場合、MySQL Server に送信される SQL ステートメント内で文字列がデータ値として使用される前に、これらの特殊文字を含む可能性のある文字列は適切にエスケープする必要があります。これには次の 2 つの方法があります。

- 特殊文字をエスケープする関数を使用して文字列を処理します。C プログラムでは、C API 関数 mysql_real_escape_string() を使用して、文字をエスケープできます。セクション 23.7.7.54「mysql_real_escape_string()」を参照してください。ほかの SQL ステートメントを構成する SQL ステートメント内では、QUOTE() 関数を使用できます。Perl DBI インタフェースでは、quote メソッドを使用して特殊文字を適切なエスケープシーケンスに変換できます。セクション23.9「MySQL Perl API」を参照してください。ほかの言語インタフェースでも同様の機能を利用できることがあります。
- 特殊文字を明示的にエスケープする方法以外に、多くの MySQL API には、ステートメント文字列に特殊なマーカーを挿入し、ステートメントの発行時にデータ値をそれらのマーカーにバインドできるプレースホルダー機能が備わっています。この場合、値内の特殊文字のエスケープ処理は API によって自動で行われます。

9.1.2 数値リテラル

数値リテラルには、正確値 (整数と DECIMAL) リテラルと近似値 (浮動小数点) リテラルが含まれます。

整数は数字の列として表現されます。数値には小数点として「.」が含まれる場合があります。数値には、負の値または正の値を示すためにそれぞれ「-」または「+」が直前に付けられる場合があります。仮数と指数による指数表現で表される数値は、近似値の数値です。

正確値の数値リテラルには、整数部または小数部、あるいはその両方が含まれています。これらには符号を付けることができます。例: 1、.2、3.4、-5、-6.78、+9.10。

近似値の数値リテラルは仮数と指数による指数表現で表されます。一方または両方の部分に符号を付けることができます。例: 1.2E3、1.2E-3、-1.2E3、-1.2E-3。

2 つの数値が同じように見えても、別々に扱われることがあります。たとえば、2.34 は (固定小数点の) 正確値ですが、2.34E0 は (浮動小数点の) 近似値です。

DECIMAL データ型は固定小数点型で、計算は正確です。MySQL では、DECIMAL 型には、NUMERIC、DEC、FIXED という複数のシノニムがあります。整数型も正確値型です。正確値計算の詳細は、セクション12.20「高精度計算」を参照してください。

FLOAT データ型および DOUBLE データ型は浮動小数点型で、計算によって近似値が得られます。MySQL では、FLOAT または DOUBLE のシノニムである型は DOUBLE PRECISION および REAL です。

浮動小数点のコンテキストで整数を使用することもできます。この場合、整数は同等の浮動小数点数として解釈されます。

9.1.3 日付リテラルと時間リテラル

日付値と時間値は、値の正確型とほかの要因に応じて、引用符付きの文字列や数値など、複数の形式で表現できます。たとえば、MySQL が日付を予想するコンテキストでは、'2015-07-21'、'20150721'、20150721 のいずれも日付と解釈します。

このセクションでは日付リテラルと時間リテラルの許容される形式について説明します。許可されている値の範囲など、時間データ型の詳細は、次のセクションを参照してください。

- セクション11.1.2「日付と時間型の概要」
- セクション11.3「日付と時間型」

標準 SQL と ODBC の日付および時間リテラル　　標準 SQL では、型キーワードおよび文字列を使用して、時間リテラルを指定できます。キーワードと文字列の間の空白はオプションです。

```
DATE 'str'
TIME 'str'
TIMESTAMP 'str'
```

MySQL は、これらの構造と、対応する ODBC 構文も認識します。

```
{ d 'str' }
{ t 'str' }
{ ts 'str' }
```

MySQL 5.6.4 より前では、MySQL は型キーワードを無視し、前述の各構造は、型が VARCHAR の文字列値 'str' を生成します。

5.6.4 以降、MySQL は型キーワードを使用し、これらの構造はそれぞれ、後続の小数秒部分が指定されている場合はこの部分を含む DATE、TIME、および DATETIME の値を生成します。DATETIME には、標準 SQL の TIMESTAMP 型により密接に対応する範囲があり、ここには 0001 から 9999 の年範囲が含まれるので、TIMESTAMP 構文は、MySQL で DATETIME 値を生成します。(MySQL の TIMESTAMP 年範囲は 1970 から 2038 です。)

日時コンテキストでの文字列リテラルと数値リテラル　　MySQL は次の形式で DATE 値を認識します。

- 'YYYY-MM-DD' または 'YY-MM-DD' 形式の文字列として。「緩やかな」構文が許可されます。どの句読点文字でも、日付部分間の区切り文字として使用できます。たとえば、'2012-12-31'、'2012/12/31'、'2012^12^31'、および '2012@12@31' は同等です。
- 'YYYYMMDD' または 'YYMMDD' 形式の、区切り文字がない文字列 (日付として適切なもの) として。たとえば、'20070523' と '070523' は '2007-05-23' として解釈されますが、'071332' は正しくないため (月と日の部分が不適切)、'0000-00-00' になります。
- YYYYMMDD または YYMMDD 形式の数値 (日付として適切なもの) として。たとえば、19830905 と 830905 は '1983-09-05' として解釈されます。

MySQL は次の形式で DATETIME および TIMESTAMP 値を認識します。

- 'YYYY-MM-DD HH:MM:SS' または 'YY-MM-DD HH:MM:SS' 形式の文字列として。「緩やかな」構文はここでも許可されます。どの句読点文字でも、日付部分または時間部分の間の区切り文字として使用できます。たとえば、'2012-12-31 11:30:45'、'2012^12^31 11+30+45'、'2012/12/31 11*30*45'、および '2012@12@31 11^30^45' は同等です。

  日付および時間の部分と小数秒部分との間の区切り文字として認識される唯一の文字が小数点です。

  日付部分と時間部分は、空白ではなく T で区切ることもできます。たとえば、'2012-12-31 11:30:45' と '2012-12-31T11:30:45' は同等です。
- 'YYYYMMDDHHMMSS' または 'YYMMDDHHMMSS' 形式の、区切り文字がない文字列 (日付として適切なもの) として。たとえば、'20070523091528' と '070523091528' は '2007-05-23 09:15:28' として解釈されますが、'071122129015' は正しくないため (分の部分が不適切)、'0000-00-00 00:00:00' になります。
- YYYYMMDDHHMMSS または YYMMDDHHMMSS 形式の数値 (日付として適切なもの) として。たとえば、19830905132800 と 830905132800 は '1983-09-05 13:28:00' として解釈されます。

DATETIME または TIMESTAMP 値には、マイクロ秒 (6 桁) までの精度の後続の小数秒部分を含めることができます。小数部は、常に時間の残りの部分から小数点で区分する必要があります。これ以外の小数秒区切り文字は認識されません。MySQL の小数秒のサポートの詳細は、セクション11.3.6「時間値での小数秒」を参照してください。

2 桁の年を含む日付の値は、世紀が不明なためあいまいです。MySQL は次のルールを使用して 2 桁の年の値を解釈します。

- 70-99 の範囲の値は 1970-1999 に変換されます。
- 00-69 の範囲の値は 2000-2069 に変換されます。

セクション11.3.8「日付での 2 桁の年」も参照してください。

日付部分の区切り文字を含む文字列として指定される値の場合、10 未満の月または日の値に 2 桁を指定する必要はありません。'2015-6-9' は '2015-06-09' と同じです。同様に、時間部分の区切り文字を含む文字列として指定される値の場合、10 未満の時、分、または秒の値に 2 桁を指定する必要はありません。'2015-10-30 1:2:3' は '2015-10-30 01:02:03' と同じです。

数値として指定する値は、6、8、12、14 のいずれかの桁数にするようにしてください。数値を 8 桁または 14 桁の長さにすると、YYYYMMDD または YYYYMMDDHHMMSS 形式であり、最初の 4 桁が年であると想定されます。数値を 6 桁または 12 桁の長さにすると、YYMMDD または YYMMDDHHMMSS 形式であり、最初の 2 桁が年であると想定されます。これらの長さではない数字は、いちばん近い長さまで先行ゼロで埋められているかのように解釈されます。

区切り文字がない文字列として指定された値は、その長さに従って解釈されます。8 または 14 文字の長さの文字列の場合、年は最初の 4 文字で表されていると見なされます。そうでなければ、年は最初の 2 文字で表されていると見なされます。文字列は、その文字列に存在するだけの部分について、左から右に順番に、年、月、日、時、分、秒として解釈されます。これは、6 文字より少ない文字列は利用してはいけないということを意味します。たとえば、1999 年 3 月を表すと考えて '9903' を指定しても、MySQL は「ゼロ」日付値に変換します。これは、年および月の値は 99 と 03 であるけれども、日の部分が完全に欠落しているために起こります。ただし、明示的に値ゼロを指定することによって、欠落している月や日の部分を表すことができます。たとえば、'1999-03-00' の値を挿入するには、'990300' を使用します。

MySQL は次の形式で TIME 値を認識します。

- 'D HH:MM:SS' 形式の文字列として。'HH:MM:SS'、'HH:MM'、'D HH:MM'、'D HH'、'SS' のいずれかの「緩やかな」構文も使用できます。この場合、D は日を表し、0 から 34 の値を指定できます。
- 'HHMMSS' 形式の区切り文字がない文字列 (時間として適切なもの) として。たとえば、'101112' は '10:11:12' として認識されますが、'109712' は正しくないため (分の部分が不適切)、'00:00:00' になります。
- HHMMSS 形式の数値 (時間として適切なもの) として。たとえば、101112 は '10:11:12' として認識されます。SS、MMSS、HHMMSS の代替形式も認識されます。

後続の小数秒部分は、'D HH:MM:SS.fraction'、'HH:MM:SS.fraction'、'HHMMSS.fraction'、および HHMMSS.fraction の時間形式で認識されます。ここで、fraction はマイクロ秒 (6 桁) までの精度で表される小数部分です。小数部は、常に時間の残りの部分から小数点で区分する必要があります。これ以外の小数秒区切り文字は認識されません。MySQL の小数秒のサポートの詳細は、セクション11.3.6「時間値での小数秒」を参照してください。

時間部分の区切り文字を含む文字列として指定される TIME 値の場合、10 未満の時、分、秒の値に 2 桁を指定する必要はありません。'8:3:2' は '08:03:02' と同じです。

9.1.4 16 進数リテラル

MySQL では、X'val'、x'val'、または 0xval 形式を使って記述された 16 進値をサポートしています。この場合、val には 16 進数字 (0..9、A..F) を指定します。数字の大文字と小文字は区別されません。X'val' または x'val' 形式で値を記述する場合、val には偶数の桁を入れる必要があります。0xval 構文を使用して値を記述する場合、奇数の桁が含まれる値は、追加の先行 0 が付いているものとして扱われます。たとえば、0x0a と 0xaaa は、0x0a と 0x0aaa として解釈されます。

数値のコンテキストでは、16 進値は整数 (64 ビット精度) のように機能します。文字列のコンテキストでは、16 進値はバイナリ文字列のように機能します。この場合、16 進数の各ペアが 1 文字に変換されます。

```
mysql> SELECT X'4D7953514C';
    -> 'MySQL'
mysql> SELECT 0x0a+0;
```

```
    -> 10
mysql> SELECT 0x5061756c;
    -> 'Paul'
```

16 進値のデフォルトのデータ型は文字列です。この値が数値として扱われるようにしたい場合は、CAST(... AS UNSIGNED) を使用できます。

```
mysql> SELECT 0x41, CAST(0x41 AS UNSIGNED);
    -> 'A', 65
```

X'hexstring' 構文は標準 SQL に基づいています。0x 構文は ODBC に基づいています。16 進文字列は、BLOB カラムの値を提供するために、ODBC によって使用されることがよくあります。

文字列または数値を 16 進形式の文字列に変換するには、HEX() 関数を使用できます。

```
mysql> SELECT HEX('cat');
    -> '636174'
mysql> SELECT 0x636174;
    -> 'cat'
```

9.1.5 boolean リテラル

TRUE および FALSE 定数はそれぞれ 1 と 0 として評価されます。定数名は大文字でも小文字でも記述できます。

```
mysql> SELECT TRUE, true, FALSE, false;
    -> 1, 1, 0, 0
```

9.1.6 ビットフィールドリテラル

ビットフィールド値は、b'value' または 0bvalue 表記を使用して記述できます。value は、0 と 1 で書かれた 2 進値です。

ビットフィールド表記は BIT カラムに割り当てる値を指定するのに便利です。

```
mysql> CREATE TABLE t (b BIT(8));
mysql> INSERT INTO t SET b = b'11111111';
mysql> INSERT INTO t SET b = b'1010';
mysql> INSERT INTO t SET b = b'0101';
```

ビット値は 2 進値として返されます。それらを出力可能な形式で表示するには、0 を追加するか、BIN() などの変換関数を使用します。変換された値には上位の 0 ビットは表示されません。

```
mysql> SELECT b+0, BIN(b+0), OCT(b+0), HEX(b+0) FROM t;
+------+----------+----------+----------+
| b+0  | BIN(b+0) | OCT(b+0) | HEX(b+0) |
+------+----------+----------+----------+
|  255 | 11111111 | 377      | FF       |
|   10 | 1010     | 12       | A        |
|    5 | 101      | 5        | 5        |
+------+----------+----------+----------+
```

ユーザー変数に割り当てられたビット値は、バイナリ文字列として扱われます。ビット値を数値としてユーザー変数に割り当てるには、CAST() または +0 を使用します。

```
mysql> SET @v1 = 0b1000001;
mysql> SET @v2 = CAST(0b1000001 AS UNSIGNED), @v3 = 0b1000001+0;
mysql> SELECT @v1, @v2, @v3;
+------+------+------+
| @v1  | @v2  | @v3  |
+------+------+------+
| A    |   65 |   65 |
+------+------+------+
```

9.1.7 NULL 値

NULL 値は「データなし」を意味します。NULL は大文字と小文字のどちらでも記述できます。シノニムは \N (大文字と小文字を区別) です。

LOAD DATA INFILE または SELECT ... INTO OUTFILE で実行されるテキストファイルのインポートまたはエクスポート操作の場合、NULL は \N シーケンスで表現されます。セクション13.2.6「LOAD DATA INFILE 構文」を参照してください。

NULL 値は、数値型での 0 や文字列型での空文字列などの値とは異なります。詳細は、セクションB.5.5.3「NULL 値に関する問題」を参照してください。

9.2 スキーマオブジェクト名

データベース、テーブル、インデックス、カラム、エイリアス、ビュー、ストアドプロシージャー、パーティション、テーブルスペース、その他のオブジェクト名など、MySQL 内のある種のオブジェクトは、識別子として知られています。このセクションでは、MySQL で識別子について許可されている構文について説明します。セクション9.2.2「識別子の大文字と小文字の区別」では、どのタイプの識別子がどの条件下で大文字と小文字を区別するかについて説明します。

識別子は引用符で囲むことも囲まないこともあります。識別子に特殊文字が含まれている場合、または識別子が予約語である場合、その識別子を参照するときは必ず引用符で囲む必要があります。(例外: 修飾名内でピリオドのあとに続く予約語は識別子である必要があるため、引用符で囲む必要はありません。)予約語はセクション9.3「予約語」に記載されています。

識別子は内部で Unicode に変換されます。以下の文字を含めることができます。

- 引用符で囲まれていない識別子で許可される文字。
  - ASCII: [0-9,a-z,A-Z$_] (基本的なラテン文字、0-9 の数字、ドル、下線)
  - 拡張: U+0080 ..U+FFFF
- 引用符で囲まれている識別子で許可される文字には、U+0000 を除き、完全な Unicode Basic Multilingual Plane (BMP) が含まれます。
  - ASCII: U+0001 ..U+007F
  - 拡張: U+0080 ..U+FFFF
- ASCII NUL (U+0000) と補助文字 (U+10000 以上) は、引用符で囲まれた識別子または引用符で囲まれていない識別子では許可されません。
- 識別子は数字で始めることができますが、引用符で囲まれていないかぎり、数字のみで構成することはできません。
- データベース名、テーブル名、およびカラム名は、空白文字で終えることはできません。

識別子引用符文字は逆引用符 (「`」) です。

```
mysql> SELECT * FROM `select` WHERE `select`.id > 100;
```

ANSI_QUOTES SQL モードが有効になっている場合、二重引用符内で識別子を引用符で囲むことも許可されています。

```
mysql> CREATE TABLE "test" (col INT);
ERROR 1064: You have an error in your SQL syntax...
mysql> SET sql_mode='ANSI_QUOTES';
mysql> CREATE TABLE "test" (col INT);
Query OK, 0 rows affected (0.00 sec)
```

ANSI_QUOTES モードでは、サーバーは二重引用符で囲まれた文字列を識別子として解釈します。この結果、このモードが有効になっているときには、文字列リテラルは単一引用符で囲む必要があります。二重引用符で囲むことはできません。サーバー SQL モードは、セクション5.1.7「サーバー SQL モード」で説明しているように制御されます。

識別子が引用符で囲まれていれば、識別子引用符文字を識別子内に含めることができます。識別子内に含める文字が識別子自体を囲むのに使用している引用符と同じ場合、文字を二重にする必要があります。次のステートメントは、c"d という名前のカラムを含んだ a`b という名前のテーブルを作成します。

```
mysql> CREATE TABLE `a``b` (`c"d` INT);
```

クエリーの選択リストで、引用したカラムエイリアスを指定するには、識別子または文字列引用文字を使用します。

```
mysql> SELECT 1 AS `one`, 2 AS 'two';
+-----+-----+
```

```
| one | two |
+-----+-----+
|   1 |   2 |
+-----+-----+
```

ステートメント内のどこに指定する場合でも、エイリアスへの引用した参照には、識別子引用符を使用する必要があります。そうしないと、参照は文字列リテラルとして扱われます。

Me や MeN (この場合の M と N は整数) で始まる名前を使用しないことが推奨されています。たとえば、1e を識別子として使用しないでください。これは、1e+3 などの式があいまいになるためです。コンテキストに応じて、式 1e + 3 として、または数値 1e+3 として解釈される場合があります。

テーブルネーム名を作成するのに MD5() を使用する場合は注意が必要です。なぜなら、これは、前述のような不正な形式やあいまいな形式で名前を生成する可能性があるからです。

ユーザー変数は、識別子または識別子の一部として SQL ステートメントの中で直接使用することはできません。回避策の詳細と例については、セクション9.4「ユーザー定義変数」を参照してください。

セクション9.2.3「識別子とファイル名のマッピング」で説明しているように、データベース名とテーブル名内の特殊文字は、対応するファイルシステム名でエンコードされます。特殊文字を含む古いバージョンの MySQL のデータベースまたはテーブルがあり、それの基になるディレクトリ名またはファイル名が新しいエンコーディングを使用するように更新されていない場合、サーバーはそれらの名前に #mysql50# というプリフィクスを付けて表示します。このような名前の参照、または新しいエンコーディングへの変換の詳細は、そのセクションを参照してください。

次の表には、識別子のタイプごとの最大の長さが示されています。

識別子	最大の長さ (文字)
Database	64 (NDB ストレージエンジン: 63)
テーブル	64 (NDB ストレージエンジン: 63)
カラム	64
インデックス	64
制約	64
ストアドプログラム	64
ビュー	64
テーブルスペース	64
サーバー	64
ログファイルグループ	64
エイリアス	256 (表のあとの例外を参照してください)
複合ステートメントラベル	16

CREATE VIEW ステートメント内のカラム名に対するエイリアスは、(256 文字の最大のエイリアス長ではなく) 64 文字の最大のカラム長に対してチェックされます。

識別子は Unicode (UTF-8) を使用して格納されます。これは、.frm ファイル内に格納されたテーブル定義の識別子と、mysql データベース内の付与テーブルに格納された識別子に適用されます。付与テーブル内の識別子文字列カラムのサイズは文字数で測定されます。これらのカラムに格納されている値に許可されている文字数を減らすことなく、マルチバイト文字を使用できます。これは MySQL 4.1 より前には当てはまりません。前述のように、許容されている Unicode 文字は、Basic Multilingual Plane (BMP) の文字です。補助文字は許可されません。

MySQL Cluster では、データベースおよびテーブルの名前に 63 文字の最大長を課します。セクション18.1.6.5「MySQL Cluster 内のデータベースオブジェクトに関する制限」を参照してください。

9.2.1 識別子の修飾子

MySQL では、単一の識別子または複数の識別子から構成される名前を使用できます。複数部分名のコンポーネントは、ピリオド (「.」) 文字で区切る必要があります。複数部分名の最初の部分は、最後の識別子が解釈されるコンテキストに影響を与える修飾子として機能します。

MySQL では、次の形式を使用してテーブル内のカラムを参照できます。

カラム参照	意味
col_name	ステートメントで使用されるテーブルのカラム col_name には、その名前のカラムが含まれています。
tbl_name.col_name	デフォルトデータベースのテーブル tbl_name 内のカラム col_name。
db_name.tbl_name.col_name	データベース db_name のテーブル tbl_name 内のカラム col_name。

修飾子文字は別個のトークンであり、関連付けられた識別子と隣接する必要はありません。たとえば、tbl_name.col_name と tbl_name . col_name は同等です。

複数部分名のコンポーネントを引用符で囲む必要がある場合、名前全体を引用符で囲むのではなく、各コンポーネントを個別に引用符で囲んでください。たとえば、`my-table.my-column` ではなく、`my-table`.`my-column` と記述します。

修飾名内でピリオドのあとに続く予約語は識別子である必要があるため、そのコンテキストでは引用符で囲む必要はありません。

参照があいまいでないかぎり、ステートメント内のカラム参照の前に tbl_name や db_name.tbl_name プリフィクスを付ける必要はありません。たとえば、テーブル t1 と t2 のそれぞれにカラム c があり、t1 と t2 の両方を使用する SELECT ステートメントで c を取得するとします。この場合、c はステートメント内で使用されるテーブルの中で一意でないため、あいまいになります。これを t1.c または t2.c としてテーブル名で修飾し、どちらのテーブルを指しているかを示す必要があります。同様に、同じステートメント内のデータベース db1 のテーブル t とデータベース db2 のテーブル t から取得するには、それぞれのテーブルのカラムを db1.t.col_name と db2.t.col_name として参照する必要があります。

構文 .tbl_name は、デフォルトデータベースのテーブル tbl_name を意味します。一部の ODBC プログラムではテーブル名の先頭に「.」文字が付けられるため、ODBC 互換性を確保するためにこの構文が認められています。

9.2.2 識別子の大文字と小文字の区別

MySQL において、データベースはデータディレクトリ内のディレクトリに対応しています。データベース内の各テーブルも、データベースディレクトリ内の少なくとも 1 つ (ストレージエンジンによってはそれ以上) のファイルに対応しています。トリガーもファイルに対応しています。この結果、基になるオペレーティングシステムで大文字と小文字が区別されるかどうかが、データベース名、テーブル名、およびトリガー名で大文字と小文字が区別されるかどうかに影響します。これは、Windows ではこれらの名前は大文字と小文字が区別されませんが、多くの Unix では大文字と小文字が区別されることを意味します。ただし、注意が必要な例外の 1 つに OS X があります。OS X は Unix をベースにしますが、大文字と小文字が区別されないデフォルトのファイルシステムタイプ (HFS+) を使用します。ただし、OS X は UFS ボリュームもサポートしています。UFS ボリュームでは Unix の場合と同じように大文字と小文字が区別されます。セクション1.7.1「標準 SQL に対する MySQL 拡張機能」を参照してください。このセクションで後述するように、lower_case_table_names システム変数も、サーバーが識別子の大文字と小文字をどのように扱うかに影響を与えます。

> **注記**
>
> 一部のプラットフォームでは、データベース名、テーブル名、およびトリガー名で大文字と小文字が区別されませんが、これらの名前の 1 つを参照するのに、同じステートメント内で大文字と小文字が異なる名前を使用しないでください。次のステートメントは、同じテーブルを my_table および MY_TABLE として参照するため、機能しません。
>
> ```
> mysql> SELECT * FROM my_table WHERE MY_TABLE.col=1;
> ```

カラム名、インデックス名、ストアドルーチン名、およびイベント名は、どのプラットフォームでも大文字と小文字が区別されません。カラムのエイリアスも同様です。

ただし、ログファイルグループの名前では大文字と小文字が区別されます。これは標準 SQL とは異なります。

デフォルトで、Unix ではテーブルのエイリアスは大文字と小文字が区別されますが、Windows または OS X では区別されません。Unix では、次のステートメントは、エイリアスを a と A の両方で参照しているので機能しません。

```
mysql> SELECT col_name FROM tbl_name AS a
    -> WHERE a.col_name = 1 OR A.col_name = 2;
```

ただし、Windows ではこの同じステートメントは許可されます。このような違いで生じる問題を回避するために、データベースとテーブルの作成および参照では常に小文字の名前を使用するなどの一貫した規則を設けることをお勧めします。この規則は移植性と使いやすさを最大限にするために推奨されています。

テーブル名やデータベース名がどのようにディスクに格納され、MySQL で使用されるかは、lower_case_table_names システム変数の影響を受けます。このシステム変数は、mysqld の起動時に設定できます。lower_case_table_names には、次の表に示す値を設定できます。この変数は、トリガー識別子の大文字と小文字の区別には影響を及ぼしません。Unix では、lower_case_table_names のデフォルト値は 0 です。Windows では、デフォルト値は 1 です。OS X では、デフォルト値は 2 です。

値	意味
0	テーブル名とデータベース名は、CREATE TABLE または CREATE DATABASE ステートメントで指定された大文字または小文字を使用してディスク上に格納されます。名前比較では大文字と小文字が区別されます。大文字小文字を区別しないファイル名を持つシステム (Windows や OS X など) で MySQL を実行する場合、この変数を 0 に設定しないでください。大文字と小文字を区別しないファイルシステムで --lower-case-table-names=0 を使用して強制的にこの変数を 0 に設定し、大文字と小文字を変えて MyISAM テーブル名にアクセスした場合、インデックスが破損することがあります。
1	テーブル名はディスク上に小文字で格納され、名前比較では大文字と小文字は区別されません。MySQL では、保存およびルックアップ時にすべてのテーブル名が小文字に変換されます。この動作はデータベース名やテーブルエイリアスにも適用されます。
2	テーブル名とデータベース名は、CREATE TABLE または CREATE DATABASE ステートメントで指定された大文字または小文字を使用してディスク上に格納されますが、MySQL ではルックアップ時に小文字に変換されます。名前比較では大文字と小文字が区別されません。これは大文字と小文字が区別されないファイルシステムでのみ機能します。InnoDB テーブル名は lower_case_table_names=1 のように、小文字で格納されます。

MySQL を 1 つのプラットフォームでのみ使用している場合は通常、lower_case_table_names 変数をデフォルト値から変更する必要はありません。ただし、ファイルシステム上の大文字と小文字の区別が異なるプラットフォーム間でテーブルを転送する場合は、問題が生じる可能性があります。たとえば、Unix 上では my_table と MY_TABLE という名前の異なる 2 つのテーブルを使用できますが、Windows 上ではこれらは同一のものとして扱われます。データベース名やテーブル名の大文字と小文字の区別が原因で発生するデータ転送の問題を回避するには、次の 2 つのオプションがあります。

- lower_case_table_names=1 を全システムで使用してください。これの主な欠点は、SHOW TABLES または SHOW DATABASES を使用したときに、元の大文字または小文字で名前が表示されないことです。
- Unix 上では lower_case_table_names=0 を、Windows 上では lower_case_table_names=2 を使用してください。これでデータベース名とテーブル名の大文字と小文字の区別が保持されます。この欠点は、ユーザーのステートメントが、Windows 上で正しい大文字または小文字でデータベース名およびテーブル名を常に参照していることを確認する必要があることです。大文字と小文字が区別される Unix にステートメントを転送する場合、大文字と小文字が正しくなければこのステートメントは機能しません。

  例外: InnoDB テーブルを使用している場合、データ転送に関するこのような問題を避けるには、すべてのプラットフォーム上で lower_case_table_names を 1 に設定して、強制的に名前が小文字に変換されるようにしてください。

Unix 上で lower_case_table_names システム変数を 1 に設定する場合は、最初に古いデータベース名とテーブル名を小文字に変換してから、mysqld を停止し、新しい変数設定で再起動する必要があります。個々のテーブルに対してこれを行うには、RENAME TABLE を使用します。

```
RENAME TABLE T1 TO t1;
```

1 つ以上のデータベース全体を変換するには、lower_case_table_names の設定前にこれらをダンプし、続いてデータベースを削除して、lower_case_table_names の設定後にこれらをリロードします。

1. mysqldump を使用して各データベースをダンプします。

   ```
   mysqldump --databases db1 > db1.sql
   mysqldump --databases db2 > db2.sql
   ...
   ```

   再作成する必要のあるデータベースごとにこれを行います。

2. DROP DATABASE を使用して各データベースを削除します。
3. サーバーを停止し、lower_case_table_names を設定して、サーバーを再起動します。
4. データベースごとにダンプファイルをリロードします。lower_case_table_names が設定されているので、それぞれのデータベース名とテーブル名は、再作成されるときに小文字に変換されます。

   ```
   mysql < db1.sql
   ```

```
mysql < db2.sql
...
```

バイナリ照合順序に応じて大文字形式が同等である場合、オブジェクト名は複製と見なされる場合があります。これは、カーソル、条件、プロシージャー、関数、セーブポイント、ストアドルーチンパラメータ、ストアドプログラムローカル変数、およびプラグインの名前にも当てはまります。カラム、制約、データベース、パーティション、PREPARE を使用して準備されたステートメント、テーブル、トリガー、ユーザー、およびユーザー定義変数の名前には当てはまりません。

ファイルシステムの大文字と小文字の区別は、INFORMATION_SCHEMA テーブルの文字列カラムでの検索に影響する場合があります。詳細は、セクション10.1.7.9「照合順序と INFORMATION_SCHEMA 検索」を参照してください。

9.2.3 識別子とファイル名のマッピング

データベース識別子やテーブル識別子とファイルシステム内の名前との間には対応があります。基本構造では、MySQL は各データベースをデータディレクトリ内のディレクトリとして表現し、各テーブルを適切なデータベースディレクトリ内の 1 つ以上のファイルで表現します。テーブル形式のファイル (.FRM) の場合、データは常にこの構造と場所に格納されます。

データファイルとインデックスファイルの場合、ディスク上の正確な表現はストレージエンジンによって異なります。これらのファイルを FRM ファイルと同じ場所に格納することも、情報を別のファイルに格納することもできます。InnoDB データは InnoDB データファイルに格納されます。InnoDB でテーブルスペースを使用する場合は、新たに作成した特定のテーブルスペースファイルが代わりに使用されます。

データベース識別子またはテーブル識別子では、ASCII NUL (0x00) を除いたすべての文字が正当です。MySQL では、データベースディレクトリやテーブルファイルを作成するとき、対応するファイルシステムオブジェクト内で問題のある文字をすべてエンコードします。

- 基本的なラテン文字 (a..zA..Z)、数字 (0..9)、および下線 (_) はそのまま エンコードされます。このため、それらが大文字と小文字を区別するかどうかは、ファイルシステムの特性に直接依存します。
- 大文字と小文字のマッピングを持つアルファベット起源のほかの国の文字はすべて、次の表に示すように エンコードされます。コード範囲カラムの値は UCS-2 値です。

コード範囲	パターン	番号	使用	未使用	ブロック
00C0..017F	[@][0..4][g..z]	5*20= 100	97	3	補足ラテン語-1 + 拡張ラテン語-A
0370..03FF	[@][5..9][g..z]	5*20= 100	88	12	ギリシア語およびコプト語
0400..052F	[@][g..z][0..6]	20*7= 140	137	3	キリル文字 + 補足キリル文字
0530..058F	[@][g..z][7..8]	20*2= 40	38	2	米語
2160..217F	[@][g..z][9]	20*1= 20	16	4	数の形式
0180..02AF	[@][g..z][a..k]	20*11=220	203	17	拡張ラテン語-B + 拡張 IPA
1E00..1EFF	[@][g..z][l..r]	20*7= 140	136	4	拡張ラテン語追加
1F00..1FFF	[@][g..z][s..z]	20*8= 160	144	16	拡張ギリシャ語
....	[@][a..f][g..z]	6*20= 120	0	120	RESERVED
24B6..24E9	[@][@][a..z]	26	26	0	囲み文字
FF21..FF5A	[@][a..z][@]	26	26	0	全角と半角

シーケンス内の 1 バイトが大文字と小文字の区別をエンコードします。例: LATIN CAPITAL LETTER A WITH GRAVE は @0G としてエンコードされ、LATIN SMALL LETTER A WITH GRAVE は @0g としてエンコードされます。ここでは、3 番目のバイト (G または g) が大文字と小文字の区別を示します。(大文字と小文字を区別しないファイルシステムでは、両文字は同じ文字として扱われます。)

言語ブロックの中にはキリル文字のように、2 番目のバイトが大文字と小文字の区別を決定することもあります。補足ラテン語 1 などのほかの言語ブロックでは、3 番目のバイトが大文字と小文字の区別を決定します。シーケンス内の 2 バイトが文字の場合は (拡張ギリシャ語など)、いちばん左の文字が大文字と小文字の区別を表します。ほかの文字バイトはすべて、小文字である必要があります。

- 下線 (_) を除くすべての文字以外のキャラクタは、大文字と小文字のマッピングのないアルファベット起源の文字 (ヘブライ語など) とともに、a..f の 16 進値に小文字を使用した 16 進表現を使用してエンコードされます。

  ```
  0x003F -> @003f
  0xFFFF -> @ffff
  ```

  16 進値は、ucs2 ダブルバイト文字セット内のキャラクタ値に対応します。

Windows では、nul、prn、aux などの一部の名前は、サーバーが対応するファイルまたはディレクトリを作成するときに、@@@ を名前に付加することによってエンコードされます。これは、対応するデータベースオブジェクトのプラットフォーム間での移植性のためにすべてのプラットフォームで行われます。

MySQL 5.1.6 より前のバージョンのデータベースやテーブル内で特殊文字を使用し、その下位のディレクトリ名やファイル名が、その特殊文字に新しいエンコーディングを使用するように更新されていない場合、サーバーは INFORMATION_SCHEMA テーブルや SHOW ステートメントの出力内でそれらの名前に #mysql50# というプリフィクスを付けて表示します。たとえば、a@b という名前のテーブルがあり、その名前エンコーディングが更新されていない場合、SHOW TABLES で次のように表示されます。

```
mysql> SHOW TABLES;
+---------------+
| Tables_in_test |
+---------------+
| #mysql50#a@b   |
+---------------+
```

エンコーディングが更新されてないこのような名前を参照するには、#mysql50# プリフィクスを付加する必要があります。

```
mysql> SHOW COLUMNS FROM `a@b`;
ERROR 1146 (42S02): Table 'test.a@b' doesn't exist

mysql> SHOW COLUMNS FROM `#mysql50#a@b`;
+-------+---------+------+-----+---------+-------+
| Field | Type    | Null | Key | Default | Extra |
+-------+---------+------+-----+---------+-------+
| i     | int(11) | YES  |     | NULL    |       |
+-------+---------+------+-----+---------+-------+
```

特殊なプリフィクスを使用する必要をなくすために旧名を更新するには、mysqlcheck で再エンコードしてください。次のコマンドは、すべての名前を新しいエンコーディングに更新します。

```
shell> mysqlcheck --check-upgrade --all-databases
shell> mysqlcheck --fix-db-names --fix-table-names --all-databases
```

特定のデータベースまたはテーブルのみを確認するには、--all-databases を削除し、適切なデータベースやテーブル引数を付け加えてください。mysqlcheck 呼び出し構文については、セクション4.5.3「mysqlcheck — テーブル保守プログラム」を参照してください。

> 注記
>
> #mysql50# プリフィクスは、サーバーが内部的に使用するためだけのものです。このプリフィクスを使用する名前でデータベースやテーブルを作成しないようにしてください。
>
> また、mysqlcheck は、特殊文字をエンコードするために使用する @ 文字のリテラルインスタンスが含まれる名前を修正できません。この文字が含まれるデータベースやテーブルを使用している場合は、mysqldump でそれらをダンプしてから MySQL 5.1.6 以降にアップグレードし、アップグレード後にダンプファイルをリロードします。

9.2.4 関数名の構文解析と解決

MySQL 5.6 は、組み込み (ネイティブ) 関数、ユーザー定義関数 (UDF)、およびストアドファンクションをサポートします。このセクションでは、組み込み関数名が関数呼び出しとして使用されているか、識別子として使用されているかをサーバーで認識する方法と、指定された名前で異なる型の関数が存在する場合に使用する関数をサーバーが特定する方法について説明します。

組み込み関数名の構文解析

パーサーは組み込み関数名を構文解析するためのデフォルトのルールを使用します。これらのルールは IGNORE_SPACE SQL モードを有効にすることで変更できます。

パーサーは、組み込み関数の名前である単語を検出すると、その名前が関数呼び出しを示しているのか、それともテーブル名やカラム名などの識別子の式以外の参照であるのかを判別する必要があります。たとえば、次のステートメントでは count に対する最初の参照は関数呼び出しですが、2 番目の参照はテーブル名です。

```
SELECT COUNT(*) FROM mytable;
CREATE TABLE count (i INT);
```

パーサーは、式であると予想される対象を解析しているときにのみ、組み込み関数の名前を、関数呼び出しを示している名前として認識する必要があります。つまり、式以外のコンテキストでは、関数名は識別子として許可されます。

ただし、組み込み関数の中には構文解析や実装に関する特別な考慮事項を含んでいるものがあるため、パーサーはデフォルトで次のルールに従って、その名前が関数呼び出しとして使用されているのか、それとも式以外のコンテキストで識別子として使用されているのかを判別します。

- 式の中で関数呼び出しとして名前を使用するには、名前とそれに続く括弧文字「(」の間に空白が存在しないことが必要です。
- 反対に、関数名を識別子として使用するには、括弧文字を直後に続けないでください。

名前と括弧の間に空白を入れずに関数呼び出しを記述するという要件は、特別な考慮事項を持つ組み込み関数にのみ適用されます。COUNT がこのような名前の 1 つです。後続の空白によって解釈が決まる関数名の正確な一覧は、sql/lex.h ソースファイルの sql_functions[] 配列に表示されます。MySQL 5.1 よりも前のバージョンでは、このような関数名が多数 (約 200) あるため、空白なしという要件をすべての関数呼び出しに適用させる方法がもっとも簡単であると考えられます。MySQL 5.1 以降では、パーサーの改善によって、影響を受ける関数名の数は約 30 にまで減少しています。

sql_functions[] 配列に一覧表示されていない関数には、空白は関係ありません。それらは式のコンテキストで使用されるときにのみ関数呼び出しとして解釈され、それ以外では識別子として自由に使用できます。ASCII がこのような名前の 1 つです。ただし、このような影響を受けない関数名に対する解釈は、式のコンテキストによって変わることがあります。つまり、func_name () は、指定の名前が単独で使用される場合は組み込み関数として解釈されますが、単独ではない場合、func_name () はユーザー定義関数またはストアドファンクション (その名前の関数が存在する場合) として解釈されます。

IGNORE_SPACE SQL モードは、空白の有無で区別される関数名をパーサーで扱う方法を変更するために使用できます。

- IGNORE_SPACE が無効になっていると、名前と後続の括弧の間に空白がない場合、パーサーはその名前を関数呼び出しと解釈します。これは、関数名が式以外のコンテキストで使用されているときにも行われます。

  ```
  mysql> CREATE TABLE count(i INT);
  ERROR 1064 (42000): You have an error in your SQL syntax ...
  near 'count(i INT)'
  ```

  エラーを取り除き、名前が識別子として扱われるようにするには、名前のあとに続く空白を使うか、引用符で囲んだ識別子として記述してください (あるいはこの両方)。

  ```
  CREATE TABLE count (i INT);
  CREATE TABLE `count`(i INT);
  CREATE TABLE `count` (i INT);
  ```

- IGNORE_SPACE が有効になっていると、パーサーは関数名と後続の括弧の間に空白は存在しないという要件を緩和します。このことで、関数呼び出しの記述がより自由に行えるようになります。たとえば、次のどちらの関数呼び出しも有効です。

  ```
  SELECT COUNT(*) FROM mytable;
  SELECT COUNT (*) FROM mytable;
  ```

  ただし、IGNORE_SPACE を有効にした場合、影響を受ける関数名をパーサーが予約語として扱うという副作用も生じます (セクション9.3「予約語」を参照してください)。つまり、名前のあとに続く空白は、それが識別子として使用されることを示すものではなくなります。後続の空白の有無を問わず、名前は関数呼び出しとして使用できますが、引用符で囲まれていないかぎり、式以外のコンテキストでは構文エラーが発生します。たとえば、IGNORE_SPACE が有効になっていると、パーサーが count を予約語として解釈するため、次のステートメントはどちらも構文エラーが表示されて失敗します。

  ```
  CREATE TABLE count(i INT);
  CREATE TABLE count (i INT);
  ```

  式以外のコンテキストで関数名を使用するには、これを引用符で囲まれた識別子として記述します。

  ```
  CREATE TABLE `count`(i INT);
  ```

```
CREATE TABLE `count` (i INT);
```

IGNORE_SPACE SQL モードを有効にするには、次のステートメントを使用します。

```
SET sql_mode = 'IGNORE_SPACE';
```

IGNORE_SPACE は、ANSI などのほかの特定のコンポジットモードの値に含められている場合にも有効になります。

```
SET sql_mode = 'ANSI';
```

どのコンポジットモードが IGNORE_SPACE を有効にするかを調べるには、セクション5.1.7「サーバー SQL モード」を参照してください。

IGNORE_SPACE 設定における SQL コードの依存関係を最小にするには、以下のガイドラインを使用してください。

- 組み込み関数と同じ名前の UDF またはストアドファンクションは作成しないでください。
- 式以外のコンテキストでは関数名を使用しないでください。たとえば、これらのステートメントは、count (IGNORE_SPACE の影響を受ける関数名の 1 つ) を使用するため、IGNORE_SPACE が有効であれば、名前に続く空白の有無によらずこれらのステートメントは失敗します。

  ```
  CREATE TABLE count(i INT);
  CREATE TABLE count (i INT);
  ```

  式以外のコンテキストで関数名を使用する必要がある場合は、これを引用符で囲まれた識別子として記述します。

  ```
  CREATE TABLE `count`(i INT);
  CREATE TABLE `count` (i INT);
  ```

MySQL 5.1.13 では、IGNORE_SPACE の影響を受ける関数名の数がおよそ 200 から 30 に大幅に減少しました。MySQL 5.1.13 以降、IGNORE_SPACE 設定の影響を引き続き受けるのは、次の関数だけです。

ADDDATE	BIT_AND	BIT_OR	BIT_XOR
CAST	COUNT	CURDATE	CURTIME
DATE_ADD	DATE_SUB	EXTRACT	GROUP_CONCAT
MAX	MID	MIN	NOW
POSITION	SESSION_USER	STD	STDDEV
STDDEV_POP	STDDEV_SAMP	SUBDATE	SUBSTR
SUBSTRING	SUM	SYSDATE	SYSTEM_USER
TRIM	VARIANCE	VAR_POP	VAR_SAMP

MySQL の以前のバージョンでは、sql/lex.h ソースファイルの sql_functions[] 配列の内容を確認して、どの関数が IGNORE_SPACE に影響を受けるかを確認してください。

非互換性に関する警告: MySQL 5.1.13 では IGNORE_SPACE の影響を受ける関数名の数を抑えることでパーサー操作に一貫性をもたらしました。ただし、次の条件に依存する旧 SQL コードの非互換性の可能性も生じます。

- IGNORE_SPACE は無効になっています。
- 関数名に続く空白の有無は、同じ名前を持つ組み込み関数とストアドファンクション (たとえば、PI() と PI () など) を区別するために使用されます。

MySQL 5.1.13 以降に IGNORE_SPACE の影響を受けなくなった関数に対しては、その方法は機能しません。前述の非互換性の影響を受けるコードがある場合は、次のどちらのアプローチも使用できます。

- ストアドファンクションに組み込み関数と競合する名前が存在する場合、空白の有無にかかわらず、スキーマ名修飾子を持つストアドファンクションを参照してください。たとえば、schema_name.PI() または schema_name.PI () と記述します。
- または、競合しない名前を使用するようにストアドファンクションの名前を変更し、新しい名前を使用するように関数の呼び出しを変更します。

関数名の解決

次のルールは、関数の作成と呼び出しのために、サーバーで関数名の参照を解決する方法について記述します。

- 組み込み関数とユーザー定義関数

  組み込み関数と同じ名前で UDF を作成しようとした場合、エラーが発生します。

- 組み込み関数とストアドファンクション

  組み込み関数と同名のストアドファンクションを作成することは可能ですが、このストアドファンクションを呼び出すにはスキーマ名で修飾する必要があります。たとえば、test スキーマ内で PI という名のストアドファンクションを作成する場合、サーバーが PI() を組み込み関数の参照として解決するため、test.PI() として呼び出します。サーバーは、ストアドファンクション名が組み込み関数名と一致しない場合、警告を作成します。この警告は SHOW WARNINGS で表示できます。

- ユーザー定義関数とストアドファンクション

  ユーザー定義関数とストアドファンクションは同じ名前空間を共有します。したがって、同名の UDF とストアドファンクションを作成することはできません。

前述の関数名の解決ルールは、新しい組み込み関数を実装する MySQL のバージョンへのアップグレードに影響を及ぼします。

- 指定の名前のユーザー定義関数がすでに作成されており、同じ名前で新しい組み込み関数を実装するバージョンに MySQL をアップグレードすると、この UDF にはアクセスできなくなります。これを修正するには、DROP FUNCTION を使用して UDF を削除してから、CREATE FUNCTION を使用して競合しない別の名前で UDF を再作成します。

- 新しいバージョンの MySQL で、既存のストアドファンクションと同じ名前で組み込み関数を実装する場合、ストアドファンクションの名前を競合しない名前に変更するか、スキーマ修飾子を使用するように関数の呼び出しを変更する (つまり、schema_name.func_name() 構文を使用する) という 2 つの選択肢があります。

9.3 予約語

SELECT、DELETE、または BIGINT などの特定の語は、テーブル名やカラム名などの識別子として使用するために予約されており、特別な取り扱いが必要になります。これは、組み込み関数の名前にも当てはまる場合があります。

セクション9.2「スキーマオブジェクト名」で説明しているように、予約語は、引用符で囲まれている場合、識別子として許可されます。

```
mysql> CREATE TABLE interval (begin INT, end INT);
ERROR 1064 (42000): You have an error in your SQL syntax ...
near 'interval (begin INT, end INT)'

mysql> CREATE TABLE `interval` (begin INT, end INT);
Query OK, 0 rows affected (0.01 sec)
```

例外: 修飾名でピリオドに続く語は識別子のため、予約されていても引用符で囲む必要はありません。

```
mysql> CREATE TABLE mydb.interval (begin INT, end INT);
Query OK, 0 rows affected (0.01 sec)
```

組み込み関数名は識別子として許可されますが、識別子として使用する場合は注意してください。たとえば、COUNT はカラム名として許容されます。ただし、デフォルトでは、関数呼び出しにおいて、関数名と後続の「(」文字の間に空白を入れないことが許可されています。この要件によって、パーサーはその名前が関数の呼び出しで使用されるのか、それとも関数でないコンテキストで使用されるのかを判別できます。関数名の識別についての詳細は、セクション9.2.4「関数名の構文解析と解決」を参照してください。

次の表の語は、MySQL 5.6 で明示的に予約されています。さらに、_FILENAME も予約されています。今後バージョンアップするときのことを考慮に入れて、使用予定のある予約語を確認しておくことをお勧めします。新しいバージョンの MySQL を扱ったマニュアルでもこれらを確認できます。表内のほとんどの語は、カラム名やテーブル名として標準 SQL で許可されていません (GROUP など)。いくつかは、MySQL が必要とし、yacc パーサーを使用するので予約されています。予約語は、引用符で囲んだ場合、識別子として使用できます。

バージョン間の違いも含めた予約語の詳細なリストについては、Keywords and Reserved Words in MySQL 5.6を参照してください。

A | B | C | D | E | F | G | H | I | J | K | L | M | N | O | P | R | S | T | U | V | W | X | Y | Z

A

- ACCESSIBLE
- ADD
- ALL
- ALTER
- ANALYZE
- AND
- AS
- ASC
- ASENSITIVE

B

- BEFORE
- BETWEEN
- BIGINT
- BINARY
- BLOB
- BOTH
- BY

C

- CALL
- CASCADE
- CASE
- CHANGE
- CHAR
- CHARACTER
- CHECK
- COLLATE
- COLUMN
- CONDITION
- CONSTRAINT
- CONTINUE
- CONVERT
- CREATE
- CROSS
- CURRENT_DATE
- CURRENT_TIME

- CURRENT_TIMESTAMP
- CURRENT_USER
- CURSOR

D

- DATABASE
- DATABASES
- DAY_HOUR
- DAY_MICROSECOND
- DAY_MINUTE
- DAY_SECOND
- DEC
- DECIMAL
- DECLARE
- DEFAULT
- DELAYED
- DELETE
- DESC
- DESCRIBE
- DETERMINISTIC
- DISTINCT
- DISTINCTROW
- DIV
- DOUBLE
- DROP
- DUAL

E

- EACH
- ELSE
- ELSEIF
- ENCLOSED
- ESCAPED
- EXISTS
- EXIT
- EXPLAIN

F

- FALSE

- FETCH
- FLOAT
- FLOAT4
- FLOAT8
- FOR
- FORCE
- FOREIGN
- FROM
- FULLTEXT

G

- GET
- GRANT
- GROUP

H

- HAVING
- HIGH_PRIORITY
- HOUR_MICROSECOND
- HOUR_MINUTE
- HOUR_SECOND

I

- IF
- IGNORE
- IN
- INDEX
- INFILE
- INNER
- INOUT
- INSENSITIVE
- INSERT
- INT
- INT1
- INT2
- INT3
- INT4
- INT8
- INTEGER

- INTERVAL
- INTO
- IO_AFTER_GTIDS
- IO_BEFORE_GTIDS
- IS
- ITERATE

J

- JOIN

K

- KEY
- KEYS
- KILL

L

- LEADING
- LEAVE
- LEFT
- LIKE
- LIMIT
- LINEAR
- LINES
- LOAD
- LOCALTIME
- LOCALTIMESTAMP
- LOCK
- LONG
- LONGBLOB
- LONGTEXT
- LOOP
- LOW_PRIORITY

M

- MASTER_BIND
- MASTER_SSL_VERIFY_SERVER_CERT
- MATCH
- MAXVALUE
- MEDIUMBLOB
- MEDIUMINT

- MEDIUMTEXT
- MIDDLEINT
- MINUTE_MICROSECOND
- MINUTE_SECOND
- MOD
- MODIFIES

N

- NATURAL
- NOT
- NO_WRITE_TO_BINLOG
- NULL
- NUMERIC

O

- ON
- OPTIMIZE
- OPTION
- OPTIONALLY
- OR
- ORDER
- OUT
- OUTER
- OUTFILE

P

- PARTITION
- PRECISION
- PRIMARY
- PROCEDURE
- PURGE

R

- RANGE
- READ
- READS
- READ_WRITE
- REAL
- REFERENCES
- REGEXP

- RELEASE
- RENAME
- REPEAT
- REPLACE
- REQUIRE
- RESIGNAL
- RESTRICT
- RETURN
- REVOKE
- RIGHT
- RLIKE

S

- SCHEMA
- SCHEMAS
- SECOND_MICROSECOND
- SELECT
- SENSITIVE
- SEPARATOR
- SET
- SHOW
- SIGNAL
- SMALLINT
- SPATIAL
- SPECIFIC
- SQL
- SQLEXCEPTION
- SQLSTATE
- SQLWARNING
- SQL_BIG_RESULT
- SQL_CALC_FOUND_ROWS
- SQL_SMALL_RESULT
- SSL
- STARTING
- STRAIGHT_JOIN

T

- TABLE

- TERMINATED
- THEN
- TINYBLOB
- TINYINT
- TINYTEXT
- TO
- TRAILING
- TRIGGER
- TRUE

U

- UNDO
- UNION
- UNIQUE
- UNLOCK
- UNSIGNED
- UPDATE
- USAGE
- USE
- USING
- UTC_DATE
- UTC_TIME
- UTC_TIMESTAMP

V

- VALUES
- VARBINARY
- VARCHAR
- VARCHARACTER
- VARYING

W

- WHEN
- WHERE
- WHILE
- WITH
- WRITE

X

- XOR

Y

- YEAR_MONTH

Z

- ZEROFILL
- GET
- IO_AFTER_GTIDS
- IO_BEFORE_GTIDS
- MASTER_BIND
- PARTITION

MySQL では、以前に多くの人々が使用していたので、一部のキーワードを引用符で囲まない識別子として使用することを許可しています。次の一覧に例を示します。

- ACTION
- BIT
- DATE
- ENUM
- NO
- TEXT
- TIME
- TIMESTAMP

9.4 ユーザー定義変数

あるステートメント内のユーザー定義変数に値を格納し、あとから別のステートメントでこれを参照できます。これにより、あるステートメントから別のステートメントに値を渡すことができます。ユーザー定義変数はセッション固有です。つまり、あるクライアントが定義したユーザー変数を、ほかのクライアントが表示したり、使用したりすることはできません。所定のクライアントセッションのすべての変数は、クライアントが終了すると自動的に解放されます。

ユーザー変数は @var_name と記述されます。ここで変数名 var_name は、英数字文字、「.」、「_」、および「$」から構成されます。ユーザー変数名を文字列や識別子として引用符で囲めば、ほかの文字も含めることができます (@'my-var'、@"my-var"、@`my-var` など)。

ユーザー変数名では大文字と小文字を区別しません。

ユーザー定義変数を設定する方法の 1 つに、SET ステートメントを発行する方法が挙げられます。

```
SET @var_name = expr [, @var_name = expr] ...
```

SET では、= または := のどちらかを割り当て演算子として使用できます。

SET 以外のステートメントで、ユーザー変数に値を割り当てることもできます。この場合、= は SET 以外のステートメントでは比較演算子 = として扱われるので、割り当て演算子にはこちらではなく、:= を使用する必要があります。

```
mysql> SET @t1=1, @t2=2, @t3:=4;
mysql> SELECT @t1, @t2, @t3, @t4 := @t1+@t2+@t3;
+------+------+------+-------------------+
| @t1  | @t2  | @t3  | @t4 := @t1+@t2+@t3 |
+------+------+------+-------------------+
|    1 |    2 |    4 |                 7 |
+------+------+------+-------------------+
```

ユーザー変数には、限定された一連のデータ型の値 (整数、小数、浮動小数点、バイナリ文字列、非バイナリ文字列、または NULL 値) を割り当てることができます。10 進値と実数値の割り当てでは、値の精度やスケールは維持されません。許可されている型以外の型の値は、許可されている型に変換されます。たとえば、時間を表すデータ型や空間データ型の値は、バイナリ文字列に変換されます。

ユーザー変数に非バイナリ (文字) 文字列値を割り当てた場合、その変数には文字列と同じ文字セットと照合順序が含まれます。ユーザー変数の強制性は暗黙的です。(これはテーブルカラム値と同等の強制性です。)

ユーザー変数に割り当てられたビット値は、バイナリ文字列として扱われます。ビット値を数値としてユーザー変数に割り当てるには、CAST() または +0 を使用します。

```
mysql> SET @v1 = b'1000001';
mysql> SET @v2 = CAST(b'1000001' AS UNSIGNED), @v3 = b'1000001'+0;
mysql> SELECT @v1, @v2, @v3;
+------+------+------+
| @v1  | @v2  | @v3  |
+------+------+------+
| A    |   65 |   65 |
+------+------+------+
```

結果セットでユーザー変数の値が選択された場合、それは文字列としてクライアントに返されます。

初期化されていない変数を参照する場合、その値は NULL で、型は文字列です。

ユーザー変数は、式が許可されているほとんどのコンテキストで使用できます。これには現在、SELECT ステートメントの LIMIT 句の中や、LOAD DATA ステートメントの IGNORE N LINES 句の中など、リテラル値を明示的に要求するコンテキストは含まれません。

一般的なルールとして、SET ステートメント以外では、同じステートメント内で、ユーザー変数に値を割り当ててその値を読み取ることは決してしないでください。たとえば、変数を増分する場合、次のようにしても問題ありません。

```
SET @a = @a + 1;
```

SELECT などのほかのステートメントでは、予想した結果が得られることもありますが、これは保証されません。次のステートメントでは、MySQL が最初に @a を評価し、続いて 2 番目の割り当てを行います。

```
SELECT @a, @a:=@a+1, ...;
```

ただし、ユーザー変数を含む式の評価の順序は、定義されていません。

SET 以外の同じステートメント内で変数に値を割り当てて、その値を読み取る場合に生じるもう 1 つの問題は、変数のデフォルトの結果型がステートメントの開始時の型に基づくということです。次の例でこれについて説明します。

```
mysql> SET @a='test';
mysql> SELECT @a,(@a:=20) FROM tbl_name;
```

この SELECT ステートメントの場合、MySQL は、カラム 1 が文字列であり、@a が 2 行目の数値に設定されていても、@a のすべてのアクセスを文字列に変換すると、クライアントにレポートします。SELECT ステートメントの実行後、@a は次のステートメントの数値と見なされます。

この動作による問題を回避するには、単一のステートメント内で同じ変数に値を割り当ててその値を読み取ることを行わないか、使用する前に変数を 0、0.0、または '' に設定して、その型を定義してください。

SELECT ステートメントでは、それぞれの選択式は、クライアントに送信されるときにのみ評価されます。つまり、HAVING、GROUP BY、または ORDER BY 句では、選択式リストで値を割り当てられた変数を参照しても、予想どおりには機能しないということです。

```
mysql> SELECT (@aa:=id) AS a, (@aa+3) AS b FROM tbl_name HAVING b=5;
```

HAVING 句での b の参照は、@aa を使用する選択リスト内の式のエイリアスを参照します。これは予想どおりには機能しません。@aa には、現在の行ではなく、以前に選択した行の id の値が含まれます。

ユーザー変数は、データ値を提供するためのものです。これらは、テーブル名やデータベース名が想定されるコンテキストなどでの識別子または識別子の一部として、または SELECT などの予約語として、SQL ステートメントの中で直接使用することはできません。これは、次の例に示すように、変数が引用符で囲まれている場合でも同じです。

```
mysql> SELECT c1 FROM t;
+----+
| c1 |
+----+
|  0 |
+----+
|  1 |
```

```
+----+
2 rows in set (0.00 sec)

mysql> SET @col = "c1";
Query OK, 0 rows affected (0.00 sec)

mysql> SELECT @col FROM t;
+------+
| @col |
+------+
| c1   |
+------+
1 row in set (0.00 sec)

mysql> SELECT `@col` FROM t;
ERROR 1054 (42S22): Unknown column '@col' in 'field list'

mysql> SET @col = "`c1`";
Query OK, 0 rows affected (0.00 sec)

mysql> SELECT @col FROM t;
+------+
| @col |
+------+
| `c1` |
+------+
1 row in set (0.00 sec)
```

識別子を提供するためにユーザー変数を使用できないというこの原則の例外が、あとから実行する準備されたステートメントとして使用するために文字列を構築している場合です。この場合、ユーザー変数はステートメントの一部を提供するために使用できます。次の例は、その方法を示しています。

```
mysql> SET @c = "c1";
Query OK, 0 rows affected (0.00 sec)

mysql> SET @s = CONCAT("SELECT ", @c, " FROM t");
Query OK, 0 rows affected (0.00 sec)

mysql> PREPARE stmt FROM @s;
Query OK, 0 rows affected (0.04 sec)
Statement prepared

mysql> EXECUTE stmt;
+----+
| c1 |
+----+
|  0 |
+----+
|  1 |
+----+
2 rows in set (0.00 sec)

mysql> DEALLOCATE PREPARE stmt;
Query OK, 0 rows affected (0.00 sec)
```

詳細は、セクション13.5「準備済みステートメントのための SQL 構文」を参照してください。

PHP 5 を使用して次に示すように、同じような手法をアプリケーションプログラムの中で使用することによって、プログラム変数を使用する SQL ステートメントを作成できます。

```
<?php
  $mysqli = new mysqli("localhost", "user", "pass", "test");

  if( mysqli_connect_errno() )
    die("Connection failed: %s\n", mysqli_connect_error());

  $col = "c1";

  $query = "SELECT $col FROM t";

  $result = $mysqli->query($query);

  while($row = $result->fetch_assoc())
  {
    echo "<p>" . $row["$col"] . "</p>\n";
  }

  $result->close();
```

```
  $mysqli->close();
?>
```

この方法で SQL ステートメントを作成することを「ダイナミック SQL」と呼ぶことがあります。

9.5 式の構文

次のルールによって、MySQL での式の構文が定義されます。ここで示す文法は、MySQL ソース配布の sql/sql_yacc.yy ファイルで与えられた文法に基づいています。一部の条件に関する追加情報については、文法のあとに表示される注意事項を参照してください。

```
expr:
    expr OR expr
  | expr || expr
  | expr XOR expr
  | expr AND expr
  | expr && expr
  | NOT expr
  | ! expr
  | boolean_primary IS [NOT] {TRUE | FALSE | UNKNOWN}
  | boolean_primary

boolean_primary:
    boolean_primary IS [NOT] NULL
  | boolean_primary <=> predicate
  | boolean_primary comparison_operator predicate
  | boolean_primary comparison_operator {ALL | ANY} (subquery)
  | predicate

comparison_operator: = | >= | > | <= | < | <> | !=

predicate:
    bit_expr [NOT] IN (subquery)
  | bit_expr [NOT] IN (expr [, expr] ...)
  | bit_expr [NOT] BETWEEN bit_expr AND predicate
  | bit_expr SOUNDS LIKE bit_expr
  | bit_expr [NOT] LIKE simple_expr [ESCAPE simple_expr]
  | bit_expr [NOT] REGEXP bit_expr
  | bit_expr

bit_expr:
    bit_expr | bit_expr
  | bit_expr & bit_expr
  | bit_expr << bit_expr
  | bit_expr >> bit_expr
  | bit_expr + bit_expr
  | bit_expr - bit_expr
  | bit_expr * bit_expr
  | bit_expr / bit_expr
  | bit_expr DIV bit_expr
  | bit_expr MOD bit_expr
  | bit_expr % bit_expr
  | bit_expr ^ bit_expr
  | bit_expr + interval_expr
  | bit_expr - interval_expr
  | simple_expr

simple_expr:
    literal
  | identifier
  | function_call
  | simple_expr COLLATE collation_name
  | param_marker
  | variable
  | simple_expr || simple_expr
  | + simple_expr
  | - simple_expr
  | ~ simple_expr
  | ! simple_expr
  | BINARY simple_expr
  | (expr [, expr] ...)
  | ROW (expr, expr [, expr] ...)
  | (subquery)
  | EXISTS (subquery)
  | {identifier expr}
  | match_expr
  | case_expr
  | interval_expr
```

注:

演算子の優先順位については、セクション12.3.1「演算子の優先順位」を参照してください。

リテラル値の構文については、セクション9.1「リテラル値」を参照してください。

識別子の構文については、セクション9.2「スキーマオブジェクト名」を参照してください。

変数には、ユーザー変数、システム変数、ストアドプログラムのローカル変数またはパラメータがあります。

- ユーザー変数: セクション9.4「ユーザー定義変数」
- システム変数: セクション5.1.5「システム変数の使用」
- ローカル変数: セクション13.6.4.1「ローカル変数の DECLARE 構文」
- パラメータ: セクション13.1.15「CREATE PROCEDURE および CREATE FUNCTION 構文」

param_marker は、準備されたステートメントでプレースホルダーに使用されているように ? です。セクション13.5.1「PREPARE 構文」を参照してください。

(subquery) は、単一の値を返すサブクエリー、つまりスカラーサブクエリーを示します。セクション13.2.10.1「スカラーオペランドとしてのサブクエリー」を参照してください。

{identifier expr} は、ODBC エスケープ構文であり、ODBC との互換性のために認められています。値は expr です。構文内のカールした中括弧は文字どおりに書き込まれる必要があります。それらは構文説明の別の部分で利用されているようなメタ構文ではありません。

match_expr は MATCH 式を示します。セクション12.9「全文検索関数」を参照してください。

case_expr は CASE 式を示します。セクション12.4「制御フロー関数」を参照してください。

interval_expr は時間間隔を表します。構文は INTERVAL expr unit です。ここで unit は HOUR、DAY、WEEK などの指定子です。unit 指定子の完全なリストについては、セクション12.7「日付および時間関数」の DATE_ADD() 関数の説明を参照してください。

一部の演算子の意味は、SQL モードによって異なります。

- デフォルトでは、|| は論理 OR 演算子です。PIPES_AS_CONCAT が有効になっている場合は、|| は ^ と単項演算子間の優先順位を持つ文字列連結です。
- デフォルトでは、! は NOT よりも高い優先順位です。HIGH_NOT_PRECEDENCE が有効になっている場合は、! と NOT の優先順位は同じです。

セクション5.1.7「サーバー SQL モード」を参照してください。

9.6 コメントの構文

MySQL Server では、次の 3 つのコメントスタイルをサポートしています。

- 「#」文字から行末まで。
- 「-- 」シーケンスから行末まで。MySQL では、「-- 」(二重ダッシュ) のコメントスタイルは、2 番目のダッシュに少なくとも 1 つの空白または制御文字 (空白、タブ、改行など) を続ける必要があります。セクション1.7.2.5「コメントの先頭としての「--」」で述べているように、この構文は標準 SQL のコメントの構文とは少し異なります。
- C プログラミング言語のように、/* シーケンスから次の */ シーケンスまで。この構文では、開始と終了のシーケンスは同じ行にある必要はないので、複数の行にわたってコメントを記すことができます。

次の例には、3 つのコメントスタイルがすべて示されています。

```
mysql> SELECT 1+1;     # This comment continues to the end of line
mysql> SELECT 1+1;     -- This comment continues to the end of line
mysql> SELECT 1 /* this is an in-line comment */ + 1;
mysql> SELECT 1+
/*
this is a
multiple-line comment
*/
```

```
1;
```

ネストされたコメントはサポートされていません。(一部の条件下では、ネストされたコメントが可能な場合がありますが、通常は可能ではなく、ユーザーはネストされたコメントを使用しないでください。)

MySQL Server では C-スタイルコメントのバリアントがいくつかサポートされています。これらでは、次の形式のコメントを使用することにより、MySQL 拡張を含んでいるが、移植性を維持しているコードを記述できます。

```
/*! MySQL-specific code */
```

この場合、MySQL Server は、ほかの SQL ステートメントのようにコメント内のコードを構文解析して実行しますが、ほかの SQL サーバーはその拡張機能を認識しません。たとえば、MySQL Server は次のステートメント内の STRAIGHT_JOIN キーワードを認識しますが、ほかのサーバーは認識しません。

```
SELECT /*! STRAIGHT_JOIN */ col1 FROM table1,table2 WHERE ...
```

「!」文字のあとにバージョン番号を追加すると、コメント内の構文は、MySQL のバージョンが指定されたバージョン番号以上の場合にだけ実行されます。次のコメント内の TEMPORARY キーワードは MySQL 3.23.02 以降のサーバーでのみ実行されます。

```
CREATE /*!32302 TEMPORARY */ TABLE t (a INT);
```

前述のコメントの構文は、mysqld サーバーによる SQL ステートメントの構文解析に適用されます。mysql クライアントプログラムは、ステートメントをサーバーに送信する前に、その一部の構文解析も実行します。(これは複数ステートメント入力行で、ステートメント境界を決定するために行われます。)

この形式のコメント (/*!12345 ... */) はサーバーに格納されません。ストアドルーチンのコメントにこの形式が使用されている場合、このコメントはサーバーで保持されません。

複数行にわたる /* ... */ コメント内で、\C などの短い形式の mysql コマンドを使用することはサポートされていません。

第 10 章 グローバリゼーション

目次

10.1 文字セットのサポート 945
10.1.1 一般の文字セットおよび照合順序 946
10.1.2 MySQL での文字セットと照合順序 946
10.1.3 文字セットと照合順序の指定 948
10.1.4 接続文字セットおよび照合順序 954
10.1.5 アプリケーションの文字セットおよび照合順序の構成 956
10.1.6 エラーメッセージの文字セット 957
10.1.7 照合順序の問題 958
10.1.8 文字列のレパートリー 965
10.1.9 文字セットのサポートによる影響を受ける演算 967
10.1.10 Unicode のサポート 969
10.1.11 以前の Unicode サポートから現在の Unicode サポートへのアップグレード 973
10.1.12 メタデータ用の UTF-8 975
10.1.13 カラム文字セットの変換 976
10.1.14 MySQL でサポートされる文字セットと照合順序 977
10.2 エラーメッセージ言語の設定 988
10.3 文字セットの追加 989
10.3.1 文字定義配列 990
10.3.2 複雑な文字セットの文字列照合のサポート 991
10.3.3 複雑な文字セットのマルチバイト文字のサポート 991
10.4 文字セットへの照合順序の追加 992
10.4.1 照合順序の実装タイプ 993
10.4.2 照合順序 ID の選択 995
10.4.3 8 ビットの文字セットへの単純な照合順序の追加 996
10.4.4 Unicode 文字セットへの UCA 照合順序の追加 997
10.5 文字セットの構成 1002
10.6 MySQL Server でのタイムゾーンのサポート 1003
10.6.1 タイムゾーンの変更による現在の時間の維持 1005
10.6.2 タイムゾーンのうるう秒のサポート 1006
10.7 MySQL Server の口ケールサポート 1007

この章では、国際化 (各地域での使用に対応するための MySQL の機能) と口ーカライズ (特定の地域的な規則の選択) を含む、次のようなグローバリゼーションの問題を取り上げます。

- SQL ステートメント内の文字セットに対する MySQL でのサポート。
- さまざまな文字セットをサポートするようにサーバーを構成する方法。
- エラーメッセージの言語の選択。
- サーバーのタイムゾーンを設定し、接続ごとのタイムゾーンをサポートできるようにする方法。
- 曜日と月の名前に使用する口ケールの選択。

10.1 文字セットのサポート

MySQL では、さまざまな文字セットを使用してデータを格納し、さまざまな照合順序に従って比較を実行できます。サーバー、データベース、テーブル、およびカラムレベルで文字セットを指定できます。MySQL では、MyISAM、MEMORY、および InnoDB のストレージエンジンの文字セットの使用をサポートします。

この章では次のトピックについて説明します。

- 文字セットと照合順序とは。
- 文字セットの割り当てに対する複数レベルのデフォルトシステム。
- 文字セットと照合順序を指定するための構文。

- 影響を受ける関数と演算。
- Unicode のサポート。
- 使用可能な文字セットと照合順序 (ノート付き)。

文字セットの問題は、データストレージだけではなく、クライアントプログラムと MySQL Server との通信にも影響を与えます。デフォルトと異なる文字セットを使用してクライアントプログラムとサーバー間の通信を行う場合、どの文字セットを使用するのかを示す必要があります。たとえば、utf8 Unicode 文字セットを使用するには、サーバー接続後に次のステートメントを発行してください。

```
SET NAMES 'utf8';
```

アプリケーションで使用する文字セットの構成と、クライアント/サーバー間の通信に関する文字セット関連の問題の詳細は、セクション10.1.5「アプリケーションの文字セットおよび照合順序の構成」およびセクション10.1.4「接続文字セットおよび照合順序」を参照してください。

10.1.1 一般の文字セットおよび照合順序

文字セットとは、記号とエンコーディングのセットです。照合順序とは、文字セット内の文字を比較するためのルールを集めたものです。架空の文字セットを例にして、文字セットと照合順序の違いを見てみましょう。

「A」、「B」、「a」、「b」の 4 つのアルファベットがあるとします。それぞれの文字に「A」= 0、「B」= 1、「a」= 2、「b」= 3 の数字を割り当てます。文字「A」は記号で、数字 0 は「A」に対するエンコーディングであり、4 つの文字とそのエンコーディングの組み合わせが文字セットになります。

「A」と「B」の 2 つの文字列値を比較するとします。これを行うには、「A」の 0、「B」の 1 というエンコーディングを調べる方法がもっとも簡単です。0 は 1 より小さいので、「A」は「B」より小さいと見なします。今ここで行なったのは、文字セットに対する照合順序の適用です。照合順序はルールの集まりであり、この場合、ルールは「エンコーディングの比較」の 1 つだけになります。これは可能な照合順序のうちでもっとも単純なものであり、バイナリ照合順序と呼ばれています。

しかし、小文字と大文字が同等であることを表すにはどうなるのでしょうか。この場合、少なくとも次の 2 つのルールが必要です。(1) 小文字「a」および「b」を「A」および「B」と同等のものとして扱います。(2) 続いてそのエンコーディングを比較します。これは大文字と小文字を区別しない照合順序と呼ばれます。バイナリ照合順序よりも少し複雑になります。

実際には、ほとんどの文字セットには多数の文字があり、「A」と「B」だけではなくアルファベット全体が含まれます。複数のアルファベットが含まれる場合も、数千種類の文字や多くの特殊記号および句読点から構成される東洋書記体系の場合もあります。また実際には、ほとんどの照合順序は、大文字と小文字の区別だけでなく、アクセント符号 (「アクセント符号」とはドイツ語の「Ö」に見られるような文字に付けられた符号です) を区別するかどうかに関するルールや、複数文字のマッピングのルール (2 つのドイツ語の照合順序の一方における「Ö」=「OE」というルールなど) など多くのルールを含んでいます。

MySQL では以下が可能です。

- さまざまな文字セットを使用して文字列を格納します。
- さまざまな照合順序を使用して文字列を比較します。
- 文字セットまたは照合順序が異なる文字列を、同じサーバー、同じデータベース、または同じテーブル内にも混在させます。
- どのレベルでも文字セットと照合順序を指定できるようにします。

MySQL はこれらの点において、ほかのほとんどのデータベース管理システムに大きく差をつけています。ただし、これらの機能を効果的に使用するには、利用可能な文字セットと照合順序、各デフォルトの変更方法、および文字列演算子および関数の動作にこれらの機能が与える影響を知っておく必要があります。

10.1.2 MySQL での文字セットと照合順序

MySQL Server では複数の文字セットがサポートされています。利用可能な文字セットを一覧表示するには、SHOW CHARACTER SET ステートメントを使用します。リストの一部は次のとおりです。完全な情報については、セクション10.1.14「MySQL でサポートされる文字セットと照合順序」を参照してください。

```
mysql> SHOW CHARACTER SET;
+----------+-----------------------------+---------------------+--------+
```

```
| Charset  | Description                | Default collation   | Maxlen |
+----------+----------------------------+---------------------+--------+
| big5     | Big5 Traditional Chinese   | big5_chinese_ci     |      2 |
| dec8     | DEC West European          | dec8_swedish_ci     |      1 |
| cp850    | DOS West European          | cp850_general_ci    |      1 |
| hp8      | HP West European           | hp8_english_ci      |      1 |
| koi8r    | KOI8-R Relcom Russian      | koi8r_general_ci    |      1 |
| latin1   | cp1252 West European       | latin1_swedish_ci   |      1 |
| latin2   | ISO 8859-2 Central European | latin2_general_ci   |      1 |
| swe7     | 7bit Swedish               | swe7_swedish_ci     |      1 |
| ascii    | US ASCII                   | ascii_general_ci    |      1 |
| ujis     | EUC-JP Japanese            | ujis_japanese_ci    |      3 |
| sjis     | Shift-JIS Japanese         | sjis_japanese_ci    |      2 |
| hebrew   | ISO 8859-8 Hebrew          | hebrew_general_ci   |      1 |
| tis620   | TIS620 Thai                | tis620_thai_ci      |      1 |
| euckr    | EUC-KR Korean              | euckr_korean_ci     |      2 |
| koi8u    | KOI8-U Ukrainian           | koi8u_general_ci    |      1 |
| gb2312   | GB2312 Simplified Chinese  | gb2312_chinese_ci   |      2 |
| greek    | ISO 8859-7 Greek           | greek_general_ci    |      1 |
| cp1250   | Windows Central European   | cp1250_general_ci   |      1 |
| gbk      | GBK Simplified Chinese     | gbk_chinese_ci      |      2 |
| latin5   | ISO 8859-9 Turkish         | latin5_turkish_ci   |      1 |
...
```

どの文字セットにも常に少なくとも 1 つの照合順序が対応しています。複数の照合順序が対応することもあります。文字セットの照合順序を一覧表示するには、SHOW COLLATION ステートメントを使用します。たとえば、latin1 (cp1252 西ヨーロッパ言語) 文字セットの照合順序を表示するには、このステートメントを使用して、名前が latin1 で始まる照合順序を探します。

```
mysql> SHOW COLLATION LIKE 'latin1%';
+-------------------+---------+----+---------+----------+---------+
| Collation         | Charset | Id | Default | Compiled | Sortlen |
+-------------------+---------+----+---------+----------+---------+
| latin1_german1_ci | latin1  |  5 |         |          |       0 |
| latin1_swedish_ci | latin1  |  8 | Yes     | Yes      |       1 |
| latin1_danish_ci  | latin1  | 15 |         |          |       0 |
| latin1_german2_ci | latin1  | 31 |         | Yes      |       2 |
| latin1_bin        | latin1  | 47 |         | Yes      |       1 |
| latin1_general_ci | latin1  | 48 |         |          |       0 |
| latin1_general_cs | latin1  | 49 |         |          |       0 |
| latin1_spanish_ci | latin1  | 94 |         |          |       0 |
+-------------------+---------+----+---------+----------+---------+
```

latin1 の照合順序の意味は次のとおりです。

照合順序	意味
latin1_german1_ci	ドイツ語 DIN-1
latin1_swedish_ci	スウェーデン語/フィンランド語
latin1_danish_ci	デンマーク語/ノルウェー語
latin1_german2_ci	ドイツ語 DIN-2
latin1_bin	latin1 エンコーディングに基づくバイナリ
latin1_general_ci	マルチリンガル (西ヨーロッパ言語)
latin1_general_cs	マルチリンガル (ISO 西ヨーロッパ言語)、大文字と小文字を区別
latin1_spanish_ci	現代スペイン語

照合順序には次のような一般的な特徴があります。

- 2 つの異なる文字セットで、同じ照合順序を共有できません。
- 各文字セットには、デフォルト照合順序が 1 つ存在します。たとえば、latin1 のデフォルト照合順序は latin1_swedish_ci です。SHOW CHARACTER SET の出力には、表示されている各文字セットのデフォルト照合順序が示されます。
- 照合順序名には、関連する文字セットの名前で始まる、通常は言語名を含む、_ci (大文字と小文字を区別しない)、_cs (大文字と小文字を区別する)、_bin (バイナリ) のいずれかで終わる、という規則が適用されます。

文字セットに複数の照合順序が存在する場合、どの照合順序が所定のアプリケーションにもっとも適しているかが明確でないことがあります。正しくない照合順序を選択しないようにするには、代表的なデータ値で比較を行い、特定の照合順序で期待どおりに値がソートされることを確認すると役立ちます。

10.1.3 文字セットと照合順序の指定

サーバー、データベース、テーブル、カラムの 4 つのレベルで、文字セットと照合順序のデフォルト設定が用意されています。以降のセクションの説明は複雑に思われますが、実際、複数レベルのデフォルト設定によって自然で明白な結果が得られることがわかっています。

CHARACTER SET は文字セットを指定する句で使用します。CHARSET は、CHARACTER SET のシノニムとして使用できます。

文字セットの問題は、データストレージだけではなく、クライアントプログラムと MySQL Server との通信にも影響を与えます。デフォルトと異なる文字セットを使用してクライアントプログラムとサーバー間の通信を行う場合、どの文字セットを使用するのかを示す必要があります。たとえば、utf8 Unicode 文字セットを使用するには、サーバー接続後に次のステートメントを発行してください。

```
SET NAMES 'utf8';
```

クライアント/サーバー間の通信に関する文字セット関連の問題の詳細は、セクション10.1.4「接続文字セットおよび照合順序」を参照してください。

10.1.3.1 サーバー文字セットおよび照合順序

MySQL Server にはサーバー文字セットとサーバー照合順序があります。これらは、コマンド行またはオプションファイルで、サーバーの起動時に設定し、実行時に変更できます。

サーバー文字セットおよび照合順序は最初、mysqld の起動時に使用するオプションに依存します。文字セットに --character-set-server を使用できます。これに加え、照合順序に --collation-server を追加できます。文字セットを指定しない場合は、--character-set-server=latin1 を指定した場合と同じになります。文字セット (たとえば latin1) だけを指定して照合順序を指定しない場合は、--character-set-server=latin1 --collation-server=latin1_swedish_ci を指定した場合と同じになります。これは latin1_swedish_ci が latin1 のデフォルト照合順序であるためです。したがって、次の 3 つのコマンドを実行した結果はいずれも同じになります。

```
shell> mysqld
shell> mysqld --character-set-server=latin1
shell> mysqld --character-set-server=latin1 \
         --collation-server=latin1_swedish_ci
```

設定を変更する手段の 1 つは再コンパイルです。ソースから構築するときに、デフォルトのサーバー文字セットおよび照合順序を変更するには、CMake の DEFAULT_CHARSET および DEFAULT_COLLATION オプションを使用します。例:

```
shell> cmake . -DDEFAULT_CHARSET=latin1
```

または:

```
shell> cmake . -DDEFAULT_CHARSET=latin1 \
         -DDEFAULT_COLLATION=latin1_german1_ci
```

mysqld と CMake の両方は、文字セットと照合順序の組み合わせが有効であることを検証します。組み合わせが有効でない場合、各プログラムによってエラーメッセージが表示され、強制終了されます。

サーバー文字セットおよび照合順序は、データベース文字セットおよび照合順序が CREATE DATABASE ステートメントで指定されていない場合にデフォルト値として使用されます。これらにほかの用途はありません。

現在のサーバー文字セットおよび照合順序は、character_set_server および collation_server システム変数の値で判別できます。これらの変数は実行時に変更できます。

10.1.3.2 データベース文字セットおよび照合順序

各データベースにはデータベース文字セットとデータベース照合順序があります。CREATE DATABASE および ALTER DATABASE ステートメントには、データベース文字セットおよび照合順序を指定するためのオプション句があります。

```
CREATE DATABASE db_name
    [[DEFAULT] CHARACTER SET charset_name]
    [[DEFAULT] COLLATE collation_name]

ALTER DATABASE db_name
    [[DEFAULT] CHARACTER SET charset_name]
```

```
  [[DEFAULT] COLLATE collation_name]
```

SCHEMA というキーワードは DATABASE の代わりに使用できます。

すべてのデータベースオプションは、データベースディレクトリ内の db.opt というテキストファイルに格納されます。

CHARACTER SET および COLLATE 句を使用すると、文字セットと照合順序が異なる複数のデータベースを同一の MySQL Server に作成できます。

例:

```
CREATE DATABASE db_name CHARACTER SET latin1 COLLATE latin1_swedish_ci;
```

MySQL では、データベース文字セットとデータベース照合順序が次のように選択されます。

- CHARACTER SET X と COLLATE Y の両方が指定されている場合、文字セット X と照合順序 Y が使用されます。
- CHARACTER SET X は指定されているが COLLATE は指定されていない場合、文字セット X とそのデフォルト照合順序が使用されます。各文字セットのデフォルトの照合順序を確認するには、SHOW COLLATION ステートメントを使用します。
- COLLATE Y は指定されているが CHARACTER SET は指定されていない場合、Y に関連付けられた文字セットと照合順序 Y が使用されます。
- これ以外の場合は、サーバー文字セットとサーバー照合順序が使用されます。

デフォルトのデータベースに対する文字セットと照合順序は、character_set_database および collation_database システム変数の値から判別できます。デフォルトのデータベースが変わるたびに、サーバーはこれらの変数を設定します。デフォルトのデータベースがない場合、変数は、character_set_server および collation_server という対応するサーバーレベルのシステム変数と同値になります。

特定のデータベースのデフォルト文字セットおよび照合順序を確認するには、次のステートメントを使用します。

```
USE db_name;
SELECT @@character_set_database, @@collation_database;
```

また、デフォルトのデータベースを変更しないで値を表示するには、次のステートメントを使用します。

```
SELECT DEFAULT_CHARACTER_SET_NAME, DEFAULT_COLLATION_NAME
FROM INFORMATION_SCHEMA.SCHEMATA WHERE SCHEMA_NAME = 'db_name';
```

データベース文字セットおよび照合順序は、サーバー操作の次の側面に影響します。

- CREATE TABLE ステートメントでは、テーブル文字セットおよび照合順序が指定されていない場合、データベース文字セットおよび照合順序がテーブル定義のデフォルト値として使用されます。これをオーバーライドするには、CHARACTER SET および COLLATE テーブルオプションを明示的に指定します。
- CHARACTER SET 句を含まない LOAD DATA ステートメントでは、サーバーは、character_set_database システム変数によって示された文字セットを使用して、ファイル内の情報を解釈します。これをオーバーライドするには、CHARACTER SET 句を明示的に指定します。
- ストアドルーチン (ストアドプロシージャーおよびストアドファンクション) では、CHARACTER SET 属性も COLLATE 属性も宣言に含まれていない文字データパラメータの文字セットおよび照合順序として、ルーチンの作成時に有効なデータベース文字セットおよび照合順序が使用されます。これをオーバーライドするには、CHARACTER SET および COLLATE 属性を明示的に指定します。

10.1.3.3 テーブル文字セットおよび照合順序

各テーブルにはテーブル文字セットとテーブル照合順序があります。CREATE TABLE および ALTER TABLE ステートメントには、テーブル文字セットおよび照合順序を指定するためのオプション句があります。

```
CREATE TABLE tbl_name (column_list)
  [[DEFAULT] CHARACTER SET charset_name]
  [COLLATE collation_name]]

ALTER TABLE tbl_name
  [[DEFAULT] CHARACTER SET charset_name]
```

```
  [COLLATE collation_name]
```

例:

```
CREATE TABLE t1 ( ... )
CHARACTER SET latin1 COLLATE latin1_danish_ci;
```

MySQL では、テーブル文字セットとテーブル照合順序が次のように選択されます。

- CHARACTER SET X と COLLATE Y の両方が指定されている場合、文字セット X と照合順序 Y が使用されます。
- CHARACTER SET X は指定されているが COLLATE は指定されていない場合、文字セット X とそのデフォルト照合順序が使用されます。各文字セットのデフォルトの照合順序を確認するには、SHOW COLLATION ステートメントを使用します。
- COLLATE Y は指定されているが CHARACTER SET は指定されていない場合、Y に関連付けられた文字セットと照合順序 Y が使用されます。
- これ以外の場合は、データベース文字セットとデータベース照合順序が使用されます。

個々のカラム定義でカラム文字セットおよび照合順序が指定されていない場合は、テーブル文字セットおよび照合順序がカラム定義のデフォルト値として使用されます。テーブル文字セットおよび照合順序は MySQL の拡張機能です。このような機能は標準 SQL には存在しません。

10.1.3.4 カラム文字セットおよび照合順序

各「文字」カラム (CHAR、VARCHAR、または TEXT 型のカラム) にはカラム文字セットとカラム照合順序があります。CREATE TABLE および ALTER TABLE のカラム定義構文には、カラム文字セットおよび照合順序を指定するためのオプション句があります。

```
col_name {CHAR | VARCHAR | TEXT} (col_length)
  [CHARACTER SET charset_name]
  [COLLATE collation_name]
```

これらの句は、ENUM および SET カラムにも使用できます。

```
col_name {ENUM | SET} (val_list)
  [CHARACTER SET charset_name]
  [COLLATE collation_name]
```

例:

```
CREATE TABLE t1
(
  col1 VARCHAR(5)
    CHARACTER SET latin1
    COLLATE latin1_german1_ci
);

ALTER TABLE t1 MODIFY
  col1 VARCHAR(5)
    CHARACTER SET latin1
    COLLATE latin1_swedish_ci;
```

MySQL では、カラム文字セットとカラム照合順序が次のように選択されます。

- CHARACTER SET X と COLLATE Y の両方が指定されている場合、文字セット X と照合順序 Y が使用されます。

  ```
  CREATE TABLE t1
  (
    col1 CHAR(10) CHARACTER SET utf8 COLLATE utf8_unicode_ci
  ) CHARACTER SET latin1 COLLATE latin1_bin;
  ```

  カラムに対して文字セットと照合順序が指定されているので、これらが使用されます。このカラムには、文字セット utf8 と照合順序 utf8_unicode_ci があります。

- CHARACTER SET X は指定されているが COLLATE は指定されていない場合、文字セット X とそのデフォルト照合順序が使用されます。

  ```
  CREATE TABLE t1
  (
  ```

```
   col1 CHAR(10) CHARACTER SET utf8
) CHARACTER SET latin1 COLLATE latin1_bin;
```

カラムに対して文字セットは指定されていますが、照合順序は指定されていません。このカラムには、文字セット utf8 と、utf8 のデフォルトの照合順序である utf8_general_ci があります。各文字セットのデフォルトの照合順序を確認するには、SHOW COLLATION ステートメントを使用します。

- COLLATE Y は指定されているが CHARACTER SET は指定されていない場合、Y に関連付けられた文字セットと照合順序 Y が使用されます。

```
CREATE TABLE t1
(
   col1 CHAR(10) COLLATE utf8_polish_ci
) CHARACTER SET latin1 COLLATE latin1_bin;
```

カラムに対して照合順序は指定されていますが、文字セットは指定されていません。このカラムには照合順序 utf8_polish_ci があり、この照合順序に関連付けられた文字セットである utf8 があります。

- これ以外の場合は、テーブル文字セットとテーブル照合順序が使用されます。

```
CREATE TABLE t1
(
   col1 CHAR(10)
) CHARACTER SET latin1 COLLATE latin1_bin;
```

カラムに対して文字セットも照合順序も指定されていないので、テーブルのデフォルトが使用されます。このカラムには、文字セット latin1 と照合順序 latin1_bin があります。

CHARACTER SET および COLLATE 句は標準 SQL です。

ALTER TABLE を使用して、ある文字セットから別の文字セットにカラムを変換する場合、MySQL はデータ値をマップしようとしますが、文字セットに互換性がない場合、データの損失が生じる可能性があります。

10.1.3.5 文字列リテラルの文字セットおよび照合順序

各文字列リテラルには文字セットと照合順序があります。

文字列リテラルでは、オプションとして文字セットイントロデューサと COLLATE 句を指定できます。

```
[_charset_name]'string' [COLLATE collation_name]
```

例:

```
SELECT 'string';
SELECT _latin1'string';
SELECT _latin1'string' COLLATE latin1_danish_ci;
```

単純なステートメント SELECT 'string' に対して、文字列には character_set_connection および collation_connection システム変数で定義された文字セットと照合順序が適用されます。

_charset_name 式は正式にはイントロデューサと呼ばれます。これは、「後続する文字列が文字セット X を使用する」ことをパーサーに通知します。わかりにくいので強調しておきますが、イントロデューサは、CONVERT() が行うようなイントロデューサ文字セットへの文字列の変換は行いません。パディングが行われる可能性がありますが、文字列の値は変更しません。イントロデューサは単なる信号です。イントロデューサは、標準の 16 進リテラルおよび数値の 16 進リテラル表記 (x'literal' および 0xnnnn) の前でも、ビットフィールドリテラル表記 (b'literal' および 0bnnnn) の前でも有効です。

例:

```
SELECT _latin1 x'AABBCC';
SELECT _latin1 0xAABBCC;
SELECT _latin1 b'1100011';
SELECT _latin1 0b1100011;
```

MySQL では、リテラルの文字セットおよび照合順序が次のように決定されます。

- _X と COLLATE Y の両方が指定されている場合、文字セット X と照合順序 Y が使用されます。
- _X は指定されているが、COLLATE が指定されていない場合、文字セット X とそのデフォルト照合順序が使用されます。各文字セットのデフォルトの照合順序を確認するには、SHOW COLLATION ステートメントを使用します。

- これ以外の場合は、character_set_connection および collation_connection システム変数で指定される文字セットと照合順序が使用されます。

例:

- 文字列に latin1 文字セットと latin1_german1_ci 照合順序が指定されている場合

```
SELECT _latin1'Müller' COLLATE latin1_german1_ci;
```

- 文字列に latin1 文字セットとそのデフォルト照合順序 (つまり、latin1_swedish_ci) が指定されている場合

```
SELECT _latin1'Müller';
```

- 文字列に接続デフォルト文字セットおよび照合順序が指定されている場合

```
SELECT 'Müller';
```

文字セットイントロデューサと COLLATE 句は、標準 SQL 仕様に基づいて実装されています。

イントロデューサは後続の文字列の文字セットを指定しますが、現在のところ、その文字列内でパーサーがエスケープ処理を実行する方法までは変更しません。エスケープは常に、character_set_connection で指定された文字セットに従ってパーサーが解釈します。

次の例は、イントロデューサが存在する場合でも、character_set_connection を使用して、エスケープ処理が行われることを示します。例では、SET NAMES (セクション10.1.4「接続文字セットおよび照合順序」で説明しているように、character_set_connection を変更します) を使用し、正確な文字列の内容を確認できるように HEX() 関数を使用して結果の文字列を表示します。

例 1:

```
mysql> SET NAMES latin1;
Query OK, 0 rows affected (0.01 sec)

mysql> SELECT HEX('à\n'), HEX(_sjis'à\n');
+------------+-----------------+
| HEX('à\n') | HEX(_sjis'à\n') |
+------------+-----------------+
| E00A       | E00A            |
+------------+-----------------+
1 row in set (0.00 sec)
```

ここでは、「à」(16 進値 E0) のあとに、改行のエスケープシーケンスである「\n」が続いています。このエスケープシーケンスは、latin1 の character_set_connection 値を使用して解釈され、リテラルの改行 (16 進値 0A) を生成します。これは 2 番目の文字列にも行われます。つまり、_sjis のイントロデューサはパーサーのエスケープ処理に影響を及ぼしません。

例 2:

```
mysql> SET NAMES sjis;
Query OK, 0 rows affected (0.00 sec)

mysql> SELECT HEX('à\n'), HEX(_latin1'à\n');
+------------+-------------------+
| HEX('à\n') | HEX(_latin1'à\n') |
+------------+-------------------+
| E05C6E     | E05C6E            |
+------------+-------------------+
1 row in set (0.04 sec)
```

ここでは、character_set_connection は sjis です。この文字セットでは、「à」のあとに「\」(16 進値 05 および 5C) が続いたシーケンスは、有効なマルチバイト文字になります。したがって、文字列の最初の 2 バイトは、単一の sjis 文字として解釈され、「\」はエスケープ文字として解釈されません。後続の「n」(16 進値 6E) はエスケープシーケンスの一部としては解釈されません。これは 2 番目の文字列にも当てはまります。_latin1 のイントロデューサはエスケープ処理に影響を及ぼしません。

10.1.3.6 各国文字セット

標準 SQL では、NCHAR または NATIONAL CHAR は、CHAR カラムで事前定義された文字セットを使用するように指定する方法として定義されています。MySQL 5.6 では、utf8 を事前定義された文字セットとして使用します。たとえば、次のデータ型宣言は同等です。

```
CHAR(10) CHARACTER SET utf8
NATIONAL CHARACTER(10)
```

```
NCHAR(10)
```

次も同様です。

```
VARCHAR(10) CHARACTER SET utf8
NATIONAL VARCHAR(10)
NCHAR VARCHAR(10)
NATIONAL CHARACTER VARYING(10)
NATIONAL CHAR VARYING(10)
```

N'literal' (または n'literal') を使用すると、各国文字セットの文字列を作成できます。次のステートメントは同等です。

```
SELECT N'some text';
SELECT n'some text';
SELECT _utf8'some text';
```

4.1 以前のバージョンから MySQL 5.6 への文字セットのアップグレードに関する情報については、MySQL 3.23、4.0、4.1 リファレンスマニュアルを参照してください。

10.1.3.7 文字セットと照合順序の割り当ての例

MySQL でどのようにしてデフォルトの文字セットおよび照合順序の値が決定されるかを、次の例で示します。

例 1: テーブルおよびカラムの定義

```
CREATE TABLE t1
(
  c1 CHAR(10) CHARACTER SET latin1 COLLATE latin1_german1_ci
) DEFAULT CHARACTER SET latin2 COLLATE latin2_bin;
```

ここでは latin1 文字セットと latin1_german1_ci 照合順序がカラムに指定されています。この定義は明確であり、簡単明瞭です。なお、latin1 カラムを latin2 テーブルに格納することに問題はありません。

例 2: テーブルおよびカラムの定義

```
CREATE TABLE t1
(
  c1 CHAR(10) CHARACTER SET latin1
) DEFAULT CHARACTER SET latin1 COLLATE latin1_danish_ci;
```

ここでは、latin1 文字セットとデフォルト照合順序がカラムに指定されています。当然のようですが、デフォルトの照合順序はテーブルレベルからは取得されません。latin1 のデフォルト照合順序は常に latin1_swedish_ci なので、カラム c1 には latin1_danish_ci ではなく latin1_swedish_ci の照合順序が設定されます。

例 3: テーブルおよびカラムの定義

```
CREATE TABLE t1
(
  c1 CHAR(10)
) DEFAULT CHARACTER SET latin1 COLLATE latin1_danish_ci;
```

ここでは、デフォルト文字セットとデフォルト照合順序がカラムに指定されています。この状況では、MySQL はテーブルレベルをチェックして、カラムの文字セットおよび照合順序を特定します。したがって、カラム c1 の文字セットは latin1、照合順序は latin1_danish_ci となります。

例 4: データベース、テーブル、およびカラムの定義

```
CREATE DATABASE d1
  DEFAULT CHARACTER SET latin2 COLLATE latin2_czech_ci;
USE d1;
CREATE TABLE t1
(
  c1 CHAR(10)
);
```

文字セットと照合順序を指定せずにカラムを作成します。テーブルレベルの文字セットと照合順序も指定しません。この状況では、MySQL は、データベースレベルをチェックして、テーブル設定を特定します。その後、この設定がカラム設定になります。したがって、カラム c1 の文字セットは latin2、照合順序は latin2_czech_ci となります。

10.1.3.8 ほかの DBMS との互換性

MaxDB との互換性について、次の 2 つのステートメントは同じです。

```
CREATE TABLE t1 (f1 CHAR(N) UNICODE);
CREATE TABLE t1 (f1 CHAR(N) CHARACTER SET ucs2);
```

10.1.4 接続文字セットおよび照合順序

複数の文字セットおよび照合順序のシステム変数は、サーバーとのクライアントの通信に関係しています。これらのいくつかは、これまでのセクションですでに説明されています。

- サーバー文字セットおよび照合順序は、character_set_server および collation_server システム変数の値です。
- デフォルトのデータベースの文字セットおよび照合順序は、character_set_database および collation_database システム変数の値です。

その他の文字セットおよび照合順序システム変数は、クライアントとサーバー間の接続のトラフィックの処理に関わっています。どのクライアントも接続関連の文字セットおよび照合順序システム変数を持っています。

「接続」は、サーバーに接続するときに作成するものです。クライアントは接続を介して、SQL ステートメント (クエリーなど) をサーバーに送信します。サーバーは接続を介して、結果セットやエラーメッセージなどの応答をクライアントに送信します。これによって、次のようなクライアント接続を扱う文字セットおよび照合順序に関する疑問が生じますが、これらはシステム変数の点から回答できます。

- クライアントから送信されるときに、ステートメントはどの文字セットで送信されますか。

  サーバーは、character_set_client システム変数値を、クライアントが送信するステートメントの文字セットにします。

- ステートメントを受信したあとで、サーバーはこれをどの文字セットに変換しますか。

  これには、サーバーは character_set_connection および collation_connection システム変数値を使用します。クライアントから送信されたステートメントは、character_set_client から character_set_connection に変換されます (_latin1 や _utf8 などのイントロデューサがある文字列リテラルを除きます)。collation_connection はリテラル文字列の比較で重要です。カラム値のある文字列の比較には、collation_connection は重要視されません。なぜなら、カラムには独自の照合順序があり、この照合順序が優先されるからです。

- 結果セットまたはエラーメッセージをクライアントに返送する前に、サーバーはこれらをどの文字セットに変換しますか。

  character_set_results システム変数値は、サーバーがクライアントにクエリー結果を返信するときに使用する文字セットを示します。これには、カラム値などの結果データと、カラム名やエラーメッセージなどの結果メタデータが含まれます。

クライアントは、これらの変数の設定を微調整することも、デフォルトに従うこともできます (この場合は、このセクションの残りをスキップできます)。デフォルトを使用しない場合、サーバーへの接続ごとに文字設定を変更する必要があります。

2 つのステートメントは、接続関連の文字セット変数にグループとして影響します。

- SET NAMES 'charset_name' [COLLATE 'collation_name']

  SET NAMES は、クライアントからサーバーへの SQL ステートメントの送信に使用される文字セットを示します。したがって、SET NAMES 'cp1251' は、「このクライアントから今後受信するメッセージが文字セット cp1251 で送信される」ことを、サーバーに知らせます。また、クライアントに結果を返信するときにサーバーが使用する文字セットも指定します。(たとえば、SELECT ステートメントを使用する場合に、カラム値に使用する文字セットを指定します。)

  SET NAMES 'charset_name' ステートメントは次の 3 つのステートメントと同等です。

  ```
  SET character_set_client = charset_name;
  SET character_set_results = charset_name;
  SET character_set_connection = charset_name;
  ```

  character_set_connection を charset_name に設定すると、collation_connection も暗黙的に charset_name のデフォルト照合順序に設定されます。この照合順序を明示的に設定する必要はありません。特定の照合順序を指定するには、オプションの COLLATE 句を使用します。

  ```
  SET NAMES 'charset_name' COLLATE 'collation_name'
  ```

- SET CHARACTER SET charset_name

SET CHARACTER SET は SET NAMES に似ていますが、character_set_connection と collation_connection を character_set_database と collation_database に設定します。SET CHARACTER SET charset_name ステートメントは次の 3 つのステートメントと同等です。

```
SET character_set_client = charset_name;
SET character_set_results = charset_name;
SET collation_connection = @@collation_database;
```

collation_connection を設定すると、character_set_connection も、関連付けられた文字セットに暗黙的に設定されます (SET character_set_connection = @@character_set_database の実行と同等です)。character_set_connection を明示的に設定する必要はありません。

> **注記**
>
> ucs2、utf16、utf16le、および utf32 をクライアント文字セットとして使用することはできません。つまり、これらは SET NAMES または SET CHARACTER SET には機能しません。

MySQL クライアントプログラム mysql、mysqladmin、mysqlcheck、mysqlimport、および mysqlshow は、次のように、使用するデフォルトの文字セットを特定します。

- ほかの情報が欠如している場合、プログラムは、コンパイル時のデフォルトの文字セット (通常は latin1) を使用します。
- プログラムは、オペレーティングシステム設定 (たとえば、Unix システムでは LANG や LC_ALL ローカル環境変数の値、Windows システムではコードページ設定) に基づいて、使用する文字セットを自動検出できます。ロケールが OS から利用できるシステムの場合、クライアントはコンパイル時のデフォルトを使用するのではなく、このロケールを使用してデフォルトの文字セットを設定します。たとえば、LANG を ru_RU.KOI8-R に設定すると、koi8r 文字セットが使用されます。したがってユーザーは、MySQL クライアントが使用できるように、自身の環境内でロケールを構成できます。

  OS 文字セットは、正確に一致するものがない場合は、もっとも近い MySQL 文字セットにマップされます。一致した文字セットをサポートしていない場合、クライアントはコンパイルイ時のデフォルトを使用します。たとえば、ucs2 は接続文字セットとしてはサポートされていません。

  C アプリケーションは、サーバーに接続する前に次のように mysql_options() を呼び出すことによって、OS 設定に基づいて文字セットの自動検出を使用できます。

  ```
  mysql_options(mysql,
          MYSQL_SET_CHARSET_NAME,
          MYSQL_AUTODETECT_CHARSET_NAME);
  ```

- プログラムは --default-character-set オプションをサポートしており、ユーザーはこのオプションを使用すると文字セットを明示的に指定でき、クライアントがそれ以外のどのデフォルトを指定していても、それをオーバーライドできます。

クライアントはサーバーに接続するときに、使用する文字セットの名前を送信します。サーバーはこの名前を使用して、character_set_client、character_set_results、および character_set_connection システム変数を設定します。実際には、サーバーは文字セット名を使用して SET NAMES 操作を実行します。

mysql クライアントの場合、デフォルトとは別の文字セットを使用するには、起動するたびに、SET NAMES を明示的に実行できます。より簡単に同じ結果を得るには、--default-character-set オプション設定を mysql コマンド行またはオプションファイルに追加します。たとえば、次のオプションファイル設定は、mysql を呼び出すたびに、koi8r に設定された 3 つの接続関連の文字セット変数を変更します。

```
[mysql]
default-character-set=koi8r
```

自動再接続を有効にして mysql クライアントを使用している場合は (推奨しません)、SET NAMES ではなく charset コマンドを使用することをお勧めします。例:

```
mysql> charset utf8
Charset changed
```

charset コマンドは、SET NAMES ステートメントを発行し、接続の切断後に再接続するときに mysql が使用するデフォルトの文字セットも変更します。

例: column1 が CHAR(5) CHARACTER SET latin2 として定義されているとします。SET NAMES または SET CHARACTER SET を指定しない場合、SELECT column1 FROM t に対して、サーバーは、接続時にクライアント

が指定した文字セットを使用して、column1 のすべての値を送り返します。反対に、SET NAMES 'latin1' または SET CHARACTER SET latin1 を SELECT ステートメントを発行する前に指定した場合、サーバーは結果を返信する直前に、latin2 の値を latin1 に変換します。両方の文字セットに存在しない文字がある場合、変換の損失が大きくなる可能性があります。

サーバーに結果セットまたはエラーメッセージの変換を実行させない場合は、character_set_results を NULL または binary に設定してください。

```
SET character_set_results = NULL;
```

接続に適用する文字セットおよび照合順序システム変数の値を確認するには、次のステートメントを使用してください。

```
SHOW VARIABLES LIKE 'character_set%';
SHOW VARIABLES LIKE 'collation%';
```

MySQL アプリケーションを実行する環境も考慮する必要があります。セクション10.1.5「アプリケーションの文字セットおよび照合順序の構成」を参照してください。

文字セットおよびエラーメッセージの詳細は、セクション10.1.6「エラーメッセージの文字セット」を参照してください。

10.1.5 アプリケーションの文字セットおよび照合順序の構成

デフォルトの MySQL 文字セットおよび照合順序 (latin1、latin1_swedish_ci) を使用してデータを格納するアプリケーションの場合、特別な構成は必要ありません。別の文字セットまたは照合順序を使用したデータストレージが必要なアプリケーションの場合は、次の複数の方法で、文字セット情報を構成できます。

- データベースごとに文字設定を指定します。たとえば、あるデータベースを使用するアプリケーションでは utf8 が必要で、別のデータベースを使用するアプリケーションでは sjis が必要な場合があります。
- サーバーの起動時に文字設定を指定します。これにより、サーバーは、ほかの調整を行わないすべてのアプリケーションに、所定の設定を使用します。
- ソースから MySQL を構築する場合は、構成時に文字設定を指定します。これにより、サーバーはすべてのアプリケーションに対して所定の設定を使用し、サーバーの起動時に指定する必要がなくなります。

異なるアプリケーションで別々の文字設定が必要な場合は、データベースごとの手法で十分対応できます。ほとんどまたはすべてのアプリケーションで同じ文字セットを使用する場合は、サーバーの起動時または構成時に文字設定を指定する方法がもっとも便利です。

データベースごとの手法またはサーバー起動の手法では、設定によって、データストレージの文字セットが制御されます。アプリケーションはまた、次の手順で説明するように、クライアントとサーバー間の通信に使用する文字セットをサーバーに知らせる必要もあります。

ここで示す例では、utf8 文字セットと utf8_general_ci 照合順序を使用することを想定しています。

データベースごとに文字設定を指定します。テーブルで所定のデフォルトの文字セットおよび照合順序をデータストレージに使用するようなデータベースを作成するには、次のような CREATE DATABASE ステートメントを使用します。

```
CREATE DATABASE mydb
  DEFAULT CHARACTER SET utf8
  DEFAULT COLLATE utf8_general_ci;
```

データベース内に作成されたテーブルは、すべての文字カラムに対してデフォルトで utf8 と utf8_general_ci を使用します。

データベースを使用するアプリケーションは、接続するたびに、サーバーへの接続を構成する必要もあります。これは、接続後に SET NAMES 'utf8' ステートメントを実行することによって行えます。このステートメントは、接続方法 (mysql クライアント、PHP スクリプトなど) とは無関係に使用できます。

場合によっては、必要な文字セットを別の方法で使用するように接続を構成できます。たとえば、mysql を使用して行われた接続の場合、--default-character-set=utf8 コマンド行オプションを指定すると、SET NAMES 'utf8' と同じ効果を得ることができます。

クライアント接続の構成の詳細は、セクション10.1.4「接続文字セットおよび照合順序」を参照してください。

データベースのデフォルトの文字セットまたは照合順序を変更する場合、データベースのデフォルトを使用するストアドルーチンを削除して、新しいデフォルトを使用するように再作成する必要があります。(ストアドルーチンでは、文字セットまたは照合順序が明示的に指定されていない場合、文字データ型を伴う変数は、データベースのデフォルトを使用します。セクション13.1.15「CREATE PROCEDURE および CREATE FUNCTION 構文」を参照してください。)

サーバーの起動時に文字設定を指定します。サーバーの起動時に文字セットおよび照合順序を選択するには、--character-set-server および --collation-server オプションを使用します。たとえば、オプションファイルでオプションを指定するには、次の行を含めます。

```
[mysqld]
character-set-server=utf8
collation-server=utf8_general_ci
```

これらの設定は、サーバー全体に適用され、任意のアプリケーションによって作成されたデータベースのデフォルトおよびこれらのデータベース内に作成されたテーブルのデフォルトとして適用されます。

前述のように、アプリケーションは引き続き、接続したあとで SET NAMES または同等のステートメントを使用して、その接続を構成する必要があります。--init_connect="SET NAMES 'utf8'" オプションでサーバーを起動して、接続するクライアントごとに SET NAMES を自動的に実行させることができます。ただし、SUPER 権限を持つユーザーには init_connect 値は実行されないので、これによって一貫しない結果が生じます。

MySQL の構成時に文字設定を指定します。ソースから MySQL を構成および構築するときに、文字セットおよび照合順序を選択するには、CMake で DEFAULT_CHARSET および DEFAULT_COLLATION オプションを使用します。

```
shell> cmake . -DDEFAULT_CHARSET=utf8 \
        -DDEFAULT_COLLATION=utf8_general_ci
```

この結果構成されたサーバーは、データベースおよびテーブルのデフォルトと、クライアント接続のデフォルトとして、utf8 および utf8_general_ci を使用します。--character-set-server と --collation-server を使用して、サーバー起動時にこれらのデフォルトを指定する必要はありません。また、アプリケーションがサーバーに接続したあとで SET NAMES または同等のステートメントを使用して、その接続を構成する必要もありません。

アプリケーション用に MySQL 文字セットを構成する方法とは無関係に、これらのアプリケーションが実行する環境も考慮する必要があります。エディタで作成したファイルから取得された UTF-8 テキストを使用してステートメントを送信する場合、ファイルのエンコーディングが正しく、オペレーティングシステムで正しく処理されるように、環境の口ケールを UTF-8 に設定してファイルを編集する必要があります。端末ウィンドウ内から mysql クライアントを使用する場合、UTF-8 を使用するようにこのウィンドウを構成する必要があります。そのようにしないと、文字が正しく表示されない可能性があります。Web 環境で実行するスクリプトの場合、このスクリプトは、MySQL Server とやり取りできるように文字のエンコーディングを正しく処理する必要があり、ページ内容の表示方法をブラウザが認識できるようにエンコーディングを正しく指定したページを生成する必要があります。たとえば、次の <meta> タグを <head> 要素内に含められます。

```
<meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
```

10.1.6 エラーメッセージの文字セット

このセクションでは、サーバーがエラーメッセージを構築しクライアントに返信する場合にどのように文字セットを使用するかについて説明します。エラーメッセージの (文字セットではなく) 言語の詳細は、セクション10.2「エラーメッセージ言語の設定」を参照してください。

MySQL 5.6 では、サーバーは、UTF-8 を使用してエラーメッセージを構築し、character_set_results システム変数で指定された文字セットでクライアントに返信します。

サーバーは次のようにエラーメッセージを構築します。

- メッセージテンプレートで UTF-8 が使用されます。
- メッセージテンプレートのパラメータが、特定のエラーの発生に適用される値に置き換えられます。
  - テーブル名やカラム名などの識別子は、UTF-8 を内部で使用するので、そのままコピーされます。
  - 文字 (非バイナリ) 文字列値は、その文字セットから UTF-8 に変換されます。
  - バイナリ文字列値は、0x20 から 0x7E の範囲のバイトについてはそのままコピーされ、この範囲外のバイトについては \x 16 進エンコーディングを使用してコピーされます。たとえば、0x41CF9F を VARBINARY ー

意カラムに挿入しようとしたときに重複キーエラーが発生した場合、この結果生成されるエラーメッセージでは UTF-8 が使用され、一部のバイトは 16 進数でエンコードされます。

```
Duplicate entry 'A\xC3\x9F' for key 1
```

メッセージの構築後、これをクライアントに返すために、サーバーはこのメッセージを UTF-8 から、character_set_results システム変数によって指定された文字セットに変換します。character_set_results に NULL または binary の値がある場合、変換は行われません。変数値が utf8 である場合にも、これが元のエラーメッセージの文字セットに一致するので、変換は行われません。

character_set_results で表すことができない文字の場合、変換中に一部のエンコーディングが行われることがあります。エンコーディングは、Unicode コードポイント値を使用します。

- Basic Multilingual Plane (BMP) 範囲 (0x0000 から 0xFFFF) 内の文字は、\nnnn 表記を使用して書き込まれます。
- BMP 範囲外 (0x01000 から 0x10FFFF) の文字は、\+nnnnnn 表記を使用して書き込まれます。

クライアントは、character_set_results を設定して、エラーメッセージを受信するときの文字セットを制御できます。この変数は直接設定することも、SET NAMES などの手段で間接的に設定することもできます。character_set_results の詳細は、セクション10.1.4「接続文字セットおよび照合順序」を参照してください。

クライアントにエラーメッセージを返信する前に character_set_results に変換する間に行われるエンコーディングの結果、以前のバージョン (MySQL 5.5 より前のバージョン) とは異なるメッセージ内容になることがあります。たとえば、ペ (カタカナのペ) という名前のテーブルを削除しようとしたときにエラーが発生し、character_set_results が、この文字を含まない latin1 などの文字セットである場合、クライアントに送信される結果のメッセージには、次のようにエンコードされたテーブル名が表示されます。

```
ERROR 1051 (42S02): Unknown table '\30DA'
```

MySQL 5.5 より前では、次のように名前はエンコードされません。

```
ERROR 1051 (42S02): Unknown table 'ペ'
```

10.1.7 照合順序の問題

以降のセクションでは、文字セットの照合順序のさまざまな側面について説明します。

10.1.7.1 照合順序名

MySQL 照合順序名は次のルールに従います。

- _ci で終わる名前は大文字と小文字を区別しない照合順序を示します。
- _cs で終わる名前は、大文字と小文字を区別する照合順序を示します。
- _bin で終わる名前はバイナリ照合順序を示します。文字の比較は、文字バイナリコード値に従って行われます。
- Unicode 照合順序名には、照合順序が従う Unicode 照合アルゴリズム (UCA) のバージョンを示すバージョン番号が含まれることがあります。名前にバージョン番号が含まれない UCA ベースの照合順序は、バージョン 4.0.0 UCA 重みキー (http://www.unicode.org/Public/UCA/4.0.0/allkeys-4.0.0.txt) を使用します。utf8_unicode_520_ci などの照合順序名は、UCA 5.2.0 重みキー (http://www.unicode.org/Public/UCA/5.2.0/allkeys.txt) に基づきます。

10.1.7.2 SQL ステートメントでの COLLATE の使用

COLLATE 句では、比較に対するデフォルト照合順序が何であれ、オーバーライドできます。SQL ステートメントのさまざまな個所で COLLATE を使用できます。次にいくつかの例を示します。

- ORDER BY を指定した場合

```
SELECT k
FROM t1
ORDER BY k COLLATE latin1_german2_ci;
```

- AS を指定した場合

```
SELECT k COLLATE latin1_german2_ci AS k1
```

```
FROM t1
ORDER BY k1;
```

- GROUP BY を指定した場合

```
SELECT k
FROM t1
GROUP BY k COLLATE latin1_german2_ci;
```

- 集計関数を指定した場合

```
SELECT MAX(k COLLATE latin1_german2_ci)
FROM t1;
```

- DISTINCT を指定した場合

```
SELECT DISTINCT k COLLATE latin1_german2_ci
FROM t1;
```

- WHERE を指定した場合

```
  SELECT *
  FROM t1
  WHERE _latin1 'Müller' COLLATE latin1_german2_ci = k;
```

```
  SELECT *
  FROM t1
  WHERE k LIKE _latin1 'Müller' COLLATE latin1_german2_ci;
```

- HAVING を指定した場合

```
SELECT k
FROM t1
GROUP BY k
HAVING k = _latin1 'Müller' COLLATE latin1_german2_ci;
```

10.1.7.3 COLLATE 句の優先順位

COLLATE 句は優先順位が高いため (|| より上位)、次の 2 つの式は同等です。

```
x || y COLLATE z
x || (y COLLATE z)
```

10.1.7.4 照合順序と文字セットとの適切な対応

各文字セットには 1 つ以上の照合順序があり、各照合順序は 1 つの文字セットにのみ関連付けられています。したがって、次のステートメントではエラーメッセージが表示されます。latin2_bin 照合順序は latin1 文字セットに対して有効ではないからです。

```
mysql> SELECT _latin1 'x' COLLATE latin2_bin;
ERROR 1253 (42000): COLLATION 'latin2_bin' is not valid
for CHARACTER SET 'latin1'
```

10.1.7.5 式の照合順序

大多数のステートメントでは、MySQL がどの照合順序を使用して比較演算を行うかは明確になっています。たとえば、次の場合、照合順序がカラム charset_name の照合順序であることは明らかです。

```
SELECT x FROM T ORDER BY x;
SELECT x FROM T WHERE x = x;
SELECT DISTINCT x FROM T;
```

ただし、複数のオペランドがあると、あいまいさが生じることがあります。例:

```
SELECT x FROM T WHERE x = 'Y';
```

カラム x の照合順序を使用して比較したほうがよいでしょうか、それとも文字列リテラル 'Y' の照合順を使用して比較したほうがよいででしょうか。x と 'Y' の両方に照合順序がありますが、どちらの照合順序が優先されるでしょうか。

標準 SQL では、「強制性」ルールと呼ばれていた方法で上記の問題を解決しています。MySQL は次のように強制性値を割り当てます。

- 明示的な COLLATE 句の強制性は 0 です。(強制性がまったくありません。)
- 照合順序の異なる 2 つの文字列を連結すると強制性は 1 になります。
- カラムまたはストアドルーチンのパラメータまたはローカル変数の照合順序の強制性は 2 です。
- 「システム定数」 (USER() または VERSION() などの関数で返される文字列) の強制性は 3 です。
- リテラルの照合順序の強制性は 4 です。
- NULL または NULL から派生した式の強制性 5 です。

MySQL は、次のルールとともに強制性値を使用して、あいまいさを解決します。

- 強制性値がもっとも低い照合順序を使用します。
- 両方の側で強制性が同じ場合は、次のようになります。
  - 両方の側が Unicode であるか、両方の側が Unicode ではない場合は、エラーになります。
  - どちらかの側に Unicode 文字セットがあり、もう一方の側に Unicode 以外の文字セットがある場合、Unicode 文字セットの側が優先され、Unicode 以外の側に自動文字セット変換が適用されます。たとえば、次のステートメントはエラーを返しません。

    ```
    SELECT CONCAT(utf8_column, latin1_column) FROM t1;
    ```

    これは、utf8 の文字セットと、utf8_column と同じ照合順序を含む結果を返します。連結する前に、latin1_column の値は自動的に、utf8 に変換されます。
  - 同じ文字セットのオペランドだが、_bin 照合順序と _ci または _cs 照合順序が混在したオペランドを使用した演算の場合、_bin 照合順序が使用されます。これは、対象がデータ型でなく照合順序である点を除けば、非バイナリ文字列とバイナリ文字列が混在した演算でオペランドがバイナリ文字列に評価される方法に似ています。

自動変換機能は SQL 標準には含まれていません。ただし、どの文字セットも (サポートされている文字に関して) Unicode の「サブセット」であることが SQL 標準のドキュメントに記載されています。「スーパーセットに適用されるものはサブセットにも適用される」というよく知られた原則があるので、Unicode の照合順序は Unicode 以外の文字列との比較にも適用できると考えられます。

例:

比較	使用される照合順序
column1 = 'A'	column1 の照合順序を使用します
column1 = 'A' COLLATE x	'A' COLLATE x の照合順序を使用します
column1 COLLATE x = 'A' COLLATE y	エラー

COERCIBILITY() 関数を使用すると、文字列式の強制性を特定できます。

```
mysql> SELECT COERCIBILITY('A' COLLATE latin1_swedish_ci);
    -> 0
mysql> SELECT COERCIBILITY(VERSION());
    -> 3
mysql> SELECT COERCIBILITY('A');
    -> 4
```

セクション12.14「情報関数」を参照してください。

式 CONCAT(1, 'abc') での引数 1 に対して行われる変換など、数値または時間値の文字列への暗黙の変換の場合、その結果は、character_set_connection および collation_connection システム変数で指定された文字セットおよび照合順序を持つ文字 (非バイナリ) 列になります。セクション12.2「式評価での型変換」を参照してください。

10.1.7.6 _bin および binary 照合順序

このセクションでは、非バイナリ文字列の _bin 照合順序が、バイナリ文字列の binary 「照合順序」とどのように異なるかについて説明します。

非バイナリ文字列 (CHAR、VARCHAR、および TEXT データ型に格納) には、文字セットと照合順序があります。特定の文字セットには複数の照合順序が対応していることがあり、そのそれぞれが、セット内の文字の具体

的なソートおよび比較順序を定義しています。これらの中の 1 つが文字セットのバイナリ照合順序であり、その照合順序名の中には _bin サフィクスがあります。たとえば、latin1 と utf8 には、latin1_bin および utf8_bin という名前のバイナリ照合順序があります。

バイナリ文字列 (BINARY、VARBINARY、および BLOB データ型に格納) には、非バイナリ文字列にあるような文字セットや照合順序はありません。(CHARSET() と COLLATION() 関数はどちらも、バイナリ文字列に適用されると binary の値を返します。)バイナリ文字列はバイトのシーケンスであり、これらのバイトの数値がソート順序を決定します。

_bin 照合順序は、いくつかの点で binary 照合順序と異なります。

ソートおよび比較の単位。バイナリ文字列は、バイトのシーケンスです。ソートおよび比較は、常に数値のバイト値に基づきます。非バイナリ文字列は文字のシーケンスであり、これはマルチバイトであることもあります。非バイナリ文字列の照合順序は、文字値のソートおよび比較の順序を定義します。_bin 照合順序の場合、この順序は文字のバイナリコード値にのみ基づきます (これは、_bin 照合順序では複数バイトの文字の可能性を考慮に入れる必要がある点を除き、バイナリ文字列の順序に似ています)。ほかの照合順序の場合、文字の順序は、大文字と小文字の区別などのその他の要素を考慮に入れることがあります。

文字セットの変換。非バイナリ文字列には文字セットがあり、多くの場合、文字列に _bin 照合順序があるときでも、別の文字セットに変換されます。

- 別の文字セットを持つほかのカラムから、カラム値を割り当てる場合:

```
UPDATE t1 SET utf8_bin_column=latin1_column;
INSERT INTO t1 (latin1_column) SELECT utf8_bin_column FROM t2;
```

- 文字列リテラルを使用して、INSERT または UPDATE のカラム値を割り当てる場合:

```
SET NAMES latin1;
INSERT INTO t1 (utf8_bin_column) VALUES ('string-in-latin1');
```

- サーバーからクライアントに結果を送信する場合:

```
SET NAMES latin1;
SELECT utf8_bin_column FROM t2;
```

バイナリ文字列カラムの場合、変換は行われません。前述の場合、文字列値はバイトのようにコピーされます。

大文字と小文字の変換。照合順序は文字の大文字と小文字に関する情報をもたらすので、非バイナリ文字列内の文字は大文字と小文字を変換できます。これは、順序で大文字と小文字の区別を無視する _bin 照合順序の場合でも該当します。

```
mysql> SET NAMES latin1 COLLATE latin1_bin;
Query OK, 0 rows affected (0.02 sec)

mysql> SELECT LOWER('aA'), UPPER('zZ');
+-------------+-------------+
| LOWER('aA') | UPPER('zZ') |
+-------------+-------------+
| aa          | ZZ          |
+-------------+-------------+
1 row in set (0.13 sec)
```

大文字と小文字という概念は、バイナリ文字列内のバイトには適用されません。大文字と小文字の変換を行うには、文字列を非バイナリ文字列に変換する必要があります。

```
mysql> SET NAMES binary;
Query OK, 0 rows affected (0.00 sec)

mysql> SELECT LOWER('aA'), LOWER(CONVERT('aA' USING latin1));
+-------------+--------------------------------------+
| LOWER('aA') | LOWER(CONVERT('aA' USING latin1)) |
+-------------+--------------------------------------+
| aA          | aa                                   |
+-------------+--------------------------------------+
1 row in set (0.00 sec)
```

比較での後続スペースの処理。非バイナリ文字列には、_bin 照合順序を含むすべての照合順序に対する PADSPACE 動作があります。後続スペースは比較で意味がありません。

```
mysql> SET NAMES utf8 COLLATE utf8_bin;
Query OK, 0 rows affected (0.00 sec)
```

```
mysql> SELECT 'a ' = 'a';
+------------+
| 'a ' = 'a' |
+------------+
|          1 |
+------------+
1 row in set (0.00 sec)
```

バイナリ文字列では、後続スペースを含むすべての文字が比較で重要になります。

```
mysql> SET NAMES binary;
Query OK, 0 rows affected (0.00 sec)

mysql> SELECT 'a ' = 'a';
+------------+
| 'a ' = 'a' |
+------------+
|          0 |
+------------+
1 row in set (0.00 sec)
```

挿入および取り出しでの後続スペースの処理。CHAR(N) 列には非バイナリ文字列が格納されます。N 文字より短い値は、挿入時にスペースで拡張されます。取り出しの場合、後続スペースは削除されます。

BINARY(N) 列にはバイナリ文字列が格納されます。N バイトより短い値は、挿入時に 0x00 バイトで拡張されます。取り出しの場合、何も削除されず、宣言された長さの値が常に返されます。

```
mysql> CREATE TABLE t1 (
    ->   a CHAR(10) CHARACTER SET utf8 COLLATE utf8_bin,
    ->   b BINARY(10)
    -> );
Query OK, 0 rows affected (0.09 sec)

mysql> INSERT INTO t1 VALUES ('a','a');
Query OK, 1 row affected (0.01 sec)

mysql> SELECT HEX(a), HEX(b) FROM t1;
+--------+----------------------+
| HEX(a) | HEX(b)               |
+--------+----------------------+
| 61     | 61000000000000000000 |
+--------+----------------------+
1 row in set (0.04 sec)
```

10.1.7.7 BINARY 演算子

BINARY 演算子は、それに続く文字列をバイナリ文字列にキャストします。これは、比較を文字ごとでなくバイトごとに強制的に実行させる簡単な方法です。また、BINARY では末尾の空白文字が重要になります。

```
mysql> SELECT 'a' = 'A';
       -> 1
mysql> SELECT BINARY 'a' = 'A';
       -> 0
mysql> SELECT 'a' = 'a ';
       -> 1
mysql> SELECT BINARY 'a' = 'a ';
       -> 0
```

BINARY str は CAST(str AS BINARY) の略でもあります。

文字カラム定義内の BINARY 属性には別の効果があります。BINARY 属性で定義された文字カラムには、カラム文字セットのバイナリ照合順序が割り当てられます。すべての文字セットにバイナリ照合順序があります。たとえば、latin1 文字セットのバイナリ照合順序は latin1_bin です。このためテーブルのデフォルト文字セットが latin1 の場合、次の 2 つのカラム定義は同等になります。

```
CHAR(10) BINARY
CHAR(10) CHARACTER SET latin1 COLLATE latin1_bin
```

BINARY のカラム属性としての効果は、MySQL 4.1 より前の効果とは異なります。以前は、BINARY は、バイナリ文字列として扱われたカラムになっていました。バイナリ文字列は、文字セットや照合順序がないバイトの文字列であり、バイナリ照合順序を持つ非バイナリ文字列とは異なります。どちらのタイプの文字列についても、比較は文字列単位の数値に基づいて行われますが、非バイナリ文字列については、単位は文字であり、文字セットによってはマルチバイト文字をサポートします。セクション11.4.2「BINARY および VARBINARY 型」を参照してください。

CHAR、VARCHAR、または TEXT カラムの定義で CHARACTER SET binary を使用すると、このカラムはバイナリデータ型として扱われます。たとえば、次のペアになった定義は同等です。

```
CHAR(10) CHARACTER SET binary
BINARY(10)

VARCHAR(10) CHARACTER SET binary
VARBINARY(10)

TEXT CHARACTER SET binary
BLOB
```

10.1.7.8 照合順序の効果の例

例 1: ドイツ語のウムラウトのソート

テーブル T のカラム X に次の latin1 カラムの値が設定されているとします。

```
Muffler
Müller
MX Systems
MySQL
```

さらに、次のステートメントを使用してカラムの値を取得するとします。

```
SELECT X FROM T ORDER BY X COLLATE collation_name;
```

次の表に、別の照合順序で ORDER BY を使用した場合に得られる値の順序を示します。

latin1_swedish_ci	latin1_german1_ci	latin1_german2_ci
Muffler	Muffler	Müller
MX Systems	Müller	Muffler
Müller	MX Systems	MX Systems
MySQL	MySQL	MySQL

この例でソート順の違いを生じさせている文字は、頭に 2 つの点が付いた U (ü) であり、これはドイツ語で「U ウムラウト」と呼ばれているものです。

- 最初のカラムには、スウェーデン語/フィンランド語の照合ルールを使用した SELECT の結果が示されています。この照合ルールによると、U ウムラウトは Y とソート順が一致します。
- 2 番目のカラムには、German DIN-1 ルールを使用した SELECT の結果が示されています。この照合ルールによると、U ウムラウトは U とソート順が一致します。
- 3 番目のカラムには、German DIN-2 ルールを使用した SELECT の結果が示されています。この照合ルールによると、U ウムラウトは UE とソート順が一致します。

例 2: ドイツ語のウムラウトの検索

使用される文字セットと照合順序だけが異なる次の 3 つのテーブルがあるとします。

```
mysql> SET NAMES utf8;
mysql> CREATE TABLE german1 (
    ->   c CHAR(10)
    -> ) CHARACTER SET latin1 COLLATE latin1_german1_ci;
mysql> CREATE TABLE german2 (
    ->   c CHAR(10)
    -> ) CHARACTER SET latin1 COLLATE latin1_german2_ci;
mysql> CREATE TABLE germanutf8 (
    ->   c CHAR(10)
    -> ) CHARACTER SET utf8 COLLATE utf8_unicode_ci;
```

各テーブルには次の 2 つのレコードが含まれます。

```
mysql> INSERT INTO german1 VALUES ('Bar'), ('Bär');
mysql> INSERT INTO german2 VALUES ('Bar'), ('Bär');
mysql> INSERT INTO germanutf8 VALUES ('Bar'), ('Bär');
```

上記の照合順序の 2 つには A = Ä の等式が含まれ、1 つにはこのような等式はありません (latin1_german2_ci)。そのため、比較では次の結果が得られます。

```
mysql> SELECT * FROM german1 WHERE c = 'Bär';
+------+
| c    |
+------+
| Bar  |
| Bär  |
+------+
mysql> SELECT * FROM german2 WHERE c = 'Bär';
+------+
| c    |
+------+
| Bär  |
+------+
mysql> SELECT * FROM germanutf8 WHERE c = 'Bär';
+------+
| c    |
+------+
| Bar  |
| Bär  |
+------+
```

これはバグではなく、latin1_german1_ci と utf8_unicode_ci のソートプロパティーの結果です (示されたソートは、German DIN 5007 標準に従って行われています)。

10.1.7.9 照合順序と INFORMATION_SCHEMA 検索

INFORMATION_SCHEMA テーブルでの文字列カラムには utf8_general_ci の照合順序があり、これは大文字と小文字を区別しません。ただし、INFORMATION_SCHEMA 文字列カラムでの検索には、ファイルシステムでの大文字と小文字の区別も影響します。データベースやテーブルの名前など、ファイルシステムで表されるオブジェクトに対応する値については、ファイルシステムで大文字と小文字を区別する場合は、検索で大文字と小文字が区別されることがあります。このセクションでは、必要に応じて、この問題を解決する方法を説明します。Bug #34921 も参照してください。

クエリーで、test データベースに対し SCHEMATA.SCHEMA_NAME カラムを検索するとします。Linux では、ファイルシステムで大文字と小文字を区別するので、SCHEMATA.SCHEMA_NAME の 'test' との比較は一致しますが、'TEST' との比較は一致しません。

```
mysql> SELECT SCHEMA_NAME FROM INFORMATION_SCHEMA.SCHEMATA
    -> WHERE SCHEMA_NAME = 'test';
+-------------+
| SCHEMA_NAME |
+-------------+
| test        |
+-------------+
1 row in set (0.01 sec)

mysql> SELECT SCHEMA_NAME FROM INFORMATION_SCHEMA.SCHEMATA
    -> WHERE SCHEMA_NAME = 'TEST';
Empty set (0.00 sec)
```

Windows または OS X では、ファイルシステムで大文字と小文字を区別しないので、比較は 'test' と 'TEST' の両方と一致します。

```
mysql> SELECT SCHEMA_NAME FROM INFORMATION_SCHEMA.SCHEMATA
    -> WHERE SCHEMA_NAME = 'test';
+-------------+
| SCHEMA_NAME |
+-------------+
| test        |
+-------------+
1 row in set (0.00 sec)

mysql> SELECT SCHEMA_NAME FROM INFORMATION_SCHEMA.SCHEMATA
    -> WHERE SCHEMA_NAME = 'TEST';
+-------------+
| SCHEMA_NAME |
+-------------+
| TEST        |
+-------------+
1 row in set (0.00 sec)
```

lower_case_table_names システム変数の値はこのコンテキストでは違いをもたらしません。

ファイルシステム内でデータベースオブジェクトを検索するときに、INFORMATION_SCHEMA クエリーで utf8_general_ci 照合順序が使用されていないために、このような処理が行われます。これは、MySQL での

INFORMATION_SCHEMA 検索に対して実装された最適化の結果です。これらの最適化の詳細は、セクション 8.2.4「INFORMATION_SCHEMA クエリーの最適化」を参照してください。

INFORMATION_SCHEMA は 「仮想」データベースであり、ファイルシステムには表されないので、INFORMATION_SCHEMA 文字列カラム内で INFORMATION_SCHEMA 自体を参照する値を検索する場合、utf8_general_ci 照合順序を使用します。たとえば、SCHEMATA.SCHEMA_NAME との比較は、プラットフォームに関係なく、'information_schema' または 'INFORMATION_SCHEMA' に一致します。

```
mysql> SELECT SCHEMA_NAME FROM INFORMATION_SCHEMA.SCHEMATA
    -> WHERE SCHEMA_NAME = 'information_schema';
+--------------------+
| SCHEMA_NAME        |
+--------------------+
| information_schema |
+--------------------+
1 row in set (0.00 sec)

mysql> SELECT SCHEMA_NAME FROM INFORMATION_SCHEMA.SCHEMATA
    -> WHERE SCHEMA_NAME = 'INFORMATION_SCHEMA';
+--------------------+
| SCHEMA_NAME        |
+--------------------+
| information_schema |
+--------------------+
1 row in set (0.00 sec)
```

INFORMATION_SCHEMA カラムでの文字列演算の結果が予想と異なる場合、回避策は、明示的な COLLATE 句を使用して、適切な照合順序を強制的に使用することです (セクション10.1.7.2「SQL ステートメントでの COLLATE の使用」)。たとえば、大文字と小文字を区別しない検索を実行するには、INFORMATION_SCHEMA カラム名とともに COLLATE を使用します。

```
mysql> SELECT SCHEMA_NAME FROM INFORMATION_SCHEMA.SCHEMATA
    -> WHERE SCHEMA_NAME COLLATE utf8_general_ci = 'test';
+-------------+
| SCHEMA_NAME |
+-------------+
| test        |
+-------------+
1 row in set (0.00 sec)

mysql> SELECT SCHEMA_NAME FROM INFORMATION_SCHEMA.SCHEMATA
    -> WHERE SCHEMA_NAME COLLATE utf8_general_ci = 'TEST';
| SCHEMA_NAME |
+-------------+
| test        |
+-------------+
1 row in set (0.00 sec)
```

UPPER() または LOWER() 関数を使用することもできます。

```
WHERE UPPER(SCHEMA_NAME) = 'TEST'
WHERE LOWER(SCHEMA_NAME) = 'test'
```

大文字と小文字を区別するファイルシステムのプラットフォームでも、大文字と小文字を区別しない比較を実行できますが、前述のように、必ずしも常に正しい処理になるとはかぎりません。このようなプラットフォームでは、大文字と小文字だけが異なる名前の複数のオブジェクトが存在する可能性があります。たとえば、city、CITY、および City という名前のテーブルがすべて同時に存在することが可能です。検索でこれらのすべての名前に一致するか、1 つだけに一致するかを検討し、それに応じてクエリーを作成します。

```
WHERE TABLE_NAME COLLATE utf8_bin = 'City'
WHERE TABLE_NAME COLLATE utf8_general_ci = 'city'
WHERE UPPER(TABLE_NAME) = 'CITY'
WHERE LOWER(TABLE_NAME) = 'city'
```

これらのうち最初の比較 (utf8_bin 使用) では大文字と小文字を区別します。ほかの比較では区別しません。

10.1.8 文字列のレパートリー

文字セットのレパートリーとは、そのセット内の文字の集合です。

文字列式にはレパートリー属性があり、その値は次の 2 つです。

- ASCII: U+0000 から U+007F の Unicode 範囲内の文字だけを式に含めることができます。

- UNICODE: U+0000 から U+FFFF の Unicode 範囲内の文字を式に含めることができます。

ASCII 範囲は UNICODE 範囲のサブセットであるので、ASCII レパートリーの文字列は、UNICODE レパートリーの文字列の文字セットに、または ASCII のスーパーセットである文字セットに、情報を失うことなく安全に変換できます。(すべての MySQL 文字セットは、スウェーデン語のアクセント符号付き文字に一部の句読点文字を再使用する swe7 を除き、ASCII のスーパーセットです。)MySQL から「照合順序の不正な混在」エラーが返されるような多くの場合でも、レパートリーを使用すると、式内の文字セットを変換できます。

次の説明では、式とそのレパートリーの例を挙げ、レパートリーの使用によって、文字列式の評価がどのように変わるかを示します。

- 文字列定数のレパートリーは、文字列の内容に応じて異なります。

```
SET NAMES utf8; SELECT 'abc';
SELECT _utf8'def';
SELECT N'MySQL';
```

前述のそれぞれの場合では文字セットは utf8 ですが、実際には ASCII 範囲外の文字は 1 つも文字列に含まれていないので、そのレパートリーは UNICODE ではなく ASCII です。

- ascii 文字セットを含むカラムには、その文字セットのために、ASCII レパートリーがあります。次のテーブルでは、c1 には ASCII レパートリーがあります。

```
CREATE TABLE t1 (c1 CHAR(1) CHARACTER SET ascii);
```

次の例では、レパートリーがないときにエラーが発生する場合に、レパートリーによってどのように結果が求められるようになるかを示しています。

```
CREATE TABLE t1 (
  c1 CHAR(1) CHARACTER SET latin1,
  c2 CHAR(1) CHARACTER SET ascii
);
INSERT INTO t1 VALUES ('a','b');
SELECT CONCAT(c1,c2) FROM t1;
```

レパートリーがない場合、次のエラーが発生します。

```
ERROR 1267 (HY000): Illegal mix of collations (latin1_swedish_ci,IMPLICIT)
and (ascii_general_ci,IMPLICIT) for operation 'concat'
```

レパートリーを使用すると、サブセットからスーパーセットへ (ascii から latin1 へ) の変換を行うことができ、結果が返されます。

```
+--------------+
| CONCAT(c1,c2) |
+--------------+
| ab           |
+--------------+
```

- 1 つの文字列引数を持つ関数は、その引数のレパートリーを継承します。UPPER(_utf8'abc') の場合、引数に ASCII レパートリーがあるので、その結果には ASCII レパートリーが存在します。

- 文字列を返すが、文字列引数を持たず、結果文字セットとして character_set_connection を使用する関数の場合、結果のレパートリーは、character_set_connection が ascii である場合は ASCII に、それ以外の場合は UNICODE になります。

```
FORMAT(numeric_column, 4);
```

レパートリーを使用するかどうかによって、MySQL が次の例をどのように評価するかが変わります。

```
SET NAMES ascii;
CREATE TABLE t1 (a INT, b VARCHAR(10) CHARACTER SET latin1);
INSERT INTO t1 VALUES (1,'b');
SELECT CONCAT(FORMAT(a, 4), b) FROM t1;
```

レパートリーがない場合、次のエラーが発生します。

```
ERROR 1267 (HY000): Illegal mix of collations (ascii_general_ci,COERCIBLE)
and (latin1_swedish_ci,IMPLICIT) for operation 'concat'
```

レパートリーを使用すると、次のように結果が返されます。

```
+------------------------+
| CONCAT(FORMAT(a, 4), b) |
```

```
+------------------------+
| 1.0000b                |
+------------------------+
```

- 2 つ以上の文字列引数を持つ関数は、結果のレパートリーとして、「もっとも範囲の広い」引数レパートリーを使用します (UNICODE は ASCII よりも範囲が広くなります)。次の CONCAT() 呼び出しを考えてみます。

```
CONCAT(_ucs2 0x0041, _ucs2 0x0042)
CONCAT(_ucs2 0x0041, _ucs2 0x00C2)
```

  最初の呼び出しでは、両方の引数が ascii 文字セットの範囲内にあるので、レパートリーは ASCII です。2 番目の呼び出しでは、2 番目の引数が ascii 文字セット範囲外にあるので、レパートリーは UNICODE です。

- 関数の戻り値のレパートリーは、結果の文字セットおよび照合順序に影響する引数のレパートリーにのみ基づいて定められます。

```
IF(column1 < column2, 'smaller', 'greater')
```

  結果のレパートリーは、2 つの文字列引数 (2 番目の引数と 3 番目の引数) がどちらも ASCII レパートリーなので、ASCII です。最初の引数は、式で文字列値を使用する場合でも、結果のレパートリーにとっては重要ではありません。

10.1.9 文字セットのサポートによる影響を受ける演算

このセクションでは、文字セット情報を考慮した演算について説明します。

10.1.9.1 結果文字列

MySQL には、文字列を返す多数の演算子と関数があります。このセクションでは、そのような文字列の文字セットと照合順序について説明します。

文字列の入力を取得して文字列の結果を出力として返す単純な関数では、出力の文字セットおよび照合順序は、主要な入力値の文字セットおよび照合順序と同じです。たとえば、UPPER(X) は、文字セットおよび照合順序が X と同じ文字列を返します。同じことは、INSTR()、LCASE()、LOWER()、LTRIM()、MID()、REPEAT()、REPLACE()、REVERSE()、RIGHT()、RPAD()、RTR および UPPER() についても当てはまります。

注: REPLACE() 関数はほかのどの関数とも異なり、文字列入力の照合順序を無視し、大文字と小文字が区別される比較を毎回実行します。

入力文字列または関数の結果がバイナリ文字列の場合、その文字列には文字セットも照合順序もありません。これは CHARSET() と COLLATION() 関数を使ってチェックできます。両関数とも、引数がバイナリ文字列であることを示す binary を返します。

```
mysql> SELECT CHARSET(BINARY 'a'), COLLATION(BINARY 'a');
+--------------------+----------------------+
| CHARSET(BINARY 'a') | COLLATION(BINARY 'a') |
+--------------------+----------------------+
| binary             | binary               |
+--------------------+----------------------+
```

複数の文字列入力を組み合わせて単一の文字列出力を返す演算では、結果の照合順序の特定に次の標準 SQL の「アグリゲーションルール」が適用されます。

- 明示的な COLLATE X が行われた場合、X を使用します。
- 明示的な COLLATE X と COLLATE Y が行われた場合、エラーが発生します。
- 上記以外の場合ですべての照合順序が X であるときは、X を使用します。
- その他の場合、結果に照合順序はありません。

たとえば、CASE ... WHEN a THEN b WHEN b THEN c COLLATE X END と指定されている場合、結果の照合順序は X になります。同じことは、UNION、||、CONCAT()、ELT()、GREATEST()、IF()、および LEAST() にも当てはまります。

文字データに変換する演算の場合、この演算の結果得られる文字列の文字セットと照合順序は、character_set_connection と collation_connection システム変数値によって定義されています。このことは、CAST()、CONV()、FORMAT()、HEX()、および SPACE() にのみ当てはまります。

文字列関数で返される結果の文字セットまたは照合順序について不確かな場合は、CHARSET() または COLLATION() 関数を使用して調べることができます。

```
mysql> SELECT USER(), CHARSET(USER()), COLLATION(USER());
+---------------+-----------------+-------------------+
| USER()        | CHARSET(USER()) | COLLATION(USER()) |
+---------------+-----------------+-------------------+
| test@localhost | utf8            | utf8_general_ci   |
+---------------+-----------------+-------------------+
```

10.1.9.2 CONVERT() と CAST()

CONVERT() を使用すると、異なる文字セット間でデータを変換できます。構文は次のとおりです。

```
CONVERT(expr USING transcoding_name)
```

MySQL では、トランスコーディング名は対応する文字セット名と同じです。

例:

```
SELECT CONVERT(_latin1'Müller' USING utf8);
INSERT INTO utf8table (utf8column)
   SELECT CONVERT(latin1field USING utf8) FROM latin1table;
```

CONVERT(... USING ...) は、標準 SQL の仕様に基づき実装されています。

CAST() を使用し、文字列を別の文字セットに変換することもできます。構文は次のとおりです。

```
CAST(character_string AS character_data_type CHARACTER SET charset_name)
```

例:

```
SELECT CAST(_latin1'test' AS CHAR CHARACTER SET utf8);
```

CHARACTER SET を指定せずに CAST() を使用した場合、文字セットと照合順序は character_set_connection および collation_connection システム変数で定義されます。CHARACTER SET X を指定して CAST() を使用した場合、文字セットはX、照合順序は X のデフォルト照合順序になります。

CONVERT() または CAST() 呼び出し内では COLLATE 句を使用できませんが、呼び出し外では使用できます。たとえば、CAST(... COLLATE ...) は無効ですが、CAST(...) COLLATE ... は有効です。

```
SELECT CAST(_latin1'test' AS CHAR CHARACTER SET utf8) COLLATE utf8_bin;
```

10.1.9.3 SHOW ステートメントと INFORMATION_SCHEMA

複数の SHOW ステートメントから文字セットの追加情報が得られます。これらには SHOW CHARACTER SET、SHOW COLLATION、SHOW CREATE DATABASE、SHOW CREATE TABLE、および SHOW COLUMNS が含まれます。ここでは、これらのステートメントについて簡単に説明します。詳細は、セクション 13.7.5「SHOW 構文」を参照してください。

INFORMATION_SCHEMA には、SHOW ステートメントで表示されるものに類似した情報を含む複数のテーブルが含まれます。たとえば、CHARACTER_SETS および COLLATIONS テーブルには、SHOW CHARACTER SET および SHOW COLLATION で表示される情報が含まれます。第21章「INFORMATION_SCHEMA テーブル」を参照してください。

SHOW CHARACTER SET ステートメントは使用可能な文字セットをすべて表示します。一致する文字セット名を指定するには、オプションの LIKE 句を使用します。例:

```
mysql> SHOW CHARACTER SET LIKE 'latin%';
+---------+----------------------------+-------------------+--------+
| Charset | Description                | Default collation | Maxlen |
+---------+----------------------------+-------------------+--------+
| latin1  | cp1252 West European       | latin1_swedish_ci |      1 |
| latin2  | ISO 8859-2 Central European | latin2_general_ci |      1 |
| latin5  | ISO 8859-9 Turkish         | latin5_turkish_ci |      1 |
| latin7  | ISO 8859-13 Baltic         | latin7_general_ci |      1 |
+---------+----------------------------+-------------------+--------+
```

SHOW COLLATION からの出力には、使用可能なすべての文字セットが含まれます。一致する照合順序名を指定するには、オプションの LIKE 句を使用します。例:

```
mysql> SHOW COLLATION LIKE 'latin1%';
```

```
+------------------+---------+----+---------+----------+---------+
| Collation        | Charset | Id | Default | Compiled | Sortlen |
+------------------+---------+----+---------+----------+---------+
| latin1_german1_ci | latin1  |  5 |         |          |       0 |
| latin1_swedish_ci | latin1  |  8 | Yes     | Yes      |       0 |
| latin1_danish_ci  | latin1  | 15 |         |          |       0 |
| latin1_german2_ci | latin1  | 31 |         | Yes      |       2 |
| latin1_bin        | latin1  | 47 |         | Yes      |       0 |
| latin1_general_ci | latin1  | 48 |         |          |       0 |
| latin1_general_cs | latin1  | 49 |         |          |       0 |
| latin1_spanish_ci | latin1  | 94 |         |          |       0 |
+------------------+---------+----+---------+----------+---------+
```

SHOW CREATE DATABASE は、所定のデータベースを作成する CREATE DATABASE ステートメントを表示します。

```
mysql> SHOW CREATE DATABASE test;
+----------+-----------------------------------------------------------------+
| Database | Create Database                                                 |
+----------+-----------------------------------------------------------------+
| test     | CREATE DATABASE `test` /*!40100 DEFAULT CHARACTER SET latin1 */ |
+----------+-----------------------------------------------------------------+
```

COLLATE 句が表示されていなければ、文字セットのデフォルト照合順序が適用されます。

SHOW CREATE TABLE は類似していますが、所定のテーブルを作成する CREATE TABLE ステートメントを表示します。カラム定義は文字セット仕様を指定し、テーブルオプションは文字セット情報を含んでいます。

SHOW COLUMNS ステートメントは SHOW FULL COLUMNS として呼び出された場合、テーブルのカラムの照合順序を表示します。CHAR、VARCHAR、または TEXT データ型を含むカラムには照合順序があります。数値型およびほかの非文字型には照合順序はありません (Collation 値として NULL で示されます)。例:

```
mysql> SHOW FULL COLUMNS FROM person\G
*************************** 1. row ***************************
     Field: id
      Type: smallint(5) unsigned
 Collation: NULL
      Null: NO
       Key: PRI
   Default: NULL
     Extra: auto_increment
Privileges: select,insert,update,references
   Comment:
*************************** 2. row ***************************
     Field: name
      Type: char(60)
 Collation: latin1_swedish_ci
      Null: NO
       Key:
   Default:
     Extra:
Privileges: select,insert,update,references
   Comment:
```

文字セットは表示の中にはなく、照合順序名で示されます。

10.1.10 Unicode のサポート

Unicode サポートの (MySQL 4.1 での) 初期実装には、Unicode データを格納するための次の 2 つの文字セットが含まれていました。

- ucs2。文字ごとに 16 ビットを使用する Unicode 文字セットの UCS-2 エンコーディング。
- utf8。文字ごとに 1 から 3 バイトを使用する Unicode 文字セットの UTF-8 エンコーディング。

これらの 2 つの文字セットは、Unicode バージョン 3.0 の Basic Multilingual Plane (BMP) の文字をサポートします。BMP 文字には次の 3 つの特性があります。

- これらのコード値は 0 から 65535 (または U+0000 ..U+FFFF) です。
- これらは、ucs2 と同様に、16 ビット固定の単語でエンコードできます。
- これらは、utf8 と同様に、8、16、または 24 ビットでエンコードできます。

- 主要言語のほとんどすべての文字には、これらで十分です。

前述の文字セットではサポートされていない文字には、BMP の範囲外にある補助文字が含まれます。BMP の範囲外の文字は、REPLACEMENT CHARACTER と見なされ、Unicode 文字セットへの変換時に、'?' に変換されます。

MySQL 5.6 での Unicode のサポートには補助文字が含まれるので、範囲がより広いため、より多くのスペースを占める新しい文字セットが必要になります。次の表では、以前の Unicode サポートと現在の Unicode サポートの機能を簡単に比較しています。

MySQL 5.5 より前	MySQL 5.5 以降
すべての Unicode 3.0 文字	すべての Unicode 5.0 および 6.0 文字
補助文字なし	補助文字あり
ucs2 文字セット、BMP のみ	変更なし
3 バイトまでの utf8 文字セット、BMP のみ	変更なし
	4 バイトまでの新しい utf8mb4 文字セット、BMP または補助
	新しい utf16 文字セット、BMP または補助
	新しい utf16le 文字セット、BMP または補助 (5.6.1 以降)
	新しい utf32 文字セット、BMP または補助

これらの変更には上位互換性があります。新しい文字セットを使用する場合、アプリケーションの非互換性の問題が生じる可能性があります。セクション10.1.11「以前の Unicode サポートから現在の Unicode サポートへのアップグレード」を参照してください。さらにこのセクションでは、utf8 から (4 バイトの) utf8mb4 文字セットに変換する方法と、そのときに適用される可能性がある制約について説明します。

MySQL 5.6 は、次の Unicode 文字セットをサポートします。

- ucs2。文字ごとに 16 ビットを使用する Unicode 文字セットの UCS-2 エンコーディング。
- utf16。Unicode 文字セットの UTF-16 エンコーディング。ucs2 に似ていますが、補助文字用の拡張機能があります。
- utf16le。Unicode 文字セットの UTF-16LE エンコーディング。utf16 と似ていますが、ビッグエンディアンではなくリトルエンディアンです。
- utf32。文字ごとに 32 ビットを使用した Unicode 文字セットの UTF-32 エンコーディング。
- utf8。文字ごとに 1 から 3 バイトを使用する Unicode 文字セットの UTF-8 エンコーディング。
- utf8mb4。文字ごとに 1 から 4 バイトを使用した Unicode 文字セットの UTF-8 エンコーディング。

ucs2 および utf8 は BMP 文字をサポートします。utf8mb4、utf16、utf16le、および utf32 は、BMP および補助文字をサポートします。

類似した一連の照合順序を、ほとんどの Unicode 文字セットで使用できます。たとえば、それぞれにはデンマーク語の照合順序があり、その名前は ucs2_danish_ci、utf16_danish_ci、utf32_danish_ci、utf8_danish_ci、および utf8mb4_danish_ci です。例外は utf16le であり、2 つの照合順序しかありません。セクション10.1.14.1「Unicode 文字セット」にはすべての Unicode 照合順序が一覧表示されており、補助文字の照合順序プロパティーについても説明しています。

多くの補助文字は東アジアの言語のものですが、MySQL 5.6 で追加するサポートは、Unicode 文字セットでのより多くの日本語と中国語の文字に対するサポートであり、新しい日本語および中国語の文字セットのサポートではありません。

UCS-2、UTF-16、および UTF-32 の MySQL での実装は、ビッグエンディアンのバイト順で文字を格納し、値の先頭にバイト順マーク (BOM) を使用しません。ほかのデータベースシステムでは、リトルエンディアンのバイト順または BOM を使用していることもあります。このような場合、これらのシステムと MySQL の間でデータを転送するときに、値の変換を実行する必要があります。UTF-16LE の実装はリトルエンディアンです。

MySQL は UTF-8 値に BOM を使用しません。

Unicode を使用したサーバーと通信する必要があるクライアントアプリケーションは、たとえば SET NAMES 'utf8' ステートメントを発行することによって、クライアント文字セットをそれに従って設定する必要があります。ucs2、utf16、utf16le、および utf32 は、クライアント文字セットとして使用できません。つまり、これらは SET NAMES または SET CHARACTER SET で機能しません。(セクション10.1.4「接続文字セットおよび照合順序」を参照してください。)

以降のセクションでは、MySQL における Unicode 文字セットについてさらに詳しく説明します。

10.1.10.1 ucs2 文字セット (UCS-2 Unicode エンコーディング)

UCS-2 では、すべての文字が 2 バイトの Unicode コードで表され、もっとも重要なバイトは 1 番目のバイトです。例: LATIN CAPITAL LETTER A にはコード 0x0041 があり、これは 2 バイトのシーケンス (0x00 0x41) として格納されます。CYRILLIC SMALL LETTER YERU (Unicode 0x044B) は、2 バイトシーケンス (0x04 0x4B) として格納されます。Unicode 文字とそのコードについては、Unicode ホームページを参照してください。

MySQL では、ucs2 文字セットは、Unicode BMP 文字の 16 ビット固定長のエンコーディングです。

10.1.10.2 utf16 文字セット (UTF-16 Unicode エンコーディング)

utf16 文字セットは、補助文字のエンコーディングを可能にする拡張機能を備えた ucs2 文字セットです。

- BMP 文字の場合、utf16 と ucs2 には、同じコード値、同じエンコーディング、同じ長さという同一のストレージ特性があります。
- 補助文字の場合、utf16 には、32 ビットを使用する文字を表すための特殊シーケンスがあります。これは、「サロゲート」メカニズムと呼ばれます。0xffff より大きな数値の場合、10 ビットを確保して、これらを 0xd800 に追加し、最初の 16 ビット語に配置します。さらに 10 ビットを確保して、これらを 0xdc00 に追加し、次の 16 ビット語に配置します。この結果、すべての補助文字に 32 ビットが必要になります。このうち最初の 16 ビットは 0xd800 と 0xdbff の間の数値であり、残りの 16 ビットは 0xdc00 と 0xdfff の間の数値になります。Unicode 4.0 ドキュメントの「15.5 Surrogates Area」に例が記載されています。

utf16 はサロゲートをサポートし、ucs2 をサポートしていないので、utf16 でのみ適用される妥当性チェックがあります。下位サロゲートがなければ上位サロゲートを挿入できず、逆も同様です。例:

```
INSERT INTO t (ucs2_column) VALUES (0xd800); /* legal */
INSERT INTO t (utf16_column)VALUES (0xd800); /* illegal */
```

技術的に有効であるが真の Unicode ではない文字 (つまり、Unicode が「未割り当てコードポイント」または「個人使用」文字、さらには 0xffff のように「不正」と見なす文字) に対する妥当性チェックはありません。たとえば、U+F8FF は Apple の口ゴなので、これは有効です。

```
INSERT INTO t (utf16_column)VALUES (0xf8ff); /* legal */
```

このような文字は、すべてのユーザーに対し同じ意味を持たせることは期待できません。

MySQL は、最悪の場合 (文字が 4 バイトを必要とする場合) に対応する必要があるので、utf16 カラムまたはインデックスの最大長は、ucs2 カラムまたはインデックスの最大長の半分しかありません。たとえば、MySQL 5.6 では、MEMORY テーブルインデックスキーの最大長は、3072 バイトなので、これらのステートメントは、ucs2 および utf16 カラムに対し、最大許容長のインデックスを持つテーブルを作成します。

```
CREATE TABLE tf (s1 VARCHAR(1536) CHARACTER SET ucs2) ENGINE=MEMORY;
CREATE INDEX i ON tf (s1);
CREATE TABLE tg (s1 VARCHAR(768) CHARACTER SET utf16) ENGINE=MEMORY;
CREATE INDEX i ON tg (s1);
```

10.1.10.3 utf16le 文字セット (UTF-16LE Unicode エンコーディング)

これは、utf16 と同じですが、ビッグエンディアンではなくリトルエンディアンです。

10.1.10.4 utf32 文字セット (UTF-32 Unicode エンコーディング)

utf32 文字セットは固定長です (ucs2 と同様で、utf16 とは異なります)。utf32 はすべての文字に 32 ビットを使用し、ucs2 (すべての文字に 16 ビットを使用します) とも、utf16 (一部の文字に 16 ビットを、ほかの文字に 32 ビットを使用します) とも異なります。

utf32 は、ucs2 の 2 倍のスペース、utf16 よりも多くのスペースを必要としますが、utf32 には、ストレージについて予測可能であるという ucs2 と同じ利点があります。utf32 に必要なバイト数は文字数の 4 倍になります。ま

た、utf16 とは異なり、utf32 でのエンコーディングにはトリックはないので、格納された値はコード値と等しくなります。

後者の利点がどのように役立つかを説明するために、utf32 コード値のときに utf8mb4 値を求める方法を示した例を挙げます。

```
/* Assume code value = 100cc LINEAR B WHEELED CHARIOT */
CREATE TABLE tmp (utf32_col CHAR(1) CHARACTER SET utf32,
                  utf8mb4_col CHAR(1) CHARACTER SET utf8mb4);
INSERT INTO tmp VALUES (0x000100cc,NULL);
UPDATE tmp SET utf8mb4_col = utf32_col;
SELECT HEX(utf32_col),HEX(utf8mb4_col) FROM tmp;
```

MySQL では、未割り当ての Unicode 文字または個人使用領域の文字の追加について広く許容しています。実際、utf32 の妥当性チェックは 1 つしかありません。0x10ffff よりも大きなコード値はありません。たとえば次の場合は不正です。

```
INSERT INTO t (utf32_column) VALUES (0x110000); /* illegal */
```

10.1.10.5 utf8 文字セット (3 バイト UTF-8 Unicode エンコーディング)

UTF-8 (8 ビット単位の Unicode Transformation Format) は Unicode データを格納する別の方法です。これは、1 から 4 バイトを使用するエンコーディングシーケンスについて記述した RFC 3629 に従って実装されています。(UTF-8 エンコーディングの以前の標準である RFC 2279 では、1 から 6 バイトを使用する UTF-8 シーケンスについて記述しています。RFC 3629 は RFC 2279 を無効にするため、5 と 6 バイトのシーケンスはすでに使用されていません。)

UTF-8 の概念は、長さの異なるバイトシーケンスを使用してさまざまな Unicode 文字をエンコードするというものです。

- 基本的なラテン文字、数字、句読点は 1 バイトを使用します。
- 拡張ラテン文字 (チルダ、長音符号、アキュート、グラーヴェ、およびほかのアクセント符号)、キリル文字、ギリシャ語、アルメニア語、ヘブライ語、アラビア語、シリア語などのほとんどのヨーロッパおよび中東のスクリプト文字は、2 バイトシーケンスに収まります。
- 韓国語、中国語、および日本語の表意文字は、3 バイトまたは 4 バイトのシーケンスを使用します。

MySQL 5.6 の utf8 文字セットは 5.6 以前と同じであり、特性もまったく同じです。

- 補助文字のサポートなし (BMP 文字のみ)。
- マルチバイト文字ごとに最大 3 バイト。

utf8 では ucs2 とちょうど同じ文字セットを使用できます。つまり、レパートリーも同じです。

ヒント: スペースを UTF-8 で保存する場合は、CHAR ではなく VARCHAR を使用してください。そのようにしないと、MySQL では CHAR CHARACTER SET utf8 カラムに対して 3 バイトを確保する必要があります。これは、可能性のある最大長が 3 バイトであるためです。たとえば、MySQL は CHAR(10) CHARACTER SET utf8 カラムに対して 30 バイトを確保する必要があります。

データ型のストレージの詳細は、セクション11.7「データ型のストレージ要件」を参照してください。COMPACT 行フォーマットを使用する InnoDB テーブルが UTF-8 CHAR(N) カラムを内部で処理する方法を含め、InnoDB 物理行ストレージの詳細は、セクション14.2.13.7「物理的な行構造」を参照してください。

10.1.10.6 utf8mb3 文字セット (utf8 のエイリアス)

MySQL の今後のバージョンでは、utf8 が 4 バイトの utf8 になり、3 バイトの utf8 を指定するときに utf8mb3 を示す必要が生じる可能性があります。マスターサーバーとスレーブサーバーの MySQL のバージョンが異なるときにレプリケーションで生じる将来の問題を回避するために、ユーザーは、CHARACTER SET 句で utf8mb3 を、COLLATE 句で utf8mb3_collation_substring を指定できます。ここで、collation_substring は bin、czech_ci、danish_ci、esperanto_ci、estonian_ci などです。例:

```
CREATE TABLE t (s1 CHAR(1) CHARACTER SET utf8mb3;
SELECT * FROM t WHERE s1 COLLATE utf8mb3_general_ci = 'x';
DECLARE x VARCHAR(5) CHARACTER SET utf8mb3 COLLATE utf8mb3_danish_ci;
SELECT CAST('a' AS CHAR CHARACTER SET utf8) COLLATE utf8_czech_ci;
```

MySQL は即座に、エイリアス内の utf8mb3 のインスタンスを、utf8 に変換するので、SHOW CREATE TABLE、SELECT CHARACTER_SET_NAME FROM INFORMATION_SCHEMA.COLUMNS、SELECT COLLATION_NAME FROM INFORMATION_SCHEMA.COLUMNS などのステートメントでは、真の名前、utf8、または utf8_collation_substring がユーザーに表示されます。

utf8mb3 エイリアスは、CHARACTER SET 句と、ほかの特定の場所でのみ有効です。たとえば、次の場合は有効です。

```
mysqld --character-set-server=utf8mb3
SET NAMES 'utf8mb3'; /* and other SET statements that have similar effect */
SELECT _utf8mb3 'a';
```

照合順序が基づいている Unicode 照合アルゴリズムのバージョンを示すバージョン番号を含む照合順序名 (utf8_unicode_520_ci など) を表す、対応する utf8 照合順序の utf8mb3 エイリアスはありません。

10.1.10.7 utf8mb4 文字セット (4 バイトの UTF-8 Unicode エンコーディング)

utf8 という名前の文字セットは、文字あたり最大 3 バイトを使用し、BMP 文字だけを含みます。utf8mb4 文字セットは、文字ごとに最大 4 バイトを使用し、補助文字をサポートします。

- BMP 文字の場合、utf8 と utf8mb4 のストレージ特性は同一で、コード値、エンコーディング、長さが同じです。
- 補助文字については、utf8 はこの文字をまったく格納できませんが、utf8mb4 は文字の格納に 4 バイトを必要とします。utf8 はこの文字をまったく格納しないので、utf8 カラムには補助文字がなく、utf8 データを古いバージョンの MySQL からアップグレードするときに、文字の変換やデータの損失について心配する必要はありません。

utf8mb4 は utf8 のスーパーセットであるので、次の連結のような演算の場合、その結果には utf8mb4 の文字セットと utf8mb4_col の照合順序が含まれます。

```
SELECT CONCAT(utf8_col, utf8mb4_col);
```

同様に、次の WHERE 句内の比較は、utf8mb4_col の照合順序に従って行われます。

```
SELECT * FROM utf8_tbl, utf8mb4_tbl
WHERE utf8_tbl.utf8_col = utf8mb4_tbl.utf8mb4_col;
```

ヒント: スペースを UTF-8 で保存する場合は、CHAR ではなく VARCHAR を使用してください。そのようにしないと、MySQL では CHAR CHARACTER SET utf8 (または utf8mb4) カラムに対して 3 (または 4) バイトを確保する必要があります。これは、可能性のある最大長が 3 (または 4) バイトであるためです。たとえば、MySQL は CHAR(10) CHARACTER SET utf8mb4 カラムに対して 40 バイトを確保する必要があります。

10.1.11 以前の Unicode サポートから現在の Unicode サポートへのアップグレード

このセクションでは、古い MySQL リリースから MySQL 5.6 にアップグレードするときに起きる可能性のある、Unicode サポートに関する問題について説明します。また、MySQL 5.6 から古いリリースにダウングレードするためのガイドラインも示します。

ほとんどの点で MySQL 5.6 にアップグレードしても、Unicode の使用法について問題が生じることはほとんどありませんが、非互換性の可能性のあるいくつかの領域があります。主な対象領域は次のとおりです。

- 可変長文字データ型 (VARCHAR 型と TEXT 型) の場合、文字の最大長は、utf8mb4 カラムのほうが utf8 カラムよりも短くなります。
- すべての文字データ型 (CHAR 型、VARCHAR 型、および TEXT 型) で、インデックスを付けることができる文字の最大数は、utf8mb4 カラムのほうが utf8 カラムよりも少なくなります。

このため、utf8 から utf8mb4 にテーブルをアップグレードして補助文字サポートを利用する場合、一部のカラムまたはインデックス定義を変更する必要が生じることがあります。

テーブルは、ALTER TABLE を使用することにより、utf8 から utf8mb4 に変換できます。テーブルがもともと次のように定義されていたとします。

```
CREATE TABLE t1 (
  col1 CHAR(10) CHARACTER SET utf8 COLLATE utf8_unicode_ci NOT NULL,
```

```
  col2 CHAR(10) CHARACTER SET utf8 COLLATE utf8_bin NOT NULL
) CHARACTER SET utf8;
```

次のステートメントは、utf8mb4 を使用するように t1 を変換します。

```
ALTER TABLE t1
  DEFAULT CHARACTER SET utf8mb4,
  MODIFY col1 CHAR(10)
    CHARACTER SET utf8mb4 COLLATE utf8mb4_unicode_ci NOT NULL,
  MODIFY col2 CHAR(10)
    CHARACTER SET utf8mb4 COLLATE utf8mb4_bin NOT NULL;
```

テーブルの内容については、utf8 から utf8mb4 への変換で問題は起こりません。

- BMP 文字の場合、utf8 と utf8mb4 のストレージ特性は同一で、コード値、エンコーディング、長さが同じです。
- 補助文字については、utf8 はこの文字をまったく格納できませんが、utf8mb4 は文字の格納に 4 バイトを必要とします。utf8 はこの文字をまったく格納しないので、utf8 カラムには補助文字がなく、utf8 データを古いバージョンの MySQL からアップグレードするときに、文字の変換やデータの損失について心配する必要はありません。

テーブル構造については、utf8 から utf8mb4 への変換時に、カラムまたはインデックスキーの最大長がバイトの点では変更されないという問題が生じます。そのため、文字の最大長が 3 ではなく 4 なので、文字の点ではこれはより小さくなります。CHAR、VARCHAR、および TEXT データ型の場合、MySQL テーブルを変換するときに、次の点に注意してください。

- utf8 カラムのすべての定義を調べて、それらがストレージエンジンの最大長を超えていないことを確認します。
- utf8 カラムのすべてのインデックスを調べて、それらがストレージエンジンの最大長を超えていないことを確認します。最大値は、ストレージエンジンの機能強化によって変更されることがあります。

前述の条件が当てはまる場合は、カラムまたはインデックスの定義された長さを減らすか、utf8mb4 ではなく utf8 を使用し続ける必要があります。

次に、構造的な変更が必要な例をいくつか示します。

- TINYTEXT カラムは、最大 255 バイトを保持できるので、3 バイトの文字は 85 個まで、4 バイトの文字は 63 個まで保持できます。utf8 を使用する TINYTEXT カラムがあるが、63 個以上の文字を含められることが必要だとします。データ型も TEXT などのより長い型に変更しないかぎり、これを utf8mb4 に変換できません。

  同様に非常に長い VARCHAR カラムは、utf8 から utf8mb4 に変換する場合は、より長い TEXT 型のいずれかに変更する必要があります。

- InnoDB には、COMPACT または REDUNDANT 行フォーマットを使用するテーブル用に 767 バイトの最大インデックス長があり、utf8 または utf8mb4 カラムの場合、それぞれ最大 255 または 191 個の文字にインデックスを付けることができます。現在、インデックスが 191 個の文字より長い utf8 カラムがある場合、インデックスを付ける文字数を減らす必要があります。

  COMPACT または REDUNDANT 行フォーマットを使用する InnoDB テーブルでは、次のカラムおよびインデックス定義が有効です。

  ```
  col1 VARCHAR(500) CHARACTER SET utf8, INDEX (col1(255))
  ```

  代わりに utf8mb4 を使用するには、インデックスをより小さくする必要があります。

  ```
  col1 VARCHAR(500) CHARACTER SET utf8mb4, INDEX (col1(191))
  ```

> **注記**
>
> COMPRESSED または DYNAMIC 行フォーマットを使用する InnoDB テーブルの場合、innodb_large_prefix オプションを有効にすると、767 バイトより長いインデックスキープリフィクス (最大 3072 バイト) を許容できます。このようなテーブルの作成には、innodb_file_format=barracuda および innodb_file_per_table=true のオプション値も必要になります。)この場合、innodb_large_prefix オプションを有効にすると、utf8 または utf8mb4 カラムに対しそれぞれ最大 1024 または 768 個の文字にインデックスを付けることができます。関連情報については、セクション14.6.7「InnoDB テーブル上の制限」を参照してください。

前述の変更のタイプは、非常に長いカラムまたはインデックスを持つ場合にのみ必要になる可能性が高くなります。それ以外の場合は、utf8 から utf8mb4 に問題なくテーブルを変換できます。これを行うには、5.6 に適切にアップグレードしたあとで、このセクションですでに述べたように ALTER TABLE を使用します。

次の項目は、非互換性の可能性のあるほかの領域についてまとめたものです。

- 4 バイトの UTF-8 (utf8mb4) のパフォーマンスは、3 バイトの UTF-8 (utf8) のパフォーマンスより低下します。このペナルティーを望まない場合は、utf8 を引き続き使用してください。
- SET NAMES 'utf8mb4' では、接続文字セットに対して 4 バイトの文字セットを使用する必要があります。4 バイト文字がサーバーから送信されていないかぎり、問題は起こりません。それ以外の場合は、文字ごとに最大 3 バイトの受信を要求するアプリケーションで問題が発生する可能性があります。反対に、4 バイトの文字の送信を要求するアプリケーションは、その文字がサーバーで認識されることを確認する必要があります。
- アプリケーションは、utf16、utf16le、または utf32 文字データを認識しない古いサーバーにこれらのデータを送信できません。
- レプリケーションでは、補助文字をサポートする文字セットがマスターで使用される場合、すべてのスレーブでもこれらの文字を認識する必要があります。MySQL 5.6 マスターから古いスレーブに複製しようとしたときに、utf8 データはスレーブで utf8 と認識され、正しく複製されます。ただし、utf8mb4、utf16、utf16le、または utf32 データは送信できません。

  また、テーブルにマスターとスレーブについて異なる定義がある場合、予想外の結果を招くことがあるという一般的な原則に留意してください。たとえば、インデックスキー長の制限が異なると、マスターで utf8 を使用し、スレーブで utf8mb4 を使用することは危険です。

MySQL 5.6 にアップグレードしてあり、古いリリースにダウングレードすることにした場合、次の考慮事項が該当します。

- ucs2 および utf8 データで問題が生じていない必要があります。
- utf8mb4、utf16、utf16le、または utf32 文字セットを参照するすべての定義は、古いサーバーで認識されません。
- utf8mb4 文字セットを参照するオブジェクト定義の場合、データ内に 4 バイト文字が存在しないかぎり、MySQL 5.6 で mysqldump を使用してこれらをダンプし、utf8mb4 のインスタンスを utf8 に変更するようにダンプファイルを編集し、このファイルを古いサーバーにリロードできます。古いサーバーは、ダンプファイルオブジェクト定義内の utf8 を認識し、(3 バイトの) utf8 文字セットを使用する新しいオブジェクトを作成します。

10.1.12 メタデータ用の UTF-8

メタデータは「データに関するデータ」です。データベースについて記述しているすべてのものがメタデータであり、データベースの内容ではありません。したがって、カラム名、データベース名、ユーザー名、バージョン名、および SHOW の文字列結果のほとんどがメタデータです。INFORMATION_SCHEMA 内のテーブルは定義上、データベースオブジェクトに関する情報を含んでいるので、これは、このテーブルの内容にも当てはまります。

メタデータの表現は次の要件を満たしている必要があります。

- すべてのメタデータで文字セットが一致している必要があります。それ以外の場合、INFORMATION_SCHEMA 内のテーブルに対する SHOW ステートメントも SELECT ステートメントも正しく機能しません。これらの演算結果の同一カラム内の各行で文字セットが異なるからです。
- メタデータはすべての言語のすべての文字が含まれている必要があります。そうでない場合、ユーザーはそれぞれの言語を使用してカラムとテーブルに名前を付けることはできません。

両方の要件を満たすために、MySQL では、Unicode 文字セット、つまり UTF-8 でメタデータを格納します。アクセント符号付きの文字またはラテン語以外の文字を使用しなければ、混乱が生じることはありません。ただし、使用した場合は、メタデータの文字セットが UTF-8 であることを認識する必要があります。

このメタデータ要件は、USER()、CURRENT_USER()、SESSION_USER()、SYSTEM_USER()、DATABASE()、および VERSION() の関数の戻り値で、UTF-8 文字セットがデフォルトで使用されることを意味します。

サーバーは、character_set_system システム変数をメタデータ文字セットの名前に設定します。

```
mysql> SHOW VARIABLES LIKE 'character_set_system';
+----------------------+-------+
```

```
| Variable_name        | Value |
+----------------------+-------+
| character_set_system | utf8  |
+----------------------+-------+
```

Unicode を使用してメタデータを格納しても、サーバーが、カラムのヘッダーや DESCRIBE 関数の結果を、デフォルトで character_set_system 文字セットで返すことにはなりません。SELECT column1 FROM t を使用すると、character_set_results システム変数の値 (デフォルト値は latin1) で特定される文字セットで、名前 column1 自体がサーバーからクライアントに返されます。別の文字セットでメタデータの結果をサーバーに返させる場合は、SET NAMES ステートメントを使用してサーバーに文字セット変換を強制的に実行させてください。SET NAMES は character_set_results および関連するほかのシステム変数を設定します。(セクション10.1.4「接続文字セットおよび照合順序」を参照してください。)また、サーバーから結果を受け取ったあとで、クライアントプログラムが変換を実行できます。クライアントが変換を実行するとより効率的ですが、このオプションは、すべてのクライアントが常に使用できるとはかぎりません。

character_set_results が NULL に設定されている場合、変換は実行されず、サーバーはオリジナルの文字セット (character_set_system によって指定されたセット) を使用してメタデータを返します。

サーバーからクライアントに返されるエラーメッセージは、メタデータと同様に自動的にクライアントの文字セットに変換されます。

たとえば、USER() 関数を比較または割り当てのために単一のステートメント内で使用している場合、問題はありません。MySQL が自動的に変換を実行します。

```
SELECT * FROM t1 WHERE USER() = latin1_column;
```

これが機能するのは、latin1_column の内容が UTF-8 に自動的に変換されてから比較が行われるからです。

```
INSERT INTO t1 (latin1_column) SELECT USER();
```

これが機能するのは、USER() の内容が latin1 に自動的に変換されてから割り当てが行われるからです。

自動変換機能は SQL 標準には含まれていません。ただし、どの文字セットも (サポートされている文字に関して) Unicode の「サブセット」であることが SQL 標準のドキュメントに記載されています。「スーパーセットに適用されるものはサブセットにも適用される」というよく知られた原則があるので、Unicode の照合順序は Unicode 以外の文字列との比較にも適用できると考えられます。文字列の強制力の詳細は、セクション10.1.7.5「式の照合順序」を参照してください。

10.1.13 カラム文字セットの変換

特定の文字セットを使用するようにバイナリ文字列または非バイナリ文字列カラムを変換するには、ALTER TABLE を使用します。正しく変換が行われるには、次の条件のいずれかを適用する必要があります。

- カラムにバイナリデータ型 (BINARY、VARBINARY、BLOB) がある場合、含まれるすべての値は、単一の文字セット (カラムの変換後の文字セット) を使用してエンコードされる必要があります。バイナリカラムを使用して複数の文字セットで情報を格納する場合、MySQL はどの値がどの文字セットを使用するかを認識できず、データを正確に変換できません。
- カラムに非バイナリデータ型 (CHAR、VARCHAR、TEXT) がある場合、その内容は、カラムの文字セットでエンコードする必要があり、ほかの文字セットは使用できません。内容が別の文字セットでエンコードされている場合、最初にバイナリデータ型を使用するようにカラムを変換してから、使用する文字セットで非バイナリカラムに変換できます。

VARBINARY(50) として定義された col1 という名前のテーブル t にバイナリカラムがあるとします。カラム内の情報が、単一の文字セットを使用してエンコードされているとすると、このカラムを、その文字セットを含む非バイナリカラムに変換できます。たとえば、col1 に greek 文字セットの文字を表すバイナリデータが含まれる場合、次のように変換できます。

```
ALTER TABLE t MODIFY col1 VARCHAR(50) CHARACTER SET greek;
```

元のカラムに BINARY(50) の型がある場合、これを CHAR(50) に変換できますが、その結果得られる値は、末尾が 0x00 バイトでパディングされ、これが望ましくない場合があります。これらのバイトを削除するには、TRIM() 関数を使用します。

```
UPDATE t SET col1 = TRIM(TRAILING 0x00 FROM col1);
```

テーブル t に CHAR(50) CHARACTER SET latin1 として定義された col1 という名前の非バイナリ列があるが、utf8 を使用するようにこれを変換し、多くの言語の値を格納できるようにするとします。次のステートメントでこれを実行できます。

```
ALTER TABLE t MODIFY col1 CHAR(50) CHARACTER SET utf8;
```

両方の文字セットにない文字がカラムに含まれている場合、変換の損失が大きくなることがあります。

MySQL 4.1 より前の古いテーブルがある場合、実際にはサーバーのデフォルトの文字セットと異なる文字セットでエンコードされた値が、非バイナリカラムに含まれるという特殊な状況が起こります。たとえば、MySQL のデフォルト文字セットが latin1 であっても、アプリケーションは sjis 値をカラムに格納します。適切な文字セットを使用するために、カラムを変換することは可能ですが、追加ステップが必要になります。たとえば、サーバーのデフォルト文字セットが latin1 で、col1 は CHAR(50) と定義されているにもかかわらず、内容は sjis 値であるとします。最初のステップでは、バイナリデータ型にカラムを変換することで、文字変換を実行しないで既存の文字セット情報を取り除きます。

```
ALTER TABLE t MODIFY col1 BLOB;
```

次のステップでは、適切な文字セットを使用して、非バイナリデータ型にカラムを変換します。

```
ALTER TABLE t MODIFY col1 CHAR(50) CHARACTER SET sjis;
```

この手順では、MySQL 4.1 以降にアップグレードしたあと、テーブルが INSERT や UPDATE などのステートメントで変更されていない必要があります。その場合、MySQL は latin1 を使用してカラムに新しい値を格納し、そのカラムは sjis 値と latin1 値を同時に含んでおり、正確に変換できません。

最初にカラムを作成するときに属性を指定した場合、ALTER TABLE を使用してテーブルを変更しているときにも属性を指定する必要があります。たとえば、NOT NULL と明示的な DEFAULT 値を指定した場合、ALTER TABLE ステートメントでも指定する必要があります。指定しない場合、結果のカラム定義にはこれらの属性が含まれません。

テーブル内のすべての文字カラムを変換するには、ALTER TABLE ... CONVERT TO CHARACTER SET charset ステートメントが役立つことがあります。セクション13.1.7「ALTER TABLE 構文」を参照してください。

10.1.14 MySQL でサポートされる文字セットと照合順序

MySQL では、30 を超える文字セットに対して 70 を超える照合順序がサポートされています。このセクションでは MySQL がどの文字セットをサポートするかを示します。関連する文字セットのグループごとに、1 つのサブセクションがあります。文字セットごとに、許容される照合順序が一覧表示されます。

SHOW CHARACTER SET ステートメント使用すると、使用可能な文字セットとそのデフォルト照合順序を常に表示できます。

```
mysql> SHOW CHARACTER SET;
+----------+-----------------------------+---------------------+
| Charset  | Description                 | Default collation   |
+----------+-----------------------------+---------------------+
| big5     | Big5 Traditional Chinese    | big5_chinese_ci     |
| dec8     | DEC West European           | dec8_swedish_ci     |
| cp850    | DOS West European           | cp850_general_ci    |
| hp8      | HP West European            | hp8_english_ci      |
| koi8r    | KOI8-R Relcom Russian       | koi8r_general_ci    |
| latin1   | cp1252 West European        | latin1_swedish_ci   |
| latin2   | ISO 8859-2 Central European | latin2_general_ci   |
| swe7     | 7bit Swedish                | swe7_swedish_ci     |
| ascii    | US ASCII                    | ascii_general_ci    |
| ujis     | EUC-JP Japanese             | ujis_japanese_ci    |
| sjis     | Shift-JIS Japanese          | sjis_japanese_ci    |
| hebrew   | ISO 8859-8 Hebrew           | hebrew_general_ci   |
| tis620   | TIS620 Thai                 | tis620_thai_ci      |
| euckr    | EUC-KR Korean               | euckr_korean_ci     |
| koi8u    | KOI8-U Ukrainian            | koi8u_general_ci    |
| gb2312   | GB2312 Simplified Chinese   | gb2312_chinese_ci   |
| greek    | ISO 8859-7 Greek            | greek_general_ci    |
| cp1250   | Windows Central European    | cp1250_general_ci   |
| gbk      | GBK Simplified Chinese      | gbk_chinese_ci      |
| latin5   | ISO 8859-9 Turkish          | latin5_turkish_ci   |
| armscii8 | ARMSCII-8 Armenian          | armscii8_general_ci |
| utf8     | UTF-8 Unicode               | utf8_general_ci     |
| ucs2     | UCS-2 Unicode               | ucs2_general_ci     |
| cp866    | DOS Russian                 | cp866_general_ci    |
| keybcs2  | DOS Kamenicky Czech-Slovak  | keybcs2_general_ci  |
| macce    | Mac Central European        | macce_general_ci    |
| macroman | Mac West European           | macroman_general_ci |
| cp852    | DOS Central European        | cp852_general_ci    |
| latin7   | ISO 8859-13 Baltic          | latin7_general_ci   |
| utf8mb4  | UTF-8 Unicode               | utf8mb4_general_ci  |
```

```
| cp1251   | Windows Cyrillic          | cp1251_general_ci   |
| utf16    | UTF-16 Unicode            | utf16_general_ci    |
| utf16le  | UTF-16LE Unicode          | utf16le_general_ci  |
| cp1256   | Windows Arabic            | cp1256_general_ci   |
| cp1257   | Windows Baltic            | cp1257_general_ci   |
| utf32    | UTF-32 Unicode            | utf32_general_ci    |
| binary   | Binary pseudo charset     | binary              |
| geostd8  | GEOSTD8 Georgian          | geostd8_general_ci  |
| cp932    | SJIS for Windows Japanese | cp932_japanese_ci   |
| eucjpms  | UJIS for Windows Japanese | eucjpms_japanese_ci |
+----------+---------------------------+---------------------+
```

文字セットに複数の照合順序が存在する場合、どの照合順序が所定のアプリケーションにもっとも適しているかが明確でないことがあります。正しくない照合順序を選択しないようにするには、代表的なデータ値で比較を行い、特定の照合順序で期待どおりに値がソートされることを確認すると役立ちます。

10.1.14.1 Unicode 文字セット

MySQL 5.6 は、次の Unicode 文字セットをサポートします。

- ucs2。文字ごとに 16 ビットを使用する Unicode 文字セットの UCS-2 エンコーディング。
- utf16。Unicode 文字セットの UTF-16 エンコーディング。ucs2 に似ていますが、補助文字用の拡張機能があります。
- utf16le。Unicode 文字セットの UTF-16LE エンコーディング。utf16 と似ていますが、ビッグエンディアンではなくリトルエンディアンです。
- utf32。文字ごとに 32 ビットを使用した Unicode 文字セットの UTF-32 エンコーディング。
- utf8。文字ごとに 1 から 3 バイトを使用する Unicode 文字セットの UTF-8 エンコーディング。
- utf8mb4。文字ごとに 1 から 4 バイトを使用した Unicode 文字セットの UTF-8 エンコーディング。

ucs2 および utf8 は、Basic Multilingual Plane (BMP) 文字をサポートします。utf8mb4、utf16、utf16le、および utf32 は BMP と補助文字をサポートします。utf16le は MySQL 5.6.1 で追加されました。

これらの文字セットを使用すると、約 650 の言語でテキストを格納できます。このセクションでは、Unicode 文字セットごとに利用可能な照合順序を示し、それらを区別するプロパティーについて説明します。文字セットの一般的な情報については、セクション10.1.10「Unicode のサポート」を参照してください。

類似した一連の照合順序を、ほとんどの Unicode 文字セットで使用できます。これらは次のリストに表示されます。ここで xxx は文字セット名を表します。たとえば、xxx_danish_ci はデンマーク語の照合順序を表し、具体的には ucs2_danish_ci、utf16_danish_ci、utf32_danish_ci、utf8_danish_ci、および utf8mb4_danish_ci の名前を構成します。

utf16le に対する照合順序のサポートはさらに限定されます。使用可能な唯一の照合順序は、utf16le_general_ci と utf16le_bin です。これらは、utf16_general_ci と utf16_bin に似ています。

このセクションで後述するように、Unicode 照合順序名には、照合順序が基づいている Unicode 照合順序アルゴリズムのバージョンを示すバージョン番号も含まれる場合があります (たとえば xxx_unicode_520_ci)。このような照合順序の場合、対応する utf8 照合順序の utf8mb3 エイリアスはありません。セクション10.1.10.6「utf8mb3 文字セット (utf8 のエイリアス)」を参照してください。

- xxx_bin
- xxx_croatian_ci
- xxx_czech_ci
- xxx_danish_ci
- xxx_esperanto_ci
- xxx_estonian_ci
- xxx_general_ci (デフォルト)
- xxx_german2_ci
- xxx_general_mysql500_ci

- xxx_hungarian_ci
- xxx_icelandic_ci
- xxx_latvian_ci
- xxx_lithuanian_ci
- xxx_persian_ci
- xxx_polish_ci
- xxx_roman_ci
- xxx_romanian_ci
- xxx_sinhala_ci
- xxx_slovak_ci
- xxx_slovenian_ci
- xxx_spanish_ci
- xxx_spanish2_ci
- xxx_swedish_ci
- xxx_turkish_ci
- xxx_unicode_ci
- xxx_vietnamese_ci

MySQL は、http://www.unicode.org/reports/tr10/ で説明している Unicode 照合順序アルゴリズム (UCA) に従って xxx_unicode_ci 照合順序を実装します。照合順序は、バージョン 4.0.0 UCA 重みキー (http://www.unicode.org/Public/UCA/4.0.0/allkeys-4.0.0.txt) を使用します。現在、xxx_unicode_ci 照合順序は、Unicode 照合順序アルゴリズムを一部だけサポートします。中にはまだサポートされていない文字もあります。また、結合マークは完全にはサポートされていません。これは主に、ベトナム語、ヨルバ語、ナバホ語などの一部のより小さな言語に影響します。組み合わされた文字は、文字列比較では、単一の Unicode 文字で書き込まれた同じ文字とは異なると見なされ、2 つの文字は長さが異なると考えられます (たとえば、CHAR_LENGTH() 関数で返されたものや、結果セットのメタデータにおけるもの)。

xxx_unicode_ci での順序付けが各言語で適切に機能しない場合にのみ、MySQL は言語固有の Unicode 照合順序を実装します。言語固有の照合順序は UCA ベースです。これらは、追加の言語の調整ルールを使用して、xxx_unicode_ci から派生されます。

4.0.0 以降の UCA バージョンに基づく照合順序では、照合順序名の中にバージョンが含まれます。したがって、xxx_unicode_520_ci 照合順序は、UCA 5.2.0 重みキー (http://www.unicode.org/Public/UCA/5.2.0/allkeys.txt) に基づきます。

LOWER() および UPPER() は、その引数の照合順序に従って、大文字と小文字の変換を実行します。4.0.0 より新しい Unicode バージョンでのみ大文字バージョンと小文字バージョンがある文字は、新しい UCA バージョンを使用する照合順序が引数にある場合にのみ、これらの機能によって変換されます。

Unicode 文字セットの場合、xxx_general_ci 照合順序を使用して実行する演算は、xxx_unicode_ci 照合順序のものよりも高速です。たとえば、utf8_general_ci 照合順序の比較は、utf8_unicode_ci の比較よりも高速ですが、精度は少し低くなります。この理由は、utf8_unicode_ci では、ある文字がほかの文字の組み合わせに等しいものと見なされる拡張形式などのマッピングをサポートしているためです。たとえば、ドイツ語とほかのいくつかの言語では、「ß」は「ss」と同じです。utf8_unicode_ci は、短縮形式と無視可能な文字もサポートします。utf8_general_ci は、拡張形式、短縮形式、無視可能な文字をサポートしない従来の照合順序です。文字間で 1 対 1 の比較しかできません。

さらに説明するために、次の等式は utf8_general_ci と utf8_unicode_ci の両方で構成されています (比較または検索を行うときのこの効果については、セクション10.1.7.8「照合順序の効果の例」を参照してください)。

```
Ä = A
Ö = O
```

```
Ü = U
```

照合順序間の違いは、次の式が utf8_general_ci に当てはまるという点です。

```
ß = s
```

一方、次の式は、ドイツ語 DIN-1 順序 (辞書順序とも呼ばれます) をサポートする utf8_unicode_ci に当てはまります。

```
ß = ss
```

MySQL が utf8 文字セットに対する言語固有の照合順序を実行するのは、utf8_unicode_ci による順序付けが言語に対してうまく機能しないときだけです。たとえば、utf8_unicode_ci は、ドイツ語辞書順序とフランス語には適切に機能するので、特殊な utf8 照合順序を作成する必要はありません。

utf8_general_ci も、ドイツ語とフランス語の両方にとって十分ですが、「ß」が「s」に等しく、「ss」に等しくない点が異なります。これがアプリケーションで許容可能な場合は、utf8_general_ci のほうが高速なので、こちらを使用する必要があります。これが許容できない場合 (たとえば、ドイツ語辞書順序が必要な場合) は、utf8_unicode_ci のほうがより正確なので、こちらを使用してください。

ドイツ語 DIN-2 (電話帳) 順序が必要な場合は、utf8_german2_ci 照合順序を使用してください。これは次の文字セットを等しいものと見なします。

```
Ä = Æ = AE
Ö = Œ = OE
Ü = UE
ß = ss
```

utf8_german2_ci は、latin1_german2_ci に似ていますが、後者は、「Æ」を「AE」に、または「Œ」を「OE」に等しいものとは見なしません。utf8_general_ci で十分であるので、ドイツ語辞書順序のための latin1_german_ci に対応する utf8_german_ci はありません。

xxx_swedish_ci には、スウェーデン語のルールが含まれます。たとえば、スウェーデン語には、ドイツ語やフランス語を使用するユーザーでは予期しないような次の関係があります。

```
Ü = Y < Ö
```

xxx_spanish_ci および xxx_spanish2_ci 照合順序は、それぞれ現代のスペイン語と伝統的なスペイン語に対応しています。どちらの照合順序でも、「ñ」 (n チルダ) は、「n」と「o」の間の独立した文字です。さらに、伝統的なスペイン語の場合、「ch」は、「c」と「d」の間の独立した文字であり、「ll」は、「l」と「m」の間の独立した文字です。

xxx_spanish2_ci 照合順序は、アストゥリアス語およびガリーシア語にも使用できます。

xxx_danich_ci 照合順序は、ノルウェー語にも使用できます。

xxx_roman_ci 照合順序では、I と J は等しいと見なされ、U と V は等しいと見なされます。

xxx_croatian_ci 照合順序は、Č、Ć、Dž、Đ、Lj、Nj、Š、Ž のクロアチア語の文字を調整します。

「バイナリ」 (xxx_bin) 照合順序を除くすべての Unicode 照合順序に対し、MySQL は文字の照合重みを探すようにテーブルルックアップを実行します。この重みは、WEIGHT_STRING() 関数を使用して表示できます。(セクション12.5「文字列関数」を参照してください。)文字がテーブル内にない場合 (たとえば、「新しい」文字であるため)、照合重み判定がより複雑になります。

- 一般的な照合順序 (xxx_general_ci) での BMP 文字の場合、重み = コードポイント。
- UCA 照合順序 (たとえば、xxx_unicode_ci と言語固有の照合順序) での BMP 文字の場合、次のアルゴリズムが適用されます。

```
if (code >= 0x3400 && code <= 0x4DB5)
  base= 0xFB80; /* CJK Ideograph Extension */
else if (code >= 0x4E00 && code <= 0x9FA5)
  base= 0xFB40; /* CJK Ideograph */
else
  base= 0xFBC0; /* All other characters */
aaaa= base +  (code >> 15);
bbbb= (code & 0x7FFF) | 0x8000;
```

結果は、aaaa に bbbb が続いた 2 つの照合要素のシーケンスになります。例:

```
mysql> SELECT HEX(WEIGHT_STRING(_ucs2 0x04CF COLLATE ucs2_unicode_ci));
+-------------------------------------------------------------+
| HEX(WEIGHT_STRING(_ucs2 0x04CF COLLATE ucs2_unicode_ci)) |
+-------------------------------------------------------------+
| FBC084CF                                                    |
+-------------------------------------------------------------+
```

したがって、U+04cf CYRILLIC SMALL LETTER PALOCHKA は、すべての UCA 4.0.0 照合順序で、U+04c0 CYRILLIC LETTER PALOCHKA より大きくなります。UCA 5.2.0 照合順序では、すべての palochka は一緒にソートされます。

- 一般的な照合順序の補助文字の場合、重みは 0xfffd REPLACEMENT CHARACTER の重みです。UCA 4.0.0 照合順序の補助文字の場合、照合重みは 0xfffd です。つまり、MySQL では、すべての補助文字は互いに等しく、ほとんどすべての BMP 文字より大きくなります。

  デザレット文字と COUNT(DISTINCT) を使用した例:

```
CREATE TABLE t (s1 VARCHAR(5) CHARACTER SET utf32 COLLATE utf32_unicode_ci);
INSERT INTO t VALUES (0xfffd);   /* REPLACEMENT CHARACTER */
INSERT INTO t VALUES (0x010412); /* DESERET CAPITAL LETTER BEE */
INSERT INTO t VALUES (0x010413); /* DESERET CAPITAL LETTER TEE */
SELECT COUNT(DISTINCT s1) FROM t;
```

  結果は 2 です。これは、MySQL xxx_unicode_ci 照合順序で、置換文字は 0x0dc6 の重みを持ちますが、デザレット B とデザレット T はどちらも、0xfffd の重みを持つからです。(utf32_general_ci 照合順序が代わりに使用された場合、この照合順序ではすべての 3 文字が 0xfffd の重みを持つので、結果は 1 になります。)

  くさび形文字と WEIGHT_STRING() を使用した例:

```
/*
The four characters in the INSERT string are
00000041  # LATIN CAPITAL LETTER A
0001218F  # CUNEIFORM SIGN KAB
000121A7  # CUNEIFORM SIGN KISH
00000042  # LATIN CAPITAL LETTER B
*/
CREATE TABLE t (s1 CHAR(4) CHARACTER SET utf32 COLLATE utf32_unicode_ci);
INSERT INTO t VALUES (0x00000041000121 8f000121a700000042);
SELECT HEX(WEIGHT_STRING(s1)) FROM t;
```

  結果は次のようになります。

```
0E33 FFFD FFFD 0E4A
```

  0E33 と 0E4A は、UCA 4.0.0 の主要な重みです。FFFD は KAB の重みであり、KISH の重みでもあります。

  すべての補助文字が互いに同じであるというルールは、最適ではありませんが、問題を引き起こすとは考えられません。これらの文字は非常に珍しく、マルチ文字文字列全体が補助文字から構成されることは非常にまれです。日本では、補助文字はあいまいな漢字表意文字であるので、一般的なユーザーは、いずれにしてもそれらの順序を気にしません。実際には、MySQL のルールに従って行をソートし、二次的にコードポイント値でソートする場合、次のようにすることが簡単です。

```
ORDER BY s1 COLLATE utf32_unicode_ci, s1 COLLATE utf32_bin
```

- 4.0.0 以降の UCA バージョンに基づいた補助文字の場合 (たとえば、xxx_unicode_520_ci)、必ずしもすべての補助文字が同じ照合順序重みを持つとはかぎりません。一部には、UCA allkeys.txt ファイルからの明示的な重みがあります。それ以外には、このアルゴリズムから計算された重みがあります。

```
aaaa= base +  (code >> 15);
bbbb= (code & 0x7FFF) | 0x8000;
```

utf16_bin 照合順序

「文字のコード値による順序付け」と「文字のバイナリ表現による順序付け」と間には違いがあります。これは、サロゲートのために、utf16_bin でのみ表示される違いです。

utf16_bin (utf16 のバイナリ照合順序) が、「文字ごと」ではなく「バイトごと」のバイナリ比較であったとします。その場合は、utf16_bin での文字の順序は utf8_bin での順序とは異なります。たとえば、次の表には 2 つの珍しい文字が示されています。最初の文字は E000-FFFF の範囲にあるので、サロゲートより大きくなりますが、補助文字より小さくなります。2 番目の文字は補助文字です。

Code point	Character	utf8	utf16

```
----------  ---------                   ----        -----
0FF9D       HALFWIDTH KATAKANA LETTER N  EF BE 9D     FF 9D
10384       UGARITIC LETTER DELTA        F0 90 8E 84  D8 00 DF 84
```

表の 2 つの文字は、0xff9d < 0x10384 であるので、コードポイント値による順序になります。また、0xef < 0xf0 なので、utf8 値による順序になります。ただし、0xff > 0xd8 なので、バイトごとの比較を使用する場合は、utf16 値による順序にはなりません。

したがって、MySQL の utf16_bin 照合順序は「バイトごと」にはなりません。「コードポイントによる」順序になります。MySQL は、utf16 での補助文字エンコーディングを認識すると、文字のコードポイント値に変換してから比較します。したがって、utf8_bin と utf16_bin の順序付けは同じになります。これは、UCS_BASIC 照合順序の SQL:2008 標準の要件に一致します。「UCS_BASIC は、ソートされている文字列の文字の Unicode スカラー値によって、順序付け全体が決定される照合順序です。これは UCS 文字レパートリーに適用できます。すべての文字レパートリーは、UCS レパートリーのサブセットなので、UCS_BASIC 照合順序は、すべての文字セットに適用できる可能性があります。ノート 11: 文字の Unicode スカラー値は、符号なしの整数として扱われるそのコードポイントです。」

文字セットが ucs2 である場合、比較はバイトごとになりますが、いずれにしても、ucs2 文字列にはサロゲートを含めないでください。

MySQL 5.6.5 で xxx_general_mysql500_ci 照合順序が追加されました。これらは、元の xxx_general_ci 照合順序の 5.1.24 以前の順序付けを維持し、MySQL 5.1.24 より前に作成されたテーブルのアップグレードを許可します。詳細は、セクション2.11.3「テーブルまたはインデックスの再構築が必要かどうかのチェック」およびセクション2.11.4「テーブルまたはインデックスの再作成または修復」を参照してください。

10.1.14.2 西ヨーロッパの文字セット

西ヨーロッパの文字セットには、大部分の西ヨーロッパ言語 (フランス語、スペイン語、カタロニア語、バスク語、ポルトガル語、イタリア語、アルバニア語、オランダ語、ドイツ語、デンマーク語、スウェーデン語、ノルウェー語、フィンランド語、フェロー語、アイスランド語、アイルランド語、スコットランド語、英語など) が含まれます。

- ascii (US ASCII) 照合順序:
  - ascii_bin
  - ascii_general_ci (デフォルト)
- cp850 (DOS 西ヨーロッパ言語) 照合順序:
  - cp850_bin
  - cp850_general_ci (デフォルト)
- dec8 (DEC 西ヨーロッパ言語) 照合順序:
  - dec8_bin
  - dec8_swedish_ci (デフォルト)
- hp8 (HP 西ヨーロッパ言語) 照合順序:
  - hp8_bin
  - hp8_english_ci (デフォルト)
- latin1 (cp1252 西ヨーロッパ言語) 照合順序:
  - latin1_bin
  - latin1_danish_ci
  - latin1_general_ci
  - latin1_general_cs
  - latin1_german1_ci
  - latin1_german2_ci

- latin1_spanish_ci
- latin1_swedish_ci (デフォルト)

latin1 はデフォルト文字セットです。MySQL の latin1 は Windows cp1252 文字セットと同じです。つまり、これは公式の ISO 8859-1 または IANA (Internet Assigned Numbers Authority) latin1 と同じですが、IANA latin1 が、0x80 と 0x9f の間のコードポイントを「未定義」として扱うのに対し、cp1252、および MySQL の latin1 はこれらの位置の文字を割り当てる点が異なることを示します。たとえば 0x80 はユーロの記号です。cp1252 の「定義されていない」エントリでは、MySQL は 0x81 を Unicode 0x0081 に、0x8d を0x008d に、0x8f を0x008f に、0x90 を0x0090 に、0x9d を0x009d に変換します。

latin1_swedish_ci 照合順序は、大部分の MySQL カスタマが使用しているデフォルトです。スウェーデン語/フィンランド語の照合順序ルールに基づいているとされていますが、スウェーデン人やフィンランド人の中にはこの意見に賛同しないユーザーもいます。

latin1_german1_ci と latin1_german2_ci 照合順序は DIN-1 および DIN-2 標準に基づきます。ここで、DIN は Deutsches Institut für Normung (ANSI のドイツ版) を表しています。DIN-1 は「辞書の照合順序」と呼ばれ、DIN-2 は「電話帳の照合順序」と呼ばれています。比較、または検索を行うときのこの効果の例については、セクション10.1.7.8「照合順序の効果の例」を参照してください。

- latin1_german1_ci (辞書) ルール:

```
Ä = A
Ö = O
Ü = U
ß = s
```

- latin1_german2_ci (電話帳) ルール:

```
Ä = AE
Ö = OE
Ü = UE
ß = ss
```

latin1_spanish_ci 照合順序では、「ñ」 (n チルダ) は、「n」と「o」の間の独立した文字です。

- macroman (Mac 西ヨーロッパ言語) 照合順序:
  - macroman_bin
  - macroman_general_ci (デフォルト)
- swe7 (7 ビットスウェーデン語) 照合順序:
  - swe7_bin
  - swe7_swedish_ci (デフォルト)

10.1.14.3 中央ヨーロッパの文字セット

MySQL ではチェコ、スロバキア、ハンガリー、ルーマニア、スロベニア、クロアチア、ポーランド、セルビア (ラテン) で使用される文字セットも一部サポートされています。

- cp1250 (Windows 中央ヨーロッパ言語) 照合順序:
  - cp1250_bin
  - cp1250_croatian_ci
  - cp1250_czech_cs
  - cp1250_general_ci (デフォルト)
  - cp1250_polish_ci
- cp852 (DOS 中央ヨーロッパ言語) 照合順序:
  - cp852_bin
  - cp852_general_ci (デフォルト)

- keybcs2 (DOS Kamenicky Czech-Slovak) 照合順序:
  - keybcs2_bin
  - keybcs2_general_ci (デフォルト)
- latin2 (ISO 8859-2 中央ヨーロッパ言語) 照合順序:
  - latin2_bin
  - latin2_croatian_ci
  - latin2_czech_cs
  - latin2_general_ci (デフォルト)
  - latin2_hungarian_ci
- macce (Mac 中央ヨーロッパ言語) 照合順序:
  - macce_bin
  - macce_general_ci (デフォルト)

10.1.14.4 南ヨーロッパおよび中東の文字セット

MySQL では南ヨーロッパや中東、アルメニア語、アラビア語、グルジア語、ギリシャ語、ヘブライ語、トルコ語の文字セットがサポートされています。

- armscii8 (ARMSCII-8 アルメニア語) 照合順序:
  - armscii8_bin
  - armscii8_general_ci (デフォルト)
- cp1256 (Windows アラビア語) 照合順序:
  - cp1256_bin
  - cp1256_general_ci (デフォルト)
- geostd8 (GEOSTD8 グルジア語) 照合順序:
  - geostd8_bin
  - geostd8_general_ci (デフォルト)
- greek (ISO 8859-7 ギリシャ語) 照合順序:
  - greek_bin
  - greek_general_ci (デフォルト)
- hebrew (ISO 8859-8 ヘブライ語) 照合順序:
  - hebrew_bin
  - hebrew_general_ci (デフォルト)
- latin5 (ISO 8859-9 トルコ語) 照合順序:
  - latin5_bin
  - latin5_turkish_ci (デフォルト)

10.1.14.5 バルト語の文字セット

バルト語に含まれる文字セットにはエストニア語、ラトビア語、リトアニア言語が含まれます。

- cp1257 (Windows バルト語) 照合順序:

  - cp1257_bin
  - cp1257_general_ci (デフォルト)
  - cp1257_lithuanian_ci
- latin7 (ISO 8859-13 バルト語) 照合順序:
  - latin7_bin
  - latin7_estonian_cs
  - latin7_general_ci (デフォルト)
  - latin7_general_cs

10.1.14.6 キリル文字の文字セット

ベラルーシ語、ブルガリア語、ロシア語、ウクライナ語、セルビア語 (キリル) とともに使用するキリル文字の文字セットおよび照合順序を以下に示します。

- cp1251 (Windows キリル文字) 照合順序:
  - cp1251_bin
  - cp1251_bulgarian_ci
  - cp1251_general_ci (デフォルト)
  - cp1251_general_cs
  - cp1251_ukrainian_ci
- cp866 (DOS ロシア語) 照合順序:
  - cp866_bin
  - cp866_general_ci (デフォルト)
- koi8r (KOI8-R Relcom Russian) 照合順序:
  - koi8r_bin
  - koi8r_general_ci (デフォルト)
- koi8u (KOI8-U ウクライナ語) 照合順序:
  - koi8u_bin
  - koi8u_general_ci (デフォルト)

10.1.14.7 アジアの文字セット

サポートされているアジアの文字セットには、中国語、日本語、韓国語、タイ語が含まれています。これらは複雑な場合があります。たとえば、中国語の文字セットは数千種類の文字に対応している必要があります。cp932 および sjis 文字セットの追加情報については、cp932 文字セットを参照してください。

MySQL でのアジアの文字セットのサポートに関連したよくある質問および問題に対する回答については、セクションA.11「MySQL 5.6 FAQ: MySQL の中国語、日本語、および韓国語の文字セット」を参照してください。

- big5 (Big5 繁体字中国語) 照合順序:
  - big5_bin
  - big5_chinese_ci (デフォルト)
- cp932 (SJIS for Windows 日本語) 照合順序:
  - cp932_bin

    - cp932_japanese_ci (デフォルト)
- eucjpms (UJIS for Windows 日本語) 照合順序:
    - eucjpms_bin
    - eucjpms_japanese_ci (デフォルト)
- euckr (EUC-KR 韓国語) 照合順序:
    - euckr_bin
    - euckr_korean_ci (デフォルト)
- gb2312 (GB2312 簡体字中国語) 照合順序:
    - gb2312_bin
    - gb2312_chinese_ci (デフォルト)
- gbk (GBK 簡体字中国語) 照合順序:
    - gbk_bin
    - gbk_chinese_ci (デフォルト)
- sjis (Shift-JIS 日本語) 照合順序:
    - sjis_bin
    - sjis_japanese_ci (デフォルト)
- tis620 (TIS620 タイ語) 照合順序:
    - tis620_bin
    - tis620_thai_ci (デフォルト)
- ujis (EUC-JP 日本語) 照合順序:
    - ujis_bin
    - ujis_japanese_ci (デフォルト)

big5_chinese_ci 照合順序は画数でソートします。

cp932 文字セット

cp932 が必要な理由

MySQL では sjis 文字セットは IANA で定義される Shift_JIS 文字セットに対応しており、これらは JIS X0201 および JIS X0208 文字セットをサポートしています。(http://www.iana.org/assignments/character-sets を参照してください。)

ただし、記述用語としての「SHIFT JIS」の意味は非常にあいまいになっており、さまざまなベンダーが定義した Shift_JIS に対する拡張も含める場合もあります。

たとえば、日本語 Windows 環境で使用される「SHIFT JIS」は、Microsoft による Shift_JIS の拡張であり、正式な名称は Microsoft Windows Codepage : 932 または cp932 です。Shift_JIS でサポートされる文字に加え、cp932 では、NEC 特殊文字、NEC 選定 IBM 拡張文字、IBM 選定文字などの拡張文字をサポートします。

多くの日本語ユーザーは、これらの拡張文字を使用するときに問題に直面してきました。これらの問題は次の要因によって生じていました。

- MySQL が自動的に文字セットの変換を行なっていること。
- 文字セットが Unicode (ucs2) を使用して変換されていること。
- sjis 文字セットが、これらの拡張文字の変換をサポートしていないこと。

- いわゆる「SHIFT JIS」から Unicode への変換には複数の変換ルールが存在し、文字によっては、変換ルールに従って別々に Unicode に変換される場合があること。MySQL ではこれらの変換ルールのうち、1 つしかサポートしていません (詳細は後述します)。

MySQL の cp932 文字セットは、これらの問題を解決するように設計されています。

MySQL が文字セットの変換をサポートするので、異なる変換ルールを持つ IANA の Shift_JIS と cp932 を 2 つの異なる文字セットに区分することが重要になります。

cp932 と sjis との相違点

cp932 文字セットは次の点で sjis と異なります。

- cp932 は、NEC 特殊文字、NEC 選定 IBM 拡張文字、IBM 選定文字をサポートします。
- 一部の cp932 文字には、2 つの異なるコードポイントがあり、両方とも同一の Unicode コードポイントに変換されます。Unicode から cp932 に戻すときに、どちらかのコードポイントを選択する必要があります。この「ラウンドトリップ変換」については、Microsoft が推奨するルールが使用されます。(http://support.microsoft.com/kb/170559/EN-US/を参照してください。)

  この変換ルールは次のように機能します。

  - 文字が JIS X 0208 文字と NEC 特殊文字の両方に存在する場合には、JIS X 0208 のコードポイントを使用します。
  - 文字が NEC 特殊文字と IBM 選定文字の両方に存在する場合には、NEC 特殊文字のコードポイントを使用します。
  - 文字が IBM 選定文字と NEC 選定 IBM 拡張文字の両方に存在する場合は、IBM 拡張文字のコードポイントを使用します。

  http://www.microsoft.com/globaldev/reference/dbcs/932.htm に示す表には、cp932 文字の Unicode 値に関する情報が記載されています。cp932 の表で下に 4 桁の数字が表示されている項目については、その数字は対応する Unicode (ucs2) エンコーディングを表します。下線付きの 2 桁の値のある表項目では、これらの 2 桁の値で始まる一定範囲の cp932 文字があります。このような表項目をクリックすると、これらの桁の値で始まる cp932 文字対する各 Unicode 値を示したページが表示されます。

  詳細は次のリンクを参照してください。それぞれ、次の文字セットのエンコーディングに対応します。

  - NEC 特殊文字:

    ```
    http://www.microsoft.com/globaldev/reference/dbcs/932/932_87.htm
    ```

  - NEC 選定 IBM 拡張文字:

    ```
    http://www.microsoft.com/globaldev/reference/dbcs/932/932_ED.htm
    http://www.microsoft.com/globaldev/reference/dbcs/932/932_EE.htm
    ```

  - IBM 選定文字:

    ```
    http://www.microsoft.com/globaldev/reference/dbcs/932/932_FA.htm
    http://www.microsoft.com/globaldev/reference/dbcs/932/932_FB.htm
    http://www.microsoft.com/globaldev/reference/dbcs/932/932_FC.htm
    ```

- cp932 は、eucjpms と組み合わせて使用することで、ユーザー定義の文字の変換をサポートし、sjis/ujis の変換での問題に対応します。詳細は、http://www.sljfaq.org/afaq/encodings.htmlを参照してください。

一部の文字については、ucs2 との間の変換は、sjis と cp932 との場合と異なります。次の表に、これらの違いを示します。

ucs2 への変換:

sjis/cp932 値	sjis -> ucs2 の変換	cp932 -> ucs2 の変換
5C	005C	005C
7E	007E	007E
815C	2015	2015
815F	005C	FF3C

sjis/cp932 値	sjis -> ucs2 の変換	cp932 -> ucs2 の変換
8160	301C	FF5E
8161	2016	2225
817C	2212	FF0D
8191	00A2	FFE0
8192	00A3	FFE1
81CA	00AC	FFE2

ucs2 からの変換:

ucs2 値	ucs2 -> sjis の変換	ucs2 -> cp932 の変換
005C	815F	5C
007E	7E	7E
00A2	8191	3F
00A3	8192	3F
00AC	81CA	3F
2015	815C	815C
2016	8161	3F
2212	817C	3F
2225	3F	8161
301C	8160	3F
FF0D	3F	817C
FF3C	3F	815F
FF5E	3F	8160
FFE0	3F	8191
FFE1	3F	8192
FFE2	3F	81CA

日本語文字セットのユーザーは、--character-set-client-handshake (または --skip-character-set-client-handshake) を使用すると大きな効果が得られることに注意してください。セクション5.1.3「サーバーコマンドオプション」を参照してください。

10.2 エラーメッセージ言語の設定

デフォルトで、mysqld はエラーメッセージを英語で生成しますが、チョコ語、デンマーク語、オランダ語、エストニア語、フランス語、ドイツ語、ギリシャ語、ハンガリー語、イタリア語、日本語、韓国語、ノルウェー語、ノルウェー語 (ニーノシュク)、ポーランド語、ポルトガル語、ルーマニア語、ロシア語、スロバキア語、スペイン語、スウェーデン語のほかの言語でも表示できます。

このセクションの説明を使用して、サーバーでエラーメッセージに使用する言語を選択できます。

MySQL 5.6 では、サーバーは、次の 2 つの場所でエラーメッセージファイルを検索します。

- lc_messages_dir と lc_messages (後者は言語名に変換されます) の 2 つのシステム変数値から構築されたディレクトリでファイルを検索しようとします。次のコマンドを使用してサーバーを起動するとします。

  ```
  shell> mysqld --lc_messages_dir=/usr/share/mysql --lc_messages=fr_FR
  ```

  この場合、mysqld は、ロケール fr_FR を言語 french にマップし、/usr/share/mysql/french ディレクトリでエラーファイルを検索します。

- 前述のように構築されたディレクトリでメッセージファイルが見つからない場合、サーバーは、lc_messages 値を無視し、検索する場所として lc_messages_dir 値だけを使用します。

lc_messages_dir システム変数は、グローバル値だけを含み、読み取り専用です。lc_messages は、グローバル値とセッション値を含み実行時に変更できるので、サーバーの実行中にエラーメッセージ言語を変更でき、個々の

クライアントはそれぞれ、そのセッション lc_messages 値を別のロケール名に変更することによって、異なるエラーメッセージ言語を使用できます。たとえば、サーバーがエラーメッセージに fr_FR ロケールを使用している場合、クライアントは、次のステートメントを実行すると、英語でエラーメッセージを受信できます。

```
mysql> SET lc_messages = 'en_US';
```

デフォルトでは、言語ファイルは、MySQL ベースディレクトリ下の share/mysql/LANGUAGE ディレクトリにあります。

エラーメッセージの (言語ではなく) 文字セットの変更の詳細は、セクション10.1.6「エラーメッセージの文字セット」を参照してください。

「MySQL Internals: Error Messages」で入手できる MySQL Internals マニュアルの手順を使用すると、サーバーで生成されるエラーメッセージの内容を変更できます。エラーメッセージの内容を変更する場合は、MySQL のより新しいバージョンにアップグレードするごとに必ず変更を繰り返してください。

10.3 文字セットの追加

このセクションでは、MySQL に文字セットを追加する手順について説明します。適切な手順は、文字セットが単純か複雑かによって異なります。

- ソートに特別な文字列照合ルーチンを必要とせず、マルチバイト文字のサポートを必要としない文字セットが、単純な文字セットです。
- これらのどちらかの機能が必要な文字セットが、複雑な文字セットです。

たとえば、greek と swe7 は単純な文字セットですが、big5 と czech は複雑な文字セットです。

次の手順を使用するには、MySQL ソース配布が必要です。この手順では、MYSET は追加する文字セットの名前を表します。

1. MYSET の <charset> 要素を sql/share/charsets/Index.xml ファイルに追加します。ファイル内既存の内容を、新しい内容を追加するためのガイドとして使用します。latin1 <charset> 要素のリストの一部を以下に示します。

   ```
   <charset name="latin1">
    <family>Western</family>
    <description>cp1252 West European</description>
    ...
    <collation name="latin1_swedish_ci" id="8" order="Finnish, Swedish">
      <flag>primary</flag>
      <flag>compiled</flag>
    </collation>
    <collation name="latin1_danish_ci" id="15" order="Danish"/>
    ...
    <collation name="latin1_bin" id="47" order="Binary">
      <flag>binary</flag>
      <flag>compiled</flag>
    </collation>
    ...
   </charset>
   ```

   <charset> 要素は、文字セットのすべての照合順序を一覧表示します。これらには少なくも、バイナリ照合順序とデフォルト (プライマリ) 照合順序が含まれます。デフォルト照合順序は多くの場合、general_ci (一般、大文字と小文字を区別) のサフィクスを使用して名前が付けられます。バイナリ照合順序をデフォルト照合順序にすることは可能ですが、通常、これらは異なります。デフォルト照合順序には primary フラグを付ける必要があります。バイナリ照合順序には binary フラグを付ける必要があります。

   それぞれの照合順序に一意の ID 番号を割り当てる必要があります。1024 から 2047 の ID 範囲は、ユーザー定義の照合順序に予約されています。現在使用されている照合順序 ID の最大値を検索するには、次のクエリーを使用します。

   ```
   SELECT MAX(ID) FROM INFORMATION_SCHEMA.COLLATIONS;
   ```

2. このステップは、追加しているのが単純な文字セットか、複雑な文字セットかにより異なります。単純な文字セットには、構成ファイルだけが必要ですが、複雑な文字セットには、照合順序関数またはマルチバイト関数あるいはその両方を定義する C ソースファイルが必要です。

   単純な文字セットの場合、文字セットプロパティーについて記した構成ファイル (MYSET.xml) を作成します。sql/share/charsets ディレクトリにこのファイルを作成します。このファイルの土台として latin1.xml のコピーを使用できます。ファイルの構文は非常に単純です。

- コメントは、通常の XML コメント (<!-- text -->) として記述されます。
- <map> 配列要素内の単語は、任意の数の空白によって区切られます。
- <map> 配列要素内の各単語は、16 進形式の数値で表す必要があります。
- <ctype> 要素の <map> 配列要素には 257 語が含まれます。そのあとのほかの <map> 配列要素には 256 語が含まれます。セクション10.3.1「文字定義配列」を参照してください。
- Index.xml 内の文字セットに対して <charset> 要素に一覧表示された照合順序ごとに、文字の順序を定義する <collation> 要素を MYSET.xml に含める必要があります。

複雑な文字セットの場合、文字セットプロパティーについて記述し、文字セットに対する演算を適切に実行するために必要なサポートルーチンを定義した C ソースファイルを作成します。

- strings ディレクトリに ctype-MYSET.c ファイルを作成します。既存の ctype-*.c ファイルのいずれか (ctype-big5.c など) を調べて、定義する必要のあるものを確認します。ファイル内の配列には、ctype_MYSET、to_lower_MYSET などの名前を付ける必要があります。これらは、単純な文字セットの配列に対応します。セクション10.3.1「文字定義配列」を参照してください。
- Index.xml 内の文字セットに対して <charset> 要素に一覧表示された <collation> 要素ごとに、ctype-MYSET.c ファイルが照合順序の実装を提供する必要があります。
- 文字セットで文字列照合関数が必要な場合は、セクション10.3.2「複雑な文字セットの文字列照合のサポート」を参照してください。
- 文字セットでマルチバイト文字のサポートが必要な場合は、セクション10.3.3「複雑な文字セットのマルチバイト文字のサポート」を参照してください。

3. 構成情報を変更します。MYSYS の情報を追加するためのガイドとして、既存の構成情報を使用します。ここでの例では、文字セットにデフォルト照合順序とバイナリ照合順序があることを想定していますが、MYSET に追加の照合順序がある場合には、さらに多くの行が必要になります。

   a. mysys/charset-def.c を編集し、新しい文字セットの照合順序を「登録」します。

      「宣言」セクションに次の行を追加します。

      ```
      #ifdef HAVE_CHARSET_MYSET
      extern CHARSET_INFO my_charset_MYSET_general_ci;
      extern CHARSET_INFO my_charset_MYSET_bin;
      #endif
      ```

      「登録」セクションに次の行を追加します。

      ```
      #ifdef HAVE_CHARSET_MYSET
        add_compiled_collation(&my_charset_MYSET_general_ci);
        add_compiled_collation(&my_charset_MYSET_bin);
      #endif
      ```

   b. 文字セットが ctype-MYSET.c を使用する場合、strings/CMakeLists.txt を編集して、ctype-MYSET.c を STRINGS_SOURCES 変数の定義に追加します。

   c. cmake/character_sets.cmake を編集します。

      i. アルファベット順で CHARSETS_AVAILABLE の値に MYSET を追加します。

      ii. アルファベット順で CHARSETS_COMPLEX の値に MYSET を追加します。これは単純な文字セットにも必要です。ない場合は、CMake は -DDEFAULT_CHARSET=MYSET を認識しません。

4. 再構成し、再コンパイルし、テストします。

10.3.1 文字定義配列

それぞれの単純な文字セットには、sql/share/charsets ディレクトリに置かれた構成ファイルがあります。MYSYS という名前の文字セットの場合、ファイルには MYSET.xml の名前が付けられます。これは、<map> 配列要素を使用して、文字セットプロパティーを一覧表示します。<map> 要素は、これらの要素内に表示されます。

- <ctype> は文字ごとに属性を定義します。

- <lower> と <upper> は、小文字と大文字を一覧表示します。
- <unicode> は、8 ビット文字値を Unicode 値にマップします。
- <collation> 要素は、比較とソートでの文字の順序を、照合順序ごとに 1 つの要素で示します。文字コード自体が順序を提供するので、バイナリ照合順序には <map> 要素は不要です。

strings ディレクトリ内の ctype-MYSET.c ファイルに実装された複雑な文字セットには、ctype_MYSET[]、to_lower_MYSET[] などの対応する配列があります。すべての複雑な文字セットがすべての配列を持つわけではありません。例については、既存の ctype-*.c ファイルも参照してください。追加情報については、strings ディレクトリ内の CHARSET_INFO.txt ファイルを参照してください。

ほとんどの配列には、文字値でインデックスが付けられ、256 個の要素があります。<ctype> 配列には、文字値 + 1 でインデックスが付けられ、257 個の要素があります。これは、EOF を処理するための従来の規則です。

<ctype> 配列要素は、ビット値です。各要素は、文字セット内の単一の文字の属性について記述します。各属性は、include/m_ctype.h での定義に従って、ビットマスクに関連付けられます。

```
#define _MY_U   01      /* Upper case */
#define _MY_L   02      /* Lower case */
#define _MY_NMR 04      /* Numeral (digit) */
#define _MY_SPC 010     /* Spacing character */
#define _MY_PNT 020     /* Punctuation */
#define _MY_CTR 040     /* Control character */
#define _MY_B   0100    /* Blank */
#define _MY_X   0200    /* heXadecimal digit */
```

所定の文字の <ctype> 値は、その文字について記述した適用可能なビットマスク値の結合である必要があります。たとえば、'A' は 16 進数 (_MY_X) と同様に大文字 (_MY_U) であるので、その ctype 値は次のように定義する必要があります。

```
ctype['A'+1] = _MY_U | _MY_X = 01 | 0200 = 0201
```

m_ctype.h のビットマスク値は 8 進数値ですが、MYSET.xml 内の <ctype> 配列の要素は 16 進値として書き込む必要があります。

<lower> と <upper> 配列は、文字セットの各メンバーに対応した小文字と大文字を保持します。例:

```
lower['A'] should contain 'a'
upper['a'] should contain 'A'
```

各 <collation> 配列は、比較およびソートでどのように文字を順序付けする必要があるかを示します。MySQL は、この情報の値に基づいて文字をソートします。場合によっては、これは <upper> 配列と同じであり、ソートで大文字と小文字が区別されないことになります。さらに複雑なソートルールについては (複雑な文字セットの場合)、セクション10.3.2「複雑な文字セットの文字列照合のサポート」での文字列照合の説明を参照してください。

10.3.2 複雑な文字セットの文字列照合のサポート

MYSET という名前の単純な文字セットの場合、ソートルールは、<collation> 要素内の <map> 配列要素を使用して、MYSET.xml 構成ファイルで指定されます。言語に関するソートルールが非常に複雑で、単純な配列では扱えない場合、strings ディレクトリ内の ctype-MYSET.c ソースファイルで文字列照合関数を定義する必要があります。

既存の文字セットからは、これらの関数がどのように実装されているかを示す最適なドキュメントおよび例が得られます。big5、czech、gbk、sjis、tis160 文字セットのファイルなど、strings ディレクトリ内の ctype-*.c ファイルを調べます。MY_COLLATION_HANDLER 構造を見て、どのように使用されているかを確認します。追加情報については、strings ディレクトリ内の CHARSET_INFO.txt ファイルも参照してください。

10.3.3 複雑な文字セットのマルチバイト文字のサポート

マルチバイト文字を含む MYSET という名前の新しい文字セットのサポートを追加する場合、strings ディレクトリ内の ctype-MYSET.c ソースファイルのマルチバイト文字関数を使用する必要があります。

既存の文字セットからは、これらの関数がどのように実装されているかを示す最適なドキュメントおよび例が得られます。euc_kr、gb2312、gbk、sjis、ujis 文字セットのファイルなど、strings ディレクトリ内の ctype-*.c ファイルを調べます。MY_CHARSET_HANDLER 構造を見て、どのように使用されているかを確認します。追加情報については、strings ディレクトリ内の CHARSET_INFO.txt ファイルも参照してください。

10.4 文字セットへの照合順序の追加

照合順序は、文字列を比較およびソートする方法を定義した一連のルールです。MySQL でのそれぞれの照合順序は、単一の文字セットに属しています。すべての文字セットには少なくとも 1 つの照合順序が属し、ほとんどの文字セットのは 2 つ以上の照合順序が属しています。

照合順序は重みに基づいて文字を順序付けします。文字セット内のそれぞれの文字が重みにマップされています。重みが等しい文字は同等と見なされ、重みが等しくない文字は、その重みの相対的な大きさに従って比較されます。

WEIGHT_STRING() 関数を使用すると、文字列内の文字の重みを確認できます。重みを示した返される値はバイナリ文字列であるので、HEX(WEIGHT_STRING(str)) を使用して重みを出力可能な形式で表示すると便利です。次の例は、大文字と小文字を区別しない非バイナリ文字列である場合は、'AaBb' 内の大文字と小文字で重みは異ならないが、バイナリ文字列である場合は異なることを示します。

```
mysql> SELECT HEX(WEIGHT_STRING('AaBb' COLLATE latin1_swedish_ci));
+-------------------------------------------------+
| HEX(WEIGHT_STRING('AaBb' COLLATE latin1_swedish_ci)) |
+-------------------------------------------------+
| 41414242                                        |
+-------------------------------------------------+
mysql> SELECT HEX(WEIGHT_STRING(BINARY 'AaBb'));
+----------------------------------+
| HEX(WEIGHT_STRING(BINARY 'AaBb')) |
+----------------------------------+
| 41614262                         |
+----------------------------------+
```

セクション10.4.1「照合順序の実装タイプ」で説明するように、MySQL では複数の照合順序の実装をサポートしています。これらの中には、再コンパイルせずに、MySQL に追加できるものもあります。

- 8 ビットの文字セットの単純な照合順序。
- Unicode 文字セットの UCA ベースの照合順序。
- バイナリ (xxx_bin) 照合順序。

以降のセクションでは、最初の 2 種類の照合順序を既存の文字セットに追加する方法について説明します。バイナリ照合順序については、既存の文字セットにもすでに用意されているので、ここでは追加方法は説明しません。

新しい照合順序を追加する手順のサマリー:

1. 照合順序 ID を選択します。
2. 照合順序に名前を付ける構成情報を追加し、文字順序付けルールについて記述します。
3. サーバーを再起動します。
4. 照合順序が存在していることを検証します。

ここでの説明では、MySQL を再コンパイルすることなく追加できる照合順序だけを取り上げます。再コンパイルを必要とする照合順序 (C ソースファイル内の関数を利用して実装されたものなど) を追加するには、セクション10.3「文字セットの追加」の手順を使用してください。ただし、完全な文字セットに必要なすべての情報を追加するのではなく、既存の文字セットに合わせて適切なファイルを変更します。つまり、文字セットの現在の照合順序ですでに存在するものに基づいて、新しい照合順序のデータ構造、関数、構成情報を追加します。

> **注記**
>
> 既存の照合順序を変更すると、その照合順序を使用するカラムでインデックスの行順序に影響が及ぶことがあります。この場合、間違ったクエリー結果などの問題が起こらないように、これらのインデックスを再構築してください。詳細は、セクション2.11.3「テーブルまたはインデックスの再構築が必要かどうかのチェック」を参照してください。

追加のリソース

- Unicode 照合順序アルゴリズム (UCA) の仕様: http://www.unicode.org/reports/tr10/

- Locale Data Markup Language (LDML) の仕様: http://www.unicode.org/reports/tr35/
- MySQL ブログ記事「新しい Unicode 照合順序を追加するための手順」: http://blogs.mysql.com/peterg/2008/05/19/instructions-for-adding-a-new-unicode-collation/

10.4.1 照合順序の実装タイプ

MySQL は複数のタイプの照合順序を実装します。

8 ビットの文字セットに対する単純な照合順序

この種の照合順序は、文字コードと重みの 1 対 1 のマッピングを定義した 256 個の重みの配列を使用して実装されます。latin1_swedish_ci がその一例です。これは、大文字と小文字を区別しない照合順序なので、大文字と小文字は同じ重みで、等しいものと見なされます。

```
mysql> SET NAMES 'latin1' COLLATE 'latin1_swedish_ci';
Query OK, 0 rows affected (0.01 sec)

mysql> SELECT HEX(WEIGHT_STRING('a')), HEX(WEIGHT_STRING('A'));
+-------------------------+-------------------------+
| HEX(WEIGHT_STRING('a')) | HEX(WEIGHT_STRING('A')) |
+-------------------------+-------------------------+
| 41                      | 41                      |
+-------------------------+-------------------------+
1 row in set (0.01 sec)

mysql> SELECT 'a' = 'A';
+-----------+
| 'a' = 'A' |
+-----------+
|         1 |
+-----------+
1 row in set (0.12 sec)
```

実装の手順については、セクション10.4.3「8 ビットの文字セットへの単純な照合順序の追加」を参照してください。

8 ビット文字セットに対する複雑な照合順序

この種の照合順序は、セクション10.3「文字セットの追加」で説明しているように、文字を順序付けする方法を定義した C ソースファイル内の関数を使用して実装されます。

Unicode 以外のマルチバイト文字セットの照合順序

この種の照合順序では、8 ビット (シングルバイト) 文字とマルチバイト文字が異なる方法で処理されます。8 ビットの文字の場合、文字コードは大文字と小文字を区別しない形式で重みにマップされます。(たとえば、シングルバイト文字 'a' と 'A' はどちらも重みが 0x41 です。)マルチバイト文字の場合、文字コードと重みの間には、2 種類の関係があります。

- 重みが文字コードと等しい場合。sjis_japanese_ci がこの種の照合順序の一例です。マルチバイト文字 'ぢ' の文字コードは 0x82C0 であり、重みも 0x82C0 です。

```
mysql> CREATE TABLE t1
    -> (c1 VARCHAR(2) CHARACTER SET sjis COLLATE sjis_japanese_ci);
Query OK, 0 rows affected (0.01 sec)

mysql> INSERT INTO t1 VALUES ('a'),('A'),(0x82C0);
Query OK, 3 rows affected (0.00 sec)
Records: 3  Duplicates: 0  Warnings: 0

mysql> SELECT c1, HEX(c1), HEX(WEIGHT_STRING(c1)) FROM t1;
+------+---------+------------------------+
| c1   | HEX(c1) | HEX(WEIGHT_STRING(c1)) |
+------+---------+------------------------+
| a    | 61      | 41                     |
| A    | 41      | 41                     |
| ぢ   | 82C0    | 82C0                   |
+------+---------+------------------------+
3 rows in set (0.00 sec)
```

- 文字コードが 1 対 1 で重みにマップされていても、必ずしもコードと重みが等しくない場合。gbk_chinese_ci がこの種の照合順序の一例です。マルチバイト文字 '膰' の文字コードは 0x81B0 ですが、重みは 0xC286 です。

```
mysql> CREATE TABLE t1
    -> (c1 VARCHAR(2) CHARACTER SET gbk COLLATE gbk_chinese_ci);
Query OK, 0 rows affected (0.33 sec)

mysql> INSERT INTO t1 VALUES ('a'),('A'),(0x81B0);
Query OK, 3 rows affected (0.00 sec)
Records: 3  Duplicates: 0  Warnings: 0

mysql> SELECT c1, HEX(c1), HEX(WEIGHT_STRING(c1)) FROM t1;
+------+---------+------------------------+
| c1   | HEX(c1) | HEX(WEIGHT_STRING(c1)) |
+------+---------+------------------------+
| a    | 61      | 41                     |
| A    | 41      | 41                     |
| 膰   | 81B0    | C286                   |
+------+---------+------------------------+
3 rows in set (0.00 sec)
```

実装の手順については、セクション10.3「文字セットの追加」を参照してください。

Unicode マルチバイト文字セットの照合順序

これらの照合順序の一部は Unicode 照合順序アルゴリズム (UCA) に基づきますが、それ以外は基づいていません。

UCA に基づいていない照合順序では、文字コードと重みは 1 対 1 でマップしています。MySQL では、このような照合順序は大文字と小文字を区別せず、アクセント、濁音、破裂音を区別しません。utf8_general_ci がその一例です。'a'、'A'、'À'、および 'á' のそれぞれは文字コードは別々ですが、重みはすべて 0x0041 であり、等しいものと見なされます。

```
mysql> SET NAMES 'utf8' COLLATE 'utf8_general_ci';
Query OK, 0 rows affected (0.00 sec)

mysql> CREATE TABLE t1
    -> (c1 CHAR(1) CHARACTER SET UTF8 COLLATE utf8_general_ci);
Query OK, 0 rows affected (0.01 sec)

mysql> INSERT INTO t1 VALUES ('a'),('A'),('À'),('á');
Query OK, 4 rows affected (0.00 sec)
Records: 4  Duplicates: 0  Warnings: 0

mysql> SELECT c1, HEX(c1), HEX(WEIGHT_STRING(c1)) FROM t1;
+------+---------+------------------------+
| c1   | HEX(c1) | HEX(WEIGHT_STRING(c1)) |
+------+---------+------------------------+
| a    | 61      | 0041                   |
| A    | 41      | 0041                   |
| À    | C380    | 0041                   |
| á    | C3A1    | 0041                   |
+------+---------+------------------------+
4 rows in set (0.00 sec)
```

MySQL の UCA ベースの照合順序には、次の 3 つのプロパティーがあります。

- 文字に重みがある場合、それぞれの重みは 2 バイト (16 ビット) を使用します。
- 文字の重みはゼロ (または空の重み) の場合があります。この場合、文字は無視できます。例:「U+0000 NULL」は重みがなく、無視できます。
- 1 つの文字が 1 つの重みを持つ場合があります。例: 'a' には 0x0E33 の重みがあります。

```
mysql> SET NAMES 'utf8' COLLATE 'utf8_unicode_ci';
Query OK, 0 rows affected (0.05 sec)

mysql> SELECT HEX('a'), HEX(WEIGHT_STRING('a'));
+----------+-------------------------+
| HEX('a') | HEX(WEIGHT_STRING('a')) |
+----------+-------------------------+
| 61       | 0E33                    |
+----------+-------------------------+
1 row in set (0.02 sec)
```

- 1 つの文字が複数の重みを持つ場合があります。これは拡張形式です。例: ドイツ語の文字 'ß' (SZ リガチャーまたは SHARP S) には、0x0FEA0FEA の重みがあります。

```
mysql> SET NAMES 'utf8' COLLATE 'utf8_unicode_ci';
```

```
Query OK, 0 rows affected (0.11 sec)

mysql> SELECT HEX('ß'), HEX(WEIGHT_STRING('ß'));
+-----------+-------------------------+
| HEX('ß')  | HEX(WEIGHT_STRING('ß')) |
+-----------+-------------------------+
| C39F      | 0FEA0FEA                |
+-----------+-------------------------+
1 row in set (0.00 sec)
```

- 複数の文字が 1 つの重みを持つ場合があります。これは短縮形式です。例: 'ch' は、チェコ語の単一の文字であり、0x0EE2 の重みを持ちます。

```
mysql> SET NAMES 'utf8' COLLATE 'utf8_czech_ci';
Query OK, 0 rows affected (0.09 sec)

mysql> SELECT HEX('ch'), HEX(WEIGHT_STRING('ch'));
+-----------+-------------------------+
| HEX('ch') | HEX(WEIGHT_STRING('ch')) |
+-----------+-------------------------+
| 6368      | 0EE2                    |
+-----------+-------------------------+
1 row in set (0.00 sec)
```

複数の文字と複数の重みのマッピングも可能です (これは拡張形式を使用した短縮形式です) が、MySQL ではサポートされていません。

UCA に基づいていない照合順序に関する実装の手順については、セクション10.3「文字セットの追加」を参照してください。UCA 照合順序については、セクション10.4.4「Unicode 文字セットへの UCA 照合順序の追加」を参照してください。

その他の照合順序

上記のカテゴリのどれにも該当しない照合順序も少数存在します。

10.4.2 照合順序 ID の選択

各照合順序には一意の ID が必要です。照合順序を追加するには、現在使用されていない ID 値を選択する必要があります。1024 から 2047 の ID 範囲は、ユーザー定義の照合順序に予約されています。MySQL 5.6.3 以降、InnoDB テーブルは 2 バイトの照合順序 ID をサポートしています。MySQL 5.6.3 より前では、InnoDB テーブルは、255 を最大値とするシングルバイトの照合順序 ID だけをサポートしていました。MyISAM テーブルでは、MySQL 5.5 以降の 2 バイトの照合順序 ID をサポートしています。

選択した照合順序は次のコンテキストで表示されます。

- INFORMATION_SCHEMA.COLLATIONS テーブルの ID カラム。
- SHOW COLLATION 出力の Id カラム。
- MYSQL_FIELD C API データ構造の charsetnr メンバー。
- mysql_get_character_set_info() C API 関数で返される MY_CHARSET_INFO データ構造の number メンバー。

現在使用されている最大の ID を判別するには、次のステートメントを発行します。

```
mysql> SELECT MAX(ID) FROM INFORMATION_SCHEMA.COLLATIONS;
+---------+
| MAX(ID) |
+---------+
|     210 |
+---------+
```

現在使用されているすべての ID のリストを表示するには、次のステートメントを発行します。

```
mysql> SELECT ID FROM INFORMATION_SCHEMA.COLLATIONS ORDER BY ID;
+-----+
| ID  |
+-----+
|   1 |
|   2 |
| ... |
|  52 |
|  53 |
```

```
| 57 |
| 58 |
| ... |
| 98 |
| 99 |
| 128 |
| 129 |
| ... |
| 210 |
+-----+
```

> **警告**
>
> MySQL 5.5 より前では、ユーザー定義の照合順序 ID の範囲が用意されており、1 から 254 の範囲で ID を選択する必要があります。この場合、MySQL をアップグレードすると、選択した照合順序 ID が新しい MySQL 配布に含められた照合順序に割り当てられていることがわかります。この場合、自身の照合順序には新しい値を選択する必要があります。
>
> さらに、アップグレードする前に、変更する構成ファイルを保存する必要があります。適切にアップグレードすると、そのプロセスによって変更したファイルが置き換えられます。

10.4.3 8 ビットの文字セットへの単純な照合順序の追加

このセクションでは、MySQL Index.xml ファイル内の <charset> 文字セットの記述に関連付けられた <collation> 要素を書き込むことによって、8 ビット文字セットの単純な照合順序を追加する方法について説明します。ここで説明した手順では、MySQL の再コンパイルは不要です。この例では、latin1_test_ci という名前の照合順序を latin1 文字セットに追加します。

1. セクション10.4.2「照合順序 ID の選択」で示したように、照合順序 ID を選択します。次のステップでは、1024 の ID を使用します。

2. Index.xml および latin1.xml 構成ファイルを変更します。これらのファイルは、character_sets_dir システム変数によって名前の付けられたディレクトリに置かれます。使用しているシステムではパス名が異なることがありますが、次のようにして変数値を確認できます。

   ```
   mysql> SHOW VARIABLES LIKE 'character_sets_dir';
   +--------------------+----------------------------------------+
   | Variable_name      | Value                                  |
   +--------------------+----------------------------------------+
   | character_sets_dir | /user/local/mysql/share/mysql/charsets/ |
   +--------------------+----------------------------------------+
   ```

3. 照合順序の名前を選択して、Index.xml ファイルに表示します。照合順序を追加する文字セットの <charset> 要素を探し、照合順序名および ID を指定する <collation> 要素を追加して、名前を ID に関連付けます。例:

   ```
   <charset name="latin1">
     ...
     <collation name="latin1_test_ci" id="1024"/>
     ...
   </charset>
   ```

4. latin1.xml 構成ファイルで、照合順序に名前を付ける <collation> 要素と、0 から 255 の文字コードの文字コードと重みのマッピングテーブルを定義する <map> 要素を追加します。<map> 要素内のそれぞれの値は、16 進形式の数値にする必要があります。

   ```
   <collation name="latin1_test_ci">
   <map>
    00 01 02 03 04 05 06 07 08 09 0A 0B 0C 0D 0E 0F
    10 11 12 13 14 15 16 17 18 19 1A 1B 1C 1D 1E 1F
    20 21 22 23 24 25 26 27 28 29 2A 2B 2C 2D 2E 2F
    30 31 32 33 34 35 36 37 38 39 3A 3B 3C 3D 3E 3F
    40 41 42 43 44 45 46 47 48 49 4A 4B 4C 4D 4E 4F
    50 51 52 53 54 55 56 57 58 59 5A 5B 5C 5D 5E 5F
    60 41 42 43 44 45 46 47 48 49 4A 4B 4C 4D 4E 4F
    50 51 52 53 54 55 56 57 58 59 5A 7B 7C 7D 7E 7F
    80 81 82 83 84 85 86 87 88 89 8A 8B 8C 8D 8E 8F
    90 91 92 93 94 95 96 97 98 99 9A 9B 9C 9D 9E 9F
    A0 A1 A2 A3 A4 A5 A6 A7 A8 A9 AA AB AC AD AE AF
    B0 B1 B2 B3 B4 B5 B6 B7 B8 B9 BA BB BC BD BE BF
    41 41 41 41 5B 5D 5B 43 45 45 45 45 49 49 49 49
    44 4E 4F 4F 4F 4F 5C D7 5C 55 55 55 59 59 DE DF
    41 41 41 41 5B 5D 5B 43 45 45 45 45 49 49 49 49
   ```

```
 44 4E 4F 4F 4F 4F 5C F7 5C 55 55 55 59 59 DE FF
</map>
</collation>
```

5. サーバーを再起動し、このステートメントを使用して、照合順序の有無を検証します。

```
mysql> SHOW COLLATION LIKE 'latin1_test_ci';
+---------------+---------+------+---------+----------+---------+
| Collation     | Charset | Id   | Default | Compiled | Sortlen |
+---------------+---------+------+---------+----------+---------+
| latin1_test_ci | latin1  | 1024 |         |          |       1 |
+---------------+---------+------+---------+----------+---------+
```

10.4.4 Unicode 文字セットへの UCA 照合順序の追加

このセクションでは、MySQL Index.xml ファイルで <charset> 文字セットの記述内に <collation> 要素を書き込むことによって、Unicode 文字セットの UCA 照合順序を追加する方法について説明します。ここで説明した手順では、MySQL の再コンパイルは不要です。これは、http://www.unicode.org/reports/tr35/ で入手できる Locale Data Markup Language (LDML) 仕様のサブセットを使用します。この方法を使用すれば、照合順序全体を定義する必要はありません。代わりに、既存の「基本」照合順序から始め、基本照合順序とどのように異なるかという点で新しい照合順序について記述します。次の表は、UCA 照合順序を定義できる Unicode 文字セットの基本照合順序を一覧表示しています。utf16le のユーザー定義 UCA 照合順序は作成できません。このような照合順序のベースとして役立つ utf16le_unicode_ci 照合順序はありません。

表 10.1 ユーザー定義の UCA 照合順序に使用可能な MySQL 文字セット

文字セット	基本照合順序
utf8	utf8_unicode_ci
ucs2	ucs2_unicode_ci
utf16	utf16_unicode_ci
utf32	utf32_unicode_ci

以降のセクションでは、LDML 構文を使用して定義された照合順序を追加する方法について説明し、MySQL でサポートされている LDML ルールのサマリーを示します。

10.4.4.1 LDML 構文を使用した UCA 照合順序の定義

MySQL を再コンパイルせずに Unicode 文字セットの UCA 照合順序を追加するには、次の手順を使用します。照合順序のソート特性の記述に使用する LDML ルールを把握していない場合は、セクション10.4.4.2「MySQL でサポートされる LDML 構文」を参照してください。

この例では、utf8_phone_ci という名前の照合順序を utf8 文字セットを追加します。この照合順序は、ユーザーが名前と電話番号を投稿する Web アプリケーションに関連したシナリオ用に設計されています。電話番号は、次のようなさまざまな形式で指定できます。

```
+7-12345-67
+7-12-345-67
+7 12 345 67
+7 (12) 345 67
+71234567
```

このような値を扱うときに生じる問題は、さまざまな形式が許容されることで、特定の電話番号の検索が非常に困難になるということです。この解決方法として、句読点文字を並べ替えて無視できるようにする新しい照合順序を定義します。

1. セクション10.4.4.2「照合順序 ID の選択」で示したように、照合順序 ID を選択します。次のステップでは、1029 の ID を使用します。

2. Index.xml 構成ファイルを変更します。このファイルは、character_sets_dir システム変数によって名前の付けられたディレクトリに置かれます。使用しているシステムではパス名が異なることがありますが、次のようにして変数値を確認できます。

```
mysql> SHOW VARIABLES LIKE 'character_sets_dir';
+--------------------+----------------------------------------+
| Variable_name      | Value                                  |
+--------------------+----------------------------------------+
| character_sets_dir | /user/local/mysql/share/mysql/charsets/ |
+--------------------+----------------------------------------+
```

3. 照合順序の名前を選択して、Index.xml ファイルに表示します。さらに、照合順序の順序付けルールを提供する必要があります。照合順序を追加する文字セットの <charset> 要素を探し、照合順序名および ID を指定する <collation> 要素を追加して、名前を ID に関連付けます。<collation> 要素内で、順序付けルールを含む <rules> 要素を提供します。

```
<charset name="utf8">
 ...
 <collation name="utf8_phone_ci" id="1029">
  <rules>
   <reset>\u0000</reset>
   <i>\u0020</i> <!-- space -->
   <i>\u0028</i> <!-- left parenthesis -->
   <i>\u0029</i> <!-- right parenthesis -->
   <i>\u002B</i> <!-- plus -->
   <i>\u002D</i> <!-- hyphen -->
  </rules>
 </collation>
 ...
</charset>
```

4. その他の Unicode 文字セットに同様の照合順序が必要な場合は、ほかの <collation> 要素を追加します。たとえば、ucs2_phone_ci を定義するには、<collation> 要素を <charset name="ucs2"> 要素に追加します。それぞれの照合順序には一意の独自 ID が必要になります。

5. サーバーを再起動し、このステートメントを使用して、照合順序の有無を検証します。

```
mysql> SHOW COLLATION LIKE 'utf8_phone_ci';
+---------------+---------+------+---------+----------+---------+
| Collation     | Charset | Id   | Default | Compiled | Sortlen |
+---------------+---------+------+---------+----------+---------+
| utf8_phone_ci | utf8    | 1029 |         |          |       8 |
+---------------+---------+------+---------+----------+---------+
```

次に、照合順序をテストして、目的のプロパティーがあることを確認します。

新しい照合順序を使用して、いくつかのサンプルの電話番号を含むテーブルを作成します。

```
mysql> CREATE TABLE phonebook (
    ->   name VARCHAR(64),
    ->   phone VARCHAR(64) CHARACTER SET utf8 COLLATE utf8_phone_ci
    -> );
Query OK, 0 rows affected (0.09 sec)

mysql> INSERT INTO phonebook VALUES ('Svoj','+7 912 800 80 02');
Query OK, 1 row affected (0.00 sec)

mysql> INSERT INTO phonebook VALUES ('Hf','+7 (912) 800 80 04');
Query OK, 1 row affected (0.00 sec)

mysql> INSERT INTO phonebook VALUES ('Bar','+7-912-800-80-01');
Query OK, 1 row affected (0.00 sec)

mysql> INSERT INTO phonebook VALUES ('Ramil','(7912) 800 80 03');
Query OK, 1 row affected (0.00 sec)

mysql> INSERT INTO phonebook VALUES ('Sanja','+380 (912) 8008005');
Query OK, 1 row affected (0.00 sec)
```

いくつかのクエリーを実行して、無視された句読点文字が実際にソートおよび比較で無視されているかどうかを確認します。

```
mysql> SELECT * FROM phonebook ORDER BY phone;
+-------+--------------------+
| name  | phone              |
+-------+--------------------+
| Sanja | +380 (912) 8008005 |
| Bar   | +7-912-800-80-01   |
| Svoj  | +7 912 800 80 02   |
| Ramil | (7912) 800 80 03   |
| Hf    | +7 (912) 800 80 04 |
+-------+--------------------+
5 rows in set (0.00 sec)

mysql> SELECT * FROM phonebook WHERE phone='+7(912)800-80-01';
+------+-----------------+
| name | phone           |
+------+-----------------+
```

```
| Bar  | +7-912-800-80-01 |
+------+------------------+
1 row in set (0.00 sec)

mysql> SELECT * FROM phonebook WHERE phone='79128008001';
+------+------------------+
| name | phone            |
+------+------------------+
| Bar  | +7-912-800-80-01 |
+------+------------------+
1 row in set (0.00 sec)

mysql> SELECT * FROM phonebook WHERE phone='7 9 1 2 8 0 0 8 0 0 1';
+------+------------------+
| name | phone            |
+------+------------------+
| Bar  | +7-912-800-80-01 |
+------+------------------+
1 row in set (0.00 sec)
```

10.4.4.2 MySQL でサポートされる LDML 構文

このセクションでは、MySQL が認識する LDML 構文について説明します。これは、http://www.unicode.org/reports/tr35/ で入手できる LDML 仕様で説明されている構文のサブセットであり、詳細についてはこの仕様を参照してください。MySQL は、サイズの大きい構文のサブセットを認識するので、多くの場合、Unicode 共通ロケールデータリポジトリから照合順序定義をダウンロードして、該当する部分 (<rules> タグと </rules> タグの間の部分) を MySQL Index.xml ファイルに貼り付けることが可能です。ここで説明するルールは、文字のソートがプライマリレベルのみで行われることを除いて、すべてサポートされます。セカンダリ以上のソートレベルでの差を指定するルールは認識されますが (たとえば、照合順序の定義に含めることができます)、プライマリレベルでは等式として扱われます。

MySQL Server は、Index.xml ファイルの構文解析中に問題を検出すると、診断を生成します。セクション 10.4.4.3「Index.xml の構文解析中の診断」を参照してください。

文字表現

LDML ルールで名前が付けられた文字は、文字どおりに、または \unnnn 形式で書き込めます。ただし、nnnn は 16 進 Unicode コードポイント値です。たとえば、A と á は、文字どおりに、または \u0041 および \u00E1 として書き込めます。16 進値内で、A から F の桁は、大文字と小文字を区別しません。\u00E1 と \u00e1 は同等になります。UCA 4.0.0 照合順序の場合、16 進表記は、Basic Multilingual Plane の文字にのみ使用でき、0000 から FFFF の BMP 範囲外の文字には使用できません。UCA 5.2.0 照合順序の場合、16 進表記をすべての文字に使用できます。

Index.xml ファイル自体は、UTF-8 エンコーディングを使用して書き込む必要があります。

構文ルール

LDML には、文字の順序付けを指定するリセットルールとシフトルールがあります。順序付けは、アンカーポイントを確立するリセットルールから開始し、そのアンカーポイントを基準として文字をソートする方法を示すシフトルールが続く一連のルールとして指定されます。

- <reset> ルールはそれ自体ではどの順序付けも指定しません。その代わりにこれは、所定の文字に関連して後続のシフトルールを実行できるように、順序付けを「リセット」します。次のどちらのルールも、文字 'A' に関連して実行されるように後続のシフトルールをリセットします。

  ```
  <reset>A</reset>

  <reset>\u0041</reset>
  ```

- <p>、<s>、および <t> のシフトルールは、文字と文字とのプライマリ、セカンダリ、およびターシャリの差を定義します。

  - プライマリの差を使用して、個々の文字を区別します。
  - セカンダリの差を使用して、アクセントバリエーションを区別します。
  - ターシャリの差を使用して、大文字と小文字のバリエーションを区別します。

  次のどちらのルールも、'G' 文字のプライマリシフトルールを指定します。

  ```
  <p>G</p>
  ```

```
<p>\u0047</p>
```

- シフトルールは、ある文字が別の文字とまったく同じようにソートするよう指定します。次のルールは、'b' が 'a' と同じようにソートします。

```
<reset>a</reset>
<i>b</i>
```

- 略記されたシフト構文は、タグの単一のペアを使用して、複数のシフトルールを指定します。次の表は、略記された構文ルールと同等の略記されていないルールとの対応を示します。

  表 10.2 略記されたシフト構文

略記された構文	略記されていない構文
<pc>xyz</pc>	x y z
<sc>xyz</sc>	<s>x</s><s>y</s><s>z</s>
<tc>xyz</tc>	<t>x</t><t>y</t><t>z</t>
<ic>xyz</ic>	xyz

- 拡張形式は、複数文字のシーケンスのアンカーポイントを確立するリセットルールです。MySQL は 2 から 6 文字長の拡張形式をサポートしています。次のルールは、プライマリレベルで 'z' を 3 文字のシーケンス 'abc' よりも大きくします。

```
<reset>abc</reset>
<p>z</p>
```

- 短縮形式は、複数文字のシーケンスをソートするシフトルールです。MySQL は、2 から 6 文字長の短縮形式をサポートしています。次のルールは、プライマリレベルで 3 文字のシーケンス 'xyz' を 'a' よりも大きくします。

```
<reset>a</reset>
<p>xyz</p>
```

- 長い拡張形式と長い短縮形式を一緒に使用できます。次のルールは、プライマリレベルで 3 文字のシーケンス 'xyz' を 3 文字のシーケンス 'abc' よりも大きくします。

```
<reset>abc</reset>
<p>xyz</p>
```

- 通常の拡張形式の構文では、<x> と <extend> 要素を使用して、拡張形式を指定します。次のルールは、セカンダリレベルで文字 'k' をシーケンス 'ch' よりも大きくします。つまり、'k' は、'c' に 'h' が続いたあとの文字に拡張したかのように動作します。

```
<reset>c</reset>
<x><s>k</s><extend>h</extend></x>
```

  この構文は長いシーケンスを許可します。次のルールは、ターシャリレベルでシーケンス 'ccs' をシーケンス 'cscs' よりも大きくします。

```
<reset>cs</reset>
<x><t>ccs</t><extend>cs</extend></x>
```

  LDML 仕様では、通常の拡張形式構文を「慎重を要するもの」と記述しています。詳細についてはその仕様を参照してください。

- 前コンテキスト構文は、<x> および <context> 要素を使用して、文字の前のコンテキストによってソートが変更されるよう指定します。次のルールは、セカンダリレベルで '-' を 'a' よりも大きくしますが、これは '-' の前に 'b' があったときだけです。

```
<reset>a</reset>
<x><context>b</context><s>-</s></x>
```

- 前コンテキスト構文には、<extend> 要素を含めることができます。次のルールは、プライマリレベルで 'def' を 'aghi' よりも大きくしますが、これは 'def' の前に 'abc' があったときだけです。

```
<reset>a</reset>
<x><context>abc</context><p>def</p><extend>ghi</extend></x>
```

- リセットルールでは before 属性が許可されています。通常、リセットルールのあとのシフトルールは、リセット文字のあとにソートする文字を指定します。before 属性を伴うリセットルール後のシフトルールは、リセッ

ト文字の前にソートする文字を指定します。次のルールは、プライマリレベルで文字 'b' を 'a' の直前に配置します。

```
<reset before="primary">a</reset>
<p>b</p>
```

許容されている before 属性値は、名前または同等な数値でソートレベルを指定します。

```
<reset before="primary">
<reset before="1">

<reset before="secondary">
<reset before="2">

<reset before="tertiary">
<reset before="3">
```

- リセットルールでは、リテラル文字ではなく、論理リセット位置に名前を付けることができます。

```
<first_tertiary_ignorable/>
<last_tertiary_ignorable/>
<first_secondary_ignorable/>
<last_secondary_ignorable/>
<first_primary_ignorable/>
<last_primary_ignorable/>
<first_variable/>
<last_variable/>
<first_non_ignorable/>
<last_non_ignorable/>
<first_trailing/>
<last_trailing/>
```

次のルールは、プライマリレベルで 'z' を、Default Unicode Collation Element Table (DUCET) エントリを伴い、CJK ではない無視できない文字よりも大きくします。

```
<reset><last_non_ignorable/></reset>
<p>z</p>
```

論理位置には、次の表に示すコードポイントが設定されています。

表 10.3 論理リセット位置のコードポイント

論理位置	Unicode 4.0.0 コードポイント	Unicode 5.2.0 コードポイント
<first_non_ignorable/>	U+02D0	U+02D0
<last_non_ignorable/>	U+A48C	U+1342E
<first_primary_ignorable/>	U+0332	U+0332
<last_primary_ignorable/>	U+20EA	U+101FD
<first_secondary_ignorable/>	U+0000	U+0000
<last_secondary_ignorable/>	U+FE73	U+FE73
<first_tertiary_ignorable/>	U+0000	U+0000
<last_tertiary_ignorable/>	U+FE73	U+FE73
<first_trailing/>	U+0000	U+0000
<last_trailing/>	U+0000	U+0000
<first_variable/>	U+0009	U+0009
<last_variable/>	U+2183	U+1D371

- <collation> 要素は、シフトルールの文字重み計算に影響する shift-after-method 属性を許可します。この属性には、次の値が許可されています。
  - simple: before 属性を持たないリセットルールで、文字の重みを計算します。これは、属性が指定されない場合のデフォルトです。
  - expand: リセットルールのあとのシフトに拡張形式を使用します。

  '0' と '1' が 0E29 と 0E2A の重みを持ち、すべての基本ラテン文字を '0' から '1' の間に設定するとします。

```
<reset>0</reset>
```

```
<pc>abcdefghijklmnopqrstuvwxyz</pc>
```

単純なシフトモードの場合、重みは次のように計算されます。

```
'a' has weight 0E29+1
'b' has weight 0E29+2
'c' has weight 0E29+3
...
```

ただし、'0' から '1' の間には 26 個の文字を設定するのに十分な未使用の位置がありません。数字と文字が混在する結果になります。

これを解決するには shift-after-method="expand" を使用します。この場合、重みは次のように計算されます。

```
'a' has weight [0E29][233D+1]
'b' has weight [0E29][233D+2]
'c' has weight [0E29][233D+3]
...
```

233D は、文字 0xA48C の UCA 4.0.0 重みです。これは、最後の無視できない文字 (CJK を除き、照合順序で一種のもっとも大きな文字) です。UCA 5.2.0 も同様ですが、文字 0x1342E に 3ACA を使用します。

MySQL 固有の LDML 拡張機能

MySQL 5.6 では、LDML ルールの拡張機能によって、<collation> 要素は、照合順序が基づく UCA バージョンを示す version オプション属性を、<collation> タグに含めることができます。version 属性を省略すると、そのデフォルト値は 4.0.0 になります。たとえば、次の指定は、UCA 5.2.0 に基づいている照合順序を示します。

```
<collation id="nnn" name="utf8_xxx_ci" version="5.2.0">
...
</collation>
```

10.4.4.3 Index.xml の構文解析中の診断

MySQL Server は、Index.xml ファイルの構文解析中に問題が見つかったときに、診断を生成します。

- 不明なタグはエラーログに書き込まれます。たとえば、照合順序定義に <aaa> タグが含まれる場合、次のメッセージが表示されます。

```
[Warning] Buffered warning: Unknown LDML tag:
'charsets/charset/collation/rules/aaa'
```

- 照合順序の初期化が可能でない場合、サーバーは「不明な照合順序」エラーをレポートし、前の例のように問題点を説明した警告も生成します。それ以外の場合、照合順序の説明が全体的に正しいが、いくつかの不明なタグが含まれているときに、照合順序が初期化され、使用できます。不明な部分は無視されますが、警告がエラーログに生成されます。
- 照合順序の問題によって、クライアントが SHOW WARNINGS で表示できる警告が生成されます。6 文字のサポートされる最大長より長い拡張形式が、リセットルールに含まれているとします。

```
<reset>abcdefghi</reset>
<i>x</i>
```

この照合順序を使用しようとすると、次の警告が生成されます。

```
mysql> SELECT _utf8'test' COLLATE utf8_test_ci;
ERROR 1273 (HY000): Unknown collation: 'utf8_test_ci'
mysql> SHOW WARNINGS;
+---------+------+----------------------------------------+
| Level   | Code | Message                                |
+---------+------+----------------------------------------+
| Error   | 1273 | Unknown collation: 'utf8_test_ci'      |
| Warning | 1273 | Expansion is too long at 'abcdefghi=x' |
+---------+------+----------------------------------------+
```

10.5 文字セットの構成

デフォルトのサーバー文字セットおよび照合順序は、サーバー起動時に、--character-set-server オプションと --collation-server オプションで変更できます。照合順序は、デフォルト文字セットの正当な照合順序である必要があります。(各文字列セットに使用できる照合順序を特定するには、SHOW COLLATION ステートメントを使用します。)セクション5.1.3「サーバーコマンドオプション」を参照してください。

バイナリにコンパイルされない文字セットを使用しようとすると、次の問題が生じることがあります。

- 文字セットが格納される場所 (通常、MySQL インストールディレクトリ下の share/mysql/charsets または share/charsets ディレクトリ) を判断するときに、プログラムが不正なパスを使用します。これは、該当プログラムを実行するときに --character-sets-dir オプションを使用することによって修正できます。たとえば、MySQL クライアントプログラムで使用されるディレクトリを指定するには、オプションファイルの [client] グループに記述します。ここに挙げる例は、それぞれ Unix または Windows の場合に設定がどのようになるかを示します。

  ```
  [client]
  character-sets-dir=/usr/local/mysql/share/mysql/charsets

  [client]
  character-sets-dir="C:/Program Files/MySQL/MySQL Server 5.6/share/charsets"
  ```

- 文字セットは、動的にロードできない複雑な文字セットです。この場合、文字セットのサポートを使用してプログラムを再コンパイルする必要があります。

  Unicode 文字セットの場合、LDML 表記を使用することによって、再コンパイルせずに照合順序を定義できます。セクション10.4.4「Unicode 文字セットへの UCA 照合順序の追加」を参照してください。

- 文字セットは動的な文字セットですが、その構成ファイルがありません。この場合、新しい MySQL 配布から文字セットの構成ファイルをインストールする必要があります。

- 文字セットインデックスファイルに文字セットの名前が含まれていない場合、プログラムはエラーメッセージを表示します。このファイルの名前は Index.xml で、メッセージは次のとおりです。

  ```
  Character set 'charset_name' is not a compiled character set and is not
  specified in the '/usr/share/mysql/charsets/Index.xml' file
  ```

  この問題を解決するには、新しいインデックスファイルを取得するか、欠落している文字セットの名前を手動で現在のファイルに追加する必要があります。

次のようにしてクライアントプログラムに強制的に特定の文字セットを使用させることができます。

```
[client]
default-character-set=charset_name
```

これは通常は不要です。ただし、character_set_system が character_set_server または character_set_client と異なり、(データベースオブジェクト識別子またはカラム値、あるいはその両方として) 手動で文字を入力した場合、これらの文字はクライアントからの出力に間違って表示されたり、出力自体が間違って書式設定されたりすることがあります。このような場合、--default-character-set=system_character_set を使用して MySQL クライアントを起動し、システム文字セットに一致するようにクライアント文字セットを設定すると、問題が修正されます。

MyISAM テーブルでは、myisamchk -dvv tbl_name を使用するとテーブルの文字セットの名前および数値を確認できます。

10.6 MySQL Server でのタイムゾーンのサポート

MySQL Server は複数のタイムゾーン設定を保持しています。

- システムタイムゾーン。サーバーは、起動するときに、ホストマシンのタイムゾーンを特定し、これを使用して system_time_zone システム変数で設定しようとします。その後、この値は変更しません。

  mysqld_safe で --timezone=timezone_name オプションを使用すると、起動時に MySQL Server のシステムタイムゾーンを設定できます。mysqld を起動する前に、TZ 環境変数を設定することによって設定することもできます。--timezone または TZ に許可される値は、システムによって異なります。許容可能な値を確認するには、オペレーティングシステムのドキュメントを参照してください。

- サーバーの現在のタイムゾーン。time_zone グローバルシステム変数は、サーバーが現在動作しているタイムゾーンを示します。time_zone の初期値は 'SYSTEM' であり、これはサーバーのタイムゾーンがシステムタイムゾーンと同じであることを示します。

  コマンド行で --default-time-zone=timezone オプションを使用すると、初期グローバルサーバータイムゾーン値を起動時に明示的に指定できます。または、オプションファイルで次の行を使用できます。

  ```
  default-time-zone='timezone'
  ```

  SUPER 権限がある場合は、実行時に次のステートメントを使用すると、サーバータイムゾーンのグローバル値を設定できます。

```
mysql> SET GLOBAL time_zone = timezone;
```

- 接続ごとのタイムゾーン。接続するそれぞれのクライアントには、time_zone セッション変数で指定された、それぞれのタイムゾーン設定があります。最初、セッション変数は、time_zone グローバル変数から値を取得しますが、クライアントは次のステートメントを使用して、それぞれのタイムゾーンを変更できます。

```
mysql> SET time_zone = timezone;
```

現在のセッションのタイムゾーン設定は、ゾーンを区別する時間値の表示とストレージに影響します。これには、NOW() や CURTIME() などの関数で表示される値や、TIMESTAMP カラムに保存し、そこから読み出す値も含まれます。TIMESTAMP カラムの値は、ストレージでは現在のタイムゾーンから UTC に、読み出しでは UTC から現在のタイムゾーンに変換されます。

現在のタイムゾーン設定は、UTC_TIMESTAMP() 関数などによって表示される値、または DATE、TIME、DATETIME カラムの値には影響しません。また、これらのデータ型の値も UTC で格納されません。タイムゾーンは、TIMESTAMP 値から変換するときにのみ適用されます。DATE、TIME、または DATETIME 値に対してロケール固有の演算を実行する場合、これらの値を UTC に変換し、演算を実行してから、元に変換し直します。

グローバルおよびクライアント固有のタイムゾーンの現在値は、次のように取得できます。

```
mysql> SELECT @@GLOBAL.time_zone, @@SESSION.time_zone;
```

timezone 値は、次の複数の形式で指定でき、大文字と小文字を区別しません。

- 'SYSTEM' 値は、タイムゾーンをシステムタイムゾーンと同じにする必要があることを示します。
- この値は、'+10:00' や '-6:00' など、UTC からのオフセットを示す文字列として指定できます。
- この値は、'Europe/Helsinki'、'US/Eastern'、'MET' などの名前付きのタイムゾーンとして指定できます。名前付きのタイムゾーンは、mysql データベース内のタイムゾーン情報テーブルが作成され移入されている場合にのみ使用できます。

MySQL のインストール手順では、mysql データベース内にタイムゾーンテーブルを作成しますが、これらをロードしません。次の手順を使用して、手動でこれを行う必要があります。(以前のバージョンから MySQL 4.1.3 以降にアップグレードする場合は、mysql データベースをアップグレードすることでテーブルを作成できます。セクション4.4.7「mysql_upgrade — MySQL テーブルのチェックとアップグレード」の手順を使用してください。テーブルを作成したあとでロードできます。)

> **注記**
>
> 情報は変更することがあるので、タイムゾーン情報のロードは必ずしも 1 回だけの操作とはかぎりません。たとえば、米国、メキシコ、一部のカナダのサマータイムのルールは、2007 年に変更されました。このような変更が起きた場合、古いルールを使用したアプリケーションは旧式になり、MySQL Server で使用されている情報を最新の状態に維持するために、タイムゾーンテーブルをリロードする必要が生じることがあります。このセクションの最後のノートを参照してください。

システムに独自の zoneinfo データベース (タイムゾーンについて述べたファイルセット) がある場合、タイムゾーンテーブルに入力するために mysql_tzinfo_to_sql プログラムを使用する必要があります。このようなシステムには、Linux、FreeBSD、Solaris、OS X などがあります。これらのファイルの 1 つの適切な場所は /usr/share/zoneinfo ディレクトリです。システムに zoneinfo データベースがない場合、このセクションで後述するダウンロード可能なパッケージを使用できます。

mysql_tzinfo_to_sql プログラムはタイムゾーンテーブルのロードに使用されます。コマンド行で、zoneinfo ディレクトリのパス名を mysql_tzinfo_to_sql に渡し、出力を mysql プログラムに送信します。例:

```
shell> mysql_tzinfo_to_sql /usr/share/zoneinfo | mysql -u root mysql
```

mysql_tzinfo_to_sql は、システムのタイムゾーンファイルを読み取り、そのファイルから SQL ステートメントを生成します。mysql はこれらのステートメントを処理して、タイムゾーンテーブルをロードします。

mysql_tzinfo_to_sql を使用すると、単一のタイムゾーンファイルをロードしたり、うるう秒情報を生成したりできます。

- タイムゾーン名 tz_name に対応した単一のタイムゾーンファイル tz_file をロードするには、次のように mysql_tzinfo_to_sql を呼び出します。

```
shell> mysql_tzinfo_to_sql tz_file tz_name | mysql -u root mysql
```

このアプローチでは、サーバーが認識する名前付きゾーンごとに、個別のコマンドを実行してタイムゾーンファイルをロードする必要があります。

- タイムゾーンがうるう秒に対応する必要がある場合は、次のようにうるう秒の情報を初期化します。ここで tz_file はタイムゾーンファイルの名前です。

```
shell> mysql_tzinfo_to_sql --leap tz_file | mysql -u root mysql
```

- mysql_tzinfo_to_sql の実行後、以前にキャッシュしたすべてのタイムゾーンデータを使用し続けないように、サーバーを再起動することをお勧めします。

zoneinfo データベース (Windows など) を含まないシステムの場合、MySQL Developer Zone でダウンロードに使用できる事前に構築されたタイムゾーンテーブルのパッケージを使用できます。

```
https://dev.mysql.com/downloads/timezones.html
```

このタイムゾーンパッケージには、MyISAM のタイムゾーンテーブル用に、.frm、.MYD、.MYI などのファイルが含まれています。これらのテーブルを mysql データベースの一部にする必要があるため、ファイルを MySQL Server のデータディレクトリの mysql サブディレクトリに入れてください。サーバーを停止してからこれを行い、そのあとで再起動します。

> **警告**
>
> システムに zoneinfo データベースがある場合は、ダウンロード可能なパッケージを使用しないでください。代わりに、mysql_tzinfo_to_sql ユーティリティーを使用してください。そうしないと、MySQL とシステム上のほかのアプリケーション間で日時処理に違いが生じることがあります。

レプリケーションセットアップのタイムゾーン設定の詳細は、セクション17.4.1「レプリケーションの機能と問題」を参照してください。

10.6.1 タイムゾーンの変更による現在の時間の維持

前述のように、タイムゾーンルールが変更すると、古いルールを使用するアプリケーションが旧式になります。現在の時間に維持するには、システムが現在のタイムゾーン情報を使用していることを確認する必要があります。MySQL の場合、現在の時間に維持するために考慮する 2 つの要素があります。

- オペレーティングシステムの時間は、そのタイムゾーンが SYSTEM に設定されている場合、MySQL Server が時間に使用する値に影響します。オペレーティングシステムが最新のタイムゾーン情報を使用していることを確認します。ほとんどのオペレーティングシステムでは、最新の更新またはサービスパックによってシステムは時間の変更に対応できます。時間の変更に対処した更新については、オペレーティングシステムベンダーの Web サイトを確認してください。

- システムの /etc/localtime タイムゾーンファイルを、mysqld の起動時に有効なものとは異なるルールを使用するバージョンに置き換えた場合、更新したルールを使用するように mysqld を再起動する必要があります。そうしないと、システムが時間を変更したときに mysqld が認識されないことがあります。

- MySQL で名前付きのタイムゾーンを使用する場合、mysql データベース内のタイムゾーンテーブルが最新になっていることを確認します。システムに独自の zoneinfo データベースがある場合は、zoneinfo データベースを更新するたびに、このセクションで前述した手順を使用して MySQL タイムゾーンテーブルをリロードする必要があります。システムに独自の zoneinfo データベースがない場合、MySQL Developer Zone で更新がないか調べます。新しい更新が利用できる場合は、これをダウンロードし使用して、現在のタイムゾーンテーブルを置き換えます。mysqld は、検索したタイムゾーン情報をキャッシュするので、タイムゾーンテーブルを置き換えたあとは、mysqld を再起動して、古くなったタイムゾーンデータを提供し続けないようにする必要があります。

サーバーのタイムゾーン設定として使用するか、独自のタイムゾーンを設定するクライアントが使用するために、名前付きタイムゾーンが使用できるかどうか不確かな場合は、タイムゾーンテーブルが空かどうかを調べてください。次のクエリーは、タイムゾーン名を含むテーブルに行があるかどうかを判断します。

```
mysql> SELECT COUNT(*) FROM mysql.time_zone_name;
+----------+
| COUNT(*) |
+----------+
|        0 |
+----------+
```

カウントがゼロの場合、テーブルが空であることを示します。この場合、名前付きタイムゾーンを使用できず、テーブルを更新する必要はありません。カウントがゼロより大きい場合、テーブルは空ではなく、その内容が名前付きタイムゾーンのサポートに使用できることを示します。この場合、名前付きタイムゾーンを使用するユーザーが、正しいクエリー結果が得られるように、必ずタイムゾーンテーブルをリロードしてください。

サマータイムのルール変更に対して MySQL インストールが正しく更新されているかどうかを確認するには、次のようなテストを使用します。この例では、3 月 11 日午前 2 時に米国で行われる 2007 年 DST の 1 時間の変更に適切な値を使用しています。

テストでは次の 2 つのクエリーを使用します。

```
SELECT CONVERT_TZ('2007-03-11 2:00:00','US/Eastern','US/Central');
SELECT CONVERT_TZ('2007-03-11 3:00:00','US/Eastern','US/Central');
```

2 つの時間値は、DST 変更が行われる時間を示し、名前付きタイムゾーンの使用には、タイムゾーンテーブルを使用する必要がにあります。結果では、両方のクエリーで同じ結果が返されることが期待されます (「米国/中央」タイムゾーンの同等の値に変換された入力時間)。

タイムゾーンテーブルを更新する前に、次のような正しくない結果が表示されます。

```
mysql> SELECT CONVERT_TZ('2007-03-11 2:00:00','US/Eastern','US/Central');
+-----------------------------------------------------------+
| CONVERT_TZ('2007-03-11 2:00:00','US/Eastern','US/Central') |
+-----------------------------------------------------------+
| 2007-03-11 01:00:00                                       |
+-----------------------------------------------------------+

mysql> SELECT CONVERT_TZ('2007-03-11 3:00:00','US/Eastern','US/Central');
+-----------------------------------------------------------+
| CONVERT_TZ('2007-03-11 3:00:00','US/Eastern','US/Central') |
+-----------------------------------------------------------+
| 2007-03-11 02:00:00                                       |
+-----------------------------------------------------------+
```

テーブルの更新後に、正しい結果が表示されます。

```
mysql> SELECT CONVERT_TZ('2007-03-11 2:00:00','US/Eastern','US/Central');
+-----------------------------------------------------------+
| CONVERT_TZ('2007-03-11 2:00:00','US/Eastern','US/Central') |
+-----------------------------------------------------------+
| 2007-03-11 01:00:00                                       |
+-----------------------------------------------------------+

mysql> SELECT CONVERT_TZ('2007-03-11 3:00:00','US/Eastern','US/Central');
+-----------------------------------------------------------+
| CONVERT_TZ('2007-03-11 3:00:00','US/Eastern','US/Central') |
+-----------------------------------------------------------+
| 2007-03-11 01:00:00                                       |
+-----------------------------------------------------------+
```

10.6.2 タイムゾーンのうるう秒のサポート

うるう秒値は、:59:59 で終わる時間部分を使用して返されます。これは、NOW() などの関数が、うるう秒の間、2、3 秒連続して同じ値を返すことがあることを意味します。:59:60 または :59:61 で終わる時間部分を持つリテラル時間値が無効と見なされることに変わりはありません。

うるう秒の 1 秒前の TIMESTAMP 値を検索する必要がある場合、'YYYY-MM-DD hh:mm:ss' 値での比較を使用すると異常な結果が得られることがあります。この点について次の例で説明します。ローカルタイムゾーンを UTC に変更するので、内部値 (UTC での値) と表示値 (タイムゾーン修正が適用された値) の差がなくなります。

```
mysql> CREATE TABLE t1 (
    ->   a INT,
    ->   ts TIMESTAMP DEFAULT NOW(),
    ->   PRIMARY KEY (ts)
    -> );
Query OK, 0 rows affected (0.01 sec)

mysql> -- change to UTC
mysql> SET time_zone = '+00:00';
Query OK, 0 rows affected (0.00 sec)

mysql> -- Simulate NOW() = '2008-12-31 23:59:59'
mysql> SET timestamp = 1230767999;
Query OK, 0 rows affected (0.00 sec)
```

```
mysql> INSERT INTO t1 (a) VALUES (1);
Query OK, 1 row affected (0.00 sec)

mysql> -- Simulate NOW() = '2008-12-31 23:59:60'
mysql> SET timestamp = 1230768000;
Query OK, 0 rows affected (0.00 sec)

mysql> INSERT INTO t1 (a) VALUES (2);
Query OK, 1 row affected (0.00 sec)

mysql> -- values differ internally but display the same
mysql> SELECT a, ts, UNIX_TIMESTAMP(ts) FROM t1;
+------+---------------------+--------------------+
| a    | ts                  | UNIX_TIMESTAMP(ts) |
+------+---------------------+--------------------+
|    1 | 2008-12-31 23:59:59 |         1230767999 |
|    2 | 2008-12-31 23:59:59 |         1230768000 |
+------+---------------------+--------------------+
2 rows in set (0.00 sec)

mysql> -- only the non-leap value matches
mysql> SELECT * FROM t1 WHERE ts = '2008-12-31 23:59:59';
+------+---------------------+
| a    | ts                  |
+------+---------------------+
|    1 | 2008-12-31 23:59:59 |
+------+---------------------+
1 row in set (0.00 sec)

mysql> -- the leap value with seconds=60 is invalid
mysql> SELECT * FROM t1 WHERE ts = '2008-12-31 23:59:60';
Empty set, 2 warnings (0.00 sec)
```

これを回避するには、うるう秒の修正が適用されている、実際にカラムに格納されている UTC 値に基づいた比較を使用します。

```
mysql> -- selecting using UNIX_TIMESTAMP value return leap value
mysql> SELECT * FROM t1 WHERE UNIX_TIMESTAMP(ts) = 1230768000;
+------+---------------------+
| a    | ts                  |
+------+---------------------+
|    2 | 2008-12-31 23:59:59 |
+------+---------------------+
1 row in set (0.00 sec)
```

10.7 MySQL Server のロケールサポート

lc_time_names システム変数で示されたロケールは、曜日および月の名前と短縮形を表示するために使用する言語を制御します。この変数は DATE_FORMAT()、DAYNAME()、および MONTHNAME() 関数の出力に影響を与えます。

lc_time_names は、STR_TO_DATE() または GET_FORMAT() 関数には影響しません。

lc_time_names 値は、FORMAT() の結果に影響しませんが、この関数は、結果の数値の小数点、桁区切り、および区切り文字のグルーピングに使用するロケールを指定できるようにするオプションの 3 番目のパラメータを取ります。許可されるロケール値は、lc_time_names システム変数の正当な値と同じです。

ロケール名には、'ja_JP' や 'pt_BR' など、IANA (http://www.iana.org/assignments/language-subtag-registry) に記載された言語および地域のサブタグが含まれます。システムのロケール設定とは無関係にデフォルト値は 'en_US' ですが、サーバーの起動時に値を設定することも、SUPER 権限がある場合には GLOBAL 値を設定することもできます。どのクライアントでも、lc_time_names の値を調べたり、その SESSION 値を設定してそれ自体の接続用のロケールに影響を与えたりできます。

```
mysql> SET NAMES 'utf8';
Query OK, 0 rows affected (0.09 sec)

mysql> SELECT @@lc_time_names;
+-----------------+
| @@lc_time_names |
+-----------------+
| en_US           |
+-----------------+
1 row in set (0.00 sec)

mysql> SELECT DAYNAME('2010-01-01'), MONTHNAME('2010-01-01');
```

```
+----------------------+------------------------+
| DAYNAME('2010-01-01') | MONTHNAME('2010-01-01') |
+----------------------+------------------------+
| Friday               | January                |
+----------------------+------------------------+
1 row in set (0.00 sec)

mysql> SELECT DATE_FORMAT('2010-01-01','%W %a %M %b');
+-----------------------------------------+
| DATE_FORMAT('2010-01-01','%W %a %M %b') |
+-----------------------------------------+
| Friday Fri January Jan                  |
+-----------------------------------------+
1 row in set (0.00 sec)

mysql> SET lc_time_names = 'es_MX';
Query OK, 0 rows affected (0.00 sec)

mysql> SELECT @@lc_time_names;
+----------------+
| @@lc_time_names |
+----------------+
| es_MX          |
+----------------+
1 row in set (0.00 sec)

mysql> SELECT DAYNAME('2010-01-01'), MONTHNAME('2010-01-01');
+----------------------+------------------------+
| DAYNAME('2010-01-01') | MONTHNAME('2010-01-01') |
+----------------------+------------------------+
| viernes              | enero                  |
+----------------------+------------------------+
1 row in set (0.00 sec)

mysql> SELECT DATE_FORMAT('2010-01-01','%W %a %M %b');
+-----------------------------------------+
| DATE_FORMAT('2010-01-01','%W %a %M %b') |
+-----------------------------------------+
| viernes vie enero ene                   |
+-----------------------------------------+
1 row in set (0.00 sec)
```

影響を受けた関数それぞれの曜日または月の名前は、utf8 から character_set_connection システム変数で指定される文字セットに変換されます。

lc_time_names は次のどの口ケール値にも設定できます。MySQL でサポートされる口ケールのセットは、オペレーティングシステムでサポートされるものと異なる場合があります。

ar_AE: アラビア語 - アラブ首長国連邦	ar_BH: アラビア語 - バーレーン
ar_DZ: アラビア語 - アルジェリア	ar_EG: アラビア語 - エジプト
ar_IN: アラビア語 - インド	ar_IQ: アラビア語 - イラク
ar_JO: アラビア語 - ヨルダン	ar_KW: アラビア語 - クウェート
ar_LB: アラビア語 - レバノン	ar_LY: アラビア語 - リビア
ar_MA: アラビア語 - モロッコ	ar_OM: アラビア語 - オマーン
ar_QA: アラビア語 - カタール	ar_SA: アラビア語 - サウジアラビア
ar_SD: アラビア語 - スーダン	ar_SY: アラビア語 - シリア
ar_TN: アラビア語 - チュニジア	ar_YE: アラビア語 - イエメン
be_BY: ベラルーシ語 - ベラルーシ	bg_BG: ブルガリア語 - ブルガリア
ca_ES: カタロニア語 - スペイン	cs_CZ: チェコ語 - チェコ
da_DK: デンマーク語 - デンマーク	de_AT: ドイツ語 - オーストリア
de_BE: ドイツ語 - ベルギー	de_CH: ドイツ語 - スイス
de_DE: ドイツ語 - ドイツ	de_LU: ドイツ語 - ルクセンブルク
el_GR: ギリシャ語 - ギリシャ	en_AU: 英語 - オーストラリア
en_CA: 英語 - カナダ	en_GB: 英語 - イギリス
en_IN: 英語 - インド	en_NZ: 英語 - ニュージーランド
en_PH: 英語 - フィリピン	en_US: 英語 - 米国

en_ZA: 英語 - 南アフリカ	en_ZW: 英語 - ジンバブエ
es_AR: スペイン語 - アルゼンチン	es_BO: スペイン語 - ボリビア
es_CL: スペイン語 - チリ	es_CO: スペイン語 - コロンビア
es_CR: スペイン語 - コスタリカ	es_DO: スペイン語 - ドミニカ共和国
es_EC: スペイン語 - エクアドル	es_ES: スペイン語 - スペイン
es_GT: スペイン語 - グアテマラ	es_HN: スペイン語 - ホンジュラス
es_MX: スペイン語 - メキシコ	es_NI: スペイン語 - ニカラグア
es_PA: スペイン語 - パナマ	es_PE: スペイン語 - ペルー
es_PR: スペイン語 - プエルトリコ	es_PY: スペイン語 - パラグアイ
es_SV: スペイン語 - エルサルバドル	es_US: スペイン語 - 米国
es_UY: スペイン語 - ウルグアイ	es_VE: スペイン語 - ベネズエラ
et_EE: エストニア語 - エストニア	eu_ES: バスク語 - バスク
fi_FI: フィンランド語 - フィンランド	fo_FO: フェーロー語 - フェロー諸島
fr_BE: フランス語 - ベルギー	fr_CA: フランス語 - カナダ
fr_CH: フランス語 - スイス	fr_FR: フランス語 - フランス
fr_LU: フランス語 - ルクセンブルク	gl_ES: ガリシア語 - スペイン
gu_IN: グジャラート語 - インド	he_IL: ヘブライ語 - イスラエル
hi_IN: ヒンディー語 - インド	hr_HR: クロアチア語 - クロアチア
hu_HU: ハンガリー語 - ハンガリー	id_ID: インドネシア語 - インドネシア
is_IS: アイスランド語 - アイスランド	it_CH: イタリア語 - スイス
it_IT: イタリア語 - イタリア	ja_JP: 日本語 - 日本
ko_KR: 韓国語 - 韓国	lt_LT: リトアニア語 - リトアニア
lv_LV: ラトビア語 - ラトビア	mk_MK: マケドニア語 - マケドニア
mn_MN: モンゴル語 - モンゴル	ms_MY: マレー語 - マレーシア
nb_NO: ノルウェー語 (ブークモール) - ノルウェー	nl_BE: オランダ語 - ベルギー
nl_NL: オランダ語 - オランダ	no_NO: ノルウェー語 - ノルウェー
pl_PL: ポーランド語 - ポーランド	pt_BR: ポルトガル語 - ブラジル
pt_PT: ポルトガル語 - ポルトガル	rm_CH: ロマンシュ語 - スイス
ro_RO: ルーマニア語 - ルーマニア	ru_RU: ロシア語 - ロシア
ru_UA: ロシア語 - ウクライナ	sk_SK: スロバキア語 - スロバキア
sl_SI: スロベニア語 - スロベニア	sq_AL: アルバニア語 - アルバニア
sr_RS: セルビア語 - ユーゴスラビア	sv_FI: スウェーデン語 - フィンランド
sv_SE: スウェーデン語 - スウェーデン	ta_IN: タミル語 - インド
te_IN: テルグ語 - インド	th_TH: タイ語 - タイ
tr_TR: トルコ語 - トルコ	uk_UA: ウクライナ語 - ウクライナ
ur_PK: ウルドゥー語 - パキスタン	vi_VN: ベトナム語 - ベトナム
zh_CN: 中国語 - 中国	zh_HK: 中国語 - 香港
zh_TW: 中国語 - 台湾	

第 11 章 データ型

目次

11.1 データ型の概要 1011
11.1.1 数値型の概要 1011
11.1.2 日付と時間型の概要 1014
11.1.3 文字列型の概要 1016
11.2 数値型 1019
11.2.1 整数型 (真数値) - INTEGER、INT、SMALLINT、TINYINT、MEDIUMINT、BIGINT 1019
11.2.2 固定小数点型 (真数値) - DECIMAL、NUMERIC 1020
11.2.3 浮動小数点型 (概数値) - FLOAT、DOUBLE 1020
11.2.4 ビット値型 - BIT 1020
11.2.5 数値型の属性 1021
11.2.6 範囲外およびオーバーフローの処理 1021
11.3 日付と時間型 1022
11.3.1 DATE、DATETIME、および TIMESTAMP 型 1023
11.3.2 TIME 型 1025
11.3.3 YEAR 型 1025
11.3.4 YEAR(2) の制限と YEAR(4) への移行 1026
11.3.5 TIMESTAMP および DATETIME の自動初期化および更新機能 1028
11.3.6 時間値での小数秒 1031
11.3.7 日付と時間型間での変換 1032
11.3.8 日付での 2 桁の年 1033
11.4 文字列型 1033
11.4.1 CHAR および VARCHAR 型 1033
11.4.2 BINARY および VARBINARY 型 1035
11.4.3 BLOB 型と TEXT 型 1036
11.4.4 ENUM 型 1037
11.4.5 SET 型 1040
11.5 空間データの拡張 1041
11.5.1 空間データ型 1043
11.5.2 OpenGIS 幾何モデル 1043
11.5.3 空間データの使用 1048
11.6 データ型デフォルト値 1054
11.7 データ型のストレージ要件 1055
11.8 カラムに適した型の選択 1058
11.9 その他のデータベースエンジンのデータ型の使用 1058

MySQL では、数値型、日付と時間型、文字列 (文字およびバイト) 型、空間型という複数のカテゴリにわたる多数の SQL データ型をサポートしています。この章では、これらのデータ型の概要、各カテゴリの型のプロパティーに関する詳細、およびデータ型ストレージ要件のサマリーについて説明します。最初の概要は意図的に簡単なものにしています。値を指定可能な許可される形式など、特定のデータ型に関する追加情報については、この章で後述する詳細な説明を参照してください。

データ型の説明では、次の規則を使用しています。

- M は整数型の最大表示幅を示します。浮動小数点型と固定小数点型の場合、M は格納可能な桁数の合計 (精度) です。文字列型の場合は、M は最大長です。M の許可される最大値は、データ型によって異なります。
- D は、浮動小数点型と固定小数点型に適用され、小数点以下の桁数 (スケール) を表します。指定可能な最大値は 30 ですが、M−2 以下にしてください。
- fsp は、TIME、DATETIME、および TIMESTAMP 型に適用され、小数秒の精度、つまり、秒の小数部における小数点以下の桁数を表します。fsp 値を指定する場合、0 から 6 の範囲にする必要があります。0 の値は、小数部がないことを表します。省略した場合、デフォルトの精度は 0 です。(これは、以前の MySQL バージョンと互換性を保つため、標準 SQL のデフォルトである 6 とは異なっています。)
- 角括弧 (「[」と「]」) は型定義のオプションの部分を表します。

11.1 データ型の概要

11.1.1 数値型の概要

数値データ型のサマリーについて説明します。数値型のプロパティーおよびストレージ要件の追加情報については、セクション11.2「数値型」およびセクション11.7「データ型のストレージ要件」を参照してください。

M は整数型の最大表示幅を示します。最大表示幅は 255 です。セクション11.2「数値型」で説明しているように、表示幅はその型に含めることができる値の範囲とは関係ありません。浮動小数点型と固定小数点型の場合、M は格納可能な桁数の合計です。

数値カラムに対して ZEROFILL を指定すると、MySQL は自動的にそのカラムに UNSIGNED 属性を追加します。

UNSIGNED 属性を許可している数値データ型は、SIGNED も許可します。ただし、このデータ型はデフォルトで符号付きになっているため、SIGNED 属性を指定しても効果はありません。

SERIAL は BIGINT UNSIGNED NOT NULL AUTO_INCREMENT UNIQUE のエイリアスです。

整数カラム定義の中の SERIAL DEFAULT VALUE は NOT NULL AUTO_INCREMENT UNIQUE のエイリアスです。

> **警告**
>
> 一方が UNSIGNED 型のときに 2 つの整数値の間で減算を行うと、NO_UNSIGNED_SUBTRACTION SQL モードが有効でないかぎり、結果の値は符号なしになります。セクション12.10「キャスト関数と演算子」を参照してください。

- BIT[(M)]

  ビットフィールド型。M は、値あたりのビット数 (1 から 64) を表します。M を省略した場合のデフォルトは 1 です。

- TINYINT[(M)] [UNSIGNED] [ZEROFILL]

  非常に小さい整数。符号付きの範囲は -128 から 127 です。符号なしの範囲は 0 から 255 です。

- BOOL、BOOLEAN

  これらの型は TINYINT(1) のシノニムです。ゼロの値は false と見なされます。ゼロ以外の値は true と見なされます。

```
mysql> SELECT IF(0, 'true', 'false');
+----------------------+
| IF(0, 'true', 'false') |
+----------------------+
| false                |
+----------------------+

mysql> SELECT IF(1, 'true', 'false');
+----------------------+
| IF(1, 'true', 'false') |
+----------------------+
| true                 |
+----------------------+

mysql> SELECT IF(2, 'true', 'false');
+----------------------+
| IF(2, 'true', 'false') |
+----------------------+
| true                 |
+----------------------+
```

ただし、ここに示されているように、TRUE 値と FALSE 値はそれぞれ、1 と 0 の単なるエイリアスです。

```
mysql> SELECT IF(0 = FALSE, 'true', 'false');
+--------------------------------+
| IF(0 = FALSE, 'true', 'false') |
+--------------------------------+
| true                           |
+--------------------------------+

mysql> SELECT IF(1 = TRUE, 'true', 'false');
+------------------------------+
| IF(1 = TRUE, 'true', 'false') |
+------------------------------+
| true                         |
+------------------------------+
```

```
mysql> SELECT IF(2 = TRUE, 'true', 'false');
+--------------------------------+
| IF(2 = TRUE, 'true', 'false') |
+--------------------------------+
| false                          |
+--------------------------------+

mysql> SELECT IF(2 = FALSE, 'true', 'false');
+---------------------------------+
| IF(2 = FALSE, 'true', 'false') |
+---------------------------------+
| false                           |
+---------------------------------+
```

最後の 2 つのステートメントは、2 が 1 とも 0 とも等しくないために示される結果を表示します。

- SMALLINT[(M)] [UNSIGNED] [ZEROFILL]

  小さい整数。符号付きの範囲は -32768 から 32767 です。符号なしの範囲は 0 から 65535 です。

- MEDIUMINT[(M)] [UNSIGNED] [ZEROFILL]

  中間サイズの整数。符号付きの範囲は -8388608 から 8388607 です。符号なしの範囲は 0 から 16777215 です。

- INT[(M)] [UNSIGNED] [ZEROFILL]

  普通サイズの整数。符号付きの範囲は -2147483648 から 2147483647 です。符号なしの範囲は 0 から 4294967295 です。

- INTEGER[(M)] [UNSIGNED] [ZEROFILL]

  この型は INT のシノニムです。

- BIGINT[(M)] [UNSIGNED] [ZEROFILL]

  大きい整数。符号付きの範囲は -9223372036854775808 から 9223372036854775807 です。符号なしの範囲は 0 から 18446744073709551615 です。

  SERIAL は BIGINT UNSIGNED NOT NULL AUTO_INCREMENT UNIQUE のエイリアスです。

  BIGINT カラムについて注意の必要な点は、次のとおりです。

  - すべての演算は符号付きの BIGINT 値または DOUBLE 値を使用して行われるため、ビット関数を使用しないかぎり、9223372036854775807 (63 ビット) よりも大きい符号なしの整数を使用しないでください。そのようにした場合、BIGINT 値から DOUBLE 値への変換時に、丸め誤差のために結果の最後の数桁に誤差が生じる可能性があります。

    MySQL は、次の場合に、BIGINT を扱うことができます。

    - 符号なしの大きな値を BIGINT カラムに格納するために整数を使用するとき。
    - MIN(col_name) または MAX(col_name) 内。ここで col_name は BIGINT カラムを指します。
    - 演算子 (+、-、* など) を使用する場合。ここで両方のオペランドは整数です。

  - 文字列を使用して格納すると、いつでも正確な整数値を BIGINT カラムに格納できます。この場合、MySQL は、中間倍精度表現を含まない文字列から数値に変換します。

  - 両方のオペランドが整数値の場合、-、+、および * の演算子は、BIGINT 演算を使用します。これは、2 つの大きい整数 (または整数を返す関数からの結果) を掛け合わした場合、その結果が 9223372036854775807 より大きいときには、予期しない結果になるということを意味します。

- DECIMAL[(M[,D])] [UNSIGNED] [ZEROFILL]

  パックされた「正確な」固定小数点数。M は桁数の合計 (精度) で、D は小数点以下の桁数 (スケール) です。小数点と、負の数に対する「-」の記号は M にはカウントされません。D が 0 のときは、小数点や小数部はありません。DECIMAL の最大桁数 (M) は 65 です。サポートされる小数部の最大桁数 (D) は 30 です。D が省略された場合のデフォルトは 0 です。M が省略された場合のデフォルトは 10 です。

  UNSIGNED が指定されている場合、負の値は許可されません。

DECIMAL カラムを使用したすべての基本的な計算 (+, -, *, /) は、65 桁の精度で行われます。

- DEC[(M[,D])] [UNSIGNED] [ZEROFILL], NUMERIC[(M[,D])] [UNSIGNED] [ZEROFILL], FIXED[(M[,D])] [UNSIGNED] [ZEROFILL]

  これらの型は DECIMAL のシノニムです。FIXED シノニムは、ほかのデータベースシステムとの互換性のために使用できます。

- FLOAT[(M,D)] [UNSIGNED] [ZEROFILL]

  小さい (単精度) 浮動小数点数。許可される値は、-3.402823466E+38 から -1.175494351E-38、0、および 1.175494351E-38 から 3.402823466E+38 です。これらは、IEEE スタンダードに基づいた理論的な限度です。使用しているハードウェアまたはオペレーティングシステムによっては、実際の範囲は少し小さくなる場合があります。

  M は桁数の合計で、D は小数点以下の桁数です。M と D を省略した場合、値はハードウェアで許可された限度まで格納されます。単精度小数点数はおおよそ小数第 7 位まで正確です。

  UNSIGNED が指定されている場合、負の値は許可されません。

  MySQL ではすべての計算が倍精度で行われているので、FLOAT を使用すると、予想外の問題が起きることがあります。セクションB.5.5.7「一致する行がない場合の問題の解決」を参照してください。

- DOUBLE[(M,D)] [UNSIGNED] [ZEROFILL]

  普通サイズ (倍精度) の浮動小数点数。許可されている値は、-1.7976931348623157E+308 から -2.2250738585072014E-308、0、および 2.2250738585072014E-308 から 1.7976931348623157E+308 です。これらは、IEEE スタンダードに基づいた理論的な限度です。使用しているハードウェアまたはオペレーティングシステムによっては、実際の範囲は少し小さくなる場合があります。

  M は桁数の合計で、D は小数点以下の桁数です。M と D を省略した場合、値はハードウェアで許可された限度まで格納されます。倍精度小数点数はおおよそ小数第 15 位まで正確です。

  UNSIGNED が指定されている場合、負の値は許可されません。

- DOUBLE PRECISION[(M,D)] [UNSIGNED] [ZEROFILL], REAL[(M,D)] [UNSIGNED] [ZEROFILL]

  これらの型は DOUBLE のシノニムです。例外: REAL_AS_FLOAT SQL モードが有効な場合は、DOUBLE ではなく REAL が FLOAT のシノニムになります。

- FLOAT(p) [UNSIGNED] [ZEROFILL]

  浮動小数点数です。p は精度をビットで表現しますが、MySQL は、結果として得られるデータ型に対して FLOAT または DOUBLE のどちらを使用するかを決めるためだけにこの値を使用します。p が 0 から 24 のとき、そのデータ型は M 値も D 値もない FLOAT になります。p が 25 から 53 のとき、そのデータ型は M 値も D 値もない DOUBLE になります。結果となるカラムの範囲は、このセクションで前述した単精度 FLOAT または倍精度 DOUBLE データ型の場合と同じです。

  FLOAT(p) 構文は ODBC との互換性を確保するために用意されています。

11.1.2 日付と時間型の概要

時間データ型のサマリーについて説明します。時間型のプロパティーおよびストレージ要件の追加情報については、セクション11.3「日付と時間型」およびセクション11.7「データ型のストレージ要件」を参照してください。時間値を演算する関数の説明については、セクション12.7「日付および時間関数」を参照してください。

DATE および DATETIME 範囲の説明では、「サポートされている」とは、以前の値は機能するが、保証はないことを意味します。

MySQL 5.6.4 以降では、マイクロ秒 (6 桁) までの精度を持つ TIME、DATETIME、および TIMESTAMP 値の小数秒に対応できるようになりました。小数秒部を含むカラムを定義するには、type_name(fsp) の構文を使用します。ここで、type_name は TIME、DATETIME、または TIMESTAMP であり、fsp は小数秒の精度です。例:

```
CREATE TABLE t1 (t TIME(3), dt DATETIME(6));
```

fsp 値を指定する場合、0 から 6 の範囲にする必要があります。0 の値は、小数部がないことを表します。省略した場合、デフォルトの精度は 0 です。(これは、以前の MySQL バージョンと互換性を保つため、標準 SQL のデフォルトである 6 とは異なっています。)

MySQL 5.6.5 には、時間型の拡張された自動初期化および更新機能が導入されました。テーブルごとに最大 1 つのカラムではなく、テーブル内のすべての TIMESTAMP カラムにこれらのプロパティーを割り当てられます。さらに、これらのプロパティーは、DATETIME カラムで使用できるようになりました。

YEAR(2) データ型には、使用する前に考慮する必要のある特定の問題があります。MySQL 5.6.6 以降、YEAR(2) は非推奨です。既存のテーブル内の YEAR(2) カラムは以前のとおりに扱われますが、新規または変更したテーブルでは YEAR(2) は YEAR(4) に変換されます。詳細は、セクション11.3.4「YEAR(2) の制限と YEAR(4) への移行」を参照してください。

- DATE

  日付です。サポートしている範囲は '1000-01-01' から '9999-12-31' です。MySQL は 'YYYY-MM-DD' の形式で DATE 値を表示しますが、文字列または数値のどちらかを使用した DATE カラムへの値の割り当てを許可しています。

- DATETIME[(fsp)]

  日付と時間の組み合わせです。サポートしている範囲は '1000-01-01 00:00:00.000000' から '9999-12-31 23:59:59.999999' です。MySQL は、'YYYY-MM-DD HH:MM:SS[.fraction]' の形式で DATETIME 値を表示しますが、文字列または数値のどちらかを使用した DATETIME カラムへの値の割り当てを許可しています。

  MySQL 5.6.4 以降では、小数秒の精度を指定するために 0 から 6 の範囲でオプションの fsp 値を指定できます。0 の値は、小数部がないことを表します。省略した場合、デフォルトの精度は 0 です。

  MySQL 5.6.5 以降、DATETIME カラムに対する自動初期化および現在の日時への自動更新は、セクション11.3.5「TIMESTAMP および DATETIME の自動初期化および更新機能」で説明しているように、DEFAULT および ON UPDATE カラム定義句を使用して指定できます。

- TIMESTAMP[(fsp)]

  タイムスタンプです。範囲は '1970-01-01 00:00:01.000000' UTC から '2038-01-19 03:14:07.999999' UTC です。TIMESTAMP 値は、エポック ('1970-01-01 00:00:00' UTC) からの秒数として格納されます。TIMESTAMP は、'1970-01-01 00:00:00' という値を表すことはできません。これは、エポックからの秒数が 0 であることと同等で、0 という値は '0000-00-00 00:00:00'、つまり「ゼロ」の TIMESTAMP 値を表すために予約されているからです。

  MySQL 5.6.4 以降では、小数秒の精度を指定するために 0 から 6 の範囲でオプションの fsp 値を指定できます。0 の値は、小数部がないことを表します。省略した場合、デフォルトの精度は 0 です。

  サーバーで TIMESTAMP 定義をどのように扱うかは、explicit_defaults_for_timestamp システム変数の値によって異なります (セクション5.1.4「サーバーシステム変数」を参照してください)。デフォルトでは、explicit_defaults_for_timestamp は無効であり、サーバーは次のように TIMESTAMP を扱います。

  特に指定されていないかぎり、テーブル内の最初の TIMESTAMP カラムは、明示的に値が割り当てられていなければもっとも新しい変更の日時に自動的に設定されるように定義されています。これにより、TIMESTAMP は、INSERT または UPDATE 操作のタイムスタンプの記録に役立ちます。NULL 値を許可するように NULL 属性で定義されていないかぎり、NULL 値を割り当てることによって、すべての TIMESTAMP カラムを現在の日付と時間に設定することもできます。

  自動初期化および現在の日付と時間への自動更新は、DEFAULT CURRENT_TIMESTAMP および ON UPDATE CURRENT_TIMESTAMP カラム定義句を使用して指定できます。デフォルトでは、前述のように最初の TIMESTAMP カラムにこれらのプロパティーが含まれます。MySQL 5.6.5 以降では、テーブル内のどの TIMESTAMP カラムでもこれらのプロパティーを割り当てるように定義できます。5.6.5 より前では、これらを割り当てられる TIMESTAMP カラムはテーブルごとに最大 1 つにかぎられますが、最初のカラムでは抑制し、代わりに別の TIMESTAMP カラムに割り当てることが可能です。セクション11.3.5「TIMESTAMP および DATETIME の自動初期化および更新機能」を参照してください。

  explicit_defaults_for_timestamp が有効な場合、すべての TIMESTAMP カラムへの DEFAULT CURRENT_TIMESTAMP または ON UPDATE CURRENT_TIMESTAMP 属性の自動的な割り当ては行われません。これらはカラム定義に明示的に含める必要があります。また、NOT NULL として明示的に宣言されていないすべての TIMESTAMP は、NULL 値を許可します。

  explicit_defaults_for_timestamp は、MySQL 5.6.6 以降で使用できます。5.6.6 より前では、サーバーは、explicit_defaults_for_timestamp が無効の場合について説明したように TIMESTAMP を扱います。これらの動作は、デフォルトのままになっていますが、標準外であり、5.6.6 以降では非推奨です。explicit_defaults_for_timestamp を有効化したインストールのアップグレードに関する説明については、セクション2.11.1.3「MySQL 5.5 から 5.6 へのアップグレード」を参照してください。

- TIME[(fsp)]

  時間です。範囲は、'-838:59:59.000000' から '838:59:59.000000' です。MySQL は、'HH:MM:SS[.fraction]' 形式で TIME 値を表示しますが、文字列または数値のどちらかを使用した TIME カラムの値への割り当てを許可します。

  MySQL 5.6.4 以降では、小数秒の精度を指定するために 0 から 6 の範囲でオプションの fsp 値を指定できます。0 の値は、小数部がないことを表します。省略した場合、デフォルトの精度は 0 です。

- YEAR[(2|4)]

  2 桁または 4 桁の形式の年です。デフォルトは 4 桁の形式です。YEAR(2) と YEAR(4) は表示形式が違いますが、値の範囲は同じです。4 桁の形式では、値は 1901 から 2155 と 0000 として表示されます。2 桁の形式では、値は 70 から 69 として表示され、1970 から 2069 の年を表します。MySQL では、YEAR 値は YYYY または YY の形式で表示されますが、文字列または数値を使用して YEAR カラムに値を割り当てられます。

  > 注記
  >
  > YEAR(2) データ型には、使用する前に考慮する必要のある特定の問題があります。MySQL 5.6.6 以降、YEAR(2) は非推奨です。既存のテーブル内の YEAR(2) カラムは以前のとおりに扱われますが、新規または変更したテーブルでは YEAR(2) は YEAR(4) に変換されます。詳細は、セクション11.3.4「YEAR(2) の制限と YEAR(4) への移行」を参照してください。

  入力値の YEAR の表示形式および解釈に関する追加情報については、セクション11.3.3「YEAR 型」を参照してください。

SUM() および AVG() 集計関数は時間値を扱いません。(これらは値を数字に変換するので、最初の数字以外の文字のあとのすべての情報が失われます。)この問題を回避するには、数値単位に変換し、集計操作を実行してから、時間値に戻します。例:

```
SELECT SEC_TO_TIME(SUM(TIME_TO_SEC(time_col))) FROM tbl_name;
SELECT FROM_DAYS(SUM(TO_DAYS(date_col))) FROM tbl_name;
```

> 注記
>
> MySQL Server は、MAXDB SQL モードを有効にして実行できます。この場合、TIMESTAMP は DATETIME と同じです。テーブルの作成時にこのモードが有効になっている場合、TIMESTAMP カラムは DATETIME カラムとして作成されます。その結果、このようなカラムでは DATETIME 表示形式が使用され、値の範囲は同じになり、自動初期化や現在の日付と時間への自動更新は行われなくなります。セクション5.1.7「サーバー SQL モード」を参照してください。

11.1.3 文字列型の概要

文字列データ型のサマリーについて説明します。文字列型のプロパティーおよびストレージ要件の追加情報については、セクション11.4「文字列型」およびセクション11.7「データ型のストレージ要件」を参照してください。

MySQL は、文字列カラムを CREATE TABLE または ALTER TABLE ステートメントで与えられている型とは異なる型に変更することがあります。セクション13.1.17.3「暗黙のカラム指定の変更」を参照してください。

MySQL は、文字カラム定義の長さ指定を文字単位で解釈します。これは、CHAR、VARCHAR、および TEXT 型に適応されます。

多くの文字列データ型のカラム定義には、カラムの文字セットまたは照合順序を指定する属性を含めることができます。これらの属性は CHAR、VARCHAR、TEXT 型、ENUM、および SET データ型に適用されます。

- CHARACTER SET 属性は文字セットを指定し、COLLATE 属性は文字セットの照合順序を指定します。例:

```
CREATE TABLE t
(
  c1 VARCHAR(20) CHARACTER SET utf8,
  c2 TEXT CHARACTER SET latin1 COLLATE latin1_general_cs
);
```

  このテーブル定義は、utf8 の文字セットとその文字セットのデフォルト照合順序を持つ c1 という名前のカラムと、latin1 の文字セットと大文字と小文字を区別する照合順序を持つ c2 という名前のカラムを作成します。

CHARACTER SET 属性または COLLATE 属性、あるいはその両方がない場合に、文字セットや照合順序を割り当てるためのルールは、セクション10.1.3.4「カラム文字セットおよび照合順序」で説明しています。

CHARSET は CHARACTER SET のシノニムです。

- 文字データ型に CHARACTER SET binary 属性を指定すると、カラムは対応するバイナリデータ型として作成されます。つまり、CHAR は BINARY になり、VARCHAR は VARBINARY になり、TEXT は BLOB になります。ENUM および SET データ型では、これは行われず、宣言されたとおりに作成されます。この定義を使用して、テーブルを指定したとします。

```
CREATE TABLE t
(
  c1 VARCHAR(10) CHARACTER SET binary,
  c2 TEXT CHARACTER SET binary,
  c3 ENUM('a','b','c') CHARACTER SET binary
);
```

結果のテーブルには、この定義が含まれています。

```
CREATE TABLE t
(
  c1 VARBINARY(10),
  c2 BLOB,
  c3 ENUM('a','b','c') CHARACTER SET binary
);
```

- ASCII 属性は CHARACTER SET latin1 の短縮形です。
- UNICODE 属性は CHARACTER SET ucs2 の短縮形です。
- BINARY 属性は、カラム文字セットのバイナリ照合順序を指定する短縮形です。この場合、ソートと比較は数字の値に基づきます。

文字カラムのソートと比較は、カラムに割り当てられた文字セットに基づきます。CHAR、VARCHAR、TEXT、ENUM、および SET データ型では、辞書順ではなくベースとなる文字コード値をソートおよび比較で使用できるように、バイナリ照合順序または BINARY 属性を持つカラムを宣言できます。

セクション10.1「文字セットのサポート」では、MySQL の文字セットの使用に関する追加情報が記述されています。

- [NATIONAL] CHAR[(M)] [CHARACTER SET charset_name] [COLLATE collation_name]

  格納時に必ず、指定された長さになるように右側がスペースで埋められる固定長文字列です。M はカラムの長さを文字数で表します。M の範囲は 0 から 255 です。M を省略すると、長さは 1 になります。

  > **注記**
  >
  > PAD_CHAR_TO_FULL_LENGTH SQL モードが有効になっていないかぎり、CHAR 値が取り出されるときに末尾のスペースは削除されます。

  CHAR は CHARACTER の短縮形です。NATIONAL CHAR (またはそれと同等の短縮形である NCHAR) は、CHAR カラムが事前に定義された文字セットを使用する必要があることを定義する標準 SQL の方法です。MySQL 4.1 以降では、この事前に定義された文字セットとして utf8 を使用します。セクション10.1.3.6「各国文字セット」を参照してください。

  CHAR BYTE データ型は BINARY データ型のエイリアスです。これは互換性機能です。

  MySQL では、CHAR(0) の型のカラムを作成できます。これは主に、カラムの存在に依存するが、実際にはその値を使用しない古いアプリケーションに準拠する必要があるときに役立ちます。CHAR(0) は、2 つの値しか取れないカラムが必要な場合にも非常に便利です。CHAR(0) NULL として定義されたカラムは 1 ビットだけを占め、NULL と '' (空の文字列) 値だけを取ることができます。

- [NATIONAL] VARCHAR(M) [CHARACTER SET charset_name] [COLLATE collation_name]

  可変長文字列です。M はカラムの最大長を文字数で表します。M の範囲は 0 から 65,535 です。VARCHAR の有効な最大長は、最大行サイズ (65,535 バイト、すべてのカラムで共有されます) と使用される文字セットによって決まります。たとえば、utf8 の文字は 1 文字につき最大 3 バイトを必要とする場合があるため、utf8 の文字セットを使用する VARCHAR カラムは、最大 21,844 文字になるように宣言できます。セクション D.10.4「テーブルカラム数と行サイズの制限」を参照してください。

MySQL は、VARCHAR 値を 1 バイトまたは 2 バイト長のプリフィクスが付いたデータとして格納します。長さプリフィクスは、値に含まれるバイト数を示します。VARCHAR カラムは、格納できる値が 255 バイト以下の場合は 1 バイト長のプリフィクスを使用し、255 バイトより大きい場合は 2 バイト長のプリフィクスを使用します。

> **注記**
>
> MySQL 5.6 は、標準 SQL 仕様に従い、VARCHAR 値から末尾のスペースを削除しません。

VARCHAR は CHARACTER VARYING の短縮形です。NATIONAL VARCHAR は、VARCHAR カラムが事前定義された文字セットを使用する必要があることを定義するための標準 SQL の方法です。MySQL 4.1 以降では、この事前に定義された文字セットとして utf8 を使用します。セクション10.1.3.6「各国文字セット」を参照してください。NVARCHAR は NATIONAL VARCHAR の短縮形です。

- BINARY(M)

  BINARY 型は CHAR 型と似ていますが、非バイナリ文字列ではなく、バイナリバイト文字列を格納します。M はカラムの長さをバイト数で表します。

- VARBINARY(M)

  VARBINARY 型は VARCHAR 型と似ていますが、非バイナリ文字列ではなく、バイナリバイト文字列を格納します。M はカラムの最大の長さをバイト数で表します。

- TINYBLOB

  最大長が 255 ($2^8 - 1$) バイトの BLOB カラム。各 TINYBLOB 値は、値のバイト数を示す 1 バイト長のプリフィクスを使用して格納されます。

- TINYTEXT [CHARACTER SET charset_name] [COLLATE collation_name]

  最大長が 255 ($2^8 - 1$) 文字の TEXT カラム。値にマルチバイト文字が含まれる場合、有効な最大長は少なくなります。各 TINYTEXT 値は、値のバイト数を示す 1 バイト長のプリフィクスを使用して格納されます。

- BLOB[(M)]

  最大長が 65,535 ($2^{16} - 1$) バイトの BLOB カラム。各 BLOB 値は、値のバイト数を示す 2 バイト長のプリフィクスを使用して格納されます。

  この型には、オプションの長さ M を指定できます。これが行われた場合、MySQL は M バイトの長さの値を保持するのに十分な最小の BLOB 型としてカラムを作成します。

- TEXT[(M)] [CHARACTER SET charset_name] [COLLATE collation_name]

  最大長が 65,535 ($2^{16} - 1$) 文字の TEXT カラム。値にマルチバイト文字が含まれる場合、有効な最大長は少なくなります。各 TEXT 値は、値のバイト数を示す 2 バイト長のプリフィクスを使用して格納されます。

  この型には、オプションの長さ M を指定できます。これが行われた場合、MySQL は M 文字の長さの値を保持するのに十分な最小 TEXT 型としてカラムを作成します。

- MEDIUMBLOB

  最大長が 16,777,215 ($2^{24} - 1$) バイトの BLOB カラム。各 MEDIUMBLOB 値は、値のバイト数を示す 3 バイト長のプリフィクスを使用して格納されます。

- MEDIUMTEXT [CHARACTER SET charset_name] [COLLATE collation_name]

  最大長が 16,777,215 ($2^{24} - 1$) 文字の TEXT カラム。値にマルチバイト文字が含まれる場合、有効な最大長は少なくなります。各 MEDIUMTEXT 値は、値のバイト数を示す 3 バイト長のプリフィクスを使用して格納されます。

- LONGBLOB

  最大長が 4,294,967,295 または 4G バイト ($2^{32} - 1$) バイトの BLOB カラム。LONGBLOB カラムの有効な最大長は、クライアント/サーバープロトコルと使用可能なメモリー内の構成済み最大パケットサイズにより決まります。各 LONGBLOB 値は、値のバイト数を示す 4 バイト長のプリフィクスを使用して格納されます。

- LONGTEXT [CHARACTER SET charset_name] [COLLATE collation_name]

最大長が 4,294,967,295 または 4G バイト (2^{32} − 1) 文字の TEXT カラム。値にマルチバイト文字が含まれる場合、有効な最大長は少なくなります。LONGTEXT カラムの有効な最大長もまた、クライアント/サーバープロトコルと使用可能メモリー内の構成済みの最大パケットサイズにより決まります。各 LONGTEXT 値は、値のバイト数を示す 4 バイト長のプリフィクスを使用して格納されます。

- ENUM('value1','value2',...) [CHARACTER SET charset_name] [COLLATE collation_name]

  列挙です。'value1'、'value2'、... の値、NULL、または特殊な '' エラー値のリストから選択された値を 1 つだけを持つことができる文字列オブジェクトです。ENUM 値は、内部では整数として表されます。

  ENUM カラムには、最大 65,535 個の個別の要素を含めることができます。(実用的な限度は 3000 個までです。)テーブルには、グループと見なされる ENUM および SET カラムの中の一意の要素リスト定義を、255 個以下を含めることができます。これらの制限の詳細は、セクションD.10.5「.frm ファイル構造により課せられる制限」を参照してください。

- SET('value1','value2',...) [CHARACTER SET charset_name] [COLLATE collation_name]

  セットです。ゼロ個以上の値を持つことができる文字列オブジェクトであり、そのそれぞれの値は、'value1'、'value2'、... 値のリストから選択する必要があります。SET 値は整数として内部で表されます。

  SET カラムには最大 64 個の個別のメンバーを含めることができます。テーブルには、グループと見なされる ENUM および SET カラムの中の一意の要素リスト定義を、255 個以下を含めることができます。この制限の詳細は、セクションD.10.5「.frm ファイル構造により課せられる制限」を参照してください。

11.2 数値型

MySQL はすべての標準 SQL 数値データ型をサポートします。これらの型は、概数値データ型 (FLOAT、REAL、DOUBLE PRECISION) だけでなく、真数値データ型 (INTEGER、SMALLINT、DECIMAL、NUMERIC) を含みます。キーワード INT は INTEGER のシノニムで、キーワード DEC および FIXED は DECIMAL のシノニムです。MySQL では、DOUBLE は DOUBLE PRECISION (非標準の拡張) のシノニムと見なされます。また、REAL_AS_FLOAT SQL モードが有効でないかぎり、REAL は DOUBLE PRECISION (非標準のバリエーション) のシノニムと見なされます。

BIT データ型は、ビットフィールド値を格納し、MyISAM、MEMORY、InnoDB、および NDB テーブルでサポートされています。

範囲外の値のカラムへの割り当てと、式の評価中のオーバーフローに対する MySQL での処理の詳細は、セクション11.2.6「範囲外およびオーバーフローの処理」を参照してください。

数値型のストレージの要件の詳細は、セクション11.7「データ型のストレージ要件」を参照してください。

数値オペランドで計算の結果に使用されるデータ型は、オペランドの型と実行される演算によって異なります。詳細は、セクション12.6.1「算術演算子」を参照してください。

11.2.1 整数型 (真数値) - INTEGER、INT、SMALLINT、TINYINT、MEDIUMINT、BIGINT

MySQL では、INTEGER (または INT) および SMALLINT の SQL 標準整数型をサポートします。標準に対する拡張として、MySQL では、TINYINT、MEDIUMINT、および BIGINT の整数型もサポートします。次の表に、整数型ごとの必要なストレージと範囲を示します。

型	ストレージ	最小値	最大値
	(バイト)	(符号付き/符号なし)	(符号付き/符号なし)
TINYINT	1	-128	127
		0	255
SMALLINT	2	-32768	32767
		0	65535
MEDIUMINT	3	-8388608	8388607
		0	16777215
INT	4	-2147483648	2147483647
		0	4294967295

型	ストレージ	最小値	最大値
	(バイト)	(符号付き/符号なし)	(符号付き/符号なし)
BIGINT	8	-9223372036854775808	9223372036854775807
		0	18446744073709551615

11.2.2 固定小数点型 (真数値) - DECIMAL、NUMERIC

DECIMAL および NUMERIC 型は真数値データ値を格納します。これらの型は、金銭データを扱う場合など、正確な精度を保持することが重要な場合に使用されます。MySQL では、NUMERIC は DECIMAL として実装されるので、DECIMAL に関する次の注意事項が NUMERIC にも同様に適用されます。

MySQL 5.6 は DECIMAL 値をバイナリ形式で格納します。セクション12.20「高精度計算」を参照してください。

DECIMAL カラム宣言で、精度およびスケールはたとえば次のように指定できます (通常は指定されています)。

```
salary DECIMAL(5,2)
```

この例では、5 が精度で、2 がスケールです。精度は、その値に格納された有効な桁数を表し、スケールは小数点以下に格納できる桁数を表しています。

標準 SQL では、DECIMAL(5,2) には小数部が 2 桁の合計 5 桁の値を格納できる必要があるので、salary カラムに格納できる値は、-999.99 から 999.99 の範囲になります。

標準 SQL では、構文 DECIMAL(M) は、DECIMAL(M,0) と同等です。同様に、構文 DECIMAL は DECIMAL(M,0) と同等です。M の値を決定するために、実装は許可されています。MySQL は、DECIMAL 構文のこれらのバリアント形式をどちらもサポートします。M のデフォルト値は 10 です。

スケールが 0 の場合、DECIMAL 値には小数点も小数部も含まれません。

DECIMAL の最大桁数は 65 ですが、指定した DECIMAL カラムの実際の範囲は、その指定したカラムの精度またはスケールによって制約される場合があります。指定のスケールで許可されている数より多くの桁が小数点以下にある値が、このようなカラムに割り当てられた場合、値はそのスケールに変換されます。(正確な動作はオペレーティングシステム固有ですが、一般的には効果は許可されている桁数に切り捨てられます。)

11.2.3 浮動小数点型 (概数値) - FLOAT、DOUBLE

FLOAT および DOUBLE 型は概数値データ値を表します。MySQL は、単精度値には 4 バイトを、倍精度値には 8 バイトを使用します。

FLOAT については、SQL 標準では、オプションで、キーワード FLOAT に続く括弧内のビットで (指数の範囲ではなく) 精度を指定できます。MySQL はまた、このオプションの精度指定もサポートしますが、その精度値はストレージサイズを決定するためだけに使用されます。0 から 23 の精度は、4 バイト単精度の FLOAT カラムになります。24 から 53 の精度は、8 バイト倍精度の DOUBLE カラムになります。

MySQL は、FLOAT(M,D) または REAL(M,D) または DOUBLE PRECISION(M,D) の非標準の構文を許可します。ここで、「(M、D)」は、値は合計で M 桁まで格納でき、そのうちの D 桁は小数点以下です。たとえば、FLOAT(7,4) として定義されたカラムは、表示されたときには -999.9999 のようになります。MySQL は、値を格納するときに丸めを行うので、FLOAT(7,4) カラムに 999.00009 を挿入すると、近似の結果は 999.0001 になります。

浮動小数点値は概数値であり、真数値としては格納されないので、比較で値を真数値として扱おうとすると、問題が発生することがあります。これらはまた、プラットフォームまたは実装の依存関係にも従います。詳細は、セクションB.5.5.8「浮動小数点値に関する問題」を参照してください。

移植性を最大にするために、概数値データ値のストレージを必要とするコードでは、精度または桁数が指定されていない FLOAT または DOUBLE PRECISION を使用する必要があります。

11.2.4 ビット値型 - BIT

BIT データ型は、ビットフィールド値を格納するのに使用されます。BIT(M) の型は、M ビット値のストレージを有効にします。M の範囲は 1 から 64 までが可能です。

ビット値を指定するには、b'value' 表記を使用できます。value は、0 と 1 で書かれたバイナリ値です。たとえば、b'111' と b'10000000' はそれぞれ 7 と 128 を表しています。セクション9.1.6「ビットフィールドリテラル」を参照してください。

M ビット長よりも短い BIT(M) カラムに値を割り当てた場合、その値の左側はゼロで埋められます。たとえば、b'101' という値を BIT(6) カラムに割り当てると、実際には b'000101' を割り当てた場合と同じことになります。

11.2.5 数値型の属性

MySQL では、整数データ型の基本キーワードに続く括弧内で、その型の表示幅をオプションで指定する拡張をサポートしています。たとえば、INT(4) は、4 桁の表示幅の INT を指定しています。このオプションの表示幅は、左側をスペースでパディングすることによって、カラムに対して指定された幅よりも狭く整数値を表示するために、アプリケーションで使用される場合があります。(つまり、この幅は結果セットで返されるメタデータの中にあります。これを使用するかどうかは、アプリケーションしだいです。)

表示幅は、カラムに格納できない値の範囲を制約しません。カラムの表示幅より広い値が正しく表示されなくなることもありません。たとえば、SMALLINT(3) として指定されたカラムには、-32768 から 32767 の通常の SMALLINT 範囲があり、3 桁が許可されたこの範囲外の値は、4 桁以上を使用してすべて表示されます。

オプション (非標準) の属性 ZEROFILL と一緒に使用すると、デフォルトのスペースのパディングはゼロに置き換えられます。たとえば、INT(4) ZEROFILL として宣言されたカラムの場合、5 の値は 0005 として取得されます。

> **注記**
>
> ZEROFILL 属性は、カラムが式や UNION クエリーに含まれているときは無視されます。
>
> ZEROFILL 属性を持つ整数カラムに表示幅より大きな値を格納した場合、MySQL が一部の複雑な結合に対して一時テーブルを生成するときに問題が発生することがあります。これらの場合、MySQL は、カラムの表示幅内でデータ値が適合すると想定します。

すべての整数型には、オプション (非標準) 属性 UNSIGNED を指定できます。符号なしの型は、カラムで負ではない数値しか許可しないとき、またはカラムの上限の数値範囲を大きくする必要があるときに使用できます。たとえば、INT カラムが UNSIGNED である場合、カラム範囲のサイズは同じですが、その終点は -2147483648 と 2147483647 から、0 と 4294967295 に変化します。

浮動小数点と固定小数点も UNSIGNED になり得ます。整数型と同じように、この属性は負の値がカラムに格納されるのを防ぎます。整数型とは異なり、カラム値の上限範囲は変わりません。

数値カラムに対して ZEROFILL を指定すると、MySQL は自動的にそのカラムに UNSIGNED 属性を追加します。

整数または浮動小数点のデータ型には、追加の属性 AUTO_INCREMENT を指定できます。インデックスが設定された AUTO_INCREMENT カラムに NULL (推奨) または 0 の値を挿入すると、カラムは次のシーケンス値に設定されます。通常、これは value+1 です。ここで value は現在テーブルにあるカラムの最大値です。AUTO_INCREMENT シーケンスは 1 で始まります。(AUTO_INCREMENT 値を生成するために NULL を挿入する場合、カラムを NOT NULL と宣言する必要があります。カラムを NULL と宣言した場合、NULL を挿入すると NULL が格納されます。)

MySQL 5.6.9 以降では、AUTO_INCREMENT カラムの負の値はサポートされません。

11.2.6 範囲外およびオーバーフローの処理

MySQL が、カラムデータ型の許可できる範囲外にある数値カラムに値を格納すると、結果は、その時点で有効な SQL モードによって異なります。

- 厳密な SQL モードが有効な場合、SQL 標準に従って、MySQL は範囲外の値を拒否してエラーを表示し、挿入は失敗します。
- 制限の強いモードが有効になっていない場合、MySQL は、範囲の適切な終点に値を切り落とし、その結果の値を代わりに格納します。

  範囲外の値が整数カラムに割り当てられると、MySQL は、カラムデータ型の範囲の対応する終点を表す値を格納します。TINYINT または TINYINT UNSIGNED カラムに 256 を格納すると、MySQL はそれぞれに 127 または 255 を格納します。

  浮動小数点または固定小数点カラムに、指定された (またはデフォルトの) 精度とスケールによって暗示された範囲を超えた値が割り当てられると、MySQL はその範囲の対応する終点を表す値を格納します。

MySQL が厳密モードで動作していないときの切り落としのために起きるカラム割り当て変換は、ALTER TABLE、LOAD DATA INFILE、UPDATE、および複数行の INSERT ステートメントに対する警告としてレポー

トされます。厳密モードでは、これらのステートメントは失敗し、テーブルがトランザクションテーブルかどうかやほかの要因に応じて、一部またはすべての値が挿入または変更されません。詳細は、セクション5.1.7「サーバー SQL モード」を参照してください。

MySQL 5.6 では、数値式評価中のオーバーフローはエラーになります。たとえば、符号付きの BIGINT の最大値は 9223372036854775807 なので、次の式ではエラーが発生します。

```
mysql> SELECT 9223372036854775807 + 1;
ERROR 1690 (22003): BIGINT value is out of range in '(9223372036854775807 + 1)'
```

この場合に演算を成功させるには、値を符号なしに変換します。

```
mysql> SELECT CAST(9223372036854775807 AS UNSIGNED) + 1;
+-----------------------------------------+
| CAST(9223372036854775807 AS UNSIGNED) + 1 |
+-----------------------------------------+
|                    9223372036854775808 |
+-----------------------------------------+
```

オーバーフローが起きるかどうかはオペランドの範囲に応じて異なります。したがって、前述の式を処理するもう 1 つの方法として、DECIMAL 値に整数より大きな範囲があるので正確な値の演算を使用します。

```
mysql> SELECT 9223372036854775807.0 + 1;
+--------------------------+
| 9223372036854775807.0 + 1 |
+--------------------------+
|    9223372036854775808.0 |
+--------------------------+
```

一方が UNSIGNED 型のときに 2 つの整数値の間で減算を行うと、デフォルトでは符号なしの結果が生成されます。MySQL 5.5.5 より前では、それ以外では結果が負になっていた場合、最大の整数値になります。

```
mysql> SET sql_mode = '';
mysql> SELECT CAST(0 AS UNSIGNED) - 1;
+------------------------+
| CAST(0 AS UNSIGNED) - 1 |
+------------------------+
|    18446744073709551615 |
+------------------------+
```

MySQL 5.5.5 以降では、それ以外では結果が負になっていた場合、エラーになります。

```
mysql> SET sql_mode = '';
Query OK, 0 rows affected (0.00 sec)

mysql> SELECT CAST(0 AS UNSIGNED) - 1;
ERROR 1690 (22003): BIGINT UNSIGNED value is out of range in '(cast(0 as unsigned) - 1)'
```

NO_UNSIGNED_SUBTRACTION SQL モードが有効な場合は、結果は負になります。

```
mysql> SET sql_mode = 'NO_UNSIGNED_SUBTRACTION';
mysql> SELECT CAST(0 AS UNSIGNED) - 1;
+------------------------+
| CAST(0 AS UNSIGNED) - 1 |
+------------------------+
|                     -1 |
+------------------------+
```

このような演算の結果を使用して UNSIGNED 整数カラムが更新されると、結果はそのカラム型の最大値に切り落とされます。NO_UNSIGNED_SUBTRACTION が有効になっている場合は、0 に切り落とされます。厳密な SQL モードが有効になっている場合は、エラーが発生し、カラムは変わりません。

11.3 日付と時間型

時間値を表すための日付と時間型は、DATE、TIME, DATETIME、TIMESTAMP、および YEAR です。それぞれの時間型には、一定範囲の有効な値のほかに、MySQL では表すことのできない無効な値の指定時に使用できる「ゼロ」値があります。TIMESTAMP 型には、後述するように、特殊な自動更新処理があります。時間型のストレージ要件については、セクション11.7「データ型のストレージ要件」を参照してください。

日付と時間型を処理するときに、次の考慮事項に留意してください。

- MySQL は、標準出力形式で所定の日付または時間型の値を取得しますが、(たとえば、日付または時間型に割り当てたり、比較したりする値を指定するときに) 入力した入力値に対してさまざまな形式を解釈しようとします。日付と時間型に許可されている形式の説明については、セクション9.1.3「日付リテラルと時間リテラ

ル」を参照してください。有効な値を入力する必要があります。ほかの形式で値を使用すると、予期しない結果が生じることがあります。

- MySQL は、複数の形式で値を解釈しようとしますが、日付の部分は、ほかでは一般的に使用される月-日-年や日-月-年の順 ('09-04-98' や '04-09-98' など) ではなく、年-月-日の順 ('98-09-04' など) で常に指定する必要があります。
- 2 桁の年を含む日付の値は、世紀が不明なためあいまいです。MySQL は次のルールを使用して 2 桁の年の値を解釈します。
  - 70-99 の範囲の値は 1970-1999 に変換されます。
  - 00-69 の範囲の値は 2000-2069 に変換されます。

  セクション11.3.8「日付での 2 桁の年」も参照してください。
- ある時間型から別の時間型への値の変換は、セクション11.3.7「日付と時間型間での変換」でのルールに従って行われます。
- MySQL は自動的に、値が数値のコンテキストで使用される場合には日付または時間値を数値に、またはその反対に変換します。
- デフォルトで MySQL は、日付または時間型の値で、範囲外であるか、それ以外で型にとって無効である値を見つけた場合、値をその型の「ゼロ」値に変換します。その例外では、範囲外の TIME 値は TIME 範囲の適切な終点に切り落とされます。
- SQL モードを適切な値に設定することで、MySQL がサポートする日付の種類をより正確に指定できます。(セクション5.1.7「サーバー SQL モード」を参照してください。)ALLOW_INVALID_DATES SQL モードを有効にすることによって、'2009-11-31' などの特定の日付を MySQL に受け入れさせることができます。これは、ユーザーが今後の処理のために、(たとえば Web フォームで) 指定した「間違っている可能性のある」値をデータベースに格納するときに役立ちます。このモードでは、MySQL は、月が 1 から 12 までの範囲にあることと、日付が 1 から 31 までの範囲にあることのみ検証します。
- MySQL では、DATE または DATETIME カラムに、日がゼロ、または月および日がゼロである日付の格納を許可しています。これは、正確な日付がわかっていない可能性のある生年月日を格納する必要があるアプリケーションで役立ちます。この場合は、単に日付を '2009-00-00' または '2009-01-00' として格納します。このような日付を格納する場合は、DATE_SUB() や DATE_ADD() などの完全な日付を必要とする関数で正しい結果が返されることは期待しないでください。日付でゼロの月または日の部分を無効にするには、NO_ZERO_IN_DATE SQL モードを有効にします。
- MySQL では、「ダミーの日付」として '0000-00-00' の「ゼロ」の値を格納できます。場合によっては、これは、NULL 値を使用するよりも便利であり、使用するデータおよびインデックススペースが少なくなります。'0000-00-00' を無効にするには、NO_ZERO_DATE SQL モードを有効にします。
- Connector/ODBC で使用される「ゼロ」の日付または時間の値は、ODBC がこのような値を処理できないため、NULL に自動的に変換されます。

次の表に、それぞれの型の「ゼロ」値の形式を示します。「ゼロ」値は特別ですが、表に示されている値を使用して、格納したり、明示的に参照したりできます。また、より簡単に記述できる '0' や 0 の値を使用してこれを行うこともできます。日付部分 (DATE、DATETIME、および TIMESTAMP) を含む時間型では、NO_ZERO_DATE SQL モードが有効な場合、これらの値を使用すると警告が発生します。

データ型	「ゼロ」値
DATE	'0000-00-00'
TIME	'00:00:00'
DATETIME	'0000-00-00 00:00:00'
TIMESTAMP	'0000-00-00 00:00:00'
YEAR	0000

11.3.1 DATE、DATETIME、および TIMESTAMP 型

DATE、DATETIME、および TIMESTAMP 型は関連しています。このセクションでは、これらの特徴、似ている点、および異なる点について説明します。MySQL は、セクション9.1.3「日付リテラルと時間リテラル」で説明している複数の形式で、DATE、DATETIME、および TIMESTAMP 値を認識します。DATE および DATETIME 範囲の説明では、「サポートされている」とは、以前の値は機能するが、保証はないということを意味します。

DATE 型は、日付部分を含むが時間部分は含まない値に使用されます。MySQL では、DATE 値の取得と表示は 'YYYY-MM-DD' 形式で行われます。サポートしている範囲は '1000-01-01' から '9999-12-31' です。

DATETIME 型は、日付と時間の両方の部分を含む値に使用されます。MySQL では、DATETIME 値の取得と表示は 'YYYY-MM-DD HH:MM:SS' 形式で行われます。サポートしている範囲は '1000-01-01 00:00:00' から '9999-12-31 23:59:59' です。

TIMESTAMP データ型は、日付と時間の両方の部分を含む値に使用されます。TIMESTAMP には、'1970-01-01 00:00:01' UTC から '2038-01-19 03:14:07' UTC の範囲があります。

DATETIME または TIMESTAMP 値には、マイクロ秒 (6 桁) までの精度で後続の小数秒部分を含めることができます。特に、MySQL 5.6.4 以降では、DATETIME または TIMESTAMP カラムに挿入された値の小数部はすべて破棄されずに格納されます。小数部が含まれる場合、これらの値の形式は 'YYYY-MM-DD HH:MM:SS[.fraction]' であり、DATETIME 値の範囲は '1000-01-01 00:00:00.000000' から '9999-12-31 23:59:59.999999' であり、TIMESTAMP 値の範囲は '1970-01-01 00:00:01.000000' から '2038-01-19 03:14:07.999999' です。小数部は、常に時間の残りの部分から小数点で区分する必要があります。これ以外の小数秒区切り文字は認識されません。MySQL の小数秒のサポートの詳細は、セクション11.3.6「時間値での小数秒」を参照してください。

TIMESTAMP および (MySQL 5.6.5 以降の) DATETIME データ型では、自動初期化と現在の日付および時間への自動更新機能が用意されています。詳細は、セクション11.3.5「TIMESTAMP および DATETIME の自動初期化および更新機能」を参照してください。

MySQL は、TIMESTAMP 値を、ストレージでは現在のタイムゾーンを UTC に変換し、取得では UTC から現在のタイムゾーンに戻します。(DATETIME などのほかの型ではこれは行われません。)デフォルトでは、接続ごとの現在のタイムゾーンはサーバーの時間です。タイムゾーンは接続ごとに設定できます。タイムゾーン設定が一定であるかぎり、格納した値と同じ値に戻すことができます。TIMESTAMP 値を格納したあとで、タイムゾーンを変更して値を取り出すと、取り出された値は格納した値とは異なります。これは、同じタイムゾーンが両方向への変換に使用されなかったために起こります。現在のタイムゾーンは、time_zone システム変数の値として使用できます。詳細は、セクション10.6「MySQL Server でのタイムゾーンのサポート」を参照してください。

無効な DATE、DATETIME、または TIMESTAMP 値は、適切な型の「ゼロ」値 ('0000-00-00' または '0000-00-00 00:00:00') に変換されます。

MySQL では日付値解釈の特定のプロパティーに注意してください。

- MySQL は、文字列として指定された値に、「緩やかな」形式を使用でき、この形式では、どの句読点文字でも日付部分と時間部分の区切り文字として使用できます。場合によっては、この構文は偽りになることがあります。たとえば、'10:11:12' などの値は、「:」区切り文字のために時間値のように見えることがありますが、日付のコンテキストで使用された場合は、'2010-11-12' の年と解釈されます。値 '10:45:15' は、'45' が有効な月ではないので、'0000-00-00' に変換されます。

  日付および時間の部分と小数秒部分との間の区切り文字として認識される唯一の文字が小数点です。

- サーバーは、月と日の値が、それぞれが 1 から 12 と 1 から 31 の範囲内にあるだけではなく、有効である必要があります。厳密モードが無効になっていると、'2004-04-31' のような無効な日付は '0000-00-00' に変換され、警告メッセージが表示されます。厳密モードが有効なときは、無効な日付によってエラーが発生します。このような日付を許可するには、ALLOW_INVALID_DATES を有効にします。詳細は、セクション5.1.7「サーバー SQL モード」を参照してください。

- MySQL は、日または月カラムにゼロを含んだ TIMESTAMP 値や、無効な日付の値を受け入れません。このルールに対する唯一の例外は、特殊な「ゼロ」値である '0000-00-00 00:00:00' です。

- MySQL 5.6.4 より前では、テーブルから選択しない場合、CAST() は TIMESTAMP 値を文字列として扱います。(これは、FROM DUAL を指定した場合にも当てはまります。)セクション12.10「キャスト関数と演算子」を参照してください。

- 2 桁の年を含む日付の値は、世紀が不明なためあいまいです。MySQL は次のルールを使用して 2 桁の年の値を解釈します。

  - 00-69 の範囲の値は 2000-2069 に変換されます。
  - 70-99 の範囲の値は 1970-1999 に変換されます。

  セクション11.3.8「日付での 2 桁の年」も参照してください。

> **注記**
>
> MySQL Server は、MAXDB SQL モードを有効にして実行できます。この場合、TIMESTAMP は DATETIME と同じです。テーブル作成時にこのモードが有効に

> なっている場合、TIMESTAMP カラムは DATETIME カラムとして作成されます。この結果、このようなカラムは DATETIME 表示形式を使用し、同じ範囲の値を持ち、自動初期化機能や、現在の日付と時間に自動的に更新する機能はありません。セクション 5.1.7「サーバー SQL モード」を参照してください。

11.3.2 TIME 型

MySQL では、TIME 値の取り出しと表示は 'HH:MM:SS' 形式 (時間の部分の値が大きい場合は 'HHH:MM:SS' 形式) で行われます。TIME 値の範囲は、'-838:59:59' から '838:59:59' です。TIME 型は、時間 (24 時間以下にする必要があります) を表すだけでなく、経過時間や、2 つのイベント間の時間 (24 時間よりも非常に長くなる場合も、負になる場合もあります) を表すこともできるので、時間の部分は非常に大きくなる可能性があります。

MySQL が TIME 値を認識する形式は複数あり、そのいくつかにはマイクロ秒 (6 秒) までの精度で後続の小数秒部分を含めることができます。セクション9.1.3「日付リテラルと時間リテラル」を参照してください。MySQL の小数秒のサポートの詳細は、セクション11.3.6「時間値での小数秒」を参照してください。特に、MySQL 5.6.4 以降では、TIME カラムに挿入された値の小数部はすべて破棄されずに格納されます。小数部が含まれている場合、TIME 値の範囲は '-838:59:59.000000' から '838:59:59.000000' です。

TIME カラムに省略された値を割り当てる場合は注意してください。MySQL は、コロン付きの省略された TIME 値を時間と解釈します。つまり、'11:12' は '00:11:12' ではなく '11:12:00' を意味します。MySQL は、右端の 2 桁が秒を表すという仮定を使用して (つまり、時間としてではなく経過時間として)、コロンのない省略された値を解釈します。たとえば、'1112' と 1112 は '11:12:00' (11 時 12 分) を表すように見えますが、MySQL では '00:11:12' (11 分 12 秒) と解釈されます。同様に、'12' や 12 は '00:00:12' と解釈されます。

時間部分と小数秒部分との間の区切り文字として認識される唯一の文字が小数点です。

デフォルトでは、TIME 範囲外にあるが、それ以外は有効な値は、範囲のもっとも近い終点に切り落とされます。たとえば、'-850:00:00' と '850:00:00' は、それぞれ '-838:59:59' と '838:59:59' に変換されます。無効な TIME 値は、'00:00:00' に変換されます。'00:00:00' はそれ自体が有効な TIME 値なので、元の値が '00:00:00' と指定されたのかどうか、無効であったかどうか、テーブルに格納された '00:00:00' の値から判断できません。

無効な TIME 値の制限を厳しくするには、エラーが発生するように厳密な SQL モードを有効にしてください。セクション5.1.7「サーバー SQL モード」を参照してください。

11.3.3 YEAR 型

YEAR 型は年の値を表すために使用される 1 バイトの型です。これは YEAR(4) または YEAR(2) と宣言して、4 文字または 2 文字の表示幅を指定できます。幅が指定されていない場合、デフォルトは 4 文字になります。

> 注記
>
> YEAR(2) データ型には、使用する前に考慮する必要のある特定の問題があります。また、MySQL 5.6.6 以降では、YEAR(2) は非推奨です。既存のテーブル内の YEAR(2) カラムは以前のとおりに扱われますが、新規または変更したテーブルでは YEAR(2) は YEAR(4) に変換されます。詳細は、セクション11.3.4「YEAR(2) の制限と YEAR(4) への移行」を参照してください。

YEAR(4) と YEAR(2) は表示形式が異なりますが、値の範囲は同じです。4 桁の形式の場合、MySQL は、YYYY の形式と 1901 から 2155、または 0000 の範囲で、YEAR 値を表示します。2 桁形式の場合、MySQL は、70 (1970 または 2070) や 69 (2069) など、最後 (最下位) の 2 桁だけを表示します。

YEAR の入力値は、次に示すさまざまな形式で指定できます。

- 1901 から 2155 の範囲の 4 桁の数値として。
- '1901' から '2155' の範囲の 4 桁の文字列として。
- 1 から 99 の範囲の 1 桁または 2 桁の数値として。MySQL は、1 から 69 と 70 から 99 の範囲の値を、2001 から 2069 と 1970 から 1999 の範囲の YEAR 値に変換します。
- '0' から '99' の範囲の 1 桁または 2 桁の文字列として。MySQL は、'0' から '69' と '70' から '99' の範囲の値を、2000 から 2069 と 1970 から 1999 の範囲の YEAR 値に変換します。
- 数値 0 を挿入した場合、その効果は YEAR(2) と YEAR(4) で異なります。YEAR(2) の場合、00 の表示値と 2000 の内部値の結果になります。YEAR(4) の場合、0000 の表示値と 0000 の内部値の結果になります。YEAR(4) にゼロを指定し、これを 2000 として解釈させるには、文字列 '0' または '00' としてこれを指定します。

- NOW() などの YEAR コンテキストで許容される値を返す関数の結果として。

MySQL は無効な YEAR 値を 0000 に変換します。

セクション11.3.8「日付での 2 桁の年」も参照してください。

11.3.4 YEAR(2) の制限と YEAR(4) への移行

このセクションでは、YEAR(2) の使用時に生じる可能性のある問題を取り上げ、既存の YEAR(2) カラムを YEAR(4) に変換するための情報について説明します。

YEAR(4) と YEAR(2) の値の内部範囲は同じですが (1901 から 2155、および 0000)、YEAR(2) の表示幅は、表示値が内部値の最後の 2 桁しか示さず、世紀を表す最初の 2 桁を省略するので、その型があいまいになります。特定の状況下では情報が失われる結果になることもあります。このため、YEAR データ型が必要なときは必ず、アプリケーション全体で YEAR(2) の使用を避け、YEAR(4) を使用するよう検討してください。MySQL 5.6.6 以降では、4 以外の表示値の YEAR データ型 (特に YEAR(2)) のサポートが縮小し、今後のリリースで完全に廃止される予定なので、ある時点で変換が必要になります。

YEAR(2) の制限

YEAR(2) データ型に関する問題には、表示値のあいまいさと、値のダンプおよびリロード時または文字列への変換時に情報損失の可能性があります。

- YEAR(2) の表示値はあいまいな場合があります。次の例で示すように、異なる内部値を持つ最大 3 つの YEAR(2) 値を同じ表示値にできます。

```
mysql> CREATE TABLE t (y2 YEAR(2), y4 YEAR(4));
Query OK, 0 rows affected (0.01 sec)

mysql> INSERT INTO t (y2) VALUES(1912),(2012),(2112);
Query OK, 3 rows affected (0.00 sec)
Records: 3  Duplicates: 0  Warnings: 0

mysql> UPDATE t SET y4 = y2;
Query OK, 3 rows affected (0.00 sec)
Rows matched: 3  Changed: 3  Warnings: 0

mysql> SELECT * FROM t;
+------+------+
| y2   | y4   |
+------+------+
|   12 | 1912 |
|   12 | 2012 |
|   12 | 2112 |
+------+------+
3 rows in set (0.00 sec)
```

- mysqldump を使用して、前述の項目に作成されたテーブルをダンプする場合、ダンプファイルは、同じ 2 桁の表現 (12) を使用してすべての y2 値を表します。ダンプファイルからテーブルをリロードした場合、すべての結果の行に内部値 2012 と表示値 12 を含まれるので、これらの違いが失われます。

- YEAR(2) または YEAR(4) データ値を文字列形式に変換した場合、YEAR 型の表示幅が使用されます。YEAR(2) と YEAR(4) の両方のカラムに 1970 の値が含まれるとします。それぞれのカラムを文字列に割り当てた結果、それぞれ '70' と '1970' の値になります。つまり、YEAR(2) から文字列への変換で情報の損失が起こります。

- 1970 から 2069 の範囲から外れた値は、CSV テーブルの YEAR(2) カラムに挿入されるときに、間違って格納されます。たとえば、2111 を挿入すると、表示値は 11 になりますが、内部値は 2011 になります。

これらの問題を避けるには、YEAR(2) ではなく YEAR(4) を使用してください。移行戦略に関した提案は、このセクションで後述します。

MySQL 5.6 の YEAR(2) サポートの縮小

MySQL 5.6.6 以降で、YEAR(2) のサポートは縮小されます。

- 新しいテーブルの YEAR(2) カラム定義は、YEAR(4) に (警告付きで) 変換されます。

```
mysql> CREATE TABLE t1 (y YEAR(2));
Query OK, 0 rows affected, 1 warning (0.03 sec)

mysql> SHOW WARNINGS\G
```

```
*************************** 1. row ***************************
  Level: Warning
   Code: 1818
Message: YEAR(2) column type is deprecated. Creating YEAR(4) column instead.
1 row in set (0.00 sec)

mysql> SHOW CREATE TABLE t1\G
*************************** 1. row ***************************
       Table: t1
Create Table: CREATE TABLE `t1` (
  `y` year(4) DEFAULT NULL
) ENGINE=InnoDB DEFAULT CHARSET=latin1
1 row in set (0.00 sec)
```

- 既存のテーブルの YEAR(2) は YEAR(2) として残され、古いバージョンの MySQL と同様にクエリーで処理されます。ただし、複数のプログラムまたはステートメントが、YEAR(2) を YEAR(4) に自動的に変換します。
  - テーブルの再構築を招く ALTER TABLE ステートメント。
  - REPAIR TABLE (YEAR(2) カラムがテーブルに含まれることが CHECK TABLE で検出された場合に、使用するように推奨されます)。
  - mysql_upgrade (REPAIR TABLE を使用します)。
  - mysqldump でのダンプおよびダンプファイルのリロード。上記の 3 つの項目が実行する変換とは異なり、ダンプとリロードは値を変更する可能性があります。

  MySQL アップグレードでは通常、最後の 2 つの項目のうち少なくとも 1 つを含みます。ただし、YEAR(2) については mysql_upgrade をお勧めします。前述のように値を変更する可能性があるので、mysqldump は使用しないでください。

YEAR(2) から YEAR(4) への移行

YEAR(2) カラムを YEAR(4) に変換するには、手動でアップグレードなしにいつでも行えます。または、YEAR(2) のサポートが縮小したバージョンの MySQL (MySQL 5.6.6 以降) にアップグレードしてから、MySQL で YEAR(2) カラムを自動的に変換できます。後者の場合は、データのダンプとリロードによるアップグレードは行わないでください。データ値を変更する可能性があります。さらに、レプリケーションを使用する場合は、注意の必要なアップグレードに関する考慮事項があります。

YEAR(2) カラムを YEAR(4) に手動で変換するには、ALTER TABLE を使用してください。テーブル t1 に次の定義があるとします。

```
CREATE TABLE t1 (ycol YEAR(2) NOT NULL DEFAULT '70');
```

次のように ALTER TABLE を使用してカラムを変更します。必ず、NOT NULL や DEFAULT などのすべてのカラム属性を含めてください。

```
ALTER TABLE t1 MODIFY ycol YEAR(4) NOT NULL DEFAULT '1970';
```

ALTER TABLE ステートメントは、YEAR(2) 値を変更しないでテーブルを変換します。サーバーがレプリケーションマスターである場合、ALTER TABLE ステートメントはスレーブに複製され、各対応するテーブルを変更します。

別の移行方法では、バイナリアップグレードを実行します。データのダンプおよびリロードを行わないで MySQL 5.6.6 以降をインストールします。続いて、mysql_upgrade を実行します。これは、REPAIR TABLE を使用して、データ値を変更しないで YEAR(2) カラムを YEAR(4) に変換します。サーバーがレプリケーションマスターである場合、--skip-write-binlog オプションを付けて mysql_upgrade を呼び出さないかぎり、REPAIR TABLE ステートメントはスレーブに複製され、各対応するテーブルを変更します。

レプリケーションサーバーへのアップグレードでは通常、新しいバージョンの MySQL へのスレーブのアップグレードと、マスターのアップグレードが行われます。たとえば、マスターとスレーブの両方で MySQL 5.5 が実行している場合、通常のアップグレードシーケンスでは、スレーブを 5.6 にアップグレードしてからマスターを 5.6 にアップグレードします。MySQL 5.6.6 以降での YEAR(2) の異なる扱いに関しては、このアップグレードシーケンスにより問題が生じます。スレーブがアップグレードされているが、マスターがまだアップグレードされていないとします。この場合、マスター上に YEAR(2) カラムを含んだテーブルを作成すると、スレーブ上では YEAR(4) カラムを含むテーブルが作成されます。この結果、ステートメントベースのレプリケーションを使用した場合、これらの操作はマスターとスレーブで異なる結果になります。

- 数値 0 の挿入。結果の値は、マスター上では 2000 の内部値になりますが、スレーブ上では 0000 の内部値になります。

- 文字列への YEAR(2) の変換。この操作は、マスター上では YEAR(2) の表示値を使用しますが、スレーブ上では YEAR(4) の表示値を使用します。

このような問題を避けるには、次の戦略のいずれかを使用します。

- ステートメントベースのレプリケーションの代わりに行ベースのレプリケーションを使用します。
- アップグレード前に、マスター上のすべての YEAR(2) カラムを YEAR(4) に変更します。(前述のように ALTER TABLE を使用します。)続いて、YEAR(2) から YEAR(4) の移行でマスターとスレーブ間で違いが生じることなく、通常どおりに (最初にスレーブ、次にマスター) アップグレードできます)。

mysqldump でデータをダンプし、アップグレードしたあとにダンプファイルをリロードするという移行方法は使用しないでください。前述のように、これは YEAR(2) 値を変更する可能性があります。

YEAR(2) から YEAR(4) への移行では、次のような条件下で動作が変更された可能性がないかどうかアプリケーションコードの検査も行う必要があります。

- YEAR カラムを選択すれば正確に 2 桁が生成されると予想しているコード。
- 0 を YEAR(2) または YEAR(4) に挿入すると、それぞれ 2000 または 0000 の内部値になるという、数値 0 の挿入に対する別々の処理に対応していないコード。

11.3.5 TIMESTAMP および DATETIME の自動初期化および更新機能

MySQL 5.6.5 以降では、TIMESTAMP および DATETIME カラムを自動的に初期化でき、現在の日付および時間 (つまり、現在のタイムスタンプ) に自動的に更新できます。5.6.5 より前では、これは TIMESTAMP にしか当てはまらず、テーブルあたり最大で 1 つの TIMESTAMP カラムにしか当てはまりません。次の注意事項では、最初に、MySQL 5.6.5 以降での自動初期化および更新機能について、次に 5.6.5 より前のバージョンでの相違点について説明します。

テーブル内のあらゆる TIMESTAMP または DATETIME カラムに対して、デフォルト値または自動更新値、あるいはその両方として、現在のタイムスタンプを割り当てることができます。

- 自動初期化されたカラムは、カラムに値を指定しない挿入行に対して現在のタイムスタンプに設定されます。
- 自動更新されたカラムは、行内のほかのカラムの値がその現在の値から変更されると、現在のタイムスタンプに自動的に更新されます。自動更新されたカラムは、ほかのすべてのカラムがその現在の値に設定されていれば、変更されないまま保持されます。ほかのカラムが変更したときに、自動更新したカラムが更新しないようにするには、明示的にこれを現在の値に設定します。ほかのカラムが変更しない場合でも、自動更新カラムを更新するには、明示的にこれを必要な値に設定します (たとえば CURRENT_TIMESTAMP に設定します)。

さらに、NULL 属性を使用して NULL 値を許可するように定義されていないかぎり、NULL 値を割り当てることによって、すべての TIMESTAMP カラムを初期化したり、現在の日付と時間に更新したりできます。

自動プロパティーを指定するには、カラム定義で DEFAULT CURRENT_TIMESTAMP および ON UPDATE CURRENT_TIMESTAMP 句を使用します。句の順序は関係ありません。両方がカラム定義にある場合、どちらも最初に実行できます。CURRENT_TIMESTAMP のシノニムのいずれも、CURRENT_TIMESTAMP と同じ意味があります。これらは、CURRENT_TIMESTAMP()、NOW()、LOCALTIME、LOCALTIME()、LOCALTIMESTAMP、および LOCALTIMESTAMP() です。

DEFAULT CURRENT_TIMESTAMP および ON UPDATE CURRENT_TIMESTAMP の使用は、TIMESTAMP および DATETIME に固有です。DEFAULT 句も、DEFAULT 0 や DEFAULT '2000-01-01 00:00:00' などの一定 (非自動) のデフォルト値を指定するために使用できます。

> 注記
>
> DEFAULT 0 を使用した次の例は、NO_ZERO_DATE SQL モードが有効な場合には機能しません。このモードでは「ゼロ」の日付値 (たとえば 0 '0000-00-00 00:00:00' として指定) が拒否されるからです。TRADITIONAL SQL モードに NO_ZERO_DATE が含まれています。

TIMESTAMP または DATETIME カラム定義では、現在のタイムスタンプをデフォルト値と自動更新値の両方に対して指定することも、どちらか一方について指定することも、両方について指定しないこともできます。異なるカラムは、自動プロパティーの別々の組み合わせを持つことができます。次のルールは可能性のある場合について記述しています。

- DEFAULT CURRENT_TIMESTAMP と ON UPDATE CURRENT_TIMESTAMP の両方を使用した場合、カラムは、デフォルト値が現在のタイムスタンプになり、現在のタイムスタンプに自動的に更新されます。

```
CREATE TABLE t1 (
  ts TIMESTAMP DEFAULT CURRENT_TIMESTAMP ON UPDATE CURRENT_TIMESTAMP,
  dt DATETIME DEFAULT CURRENT_TIMESTAMP ON UPDATE CURRENT_TIMESTAMP
);
```

- DEFAULT 句を使用するが ON UPDATE CURRENT_TIMESTAMP 句を使用しない場合、カラムには所定のデフォルト値が設定され、現在のタイムスタンプに自動的に更新されません。

  デフォルトは、DEFAULT 句で CURRENT_TIMESTAMP を指定するか定数値を指定するかに応じて異なります。CURRENT_TIMESTAMP を使用した場合、デフォルトは現在のタイムスタンプになります。

  ```
  CREATE TABLE t1 (
    ts TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
    dt DATETIME DEFAULT CURRENT_TIMESTAMP
  );
  ```

  定数を使用した場合、デフォルトは所定の値になります。この場合、カラムには自動的なプロパティーはありません。

  ```
  CREATE TABLE t1 (
    ts TIMESTAMP DEFAULT 0,
    dt DATETIME DEFAULT 0
  );
  ```

- ON UPDATE CURRENT_TIMESTAMP 句と定数の DEFAULT 句を使用した場合、カラムは、現在のタイムスタンプに自動的に更新され、所定の定数のデフォルト値があります。

  ```
  CREATE TABLE t1 (
    ts TIMESTAMP DEFAULT 0 ON UPDATE CURRENT_TIMESTAMP,
    dt DATETIME DEFAULT 0 ON UPDATE CURRENT_TIMESTAMP
  );
  ```

- ON UPDATE CURRENT_TIMESTAMP 句を使用するが DEFAULT 句を使用しない場合、カラムは、自動的に現在のタイムスタンプに更新され、そのデフォルト値に現在のタイムスタンプは使用されません。

  この場合のデフォルトは型により異なります。TIMESTAMP は、NULL 属性を使用して定義されていないかぎり (この場合はデフォルトは NULL です)、デフォルトは 0 です。

  ```
  CREATE TABLE t1 (
    ts1 TIMESTAMP ON UPDATE CURRENT_TIMESTAMP,     -- default 0
    ts2 TIMESTAMP NULL ON UPDATE CURRENT_TIMESTAMP -- default NULL
  );
  ```

  DATETIME は、NOT NULL 属性で定義されていないかぎり (この場合、デフォルトは 0 です)、デフォルトは NULL です。

  ```
  CREATE TABLE t1 (
    dt1 DATETIME ON UPDATE CURRENT_TIMESTAMP,        -- default NULL
    dt2 DATETIME NOT NULL ON UPDATE CURRENT_TIMESTAMP -- default 0
  );
  ```

TIMESTAMP および DATETIME カラムには、明示的に指定されないかぎり自動プロパティーはありません。ただし、DEFAULT CURRENT_TIMESTAMP と ON UPDATE CURRENT_TIMESTAMP がどちらも明示的に指定されていない場合は、デフォルトで最初の TIMESTAMP カラムに両方とも存在します。最初の TIMESTAMP カラムについて自動プロパティーを抑制するには、次のいずれかの戦略を使用します。

- explicit_defaults_for_timestamp システム変数を有効にします。この変数が有効な場合、自動初期化および更新機能を指定する DEFAULT CURRENT_TIMESTAMP および ON UPDATE CURRENT_TIMESTAMP 句は使用可能ですが、カラム定義に明示的に含まれていないかぎり、どの TIMESTAMP カラムにも割り当てられません。
- または、explicit_defaults_for_timestamp が無効な場合 (デフォルト)、次のどちらかを行います。
  - 定数のデフォルト値を指定する DEFAULT 句を含むカラムを定義します。
  - NULL 属性を指定します。またこれにより、カラムで NULL 値が許可されます。つまり、カラムを NULL に設定することによって現在のタイムスタンプを割り当てることができなくなります。NULL を割り当てると、カラムは NULL に設定されます。

次のテーブル定義を考慮してください。

```
CREATE TABLE t1 (
  ts1 TIMESTAMP DEFAULT 0,
```

```
  ts2 TIMESTAMP DEFAULT CURRENT_TIMESTAMP
            ON UPDATE CURRENT_TIMESTAMP);
CREATE TABLE t2 (
  ts1 TIMESTAMP NULL,
  ts2 TIMESTAMP DEFAULT CURRENT_TIMESTAMP
            ON UPDATE CURRENT_TIMESTAMP);
CREATE TABLE t3 (
  ts1 TIMESTAMP NULL DEFAULT 0,
  ts2 TIMESTAMP DEFAULT CURRENT_TIMESTAMP
            ON UPDATE CURRENT_TIMESTAMP);
```

テーブルには次のプロパティーがあります。

- 各テーブル定義において、最初の TIMESTAMP カラムには、自動初期化または更新機能はありません。
- 各テーブルでは、ts1 カラムで NULL 値を処理する方法が異なります。t1 の場合、ts1 は NOT NULL であり、これに NULL の値を割り当てると、現在のタイムスタンプに設定されます。t2 と t3 の場合、ts1 では NULL を使用でき、これに NULL の値を割り当てると、NULL に設定されます。
- t2 と t3 では、ts1 のデフォルト値が異なります。t2 の場合、ts1 は、NULL を許可するように定義されているので、明示的な DEFAULT 句がない場合はデフォルトも NULL です。t3 の場合、ts1 は NULL を使用できますが、明示的なデフォルトは 0 です。

TIMESTAMP または DATETIME カラム定義のいずれかの場所に明示的な小数秒精度値が含まれる場合、カラム定義全体で同じ値を使用する必要があります。次の場合は許可されます。

```
CREATE TABLE t1 (
  ts TIMESTAMP(6) DEFAULT CURRENT_TIMESTAMP(6) ON UPDATE CURRENT_TIMESTAMP(6)
);
```

次の場合は許可されません。

```
CREATE TABLE t1 (
  ts TIMESTAMP(6) DEFAULT CURRENT_TIMESTAMP ON UPDATE CURRENT_TIMESTAMP(3)
);
```

MySQL 5.6.5 より前の自動タイムスタンププロパティー

MySQL 5.6.5 より前では、自動初期化および更新機能のサポートは非常に限定的です。

- DEFAULT CURRENT_TIMESTAMP と ON UPDATE CURRENT_TIMESTAMP は、DATETIME カラムで使用できません。
- DEFAULT CURRENT_TIMESTAMP と ON UPDATE CURRENT_TIMESTAMP は、テーブルあたり最大で 1 つの TIMESTAMP カラムでしか使用できません。現在のタイムスタンプを、あるカラムのデフォルト値にして、別のカラムの自動更新値にはできません。

これらのプロパティーを使用するかどうか、どの TIMESTAMP カラムで必要になるかを選択できます。これは、自動的に初期化される、または現在のタイムスタンプに自動的に更新されるテーブル内の最初のカラムにする必要はありません。別の TIMESTAMP カラムに対して自動初期化または更新を指定するには、前述のように、最初のカラムに対する自動プロパティーを制約する必要があります。この場合、ほかの TIMESTAMP カラムでは、DEFAULT および ON UPDATE 句のルールは、最初の TIMESTAMP カラムの場合と同じですが、両方の句を省略した場合、自動初期化も更新も行われません。

TIMESTAMP の初期化と NULL 属性

デフォルトでは、TIMESTAMP カラムは NOT NULL であり、NULL 値を含めることはできず、NULL を割り当てると現在のタイムスタンプが割り当てられます。NULL を含めるように TIMESTAMP カラムを許可するには、NULL 属性で明示的に宣言します。この場合、別のデフォルト値を指定する DEFAULT 句でオーバーライドされないかぎり、デフォルト値も NULL になります。DEFAULT NULL を使用すると、デフォルト値として NULL を明示的に指定できます。(NULL 属性が宣言されていない TIMESTAMP カラムの場合、DEFAULT NULL は無効です。)TIMESTAMP カラムで NULL 値を許可する場合、NULL を割り当てると、このカラムは現在のタイムスタンプではなく NULL に設定されます。

次のテーブルには、NULL 値を許可している複数の TIMESTAMP カラムが含まれています。

```
CREATE TABLE t
(
  ts1 TIMESTAMP NULL DEFAULT NULL,
  ts2 TIMESTAMP NULL DEFAULT 0,
  ts3 TIMESTAMP NULL DEFAULT CURRENT_TIMESTAMP
);
```

NULL 値を許可する TIMESTAMP カラムは、次のいずれかの状況に当てはまる場合を除き、挿入時に現在のタイムスタンプを取りません。

- デフォルト値が CURRENT_TIMESTAMP と定義され、カラムに対して値が指定されていない
- CURRENT_TIMESTAMP、またはNOW() などのそのいずれかのシノニムが明示的にカラムに挿入されている

つまり、NULL 値を許可するように定義されている TIMESTAMP カラムは、その定義に DEFAULT CURRENT_TIMESTAMP が含まれている場合にのみ自動初期化します。

```
CREATE TABLE t (ts TIMESTAMP NULL DEFAULT CURRENT_TIMESTAMP);
```

TIMESTAMP カラムで NULL 値を許可しているが、定義に DEFAULT CURRENT_TIMESTAMP が含まれていない場合、現在の日付と時間に対応する値を明示的に挿入する必要があります。t1 および t2 テーブルに次の定義があるとします。

```
CREATE TABLE t1 (ts TIMESTAMP NULL DEFAULT '0000-00-00 00:00:00');
CREATE TABLE t2 (ts TIMESTAMP NULL DEFAULT NULL);
```

挿入時にどちらかのテーブルの TIMESTAMP カラムを現在のタイムスタンプに設定するには、明示的にそのカラムにこの値を割り当てます。例:

```
INSERT INTO t1 VALUES (NOW());
INSERT INTO t2 VALUES (CURRENT_TIMESTAMP);
```

11.3.6 時間値での小数秒

MySQL 5.6.4 より前では、時間値で小数秒部分が許可されているインスタンスは制限されています。後続の小数部は、リテラル値などのコンテキストで許可され、一部の時間関数への引数またはそこからの戻り値で許可されています。例:

```
mysql> SELECT MICROSECOND('2010-12-10 14:12:09.019473');
+-----------------------------------------+
| MICROSECOND('2010-12-10 14:12:09.019473') |
+-----------------------------------------+
|                                   19473 |
+-----------------------------------------+
```

ただし、MySQL は時間データ型のカラムに値を格納するときに、小数部を破棄し、それを格納しません。

MySQL 5.6.4 以降では、マイクロ秒 (6 桁) までの精度を持つ TIME、DATETIME、および TIMESTAMP 値に対して小数秒のサポートを拡張しています。

- 小数秒部を含むカラムを定義するには、type_name(fsp) の構文を使用します。ここで、type_name は TIME、DATETIME、または TIMESTAMP であり、fsp は小数秒の精度です。例:

  ```
  CREATE TABLE t1 (t TIME(3), dt DATETIME(6));
  ```

  fsp 値を指定する場合、0 から 6 の範囲にする必要があります。0 の値は、小数部がないことを表します。省略した場合、デフォルトの精度は 0 です。(これは、以前の MySQL バージョンと互換性を保つため、標準 SQL のデフォルトである 6 とは異なっています。)

- 小数秒部分を持つ TIME、DATE、または TIMESTAMP 値を同じ型のカラムに挿入するが、小数部の桁数が少ない場合、次の例に示すように丸めが行われます。

  ```
  mysql> CREATE TABLE fractest( c1 TIME(2), c2 DATETIME(2), c3 TIMESTAMP(2) );
  Query OK, 0 rows affected (0.33 sec)

  mysql> INSERT INTO fractest VALUES
      > ('17:51:04.777', '2014-09-08 17:51:04.777', '2014-09-08 17:51:04.777');
  Query OK, 1 row affected (0.03 sec)

  mysql> SELECT * FROM fractest;
  +-------------+------------------------+------------------------+
  | c1          | c2                     | c3                     |
  +-------------+------------------------+------------------------+
  | 17:51:04.78 | 2014-09-08 17:51:04.78 | 2014-09-08 17:51:04.78 |
  +-------------+------------------------+------------------------+
  1 row in set (0.00 sec)
  ```

  このような丸め行われたときに、警告やエラーは表示されません。この動作は、SQL 標準に従い、サーバーの sql_mode 設定の影響は受けません。

- 時間引数を取る関数は、小数秒を含む値を受け入れます。時間関数からの戻り値には、必要に応じて小数秒が含まれます。たとえば、引数を付けない NOW() は、小数部のない現在の日付と時間を返しますが、0 から 6 のオプション引数を取って、その桁数の小数秒部が戻り値に含まれていることを指定します。

- 時間リテラルの構文は、DATE 'str'、TIME 'str'、および TIMESTAMP 'str' の時間値と ODBC 構文同等の値を生み出します。指定されている場合、結果の値には後続の小数秒部分が含まれます。以前は、時間型キーワードは無視され、これらの構造は文字列値を生成していました。標準 SQL と ODBC の日付および時間リテラルを参照してください。

場合によっては、以前に受け入れられていた構文が別の結果を生成することがあります。次の項目は、問題を回避するために既存のコードのどの箇所を変更する必要があるかを示します。

- 式の中には、以前の結果とは異なる結果を生成するものがあります。例: timestamp システム変数は、整数ではなくマイクロ秒小数部を含む値を返します。現在の時間を含む結果を返す関数 (CURTIME()、SYSDATE()、UTC_TIMESTAMP() など) は、fsp 値として引数を解釈し、戻り値にはその桁の小数秒部分が含まれます。以前には、これらの関数は引数を許可していましたが、無視していました。

- TIME 値は、現在の日付に時間を追加することにより DATETIME に変換されます。(これは、時間値が '00:00:00' から '23:59:59' の範囲から外れている場合に、結果の日付部分が現在の日付と異なることを意味します。)以前には、TIME 値の DATETIME への変換は信頼性がありませんでした。セクション11.3.7「日付と時間型間での変換」を参照してください。

- TIMESTAMP(N) は古い MySQL バージョンで許可されていましたが、N は小数秒精度ではなく表示幅でした。この動作のサポートは MySQL 5.5.3 で廃止されたので、適度に最新の状態に維持されているアプリケーションであれば、この問題の影響を受けません。それ以外の場合では、コードを書き換える必要があります。

11.3.7 日付と時間型間での変換

ある程度まで、ある時間型から別の時間型に値を変換できます。ただし、値の変更や情報の損失が生じることがあります。どの場合でも、時間型間の変換は、変換される型で有効な値の範囲に依存します。たとえば、DATE、DATETIME、および TIMESTAMP 値はすべて、同じセットの形式を使用して指定できますが、すべての型で値の範囲が同じであるわけではありません。TIMESTAMP 値は、1970 UTC より古い値にしたり、'2038-01-19 03:14:07' UTC より新しい値にしたりできません。つまり、'1968-01-01' などの日付は、DATE または DATETIME 値としては有効ですが、TIMESTAMP 値としては有効ではなく、0 に変換されます。

DATE 値の変換:

- DATE 値には時間情報が含まれないので、DATETIME または TIMESTAMP 値に変換すると、'00:00:00' の時間部分が追加されます。

- TIME 値への変換は有用ではありません。結果は '00:00:00' になります。

DATETIME および TIMESTAMP 値の変換:

- DATE 型には時間情報が含まれないので、DATE 値に変換すると時間部分が破棄されます。

- TIME 型には日付情報が含まれないので、TIME 値に変換すると日付部分が破棄されます。

TIME 値のほかの時間型への変換はバージョンによって異なります。

- MySQL 5.6.4 以降では、CURRENT_DATE() の値が日付部分に使用されます。TIME は (時間ではなく) 経過時間として解釈され、日付に追加されます。これは、時間値が '00:00:00' から '23:59:59' の範囲から外れている場合に、結果の日付部分が現在の日付と異なることを意味します。

  現在の日付が '2012-01-01' であるとします。'12:00:00'、'24:00:00'、'-12:00:00' の TIME 値は、DATETIME または TIMESTAMP 値に変換されると、それぞれ '2012-01-01 12:00:00'、'2012-01-02 00:00:00'、'2011-12-31 12:00:00' になります。

  TIME から DATE への変換も同様ですが、結果から時間部分が破棄され、それぞれ '2012-01-01'、'2012-01-02'、'2011-12-31' になります。

- 5.6.4 より前の MySQL では、時間の文字列を日付または日付時間として解析することによって、時間値を日付または日付時間値に変換します。これが役立つ可能性はありません。たとえば、'23:12:31' は、日付として解釈されると '2023-12-31' になります。日付として有効でない時間は '0000-00-00' または NULL になります。

明示的な変換を使用して暗黙的な変換をオーバーライドできます。たとえば、DATE および DATETIME 値の比較で、DATE 値は、'00:00:00' の時間部分を追加することにより、強制的に DATETIME 型に変更されます。代わりに DATETIME 値の時間部分を無視して比較を実行するには、次の方法で CAST() 関数を使用します。

```
date_col = CAST(datetime_col AS DATE)
```

TIME および DATETIME 値の数値形式への (+0 の追加などによる) 変換は、次のように行われます。

- MySQL 5.6.4 以降では、TIME(N) または DATETIME(N) は、N が 0 (または省略) の場合は整数に、N が正の数の場合は N の 10 進数を含む DECIMAL 値に変換されます。

```
mysql> SELECT CURTIME(), CURTIME()+0, CURTIME(3)+0;
+-----------+-------------+--------------+
| CURTIME() | CURTIME()+0 | CURTIME(3)+0 |
+-----------+-------------+--------------+
| 09:28:00  |       92800 |    92800.887 |
+-----------+-------------+--------------+
mysql> SELECT NOW(), NOW()+0, NOW(3)+0;
+---------------------+----------------+--------------------+
| NOW()               | NOW()+0        | NOW(3)+0           |
+---------------------+----------------+--------------------+
| 2012-08-15 09:28:00 | 20120815092800 | 20120815092800.889 |
+---------------------+----------------+--------------------+
```

- MySQL 5.6.4 より前では、変換の結果は、マイクロ秒部分が .000000 である倍精度値になります。

```
mysql> SELECT CURTIME(), CURTIME()+0;
+-----------+-------------+
| CURTIME() | CURTIME()+0 |
+-----------+-------------+
| 09:28:00  | 92800.000000 |
+-----------+-------------+
mysql> SELECT NOW(), NOW()+0;
+---------------------+----------------------+
| NOW()               | NOW()+0              |
+---------------------+----------------------+
| 2012-08-15 09:28:00 | 20120815092800.000000 |
+---------------------+----------------------+
```

11.3.8 日付での 2 桁の年

2 桁の年を含む日付の値は、世紀が不明なためあいまいです。MySQL では、年は内部的に 4 桁で格納されるため、そのような値は 4 桁の形式に変換する必要があります。

DATETIME、DATE、および TIMESTAMP 型では、MySQL は、次のルールを使用して、あいまいな年の値で指定された日付を変換します。

- 00-69 の範囲の値は 2000-2069 に変換されます。
- 70-99 の範囲の値は 1970-1999 に変換されます。

YEAR ではルールは同じですが、YEAR(4) に挿入された数値 00 は 2000 ではなく 0000 になります。YEAR(4) にゼロを指定し、これを 2000 として解釈させるには、文字列 '0' または '00' としてこれを指定します。

これらのルールは、データ値が何を表すかを妥当に推測する単なる経験則であることを覚えておいてください。MySQL で使用されるルールで必要な値が生成されない場合、4 桁の年を含む明確な入力値を指定する必要があります。

ORDER BY は、2 桁の年を持つ YEAR 値を正しくソートします。

MIN() や MAX() などの一部の関数は、YEAR を数値に変換します。つまり、2 桁の年の値は、これらの関数では正しく機能しません。この場合の解決策としては、YEAR を 4 桁の年の形式に変換します。

11.4 文字列型

文字列型には、CHAR、VARCHAR、BINARY、VARBINARY、BLOB、TEXT、ENUM、および SET があります。このセクションでは、これらの型の機能と、クエリーでの使用方法について説明します。文字列型のストレージ要件については、セクション11.7「データ型のストレージ要件」を参照してください。

11.4.1 CHAR および VARCHAR 型

CHAR 型と VARCHAR 型は似ていますが、格納および取得方法が異なります。また、最大長と、末尾のスペースが保持されるかどうかという点でも異なります。

CHAR 型と VARCHAR 型には、格納する最大文字数を表す長さが宣言されています。たとえば、CHAR(30) には最大 30 文字を格納できます。

CHAR カラムの長さは、テーブルを作成したときに宣言した長さに修正されます。この長さには、0 から 255 までの任意の値を指定できます。CHAR 値は格納されると、指定された長さになるように右側がスペースで埋められます。PAD_CHAR_TO_FULL_LENGTH SQL モードが有効になっていないかぎり、CHAR 値が取り出されるときに、末尾のスペースが削除されます。

VARCHAR カラム内の値は可変長の文字列です。長さは 0 から 65,535 までの値で指定できます。VARCHAR カラムの有効な最大長は、最大行サイズ (65,535 バイト、すべてのカラムで共有される) と使用される文字セットによって決まります。セクションD.10.4「テーブルカラム数と行サイズの制限」を参照してください。

CHAR とは対照的に、VARCHAR 値は、1 バイトまたは 2 バイト長のプリフィクスの付いたデータとして格納されます。長さプリフィクスは、値に含まれるバイト数を示します。255 バイト以下の値を格納するカラムでは 1 バイト長のプリフィクスを使用し、255 バイトよりも大きい値を格納するカラムでは 2 バイト長のプリフィクスを使用します。

厳密な SQL モードが有効でない場合に、CHAR または VARCHAR カラムにその最大長を超える値を割り当てると、その値はカラムの最大長に合わせて切り捨てられ、警告メッセージが表示されます。スペース以外の文字の切り捨てに関しては、厳密な SQL モードを使用することで、警告ではなくエラーを発生させて、その値の挿入を抑制できます。セクション5.1.7「サーバー SQL モード」を参照してください。

VARCHAR カラムの場合、使用している SQL モードに関係なく、カラム長を超える末尾のスペースは挿入前に切り捨てられ、警告メッセージが表示されます。CHAR カラムの場合、SQL モードに関係なく、超過した末尾のスペースは通知なしに挿入される値から切り捨てられます。

VARCHAR 値は格納されるときに埋められません。標準 SQL に従い、値を格納し取り出すときに末尾のスペースは保持されます。

次の表は、CHAR(4) カラムと VARCHAR(4) カラムにさまざまな文字列値を格納した結果を表示して、CHAR と VARCHAR の違いを示しています (カラムには latin1 などのシングルバイト文字セットを使用するものとします)。

値	CHAR(4)	必要なストレージ	VARCHAR(4)	必要なストレージ
''	' '	4 バイト	''	1 バイト
'ab'	'ab '	4 バイト	'ab'	3 バイト
'abcd'	'abcd'	4 バイト	'abcd'	5 バイト
'abcdefgh'	'abcd'	4 バイト	'abcd'	5 バイト

テーブルの最終行に格納済みとして示されている値は、厳密モードを使用していないときにだけ当てはまります。MySQL が厳密モードで実行されている場合、カラム長を超える値は格納されず、エラーが発生します。

所定の値が CHAR(4) および VARCHAR(4) カラムに格納されると、取り出しのときに末尾のスペースが CHAR カラムから削除されるので、カラムから取り出された値は必ずしも同じではありません。次の例はこの違いを示しています。

```
mysql> CREATE TABLE vc (v VARCHAR(4), c CHAR(4));
Query OK, 0 rows affected (0.01 sec)

mysql> INSERT INTO vc VALUES ('ab  ', 'ab  ');
Query OK, 1 row affected (0.00 sec)

mysql> SELECT CONCAT('(', v, ')'), CONCAT('(', c, ')') FROM vc;
+---------------------+---------------------+
| CONCAT('(', v, ')') | CONCAT('(', c, ')') |
+---------------------+---------------------+
| (ab  )              | (ab)                |
+---------------------+---------------------+
1 row in set (0.06 sec)
```

CHAR カラムと VARCHAR カラムの値は、そのカラムに割り当てられた文字セットの照合順序に従ってソートおよび比較されます。

MySQL のすべての照合順序は、PADSPACE 型のものです。これは、MySQL 内のすべての CHAR、VARCHAR、および TEXT 値が、末尾のスペースに関係なく比較されることを意味します。このコンテキストでの「比較」には、末尾のスペースが意味を持つ LIKE パターンマッチング演算子は含まれません。例:

```
mysql> CREATE TABLE names (myname CHAR(10));
Query OK, 0 rows affected (0.03 sec)

mysql> INSERT INTO names VALUES ('Monty');
Query OK, 1 row affected (0.00 sec)
```

```
mysql> SELECT myname = 'Monty', myname = 'Monty  ' FROM names;
+-----------------+-------------------+
| myname = 'Monty' | myname = 'Monty  ' |
+-----------------+-------------------+
|               1 |                 1 |
+-----------------+-------------------+
1 row in set (0.00 sec)

mysql> SELECT myname LIKE 'Monty', myname LIKE 'Monty  ' FROM names;
+--------------------+----------------------+
| myname LIKE 'Monty' | myname LIKE 'Monty  ' |
+--------------------+----------------------+
|                  1 |                    0 |
+--------------------+----------------------+
1 row in set (0.00 sec)
```

これは MySQL のすべてのバージョンに当てはまり、サーバーの SQL モードの影響は受けません。

> **注記**
>
> MySQL の文字セットおよび照合順序の詳細は、セクション10.1「文字セットのサポート」を参照してください。ストレージ要件の追加情報については、セクション11.7「データ型のストレージ要件」を参照してください。

末尾の埋め込み文字が取り除かれたり、比較で無視されたりする場合では、一意の値を必要とするインデックスがカラムに含まれていれば、末尾の埋め込み文字の個数だけが異なるカラム値への挿入は、重複キーエラーになります。たとえば、テーブルに 'a' が含まれている場合、'a ' を格納しようとすると、重複キーエラーが発生します。

11.4.2 BINARY および VARBINARY 型

BINARY および VARBINARY 型は、CHAR および VARCHAR 型に似ていますが、非バイナリ文字列ではなく、バイナリ文字列を格納します。つまり、それらには文字の文字列ではなく、バイトの文字列が含まれています。これは、それらに文字セットがなく、ソートおよび比較は値の中のバイトの数値に基づいていることを意味します。

BINARY および VARBINARY で許可される最大長は、CHAR および VARCHAR の場合と同じですが、BINARY および VARBINARY の長さが文字数ではなくバイト数で表される点が異なります。

BINARY および VARBINARY データ型は CHAR BINARY および VARCHAR BINARY データ型とは異なります。後者の型は、BINARY 属性によってカラムがバイナリ文字列カラムとして扱われることはありません。その代わり、これによってカラムの文字セットのバイナリ照合順序が使用され、カラム自体にはバイナリバイト文字列ではなく非バイナリ文字列が格納されます。たとえば、CHAR(5) BINARY は、デフォルト文字セットが latin1 とすれば、CHAR(5) CHARACTER SET latin1 COLLATE latin1_bin として扱われます。これは、文字セットや照合順序を持たない 5 バイトのバイナリ文字列を格納する BINARY(5) とは異なります。非バイナリ文字列のバイナリ照合順序とバイナリ文字列の違いについては、セクション10.1.7.6「_bin および binary 照合順序」を参照してください。

厳密な SQL モードが有効でない場合に、BINARY または VARBINARY カラムにその最大長を超える値を割り当てると、その値はカラムの最大長に合わせて切り捨てられ、警告メッセージが表示されます。値を切り捨てる場合、厳密な SQL モードを使用することで、警告ではなくエラーを発生させて、その値の挿入を抑制できます。セクション5.1.7「サーバー SQL モード」を参照してください。

BINARY 値は格納されると、特定の長さまで右側がパッド値で埋められます。パッド値は 0x00 (ゼロバイト) です。値は挿入時には右側が 0x00 で埋められ、選択時に後続のバイトは削除されません。すべてのバイトは、ORDER BY および DISTINCT 操作を含め比較で意味があります。0x00 バイトとスペースは比較では異なり、0x00 < スペースです。

例: BINARY(3) カラムの場合、'a ' は挿入時に 'a \0' になります。'a\0' は挿入時に 'a\0\0' になります。選択時、挿入された両方の値は変更されません。

VARBINARY では、挿入時にパディングされることも、選択時にバイトが削除されることもありません。すべてのバイトは、ORDER BY および DISTINCT 操作を含め比較で意味があります。0x00 バイトとスペースは比較では異なり、0x00 < スペースです。

後続のパッドバイトが取り除かれたり、比較で無視されたりする場合では、一意の値を必要とするインデックスがカラムに含まれていれば、後続のパッドバイトの個数だけが異なるカラム値への挿入は、重複キーエラーになります。たとえば、テーブルに 'a' が含まれている場合、'a\0' を格納しようとすると、重複キーエラーが発生します。

バイナリデータの格納に BINARY データ型を使用する予定であり、取り出した値を格納した値とまったく同じにする必要がある場合は、先行のパディングと削除文字を考慮する必要があります。次の例は、BINARY 値の 0x00 パディングによって、カラム値の比較がどのような影響を受けるかについて示しています。

```
mysql> CREATE TABLE t (c BINARY(3));
Query OK, 0 rows affected (0.01 sec)

mysql> INSERT INTO t SET c = 'a';
Query OK, 1 row affected (0.01 sec)

mysql> SELECT HEX(c), c = 'a', c = 'a\0\0' from t;
+--------+---------+-------------+
| HEX(c) | c = 'a' | c = 'a\0\0' |
+--------+---------+-------------+
| 610000 |       0 |           1 |
+--------+---------+-------------+
1 row in set (0.09 sec)
```

取り出される値を、パディングなしのストレージに指定した値と同じにする必要がある場合は、代わりに VARBINARY か、いずれかの BLOB データ型を使用することをお勧めします。

11.4.3 BLOB 型と TEXT 型

BLOB はさまざまな容量のデータを保持できる大きなバイナリオブジェクトです。BLOB 型は、TINYBLOB、BLOB、MEDIUMBLOB、および LONGBLOB の 4 つがあります。これらの違いは、保持できる値の最大長だけです。TEXT 型は、TINYTEXT、TEXT、MEDIUMTEXT、および LONGTEXT の 4 つがあります。これらは 4 つの BLOB 型に対応し、最大長とストレージ要件は同じです。セクション11.7「データ型のストレージ要件」を参照してください。

BLOB 値はバイナリ文字列 (バイトの文字列) として扱われます。これらには文字セットがなく、ソートおよび比較はカラム値内のバイトの数値に基づきます。TEXT 値は非バイナリ文字列 (文字の文字列) として扱われます。これらには文字セットがあり、値は文字セットの照合順序に基づいてソートおよび比較されます。

厳密な SQL モードが有効でない場合に、BLOB または TEXT カラムにその最大長を超える値を割り当てると、その値はカラムの最大長に合わせて切り捨てられ、警告メッセージが表示されます。スペース以外の文字の切り捨てに関しては、厳密な SQL モードを使用することで、警告ではなくエラーを発生させて、その値の挿入を抑制できます。セクション5.1.7「サーバー SQL モード」を参照してください。

TEXT カラムに挿入される値から、超過した末尾のスペースを切り捨てると、SQL モードには関係なく、常に警告が生成されます。

TEXT および BLOB カラムでは、挿入時にパディングは行われず、選択時にバイトは削除されません。

TEXT カラムにインデックスが設定されている場合、インデックスエントリの比較では末尾がスペースで埋められます。これは、インデックスに一意の値が必要な場合、末尾のスペースの個数だけが異なる値に対して重複キーエラーが発生するということを意味します。たとえば、テーブルに 'a' が含まれている場合、'a ' を格納しようとすると、重複キーエラーが発生します。これは BLOB カラムには当てはまりません。

ほとんどの点で、BLOB カラムを、任意の長さに設定できる VARBINARY カラムと見なすことができます。同様に、TEXT カラムを VARCHAR カラムと見なすことができます。BLOB と TEXT は、次の点で VARBINARY と VARCHAR とは異なっています。

- BLOB と TEXT カラムのインデックスには、インデックスプリフィクス長を指定する必要があります。CHAR と VARCHAR では、プリフィクス長はオプションです。セクション8.3.4「カラムインデックス」を参照してください。
- BLOB および TEXT カラムに DEFAULT 値を含めることはできません。

BINARY 属性を TEXT データ型と一緒に使用した場合、カラム文字セットのバイナリ照合順序がそのカラムに割り当てられます。

LONG と LONG VARCHAR は MEDIUMTEXT データ型にマップします。これは互換性機能です。

MySQL Connector/ODBC は BLOB 値を LONGVARBINARY として、TEXT 値を LONGVARCHAR として定義します。

BLOB 値と TEXT 値は非常に長くなる可能性があるので、使用するときに次の制約が生じることがあります。

- ソート時には、カラムの max_sort_length バイトだけが使用されます。max_sort_length のデフォルト値は 1024 です。サーバーの起動時または実行時に、max_sort_length の値を増やすことによって、ソートまたはグ

ループ化に影響するバイトを増やすことができます。すべてのクライアントで max_sort_length セッション変数の値を変更できます。

```
mysql> SET max_sort_length = 2000;
mysql> SELECT id, comment FROM t
    -> ORDER BY comment;
```

- 一時テーブルを使用して処理されるクエリーの結果に BLOB カラムまたは TEXT カラムのインスタンスがあると、MEMORY ストレージエンジンがこれらのデータ型をサポートしていないので、サーバーはメモリー内ではなくディスク上でテーブルを使用します (セクション8.4.4「MySQL が内部一時テーブルを使用する仕組み」を参照してください)。ディスクの使用はパフォーマンスの低下を伴うので、クエリーの結果に BLOB カラムまたは TEXT カラムを含めるのは必要な場合に限定してください。たとえば、SELECT * はすべてのカラムを選択するので使用しないでください。

- BLOB または TEXT オブジェクトの最大サイズはその型で決まりますが、クライアントとサーバー間で実際に送信できる最大値は、使用可能なメモリーの容量と通信バッファーのサイズで決まります。max_allowed_packet 変数の値を変更することでメッセージバッファーサイズを変更できますが、サーバーとクライアントプログラムの両方で変更する必要があります。たとえば、mysql と mysqldump のどちらを使用しても、クライアント側の max_allowed_packet 値を変更できます。セクション8.11.2「サーバーパラメータのチューニング」、セクション4.5.1「mysql — MySQL コマンド行ツール」、セクション4.5.4「mysqldump — データベースバックアッププログラム」を参照してください。パケットサイズおよびソートしているデータオブジェクトのサイズを、ストレージ要件と比較することもできます。セクション11.7「データ型のストレージ要件」を参照してください。

BLOB 値または TEXT 値はそれぞれ、別々に割り当てられたオブジェクトによって内部的に表現されます。これは、テーブルが開かれるときにカラムごとに一度ストレージが割り当てられる、ほかのすべてのデータ型と対照的です。

メディアファイルなどのバイナリデータを BLOB または TEXT カラムに格納するほうがよい場合もあります。このようなデータの処理には、MySQL の文字列操作関数が役立つことがあります。セクション12.5「文字列関数」を参照してください。セキュリティーなどの理由のために、通常は、アプリケーションユーザーに FILE 権限を与えるのではなく、アプリケーションコードを使用して実行することをお勧めします。MySQL フォーラム (http://forums.mysql.com/) では、さまざまな言語やプラットフォームの詳細について話し合うことができます。

11.4.4 ENUM 型

ENUM は、テーブル作成時にカラム仕様に明示的に列挙された、許可されている値のリストから選択された値を持つ文字列オブジェクトです。これには次の利点があります。

- 指定可能な値のセットがカラムで制限されている状況でのコンパクトなデータストレージ。入力値として指定した文字列は自動的に数値としてエンコードされます。ENUM 型のストレージ要件については、セクション11.7「データ型のストレージ要件」を参照してください。

- 読みやすいクエリーと出力。数値は、クエリー結果で対応する文字列に戻されます。

また、次のような考慮が必要な問題が生じる可能性があります。

- Enumeration Limitationsで説明しているように、数値のように見える列挙値を作成した場合、リテラル値とその内部インデックス番号を混同しやすくなります。

- Enumeration Sortingで説明しているように、ORDER BY 句で ENUM カラムを使用するには特に注意が必要です。

ENUM カラムの作成と使用

列挙値は引用符で囲んだ文字列リテラルにする必要があります。たとえば、次のように ENUM カラムを持つテーブルを作成できます。

```
CREATE TABLE shirts (
    name VARCHAR(40),
    size ENUM('x-small', 'small', 'medium', 'large', 'x-large')
);
INSERT INTO shirts (name, size) VALUES ('dress shirt','large'), ('t-shirt','medium'),
  ('polo shirt','small');
SELECT name, size FROM shirts WHERE size = 'medium';
+---------+--------+
| name    | size   |
+---------+--------+
| t-shirt | medium |
+---------+--------+
```

```
UPDATE shirts SET size = 'small' WHERE size = 'large';
COMMIT;
```

'medium' の値を持つ 100 万個の行をこのテーブルに挿入するには、100 万バイトのストレージが必要ですが、実際の文字列 'medium' を VARCHAR カラムに格納した場合は、600 万バイト必要になります。

列挙リテラルのインデックス値

それぞれの列挙値にはインデックスが設定されています。

- カラム仕様にリストされている要素には、1 から始まるインデックス番号が割り当てられています。
- 空の文字列エラー値のインデックス値は 0 です。つまり、次の SELECT ステートメントを使用して、無効な ENUM 値が割り当てられた行を検索できます。

```
mysql> SELECT * FROM tbl_name WHERE enum_col=0;
```

- NULL 値のインデックスは NULL です。
- ここでの「インデックス」という語は、列挙値のリスト内での位置を示します。これは、テーブルインデックスとはまったく関係ありません。

たとえば、ENUM('Mercury', 'Venus', 'Earth') と指定されたカラムには、次に示すどの値でも含めることができます。それぞれの値のインデックスも示しています。

値	インデックス
NULL	NULL
''	0
'Mercury'	1
'Venus'	2
'Earth'	3

ENUM カラムには、最大 65,535 個の個別の要素を含めることができます。(実用的な限度は 3000 個までです。) テーブルには、グループと見なされる ENUM および SET カラムの中の一意の要素リスト定義を、255 個以下を含めることができます。これらの制限の詳細は、セクションD.10.5「.frm ファイル構造により課せられる制限」を参照してください。

ENUM 値を数値コンテキストで取得した場合、カラム値のインデックスが返されます。たとえば、次のように ENUM カラムから数値を取得できます。

```
mysql> SELECT enum_col+0 FROM tbl_name;
```

数値引数を取る SUM() や AVG() などの関数は、必要に応じて引数を数値にキャストします。ENUM 値の計算にはインデックス番号が使用されます。

列挙リテラルの処理

テーブルが作成されるときに、テーブル定義内の ENUM メンバー値から末尾のスペースが自動的に削除されます。

ENUM カラムに格納された値は、取得されたときに、カラム定義で使用された大文字/小文字で表示されます。ENUM カラムには文字セットと照合順序を割り当てられています。バイナリ照合順序、または大文字と小文字を区別する照合順序の場合、カラムに値を割り当てるときに、大文字/小文字が考慮されます。

ENUM カラムに数字を格納すると、その数字は指定可能な値のインデックスとして扱われ、格納された値がそのインデックスを持つ列挙メンバーとなります。(ただし、これはすべての入力を文字列として扱う LOAD DATA では機能 しません。)数値が引用符で囲まれている場合、列挙値のリストに一致する文字列がなければ、そのままインデックスとして解釈されます。これらの理由により、ENUM カラムを数字のように見える列挙値で定義することは、混乱を招きやすくなるのでお勧めできません。たとえば、次のカラムには '0'、'1'、および '2' の文字列値を持つ列挙メンバーが指定されていますが、数値インデックス値は 1、2、および 3 です。

```
numbers ENUM('0','1','2')
```

2 を格納すると、それはインデックス値と解釈され、'1' (インデックス 2 の値) になります。'2' を格納すると、それは列挙値と一致するので、'2' として格納されます。'3' を格納すると、どの列挙値とも一致しないのでインデックスとして扱われ、'2' (インデックス 3 の値) になります。

```
mysql> INSERT INTO t (numbers) VALUES(2),('2'),('3');
mysql> SELECT * FROM t;
+---------+
| numbers |
+---------+
| 1       |
| 2       |
| 2       |
+---------+
```

ENUM カラムのすべての指定可能な値を特定するには、SHOW COLUMNS FROM tbl_name LIKE 'enum_col' を使用して、出力の Type カラム内の ENUM 定義を構文解析します。

C API では、ENUM 値は文字列として返されます。結果セットのメタデータを使用してこれらをほかの文字列から区別する方法については、セクション23.7.5「C API データ構造」を参照してください。

空または NULL の列挙値

以下のような特定の状況下では、列挙値は、空の文字列 ('') や NULL になることもあります。

- ENUM に無効な値 (つまり、許可された値のリストに存在しない文字列) を挿入すると、特殊なエラー値として空の文字列が代わりに挿入されます。この文字列は、この文字列に 0 の数値が含まれていることで、「通常」の空の文字列と区別できます。列挙値の数値インデックスの詳細は、Index Values for Enumeration Literalsを参照してください。

  厳密な SQL モードが有効な場合は、無効な ENUM 値を挿入しようとするとエラーが発生します。

- ENUM カラムが NULL を許可するように宣言されている場合、NULL 値は、そのカラムに対して有効な値であり、デフォルト値は NULL になります。ENUM カラムが NOT NULL として宣言されている場合、デフォルト値は許可されている値のリストの最初の要素になります。

列挙のソート

ENUM 値は、インデックス番号に基づいてソートされますが、この数値は、カラム仕様で列挙メンバーがリストされていた順序に従います。たとえば、ENUM('b', 'a') の場合、'b' は 'a' の前にソートされます。空の文字列は空ではない文字列の前にソートされ、NULL 値はその他のすべての列挙値の前にソートされます。

ENUM カラムで ORDER BY 句の使用時に予想外の結果になることを回避するには、次のいずれかの手法を使用します。

- アルファベット順で ENUM リストを指定します。
- ORDER BY CAST(col AS CHAR) または ORDER BY CONCAT(col) をコード化することにより、カラムがインデックス番号ではなく、辞書順でソートされることを確認します。

列挙の制限

列挙値は、文字列値に評価されるものであっても、式にはできません。

たとえば、次の CREATE TABLE ステートメントは、CONCAT 関数を列挙値の構築に使用できないので、機能しません。

```
CREATE TABLE sizes (
   size ENUM('small', CONCAT('med','ium'), 'large')
);
```

ユーザー変数を列挙値として使用することもできません。次のステートメントのペアは機能しません。

```
SET @mysize = 'medium';

CREATE TABLE sizes (
   size ENUM('small', @mysize, 'large')
);
```

数字を列挙値として使用しないことを強くお勧めします。これは、適切な TINYINT または SMALLINT 型よりもストレージを節約するわけでもなく、ENUM 値を間違って引用符で囲んだ場合には、文字列とベースになる数値とを混同しやすくなる (同じでない場合もあります) からです。数字を列挙値として使用する場合は、必ず引用符で囲んでください。引用符を省略した場合は、その数字はインデックスと見なされます。Handling of Enumeration Literalsを参照して、引用符で囲まれた数字でも間違って数字のインデックス値として使用されるか場合について確認してください。

定義の中に重複した値が含まれていると、警告 (厳密な SQL モードが有効になっている場合はエラー) が発生します。

11.4.5 SET 型

SET は、ゼロ以上の値を取ることができる文字列オブジェクトであり、それぞれの値は、テーブルの作成時に指定された許可される値のリストから選択する必要があります。SET カラム値が複数のセットメンバーで構成される場合は、各メンバーはカンマ (「,」) で区切って指定されます。このため、SET メンバーの値自体にはカンマを含めないでください。

たとえば、SET('one', 'two') NOT NULL として指定したカラムは、次に示す値のいずれかを取ります。

```
''
'one'
'two'
'one,two'
```

SET カラムには最大 64 個の個別のメンバーを含めることができます。テーブルには、グループと見なされる ENUM および SET カラムの中の一意の要素リスト定義を、255 個以下を含めることができます。この制限の詳細は、セクションD.10.5「.frm ファイル構造により課せられる制限」を参照してください。

定義の中に重複した値が含まれていると、警告 (厳密な SQL モードが有効になっている場合はエラー) が発生します。

テーブルが作成されるときに、テーブル定義内の SET メンバー値から末尾のスペースが自動的に削除されます。

SET カラムに格納された値は、取得されるときに、カラム定義で使用されていた大文字/小文字で表示されます。SET カラムには、文字セットと照合順序を割り当てることができます。バイナリ照合順序、または大文字と小文字を区別する照合順序の場合、カラムに値を割り当てるときに、大文字/小文字が考慮されます。

MySQL は、最初のセットメンバーに対応する格納値の低位ビットを使用して SET 値を数値で格納します。SET 値を数値コンテキストで取得した場合、その取得された値には、カラム値を構成するセットメンバーに対応するビットセットが含まれます。たとえば、次のように SET カラムから数値を取得できます。

```
mysql> SELECT set_col+0 FROM tbl_name;
```

メンバーが SET カラムに格納されると、その数字のバイナリ表現に設定されているビットからカラム値のセットメンバーが特定されます。カラムが SET('a','b','c','d') として指定されている場合、セットメンバーは次の 10 進値と 2 進値を持ちます。

SET メンバー	10 進値	2 進値
'a'	1	0001
'b'	2	0010
'c'	4	0100
'd'	8	1000

このカラムに 9 の値を割り当てた場合、2 進数では 1001 となるため、SET 値の最初と 4 番目のメンバーである 'a' と 'd' が選択され、結果として得られる値は 'a,d' になります。

1 つ以上の SET 要素を含む値には、値を挿入するときに要素がどの順序でリストされているかは関係ありません。また、所定の要素が値の中で何回リストされているかも関係ありません。あとから値を取得するときに、値内のそれぞれの要素は、テーブル作成時に指定された順序に従って、一度表示されます。たとえば、カラムが SET('a','b','c','d') と指定されているとします。

```
mysql> CREATE TABLE myset (col SET('a', 'b', 'c', 'd'));
```

'a,d'、'd,a'、'a,d,d'、'a,d,a'、および 'd,a,d' の値を挿入した場合、

```
mysql> INSERT INTO myset (col) VALUES
-> ('a,d'), ('d,a'), ('a,d,a'), ('a,d,d'), ('d,a,d');
Query OK, 5 rows affected (0.01 sec)
Records: 5  Duplicates: 0  Warnings: 0
```

これらの値はすべて、取得されるときに 'a,d' と表示されます。

```
mysql> SELECT col FROM myset;
+------+
| col  |
```

```
+------+
| a,d  |
| a,d  |
| a,d  |
| a,d  |
| a,d  |
+------+
5 rows in set (0.04 sec)
```

サポートされていない値に SET カラムを設定すると、その値は無視され警告が表示されます。

```
mysql> INSERT INTO myset (col) VALUES ('a,d,d,s');
Query OK, 1 row affected, 1 warning (0.03 sec)

mysql> SHOW WARNINGS;
+---------+------+-----------------------------------------+
| Level   | Code | Message                                 |
+---------+------+-----------------------------------------+
| Warning | 1265 | Data truncated for column 'col' at row 1 |
+---------+------+-----------------------------------------+
1 row in set (0.04 sec)

mysql> SELECT col FROM myset;
+------+
| col  |
+------+
| a,d  |
| a,d  |
| a,d  |
| a,d  |
| a,d  |
| a,d  |
+------+
6 rows in set (0.01 sec)
```

厳密な SQL モードが有効な場合、無効な SET 値を挿入しようとするとエラーが発生します。

SET 値は数値でソートされます。NULL 値は非 NULL SET 値の前にソートされます。

数値引数を取る SUM() や AVG() などの関数は、必要に応じて引数を数値にキャストします。SET 値の場合は、キャスト操作によって数値が使用されます。

通常は、FIND_IN_SET() 関数か LIKE 演算子を使用して SET 値を検索します。

```
mysql> SELECT * FROM tbl_name WHERE FIND_IN_SET('value',set_col)>0;
mysql> SELECT * FROM tbl_name WHERE set_col LIKE '%value%';
```

最初のステートメントは set_col が value セットメンバーを含む行を見つけます。2 番目も似ていますが、同じではありません。ほかのセットメンバーの部分文字列としてであっても、set_col が value を含む行を見つけます。

次のステートメントも使用できます。

```
mysql> SELECT * FROM tbl_name WHERE set_col & 1;
mysql> SELECT * FROM tbl_name WHERE set_col = 'val1,val2';
```

これらのうち最初のステートメントが最初のセットメンバーを含む値を探します。2 番目のステートメントは正確に一致する値を探します。2 番目の型は慎重に比較してください。セット値を 'val1,val2' と比較すると、値を 'val2,val1' と比較した場合とは異なる結果が返されます。カラム定義にリストされている順序どおりに値を指定する必要があります。

SET カラムの指定可能な値をすべて特定するには、SHOW COLUMNS FROM tbl_name LIKE set_col を使用して、出力の Type カラム内の SET 定義を構文解析します。

C API では、SET 値は文字列として返されます。結果セットのメタデータを使用してこれらをほかの文字列から区別する方法については、セクション23.7.5「C API データ構造」を参照してください。

11.5 空間データの拡張

Open Geospatial Consortium (OGC) は、空間データを管理するあらゆる種類のアプリケーションで役立つ、公的に利用可能な概念的ソリューションの開発に携わっている 250 以上の企業、機関、および大学の国際的なコンソーシアムです。

空間データをサポートするように SQL RDBMS を拡張するための複数の概念的な方法を提案したドキュメントとして、Open Geospatial Consortium から「OpenGIS® Implementation Standard for Geographic information

- Simple feature access - Part 2: SQL option」が発行されています。この仕様書は、OGC Web サイト (http://www.opengeospatial.org/standards/sfs) から入手できます。

MySQL は、OGC の仕様書に従って、幾何型を含む SQL 環境のサブセットとして空間拡張を実装しています。この用語は、一連の幾何型で拡張された SQL 環境を意味しています。幾何値を含む SQL カラムは、幾何型のカラムとして実装されています。仕様書では、一連の SQL 幾何型のほか、幾何値を作成し分析するためにこれらの型に対して行われる関数について説明しています。

MySQL 空間拡張により、地理的特性の生成、ストレージ、および分析が可能になります。

- 空間値を表すデータ型
- 空間値を操作する関数
- 空間カラムへのアクセス時間を改善するための空間インデックス設定

データ型と関数は、MyISAM、InnoDB、NDB、および ARCHIVE テーブルで使用できます。空間カラムのインデックス設定については、MyISAM は、SPATIAL インデックスと非 SPATIAL インデックスの両方をサポートします。その他のストレージエンジンは、セクション13.1.13「CREATE INDEX 構文」で説明しているように、非 SPATIAL インデックスをサポートします。

地理的特性とは、位置を特定できる世界中のあらゆるもののことです。特性は次のいずれかになります。

- 実体。山、池、都市など。
- 領域。町の区域や熱帯地域など。
- 定義可能な位置。2 つの道路が交差する特定の場所となる交差点など。

ドキュメントによっては、地理空間特性という用語を地理的特性の意味で使用している場合もあります。

幾何図形も地理的特性を表す用語です。幾何図形という用語はもともと、地球の測量を意味していました。地図作成者が世界のマッピングに使用する幾何特性を指す別の意味は、地図作成の分野からのものです。

ここでの説明では、地理的特性、地理空間特性、特性、幾何図形の用語をシノニムと見なします。もっともよく使用される用語は幾何図形であり、位置を特定できる世界中のあらゆるものを表す地点または地点の集約として定義されています。

次の資料では次のトピックを取り上げます。

- MySQL モデルに実装された空間データ型
- OpenGIS 幾何モデルでの空間拡張の基本
- 空間データを表現するためのデータ形式
- MySQL で空間データを使用する方法
- 空間データのインデックスの使用方法
- OpenGIS 仕様と MySQL 実装との差異

空間データを演算する関数の詳細は、セクション12.15「空間分析関数」を参照してください。

MySQL の GIS に対する適合性と互換性

MySQL は次の GIS 機能を実装していません。

- 追加のメタデータビュー

  OpenGIS の仕様書では追加メタデータビューがいくつか提案されています。たとえば、GEOMETRY_COLUMNS という名前のシステムビューには、データベース内の幾何カラムごとに 1 行ずつ、幾何カラムの記述が含まれます。

- LineString および MultiLineString での OpenGIS 関数 Length() は、MySQL で GLength() として呼び出す必要があります。

  これは、文字列値の長さを計算する既存の SQL 関数 Length() が存在していますが、この関数がテキスト、空間のどちらのコンテキストで呼び出されたのかを判定できない場合があるからです。

追加のリソース

- 空間データをサポートするように SQL RDBMS を拡張するための複数の概念的な方法を提案したドキュメントとして、Open Geospatial Consortium から「OpenGIS® Implementation Standard for Geographic information - Simple feature access - Part 2: SQL option」が発行されています。Open Geospatial Consortium (OGC) は、http://www.opengeospatial.org/ で Web サイトを管理しています。仕様書は http://www.opengeospatial.org/standards/sfs で入手できます。ここでの資料に関連した追加情報が用意されています。
- MySQL に対する空間拡張の使用について質問や関心がある場合は、GIS フォーラム (https://forums.mysql.com/list.php?23) で議論できます。

11.5.1 空間データ型

MySQL には OpenGIS クラスに対応するデータ型が用意されています。これらのデータ型の中には、次のように単一の幾何値を格納するものがあります。

- GEOMETRY
- POINT
- LINESTRING
- POLYGON

GEOMETRY にはどの型の幾何値でも格納できます。その他の単一値型 (POINT、LINESTRING、および POLYGON) では、特定の幾何型に値が制限されます。

次に示すその他のデータ型には、値のコレクションが格納されます。

- MULTIPOINT
- MULTILINESTRING
- MULTIPOLYGON
- GEOMETRYCOLLECTION

GEOMETRYCOLLECTION には、任意の型のオブジェクトのコレクションを格納できます。その他のコレクション型 (MULTIPOINT、MULTILINESTRING、MULTIPOLYGON、および GEOMETRYCOLLECTION) では、コレクションのメンバーは、特定の幾何型を持つメンバーに制限されます。

MySQL 空間データは、セクション11.5.2「OpenGIS 幾何モデル」で説明しているように、OpenGIS 幾何モデルに基づいています。MySQL で空間データ型を使用する方法を示した例については、セクション11.5.3「空間データの使用」を参照してください。

11.5.2 OpenGIS 幾何モデル

OGC の幾何型を含む SQL 環境で提案されている一連の幾何型は、OpenGIS 幾何モデルに基づいています。このモデルの各幾何オブジェクトには、次のような一般的なプロパティーがあります。

- オブジェクトが定義されている座標空間を記述する空間参照システムに関連付けられています。
- 特定の幾何クラスに属しています。

11.5.2.1 幾何クラスの階層

幾何クラスの階層は次のように定義されています。

- Geometry (インスタンス化不可能)
  - Point (インスタンス化可能)
  - Curve (インスタンス化不可能)
    - LineString (インスタンス化可能)
      - Line

    - LinearRing
  - Surface (インスタンス化不可能)
    - Polygon (インスタンス化可能)
  - GeometryCollection (インスタンス化可能)
    - MultiPoint (インスタンス化可能)
    - MultiCurve (インスタンス化不可能)
      - MultiLineString (インスタンス化可能)
    - MultiSurface (インスタンス化不可能)
      - MultiPolygon (インスタンス化可能)

インスタンス化不可能なクラスのオブジェクトは作成できません。インスタンス化可能なクラスのオブジェクトは作成できます。どのクラスもプロパティーを持ちますが、インスタンス化可能なクラスはさらに表明 (有効なクラスインスタンスを定義するルール) も持つことができます。

Geometry は基本クラスです。これは抽象クラスです。Geometry のインスタンス化可能なサブクラスは、2 次元座標空間内に存在する 0 次元、1 次元、および 2 次元の幾何オブジェクトに限定されます。インスタンス化可能な幾何クラスはすべて、幾何クラスの有効なインスタンスが位相的に閉じている (つまり、定義されたすべての幾何図形に境界が含まれる) ように定義されています。

Geometry 基本クラスには、Point、Curve、Surface、および GeometryCollection のサブクラスがあります。

- Point は 0 次元のオブジェクトを表します。
- Curve は 1 次元のオブジェクトを表し、そのサブクラス LineString は、Line と LinearRing をサブクラスに持ちます。
- Surface は 2 次元のオブジェクト用に設計されたもので、Polygon をサブクラスに持ちます。
- GeometryCollection には MultiPoint、MultiLineString、MultiPolygon という 0、1、2 次元の特殊化コレクションクラスが用意されており、それぞれ Points、LineStrings、Polygons のコレクションに対応する幾何図形をモデル化しています。MultiCurve と MultiSurface は、このコレクションインタフェースを汎化して Curves および Surfaces を処理できるよう抽象スーパークラスとして導入されたものです。

Geometry、Curve、Surface、MultiCurve、および MultiSurface は、インスタンス化不可能なクラスとして定義されています。これらはサブクラスに共通する一連のメソッドを定義しており、今後の拡張に含められます。

Point、LineString、Polygon、GeometryCollection、MultiPoint、MultiLineString、および MultiPolygon はインスタンス化可能なクラスです。

11.5.2.2 Geometry クラス

Geometry は階層のルートクラスです。これはインスタンス化不可能なクラスですが、次のリストに説明しているように、Geometry サブクラスのいずれかから作成したすべての幾何値に共通である多数のプロパティーがあります。個々のサブクラスも独自のプロパティーを備えていますが、これについては後述します。

Geometry のプロパティー

幾何値に含まれるプロパティーは次のとおりです。

- その型。各幾何図形は、階層内のインスタンス化可能クラスのいずれかに属します。
- その SRID、つまり空間参照識別子。この値は、幾何図形に関連付けられた、その幾何オブジェクトが定義されている座標空間を記述する空間参照システムを識別します。

  MySQL の SRID 値は、幾何値に関連付けられた整数です。すべての計算はユークリッド (平面) 幾何学を前提にして実行されます。使用可能な SRID の最大値は $2^{32}-1$ です。より大きな値が指定されると、低位の 32 ビットだけが使用されます。
- 空間参照システムでの座標は、倍精度 (8 バイト) 数として表現されます。空でない幾何図形には必ず、(X,Y) 座標ペアが少なくとも 1 つ含まれます。空の幾何図形には座標は含まれません。

座標は SRID に対する相対的なものです。たとえば、異なる座標系では、オブジェクトの座標が同じ場合でも、2 つのオブジェクト間の距離が異なることがあります。これは、平面座標系での距離と地球を中心とした系 (地球表面の座標) の距離は異なるためです。

- 内部、境界、外部。

  幾何図形は必ず、ある位置の領域を占有します。幾何図形の外部とは、その幾何図形によって占有されていないすべての領域のことです。内部とは、その幾何図形によって占有されている領域のことです。境界とは、幾何図形の内部と外部が接する部分のことです。

- その MBR (最小外接矩形)、または包絡線。これは範囲を規定する幾何図形であり、次のように最小および最大の (X,Y) 座標から形成されます。

  ```
  ((MINX MINY, MAXX MINY, MAXX MAXY, MINX MAXY, MINX MINY))
  ```

- 値が単純である、単純でないのいずれであるか。LineString、MultiPoint、MultiLineString の型の幾何値は「単純である」、「単純でない」のいずれかになります。「単純である」、「単純でない」のいずれであるかの表明は、型ごとに決定されます。

- 値が閉じている、閉じていないのいずれであるか。LineString、MultiString の型の幾何値は「閉じている」「閉じていない」のいずれかになります。「閉じている」、「閉じていない」のいずれであるかの表明は、型ごとに決定されます。

- 値が空である、空でないのいずれであるか。点を 1 つも含まない幾何図形は空です。空の幾何図形の外部、内部、および境界は定義されていません (つまり、それらは NULL 値で表されます)。空の幾何図形は、常に単純で面積が 0 になるように定義されています。

- その次元。幾何図形には −1、0、1、または 2 の次元があります。

  - −1 は、空の幾何図形を表します。
  - 0 は長さも面積も持たない幾何図形を表します。
  - 1 は、長さがゼロ以外で面積がゼロの幾何図形を表します。
  - 2 は、面積がゼロ以外の幾何図形を表します。

  Point オブジェクトの次元は 0 です。LineString オブジェクトの次元は 1 です。Polygon オブジェクトの次元は 2 です。MultiPoint、MultiLineString、および MultiPolygon オブジェクトの次元は、構成要素の次元と同じになります。

11.5.2.3 Point クラス

Point は、座標空間内の単一の位置を表す幾何図形です。

Point の例

- 多数の都市を含む大規模な世界地図を想像してください。Point オブジェクトは各都市を表すことができます。
- 市内地図で、Point オブジェクトはバス停を表すことができます。

Point のプロパティー

- X 座標値。
- Y 座標値。
- Point は 0 次元の幾何図形として定義されています。
- Point の境界は空セットになります。

11.5.2.4 Curve クラス

Curve は 1 次元の幾何図形であり、通常は一連の点で表されます。点の間の補間方法は、Curve の特定のサブクラスで定義されています。Curve はインスタンス化不可能なクラスです。

Curve のプロパティー

- Curve はその点の座標を持ちます。

- Curve は 1 次元の幾何図形として定義されています。
- 同じ点を 2 度通過しなければ、Curve は単純です。
- 始点と終点が等しい場合、Curve は閉じています。
- 閉じた Curve の境界は、空になります。
- 閉じていない Curve の境界は、その 2 つの端点から構成されます。
- 単純で閉じた Curve としては、LinearRing が挙げられます。

11.5.2.5 LineString クラス

LineString は、点の間を直線で補間した Curve です。

LineString の例

- 世界地図で、LineString オブジェクトは河川を表すことができます。
- 市内地図で、LineString オブジェクトは通りを表すことができます。

LineString のプロパティー

- LineString は、隣り合う 1 対の点で定義される各線分の座標を持ちます。
- ちょうど 2 つの点から構成されている場合、LineString は Line になります。
- 閉じていて、かつ単純である場合は、LineString は LinearRing になります。

11.5.2.6 Surface クラス

Surface は 2 次元の幾何図形です。これはインスタンス化不可能なクラスです。その唯一のインスタンス化可能なサブクラスは、Polygon です。

Surface のプロパティー

- Surface は 2 次元の幾何図形として定義されています。
- OpenGIS 仕様では、単純な Surface が、1 個の外側の境界と 0 個以上の内側の境界に関連付けられた単一の「パッチ」からなる幾何図形として定義されています。
- 単純な Surface の境界は、その外側と内側の境界に対応する一連の閉じた曲線になります。

11.5.2.7 Polygon クラス

Polygon は、多辺の幾何図形を表す平面 Surface です。これは 1 個の外側の境界と 0 個以上の内側の境界で定義されますが、それらの内側の各境界によって Polygon 内の 1 個の穴が定義されます。

Polygon の例

- 地域マップで、Polygon オブジェクトは森林や区域などを表すことができます。

Polygon の表明

- Polygon の境界は、外側と内側の境界を構成する一連の LinearRing オブジェクト (つまり、単純かつ閉じた LineString オブジェクト) から構成されます。
- Polygon のリングは交差しません。Polygon の境界に含まれるリングは、Point で交わりますが、接することしかできません。
- Polygon には線分、突起、亀裂は含まれません。
- Polygon は、連続した点集合からなる内部を持ちます。
- Polygon は穴を持つことができます。穴のある Polygon の外部は、連続していません。それぞれの穴が、連続した 1 つの外部コンポーネントを定義します。

以上の表明により、Polygon は単純な幾何図形になります。

11.5.2.8 GeometryCollection クラス

GeometryCollection は、任意のクラスに属する 1 つ以上の幾何図形のコレクションとなる幾何図形です。

GeometryCollection の各要素の空間参照システム (つまり座標系) はすべて同じである必要があります。GeometryCollection の要素に関する制約はこれだけですが、後続の各セクションで説明する GeometryCollection のサブクラスでは、メンバーシップに関する制限が課される可能性があります。これらの制限は次の情報に基づくことがあります。

- 要素の型 (たとえば、MultiPoint に格納できるのは Point 要素だけです)
- 次元
- 要素間の空間的な重なり具合に関する制約

11.5.2.9 MultiPoint クラス

MultiPoint は、Point 要素から構成される幾何図形コレクションです。点の接続や順序付けは一切行われません。

MultiPoint の例

- 世界地図で、MultiPoint は一連の小さな島々を表すことができます。
- 市内地図で、MultiPoint はチケットオフィスの系列店を表すことができます。

MultiPoint のプロパティー

- MultiPoint は 0 次元の幾何図形です。
- この 2 つの Point の値 (座標値) が等しくない場合は、MultiPoint は単純になります。
- MultiPoint の境界は空セットになります。

11.5.2.10 MultiCurve クラス

MultiCurve は、Curve 要素から構成される幾何図形コレクションです。MultiCurve はインスタンス化不可能なクラスです。

MultiCurve のプロパティー

- MultiCurve は 1 次元の幾何図形です。
- MultiCurve が単純になるのは、そのすべての要素が単純である場合だけです。2 つの要素の唯一の交点は、両方の要素の境界上にある点になります。
- MultiCurve の境界を取得するには、「mod 2 union ルール」 (「odd-even ルール」とも呼ばれます) を適用します。ある点が MultiCurve の境界に含まれるのは、その点が、奇数個の MultiCurve 要素の境界に含まれている場合です。
- すべての要素が閉じている場合、MultiCurve は閉じています。
- 閉じた MultiCurve の境界は、常に空になります。

11.5.2.11 MultiLineString クラス

MultiLineString は、LineString 要素から構成される MultiCurve 幾何図形コレクションです。

MultiLineString の例

- 地域マップで、MultiLineString は河川系や高速道路システムを表すことができます。

11.5.2.12 MultiSurface クラス

MultiSurface は、面要素から構成される幾何図形コレクションです。MultiSurface はインスタンス化不可能なクラスです。その唯一のインスタンス化可能なサブクラスは、MultiPolygon です。

MultiSurface の表明

- MultiSurface の 2 つの面の内部が交差することはありません。

- MultiSurface の 2 つの要素の境界が無限個の点で交わることはありません。

11.5.2.13 MultiPolygon クラス

MultiPolygon は、Polygon 要素から構成される MultiSurface オブジェクトです。

MultiPolygon の例

- 地域マップで、MultiPolygon は湖の系列を表すことができます。

MultiPolygon の表明

- MultiPolygon のどの 2 つの Polygon 要素も、交差する内部を持つことはありません。
- MultiPolygon のどの 2 つの Polygon 要素も、交差したり (交差は 1 つ前の表明でも禁止されています)、無限個の点で接したりしません。
- MultiPolygon にカットライン、突起、亀裂を含めることはできません。MultiPolygon は通常の閉じた点集合です。
- 複数の Polygon を含む MultiPolygon は、連続していない内部を持ちます。MultiPolygon の連続する内部コンポーネントの個数は、MultiPolygon 内の Polygon 値の数と等しくなります。

MultiPolygon のプロパティー

- MultiPolygon は 2 次元の幾何図形です。
- MultiPolygon の境界は、その Polygon 要素の境界に対応する一連の閉じた曲線 (LineString 値) になります。
- MultiPolygon の境界に含まれる各 Curve は、どれか 1 つの Polygon 要素の境界にのみ含まれます。
- Polygon 要素の境界に含まれる Curve は必ず、MultiPolygon の境界にも含まれます。

11.5.3 空間データの使用

このセクションでは、空間データ型カラムを含むテーブルの作成方法と、空間情報の操作方法について説明します。

11.5.3.1 サポートされる空間データ形式

クエリーで幾何オブジェクトを表現するために、次の 2 つの標準空間データ形式が使用されます。

- WKT (Well-Known Text) 形式
- WKB (Well-Known Binary) 形式

MySQL の内部では、WKT、WKB のどちらの形式とも異なる形式で幾何値が格納されます。

異なるデータ形式間の変換に使用できる関数があります。セクション12.15.6「幾何形式変換関数」を参照してください。

WKT (Well-Known Text) 形式

幾何値の WKT (Well-Known Text) 表現は、ASCII 形式の幾何データを交換するために設計されています。OpenGIS 仕様書には、WKT 値を書き込むための公式の運用ルールを指定するバッカス-ナウア記法が用意されています (セクション11.5「空間データの拡張」を参照してください)。

幾何オブジェクトの WKT 表現の例:

- Point:

```
POINT(15 20)
```

点の座標は、区切り用のカンマなしに指定されます。これは、座標間にカンマを必要とする SQL Point() 関数の構文とは異なります。特定の空間演算のコンテキストに適した構文を慎重に使用してください。たとえば、次のステートメントはどちらも、Point オブジェクトから X 座標を抽出します。最初の場合は、Point() 関数を直接使用してオブジェクトを生成します。2 番目の場合は、GeomFromText() で Point に変換された WKT 表現を使用します。

```
mysql> SELECT X(POINT(15, 20));
```

```
+------------------+
| X(POINT(15, 20)) |
+------------------+
|               15 |
+------------------+

mysql> SELECT X(GeomFromText('POINT(15 20)'));
+-----------------------------------+
| X(GeomFromText('POINT(15 20)')) |
+-----------------------------------+
|                              15 |
+-----------------------------------+
```

- 4 つの点を含む LineString:

```
LINESTRING(0 0, 10 10, 20 25, 50 60)
```

  点の座標のペアはカンマで区切られます。

- 外側のリングと内側のリングを 1 つずつ含む Polygon:

```
POLYGON((0 0,10 0,10 10,0 10,0 0),(5 5,7 5,7 7,5 7, 5 5))
```

- 3 つの Point 値を含む MultiPoint:

```
MULTIPOINT(0 0, 20 20, 60 60)
```

- 2 つの LineString 値を含む MultiLineString:

```
MULTILINESTRING((10 10, 20 20), (15 15, 30 15))
```

- 2 つの Polygon 値を含む MultiPolygon:

```
MULTIPOLYGON(((0 0,10 0,10 10,0 10,0 0)),((5 5,7 5,7 7,5 7, 5 5)))
```

- 2 つの Point 値と 1 つの LineString から構成された GeometryCollection:

```
GEOMETRYCOLLECTION(POINT(10 10), POINT(30 30), LINESTRING(15 15, 20 20))
```

WKB (Well-Known Binary) 形式

幾何値の WKB (Well-Known Binary) 表現は、幾何 WKB 情報を含む BLOB 値によって表現されたバイナリストリームとして幾何データを交換するために使用されます。この形式は、OpenGIS 仕様によって定義されています (セクション11.5「空間データの拡張」を参照してください)。これはまた、ISO の SQL/MM Part 3: Spatial 標準でも定義されています。

WKB は、1 バイトの符号なしの整数、4 バイトの符号なしの整数、および 8 バイトの倍精度数 (IEEE 754 形式) を使用します。1 バイトは 8 ビットです。

たとえば、POINT(1 1) に対応する WKB 値は、それぞれ 2 つの 16 進数で表された次の 21 バイトのシーケンスから構成されます。

```
010100000000000000000F03F000000000000F03F
```

このシーケンスは次のコンポーネントから構成されます。

```
Byte order:   01
WKB type:     01000000
X coordinate: 000000000000F03F
Y coordinate: 000000000000F03F
```

各コンポーネントが表す内容は次のとおりです。

- バイト順序は 1 または 0 のどちらかで、リトルエンディアンまたはビッグエンディアンストレージを示します。リトルエンディアンバイト順序、ビッグエンディアンバイト順序はそれぞれ NDR (Network Data Representation)、XDR (External Data Representation) とも呼ばれます。

- WKB 型は幾何型を示すコードです。1 から 7 の値は Point、LineString、Polygon、MultiPoint、MultiLineString、MultiPolygon、および GeometryCollection を示します。

- Point 値には X 座標と Y 座標が含まれますが、それぞれ倍精度値として表現されます。

さらに複雑な幾何値の WKB 値は、OpenGIS 仕様書に詳しく記されているように、より複雑なデータ構造になります。

11.5.3.2 空間カラムの作成

MySQL には、CREATE TABLE や ALTER TABLE を使用する方法など、幾何型の空間カラムを作成するための標準的な方法が用意されています。空間カラムは、MyISAM、InnoDB、NDB、および ARCHIVE テーブルでサポートされています。セクション11.5.3.6「空間インデックスの作成」の空間インデックスに関するノートも参照してください。

- 空間カラムを含むテーブルを作成するには、CREATE TABLE ステートメントを使用します。

```
CREATE TABLE geom (g GEOMETRY);
```

- 既存のテーブルに対して空間カラムの追加や削除を行うには、ALTER TABLE ステートメントを使用します。

```
ALTER TABLE geom ADD pt POINT;
ALTER TABLE geom DROP pt;
```

11.5.3.3 空間カラムへのデータ移入

空間カラムを作成し終わったら、空間データを移入できます。

値は内部幾何形式で格納する必要がありますが、WKT (Well-Known Text)、WKB (Well-Known Binary) のいずれの形式からでも、その形式に値を変換できます。次の例は、WKT 値を内部幾何形式に変換することによって、幾何値をテーブルに挿入する方法を示しています。

- 次のように INSERT ステートメント内で直接変換を実行します。

```
INSERT INTO geom VALUES (GeomFromText('POINT(1 1)'));

SET @g = 'POINT(1 1)';
INSERT INTO geom VALUES (GeomFromText(@g));
```

- 次のように INSERT の前に変換を実行します。

```
SET @g = GeomFromText('POINT(1 1)');
INSERT INTO geom VALUES (@g);
```

次の例では、より複雑な幾何図形をテーブルに挿入しています。

```
SET @g = 'LINESTRING(0 0,1 1,2 2)';
INSERT INTO geom VALUES (GeomFromText(@g));

SET @g = 'POLYGON((0 0,10 0,10 10,0 10,0 0),(5 5,7 5,7 7,5 7, 5 5))';
INSERT INTO geom VALUES (GeomFromText(@g));

SET @g =
'GEOMETRYCOLLECTION(POINT(1 1),LINESTRING(0 0,1 1,2 2,3 3,4 4))';
INSERT INTO geom VALUES (GeomFromText(@g));
```

前述の例では、GeomFromText() を使用して幾何値を作成しています。次のように型に固有の関数を使用することもできます。

```
SET @g = 'POINT(1 1)';
INSERT INTO geom VALUES (PointFromText(@g));

SET @g = 'LINESTRING(0 0,1 1,2 2)';
INSERT INTO geom VALUES (LineStringFromText(@g));

SET @g = 'POLYGON((0 0,10 0,10 10,0 10,0 0),(5 5,7 5,7 7,5 7, 5 5))';
INSERT INTO geom VALUES (PolygonFromText(@g));

SET @g =
'GEOMETRYCOLLECTION(POINT(1 1),LINESTRING(0 0,1 1,2 2,3 3,4 4))';
INSERT INTO geom VALUES (GeomCollFromText(@g));
```

幾何値の WKB 表現を使用するクライアントアプリケーションプログラムが、クエリーで正しく作成された WKB のサーバーへの送信を担います。この要件を満たす方法は複数あります。例:

- 次のように、16 進リテラル構文を使用して、POINT(1 1) 値を挿入します。

```
mysql> INSERT INTO geom VALUES
```

```
  -> (GeomFromWKB(0x010100000000000000000F03F000000000000F03F));
```

- ODBC アプリケーションは、BLOB 型の引数を使用して WKB 表現をプレースホルダにバインドし、WKB 表現を送信できます。

```
INSERT INTO geom VALUES (GeomFromWKB(?))
```

  ほかのプログラミングインタフェースも似たようなプレースホルダメカニズムをサポートしている可能性があります。

- C プログラムでは、mysql_real_escape_string() を使用してバイナリ値をエスケープし、その結果をクエリー文字列に含めてサーバーに送信できます。セクション23.7.7.54「mysql_real_escape_string()」を参照してください。

11.5.3.4 空間データのフェッチ

テーブルに格納された幾何値は内部形式でフェッチできます。WKT 形式から WKB 形式に変換することもできます。

- 内部形式での空間データのフェッチ:

  内部形式で幾何値をフェッチする方法は、テーブル間でデータの転送を行う場合に便利です。

```
CREATE TABLE geom2 (g GEOMETRY) SELECT g FROM geom;
```

- WKT 形式での空間データのフェッチ:

  AsText() 関数は幾何図形を内部形式から WKT 文字列に変換します。

```
SELECT AsText(g) FROM geom;
```

- WKB 形式での空間データのフェッチ:

  AsBinary() 関数は幾何図形を内部形式から WKB 値を含む BLOB に変換します。

```
SELECT AsBinary(g) FROM geom;
```

11.5.3.5 空間分析の最適化

MyISAM テーブルの場合、空間データを含むカラムでの検索操作は、SPATIAL インデックスを使用して最適化できます。もっとも典型的な操作は次のとおりです。

- 指定された点を含むすべてのオブジェクトを検索する点クエリー
- 所定の領域と重なるすべてのオブジェクトを検索する領域クエリー

MySQL では、2 次分割 R ツリーを使用して空間カラムの SPATIAL インデックスが実装されています。SPATIAL インデックスは、幾何図形の最小外接矩形 (MBR) を使用して構築されます。大部分の幾何図形では、MBR はその幾何図形を囲む最小矩形となります。水平または垂直方向のライン文字列では、MBR は矩形からライン文字列に縮退します。点の場合、MBR は矩形から点に縮退します。

空間カラムに通常のインデックスを作成することも可能です。非 SPATIAL インデックスでは、POINT カラムを除くすべての空間カラムでプリフィクスを宣言する必要があります。

MyISAM は、SPATIAL インデックスと非 SPATIAL インデックスの両方をサポートします。その他のストレージエンジンはセクション13.1.13「CREATE INDEX 構文」で説明しているように、非 SPATIAL インデックスをサポートします。

11.5.3.6 空間インデックスの作成

MyISAM テーブルでは、MySQL は、通常のインデックスを作成するための似た構文を使用するが、SPATIAL キーワードを使用して、空間インデックスを作成できます。空間インデックスのカラムは、NOT NULL と宣言する必要があります。次の各例では空間インデックスの作成方法を示します。

- CREATE TABLE を使用する場合:

```
CREATE TABLE geom (g GEOMETRY NOT NULL, SPATIAL INDEX(g)) ENGINE=MyISAM;
```

- ALTER TABLE を使用する場合:

```
ALTER TABLE geom ADD SPATIAL INDEX(g);
```

- CREATE INDEX を使用する場合:

```
CREATE SPATIAL INDEX sp_index ON geom (g);
```

SPATIAL INDEX は R ツリーインデックスを作成します。空間カラムの非空間インデックスをサポートするストレージエンジンでは、B ツリーインデックスが作成されます。空間値に対する B ツリーインデックスは、正確な値の検索に役立ちますが、範囲スキャンには役立ちません。

空間カラムのインデックス作成の詳細については、セクション13.1.13「CREATE INDEX 構文」を参照してください。

空間インデックスを削除するには、次のように ALTER TABLE または DROP INDEX を使用します。

- ALTER TABLE を使用する場合:

```
ALTER TABLE geom DROP INDEX g;
```

- DROP INDEX を使用する場合:

```
DROP INDEX sp_index ON geom;
```

例: テーブル geom に 32,000 件を超える幾何図形が含まれていて、それらの図形が型 GEOMETRY のカラム g に格納されているものとします。またこのテーブルには、オブジェクト ID の値を格納するための AUTO_INCREMENT カラム fid も含まれています。

```
mysql> DESCRIBE geom;
+-------+----------+------+-----+---------+----------------+
| Field | Type     | Null | Key | Default | Extra          |
+-------+----------+------+-----+---------+----------------+
| fid   | int(11)  |      | PRI | NULL    | auto_increment |
| g     | geometry |      |     |         |                |
+-------+----------+------+-----+---------+----------------+
2 rows in set (0.00 sec)

mysql> SELECT COUNT(*) FROM geom;
+----------+
| count(*) |
+----------+
|    32376 |
+----------+
1 row in set (0.00 sec)
```

カラム g に空間インデックスを追加するには、次のステートメントを使用します。

```
mysql> ALTER TABLE geom ADD SPATIAL INDEX(g) ENGINE=MyISAM;
Query OK, 32376 rows affected (4.05 sec)
Records: 32376  Duplicates: 0  Warnings: 0
```

11.5.3.7 空間インデックスの使用

オプティマイザは、WHERE 句で MBRContains() や MBRWithin() などの関数が使用されているクエリーの検索に、使用可能な空間インデックスを含めることができるかどうかを調べます。次のクエリーは、所定の矩形に含まれるすべてのオブジェクトを検索します。

```
mysql> SET @poly =
    -> 'Polygon((30000 15000,
                 31000 15000,
                 31000 16000,
                 30000 16000,
                 30000 15000))';
mysql> SELECT fid,AsText(g) FROM geom WHERE
    -> MBRContains(GeomFromText(@poly),g);
+-----+-------------------------------------------------------------+
| fid | AsText(g)                                                   |
+-----+-------------------------------------------------------------+
|  21 | LINESTRING(30350.4 15828.8,30350.6 15845,30333.8 15845,30 ... |
|  22 | LINESTRING(30350.6 15871.4,30350.6 15887.8,30334 15887.8, ... |
|  23 | LINESTRING(30350.6 15914.2,30350.6 15930.4,30334 15930.4, ... |
|  24 | LINESTRING(30290.2 15823,30290.2 15839.4,30273.4 15839.4, ... |
|  25 | LINESTRING(30291.4 15866.2,30291.6 15882.4,30274.8 15882. ... |
|  26 | LINESTRING(30291.6 15918.2,30291.6 15934.4,30275 15934.4, ... |
| 249 | LINESTRING(30337.8 15938.6,30337.8 15946.8,30320.4 15946. ... |
|   1 | LINESTRING(30250.4 15129.2,30248.8 15138.4,30238.2 15136. ... |
```

```
|   2 | LINESTRING(30220.2 15122.8,30217.2 15137.8,30207.6 15136, ... |
|   3 | LINESTRING(30179 15114.4,30176.6 15129.4,30167 15128,3016 ... |
|   4 | LINESTRING(30155.2 15121.4,30140.4 15118.6,30142 15109,30 ... |
|   5 | LINESTRING(30192.4 15085,30177.6 15082.2,30179.2 15072.4, ... |
|   6 | LINESTRING(30244 15087,30229 15086.2,30229.4 15076.4,3024 ... |
|   7 | LINESTRING(30200.6 15059.4,30185.6 15058.6,30186 15048.8, ... |
|  10 | LINESTRING(30179.6 15017.8,30181 15002.8,30190.8 15003.6, ... |
|  11 | LINESTRING(30154.2 15000.4,30168.6 15004.8,30166 15014.2, ... |
|  13 | LINESTRING(30105 15065.8,30108.4 15050.8,30118 15053,3011 ... |
| 154 | LINESTRING(30276.2 15143.8,30261.4 15141,30263 15131.4,30 ... |
| 155 | LINESTRING(30269.8 15084,30269.4 15093.4,30258.6 15093,30 ... |
| 157 | LINESTRING(30128.2 15011,30113.2 15010.2,30113.6 15000.4, ... |
+-----+-------------------------------------------------------------+
20 rows in set (0.00 sec)
```

このクエリーがどのように実行されているのかを、EXPLAIN を使用して確認します。

```
mysql> SET @poly =
    -> 'Polygon((30000 15000,
                 31000 15000,
                 31000 16000,
                 30000 16000,
                 30000 15000))';
mysql> EXPLAIN SELECT fid,AsText(g) FROM geom WHERE
    -> MBRContains(GeomFromText(@poly),g)\G
*************************** 1. row ***************************
           id: 1
  select_type: SIMPLE
        table: geom
         type: range
possible_keys: g
          key: g
      key_len: 32
          ref: NULL
         rows: 50
        Extra: Using where
1 row in set (0.00 sec)
```

空間インデックスがないとどうなるのかを確認します。

```
mysql> SET @poly =
    -> 'Polygon((30000 15000,
                 31000 15000,
                 31000 16000,
                 30000 16000,
                 30000 15000))';
mysql> EXPLAIN SELECT fid,AsText(g) FROM g IGNORE INDEX (g) WHERE
    -> MBRContains(GeomFromText(@poly),g)\G
*************************** 1. row ***************************
           id: 1
  select_type: SIMPLE
        table: geom
         type: ALL
possible_keys: NULL
          key: NULL
      key_len: NULL
          ref: NULL
         rows: 32376
        Extra: Using where
1 row in set (0.00 sec)
```

空間インデックスを使用せずに SELECT ステートメントを実行しても結果は同じになりますが、実行時間は 0.00 秒から 0.46 秒に増大します。

```
mysql> SET @poly =
    -> 'Polygon((30000 15000,
                 31000 15000,
                 31000 16000,
                 30000 16000,
                 30000 15000))';
mysql> SELECT fid,AsText(g) FROM geom IGNORE INDEX (g) WHERE
    -> MBRContains(GeomFromText(@poly),g);
+-----+-------------------------------------------------------------+
| fid | AsText(g)                                                   |
+-----+-------------------------------------------------------------+
|   1 | LINESTRING(30250.4 15129.2,30248.8 15138.4,30238.2 15136. ... |
|   2 | LINESTRING(30220.2 15122.8,30217.2 15137.8,30207.6 15136, ... |
|   3 | LINESTRING(30179 15114.4,30176.6 15129.4,30167 15128,3016 ... |
|   4 | LINESTRING(30155.2 15121.4,30140.4 15118.6,30142 15109,30 ... |
```

```
|   5 | LINESTRING(30192.4 15085,30177.6 15082.2,30179.2 15072.4, ... |
|   6 | LINESTRING(30244 15087,30229 15086.2,30229.4 15076.4,3024 ... |
|   7 | LINESTRING(30200.6 15059.4,30185.6 15058.6,30186 15048.8, ... |
|  10 | LINESTRING(30179.6 15017.8,30181 15002.8,30190.8 15003.6, ... |
|  11 | LINESTRING(30154.2 15000.4,30168.6 15004.8,30166 15014.2, ... |
|  13 | LINESTRING(30105 15065.8,30108.4 15050.8,30118 15053,3011 ... |
|  21 | LINESTRING(30350.4 15828.8,30350.6 15845,30333.8 15845,30 ... |
|  22 | LINESTRING(30350.6 15871.4,30350.6 15887.8,30334 15887.8, ... |
|  23 | LINESTRING(30350.6 15914.2,30350.6 15930.4,30334 15930.4, ... |
|  24 | LINESTRING(30290.2 15823,30290.2 15839.4,30273.4 15839.4, ... |
|  25 | LINESTRING(30291.4 15866.2,30291.6 15882.4,30274.8 15882. ... |
|  26 | LINESTRING(30291.6 15918.2,30291.6 15934.4,30275 15934.4, ... |
| 154 | LINESTRING(30276.2 15143.8,30261.4 15141,30263 15131.4,30 ... |
| 155 | LINESTRING(30269.8 15084,30269.4 15093.4,30258.6 15093,30 ... |
| 157 | LINESTRING(30128.2 15011,30113.2 15010.2,30113.6 15000.4, ... |
| 249 | LINESTRING(30337.8 15938.6,30337.8 15946.8,30320.4 15946. ... |
+-----+-------------------------------------------------------------+
20 rows in set (0.46 sec)
```

11.6 データ型デフォルト値

データ型仕様の DEFAULT value 句は、カラムのデフォルト値を示しています。例外が 1 つあります。デフォルト値は定数である必要があるので、関数または式にはできません。これは、たとえば日付カラムのデフォルト値に NOW() や CURRENT_DATE などの関数の値を設定できないことを意味します。例外では、CURRENT_TIMESTAMP を、TIMESTAMP および DATETIME カラムのデフォルトとして指定できます。セクション11.3.5「TIMESTAMP および DATETIME の自動初期化および更新機能」を参照してください。

BLOB および TEXT カラムにはデフォルト値を割り当てられません。

カラム定義に明示的な DEFAULT 値が含まれていない場合、MySQL はデフォルト値を次のように特定します。

NULL を値として取ることができる場合は、そのカラムは明示的な DEFAULT NULL 句で定義ができます。

NULL を値として取ることができない場合は、MySQL は明示的な DEFAULT 句でカラムを定義できません。例外: カラムが PRIMARY KEY の一部として定義されているが、NOT NULL として明示的には定義されていない場合、MySQL はこれを NOT NULL カラムとして作成します (PRIMARY KEY カラムは NOT NULL である必要があるので) が、暗黙的なデフォルト値を使用してこれに DEFAULT 句も割り当てます。これを防止するには、すべての PRIMARY KEY カラムの定義に明示的な NOT NULL を含めてください。

明示的な DEFAULT 句のない NOT NULL カラムに対するデータエントリでは、INSERT または REPLACE ステートメントにカラムの値を含まれていない場合、または UPDATE ステートメントがカラムを NULL に設定する場合、MySQL はその時点で有効な SQL モードに従ってカラムを処理します。

- 厳密な SQL モードを有効にした場合、トランザクションテーブルに対してエラーが発生し、ステートメントがロールバックされます。非トランザクションテーブルではエラーが起きるが、これが複数行ステートメントの 2 行目以降の行に対するエラーの場合、先行する行が挿入されています。
- 厳密モードが有効でない場合、MySQL はカラムデータ型の暗黙的なデフォルト値にカラムを設定します。

テーブル t が次のように定義されるとします。

```
CREATE TABLE t (i INT NOT NULL);
```

この場合、i は明示的なデフォルトがないので、厳密モードでは次のそれぞれはステートメントはエラーになり、行は挿入されません。厳密モードを使用しない場合、3 番目のステートメントだけでエラーが発生します。最初の 2 つのステートメントでは暗黙のデフォルトが挿入されますが、DEFAULT(i) が値を生成できないので 3 番目のステートメントは失敗します。

```
INSERT INTO t VALUES();
INSERT INTO t VALUES(DEFAULT);
INSERT INTO t VALUES(DEFAULT(i));
```

セクション5.1.7「サーバー SQL モード」を参照してください。

所定のテーブルに対して、SHOW CREATE TABLE ステートメントを使用すると、どのカラムに明示的な DEFAULT 句があるかを確認できます。

暗黙的なデフォルトは次のように定義されます。

- 数値型のデフォルトは 0 です。ただし、例外として AUTO_INCREMENT 属性で宣言された整数型または浮動小数点型のデフォルトは、そのシーケンスの次の値になります。

- TIMESTAMP 以外の日付と時間型のデフォルトには、「ゼロ」値が適切です。explicit_defaults_for_timestamp システム変数が有効な場合、これは TIMESTAMP にも当てはまります (セクション5.1.4「サーバーシステム変数」を参照してください)。それ以外の場合、テーブルの最初の TIMESTAMP カラムのデフォルト値は現在の日付と時間になります。セクション11.3「日付と時間型」を参照してください。
- ENUM ではない文字列型のデフォルト値は空の文字列です。ENUM のデフォルトは、最初の列挙値です。

整数カラム定義の中の SERIAL DEFAULT VALUE は NOT NULL AUTO_INCREMENT UNIQUE のエイリアスです。

11.7 データ型のストレージ要件

ディスク上のテーブルデータのストレージ要件は、複数の要因によって異なります。別々のストレージエンジンは異なる方法でデータ型を表し、ローデータを格納します。カラムか行全体のどちらかでテーブルデータを圧縮できますが、テーブルまたはカラムのストレージ要件の計算が複雑になります。

ディスク上のストレージレイアウトが違っていても、テーブル行に関する情報を通信および交換する内部 MySQL API は、すべてのストレージエンジンにわたって適用される一貫したデータ構造を使用します。

このセクションでは、データ型の固定サイズ表現を使用するストレージエンジンの内部形式およびサイズを含め、MySQL がサポートするデータ型ごとのストレージ要件に関するガイドラインおよび情報について説明します。情報はカテゴリまたはストレージエンジンごとに示します。

テーブルの内部表現の最大行サイズは 65,535 バイトであり、ストレージエンジンがこれ以上のサイズの行をサポートできる場合でもこのサイズになります。BLOB または TEXT カラムはこのサイズに 9 から 12 バイトしか関与しないので、これらのカラムはこのサイズに含まれません。BLOB および TEXT データについての情報は、行バッファーとは異なるメモリー領域に内部的に格納されます。それぞれのストレージエンジンは、対応する型の処理に使用する方法に従って異なる方法で、このデータの割り当ておよびストレージを扱います。詳細は、第15章「代替ストレージエンジン」およびセクションD.10.4「テーブルカラム数と行サイズの制限」を参照してください。

InnoDB テーブルのストレージ要件

InnoDB テーブルのストレージ要件の詳細は、セクション14.2.13.7「物理的な行構造」を参照してください。

NDBCLUSTER テーブルのストレージ要件

> **重要**
>
> NDB テーブルは、4 バイトアライメントを使用します。すべての NDB データストレージは、4 バイトの倍数で行われます。したがって、通常であれば 15 バイトを使用するカラム値は、NDB テーブルでは 16 バイトを必要とします。たとえば、NDB テーブルでは、TINYINT、SMALLINT、MEDIUMINT、および INTEGER (INT) カラム型はそれぞれ、アライメント係数により、レコードあたり 4 バイトのストレージが必要になります。
>
> 各 BIT(M) カラムは M ビットのストレージ領域を使用します。各 BIT カラムは 4 バイトアライメントが行われていませんが、NDB は、BIT カラムに必要な最初の 1 から 32 ビットに行あたり 4 バイト (32 ビット) を、33 から 64 ビットに別の 4 ビットを、というように予約します。
>
> NULL 自体はストレージ領域を必要としませんが、NDB は、テーブル定義に NULL として定義されたカラム (最大 32 の NULL カラム) が含まれる場合、行あたり 4 バイトを予約します。(MySQL Cluster テーブルが 32 以上の NULL カラムから 64 の NULL カラムで定義されている場合、行あたり 8 バイトが予約されます。)

NDB ストレージエンジンを使用するすべてのテーブルで主キーが必要になります。主キーを定義していない場合、「非表示」の主キーが NDB によって作成されます。この非表示の主キーはテーブルレコードあたり 31 から 35 バイトを消費します。

ndb_size.pl Perl スクリプトを使用して、NDB ストレージ要件を評価します。これは、(MySQL Cluster ではなく) 現在の MySQL データベースに接続し、そのデータベースが NDB ストレージエンジンを使用した場合にどれだけの領域を必要とするかについてレポート作成します。詳細は、セクション18.4.25「ndb_size.pl — NDBCLUSTER サイズ要件エスティメータ」を参照してください。

数値型のストレージ要件

データ型	必要なストレージ
TINYINT	1 バイト
SMALLINT	2 バイト
MEDIUMINT	3 バイト
INT、INTEGER	4 バイト
BIGINT	8 バイト
FLOAT(p)	0 <= p <= 24 の場合は 4 バイト、25 <= p <= 53 の場合は 8 バイト
FLOAT	4 バイト
DOUBLE [PRECISION]、REAL	8 バイト
DECIMAL(M,D)、NUMERIC(M,D)	変動; 次の説明を参照
BIT(M)	約 (M+7)/8 バイト

DECIMAL (および NUMERIC) カラムの値は、9 桁の 10 進数 (10 進法) を 4 バイトにパックするバイナリ形式を使用して表現されます。各値の整数部と小数部のストレージは、個別に決定されます。9 桁の倍ごとに 4 バイトが必要であり、「余りの」桁には 4 バイトのうちの一部が必要です。余りの桁に必要なストレージ要件を次の表に示します。

余りの桁	バイト数
0	0
1	1
2	1
3	2
4	2
5	3
6	3
7	4
8	4

日付と時間型のストレージ要件

TIME、DATETIME、および TIMESTAMP カラムの場合、MySQL 5.6.4 よりも前に作成されたテーブルに必要なストレージは、5.6.4 以降で作成されたテーブルとは異なります。これは、5.6.4 で、0 から 3 バイトを必要とする小数部をこれらの型が持つことを許可するように変更されたためです。

データ型	MySQL 5.6.4 より前で必要なストレージ	MySQL 5.6.4 以降で必要なストレージ
YEAR	1 バイト	1 バイト
DATE	3 バイト	3 バイト
TIME	3 バイト	3 バイト + 小数秒ストレージ
DATETIME	8 バイト	5 バイト + 小数秒ストレージ
TIMESTAMP	4 バイト	4 バイト + 小数秒ストレージ

MySQL 5.6.4 以降、YEAR および DATE のストレージは変更ありません。ただし、TIME、DATETIME、および TIMESTAMP は異なって表現されます。DATETIME はより効率的にパックされ、非小数部に必要なバイト数は 8 バイトではなく 5 バイトであり、3 つの部分すべてに、格納値の小数秒精度に応じて 0 から 3 バイトが必要な小数部があります。

小数秒精度	必要なストレージ
0	0 バイト
1、2	1 バイト
3、4	2 バイト

小数秒精度	必要なストレージ
5、6	3 バイト

たとえば、TIME(0)、TIME(2)、TIME(4)、および TIME(6) はそれぞれ 3、4、5、6 バイトを使用します。TIME と TIME(0) は同等で、必要なストレージは同じです。

時間値の内部表現の詳細は、「MySQL Internals: Important Algorithms and Structures」を参照してください。

文字列型のストレージ要件

次の表では、M は宣言されたカラムの長さを、非バイナリ文字列型の場合は文字数で、バイナリ文字列型の場合はバイト数で表します。L は指定された文字列値の実際の長さをバイト数で表します。

データ型	必要なストレージ
CHAR(M)	M × w バイト、0 <= M <= 255、ここで w は、文字セット内の最大長の文字に必要なバイト数です。InnoDB テーブルの CHAR データ型のストレージ要件の詳細は、セクション14.2.13.7「物理的な行構造」を参照してください。
BINARY(M)	M バイト、0 <= M <= 255
VARCHAR(M)、VARBINARY(M)	カラム値が 0 から 255 バイトを必要とする場合は、L + 1 バイト、値が 255 バイト以上を必要とする可能性のある場合は、L + 2 バイト
TINYBLOB、TINYTEXT	L + 1 バイト、ここで $L < 2^8$
BLOB、TEXT	L + 2 バイト、ここで $L < 2^{16}$
MEDIUMBLOB、MEDIUMTEXT	L + 3 バイト、ここで $L < 2^{24}$
LONGBLOB、LONGTEXT	L + 4 バイト、ここで $L < 2^{32}$
ENUM('value1','value2',...)	列挙値の数 (最大 65,535 個の値) により 1 または 2 バイト
SET('value1','value2',...)	セットメンバーの数 (最大 64 メンバー) により、1、2、3、4、または 8 バイト

可変長の文字列型は、長さプリフィクスが付いたデータを使用して格納されます。長さプリフィクスにはデータ型に応じて 1 から 4 バイトが必要で、プリフィクスの値は L (文字列のバイト長) です。たとえば、MEDIUMTEXT 値のストレージには、値を格納するための L バイトに加えて、値の長さを格納するための 3 バイトが必要です。

特定の CHAR、VARCHAR、または TEXT カラム値の格納に使用されるバイト数を計算するには、そのカラムに使用される文字セットと、値にマルチバイト文字が含まれるかどうかを考慮する必要があります。特に、utf8 (または utf8mb4) Unicode 文字セットを使用する場合、すべての文字セットが同じバイト数を使用するわけではなく、文字あたり最大 3 (4) バイトを必要とするわけではないことに注意する必要があります。utf8 または utf8mb4 文字の異なるカテゴリに使用されるストレージの詳細は、セクション10.1.10「Unicode のサポート」を参照してください。

VARCHAR、VARBINARY、および BLOB と TEXT 型は可変長型です。それぞれのストレージ要件は次の要因によって決まります。

- カラム値の実際の長さ
- カラムの可能な最大の長さ
- カラムに使用される文字セット。一部の文字セットにはマルチバイト文字が含まれるため。

たとえば、VARCHAR(255) カラムには最大 255 文字の長さの文字列を格納できます。そのカラムが latin1 文字セット (1 文字あたり 1 バイト) を使用すると仮定すると、実際に必要なストレージは文字列の長さ (L) に、文字列の長さを記録するための 1 バイトを加えた大きさとなります。文字列 'abcd' の場合、L は 4 で、ストレージ要件は 5 バイトになります。同じカラムが代わりにダブルバイト文字セット ucs2 を使用するように宣言されている場合、ストレージ要件は 10 バイトになります。'abcd' の長さは 8 バイトで、カラムの最大長が 255 よりも大きい (最大 510 バイト) ため、長さを格納するために 2 バイト必要になります。

VARCHAR または VARBINARY カラムに格納できる有効な最大バイト数は最大行サイズ (65,535 バイト、すべてのカラムで共有される) によって決まります。複数バイト文字を格納する VARCHAR カラムの場合、文字の有効な最大数は少なくなります。たとえば、utf8 の文字は 1 文字につき最大 3 バイトを必要とする場合があるた

め、utf8 の文字セットを使用する VARCHAR カラムは、最大 21,844 文字になるように宣言できます。セクションD.10.4「テーブルカラム数と行サイズの制限」を参照してください。

NDB ストレージエンジンは可変幅カラムをサポートします。これは、MySQL Cluster テーブル内の VARCHAR カラムは、このような値に対して 4 バイトアライメントが行われる点を除き、ほかのストレージエンジンと同じ容量のストレージを必要とするということを意味します。したがって、latin1 文字セットを使用して VARCHAR(50) カラムに格納された文字列 'abcd' は、(MyISAM テーブル内の同じカラム値に対する 5 バイトではなく) 8 バイトを必要とします。

TEXT と BLOB カラムは、NDB ストレージエンジンでは異なって実装されます。ここでは、TEXT カラム内の各行は 2 つの別々の部分から構成されています。そのうちの 1 つは固定サイズ (256 バイト) で、実際に元のテーブルに格納されます。もう 1 つは 256 バイトを超えるデータで構成され、非表示のテーブルに格納されます。2 番目のテーブルの行の長さは常に 2,000 バイトです。これは、size <= 256 (ここで size は行のサイズを表します) の場合、TEXT カラムのサイズが 256 であり、それ以外の場合はサイズが 256 + size + (2000 − (size − 256) % 2000) であることを意味します。

ENUM オブジェクトのサイズは異なる列挙値の数によって決まります。最大 255 の値を持つ列挙に 1 バイトが使用されます。256 から 65,535 の値を持つ列挙に 2 バイトが使用されます。セクション11.4.4「ENUM 型」を参照してください。

SET オブジェクトのサイズは異なるセットメンバーの数によって決まります。セットサイズが N である場合、オブジェクトは 1、2、3、4、または 8 バイトに丸められた (N+7)/8 バイトを占めます。SET は最大 64 メンバーを持つことができます。セクション11.4.5「SET 型」を参照してください。

11.8 カラムに適した型の選択

最適なストレージのために、毎回もっとも正確な型を使用するよう試みる必要があります。たとえば、整数カラムを 1 から 99999 の範囲の値に使用する場合、MEDIUMINT UNSIGNED が最適な型になります。必要なすべての値を表す型の中で、これが、使用するストレージの容量がもっとも少ない型になります。

DECIMAL カラムを使用した基本的なすべての計算 (+、-、*、および /) は、65 桁 (10 進法) の精度で行われます。セクション11.1.1「数値型の概要」を参照してください。

精度がそれほど重要でない場合や、スピードが最優先事項である場合は、DOUBLE 型で十分と考えられます。精度を高めるために、BIGINT に格納されている固定小数点型にいつでも変換できます。これにより、64 ビット整数のすべての計算を行い、続いて必要に応じて結果を浮動小数点値に戻すことができます。

PROCEDURE ANALYSE を使用すると、最適なカラムデータ型の選択に役立つ情報を入手できます。詳細は、セクション8.4.2.4「PROCEDURE ANALYSE の使用」を参照してください。

11.9 その他のデータベースエンジンのデータ型の使用

ほかのベンダーからの SQL 実装用に作成されたコードを使用しやすくするために、次の表に示すように、MySQL はデータ型をマップします。これらのマッピングにより、ほかのデータベースシステムから MySQL へのテーブル定義の取り込みが簡単に行えるようになります。

その他のベンダーの型	MySQL の型
BOOL	TINYINT
BOOLEAN	TINYINT
CHARACTER VARYING(M)	VARCHAR(M)
FIXED	DECIMAL
FLOAT4	FLOAT
FLOAT8	DOUBLE
INT1	TINYINT
INT2	SMALLINT
INT3	MEDIUMINT
INT4	INT
INT8	BIGINT
LONG VARBINARY	MEDIUMBLOB

その他のベンダーの型	MySQL の型
LONG VARCHAR	MEDIUMTEXT
LONG	MEDIUMTEXT
MIDDLEINT	MEDIUMINT
NUMERIC	DECIMAL

データ型のマッピングはテーブル作成時に行われ、作成後に元の型の仕様は破棄されます。ほかのベンダーで使用されている型でテーブルを作成したあとで、DESCRIBE tbl_name ステートメントを発行した場合、MySQL は、その型と同等の MySQL の型を使用したテーブル構造をレポートします。例:

```
mysql> CREATE TABLE t (a BOOL, b FLOAT8, c LONG VARCHAR, d NUMERIC);
Query OK, 0 rows affected (0.00 sec)

mysql> DESCRIBE t;
+-------+---------------+------+-----+---------+-------+
| Field | Type          | Null | Key | Default | Extra |
+-------+---------------+------+-----+---------+-------+
| a     | tinyint(1)    | YES  |     | NULL    |       |
| b     | double        | YES  |     | NULL    |       |
| c     | mediumtext    | YES  |     | NULL    |       |
| d     | decimal(10,0) | YES  |     | NULL    |       |
+-------+---------------+------+-----+---------+-------+
4 rows in set (0.01 sec)
```

第 12 章 関数と演算子

目次

12.1 関数と演算子のリファレンス 1062
12.2 式評価での型変換 1071
12.3 演算子 1073
12.3.1 演算子の優先順位 1074
12.3.2 比較関数と演算子 1074
12.3.3 論理演算子 1079
12.3.4 割り当て演算子 1080
12.4 制御フロー関数 1081
12.5 文字列関数 1083
12.5.1 文字列比較関数 1096
12.5.2 正規表現 1099
12.6 数値関数と演算子 1103
12.6.1 算術演算子 1104
12.6.2 数学関数 1106
12.7 日付および時間関数 1113
12.8 MySQL で使用されるカレンダー 1131
12.9 全文検索関数 1132
12.9.1 自然言語全文検索 1133
12.9.2 ブール全文検索 1135
12.9.3 クエリー拡張を使用した全文検索 1140
12.9.4 全文ストップワード 1140
12.9.5 全文制限 1145
12.9.6 MySQL の全文検索の微調整 1146
12.9.7 全文インデックス作成用の照合順序の追加 1148
12.10 キャスト関数と演算子 1149
12.11 XML 関数 1152
12.12 ビット関数 1161
12.13 暗号化関数と圧縮関数 1163
12.14 情報関数 1170
12.15 空間分析関数 1179
12.15.1 空間関数のリファレンス 1179
12.15.2 空間関数による引数処理 1181
12.15.3 WKT 値から幾何値を作成する関数 1181
12.15.4 WKB 値から幾何値を作成する関数 1182
12.15.5 幾何値を作成する MySQL 固有の関数 1183
12.15.6 幾何形式変換関数 1183
12.15.7 幾何プロパティー関数 1184
12.15.8 空間演算子関数 1189
12.15.9 幾何オブジェクト間の空間関係をテストする関数 1190
12.16 グローバルトランザクション ID とともに使用される関数 1193
12.17 MySQL Enterprise Encryption の関数 1195
12.17.1 Enterprise Encryption のインストール 1195
12.17.2 Enterprise Encryption の使用法と例 1195
12.17.3 Enterprise Encryption 関数のリファレンス 1197
12.17.4 Enterprise Encryption 関数の説明 1197
12.18 その他の関数 1200
12.19 GROUP BY 句で使用される関数と修飾子 1207
12.19.1 GROUP BY (集約) 関数 1207
12.19.2 GROUP BY 修飾子 1210
12.19.3 MySQL での GROUP BY の処理 1213
12.20 高精度計算 1214
12.20.1 数値の型 1214
12.20.2 DECIMAL データ型の特性 1215
12.20.3 式の処理 1216
12.20.4 丸め動作 1217
12.20.5 高精度計算の例 1217

SQL ステートメントのいくつかのポイント (SELECT ステートメントの ORDER BY または HAVING 句、SELECT、DELETE、または UPDATE ステートメントの WHERE 句、SET ステートメントなど) では、式を使用できます。式は、リテラル値、カラム値、NULL、組み込み関数、ストアドファンクション、ユーザー定義関数、および演算子を使用して作成できます。この章では、MySQL で式を記述する際に許可されている関数および演算子について説明します。ストアドファンクションおよびユーザー定義関数を作成する手順については、セクション20.2「ストアドルーチン (プロシージャーと関数) の使用」およびセクション24.3「MySQL への新しい関数の追加」で説明されています。各種の関数への参照をサーバーが解釈する方法を記述したルールについては、セクション9.2.4「関数名の構文解析と解決」を参照してください。

NULL を含む式では、その関数または演算子に関するドキュメントで明記されていないかぎり、常に NULL 値が生成されます。

> **注記**
>
> デフォルトでは、関数名とそれに続く丸括弧の間には空白を入れることはできません。これは、MySQL パーサーが、関数呼び出しと、偶然に関数と同じ名前を持つテーブルまたはカラムへの参照を区別するのに役立ちます。ただし、関数の引数の前後にスペースを入れることは許可されています。

最初に --sql-mode=IGNORE_SPACE オプションを付けると、関数名のあとの空白文字を受け入れるように MySQL サーバーに指示できます。(セクション5.1.7「サーバー SQL モード」を参照してください。)各クライアントプログラムで mysql_real_connect() に CLIENT_IGNORE_SPACE オプションを使用すると、この動作をリクエストできます。どちらの場合でも、すべての関数名は予約語になります。

簡略化のため、この章で示すほとんどの例では、mysql プログラムからの出力が省略形で表示されています。例は次の書式で表示されません。

```
mysql> SELECT MOD(29,9);
+-----------+
| mod(29,9) |
+-----------+
|         2 |
+-----------+
1 rows in set (0.00 sec)
```

代わりに、次の書式が使用されます。

```
mysql> SELECT MOD(29,9);
       -> 2
```

12.1 関数と演算子のリファレンス

表 12.1 関数/演算子

名前	説明
ABS()	絶対値を返します
ACOS()	アークコサインを返します
ADDDATE()	日付値に時間値 (間隔) を加算します
ADDTIME()	時間を加算します
AES_DECRYPT()	AES を使用して復号化します
AES_ENCRYPT()	AES を使用して暗号化します
AND, &&	論理 AND
Area()	Polygon または MultiPolygon 領域を返します
AsBinary(), AsWKB()	内部幾何形式から WKB に変換します
ASCII()	左端の文字の数値を返します
ASIN()	アークサインを返します
=	(SET ステートメントの一部として、または UPDATE ステートメントの SET 句の一部として) 値を割り当てます
:=	値を割り当てます
AsText(), AsWKT()	内部幾何形式から WKT に変換します
ASYMMETRIC_DECRYPT() (導入 5.6.21)	秘密鍵または公開鍵を使用して暗号文を復号化します
ASYMMETRIC_DERIVE() (導入 5.6.21)	非対称鍵から対称鍵を導出します

名前	説明
ASYMMETRIC_ENCRYPT() (導入 5.6.21)	秘密鍵または公開鍵を使用してプレーンテキストを暗号化します
ASYMMETRIC_SIGN() (導入 5.6.21)	ダイジェストから署名を生成します
ASYMMETRIC_VERIFY() (導入 5.6.21)	署名がダイジェストと一致することを確認します
ATAN()	アークタンジェントを返します
ATAN2(), ATAN()	2 つの引数のアークタンジェントを返します
AVG()	引数の平均値を返します
BENCHMARK()	式を繰り返し実行します
BETWEEN ... AND ...	値が値の範囲内に含まれているかどうかを確認します
BIN()	数値のバイナリ表現を含む文字列を返します
BINARY	文字列をバイナリ文字列にキャストします
BIT_AND()	ビット単位の And を返します
BIT_COUNT()	設定されているビット数を返します
BIT_LENGTH()	ビット単位で引数の長さを返します
BIT_OR()	ビット単位の OR を返します
BIT_XOR()	ビット単位の XOR を返します
&	ビット単位の AND
~	ビットを反転します
|	ビット単位の OR
^	ビット単位の XOR
Buffer() (導入 5.6.1)	幾何図形から指定された距離内にある点の幾何図形を返します
CASE	CASE 演算子
CAST()	値を特定の型としてキャストします
CEIL()	引数以上のもっとも小さな整数値を返します
CEILING()	引数以上のもっとも小さな整数値を返します
Centroid()	重心を Point として返します
CHAR()	渡された各整数の文字を返します
CHAR_LENGTH()	引数の文字数を返します
CHARACTER_LENGTH()	CHAR_LENGTH() のシノニムです
CHARSET()	引数の文字セットを返します
COALESCE()	NULL 以外の最初の引数を返します
COERCIBILITY()	文字列引数の照合順序強制性値を返します
COLLATION()	文字列引数の照合順序を返します
COMPRESS()	バイナリ文字列として結果を返します
CONCAT()	連結された文字列を返します
CONCAT_WS()	連結されたものをセパレータ付きで返します
CONNECTION_ID()	接続のための接続 ID (スレッド ID) を返します
Contains()	ある幾何図形に別の幾何図形が含まれているかどうか
CONV()	数値を異なる基数間で変換します
CONVERT()	値を特定の型としてキャストします
CONVERT_TZ()	あるタイムゾーンから別のタイムゾーンに変換します
COS()	コサインを返します
COT()	コタンジェントを返します
COUNT()	返された行数のカウントを返します
COUNT(DISTINCT)	異なる値のカウントを返します

名前	説明
CRC32()	巡回冗長検査値を計算します
CREATE_ASYMMETRIC_PRIV_KEY() (導入 5.6.21)	秘密鍵を作成します
CREATE_ASYMMETRIC_PUB_KEY() (導入 5.6.21)	公開鍵を作成します
CREATE_DH_PARAMETERS() (導入 5.6.21)	共有 DH シークレットを生成します
CREATE_DIGEST() (導入 5.6.21)	文字列からダイジェストを生成します
Crosses()	ある幾何図形が別の幾何図形と交差しているかどうか
CURDATE()	現在の日付を返します
CURRENT_DATE(), CURRENT_DATE	CURDATE() のシノニムです
CURRENT_TIME(), CURRENT_TIME	CURTIME() のシノニムです
CURRENT_TIMESTAMP(), CURRENT_TIMESTAMP	NOW() のシノニムです
CURRENT_USER(), CURRENT_USER	認証済みユーザー名とホスト名
CURTIME()	現在の時間を返します
DATABASE()	デフォルト (現在) のデータベース名を返します
DATE()	日付または日付時間式の日付部分を抽出します
DATE_ADD()	日付値に時間値 (間隔) を加算します
DATE_FORMAT()	日付を指定された書式に設定します
DATE_SUB()	日付から時間値 (間隔) を引きます
DATEDIFF()	2 つの日付の差を求めます
DAY()	DAYOFMONTH() のシノニムです
DAYNAME()	曜日の名前を返します
DAYOFMONTH()	月の日を返します (0 - 31)
DAYOFWEEK()	引数の曜日インデックスを返します
DAYOFYEAR()	年の日を返します (1 - 366)
DECODE()	ENCODE() を使用して暗号化された文字列をデコードします
DEFAULT()	テーブルカラムのデフォルト値を返します
DEGREES()	ラジアンを角度に変換します
DES_DECRYPT()	文字列を復号化します
DES_ENCRYPT()	文字列を暗号化します
Dimension()	幾何図形の次元
Disjoint()	ある幾何図形が別の幾何図形から切り離されているかどうか
DIV	整数除算
/	除算演算子
ELT()	インデックス番号位置の文字列を返します
ENCODE()	文字列をエンコードします
ENCRYPT()	文字列を暗号化します
EndPoint()	LineString の終点
Envelope()	幾何図形の MBR を返します
=	等価 (等しい) 演算子
<=>	NULL 安全等価演算子
Equals()	ある幾何図形が別の幾何図形に等しいかどうか
EXP()	累乗します

名前	説明
EXPORT_SET()	値 bits 内の各ビットが設定されている場合は on 文字列を取得し、各ビットが設定されていない場合には off 文字列を取得するように、文字列を返します
ExteriorRing()	Polygon の外側のリングを返します
EXTRACT()	日付の一部を抽出します
ExtractValue()	XPath 表記法を使用して、XML 文字列から値を抽出します
FIELD()	後続の引数内で第 1 引数のインデックス (位置) を返します
FIND_IN_SET()	第 2 引数内で第 1 引数のインデックス位置を返します
FLOOR()	引数以下のもっとも大きな整数値を返します
FORMAT()	指定された小数点以下桁数に書式設定された数値を返します
FOUND_ROWS()	LIMIT 句付き SELECT で、LIMIT 句がない場合に戻される可能性がある行の数です
FROM_BASE64() (導入 5.6.1)	base 64 文字列にデコードして結果を返します
FROM_DAYS()	日数を日付に変換します
FROM_UNIXTIME()	UNIX タイムスタンプを日付として書式設定します
GeomCollFromText(), GeometryCollectionFromText()	WKT からジオメトリコレクションを返します
GeomCollFromWKB(), GeometryCollectionFromWKB()	WKB からジオメトリコレクションを返します
GeometryCollection()	幾何図形からジオメトリコレクションを構築します
GeometryN()	ジオメトリコレクションから N 番目の幾何図形を返します
GeometryType()	幾何型の名前を返します
GeomFromText(), GeometryFromText()	WKT から幾何図形を返します
GeomFromWKB()	WKB から幾何図形を返します
GET_FORMAT()	日付書式文字列を返します
GET_LOCK()	名前付きロックを取得します
GLength()	LineString の長さを返します
>	右不等 (より多い) 演算子
>=	以上 (より多いか等しい) 演算子
GREATEST()	最大の引数を返します
GROUP_CONCAT()	連結された文字列を返します
GTID_SUBSET() (導入 5.6.5)	サブセット内のすべての GTID がセット内にもある場合は、true を返します。そうでない場合は、false を返します。
GTID_SUBTRACT() (導入 5.6.5)	セット内の GTID のうち、サブセット内にないものをすべてを返します。
HEX()	10 進値または文字列値の 16 進表現を返します
HOUR()	時を抽出します
IF()	If/else 構文
IFNULL()	Null if/else 構文
IN()	ある値が値セット内に含まれているかどうかを確認します
INET_ATON()	IP アドレスの数値を返します
INET_NTOA()	数値から IP アドレスを返します
INET6_ATON() (導入 5.6.3)	IPv6 アドレスの数値を返します
INET6_NTOA() (導入 5.6.3)	数値から IPv6 アドレスを返します
INSERT()	部分文字列を、指定された位置に指定された文字数だけ挿入します

名前	説明
INSTR()	部分文字列が最初に出現する位置のインデックスを返します
InteriorRingN()	Polygon の N 番目の内側のリングを返します
Intersects()	ある幾何図形が別の幾何図形と交差しているかどうか
INTERVAL()	第 1 引数より小さい引数のインデックスを返します
IS	ブーリアンに対して値をテストします
IS_FREE_LOCK()	名前付きロックが解放されているかどうかを確認します
IS_IPV4() (導入 5.6.3)	引数が IPv4 アドレスの場合、true を返します
IS_IPV4_COMPAT() (導入 5.6.3)	引数が IPv4 互換アドレスの場合、true を返します
IS_IPV4_MAPPED() (導入 5.6.3)	引数が IPv4 マップアドレスの場合、true を返します
IS_IPV6() (導入 5.6.3)	引数が IPv6 アドレスの場合、true を返します
IS NOT	ブーリアンに対して値をテストします
IS NOT NULL	NOT NULL 値テスト
IS NULL	NULL 値テスト
IS_USED_LOCK()	名前付きロックが使用中かどうかを確認します。true の場合は接続識別子を返します。
IsClosed()	幾何図形が閉じていて単純かどうか
IsEmpty()	プレースホルダ関数
ISNULL()	引数が NULL かどうかをテストします
IsSimple()	幾何図形が単純かどうか
LAST_DAY	引数の月の最終日を返します
LAST_INSERT_ID()	前回の INSERT での AUTOINCREMENT カラムの値です
LCASE()	LOWER() のシノニムです
LEAST()	最小の引数を返します
LEFT()	左端から指定された数の文字を返します
<<	左シフト
LENGTH()	文字列の長さをバイト単位で返します
<	左不等 (より少ない) 演算子
<=	以下 (より少ないか等しい) 演算子
LIKE	単純なパターン一致
LineFromText()	WKT から LineString を構築します
LineFromWKB(), LineStringFromWKB()	WKB から LineString を構築します
LineString()	Point 値から LineString を構築します
LN()	引数の自然対数を返します
LOAD_FILE()	指定されたファイルをロードします
LOCALTIME(), LOCALTIME	NOW() のシノニムです
LOCALTIMESTAMP, LOCALTIMESTAMP()	NOW() のシノニムです
LOCATE()	部分文字列が最初に出現する位置を返します
LOG()	最初の引数の自然対数を返します
LOG10()	引数の底 10 の対数を返します
LOG2()	引数の底 2 の対数を返します
LOWER()	引数を小文字で返します
LPAD()	指定された文字列で左からパディングした文字列引数を返します
LTRIM()	先頭の空白を削除します
MAKE_SET()	bits セット内の対応するビットを持つ、カンマ区切り文字列のセットを返します

名前	説明
MAKEDATE()	年と年間通算日から日付を作成します
MAKETIME()	時、分、秒から時間を作成します
MASTER_POS_WAIT()	スレーブが指定された位置まですべての更新を読み取って適用するまで、ブロックします
MATCH	全文検索を実行します
MAX()	最大値を返します
MBRContains()	ある幾何図形の MBR に、別の幾何図形の MBR が含まれているかどうか
MBRDisjoint()	2 つの幾何図形の MBR が切り離されているかどうか
MBREqual()	2 つの幾何図形の MBR が等しいかどうか
MBRIntersects()	2 つの幾何図形の MBR が交差しているかどうか
MBROverlaps()	2 つの幾何図形の MBR がオーバーラップしているかどうか
MBRTouches()	2 つの幾何図形の MBR が接しているかどうか
MBRWithin()	ある幾何図形の MBR が、別の幾何図形の MBR の内部にあるかどうか
MD5()	MD5 チェックサムを計算します
MICROSECOND()	引数からマイクロ秒を返します
MID()	指定された位置から始まる部分文字列を返します
MIN()	最小値を返します
-	減算演算子
MINUTE()	引数から分を返します
MLineFromText(), MultiLineStringFromText()	WKT から MultiLineString を構築します
MLineFromWKB(), MultiLineStringFromWKB()	WKB から MultiLineString を構築します
MOD()	余りを返します
%, MOD	モジュロ演算子
MONTH()	渡された日付から月を返します
MONTHNAME()	月の名前を返します
MPointFromText(), MultiPointFromText()	WKT から MultiPoint を構築します
MPointFromWKB(), MultiPointFromWKB()	WKB から MultiPoint を構築します
MPolyFromText(), MultiPolygonFromText()	WKT から MultiPolygon を構築します
MPolyFromWKB(), MultiPolygonFromWKB()	WKB から MultiPolygon を構築します
MultiLineString()	LineString 値から MultiLineString を構築します
MultiPoint()	Point 値から MultiPoint を構築します
MultiPolygon()	Polygon 値から MultiPolygon を構築します
NAME_CONST()	指定された名前がカラムに付けられます
NOT, !	値を否定します
NOT BETWEEN ... AND ...	値が値の範囲内に含まれていないかどうかを確認します
!=, <>	不等価 (等しくない) 演算子
NOT IN()	値が値セット内に含まれていないかどうかを確認します
NOT LIKE	単純なパターン一致の否定
NOT REGEXP	REGEXP の否定
NOW()	現在の日付と時間を返します
NULLIF()	expr1 = expr2 の場合に NULL を返します
NumGeometries()	ジオメトリコレクション内の幾何図形数を返します

名前	説明
NumInteriorRings()	Polygon 内の内側のリング数を返します
NumPoints()	LineString 内の Point の数を返します
OCT()	数値の 8 進数表現を含む文字列を返します
OCTET_LENGTH()	LENGTH() のシノニムです
OLD_PASSWORD() (非推奨 5.6.5)	4.1 より前の PASSWORD 実装の値を返します
||, OR	論理 OR
ORD()	引数の左端の文字の文字コードを返します
Overlaps()	ある幾何図形が別の幾何図形とオーバーラップしているかどうか
PASSWORD()	パスワード文字列を計算して返します
PERIOD_ADD()	年月に期間を加算します
PERIOD_DIFF()	期間内の月数を返します
PI()	pi の値を返します
+	加算演算子
Point()	座標から Point を構築します
PointFromText()	WKT から Point を構築します
PointFromWKB()	WKB から Point を構築します
PointN()	LineString から N 番目の Point を返します
PolyFromText(), PolygonFromText()	WKT から Polygon を構築します
PolyFromWKB(), PolygonFromWKB()	WKB から Polygon を構築します
Polygon()	LineString 引数から Polygon を構築します
POSITION()	LOCATE() のシノニムです
POW()	指定した指数で累乗された引数を返します
POWER()	指定した指数で累乗された引数を返します
PROCEDURE ANALYSE()	クエリーの結果を解析します
QUARTER()	日付引数から四半期を返します
QUOTE()	SQL ステートメント内で使用するために引数をエスケープします
RADIANS()	ラジアンに変換された引数を返します
RAND()	ランダムな浮動小数点値を返します
RANDOM_BYTES() (導入 5.6.17)	ランダムなバイトベクトルを返します
REGEXP	正規表現を使用したパターン一致
RELEASE_LOCK()	名前付きロックを解放します
REPEAT()	文字列を指定された回数だけ繰り返します
REPLACE()	指定された文字列の出現箇所を置き換えます
REVERSE()	文字列内の文字を逆順に並べ替えます
RIGHT()	右端から指定された数の文字を返します
>>	右シフト
RLIKE	REGEXP のシノニムです
ROUND()	引数を丸めます
ROW_COUNT()	更新された行数
RPAD()	指定された回数だけ文字列を追加します
RTRIM()	末尾の空白を削除します
SCHEMA()	DATABASE() のシノニムです
SEC_TO_TIME()	秒を「HH:MM:SS」形式に変換します
SECOND()	秒 (0-59) を返します

名前	説明
SESSION_USER()	USER() のシノニムです
SHA1(), SHA()	SHA-1 160 ビットチェックサムを計算します
SHA2()	SHA-2 チェックサムを計算します
SIGN()	引数の符号を返します
SIN()	引数のサインを返します
SLEEP()	ある秒数間スリープ状態にします
SOUNDEX()	soundex 文字列を返します
SOUNDS LIKE	音声を比較します
SPACE()	指定された数の空白で構成される文字列を返します
SQL_THREAD_WAIT_AFTER_GTIDS() (導入 5.6.5, 非推奨 5.6.9)	廃止: WAIT_UNTIL_SQL_THREAD_AFTER_GTIDS() に置き換わりました
SQRT()	引数の平方根を返します
SRID()	幾何図形の空間参照システム ID を返します
ST_Area() (導入 5.6.1)	Polygon または MultiPolygon 領域を返します
ST_Centroid() (導入 5.6.1)	重心を Point として返します
ST_Contains() (導入 5.6.1)	ある幾何図形に別の幾何図形が含まれているかどうか
ST_Crosses() (導入 5.6.1)	ある幾何図形が別の幾何図形と交差しているかどうか
ST_Difference() (導入 5.6.1)	2 つの幾何図形の点集合の差集合を返します
ST_Disjoint() (導入 5.6.1)	ある幾何図形が別の幾何図形から切り離されているかどうか
ST_Distance() (導入 5.6.1)	ある幾何図形の別の幾何図形からの距離
ST_Envelope() (導入 5.6.1)	幾何図形の MBR を返します
ST_Equals() (導入 5.6.1)	ある幾何図形が別の幾何図形に等しいかどうか
ST_Intersection() (導入 5.6.1)	2 つの幾何図形の点集合の積集合を返します
ST_Intersects() (導入 5.6.1)	ある幾何図形が別の幾何図形と交差しているかどうか
ST_Overlaps() (導入 5.6.1)	ある幾何図形が別の幾何図形とオーバーラップしているかどうか
ST_SymDifference() (導入 5.6.1)	2 つの幾何図形の点集合の対称差を返します
ST_Touches() (導入 5.6.1)	ある幾何図形が別の幾何図形に接しているかどうか
ST_Union() (導入 5.6.1)	2 つの幾何図形の点集合の和集合を返します
ST_Within() (導入 5.6.1)	ある幾何図形が別の幾何図形の内部にあるかどうか
StartPoint()	LineString の始点
STD()	母標準偏差を返します
STDDEV()	母標準偏差を返します
STDDEV_POP()	母標準偏差を返します
STDDEV_SAMP()	標本標準偏差を返します
STR_TO_DATE()	文字列を日付に変換します
STRCMP()	2 つの文字列を比較します
SUBDATE()	3 つの引数で呼び出されるときは DATE_SUB() のシノニムです
SUBSTR()	指定された部分文字列を返します
SUBSTRING()	指定された部分文字列を返します
SUBSTRING_INDEX()	文字列から、区切り文字が指定された回数出現する前の部分文字列を返します
SUBTIME()	時間の差を求めます
SUM()	集計を返します
SYSDATE()	この関数が実行される時間を返します
SYSTEM_USER()	USER() のシノニムです

名前	説明
TAN()	引数のタンジェントを返します
TIME()	渡された式の時部分を抽出します
TIME_FORMAT()	時間として書式設定します
TIME_TO_SEC()	秒に変換された引数を返します
TIMEDIFF()	時間の差を求めます
*	乗算演算子
TIMESTAMP()	引数が 1 つの場合、この関数は日付または日付時間式を返します。引数が 2 つの場合、引数の合計を返します
TIMESTAMPADD()	日付時間式に間隔を加算します
TIMESTAMPDIFF()	日付時間式から間隔を減算します
TO_BASE64() (導入 5.6.1)	base 64 文字列に変換された引数を返します
TO_DAYS()	日に変換された日付引数を返します
TO_SECONDS()	0 年以降の秒数に変換された日付または日付時間引数を返します
Touches()	ある幾何図形が別の幾何図形に接しているかどうか
TRIM()	先頭と末尾にある空白を削除します
TRUNCATE()	指定された小数点以下の桁数に切り捨てます
UCASE()	UPPER() のシノニムです
-	引数の符号を変更します
UNCOMPRESS()	圧縮された文字列を圧縮解除します
UNCOMPRESSED_LENGTH()	圧縮前の文字列長を返します
UNHEX()	数値の 16 進数表現を含む文字列を返します
UNIX_TIMESTAMP()	UNIX タイムスタンプを返します
UpdateXML()	置換後 XML フラグメントを返します
UPPER()	大文字に変換します
USER()	ユーザー名と、クライアントによって提供されるホスト名です
UTC_DATE()	現在の UTC 日付を返します
UTC_TIME()	現在の UTC 時間を返します
UTC_TIMESTAMP()	現在の UTC 日付と時間を返します
UUID()	ユニバーサル固有識別子 (UUID) を返します
UUID_SHORT()	整数値のユニバーサル識別子を返します
VALIDATE_PASSWORD_STRENGTH() (導入 5.6.6)	パスワードの強度を判断します
VALUES()	INSERT で使用される値を定義します
VAR_POP()	母標準分散を返します
VAR_SAMP()	標本分散を返します
VARIANCE()	母標準分散を返します
VERSION()	MySQL サーバーのバージョンを示す文字列を返します
WAIT_UNTIL_SQL_THREAD_AFTER_GTIDS() (導入 5.6.9)	指定された GTID がスレーブで実行されるまで待ちます。
WEEK()	週番号を返します
WEEKDAY()	曜日インデックスを返します
WEEKOFYEAR()	日付の暦週を返します (0 - 53)
WEIGHT_STRING()	文字列の重み文字列を返します
Within()	ある幾何図形が別の幾何図形の内部にあるかどうか
X()	Point の X 座標を返します

名前	説明
XOR	論理 XOR
Y()	Point の Y 座標を返します
YEAR()	年を返します
YEARWEEK()	年と週を返します

12.2 式評価での型変換

演算子が別の型のオペランドとともに使用されると、オペランドの互換性を保つために型変換が発生します。一部の変換は暗黙的に発生します。たとえば、MySQL では必要に応じて数字が文字列 (またはその逆) に自動的に変換されます。

```
mysql> SELECT 1+'1';
    -> 2
mysql> SELECT CONCAT(2,' test');
    -> '2 test'
```

また、CAST() 関数を明示的に使用して、数字を文字列に変換することもできます。CONCAT() 関数では文字列の引数が要求されるため、使用すると暗黙的に変換が発生します。

```
mysql> SELECT 38.8, CAST(38.8 AS CHAR);
    -> 38.8, '38.8'
mysql> SELECT 38.8, CONCAT(38.8);
    -> 38.8, '38.8'
```

文字セットの数字から文字列への暗黙的な変換について、および CREATE TABLE ... SELECT ステートメントに適用される変更済みのルールについては、このセクションの後半を参照してください。

次のルールでは、比較演算の際にどのように変換が発生するのかについて説明します。

- NULL-safe <=> 等価比較演算子の場合を除いて、一方または両方の引数が NULL の場合は、比較の結果も NULL になります。NULL <=> NULL の場合は、結果が true になります。変換は必要ありません。
- 比較演算の両方の引数が文字列の場合は、文字列として比較されます。
- 両方の引数が整数の場合は、整数として比較されます。
- 16 進値が数字と比較されない場合は、バイナリ文字列として処理されます。
- 引数の一方が TIMESTAMP または DATETIME カラムで他方が定数の場合は、比較が実行される前に定数がタイムスタンプに変換されます。これは、ODBC により適合させるために実行されます。これは、IN() への引数には実行されません。念のため、比較を行う際は、常に完全な日付時間、日付、または時間文字列を使用してください。たとえば、日付または時間の値とともに BETWEEN を使用したときの結果を最適にするには、CAST() を使用して、明示的に値を目的のデータ型に変換します。

  テーブル (複数可) からの単一行のサブクエリーは、定数とみなされません。たとえば、サブクエリーで DATETIME 値と比較される整数が返される場合は、比較が 2 つの整数として実行されます。整数は時間値には変換されません。オペランドを DATETIME 値として比較するには、CAST() を使用して、明示的にサブクエリーの値を DATETIME に変換します。
- 引数のいずれかが 10 進値の場合、比較はその他の引数に依存します。その他の引数が 10 進値または整数値の場合、引数は 10 進値として比較され、その他の引数が浮動小数点値の場合、引数は浮動小数点値として比較されます。
- ほかのすべてのケースでは、引数は浮動小数点 (実) 数として比較されます。

別の時間型への値の変換については、セクション11.3.7「日付と時間型間での変換」を参照してください。

次の例は、比較演算での文字列から数字への変換を示しています。

```
mysql> SELECT 1 > '6x';
    -> 0
mysql> SELECT 7 > '6x';
    -> 1
mysql> SELECT 0 > 'x6';
    -> 0
mysql> SELECT 0 = 'x6';
    -> 1
```

文字列カラムと数字との比較では、MySQL はカラム上のインデックスを使用して、値をすばやく検索できません。str_col がインデックスの付いた文字列カラムである場合は、次のステートメントで検索を実行するときに、そのインデックスを使用できません。

```
SELECT * FROM tbl_name WHERE str_col=1;
```

その理由は、'1'、' 1'、'1a' のように、値 1 に変換できるさまざまな文字列があるためです。

このような数字は不正確であるため、浮動小数点数 (または浮動小数点数に変換される値) を使用する比較は概算になります。これにより、整合性のない結果が表示される可能性があります。

```
mysql> SELECT '18015376320243458' = 18015376320243458;
    -> 1
mysql> SELECT '18015376320243459' = 18015376320243459;
    -> 0
```

このような結果は、53 ビットの精度しか持たない浮動小数点数に値が変換され、丸めの対象になるために発生する可能性があります。

```
mysql> SELECT '18015376320243459'+0.0;
    -> 1.8015376320243e+16
```

さらに、文字列から浮動小数点への変換および整数から浮動小数点への変換は、必ずしも同様に発生するとはかぎりません。整数は、CPU によって浮動小数点に変換される可能性があります。一方、文字列は、浮動小数点の乗算を伴う演算で 1 桁ずつ変換されます。

表示される結果はシステムによって異なり、コンピュータのアーキテクチャーやコンパイラのバージョンなどの要因、または最適化レベルの影響を受ける可能性があります。このような問題を回避する方法の 1 つは、値が暗黙的に浮動小数点値に変換されないように、CAST() を使用することです。

```
mysql> SELECT CAST('18015376320243459' AS UNSIGNED) = 18015376320243459;
    -> 1
```

浮動小数点の比較についての詳細は、セクションB.5.5.8「浮動小数点値に関する問題」を参照してください。

MySQL 5.6 では、サーバーに dtoa が含まれています。これは、文字列または DECIMAL 値と近似値 (FLOAT/DOUBLE) の数値間の変換を改善するための基礎を提供する変換ライブラリです。

- Unix と Windows 間の変換の相違などが除去された、プラットフォーム間で整合性のある変換結果。
- 以前の結果に十分な精度がなかった場合 (IEEE の制限に近い値など) の正確な値の表示。
- 最大限の精度を持つ文字列書式への数字の変換。dtoa の精度は、常に標準の C ライブラリ関数の精度と同じか、それ以上です。

このライブラリで生成された変換は、dtoa 以外の結果とは異なる場合があるため、以前の結果に依存するアプリケーションとの互換性が保たれない可能性があります。たとえば、以前の変換からの特定の正確な結果に依存するアプリケーションでは、追加の精度に対応するように調整が必要となる場合があります。

dtoa ライブラリでは、次のプロパティーを使用した変換が提供されます。D は DECIMAL または文字列表現を含む値を表し、F はネイティブバイナリ (IEEE) 書式の浮動小数点数を表します。

- F -> D の変換は、最大限の精度で実行され、読み取り時に F が生成されるもっとも短い文字列として D が返され、IEEE で指定されているネイティブバイナリ形式でもっとも近い値に丸められます。
- D -> F の変換は、F が入力された 10 進文字列 D にもっとも近いネイティブバイナリの数字になるように実行されます。

F が -inf、+inf、または NaN の場合を除いて、これらのプロパティーは、F -> D -> F の変換が可逆であることを暗黙的に示しています。後者の値は、SQL 標準では FLOAT または DOUBLE の無効な値として定義されているため、サポートされていません。

D -> F -> D の変換では、D が 15 桁以下の精度を使用し、非正規値 (-inf、+inf、または NaN) でないことが可逆のための十分な条件となります。D の精度が 15 桁よりも大きい場合でも、変換が可逆であるケースもありますが、常に該当するとはかぎりません。

MySQL 5.6 では、暗黙的に数値または時間値を文字列に変換すると、character_set_connection および collation_connection システム変数で決定された文字セットおよび照合順序を含む値が生成されます。(一般に、これらの変数は SET NAMES を使用して設定されます。接続文字セットについては、セクション10.1.4「接続文字セットおよび照合順序」を参照してください。)

つまり、接続文字セットが binary に設定されている場合を除いて、このような変換では、(非バイナリ) 文字列 (CHAR、VARCHAR、または LONGTEXT 値) が生成されます。この場合、変換の結果はバイナリ文字列 (BINARY、VARBINARY、または LONGBLOB 値) になります。

整数式では、式の評価に関する上記の備考は、式の割り当てには多少異なる方法で適用されます。次のようなステートメントでの例を示します。

```
CREATE TABLE t SELECT integer_expr;
```

この場合、式の結果として生成されるカラム内のテーブルの型は、整数式の長さに応じて、INT または BIGINT になります。式の最大長が INT に収まらない場合は、代わりに BIGINT が使用されます。長さは、SELECT 結果セットメタデータの max_length 値から取得されます (セクション23.7.5「C API データ構造」を参照してください)。つまり、十分に長い式を使用することで、INT ではなく、BIGINT を強制的に適用できます。

```
CREATE TABLE t SELECT 00000000000000000000;
```

12.3 演算子

表 12.2 演算子

名前	説明
AND, &&	論理 AND
=	(SET ステートメントの一部として、または UPDATE ステートメントの SET 句の一部として) 値を割り当てます
:=	値を割り当てます
BETWEEN ... AND ...	値が値の範囲内に含まれているかどうかを確認します
BINARY	文字列をバイナリ文字列にキャストします
&	ビット単位の AND
~	ビットを反転します
|	ビット単位の OR
^	ビット単位の XOR
CASE	CASE 演算子
DIV	整数除算
/	除算演算子
=	等価 (等しい) 演算子
<=>	NULL 安全等価演算子
>	右不等 (より多い) 演算子
>=	以上 (より多いか等しい) 演算子
IS	ブーリアンに対して値をテストします
IS NOT	ブーリアンに対して値をテストします
IS NOT NULL	NOT NULL 値テスト
IS NULL	NULL 値テスト
<<	左シフト
<	左不等 (より少ない) 演算子
<=	以下 (より少ないか等しい) 演算子
LIKE	単純なパターン一致
-	減算演算子
%, MOD	モジュロ演算子
NOT, !	値を否定します
NOT BETWEEN ... AND ...	値が値の範囲内に含まれていないかどうかを確認します
!=, <>	不等価 (等しくない) 演算子
NOT LIKE	単純なパターン一致の否定
NOT REGEXP	REGEXP の否定

名前	説明
||, OR	論理 OR
+	加算演算子
REGEXP	正規表現を使用したパターン一致
>>	右シフト
RLIKE	REGEXP のシノニムです
SOUNDS LIKE	音声を比較します
*	乗算演算子
-	引数の符号を変更します
XOR	論理 XOR

12.3.1 演算子の優先順位

次のリストには、演算子の優先順位をもっとも高いものから順番に示しています。同じ行に並んで記載されている演算子は、優先順位が同じものです。

```
INTERVAL
BINARY, COLLATE
!
- (unary minus), ~ (unary bit inversion)
^
*, /, DIV, %, MOD
-, +
<<, >>
&
|
= (comparison), <=>, >=, >, <=, <, <>, !=, IS, LIKE, REGEXP, IN
BETWEEN, CASE, WHEN, THEN, ELSE
NOT
&&, AND
XOR
||, OR
= (assignment), :=
```

= の優先順位は、比較演算子 (=) として使用されるのか、割り当て演算子 (=) として使用されるのかによって異なります。比較演算子として使用される場合は、優先順位が <=>、>=、>、<=、<、<>、!=、IS、LIKE、REGEXP、および IN と同じです。割り当て演算子として使用される場合は、優先順位が := と同じです。セクション13.7.4「SET 構文」およびセクション9.4「ユーザー定義変数」では、適用される = の解釈が MySQL でどのように決定されるのかについて説明されています。

一部の演算子の意味は、SQL モードによって異なります。

- デフォルトでは、|| は論理 OR 演算子です。PIPES_AS_CONCAT が有効になっている場合は、|| は ^ と単項演算子間の優先順位を持つ文字列連結です。
- デフォルトでは、! は NOT よりも高い優先順位です。HIGH_NOT_PRECEDENCE が有効になっている場合は、! と NOT の優先順位は同じです。

セクション5.1.7「サーバー SQL モード」を参照してください。

演算子の優先順位によって、式の項の評価順序が決まります。この順序をオーバーライドし、明示的に項をグループ化するには、丸括弧を使用します。例:

```
mysql> SELECT 1+2*3;
    -> 7
mysql> SELECT (1+2)*3;
    -> 9
```

12.3.2 比較関数と演算子

表 12.3 比較演算子

名前	説明
BETWEEN ... AND ...	値が値の範囲内に含まれているかどうかを確認します
COALESCE()	NULL 以外の最初の引数を返します
=	等価 (等しい) 演算子

名前	説明
<=>	NULL 安全等価演算子
>	右不等 (より多い) 演算子
>=	以上 (より多いか等しい) 演算子
GREATEST()	最大の引数を返します
IN()	ある値が値セット内に含まれているかどうかを確認します
INTERVAL()	第 1 引数より小さい引数のインデックスを返します
IS	ブーリアンに対して値をテストします
IS NOT	ブーリアンに対して値をテストします
IS NOT NULL	NOT NULL 値テスト
IS NULL	NULL 値テスト
ISNULL()	引数が NULL かどうかをテストします
LEAST()	最小の引数を返します
<	左不等 (より少ない) 演算子
<=	以下 (より少ないか等しい) 演算子
LIKE	単純なパターン一致
NOT BETWEEN ... AND ...	値が値の範囲内に含まれていないかどうかを確認します
!=, <>	不等価 (等しくない) 演算子
NOT IN()	値が値セット内に含まれていないかどうかを確認します
NOT LIKE	単純なパターン一致の否定
STRCMP()	2 つの文字列を比較します

比較演算の結果は、1 (TRUE)、0 (FALSE)、または NULL の値になります。これらの演算は、数字と文字列の両方で機能します。必要に応じて、文字列は数字に、数字は文字列に自動的に変換されます。

次の関係比較演算子を使用すれば、スカラーオペランドだけでなく行オペランドも比較できます。

```
= > < >= <= <> !=
```

行比較の例については、セクション13.2.10.5「行サブクエリー」を参照してください。

このセクションで示す関数の一部では、1 (TRUE)、0 (FALSE)、または NULL 以外の値が返されます。たとえば、LEAST() や GREATEST() です。ただし、返される値は、セクション12.2「式評価での型変換」で説明したルールに従って実行された比較演算に基づきます。

CAST() 関数を使用すると、比較目的で値を特定の型に変換できます。CONVERT() を使用すると、文字列値を別の文字セットに変換できます。セクション12.10「キャスト関数と演算子」を参照してください。

デフォルトでは、文字列の比較では大文字と小文字が区別されず、現在の文字セットが使用されます。デフォルトは latin1 (cp1252 西ヨーロッパ言語) であり、英語でも正常に機能します。

- =

  等しい:

```
mysql> SELECT 1 = 0;
    -> 0
mysql> SELECT '0' = 0;
    -> 1
mysql> SELECT '0.0' = 0;
    -> 1
mysql> SELECT '0.01' = 0;
    -> 0
mysql> SELECT '.01' = 0.01;
    -> 1
```

- <=>

  NULL - 安全等価。この演算子では、= 演算子のように等価比較が実行されますが、両方のオペランドが NULL であれば、NULL でなく 1 が返され、一方のオペランドが NULL の場合は、NULL でなく 0 が返されます。

```
mysql> SELECT 1 <=> 1, NULL <=> NULL, 1 <=> NULL;
    -> 1, 1, 0
mysql> SELECT 1 = 1, NULL = NULL, 1 = NULL;
    -> 1, NULL, NULL
```

- **<>, !=**

  等しくない:

```
mysql> SELECT '.01' <> '0.01';
    -> 1
mysql> SELECT .01 <> '0.01';
    -> 0
mysql> SELECT 'zapp' <> 'zappp';
    -> 1
```

- **<=**

  より少ないか等しい:

```
mysql> SELECT 0.1 <= 2;
    -> 1
```

- **<**

  より少ない:

```
mysql> SELECT 2 < 2;
    -> 0
```

- **>=**

  より多いか等しい:

```
mysql> SELECT 2 >= 2;
    -> 1
```

- **>**

  より多い:

```
mysql> SELECT 2 > 2;
    -> 0
```

- **IS boolean_value**

  boolean_value を TRUE、FALSE、または UNKNOWN にすることができるブール値に対して値をテストします。

```
mysql> SELECT 1 IS TRUE, 0 IS FALSE, NULL IS UNKNOWN;
    -> 1, 1, 1
```

- **IS NOT boolean_value**

  boolean_value を TRUE、FALSE、または UNKNOWN にすることができるブール値に対して値をテストします。

```
mysql> SELECT 1 IS NOT UNKNOWN, 0 IS NOT UNKNOWN, NULL IS NOT UNKNOWN;
    -> 1, 1, 0
```

- **IS NULL**

  値が NULL かどうかをテストします。

```
mysql> SELECT 1 IS NULL, 0 IS NULL, NULL IS NULL;
    -> 0, 0, 1
```

  ODBC プログラムとの連携が正しく機能するように、MySQL では IS NULL の使用時に次の追加機能がサポートされます。

  - sql_auto_is_null 変数が 1 に設定されている場合は、自動的に生成された AUTO_INCREMENT 値を正常に挿入するステートメントのあとに、次の形式のステートメントを発行すれば、その値を検索できます。

```
SELECT * FROM tbl_name WHERE auto_col IS NULL
```

ステートメントが行を返す場合、返される値は LAST_INSERT_ID() 関数を呼び出した場合と同じです。複数行の挿入後の戻り値などについての詳細は、セクション12.14「情報関数」を参照してください。AUTO_INCREMENT 値を正常に挿入できなかった場合、SELECT ステートメントは行を返しません。

IS NULL の比較を使用して AUTO_INCREMENT 値を取得する動作は、sql_auto_is_null = 0 を設定すると無効にできます。セクション5.1.4「サーバーシステム変数」を参照してください。

MySQL 5.6 では sql_auto_is_null のデフォルト値は 0 です。

- NOT NULL として宣言された DATE および DATETIME カラムでは、次のようなステートメントを使用することで、特殊な日付 '0000-00-00' を検索できます。

```
SELECT * FROM tbl_name WHERE date_column IS NULL
```

ODBC では '0000-00-00' 日付値がサポートされていないため、一部の ODBC アプリケーションを取得する際に、これが必要になります。

Obtaining Auto-Increment Values、およびConnector/ODBC Connection ParametersのFLAG_AUTO_IS_NULL オプションについての説明を参照してください。

- IS NOT NULL

値が NULL でないかどうかをテストします。

```
mysql> SELECT 1 IS NOT NULL, 0 IS NOT NULL, NULL IS NOT NULL;
    -> 1, 1, 0
```

- expr BETWEEN min AND max

expr が min より多いか等しく、expr が max より少ないか等しい場合、BETWEEN は 1 を返し、それ以外では 0 を返します。すべての引数の型が同じであれば、これは式 (min <= expr AND expr <= max) と同等です。それ以外の場合は、セクション12.2「式評価での型変換」に記載したルールに従って型変換が実行されますが、3 つのすべての引数に適用されます。

```
mysql> SELECT 2 BETWEEN 1 AND 3, 2 BETWEEN 3 and 1;
    -> 1, 0
mysql> SELECT 1 BETWEEN 2 AND 3;
    -> 0
mysql> SELECT 'b' BETWEEN 'a' AND 'c';
    -> 1
mysql> SELECT 2 BETWEEN 2 AND '3';
    -> 1
mysql> SELECT 2 BETWEEN 2 AND 'x-3';
    -> 0
```

日付または時間の値とともに BETWEEN を使用したときの結果を最適にするには、CAST() を使用して明示的に値を目的のデータ型に変換します。例 : DATETIME を 2 つの DATE 値と比較する場合は、DATE 値を DATETIME 値に変換します。DATE との比較で '2001-1-1' などの文字列定数を使用する場合は、文字列を DATE にキャストします。

- expr NOT BETWEEN min AND max

これは、NOT (expr BETWEEN min AND max) と同じです。

- COALESCE(value,...)

リストの最初の非 NULL 値を返します。非 NULL 値がない場合は、NULL を返します。

```
mysql> SELECT COALESCE(NULL,1);
    -> 1
mysql> SELECT COALESCE(NULL,NULL,NULL);
    -> NULL
```

- GREATEST(value1,value2,...)

2 つ以上の引数がある場合は、最大の (最大値の) 引数を返します。引数は、LEAST() のルールと同じルールを使用して比較されます。

```
mysql> SELECT GREATEST(2,0);
    -> 2
mysql> SELECT GREATEST(34.0,3.0,5.0,767.0);
    -> 767.0
```

```
mysql> SELECT GREATEST('B','A','C');
    -> 'C'
```

引数のいずれかが NULL である場合、GREATEST() は NULL を返します。

- expr IN (value,...)

  expr が IN リストのいずれかの値と等しい場合は 1 を返し、それ以外の場合は 0 を返します。すべての値が定数の場合は、expr の型に従って評価され、ソートされます。その際の項目の検索は、バイナリ検索を使って行われます。つまり、IN 値のリストがすべて定数で構成されている場合、IN は非常に高速です。それ以外の場合は、セクション12.2「式評価での型変換」で説明したルールに従って型変換が実行されますが、すべての引数に適用されます。

```
mysql> SELECT 2 IN (0,3,5,7);
    -> 0
mysql> SELECT 'wefwf' IN ('wee','wefwf','weg');
    -> 1
```

引用符で囲まれた値 (文字列など) と囲まれていない値 (数字など) の比較ルールは異なるため、IN リストの引用符で囲まれた値と囲まれていない値を決して混同しないでください。したがって、型を混同すると、整合性のない結果になる可能性があります。たとえば、IN 式を次のように記述しないでください。

```
SELECT val1 FROM tbl1 WHERE val1 IN (1,2,'a');
```

代わりに、次のように記述してください。

```
SELECT val1 FROM tbl1 WHERE val1 IN ('1','2','a');
```

IN リストの値の数は、max_allowed_packet 値によってのみ制限されます。

SQL の標準に準拠するために、左側の式が NULL である場合だけでなく、リストに一致が見つからない場合やリストの式のいずれかが NULL である場合にも、IN は NULL を返します。

IN() 構文は、特定のタイプのサブクエリーを作成する際にも使用できます。セクション13.2.10.3「ANY、IN、または SOME を使用したサブクエリー」を参照してください。

- expr NOT IN (value,...)

  これは、NOT (expr IN (value,...)) と同じです。

- ISNULL(expr)

  expr が NULL の場合、ISNULL() は 1 を返し、それ以外の場合は 0 を返します。

```
mysql> SELECT ISNULL(1+1);
    -> 0
mysql> SELECT ISNULL(1/0);
    -> 1
```

= の代わりに ISNULL() を使用すると、値が NULL であるかどうかをテストできます。(= を使用して値を NULL と比較すると、常に false が発生します。)

ISNULL() 関数は IS NULL 比較演算子と、いくつかの特殊な動作を共有します。IS NULL の説明を参照してください。

- INTERVAL(N,N1,N2,N3,...)

  N < N1 の場合は 0 を返し、N < N2 などの場合は 1 を返し、N が NULL の場合は -1 を返します。すべての引数は整数として処理されます。この関数が正しく機能するには、N1 < N2 < N3 < ... < Nn とする必要があります。これは、バイナリ検索が使用されていることが理由です (非常に高速)。

```
mysql> SELECT INTERVAL(23, 1, 15, 17, 30, 44, 200);
    -> 3
mysql> SELECT INTERVAL(10, 1, 10, 100, 1000);
    -> 2
mysql> SELECT INTERVAL(22, 23, 30, 44, 200);
    -> 0
```

- LEAST(value1,value2,...)

  2 つ以上の引数がある場合は、最小の (最小値の) 引数を返します。引数は、次のルールを使用して比較されます。

- 引数が NULL である場合、結果は NULL になります。比較は必要ありません。
- 戻り値が INTEGER コンテキストで使用されている場合、またはすべての引数が整数値である場合は、整数として比較されます。
- 戻り値が REAL コンテキストで使用されている場合、またはすべての引数が実数値である場合は、実数として比較されます。
- 引数が数字と文字列が混在して構成されている場合は、数字として比較されます。
- 引数が非バイナリ (文字) 文字列の場合は、非バイナリ文字列として比較されます。
- ほかのすべてのケースでは、引数はバイナリ文字列として比較されます。

```
mysql> SELECT LEAST(2,0);
    -> 0
mysql> SELECT LEAST(34.0,3.0,5.0,767.0);
    -> 3.0
mysql> SELECT LEAST('B','A','C');
    -> 'A'
```

一部のボーダーラインケースでは、前述の変換ルールで異常な結果が生成される可能性があります。

```
mysql> SELECT CAST(LEAST(3600, 9223372036854775808.0) as SIGNED);
    -> -9223372036854775808
```

これは、MySQL が 9223372036854775808.0 を整数のコンテキストで読み取ることが原因で発生します。整数表記では値を保持するのに十分でないため、符号付き整数にラップします。

12.3.3 論理演算子

表 12.4 論理演算子

名前	説明
AND, &&	論理 AND
NOT, !	値を否定します
||, OR	論理 OR
XOR	論理 XOR

SQL では、すべての論理演算子は TRUE、FALSE、または NULL (UNKNOWN) に評価されます。MySQL では、これらは 1 (TRUE)、0 (FALSE)、および NULL として実装されます。この大部分は、さまざまな SQL データベースサーバーに共通のものです。ただし、一部のサーバーは TRUE にゼロ以外の任意の値を返す場合があります。

MySQL では、ゼロ以外の任意の非 NULL 値が TRUE に評価されます。たとえば、次のステートメントはすべて TRUE に評価されます。

```
mysql> SELECT 10 IS TRUE;
-> 1
mysql> SELECT -10 IS TRUE;
-> 1
mysql> SELECT 'string' IS NOT NULL;
-> 1
```

- NOT、!

  NOT 演算。オペランドが 0 の場合は 1 に、オペランドがゼロ以外の場合は 0 にそれぞれ評価され、NOT NULL の場合は NULL が返されます。

```
mysql> SELECT NOT 10;
    -> 0
mysql> SELECT NOT 0;
    -> 1
mysql> SELECT NOT NULL;
    -> NULL
mysql> SELECT ! (1+1);
    -> 0
mysql> SELECT ! 1+1;
    -> 1
```

最後の例では、式が (!1)+1 と同様に評価されるため、1 が生成されています。

- AND、&&

  AND 演算。すべてのオペランドがゼロ以外で非 NULL の場合は 1 に、1 つ以上のオペランドが 0 の場合は 0 に評価され、それ以外の場合は NULL が返されます。

```
mysql> SELECT 1 && 1;
    -> 1
mysql> SELECT 1 && 0;
    -> 0
mysql> SELECT 1 && NULL;
    -> NULL
mysql> SELECT 0 && NULL;
    -> 0
mysql> SELECT NULL && 0;
    -> 0
```

- OR、||

  論理 OR。両方のオペランドが非 NULL であれば、オペランドのいずれかがゼロ以外である場合の結果は 1、それ以外の場合は 0 になります。NULL オペランドが 1 つあれば、ほかのオペランドがゼロ以外である場合の結果は 1、それ以外の場合は NULL になります。両方のオペランドが NULL であれば、結果は NULL になります。

```
mysql> SELECT 1 || 1;
    -> 1
mysql> SELECT 1 || 0;
    -> 1
mysql> SELECT 0 || 0;
    -> 0
mysql> SELECT 0 || NULL;
    -> NULL
mysql> SELECT 1 || NULL;
    -> 1
```

- XOR

  論理 XOR。オペランドのいずれかが NULL である場合は、NULL を返します。非 NULL のオペランドでは、奇数のオペランドがゼロ以外の場合は 1 に評価され、それ以外の場合は 0 が返されます。

```
mysql> SELECT 1 XOR 1;
    -> 0
mysql> SELECT 1 XOR 0;
    -> 1
mysql> SELECT 1 XOR NULL;
    -> NULL
mysql> SELECT 1 XOR 1 XOR 1;
    -> 1
```

  a XOR b は、数学的に (a AND (NOT b)) OR ((NOT a) and b) に等しくなります。

12.3.4 割り当て演算子

表 12.5 割り当て演算子

名前	説明
=	(SET ステートメントの一部として、または UPDATE ステートメントの SET 句の一部として) 値を割り当てます
:=	値を割り当てます

- :=

  割り当て演算子。演算子の左側にあるユーザー変数が右側にある値に代入されます。右側の値は、リテラル値、値を格納する別の変数、またはクエリーの結果を含むスカラー値を生成する任意の有効な式 (この値がスカラー値の場合) である可能性があります。同じ SET ステートメントで、複数の割り当てを実行できます。同じステートメントで、複数の割り当てを実行できます。

  = とは異なり、:= 演算子は比較演算子として解釈されません。つまり、(SET ステートメントだけでなく) 有効な任意の SQL ステートメントで := を使用すれば、値を変数に割り当てることができます。

```
mysql> SELECT @var1, @var2;
```

```
    -> NULL, NULL
mysql> SELECT @var1 := 1, @var2;
    -> 1, NULL
mysql> SELECT @var1, @var2;
    -> 1, NULL
mysql> SELECT @var1, @var2 := @var1;
    -> 1, 1
mysql> SELECT @var1, @var2;
    -> 1, 1

mysql> SELECT @var1:=COUNT(*) FROM t1;
    -> 4
mysql> SELECT @var1;
    -> 4
```

次に示すように、SELECT 以外のステートメント (UPDATE など) でも、:= を使用して値の割り当てを実行できます。

```
mysql> SELECT @var1;
    -> 4
mysql> SELECT * FROM t1;
    -> 1, 3, 5, 7

mysql> UPDATE t1 SET c1 = 2 WHERE c1 = @var1:= 1;
Query OK, 1 row affected (0.00 sec)
Rows matched: 1  Changed: 1  Warnings: 0

mysql> SELECT @var1;
    -> 1
mysql> SELECT * FROM t1;
    -> 2, 3, 5, 7
```

:= 演算子を使用すれば、単一の SQL ステートメントで同じ変数の値の設定と読み取りの両方を行うこともできますが、これは推奨されていません。セクション9.4「ユーザー定義変数」では、これを回避するべき理由について説明されています。

- =

  この演算子は、次の 2 つのパラグラフで説明する 2 つのケースで値の割り当てを実行する際に使用されます。

  SET ステートメントでは、= は、演算子の左側にあるユーザー変数を右側にある値に代入する割り当て演算子として処理されます。(言い換えると、SET ステートメントで使用されると、= は := と同じように処理されます。)右側の値は、リテラル値、値を格納する別の変数、またはクエリーの結果を含むスカラー値を生成する任意の有効な式 (この値がスカラー値の場合) である可能性があります。同じ SET ステートメントで、複数の割り当てを実行できます。

  UPDATE ステートメントの SET 句では、= は割り当て演算子としても機能します。ただし、この場合、UPDATE の一部である WHERE 条件に一致していれば、演算子の左側で指定されたカラムが右側に指定された値であるとみなされます。UPDATE ステートメントの同じ SET 句で、複数の割り当てを実行できます。

  その他のコンテキストでは、= は比較演算子として処理されます。

```
mysql> SELECT @var1, @var2;
    -> NULL, NULL
mysql> SELECT @var1 := 1, @var2;
    -> 1, NULL
mysql> SELECT @var1, @var2;
    -> 1, NULL
mysql> SELECT @var1, @var2 := @var1;
    -> 1, 1
mysql> SELECT @var1, @var2;
    -> 1, 1
```

詳細は、セクション13.7.4「SET 構文」、セクション13.2.11「UPDATE 構文」、およびセクション13.2.10「サブクエリー構文」を参照してください。

12.4 制御フロー関数

表 12.6 フロー制御演算子

名前	説明
CASE	CASE 演算子
IF()	If/else 構文

名前	説明
IFNULL()	Null if/else 構文
NULLIF()	expr1 = expr2 の場合に NULL を返します

- CASE value WHEN [compare_value] THEN result [WHEN [compare_value] THEN result ...] [ELSE result] END

  CASE WHEN [condition] THEN result [WHEN [condition] THEN result ...] [ELSE result] END

  1 番目のバージョンでは、value=compare_value の場合に result が返されます。2 番目のバージョンでは、true である最初の条件の結果が返されます。一致する結果値がなかった場合は、ELSE のあとの結果が返され、ELSE 部分がない場合は、NULL が返されます。

```
mysql> SELECT CASE 1 WHEN 1 THEN 'one'
    ->     WHEN 2 THEN 'two' ELSE 'more' END;
        -> 'one'
mysql> SELECT CASE WHEN 1>0 THEN 'true' ELSE 'false' END;
        -> 'true'
mysql> SELECT CASE BINARY 'B'
    ->     WHEN 'a' THEN 1 WHEN 'b' THEN 2 END;
        -> NULL
```

  CASE 式の戻り型は、すべての戻り値の互換性のある集約型ですが、使用されるコンテキストにも依存します。文字列のコンテキストで使用される場合は、結果は文字列として返されます。数値のコンテキストで使用される場合は、結果が 10 進数値、実数値、または整数値として返されます。

  > **注記**
  >
  > ここで示す CASE 式の構文は、ストアドプログラム内部で使用するために、セクション13.6.5.1「CASE 構文」で説明した SQL CASE ステートメントの構文とはわずかに異なります。CASE ステートメントは ELSE NULL 句を持つことができず、END でなく、END CASE で終了します。

- IF(expr1,expr2,expr3)

  expr1 が TRUE (expr1 <> 0 および expr1 <> NULL) の場合、IF() は expr2 を返します。それ以外の場合は expr3 を返します。IF() は、使用されているコンテキストに応じて、数値または文字列値を返します。

```
mysql> SELECT IF(1>2,2,3);
        -> 3
mysql> SELECT IF(1<2,'yes','no');
        -> 'yes'
mysql> SELECT IF(STRCMP('test','test1'),'no','yes');
        -> 'no'
```

  expr2 と expr3 の一方のみが明示的に NULL である場合は、IF() 関数の結果型は非 NULL 式の型になります。

  IF() のデフォルトの戻り値 (一時テーブルに格納されるときに重要となる場合があります) は、次のように計算されます。

式	戻り値
expr2 または expr3 は文字列を返す	文字列
expr2 または expr3 は浮動小数点値を返す	浮動小数点
expr2 または expr3 は整数を返す	整数

  expr2 と expr3 の両方が文字列で、どちらかの文字列で大文字と小文字が区別される場合は、結果でも大文字と小文字が区別されます。

  > **注記**
  >
  > IF ステートメントもありますが、ここで説明されている IF() 関数とは異なります。セクション13.6.5.2「IF 構文」を参照してください。

- IFNULL(expr1,expr2)

  expr1 が NULL でない場合、IFNULL() は expr1 を返し、それ以外の場合は expr2 を返します。IFNULL() は、使用されているコンテキストに応じて、数値または文字列値を返します。

```
mysql> SELECT IFNULL(1,0);
```

```
    -> 1
mysql> SELECT IFNULL(NULL,10);
    -> 10
mysql> SELECT IFNULL(1/0,10);
    -> 10
mysql> SELECT IFNULL(1/0,'yes');
    -> 'yes'
```

IFNULL(expr1,expr2) のデフォルトの結果値は、STRING、REAL、または INTEGER の順に、2 つの式のよりも「一般的」です。式や MySQL が一時テーブルの IFNULL() で返された値を内部に格納する必要のある場所に基づいて、テーブルの大文字と小文字を考慮してください。

```
mysql> CREATE TABLE tmp SELECT IFNULL(1,'test') AS test;
mysql> DESCRIBE tmp;
+-------+--------------+------+-----+---------+-------+
| Field | Type         | Null | Key | Default | Extra |
+-------+--------------+------+-----+---------+-------+
| test  | varbinary(4) | NO   |     |         |       |
+-------+--------------+------+-----+---------+-------+
```

この例では、test カラムの型は VARBINARY(4) です。

- NULLIF(expr1,expr2)

  expr1 = expr2 が true の場合は NULL を返し、それ以外の場合は expr1 を返します。これは、CASE WHEN expr1 = expr2 THEN NULL ELSE expr1 END と同じです。

```
mysql> SELECT NULLIF(1,1);
    -> NULL
mysql> SELECT NULLIF(1,2);
    -> 1
```

> **注記**
>
> 引数が等しくない場合は、MySQL で expr1 が 2 回評価されます。

12.5 文字列関数

表 12.7 文字列演算子

名前	説明
ASCII()	左端の文字の数値を返します
BIN()	数値のバイナリ表現を含む文字列を返します
BIT_LENGTH()	ビット単位で引数の長さを返します
CHAR()	渡された各整数の文字を返します
CHAR_LENGTH()	引数の文字数を返します
CHARACTER_LENGTH()	CHAR_LENGTH() のシノニムです
CONCAT()	連結された文字列を返します
CONCAT_WS()	連結されたものをセパレータ付きで返します
ELT()	インデックス番号位置の文字列を返します
EXPORT_SET()	値 bits 内の各ビットが設定されている場合は on 文字列を取得し、各ビットが設定されていない場合には off 文字列を取得するように、文字列を返します
FIELD()	後続の引数内で第 1 引数のインデックス (位置) を返します
FIND_IN_SET()	第 2 引数内で第 1 引数のインデックス位置を返します
FORMAT()	指定された小数点以下桁数に書式設定された数値を返します
FROM_BASE64() (導入 5.6.1)	base 64 文字列にデコードして結果を返します
HEX()	10 進値または文字列値の 16 進表現を返します
INSERT()	部分文字列を、指定された位置に指定された文字数だけ挿入します
INSTR()	部分文字列が最初に出現する位置のインデックスを返します

名前	説明
LCASE()	LOWER() のシノニムです
LEFT()	左端から指定された数の文字を返します
LENGTH()	文字列の長さをバイト単位で返します
LIKE	単純なパターン一致
LOAD_FILE()	指定されたファイルをロードします
LOCATE()	部分文字列が最初に出現する位置を返します
LOWER()	引数を小文字で返します
LPAD()	指定された文字列で左からパディングした文字列引数を返します
LTRIM()	先頭の空白を削除します
MAKE_SET()	bits セット内の対応するビットを持つ、カンマ区切り文字列のセットを返します
MATCH	全文検索を実行します
MID()	指定された位置から始まる部分文字列を返します
NOT LIKE	単純なパターン一致の否定
NOT REGEXP	REGEXP の否定
OCT()	数値の 8 進数表現を含む文字列を返します
OCTET_LENGTH()	LENGTH() のシノニムです
ORD()	引数の左端の文字の文字コードを返します
POSITION()	LOCATE() のシノニムです
QUOTE()	SQL ステートメント内で使用するために引数をエスケープします
REGEXP	正規表現を使用したパターン一致
REPEAT()	文字列を指定された回数だけ繰り返します
REPLACE()	指定された文字列の出現箇所を置き換えます
REVERSE()	文字列内の文字を逆順に並べ替えます
RIGHT()	右端から指定された数の文字を返します
RLIKE	REGEXP のシノニムです
RPAD()	指定された回数だけ文字列を追加します
RTRIM()	末尾の空白を削除します
SOUNDEX()	soundex 文字列を返します
SOUNDS LIKE	音声を比較します
SPACE()	指定された数の空白で構成される文字列を返します
STRCMP()	2 つの文字列を比較します
SUBSTR()	指定された部分文字列を返します
SUBSTRING()	指定された部分文字列を返します
SUBSTRING_INDEX()	文字列から、区切り文字が指定された回数出現する前の部分文字列を返します
TO_BASE64() (導入 5.6.1)	base 64 文字列に変換された引数を返します
TRIM()	先頭と末尾にある空白を削除します
UCASE()	UPPER() のシノニムです
UNHEX()	数値の 16 進数表現を含む文字列を返します
UPPER()	大文字に変換します
WEIGHT_STRING()	文字列の重み文字列を返します

文字列値の関数は、結果の長さが max_allowed_packet システム環境変数の値よりも長くなると、NULL を返します。セクション8.11.2「サーバーパラメータのチューニング」を参照してください。

文字列の位置を操作する関数では、最初の位置には数値 1 が付けられます。

長さの引数を取る関数では、整数以外の引数はもっとも近い整数に丸められます。

- ASCII(str)

  文字列 str の左端の文字の数値を返します。str が空の文字列である場合は、0 を返します。str が NULL である場合は NULL を返します。ASCII() は、8 ビット文字の場合に動作します。

```
mysql> SELECT ASCII('2');
    -> 50
mysql> SELECT ASCII(2);
    -> 50
mysql> SELECT ASCII('dx');
    -> 100
```

  ORD() 関数も参照してください。

- BIN(N)

  N のバイナリ値の文字列表現を返します。N は longlong (BIGINT) 数字です。これは、CONV(N,10,2) と同等です。N が NULL である場合は NULL を返します。

```
mysql> SELECT BIN(12);
    -> '1100'
```

- BIT_LENGTH(str)

  文字列 str の長さをビット単位で返します。

```
mysql> SELECT BIT_LENGTH('text');
    -> 32
```

- CHAR(N,... [USING charset_name])

  CHAR() は各 N 引数を整数として解釈し、それらの整数のコード値で指定された文字を構成している文字列を返します。NULL 値はスキップされます。

```
mysql> SELECT CHAR(77,121,83,81,'76');
    -> 'MySQL'
mysql> SELECT CHAR(77,77.3,'77.3');
    -> 'MMM'
```

  255 よりも大きい CHAR() 引数は、複数の結果バイトに変換されます。たとえば、CHAR(256) は CHAR(1,0) に同等で、CHAR(256*256) は CHAR(1,0,0) に同等です。

```
mysql> SELECT HEX(CHAR(1,0)), HEX(CHAR(256));
+----------------+----------------+
| HEX(CHAR(1,0)) | HEX(CHAR(256)) |
+----------------+----------------+
| 0100           | 0100           |
+----------------+----------------+
mysql> SELECT HEX(CHAR(1,0,0)), HEX(CHAR(256*256));
+------------------+--------------------+
| HEX(CHAR(1,0,0)) | HEX(CHAR(256*256)) |
+------------------+--------------------+
| 010000           | 010000             |
+------------------+--------------------+
```

  デフォルトでは、CHAR() はバイナリ文字列を返します。指定された文字セットで文字列を生成するには、オプションの USING 句を使用します。

```
mysql> SELECT CHARSET(CHAR(0x65)), CHARSET(CHAR(0x65 USING utf8));
+--------------------+--------------------------------+
| CHARSET(CHAR(0x65)) | CHARSET(CHAR(0x65 USING utf8)) |
+--------------------+--------------------------------+
| binary             | utf8                           |
+--------------------+--------------------------------+
```

  USING が指定され、結果文字列が指定された文字セットで不正である場合は、警告が発行されます。また、厳密な SQL モードが有効になっている場合は、CHAR() からの結果は NULL になります。

- CHAR_LENGTH(str)

  文字で測定された文字列 str の長さを返します。マルチバイト文字は、単一の文字としてカウントされます。つまり、5 つの 2 バイト文字を含む文字列では、LENGTH() は 10 を返し、CHAR_LENGTH() は 5 を返します。

- CHARACTER_LENGTH(str)

  CHARACTER_LENGTH() は CHAR_LENGTH() のシノニムです。

- CONCAT(str1,str2,...)

  引数を連結することで生成される文字列を返します。1 つ以上の引数を持つ場合があります。すべての引数が非バイナリ文字列の場合は、結果も非バイナリ文字列になります。引数にバイナリ文字列が含まれる場合は、結果はバイナリ文字列になります。数値の引数は、同等の非バイナリ文字列形式に変換されます。

  引数のいずれかかが NULL である場合、CONCAT() は NULL を返します。

```
mysql> SELECT CONCAT('My', 'S', 'QL');
    -> 'MySQL'
mysql> SELECT CONCAT('My', NULL, 'QL');
    -> NULL
mysql> SELECT CONCAT(14.3);
    -> '14.3'
```

  引用符で囲まれた文字列では、文字列を並べて配置することで連結が実行されます。

```
mysql> SELECT 'My' 'S' 'QL';
    -> 'MySQL'
```

- CONCAT_WS(separator,str1,str2,...)

  CONCAT_WS() は Concatenate With Separator (区切り文字を使用した連結) を表し、CONCAT() の特殊な形式です。最初の引数は、残りの引数の区切り文字です。区切り文字は、連結される文字列の間に追加されます。区切り文字は、残りの引数と同様に文字列にすることができます。区切り文字が NULL の場合は、結果も NULL になります。

```
mysql> SELECT CONCAT_WS(',','First name','Second name','Last Name');
    -> 'First name,Second name,Last Name'
mysql> SELECT CONCAT_WS(',','First name',NULL,'Last Name');
    -> 'First name,Last Name'
```

  CONCAT_WS() では、空の文字列がスキップされません。ただし、区切り文字引数のあとの NULL 値はすべてスキップされます。

- ELT(N,str1,str2,str3,...)

  ELT() は、文字列リストの N 番目の要素を返します。N = 1 の場合は str1、N = 2 の場合は str2 のように返します。N が 1 よりも小さいか、引数の数よりも大きい場合は、NULL を返します。ELT() は FIELD() の補数です。

```
mysql> SELECT ELT(1, 'ej', 'Heja', 'hej', 'foo');
    -> 'ej'
mysql> SELECT ELT(4, 'ej', 'Heja', 'hej', 'foo');
    -> 'foo'
```

- EXPORT_SET(bits,on,off[,separator[,number_of_bits]])

  値 bits 内で各ビットが設定されている場合には on 文字列を取得し、値内で各ビットが設定されていない場合には off 文字列を取得するように、文字列を返します。bits のビットは、右から左 (下位ビットから上位ビット) へと検証されます。文字列は、separator 文字列 (デフォルトはカンマ文字 「,」) で区切られた結果に左から右へと追加されます。検証されるビット数は、number_of_bits で指定されます。指定されない場合のデフォルトは 64 です。number_of_bits が 64 よりも大きい場合は、警告なしで 64 に短縮されます。符号なし整数として処理されるため、値 −1 は実際には 64 と同じです。

```
mysql> SELECT EXPORT_SET(5,'Y','N',',',4);
    -> 'Y,N,Y,N'
mysql> SELECT EXPORT_SET(6,'1','0',',',10);
    -> '0,1,1,0,0,0,0,0,0,0'
```

- FIELD(str,str1,str2,str3,...)

  str1、str2、str3、... リスト内で str のインデックス (位置) を返します。str が見つからない場合は、0 を返します。

  FIELD() へのすべての引数が文字列の場合は、すべての引数が文字列として比較されます。すべての引数が数値の場合は、数字として比較されます。それ以外の場合は、引数が倍精度として比較されます。

  NULL ではどの値との等価比較にも失敗するため、str が NULL である場合は、戻り値が 0 になります。FIELD() は ELT() の補数です。

```
mysql> SELECT FIELD('ej', 'Hej', 'ej', 'Heja', 'hej', 'foo');
    -> 2
mysql> SELECT FIELD('fo', 'Hej', 'ej', 'Heja', 'hej', 'foo');
    -> 0
```

- FIND_IN_SET(str,strlist)

  文字列 str が N 部分文字列で構成される文字列リスト strlist 内にある場合は、1 から N までの範囲内の値を返します。文字列リストは、「,」 文字で区切られた部分文字列で構成された文字列です。最初の引数が定数文字列で、2 番目が SET 型のカラムの場合、FIND_IN_SET() 関数はビット演算を使用するために最適化されます。str が strlist 内にない場合、または strlist が空の文字列の場合は、0 を返します。引数のいずれかが NULL である場合は、NULL を返します。最初の引数にカンマ (「,」) 文字が含まれる場合は、この関数が正しく動作しません。

```
mysql> SELECT FIND_IN_SET('b','a,b,c,d');
    -> 2
```

- FORMAT(X,D[,locale])

  数値 X を '#,###,###.##' のような書式に変換し、小数点第 D 位に丸めて、その結果を文字列として返します。D が 0 の場合は、結果に小数点または小数部が含まれません。

  オプションの 3 番目のパラメータを使用すると、結果数の小数点、3 桁の区切り文字、および区切り文字間のグループ化に使用されるロケールを指定できます。許可されるロケール値は、lc_time_names システム変数の有効な値と同じです (セクション10.7「MySQL Server のロケールサポート」を参照してください)。ロケールが指定されていない場合のデフォルトは、'en_US' です。

```
mysql> SELECT FORMAT(12332.123456, 4);
    -> '12,332.1235'
mysql> SELECT FORMAT(12332.1,4);
    -> '12,332.1000'
mysql> SELECT FORMAT(12332.2,0);
    -> '12,332'
mysql> SELECT FORMAT(12332.2,2,'de_DE');
    -> '12.332,20'
```

- FROM_BASE64(str)

  TO_BASE64() で使用される base-64 でエンコードされたルールでエンコードされた文字列が指定され、デコードされた結果をバイナリ文字列として返します。引数が NULL の場合または有効な base-64 文字列でない場合は、結果が NULL になります。ルールのエンコードおよびデコードについての詳細は、TO_BASE64() の説明を参照してください。

  この関数は、MySQL 5.6.1 で追加されました。

```
mysql> SELECT TO_BASE64('abc'), FROM_BASE64(TO_BASE64('abc'));
    -> 'JWJj', 'abc'
```

- HEX(str), HEX(N)

  文字列の引数 str では、HEX() は str の 16 進数文字列表現を返します。str 内の各文字の各バイトは、2 つの 16 進数字に変換されます。(したがって、マルチバイト文字は 2 桁よりも大きくなります。)この演算の逆は、UNHEX() 関数で実行されます。

  数値の引数 N では、HEX() は、longlong (BIGINT) 数字として処理される N の 16 進数文字列表現を返します。これは、CONV(N,10,16) と同等です。この演算の逆は、CONV(HEX(N),16,10) で実行されます。

```
mysql> SELECT 0x616263, HEX('abc'), UNHEX(HEX('abc'));
    -> 'abc', 616263, 'abc'
mysql> SELECT HEX(255), CONV(HEX(255),16,10);
    -> 'FF', 255
```

- INSERT(str,pos,len,newstr)

  位置 pos で始まる部分文字列と、文字列 newstr で置換された len 文字長とともに、文字列 str を返します。pos が文字列の長さに収まらない場合は、元の文字列を返します。len が残りの文字列の長さに収まらない場合は、位置 pos からの残りの文字列を置換します。引数のいずれかが NULL である場合は、NULL を返します。

```
mysql> SELECT INSERT('Quadratic', 3, 4, 'What');
    -> 'QuWhattic'
mysql> SELECT INSERT('Quadratic', -1, 4, 'What');
    -> 'Quadratic'
mysql> SELECT INSERT('Quadratic', 3, 100, 'What');
    -> 'QuWhat'
```

  この関数はマルチバイトセーフです。

- INSTR(str,substr)

  文字列 str 内で部分文字列 substr が最初に出現する位置を返します。これは、引数の順序が逆になる点を除いて、2 つの引数形式の LOCATE() と同じです。

```
mysql> SELECT INSTR('foobarbar', 'bar');
    -> 4
mysql> SELECT INSTR('xbar', 'foobar');
    -> 0
```

  この関数はマルチバイトセーフであり、1 つ以上の引数がバイナリ文字列である場合にのみ大文字と小文字が区別されます。

- LCASE(str)

  LCASE() は LOWER() のシノニムです。

- LEFT(str,len)

  文字列 str から左端の len 文字を返し、引数が NULL である場合は NULL を返します。

```
mysql> SELECT LEFT('foobarbar', 5);
    -> 'fooba'
```

  この関数はマルチバイトセーフです。

- LENGTH(str)

  バイトで測定された文字列 str の長さを返します。マルチバイト文字は、複数のバイトとしてカウントされます。つまり、5 つの 2 バイト文字を含む文字列では、LENGTH() は 10 を返し、CHAR_LENGTH() は 5 を返します。

```
mysql> SELECT LENGTH('text');
    -> 4
```

  > **注記**
  >
  > Length() OpenGIS 空間関数は、MySQL では GLength() という名前です。

- LOAD_FILE(file_name)

  ファイルを読み取り、ファイルの内容を文字列として返します。この関数を使用するには、ファイルがサーバーホストに配置されている必要があり、ファイルへのフルパス名を指定し、FILE 権限を持つ必要があります。ファイルはすべてのユーザーから読み取り可能で、max_allowed_packet バイトよりも小さなサイズである必要があります。secure_file_priv システム変数が空でないディレクトリ名に設定されている場合は、そのディレクトリ内にロード対象のファイルが配置されている必要があります。

  ファイルが存在しない場合、または上記の条件が満たされていないために、ファイルを読み取ることができない場合、この関数は NULL を返します。

  character_set_filesystem システム変数では、リテラル文字列として指定されているファイル名の解釈が制御されます。

```
mysql> UPDATE t
         SET blob_col=LOAD_FILE('/tmp/picture')
         WHERE id=1;
```

- LOCATE(substr,str), LOCATE(substr,str,pos)

  1 番目の構文は、文字列 str 内で、部分文字列 substr が最初に出現する位置を返します。2 番目の構文は、文字列 str 内の位置 pos 以降で、部分文字列 substr が最初に出現する位置を返します。str 内に substr がない場合は、0 を返します。

```
mysql> SELECT LOCATE('bar', 'foobarbar');
    -> 4
mysql> SELECT LOCATE('xbar', 'foobar');
    -> 0
mysql> SELECT LOCATE('bar', 'foobarbar', 5);
    -> 7
```

  この関数はマルチバイトセーフであり、1 つ以上の引数がバイナリ文字列である場合にのみ大文字と小文字が区別されます。

- LOWER(str)

  現在の文字セットのマッピングに従って、すべての文字が小文字に変更された文字列 str を返します。デフォルトは latin1 (cp1252 西ヨーロッパ言語) です。

```
mysql> SELECT LOWER('QUADRATICALLY');
    -> 'quadratically'
```

  LOWER() (および UPPER()) をバイナリ文字列 (BINARY、VARBINARY、BLOB) に適用しても、何の効果もありません。大文字/小文字の変換を実行するには、バイナリ文字列を非バイナリ文字列に変換します。

```
mysql> SET @str = BINARY 'New York';
mysql> SELECT LOWER(@str), LOWER(CONVERT(@str USING latin1));
+-------------+----------------------------------+
| LOWER(@str) | LOWER(CONVERT(@str USING latin1)) |
+-------------+----------------------------------+
| New York    | new york                         |
+-------------+----------------------------------+
```

  Unicode 文字セットの場合、xxx_unicode_520_ci 照合順序およびそれらから派生した言語固有の照合順序では、LOWER() および UPPER() は Unicode 照合順序アルゴリズム (UCA) 5.2.0 に従って動作します。その他の Unicode 照合では、LOWER() および UPPER() は Unicode 照合アルゴリズム (UCA) 4.0.0 に従って動作します。セクション10.1.14.1「Unicode 文字セット」を参照してください。

  この関数はマルチバイトセーフです。

- LPAD(str,len,padstr)

  len 文字の長さになるように文字列 padstr で左にパディングされた文字列 str を返します。str が len よりも長い場合は、戻り値は len 文字に短縮されます。

```
mysql> SELECT LPAD('hi',4,'??');
    -> '??hi'
mysql> SELECT LPAD('hi',1,'??');
    -> 'h'
```

- LTRIM(str)

  先頭の空白文字が削除された文字列 str を返します。

```
mysql> SELECT LTRIM('  barbar');
    -> 'barbar'
```

  この関数はマルチバイトセーフです。

- MAKE_SET(bits,str1,str2,...)

  bits セット内の対応するビットを持つ文字列で構成されるセット値 (「,」 文字で区切られた部分文字列を含む文字列) を返します。str1 はビット 0 に対応し、str2 はビット 1 に対応する、などとなります。str1、str2、... 内の NULL 値は結果に追加されません。

```
mysql> SELECT MAKE_SET(1,'a','b','c');
    -> 'a'
mysql> SELECT MAKE_SET(1 | 4,'hello','nice','world');
    -> 'hello,world'
mysql> SELECT MAKE_SET(1 | 4,'hello','nice',NULL,'world');
    -> 'hello'
mysql> SELECT MAKE_SET(0,'a','b','c');
```

```
    -> ''
```

- MID(str,pos,len)

  MID(str,pos,len) は、SUBSTRING(str,pos,len) のシノニムです。

- OCT(N)

  N の 8 進数の文字列表現を返します。N は longlong (BIGINT) 数字です。これは、CONV(N,10,8) と同等です。N が NULL である場合は NULL を返します。

```
mysql> SELECT OCT(12);
    -> '14'
```

- OCTET_LENGTH(str)

  OCTET_LENGTH() は LENGTH() のシノニムです。

- ORD(str)

  文字列 str の左端の文字がマルチバイト文字である場合は、その文字のコードを返します。コードは、次の計算式を使用して、その構成要素の数値から計算されます。

```
 (1st byte code)
+ (2nd byte code * 256)
+ (3rd byte code * 256^2) ...
```

  左端の文字がマルチバイト文字でない場合は、ORD() は ASCII() 関数と同じ値を返します。

```
mysql> SELECT ORD('2');
    -> 50
```

- POSITION(substr IN str)

  POSITION(substr IN str) は LOCATE(substr,str) のシノニムです。

- QUOTE(str)

  SQL ステートメントで、適切にエスケープされたデータ値として使用できる結果を生成する文字列を引用符で囲みます。一重引用符で囲まれ、バックスラッシュの後ろにバックスラッシュ (「\」)、一重引用符 (「'」)、ASCII NUL、および Control+Z の各インスタンスが続く文字列が返されます。引数が NULL の場合の戻り値は、一重引用符で囲まれていない単語 「NULL」 です。

```
mysql> SELECT QUOTE('Don\'t!');
    -> 'Don\'t!'
mysql> SELECT QUOTE(NULL);
    -> NULL
```

  比較するために、セクション9.1.1「文字列リテラル」およびセクション 23.7.7.54「mysql_real_escape_string()」で、リテラル文字列に対する引用ルールと C API 内の引用ルールを参照してください。

- REPEAT(str,count)

  count 回繰り返された文字列 str で構成される文字列を返します。count が 1 よりも小さい場合は、空の文字列を返します。str または count が NULL である場合は NULL を返します。

```
mysql> SELECT REPEAT('MySQL', 3);
    -> 'MySQLMySQLMySQL'
```

- REPLACE(str,from_str,to_str)

  文字列 from_str のすべての出現箇所が文字列 to_str で置換された、文字列 str を返します。REPLACE() は、from_str を検索する際に、大文字と小文字を区別した一致を実行します。

```
mysql> SELECT REPLACE('www.mysql.com', 'w', 'Ww');
    -> 'WwWwWw.mysql.com'
```

  この関数はマルチバイトセーフです。

- REVERSE(str)

  文字の順序が逆になった文字列 str を返します。

```
mysql> SELECT REVERSE('abc');
       -> 'cba'
```

  この関数はマルチバイトセーフです。

- RIGHT(str,len)

  文字列 str から右端の len 文字を返し、引数が NULL である場合は NULL を返します。

```
mysql> SELECT RIGHT('foobarbar', 4);
       -> 'rbar'
```

  この関数はマルチバイトセーフです。

- RPAD(str,len,padstr)

  len 文字の長さになるように文字列 padstr で右にパディングされた文字列 str を返します。str が len よりも長い場合は、戻り値は len 文字に短縮されます。

```
mysql> SELECT RPAD('hi',5,'?');
       -> 'hi???'
mysql> SELECT RPAD('hi',1,'?');
       -> 'h'
```

  この関数はマルチバイトセーフです。

- RTRIM(str)

  末尾の空白文字が削除された文字列 str を返します。

```
mysql> SELECT RTRIM('barbar   ');
       -> 'barbar'
```

  この関数はマルチバイトセーフです。

- SOUNDEX(str)

  str から soundex 文字列を返します。ほぼ同じ発音の 2 つの文字列は、同じ soundex 文字列を持つはずです。標準の soundex 文字列の長さは 4 文字ですが、SOUNDEX() 関数は任意の長さの文字列を返します。結果で SUBSTRING() を使用すると、標準の soundex 文字列を取得できます。str 内のアルファベット以外の文字はすべて無視されます。A から Z までの範囲外の国際アルファベット文字はすべて、母音として処理されます。

  > **重要**
  >
  > SOUNDEX() の使用時には、次の制限に注意してください。

  - 現在実装されているこの関数は、文字列の言語が英語である場合にのみ機能するように設計されています。その他の言語の文字列では、信頼できる結果が生成されない可能性があります。
  - この関数では、文字列でマルチバイト文字セット (utf-8 など) が使用されている場合に、整合性のある結果が生成されることは保証されません。

    今後のリリースで、このような制限が解除されることを期待しています。詳細は、Bug #22638 を参照してください。

```
mysql> SELECT SOUNDEX('Hello');
       -> 'H400'
mysql> SELECT SOUNDEX('Quadratically');
       -> 'Q36324'
```

  > **注記**
  >
  > この関数には、オリジナルの Soundex アルゴリズムが実装されています。より人気のある拡張バージョンではありません (この作成者も D. Knuth です)。相違点としては、元のバージョンではまず母音が破棄されてから複製が破棄されますが、拡張バージョンではまず複製が破棄されてから母音が破棄されます。

- expr1 SOUNDS LIKE expr2

これは、SOUNDEX(expr1) = SOUNDEX(expr2) と同じです。

- SPACE(N)

  N 空白文字で構成される文字列を返します。

```
mysql> SELECT SPACE(6);
    -> '      '
```

- SUBSTR(str,pos), SUBSTR(str FROM pos), SUBSTR(str,pos,len), SUBSTR(str FROM pos FOR len)

  SUBSTR() は SUBSTRING() のシノニムです。

- SUBSTRING(str,pos), SUBSTRING(str FROM pos), SUBSTRING(str,pos,len), SUBSTRING(str FROM pos FOR len)

  len 引数を付けない形式では、位置 pos で始まる文字列 str からの部分文字列が返されます。len 引数を付けた形式では、位置 pos で始まる文字列 str からの部分文字列 len 文字長が返されます。FROM を使用する形式は、標準の SQL 構文です。また、pos に負の値を使用することもできます。その場合、部分文字列の先頭は文字列の先頭でなく、文字列の末尾からの pos 文字になります。この関数のどの形式でも、pos で負の値を使用できます。

  すべての形式の SUBSTRING() で、部分文字列の抽出が開始される文字列内の最初の文字の位置が 1 とみなされます。

```
mysql> SELECT SUBSTRING('Quadratically',5);
    -> 'ratically'
mysql> SELECT SUBSTRING('foobarbar' FROM 4);
    -> 'barbar'
mysql> SELECT SUBSTRING('Quadratically',5,6);
    -> 'ratica'
mysql> SELECT SUBSTRING('Sakila', -3);
    -> 'ila'
mysql> SELECT SUBSTRING('Sakila', -5, 3);
    -> 'aki'
mysql> SELECT SUBSTRING('Sakila' FROM -4 FOR 2);
    -> 'ki'
```

  この関数はマルチバイトセーフです。

  len が 1 よりも小さい場合は、結果が空の文字列になります。

- SUBSTRING_INDEX(str,delim,count)

  文字列 str から、区切り文字 delim が count 回出現する前の部分文字列を返します。count が正の値の場合は、(左から数えて) 最後の区切り文字の左側にあるすべてが返されます。count が負の値の場合は、(右から数えて) 最後の区切り文字の右側にあるすべてが返されます。SUBSTRING_INDEX() は、delim を検索する際に、大文字と小文字を区別した一致を実行します。

```
mysql> SELECT SUBSTRING_INDEX('www.mysql.com', '.', 2);
    -> 'www.mysql'
mysql> SELECT SUBSTRING_INDEX('www.mysql.com', '.', -2);
    -> 'mysql.com'
```

  この関数はマルチバイトセーフです。

- TO_BASE64(str)

  文字列の引数を base-64 でエンコードされた形式に変換し、その結果を接続文字セットと照合順序が含まれる文字列として返します。引数が文字列でない場合は、変換が実行される前に文字列に変換されます。引数が NULL である場合は、結果も NULL になります。FROM_BASE64() 関数を使用すると、base-64 でエンコードされた文字列をデコードできます。

  この関数は、MySQL 5.6.1 で追加されました。

```
mysql> SELECT TO_BASE64('abc'), FROM_BASE64(TO_BASE64('abc'));
    -> 'JWJj', 'abc'
```

  さまざまな base-64 エンコードスキームが存在します。これらは、TO_BASE64() および FROM_BASE64() で使用されるエンコードおよびデコードのルールです。

  - アルファベット値 62 のエンコードは '+' です。

  - アルファベット値 63 のエンコードは '/' です。
  - エンコードされた出力は、出力可能な 4 文字のグループで構成されます。入力データの各 3 バイトは、4 文字を使用してエンコードされます。最後のグループが不完全な場合は、長さが 4 になるまで '=' 文字でパディングされます。
  - 長い出力を複数の行に分割するために、エンコードされた出力の各 76 文字の後ろに改行が追加されます。
  - デコードでは改行、復帰改行、タブ、および空白が認識および無視されます。

- TRIM([{BOTH | LEADING | TRAILING} [remstr] FROM] str), TRIM([remstr FROM] str)

  すべての remstr プリフィクスまたはサフィクスが削除された文字列 str を返します。BOTH、LEADING、TRAILING のいずれの指定子も指定されない場合は、BOTH が指定されたとみなされます。remstr はオプションであり、指定されない場合は空白文字が削除されます。

```
mysql> SELECT TRIM('  bar   ');
       -> 'bar'
mysql> SELECT TRIM(LEADING 'x' FROM 'xxxbarxxx');
       -> 'barxxx'
mysql> SELECT TRIM(BOTH 'x' FROM 'xxxbarxxx');
       -> 'bar'
mysql> SELECT TRIM(TRAILING 'xyz' FROM 'barxxyz');
       -> 'barx'
```

  この関数はマルチバイトセーフです。

- UCASE(str)

  UCASE() は UPPER() のシノニムです。

- UNHEX(str)

  文字列の引数 str では、UNHEX(str) は引数の各文字ペアを 16 進数として解釈し、その数字で表されたバイトに変換します。戻り値はバイナリ文字列です。

```
mysql> SELECT UNHEX('4D7953514C');
       -> 'MySQL'
mysql> SELECT 0x4D7953514C;
       -> 'MySQL'
mysql> SELECT UNHEX(HEX('string'));
       -> 'string'
mysql> SELECT HEX(UNHEX('1267'));
       -> '1267'
```

  引数文字列内の文字は、正当な 16 進数である必要があります: '0' .. '9'、'A' ..'F'、'a' .. 'f'。引数に 16 進以外の数字が含まれている場合は、結果が NULL になります。

```
mysql> SELECT UNHEX('GG');
+-------------+
| UNHEX('GG') |
+-------------+
| NULL        |
+-------------+
```

  UNHEX() への引数が BINARY カラムである場合は、格納時に値に 0x00 バイトがパディングされますが、それらのバイトは取得時に削除されないため、結果が NULL になる可能性があります。たとえば、'41' は '41 ' として CHAR(3) カラムに格納され、(末尾にパディングされた空白文字が削除された) '41' として取得されるため、カラム値に対する UNHEX() では 'A' が返されます。反対に、'41' は '41\0' として BINARY(3) カラムに格納され、(末尾にパディングされた 0x00 バイトが削除されない) '41\0' として取得されます。'\0' は不正な 16 進数であるため、カラム値に対する UNHEX() では NULL が返されます。

  数値の引数 N の場合、HEX(N) の逆は UNHEX() では実行されません。代わりに、CONV(HEX(N),16,10) を使用してください。HEX() の説明を参照してください。

- UPPER(str)

  現在の文字セットのマッピングに従って、すべての文字が大文字に変更された文字列 str を返します。デフォルトは latin1 (cp1252 西ヨーロッパ言語) です。

```
mysql> SELECT UPPER('Hej');
       -> 'HEJ'
```

UPPER() にも適用される情報については、LOWER() の説明を参照してください。これには、現在は機能が無効になっているバイナリ文字列 (BINARY、VARBINARY、BLOB) の大文字と小文字の変換を実行する方法に関する情報、および Unicode 文字セットの大文字と小文字の変換に関する情報が含まれていました。

この関数はマルチバイトセーフです。

- WEIGHT_STRING(str [AS {CHAR|BINARY}(N)] [LEVEL levels] [flags])

  levels: N [ASC|DESC|REVERSE] [, N [ASC|DESC|REVERSE]] ...

  この関数は、入力文字列の重み文字列を返します。戻り値は、文字列のソートおよび比較の値を表すバイナリ文字列です。これらのプロパティーがあります。

  - WEIGHT_STRING(str1) = WEIGHT_STRING(str2) の場合は、str1 = str2 です (str1 と str2 は等しいとみなされます)。
  - WEIGHT_STRING(str1) < WEIGHT_STRING(str2) の場合は、str1 < str2 です (str1 は str2 の前にソートされます)。

  WEIGHT_STRING() は、照合順序のテストおよびデバッグを行う際、特に新しい照合順序を追加する場合に使用できます。セクション10.4「文字セットへの照合順序の追加」を参照してください。

  入力文字列 str は文字列式です。入力が非バイナリ (文字) 文字列 (CHAR、VARCHAR、TEXT 値など) である場合は、戻り値に文字列の照合順序重みが含まれます。入力がバイナリ (バイト) 文字列 (BINARY、VARBINARY、BLOB 値など) である場合は、戻り値は入力と同じです (バイナリ文字列のバイトごとの重みはバイト値です)。入力が NULL である場合は、WEIGHT_STRING() は NULL を返します。

  例:

```
mysql> SET @s = _latin1 'AB' COLLATE latin1_swedish_ci;
mysql> SELECT @s, HEX(@s), HEX(WEIGHT_STRING(@s));
+------+---------+-------------------------+
| @s   | HEX(@s) | HEX(WEIGHT_STRING(@s)) |
+------+---------+-------------------------+
| AB   | 4142    | 4142                    |
+------+---------+-------------------------+
```

```
mysql> SET @s = _latin1 'ab' COLLATE latin1_swedish_ci;
mysql> SELECT @s, HEX(@s), HEX(WEIGHT_STRING(@s));
+------+---------+-------------------------+
| @s   | HEX(@s) | HEX(WEIGHT_STRING(@s)) |
+------+---------+-------------------------+
| ab   | 6162    | 4142                    |
+------+---------+-------------------------+
```

```
mysql> SET @s = CAST('AB' AS BINARY);
mysql> SELECT @s, HEX(@s), HEX(WEIGHT_STRING(@s));
+------+---------+-------------------------+
| @s   | HEX(@s) | HEX(WEIGHT_STRING(@s)) |
+------+---------+-------------------------+
| AB   | 4142    | 4142                    |
+------+---------+-------------------------+
```

```
mysql> SET @s = CAST('ab' AS BINARY);
mysql> SELECT @s, HEX(@s), HEX(WEIGHT_STRING(@s));
+------+---------+-------------------------+
| @s   | HEX(@s) | HEX(WEIGHT_STRING(@s)) |
+------+---------+-------------------------+
| ab   | 6162    | 6162                    |
+------+---------+-------------------------+
```

  上記の例では、HEX() を使用して WEIGHT_STRING() の結果を表示しています。結果はバイナリ値であるため、結果に出力されない値が含まれるときに出力可能な形式で表示する際に、HEX() が特に役立ちます。

```
mysql> SET @s = CONVERT(0xC39F USING utf8) COLLATE utf8_czech_ci;
mysql> SELECT HEX(WEIGHT_STRING(@s));
+----------------------+
| HEX(WEIGHT_STRING(@s)) |
+----------------------+
| 0FEA0FEA             |
```

```
+------------------------+
```

非 NULL の戻り値では、長さが VARBINARY の最大長内である場合は、値のデータ型が VARBINARY であり、その他の場合はデータ型は BLOB です。

入力文字列を非バイナリまたはバイナリの文字列にキャストし、強制的に指定した長さになるように、AS 句が指定されている場合があります。

- AS CHAR(N) は、文字列を非バイナリ文字列にキャストし、N 文字の長さになるように空白文字で右側をパディングします。N は少なくとも 1 にする必要があります。N が入力文字列の長さよりも小さい場合は、文字列が N 文字まで切り捨てられます。切り捨てられても警告は発生しません。
- AS BINARY(N) は、文字列がバイナリ文字列にキャストされ、N が (文字単位ではなく) バイト単位で測定され、パディングで (空白文字でななく) 0x00 バイトが使用される点を除いて同様です。

```
mysql> SELECT HEX(WEIGHT_STRING('ab' AS CHAR(4)));
+--------------------------------------+
| HEX(WEIGHT_STRING('ab' AS CHAR(4))) |
+--------------------------------------+
| 41422020                             |
+--------------------------------------+
```

```
mysql> SELECT HEX(WEIGHT_STRING('ab' AS BINARY(4)));
+----------------------------------------+
| HEX(WEIGHT_STRING('ab' AS BINARY(4))) |
+----------------------------------------+
| 61620000                               |
+----------------------------------------+
```

戻り値に特定の照合順序レベルに合った重みが含まれるように指定するために、LEVEL 句が指定される場合があります。

カンマで区切られた 1 つ以上の整数のリストとして、またはダッシュで区切られた 2 つの整数の範囲として、LEVEL キーワードの後ろに levels 指定子が指定される場合があります。句読文字の前後の空白の有無は影響しません。

例:

```
LEVEL 1
LEVEL 2, 3, 5
LEVEL 1-3
```

1 よりも低いレベルは 1 として処理されます。入力文字列照合順序の最大値よりも高いレベルは、照合順序の最大値として処理されます。最大値は照合順序ごとに異なりますが、6 よりも大きい値にはなりません。

レベルのリストでは、レベルを小さい順に指定する必要があります。レベルの範囲では、2 番目の数字が 1 番目よりも小さい場合は、1 番目の数字として処理されます (たとえば、4-2 は 4-4 と同じになります)。

LEVEL 句が省略された場合は、MySQL では LEVEL 1 - max であるとみなされます。ここで、max は照合順序の最大レベルです。

LEVEL が (範囲構文ではなく) リスト構文を使用して指定されている場合は、レベルの数字のあとに次の修飾子を指定できます。

- ASC: 変更なしで重みを返します。これはデフォルトです。
- DESC: ビット反転された重みを返します (たとえば、0x78f0 DESC = 0x870f です)。
- REVERSE: 逆順で重み (つまり、最初の文字を最後に、最後の文字を最後にと、逆に並べた文字列の重み) を返します。

例:

```
mysql> SELECT HEX(WEIGHT_STRING(0x007fff LEVEL 1));
+--------------------------------------+
| HEX(WEIGHT_STRING(0x007fff LEVEL 1)) |
+--------------------------------------+
| 007FFF                               |
+--------------------------------------+
```

```
mysql> SELECT HEX(WEIGHT_STRING(0x007fff LEVEL 1 DESC));
+------------------------------------------+
```

```
| HEX(WEIGHT_STRING(0x007fff LEVEL 1 DESC)) |
+-------------------------------------------+
| FF8000                                    |
+-------------------------------------------+
```

```
mysql> SELECT HEX(WEIGHT_STRING(0x007fff LEVEL 1 REVERSE));
+-------------------------------------------+
| HEX(WEIGHT_STRING(0x007fff LEVEL 1 REVERSE)) |
+-------------------------------------------+
| FF7F00                                    |
+-------------------------------------------+
```

```
mysql> SELECT HEX(WEIGHT_STRING(0x007fff LEVEL 1 DESC REVERSE));
+---------------------------------------------------+
| HEX(WEIGHT_STRING(0x007fff LEVEL 1 DESC REVERSE)) |
+---------------------------------------------------+
| 0080FF                                            |
+---------------------------------------------------+
```

現在、flags 句は使用されていません。

12.5.1 文字列比較関数

表 12.8 文字列比較演算子

名前	説明
LIKE	単純なパターン一致
NOT LIKE	単純なパターン一致の否定
STRCMP()	2 つの文字列を比較します

文字列関数に引数としてバイナリ文字列が指定されている場合は、結果の文字列もバイナリ文字列になります。文字列に変換された数字は、バイナリ文字列として処理されます。比較のみがこの影響を受けます。

通常、文字列比較の式で大文字と小文字が区別される場合は、大文字と小文字が区別される方法で比較が実行されます。

- expr LIKE pat [ESCAPE 'escape_char']

  SQL の単純な正規表現比較を使用したパターンマッチング。1 (TRUE) または 0 (FALSE) を返します。expr または pat のいずれかが NULL である場合は、結果も NULL になります。

  パターンはリテラル文字列である必要はありません。たとえば、文字列式やテーブルカラムとして指定できます。

  SQL 標準では、LIKE は文字ごとに一致を実行するため、= 比較演算子とは異なる結果が生成される可能性があります。

```
mysql> SELECT 'ä' LIKE 'ae' COLLATE latin1_german2_ci;
+-----------------------------------------+
| 'ä' LIKE 'ae' COLLATE latin1_german2_ci |
+-----------------------------------------+
|                                       0 |
+-----------------------------------------+
mysql> SELECT 'ä' = 'ae' COLLATE latin1_german2_ci;
+--------------------------------------+
| 'ä' = 'ae' COLLATE latin1_german2_ci |
+--------------------------------------+
|                                    1 |
+--------------------------------------+
```

  特に、末尾の空白は重要です。ただし、= 演算子を使って実行される CHAR や VARCHAR の比較には当てはまりません。

```
mysql> SELECT 'a' = 'a ', 'a' LIKE 'a ';
+------------+--------------+
| 'a' = 'a ' | 'a' LIKE 'a ' |
+------------+--------------+
|          1 |            0 |
+------------+--------------+
1 row in set (0.00 sec)
```

  LIKE では、パターンで次の 2 つのワイルドカード文字を使用できます。

文字	説明
%	0 個の文字も含めて、任意の数の文字に一致します
_	正確に 1 つの文字に一致します

```
mysql> SELECT 'David!' LIKE 'David_';
    -> 1
mysql> SELECT 'David!' LIKE '%D%v%';
    -> 1
```

ワイルドカード文字のリテラルインスタンスをテストするには、その前にエスケープ文字を指定します。ESCAPE 文字を指定しない場合は、「\」と仮定されます。

文字列	説明
\%	1 つの「%」文字に一致します
_	1 つの「_」文字に一致します

```
mysql> SELECT 'David!' LIKE 'David\_';
    -> 0
mysql> SELECT 'David_' LIKE 'David\_';
    -> 1
```

別のエスケープ文字を指定するには、ESCAPE 句を使用します。

```
mysql> SELECT 'David_' LIKE 'David|_' ESCAPE '|';
    -> 1
```

エスケープシーケンスは空にするか、1 文字の長さにするようにしてください。実行時に、式は定数として評価される必要があります。NO_BACKSLASH_ESCAPES SQL モードが有効になっている場合は、シーケンスを空にできません。

次の 2 つのステートメントは、オペランドのいずれかがバイナリ文字列である場合を除いて、文字列の比較では大文字と小文字が区別されないことを示しています。

```
mysql> SELECT 'abc' LIKE 'ABC';
    -> 1
mysql> SELECT 'abc' LIKE BINARY 'ABC';
    -> 0
```

MySQL では、数値式で LIKE が許可されます。(これは、標準 SQL の LIKE の拡張機能です。)

```
mysql> SELECT 10 LIKE '1%';
    -> 1
```

> 注記
>
> MySQL では文字列で C のエスケープ構文 (たとえば、改行文字を表すために「\n」) が使用されているため、LIKE 文字列で使用される「\」はすべて二重に指定する必要があります。たとえば、「\n」を検索するには、「\\n」と指定します。「\」を検索するには、「\\\\」と指定します。これは、バックスラッシュはパーサーによって一度削除され、パターン一致が実行されるときにも再度削除される結果、一致対象のバックスラッシュは 1 つしか残らないためです。
>
> 例外: パターン文字列の末尾では、バックスラッシュを「\\」と指定できます。文字列の末尾では、エスケープの後ろに何もないため、バックスラッシュはそれ自体を表します。テーブルに次の値が含まれると仮定します。
>
> ```
> mysql> SELECT filename FROM t1;
> +--------------+
> | filename     |
> +--------------+
> | C:           |
> | C:\          |
> | C:\Programs  |
> | C:\Programs\ |
> ```

```
+--------------+
```

バックスラッシュで終わる値をテストするには、次のパターンのいずれかを使用すると値をマッチングできます。

```
mysql> SELECT filename, filename LIKE '%\\' FROM t1;
+--------------+---------------------+
| filename     | filename LIKE '%\\' |
+--------------+---------------------+
| C:           |                   0 |
| C:\          |                   1 |
| C:\Programs  |                   0 |
| C:\Programs\ |                   1 |
+--------------+---------------------+

mysql> SELECT filename, filename LIKE '%\\\\' FROM t1;
+--------------+----------------------+
| filename     | filename LIKE '%\\\\' |
+--------------+----------------------+
| C:           |                    0 |
| C:\          |                    1 |
| C:\Programs  |                    0 |
| C:\Programs\ |                    1 |
+--------------+----------------------+
```

- expr NOT LIKE pat [ESCAPE 'escape_char']

  これは、NOT (expr LIKE pat [ESCAPE 'escape_char']) と同じです。

  注記

  NULL を含むカラムとの NOT LIKE 比較を伴う集計クエリーでは、予想外の結果が生成される可能性があります。たとえば、次のテーブルとデータを検討してください。

  ```
  CREATE TABLE foo (bar VARCHAR(10));

  INSERT INTO foo VALUES (NULL), (NULL);
  ```

  クエリー SELECT COUNT(*) FROM foo WHERE bar LIKE '%baz%'; は 0 を返します。SELECT COUNT(*) FROM foo WHERE bar NOT LIKE '%baz%'; は 2 を返すと想定しています。ただし、事実は異なります。2 番目のクエリーは 0 を返します。これは、expr の値に関係なく、NULL NOT LIKE expr は常に NULL を返すためです。NULL を伴う集計クエリー、および NOT RLIKE または NOT REGEXP を使用する比較でも、同じことが当てはまります。このような場合は、次に示すように、(AND ではなく) OR を使用して NOT NULL を明示的にテストする必要があります。

  ```
  SELECT COUNT(*) FROM foo WHERE bar NOT LIKE '%baz%' OR bar IS NULL;
  ```

- STRCMP(expr1,expr2)

  STRCMP() は、文字列が同じ場合は 0 を返し、現在のソート順に従って 1 番目の引数が 2 番目よりも小さい場合は -1、それ以外の場合は 1 を返します。

  ```
  mysql> SELECT STRCMP('text', 'text2');
      -> -1
  mysql> SELECT STRCMP('text2', 'text');
      -> 1
  mysql> SELECT STRCMP('text', 'text');
      -> 0
  ```

  STRCMP() は、引数の照合順序を使用して比較を実行します。

  ```
  mysql> SET @s1 = _latin1 'x' COLLATE latin1_general_ci;
  mysql> SET @s2 = _latin1 'X' COLLATE latin1_general_ci;
  mysql> SET @s3 = _latin1 'x' COLLATE latin1_general_cs;
  mysql> SET @s4 = _latin1 'X' COLLATE latin1_general_cs;
  mysql> SELECT STRCMP(@s1, @s2), STRCMP(@s3, @s4);
  +-----------------+-----------------+
  | STRCMP(@s1, @s2) | STRCMP(@s3, @s4) |
  +-----------------+-----------------+
  |               0 |               1 |
  +-----------------+-----------------+
  ```

  照合順序の互換性がない場合は、その他との互換性を保つために、引数のいずれかを変換する必要があります。セクション10.1.7.5「式の照合順序」を参照してください。

```
mysql> SELECT STRCMP(@s1, @s3);
ERROR 1267 (HY000): Illegal mix of collations (latin1_general_ci,IMPLICIT) and (latin1_general_cs,IMPLICIT) for operation 'strcmp'
mysql> SELECT STRCMP(@s1, @s3 COLLATE latin1_general_ci);
+-------------------------------------------+
| STRCMP(@s1, @s3 COLLATE latin1_general_ci) |
+-------------------------------------------+
|                                         0 |
+-------------------------------------------+
```

12.5.2 正規表現

表 12.9 文字列正規表現演算子

名前	説明
NOT REGEXP	REGEXP の否定
REGEXP	正規表現を使用したパターン一致
RLIKE	REGEXP のシノニムです

正規表現は、複雑な検索でパターンを指定する強力な方法です。

MySQL では、POSIX 1003.2 に準拠することを目的とした Henry Spencer 氏による正規表現の実装が使用されます。MySQL では、SQL ステートメントで REGEXP 演算子とともに実行されるパターンマッチング演算をサポートするために、拡張バージョンが使用されています。

このセクションでは、MySQL の REGEXP 演算で使用できる特殊な文字や構造を例を示して要約しています。Henry Spencer 氏の regex(7) マニュアルページで検索できる詳細がすべて含まれているわけではありません。そのマニュアルページは、MySQL のソース配布の regex ディレクトリにある regex.7 ファイルに含まれています。セクション3.3.4.7「パターンマッチング」も参照してください。

正規表現演算子

- expr NOT REGEXP pat, expr NOT RLIKE pat

  これは、NOT (expr REGEXP pat) と同じです。

- expr REGEXP pat、expr RLIKE pat

  パターン pat と比較して、文字列式 expr のパターン一致を実行します。パターンは、拡張正規表現にすることができます。構文については、このセクションの後半で説明します。expr が pat と一致する場合は 1 を返し、それ以外の場合は 0 を返します。expr または pat のいずれかが NULL である場合は、結果も NULL になります。RLIKE は、mSQL との互換性を確保するために用意された REGEXP のシノニムです。

  パターンはリテラル文字列である必要はありません。たとえば、文字列式やテーブルカラムとして指定できます。

  > **注記**
  >
  > MySQL では文字列で C の エスケープ構文 (たとえば、改行文字を表すために 「\n」) が使用されているため、REGEXP 文字列で使用される 「\」 はすべて二重に指定する必要があります。

  REGEXP では、バイナリ文字列で使用される場合を除いて、大文字と小文字が区別されません。

```
mysql> SELECT 'Monty!' REGEXP '.*';
    -> 1
mysql> SELECT 'new*\n*line' REGEXP 'new\\*.\\*line';
    -> 1
mysql> SELECT 'a' REGEXP 'A', 'a' REGEXP BINARY 'A';
    -> 1  0
mysql> SELECT 'a' REGEXP '^[a-d]';
    -> 1
```

REGEXP および RLIKE では、文字の型を決定する際および比較を実行する際に、引数の文字セットおよび照合順序が使用されます。引数にさまざまな文字セットまたは照合順序が含まれる場合は、セクション10.1.7.5「式の照合順序」で説明するとおりに、型変換属性ルールが適用されます。

> **警告**
>
> REGEXP および RLIKE 演算子はバイト単位で機能するため、マルチバイトセーフではなく、マルチバイト文字セットを使用すると想定外の結果が生成される可能性があります。さらに、これらの演算子ではそのバイト値に基づいて文字が比較されるため、アクセント記号付き文字は、指定された照合順序では等しいとみなされた場合でも、等しいとして比較されない可能性があります。

正規表現の構文

正規表現では、文字列のセットが記述されます。もっとも単純な正規表現は、特殊文字を使用していないものです。たとえば、正規表現 hello は hello にのみ一致します。

重要な正規表現では、複数の文字列に一致できるように特定の特殊構造が使用されます。たとえば、正規表現 hello|word は、文字列 hello または文字列 word に一致します。

さらに複雑な例として、正規表現 B[an]*s は、文字列 Bananas、Baaaaas、Bs のいずれか、および B で始まり、s で終わり、その間に任意の数字の a または n 文字が含まれるその他の文字列に一致します。

REGEXP 演算子の正規表現では、次の特殊文字および構造のいずれかが使用される場合があります。

- ^

  文字列の先頭に一致します。

```
mysql> SELECT 'fo\nfo' REGEXP '^fo$';             -> 0
mysql> SELECT 'fofo' REGEXP '^fo';                -> 1
```

- $

  文字列の末尾に一致します。

```
mysql> SELECT 'fo\no' REGEXP '^fo\no$';           -> 1
mysql> SELECT 'fo\no' REGEXP '^fo$';              -> 0
```

- .

  任意の文字 (復帰改行および改行を含む) に一致します。

```
mysql> SELECT 'fofo' REGEXP '^f.*$';              -> 1
mysql> SELECT 'fo\r\nfo' REGEXP '^f.*$';          -> 1
```

- a*

  ゼロ個以上の a 文字のシーケンスに一致します。

```
mysql> SELECT 'Ban' REGEXP '^Ba*n';               -> 1
mysql> SELECT 'Baaan' REGEXP '^Ba*n';             -> 1
mysql> SELECT 'Bn' REGEXP '^Ba*n';                -> 1
```

- a+

  1 個以上の a 文字のシーケンスに一致します。

```
mysql> SELECT 'Ban' REGEXP '^Ba+n';               -> 1
mysql> SELECT 'Bn' REGEXP '^Ba+n';                -> 0
```

- a?

  ゼロまたは 1 個の a 文字に一致します。

```
mysql> SELECT 'Bn' REGEXP '^Ba?n';                -> 1
mysql> SELECT 'Ban' REGEXP '^Ba?n';               -> 1
mysql> SELECT 'Baan' REGEXP '^Ba?n';              -> 0
```

- de|abc

  シーケンス de または abc のいずれかに一致します。

```
mysql> SELECT 'pi' REGEXP 'pi|apa';               -> 1
mysql> SELECT 'axe' REGEXP 'pi|apa';              -> 0
mysql> SELECT 'apa' REGEXP 'pi|apa';              -> 1
mysql> SELECT 'apa' REGEXP '^(pi|apa)$';          -> 1
```

```
mysql> SELECT 'pi' REGEXP '^(pi|apa)$';          -> 1
mysql> SELECT 'pix' REGEXP '^(pi|apa)$';          -> 0
```

- (abc)*

  シーケンス abc のゼロ個以上のインスタンスに一致します。

```
mysql> SELECT 'pi' REGEXP '^(pi)*$';          -> 1
mysql> SELECT 'pip' REGEXP '^(pi)*$';          -> 0
mysql> SELECT 'pipi' REGEXP '^(pi)*$';          -> 1
```

- {1}, {2,3}

  {n} または {m,n} 注釈では、パターンの前の原子 (または「部分」) の数多くの出現に一致する正規表現を記述するより一般的な方法が提供されます。m および n は整数です。

  - a*

    a{0,} と記述できます。

  - a+

    a{1,} と記述できます。

  - a?

    a{0,1} と記述できます。

  より厳密に言えば、a{n} は、a の正確に n 個のインスタンスに一致します。a{n,} は、a の n 個以上のインスタンスに一致します。a{m,n} は、a の m 個から n 個までのインスタンスに一致します。

  m および n は、0 から RE_DUP_MAX (デフォルトは 255) までの範囲内である必要があります。m および n の両方が指定されている場合は、m を n 以下にする必要があります。

```
mysql> SELECT 'abcde' REGEXP 'a[bcd]{2}e';          -> 0
mysql> SELECT 'abcde' REGEXP 'a[bcd]{3}e';          -> 1
mysql> SELECT 'abcde' REGEXP 'a[bcd]{1,10}e';          -> 1
```

- [a-dX], [^a-dX]

  a、b、c、d、または X である (^ が使用されている場合はそれ以外の) 任意の文字に一致します。2 つの文字の間の - 文字によって、1 番目の文字から 2 番目の文字までのすべての文字に一致する範囲が形成されます。たとえば、[0-9] は任意の 10 進数に一致します。リテラル文字] を含めるには、左括弧 [の直後に記述する必要があります。リテラル文字 - を含めるには、先頭または末尾に記述する必要があります。[] ペアの内側に定義された特殊な意味を持たない文字は、それ自体としか一致しません。

```
mysql> SELECT 'aXbc' REGEXP '[a-dXYZ]';          -> 1
mysql> SELECT 'aXbc' REGEXP '^[a-dXYZ]$';          -> 0
mysql> SELECT 'aXbc' REGEXP '^[a-dXYZ]+$';          -> 1
mysql> SELECT 'aXbc' REGEXP '^[^a-dXYZ]+$';          -> 0
mysql> SELECT 'gheis' REGEXP '^[^a-dXYZ]+$';          -> 1
mysql> SELECT 'gheisa' REGEXP '^[^a-dXYZ]+$';          -> 0
```

- [.characters.]

  ([と] を使用して記述された) 括弧式内で、その照合要素の文字シーケンスに一致します。characters は、単一の文字または newline などの文字名です。次の表には、許可されている文字名を一覧表示します。

  次の表には、許可されている文字名および一致する文字を表示します。数値として指定された文字では、値が 8 進数で表記されます。

名前	文字	名前	文字
NUL	0	SOH	001
STX	002	ETX	003
EOT	004	ENQ	005
ACK	006	BEL	007
alert	007	BS	010
backspace	'\b'	HT	011

名前	文字	名前	文字
tab	'\t'	LF	012
newline	'\n'	VT	013
vertical-tab	'\v'	FF	014
form-feed	'\f'	CR	015
carriage-return	'\r'	SO	016
SI	017	DLE	020
DC1	021	DC2	022
DC3	023	DC4	024
NAK	025	SYN	026
ETB	027	CAN	030
EM	031	SUB	032
ESC	033	IS4	034
FS	034	IS3	035
GS	035	IS2	036
RS	036	IS1	037
US	037	space	' '
exclamation-mark	'!'	quotation-mark	'"'
number-sign	'#'	dollar-sign	'$'
percent-sign	'%'	ampersand	'&'
apostrophe	'\''	left-parenthesis	'('
right-parenthesis	')'	asterisk	'*'
plus-sign	'+'	comma	','
hyphen	'-'	hyphen-minus	'-'
period	'.'	full-stop	'.'
slash	'/'	solidus	'/'
zero	'0'	one	'1'
two	'2'	three	'3'
four	'4'	five	'5'
six	'6'	seven	'7'
eight	'8'	nine	'9'
colon	':'	semicolon	';'
less-than-sign	'<'	equals-sign	'='
greater-than-sign	'>'	question-mark	'?'
commercial-at	'@'	left-square-bracket	'['
backslash	'\\'	reverse-solidus	'\\'
right-square-bracket	']'	circumflex	'^'
circumflex-accent	'^'	underscore	'_'
low-line	'_'	grave-accent	'`'
left-brace	'{'	left-curly-bracket	'{'
vertical-line	'|'	right-brace	'}'
right-curly-bracket	'}'	tilde	'~'
DEL	177		

```
mysql> SELECT '~' REGEXP '[[.~.]]';              -> 1
mysql> SELECT '~' REGEXP '[[.tilde.]]';          -> 1
```

- [=character_class=]

  ([と] 使用して記述された) 括弧式内の [=character_class=] は、等価クラスを表します。これは、同じ照合順序値を持つすべての文字 (それ自体を含む) に一致します。たとえば、o および (+) が等価クラスのメンバーである場合、[[=o=]]、[[=(+)=]]、および [o(+)] はすべてシノニムです。等価クラスは、範囲の終点として使用できない場合もあります。

- [:character_class:]

  ([と] を使用して記述された)括弧式内の [:character_class:] は、そのクラスに属するすべての文字と一致する文字クラスを表します。次の表には、標準のクラス名を一覧表示します。これらの名前は、ctype(3) のマニュアルページで定義されている文字クラスを表しています。特定の口ケールでは、ほかのクラス名が提供される場合もあります。文字クラスは、範囲の終点として使用できない場合もあります。

文字クラス名	意味
alnum	英数文字
alpha	アルファベット文字
blank	空白文字
cntrl	制御文字
digit	数字文字
graph	図形文字
lower	小文字アルファベット文字
print	図形または空白文字
punct	句読点文字
space	空白、タブ、改行、および復帰改行
upper	大文字アルファベット文字
xdigit	16 進数文字

```
mysql> SELECT 'justalnums' REGEXP '[[:alnum:]]+';      -> 1
mysql> SELECT '!!' REGEXP '[[:alnum:]]+';              -> 0
```

- [[:<:]], [[:>:]]

  これらのマーカーは、単語境界を表します。単語の先頭と末尾にそれぞれ一致します。単語とは、前後に単語文字が存在しない単語文字のシーケンスです。単語文字とは、alnum クラス内の英数文字またはアンダースコア (_) です。

```
mysql> SELECT 'a word a' REGEXP '[[:<:]]word[[:>:]]';   -> 1
mysql> SELECT 'a xword a' REGEXP '[[:<:]]word[[:>:]]';  -> 0
```

正規表現で特殊文字のリテラルインスタンスを使用するには、前に 2 つのバックスラッシュ (\) 文字を付けます。MySQL パーサーが 2 つのバックスラッシュの一方を解釈し、正規表現ライブラリがもう一方を解釈します。たとえば、特殊文字 + を含む文字列 1+2 に一致する正規表現は、次のうちで最後のものだけです。

```
mysql> SELECT '1+2' REGEXP '1+2';                -> 0
mysql> SELECT '1+2' REGEXP '1\+2';               -> 0
mysql> SELECT '1+2' REGEXP '1\\+2';              -> 1
```

12.6 数値関数と演算子

表 12.10 数値関数と演算子

名前	説明
ABS()	絶対値を返します
ACOS()	アークコサインを返します
ASIN()	アークサインを返します
ATAN()	アークタンジェントを返します
ATAN2(), ATAN()	2 つの引数のアークタンジェントを返します
CEIL()	引数以上のもっとも小さな整数値を返します

名前	説明
CEILING()	引数以上のもっとも小さな整数値を返します
CONV()	数値を異なる基数間で変換します
COS()	コサインを返します
COT()	コタンジェントを返します
CRC32()	巡回冗長検査値を計算します
DEGREES()	ラジアンを角度に変換します
DIV	整数除算
/	除算演算子
EXP()	累乗します
FLOOR()	引数以下のもっとも大きな整数値を返します
LN()	引数の自然対数を返します
LOG()	最初の引数の自然対数を返します
LOG10()	引数の底 10 の対数を返します
LOG2()	引数の底 2 の対数を返します
-	減算演算子
MOD()	余りを返します
%, MOD	モジュロ演算子
PI()	pi の値を返します
+	加算演算子
POW()	指定した指数で累乗された引数を返します
POWER()	指定した指数で累乗された引数を返します
RADIANS()	ラジアンに変換された引数を返します
RAND()	ランダムな浮動小数点値を返します
ROUND()	引数を丸めます
SIGN()	引数の符号を返します
SIN()	引数のサインを返します
SQRT()	引数の平方根を返します
TAN()	引数のタンジェントを返します
*	乗算演算子
TRUNCATE()	指定された小数点以下の桁数に切り捨てます
-	引数の符号を変更します

12.6.1 算術演算子

表 12.11 算術演算子

名前	説明
DIV	整数除算
/	除算演算子
-	減算演算子
%, MOD	モジュロ演算子
+	加算演算子
*	乗算演算子
-	引数の符号を変更します

通常の算術演算子を使用できます。結果は次のルールに従って決定されます。

- -、+、および * の場合は、両方のオペランドが整数であれば、結果が BIGINT (64 ビット) 精度で計算されます。
- 両方のオペランドが整数で、いずれかが符合なしの場合は、結果が符合なし整数になります。減算では、NO_UNSIGNED_SUBTRACTION SQL モードが有効になっている場合は、オペランドのいずれかが符号なしでも、結果が符号付きになります。
- +、-、/、*、% のオペランドのいずれかが実数値または文字列値である場合は、結果の精度が、最大の精度を持つオペランドの精度になります。
- / を使用して実行される除算では、2 つの正確な値のオペランドを使用したときの結果のスケールが、1 番目のオペランドに div_precision_increment システム変数 (デフォルトでは 4) の値を加えた値になります。たとえば、式 5.05 / 0.014 の結果のスケールは、小数点以下 6 桁になります (360.714286)。

ネストされた計算が各コンポーネントの精度を暗黙的に示すように、これらのルールは演算ごとに適用されます。したがって、(14620 / 9432456) / (24250 / 9432456) では、まず (0.0014) / (0.0026) に解かれて、最終的な結果は小数点以下 8 桁 (0.60288653) になります。

このようなルールおよびそれらが適用される方法があるため、計算のコンポーネントおよびサブコンポーネントで適切な精度レベルが使用されていることを慎重に確認してください。セクション12.10「キャスト関数と演算子」を参照してください。

数値式評価でのオーバーフロー処理については、セクション11.2.6「範囲外およびオーバーフローの処理」を参照してください。

算術演算子は数字に適用されます。その他の型の値では、代替の演算が使用できる場合もあります。たとえば、日付値を追加するには、DATE_ADD() を使用します。セクション12.7「日付および時間関数」を参照してください。

- +

  加算:

```
mysql> SELECT 3+5;
    -> 8
```

- -

  減算:

```
mysql> SELECT 3-5;
    -> -2
```

- -

  単項マイナス。この演算子は、オペランドの符号を変更します。

```
mysql> SELECT - 2;
    -> -2
```

> **注記**
>
> この演算子が BIGINT で使用される場合は、戻り値も BIGINT になります。つまり、値 -2^{63} を持つ可能性のある整数では、- の使用を避けるべきです。

- *

  乗算:

```
mysql> SELECT 3*5;
    -> 15
mysql> SELECT 18014398509481984*18014398509481984.0;
    -> 324518553658426726783156020576256.0
mysql> SELECT 18014398509481984*18014398509481984;
    -> out-of-range error
```

  最後の式では、整数乗算の結果が 64 ビット範囲の BIGINT 計算を超過するため、エラーが生成されます。(セクション11.2「数値型」を参照してください。)

- /

  除算:

```
mysql> SELECT 3/5;
    -> 0.60
```

ゼロによる除算では、NULL の結果が生成されます。

```
mysql> SELECT 102/(1-1);
    -> NULL
```

結果が整数に変換されるコンテキストで実行される場合にのみ、除算は BIGINT 算術を使用して計算されます。

- DIV

  整数除算。FLOOR() に類似していますが、BIGINT 値でも安全です。

  MySQL 5.6 では、オペランドのいずれかが整数以外の型である場合は、結果が BIGINT に変換される前に、オペランドが DECIMAL に変換され、DECIMAL 算術を使用して除算されます。結果が BIGINT の範囲を超過する場合は、エラーが発生します。

```
mysql> SELECT 5 DIV 2;
    -> 2
```

- N % M, N MOD M

  モジュロ演算。M で除算された N の余りを返します。詳細は、セクション12.6.2「数学関数」の MOD() 関数に関する説明を参照してください。

12.6.2 数学関数

表 12.12 数学関数

名前	説明
ABS()	絶対値を返します
ACOS()	アークコサインを返します
ASIN()	アークサインを返します
ATAN()	アークタンジェントを返します
ATAN2(), ATAN()	2 つの引数のアークタンジェントを返します
CEIL()	引数以上のもっとも小さな整数値を返します
CEILING()	引数以上のもっとも小さな整数値を返します
CONV()	数値を異なる基数間で変換します
COS()	コサインを返します
COT()	コタンジェントを返します
CRC32()	巡回冗長検査値を計算します
DEGREES()	ラジアンを角度に変換します
EXP()	累乗します
FLOOR()	引数以下のもっとも大きな整数値を返します
LN()	引数の自然対数を返します
LOG()	最初の引数の自然対数を返します
LOG10()	引数の底 10 の対数を返します
LOG2()	引数の底 2 の対数を返します
MOD()	余りを返します
PI()	pi の値を返します
POW()	指定した指数で累乗された引数を返します
POWER()	指定した指数で累乗された引数を返します
RADIANS()	ラジアンに変換された引数を返します
RAND()	ランダムな浮動小数点値を返します

名前	説明
ROUND()	引数を丸めます
SIGN()	引数の符号を返します
SIN()	引数のサインを返します
SQRT()	引数の平方根を返します
TAN()	引数のタンジェントを返します
TRUNCATE()	指定された小数点以下の桁数に切り捨てます

すべての数学関数は、エラーの発生時に NULL を返します。

- ABS(X)

  X の絶対値を返します。

```
mysql> SELECT ABS(2);
       -> 2
mysql> SELECT ABS(-32);
       -> 32
```

  この関数は、BIGINT 値でも安全に使用できます。

- ACOS(X)

  X のアークコサイン (つまり、コサインが X である値) を返します。X が -1 から 1 までの範囲内にない場合は、NULL を返します。

```
mysql> SELECT ACOS(1);
       -> 0
mysql> SELECT ACOS(1.0001);
       -> NULL
mysql> SELECT ACOS(0);
       -> 1.5707963267949
```

- ASIN(X)

  X のアークサイン (つまり、サインが X である値) を返します。X が -1 から 1 までの範囲内にない場合は、NULL を返します。

```
mysql> SELECT ASIN(0.2);
       -> 0.20135792079033
mysql> SELECT ASIN('foo');

+-------------+
| ASIN('foo') |
+-------------+
|           0 |
+-------------+
1 row in set, 1 warning (0.00 sec)

mysql> SHOW WARNINGS;
+---------+------+----------------------------------------+
| Level   | Code | Message                                |
+---------+------+----------------------------------------+
| Warning | 1292 | Truncated incorrect DOUBLE value: 'foo' |
+---------+------+----------------------------------------+
```

- ATAN(X)

  X のアークタンジェント (つまり、タンジェントが X である値) を返します。

```
mysql> SELECT ATAN(2);
       -> 1.1071487177941
mysql> SELECT ATAN(-2);
       -> -1.1071487177941
```

- ATAN(Y,X), ATAN2(Y,X)

  2 つの変数 X および Y のアークタンジェントを返します。これは、両方の引数の符号が結果の象限の判定に使用される点を除いて、Y / X のアークタンジェントの計算と同様です。

```
mysql> SELECT ATAN(-2,2);
```

```
    -> -0.78539816339745
mysql> SELECT ATAN2(PI(),0);
    -> 1.5707963267949
```

- CEIL(X)

  CEIL() は CEILING() のシノニムです。

- CEILING(X)

  X 以上で最小の整数値を返します。

```
mysql> SELECT CEILING(1.23);
    -> 2
mysql> SELECT CEILING(-1.23);
    -> -1
```

  引数が厳密値数値の場合は、戻り値の型も厳密値数値になります。引数が文字列または浮動小数点の場合は、戻り値の型が浮動小数点になります。

- CONV(N,from_base,to_base)

  数値を異なる基数間で変換します。基数 from_base から基数 to_base に変換された数値 N の文字列表現を返します。引数のいずれかが NULL である場合は、NULL を返します。引数 N は整数として解釈されますが、整数または文字列として指定される場合もあります。最小の基数は 2 で、最大の基数は 36 です。to_base が負の数字である場合は、N は符号付きの数字とみなされます。それ以外の場合は、N は符号なしとみなされます。CONV() は 64 ビット精度で動作します。

```
mysql> SELECT CONV('a',16,2);
    -> '1010'
mysql> SELECT CONV('6E',18,8);
    -> '172'
mysql> SELECT CONV(-17,10,-18);
    -> '-H'
mysql> SELECT CONV(10+'10'+'10'+0xa,10,10);
    -> '40'
```

- COS(X)

  X のコサインを返します。X はラジアンで指定されます。

```
mysql> SELECT COS(PI());
    -> -1
```

- COT(X)

  X のコタンジェントを返します。

```
mysql> SELECT COT(12);
    -> -1.5726734063977
mysql> SELECT COT(0);
    -> NULL
```

- CRC32(expr)

  巡回冗長検査値を計算し、32 ビット値の符号なし値を返します。引数が NULL である場合は、結果も NULL になります。引数は文字列であると想定され、(可能な場合は) 文字列でない場合でも文字列として処理されます。

```
mysql> SELECT CRC32('MySQL');
    -> 3259397556
mysql> SELECT CRC32('mysql');
    -> 2501908538
```

- DEGREES(X)

  ラジアンからディグリーに変換された引数 X を返します。

```
mysql> SELECT DEGREES(PI());
    -> 180
mysql> SELECT DEGREES(PI() / 2);
    -> 90
```

- EXP(X)

e (自然対数の底) の X 乗の値を返します。この関数の逆は、(単一の引数のみを使用する) LOG() または LN() です。

```
mysql> SELECT EXP(2);
    -> 7.3890560989307
mysql> SELECT EXP(-2);
    -> 0.13533528323661
mysql> SELECT EXP(0);
    -> 1
```

- FLOOR(X)

  X 以下で最大の整数値を返します。

```
mysql> SELECT FLOOR(1.23);
    -> 1
mysql> SELECT FLOOR(-1.23);
    -> -2
```

  引数が厳密値数値の場合は、戻り値の型も厳密値数値になります。引数が文字列または浮動小数点の場合は、戻り値の型が浮動小数点になります。

- FORMAT(X,D)

  数値 X を '#,###,###.##' のような書式に変換し、小数点第 D 位に丸めて、その結果を文字列として返します。詳細は、セクション12.5「文字列関数」を参照してください。

- HEX(N_or_S)

  この関数を使用すると、10 進数または文字列の 16 進表現を取得できます。その方法は、引数の型によって異なります。詳細は、セクション12.5「文字列関数」で、この関数の説明を参照してください。

- LN(X)

  X の自然対数 (つまり、X の底 e の対数) を返します。X が 0 以下である場合は、NULL が返されます。

```
mysql> SELECT LN(2);
    -> 0.69314718055995
mysql> SELECT LN(-2);
    -> NULL
```

  この関数は LOG(X) のシノニムです。この関数の逆は、EXP() 関数です。

- LOG(X), LOG(B,X)

  1 つのパラメータで呼び出される場合、この関数は X の自然対数を返します。X が 0 以下である場合は、NULL が返されます。

  この関数 (単一の引数で呼び出された場合) の逆は、EXP() 関数です。

```
mysql> SELECT LOG(2);
    -> 0.69314718055995
mysql> SELECT LOG(-2);
    -> NULL
```

  この関数が 2 つのパラメータで呼び出される場合は、B を底とする X の対数が返されます。X が 0 以下である場合、または B が 1 以下である場合は、NULL が返されます。

```
mysql> SELECT LOG(2,65536);
    -> 16
mysql> SELECT LOG(10,100);
    -> 2
mysql> SELECT LOG(1,100);
    -> NULL
```

  LOG(B,X) は LOG(X) / LOG(B) と同等です。

- LOG2(X)

  X の底 2 の対数を返します。

```
mysql> SELECT LOG2(65536);
    -> 16
```

```
mysql> SELECT LOG2(-100);
    -> NULL
```

LOG2() は、格納に必要なビット数を調べる際に役立ちます。この関数は式 LOG(X) / LOG(2) と同等です。

- LOG10(X)

  X の底 10 の対数を返します。

```
mysql> SELECT LOG10(2);
    -> 0.30102999566398
mysql> SELECT LOG10(100);
    -> 2
mysql> SELECT LOG10(-100);
    -> NULL
```

LOG10(X) は LOG(10,X) と同等です。

- MOD(N,M)、N % M、N MOD M

  モジュロ演算。M で除算された N の余りを返します。

```
mysql> SELECT MOD(234, 10);
    -> 4
mysql> SELECT 253 % 7;
    -> 1
mysql> SELECT MOD(29,9);
    -> 2
mysql> SELECT 29 MOD 9;
    -> 2
```

この関数は、BIGINT 値でも安全に使用できます。

MOD() は、小数部を持つ値でも機能し、除算後の正確な余りを返します。

```
mysql> SELECT MOD(34.5,3);
    -> 1.5
```

MOD(N,0) は NULL を返します。

- PI()

  π (pi) の値を返します。表示されるデフォルトの小数点以下の桁数は 7 ですが、MySQL では内部的に全倍精度値が使用されます。

```
mysql> SELECT PI();
    -> 3.141593
mysql> SELECT PI()+0.000000000000000000;
    -> 3.141592653589793116
```

- POW(X,Y)

  X の Y 乗の値を返します。

```
mysql> SELECT POW(2,2);
    -> 4
mysql> SELECT POW(2,-2);
    -> 0.25
```

- POWER(X,Y)

  これは POW() のシノニムです。

- RADIANS(X)

  ディグリーからラジアンに変換された引数 X を返します。

  > **注記**
  >
  > π ラジアンは、180 ディグリーと同等です。

```
mysql> SELECT RADIANS(90);
    -> 1.5707963267949
```

- RAND(), RAND(N)

0 <= v < 1.0 の範囲内で、ランダムな浮動小数点値 v を返します。定数整数引数 N が指定されている場合は、カラム値の反復可能なシーケンスを生成するシード値として使用されます。次の例では、RAND(3) で生成される値のシーケンスが、発生した両方の場所で同じです。

```
mysql> CREATE TABLE t (i INT);
Query OK, 0 rows affected (0.42 sec)

mysql> INSERT INTO t VALUES(1),(2),(3);
Query OK, 3 rows affected (0.00 sec)
Records: 3  Duplicates: 0  Warnings: 0

mysql> SELECT i, RAND() FROM t;
+------+------------------+
| i    | RAND()           |
+------+------------------+
|    1 | 0.61914388706828 |
|    2 | 0.93845168309142 |
|    3 | 0.83482678498591 |
+------+------------------+
3 rows in set (0.00 sec)

mysql> SELECT i, RAND(3) FROM t;
+------+------------------+
| i    | RAND(3)          |
+------+------------------+
|    1 | 0.90576975597606 |
|    2 | 0.37307905813035 |
|    3 | 0.14808605345719 |
+------+------------------+
3 rows in set (0.00 sec)

mysql> SELECT i, RAND() FROM t;
+------+------------------+
| i    | RAND()           |
+------+------------------+
|    1 | 0.35877890638893 |
|    2 | 0.28941420772058 |
|    3 | 0.37073435016976 |
+------+------------------+
3 rows in set (0.00 sec)

mysql> SELECT i, RAND(3) FROM t;
+------+------------------+
| i    | RAND(3)          |
+------+------------------+
|    1 | 0.90576975597606 |
|    2 | 0.37307905813035 |
|    3 | 0.14808605345719 |
+------+------------------+
3 rows in set (0.01 sec)
```

定数イニシャライザを使用すると、シードは実行される前に、ステートメントのコンパイル時に一度初期化されます。引数として非定数イニシャライザ (カラム名など) が使用される場合は、RAND() が起動されるたび

に、その値でシードが初期化されます。(引数値が等しい場合は、RAND() で毎回同じ値が返されることを示しています。)

i <= R < j の範囲内でランダムな整数 R を取得するには、式 FLOOR(i + RAND() * (j − i)) を使用します。たとえば、7 <= R < 12 の範囲内でランダムな整数を取得するには、次のようなステートメントを使用します。

```
SELECT FLOOR(7 + (RAND() * 5));
```

WHERE 句内の RAND() は、WHERE が実行されるたびに再評価されます。

ORDER BY ではカラムが複数回評価されるため、ORDER BY 句内では RAND() 値を持つカラムを使用できません。ただし、次のようにランダムな順序で行を取得できます。

```
mysql> SELECT * FROM tbl_name ORDER BY RAND();
```

LIMIT と組み合わせて ORDER BY RAND() を使用すれば、行のセットからランダムなサンプルを選択する際に役立ちます。

```
mysql> SELECT * FROM table1, table2 WHERE a=b AND c<d -> ORDER BY RAND() LIMIT 1000;
```

RAND() は、完全なランダムジェネレータとしては設計されていません。要求に応じてランダムな数字をすばやく生成する方法であり、同じ MySQL バージョンのプラットフォーム間で移植可能です。

この関数は、ステートメントベースのレプリケーションでは安全に使用できません。binlog_format が STATEMENT に設定されているときに、この関数を使用すると、警告のログが記録されます。(Bug #49222)

- ROUND(X), ROUND(X,D)

  引数 X を D 小数点に丸めます。丸めアルゴリズムは、X のデータ型に依存します。D が指定されていない場合は、デフォルトで 0 に設定されます。D を負の数に指定すると、値 X の小数点左側の D 桁をゼロにすることができます。

```
mysql> SELECT ROUND(-1.23);
       -> -1
mysql> SELECT ROUND(-1.58);
       -> -2
mysql> SELECT ROUND(1.58);
       -> 2
mysql> SELECT ROUND(1.298, 1);
       -> 1.3
mysql> SELECT ROUND(1.298, 0);
       -> 1
mysql> SELECT ROUND(23.298, -1);
       -> 20
```

  戻り型は、(整数、重複、または 10 進数と仮定すると) 1 番目の引数と同じ型です。つまり、引数が整数の場合は、結果が整数 (小数点なし) になります。

```
mysql> SELECT ROUND(150.000,2), ROUND(150,2);
+------------------+--------------+
| ROUND(150.000,2) | ROUND(150,2) |
+------------------+--------------+
|           150.00 |          150 |
+------------------+--------------+
```

  ROUND() では、第 1 引数の型に応じて次のルールが使用されます。

  - 真値の数字の場合、ROUND() では「四捨五入」または「切り捨て (切り上げ)」 ルールが使用されます。0.5 以上の小数部を持つ値は、正の場合は次の整数に切り上げられ、負の場合は次の整数に切り下げられます。(つまり、ゼロから遠い方に丸められます。)0.5 未満の小数部を持つ値は、正の場合は次の整数に切り下げられ、負の場合は次の整数に切り上げられます。
  - 近似値の数字の場合、結果は C ライブラリによって異なります。多くのシステムでは、これは、ROUND() で「偶数丸め」ルールが使用されることを意味します。任意の小数部を持つ値は、もっとも近い偶数の整数に丸められます。

  次の例では、正確な値の丸めと近似値の丸めの相違点を示します。

```
mysql> SELECT ROUND(2.5), ROUND(25E-1);
+------------+--------------+
| ROUND(2.5) | ROUND(25E-1) |
+------------+--------------+
| 3          |            2 |
```

```
+------------+-------------+
```

詳細は、セクション12.20「高精度計算」を参照してください。

- SIGN(X)

  X が負、ゼロ、または正のいずれであるのかに応じて、引数の符号を -1、0、または 1 として返します。

```
mysql> SELECT SIGN(-32);
       -> -1
mysql> SELECT SIGN(0);
       -> 0
mysql> SELECT SIGN(234);
       -> 1
```

- SIN(X)

  X のサインを返します。X はラジアンで指定されます。

```
mysql> SELECT SIN(PI());
       -> 1.2246063538224e-16
mysql> SELECT ROUND(SIN(PI()));
       -> 0
```

- SQRT(X)

  負ではない数字 X の平方根を返します。

```
mysql> SELECT SQRT(4);
       -> 2
mysql> SELECT SQRT(20);
       -> 4.4721359549996
mysql> SELECT SQRT(-16);
       -> NULL
```

- TAN(X)

  X のタンジェントを返します。X はラジアンで指定されます。

```
mysql> SELECT TAN(PI());
       -> -1.2246063538224e-16
mysql> SELECT TAN(PI()+1);
       -> 1.5574077246549
```

- TRUNCATE(X,D)

  D 小数点に切り捨てて、数字 X を返します。D が 0 の場合は、結果に小数点または小数部が含まれません。D を負の数に指定すると、値 X の小数点左側の D 桁をゼロにすることができます。

```
mysql> SELECT TRUNCATE(1.223,1);
       -> 1.2
mysql> SELECT TRUNCATE(1.999,1);
       -> 1.9
mysql> SELECT TRUNCATE(1.999,0);
       -> 1
mysql> SELECT TRUNCATE(-1.999,1);
       -> -1.9
mysql> SELECT TRUNCATE(122,-2);
      -> 100
mysql> SELECT TRUNCATE(10.28*100,0);
      -> 1028
```

すべての数字は、ゼロ方向に丸められます。

12.7 日付および時間関数

このセクションでは、時間値の処理に使用できる関数について説明します。各日付日時型が持つ値の範囲、および値を指定する際の有効な書式については、セクション11.3「日付と時間型」を参照してください。

表 12.13 日付/時間関数

名前	説明
ADDDATE()	日付値に時間値 (間隔) を加算します
ADDTIME()	時間を加算します

名前	説明
CONVERT_TZ()	あるタイムゾーンから別のタイムゾーンに変換します
CURDATE()	現在の日付を返します
CURRENT_DATE(), CURRENT_DATE	CURDATE() のシノニムです
CURRENT_TIME(), CURRENT_TIME	CURTIME() のシノニムです
CURRENT_TIMESTAMP(), CURRENT_TIMESTAMP	NOW() のシノニムです
CURTIME()	現在の時間を返します
DATE()	日付または日付時間式の日付部分を抽出します
DATE_ADD()	日付値に時間値 (間隔) を加算します
DATE_FORMAT()	日付を指定された書式に設定します
DATE_SUB()	日付から時間値 (間隔) を引きます
DATEDIFF()	2 つの日付の差を求めます
DAY()	DAYOFMONTH() のシノニムです
DAYNAME()	曜日の名前を返します
DAYOFMONTH()	月の日を返します (0 - 31)
DAYOFWEEK()	引数の曜日インデックスを返します
DAYOFYEAR()	年の日を返します (1 - 366)
EXTRACT()	日付の一部を抽出します
FROM_DAYS()	日数を日付に変換します
FROM_UNIXTIME()	UNIX タイムスタンプを日付として書式設定します
GET_FORMAT()	日付書式文字列を返します
HOUR()	時を抽出します
LAST_DAY	引数の月の最終日を返します
LOCALTIME(), LOCALTIME	NOW() のシノニムです
LOCALTIMESTAMP, LOCALTIMESTAMP()	NOW() のシノニムです
MAKEDATE()	年と年間通算日から日付を作成します
MAKETIME()	時、分、秒から時間を作成します
MICROSECOND()	引数からマイクロ秒を返します
MINUTE()	引数から分を返します
MONTH()	渡された日付から月を返します
MONTHNAME()	月の名前を返します
NOW()	現在の日付と時間を返します
PERIOD_ADD()	年月に期間を加算します
PERIOD_DIFF()	期間内の月数を返します
QUARTER()	日付引数から四半期を返します
SEC_TO_TIME()	秒を「HH:MM:SS」形式に変換します
SECOND()	秒 (0-59) を返します
STR_TO_DATE()	文字列を日付に変換します
SUBDATE()	3 つの引数で呼び出されるときは DATE_SUB() のシノニムです
SUBTIME()	時間の差を求めます
SYSDATE()	この関数が実行される時間を返します
TIME()	渡された式の時部分を抽出します
TIME_FORMAT()	時間として書式設定します
TIME_TO_SEC()	秒に変換された引数を返します

名前	説明
TIMEDIFF()	時間の差を求めます
TIMESTAMP()	引数が 1 つの場合、この関数は日付または日付時間式を返します。引数が 2 つの場合、引数の合計を返します
TIMESTAMPADD()	日付時間式に間隔を加算します
TIMESTAMPDIFF()	日付時間式から間隔を減算します
TO_DAYS()	日に変換された日付引数を返します
TO_SECONDS()	0 年以降の秒数に変換された日付または日付時間引数を返します
UNIX_TIMESTAMP()	UNIX タイムスタンプを返します
UTC_DATE()	現在の UTC 日付を返します
UTC_TIME()	現在の UTC 時間を返します
UTC_TIMESTAMP()	現在の UTC 日付と時間を返します
WEEK()	週番号を返します
WEEKDAY()	曜日インデックスを返します
WEEKOFYEAR()	日付の暦週を返します (0 - 53)
YEAR()	年を返します
YEARWEEK()	年と週を返します

次に、日付関数の使用例を示します。次のクエリーは、過去 30 日以内の date_col 値を含むすべての行を選択します。

```
mysql> SELECT something FROM tbl_name
    -> WHERE DATE_SUB(CURDATE(),INTERVAL 30 DAY) <= date_col;
```

このクエリーは、将来の日付を持つ行も選択します。

通常、日付値が要求される関数では、日付時間値が受け入れられ、時間の部分は無視されます。通常、時間値が要求される関数では、日付時間値が受け入れられ、日付の部分は無視されます。

現在の日付または時間をそれぞれ返す関数は、クエリー実行の開始時にクエリーごとに 1 回だけ評価されます。つまり、NOW() などの関数が単一クエリー内で複数回参照されても、常に同じ結果が生成されます。(設計上、単一クエリーにはストアドプログラム (ストアドルーチン、トリガー、またはイベント) の呼び出し、およびそのプログラムによって呼び出されるすべてのサブプログラムも含まれています。)この原則は、CURDATE()、CURTIME()、UTC_DATE()、UTC_TIME()、UTC_TIMESTAMP()、およびそれらのシノニムにも適用されます。

CURRENT_TIMESTAMP()、CURRENT_TIME()、CURRENT_DATE()、および FROM_UNIXTIME() 関数は、time_zone システム環境変数の値として使用できる接続の現在のタイムゾーンで値を返します。さらに、UNIX_TIMESTAMP() では、その引数が現在のタイムゾーンでの日付時間値であるとみなされます。セクション10.6「MySQL Server でのタイムゾーンのサポート」を参照してください。

「zero」 日付または '2001-11-00' のような不完全な日付とともに使用できる日付関数もありますが、使用できない日付関数もあります。通常、日付の一部を抽出する関数は不完全な日付でも正しく機能するため、ゼロ以外の値が要求される場合に 0 を返すことができます。例:

```
mysql> SELECT DAYOFMONTH('2001-11-00'), MONTH('2005-00-00');
    -> 0, 0
```

その他の関数では完全な日付が要求され、日付が不完全な場合は NULL が返されます。これらには、日付演算を実行する関数や日付の一部を名前にマップする関数が含まれます。例:

```
mysql> SELECT DATE_ADD('2006-05-00',INTERVAL 1 DAY);
    -> NULL
mysql> SELECT DAYNAME('2006-05-00');
    -> NULL
```

MySQL 5.6.4 の時点では、いくつかの関数は引数として DATE() 関数の値を渡す際により厳密になったため、ゼロの日付部分を持つ不完全な日付は拒否されます。CONVERT_TZ()、DATE_ADD()、DATE_SUB()、DAYOFYEAR()、LAST_DAY()、TIMESTAMPDIFF()、TO_DAYS()、T
関数が影響を受けます。5.6.5 の LAST_DAY() では、この制限は緩やかで、ゼロの日付部分は許可されていました。

MySQL 5.6.4 以上では、マイクロ秒までの精度を持つ小数秒が TIME、DATETIME、および TIMESTAMP 値でサポートされています。時間引数を取る関数は、小数秒を含む値を受け入れます。時間関数からの戻り値には、必要に応じて小数秒が含まれます。

- ADDDATE(date,INTERVAL expr unit)、ADDDATE(expr,days)

  INTERVAL 形式の 2 番目の引数を付けて呼び出されると、ADDDATE() は DATE_ADD() のシノニムになります。関連する関数 SUBDATE() は DATE_SUB() のシノニムです。INTERVAL unit 引数については、DATE_ADD() の説明を参照してください。

```
mysql> SELECT DATE_ADD('2008-01-02', INTERVAL 31 DAY);
    -> '2008-02-02'
mysql> SELECT ADDDATE('2008-01-02', INTERVAL 31 DAY);
    -> '2008-02-02'
```

  days 形式の 2 番目の引数を付けて呼び出されると、MySQL では expr に加算される整数の日数として処理されます。

```
mysql> SELECT ADDDATE('2008-01-02', 31);
    -> '2008-02-02'
```

- ADDTIME(expr1,expr2)

  ADDTIME() は expr2 と expr1 を加算し、その結果を返します。expr1 は時間または日付時間式であり、expr2 は時間式です。

```
mysql> SELECT ADDTIME('2007-12-31 23:59:59.999999', '1 1:1:1.000002');
    -> '2008-01-02 01:01:01.000001'
mysql> SELECT ADDTIME('01:00:00.999999', '02:00:00.999998');
    -> '03:00:01.999997'
```

- CONVERT_TZ(dt,from_tz,to_tz)

  CONVERT_TZ() は、日付時間値 dt を from_tz で指定されたタイムゾーンから、to_tz で指定されたタイムゾーンに変換し、結果の値を返します。タイムゾーンは、セクション10.6「MySQL Server でのタイムゾーンのサポート」で説明されているように指定されます。引数が無効な場合、この関数は NULL を返します。

  from_tz から UTC に変換される際に、値が TIMESTAMP でサポートされている範囲から外れている場合は、変換が実行されません。TIMESTAMP の範囲については、セクション11.1.2「日付と時間型の概要」で説明されています。

```
mysql> SELECT CONVERT_TZ('2004-01-01 12:00:00','GMT','MET');
    -> '2004-01-01 13:00:00'
mysql> SELECT CONVERT_TZ('2004-01-01 12:00:00','+00:00','+10:00');
    -> '2004-01-01 22:00:00'
```

  > **注記**
  >
  > 名前付きタイムゾーン ('MET' または 'Europe/Moscow' など) を使用するには、タイムゾーンテーブルが正しく設定されている必要があります。手順については、セクション10.6「MySQL Server でのタイムゾーンのサポート」を参照してください。

- CURDATE()

  関数が文字列と数値コンテキストのどちらで使用されているのかに応じて、現在の日付を 'YYYY-MM-DD' または YYYYMMDD 書式の値として返します。

```
mysql> SELECT CURDATE();
    -> '2008-06-13'
mysql> SELECT CURDATE() + 0;
    -> 20080613
```

- CURRENT_DATE、CURRENT_DATE()

  CURRENT_DATE および CURRENT_DATE() は CURDATE() のシノニムです。

- CURRENT_TIME、CURRENT_TIME([fsp])

  CURRENT_TIME および CURRENT_TIME() は CURTIME() のシノニムです。

- CURRENT_TIMESTAMP、CURRENT_TIMESTAMP([fsp])

  CURRENT_TIMESTAMP および CURRENT_TIMESTAMP() は NOW() のシノニムです。

- CURTIME([fsp])

  関数が文字列と数値コンテキストのどちらで使用されているのかに応じて、現在の時間を 'HH:MM:SS' または HHMMSS 書式の値で返します。値は、現在のタイムゾーンで表現されています。

  MySQL 5.6.4 の時点では、0 から 6 までの小数秒の精度を指定するために fsp 引数が指定されている場合は、その桁数の小数秒部分が戻り値に含まれます。5.6.4 よりも前では、すべての引数が無視されます。

```
mysql> SELECT CURTIME();
    -> '23:50:26'
mysql> SELECT CURTIME() + 0;
    -> 235026.000000
```

- DATE(expr)

  日付または日付時間式 expr の日付部分を抽出します。

```
mysql> SELECT DATE('2003-12-31 01:02:03');
    -> '2003-12-31'
```

- DATEDIFF(expr1,expr2)

  DATEDIFF() は、ある日付から別の日付までの日数の値として表現された expr1 − expr2 を返します。expr1 および expr2 は、日付または日付時間式です。値の日付部分のみが計算に使用されます。

```
mysql> SELECT DATEDIFF('2007-12-31 23:59:59','2007-12-30');
    -> 1
mysql> SELECT DATEDIFF('2010-11-30 23:59:59','2010-12-31');
    -> -31
```

- DATE_ADD(date,INTERVAL expr unit), DATE_SUB(date,INTERVAL expr unit)

  これらの関数は日付演算を実行します。date 引数は、開始日付値または開始日付時間値を指定します。expr は、開始日付から加算または減算される間隔値を指定する式です。expr は文字列であり、負の間隔の場合は「-」 で始めることができます。unit は、式を解釈する際の単位を示すキーワードです。

  INTERVAL キーワードおよび unit 指定子では、大文字と小文字が区別されません。

  次の表には、unit 値ごとに要求される形式の expr 引数を表示します。

unit 値	要求される expr 書式
MICROSECOND	MICROSECONDS
SECOND	SECONDS
MINUTE	MINUTES
HOUR	HOURS
DAY	DAYS
WEEK	WEEKS
MONTH	MONTHS
QUARTER	QUARTERS
YEAR	YEARS
SECOND_MICROSECOND	'SECONDS.MICROSECONDS'
MINUTE_MICROSECOND	'MINUTES:SECONDS.MICROSECONDS'
MINUTE_SECOND	'MINUTES:SECONDS'
HOUR_MICROSECOND	'HOURS:MINUTES:SECONDS.MICROSECONDS'
HOUR_SECOND	'HOURS:MINUTES:SECONDS'
HOUR_MINUTE	'HOURS:MINUTES'
DAY_MICROSECOND	'DAYS HOURS:MINUTES:SECONDS.MICROSECONDS'
DAY_SECOND	'DAYS HOURS:MINUTES:SECONDS'
DAY_MINUTE	'DAYS HOURS:MINUTES'
DAY_HOUR	'DAYS HOURS'

unit 値	要求される expr 書式
YEAR_MONTH	'YEARS-MONTHS'

戻り値は引数によって異なります。

- 第 1 引数が DATETIME (または TIMESTAMP) 値である場合と、第 1 引数が DATE で、unit 値に HOURS、MINUTES、または SECONDS が使用されている場合は、DATETIME です。
- それ以外の場合は文字列です。

必ず結果が DATETIME になるようにするには、CAST() を使用すれば、第 1 引数を DATETIME に変換できます。

MySQL では、expr 書式の句読点区切り文字が許可されます。表には、提案される区切り文字を表示します。date 引数が DATE 値であり、計算に YEAR、MONTH、および DAY 部分のみが含まれる (つまり、時間部分は含まれない) 場合は、結果が DATE 値になります。その他の場合は、結果が DATETIME 値になります。

また、日付演算は、+ または - 演算子とともに INTERVAL を使用して実行することもできます。

```
date + INTERVAL expr unit
date - INTERVAL expr unit
```

INTERVAL expr unit は、他方の側の式が日付または日付間値である場合に、+ 演算子の一方の側で許可されます。- 演算子では、間隔から日付または日付間値を抽出しても意味がないため、INTERVAL expr unit は右側でのみ許可されます。

```
mysql> SELECT '2008-12-31 23:59:59' + INTERVAL 1 SECOND;
    -> '2009-01-01 00:00:00'
mysql> SELECT INTERVAL 1 DAY + '2008-12-31';
    -> '2009-01-01'
mysql> SELECT '2005-01-01' - INTERVAL 1 SECOND;
    -> '2004-12-31 23:59:59'
mysql> SELECT DATE_ADD('2000-12-31 23:59:59',
    ->              INTERVAL 1 SECOND);
    -> '2001-01-01 00:00:00'
mysql> SELECT DATE_ADD('2010-12-31 23:59:59',
    ->              INTERVAL 1 DAY);
    -> '2011-01-01 23:59:59'
mysql> SELECT DATE_ADD('2100-12-31 23:59:59',
    ->              INTERVAL '1:1' MINUTE_SECOND);
    -> '2101-01-01 00:01:00'
mysql> SELECT DATE_SUB('2005-01-01 00:00:00',
    ->              INTERVAL '1 1:1:1' DAY_SECOND);
    -> '2004-12-30 22:58:59'
mysql> SELECT DATE_ADD('1900-01-01 00:00:00',
    ->              INTERVAL '-1 10' DAY_HOUR);
    -> '1899-12-30 14:00:00'
mysql> SELECT DATE_SUB('1998-01-02', INTERVAL 31 DAY);
    -> '1997-12-02'
mysql> SELECT DATE_ADD('1992-12-31 23:59:59.000002',
    ->           INTERVAL '1.999999' SECOND_MICROSECOND);
    -> '1993-01-01 00:00:01.000001'
```

指定した間隔値 (unit キーワードから要求されるすべての間隔部分は含まれません) が短すぎる場合は、MySQL では間隔値の左端部分が省略されているとみなされます。たとえば、DAY_SECOND の unit を指定した場合は、expr の値には日、時間、分、秒の部分が含まれるとみなされます。'1:10' のような値を指定すると、MySQL では日と時間の部分が欠落していて、値は分と秒を表しているとみなされます。つまり、'1:10' DAY_SECOND は、'1:10' MINUTE_SECOND と同等の方法で解釈されます。これは、MySQL で TIME 値が時間ではなく経過時間を表していると解釈される方法に類似しています。

expr は文字列として処理されるため、INTERVAL に文字列以外の値を指定する場合は注意してください。たとえば、間隔指定子が HOUR_MINUTE の場合は、6/4 が 1.5000 に評価され、1 時間 5000 分として処理されます。

```
mysql> SELECT 6/4;
    -> 1.5000
mysql> SELECT DATE_ADD('2009-01-01', INTERVAL 6/4 HOUR_MINUTE);
```

```
    -> '2009-01-04 12:20:00'
```

間隔値が予想どおりに解釈されるようにするには、CAST() 演算を使用します。6/4 を 1 時間 5 分として処理するには、小数点以下の桁数が 1 桁の DECIMAL 値にキャストします。

```
mysql> SELECT CAST(6/4 AS DECIMAL(3,1));
    -> 1.5
mysql> SELECT DATE_ADD('1970-01-01 12:00:00',
    ->              INTERVAL CAST(6/4 AS DECIMAL(3,1)) HOUR_MINUTE);
    -> '1970-01-01 13:05:00'
```

時間部分が含まれるものを日付値に加算したり、日付値から減算したりすると、自動的に結果が日付時間値に変換されます。

```
mysql> SELECT DATE_ADD('2013-01-01', INTERVAL 1 DAY);
    -> '2013-01-02'
mysql> SELECT DATE_ADD('2013-01-01', INTERVAL 1 HOUR);
    -> '2013-01-01 01:00:00'
```

MONTH、YEAR_MONTH、または YEAR を加算した結果の日付に、新しい月の最大日数よりも大きな日が含まれる場合は、その日が新しい月の最大日数に調整されます。

```
mysql> SELECT DATE_ADD('2009-01-30', INTERVAL 1 MONTH);
    -> '2009-02-28'
```

日付算術演算では、完全な日付が必須であるため、'2006-07-00' のような不完全な日付や、誤った形式の日付では正常に機能しません。

```
mysql> SELECT DATE_ADD('2006-07-00', INTERVAL 1 DAY);
    -> NULL
mysql> SELECT '2005-03-32' + INTERVAL 1 MONTH;
    -> NULL
```

- DATE_FORMAT(date,format)

  format 文字列に従って、date 値を書式設定します。

  次の指定子が format 文字列で使用されている場合があります。書式指定子文字の前には、「%」 文字を付ける必要があります。

指定子	説明
%a	簡略曜日名 (Sun..Sat)
%b	簡略月名 (Jan..Dec)
%c	月、数字 (0..12)
%D	英語のサフィクスを持つ日付 (0th, 1st, 2nd, 3rd, …)
%d	日、数字 (00..31)
%e	日、数字 (0..31)
%f	マイクロ秒 (000000..999999)
%H	時間 (00..23)
%h	時間 (01..12)
%I	時間 (01..12)
%i	分、数字 (00..59)
%j	年間通算日 (001..366)
%k	時 (0..23)
%l	時 (1..12)
%M	月名 (January..December)
%m	月、数字 (00..12)
%p	AM または PM
%r	時間、12 時間単位 (hh:mm:ss に AM または PM が続く)
%S	秒 (00..59)
%s	秒 (00..59)

指定子	説明
%T	時間、24 時間単位 (hh:mm:ss)
%U	週 (00..53)、日曜日が週の初日、WEEK() モード 0
%u	週 (00..53)、月曜日が週の初日、WEEK() モード 1
%V	週 (01..53)、日曜日が週の初日、WEEK() モード 2、%X とともに使用
%v	週 (01..53)、月曜日が週の初日、WEEK() モード 3、%x とともに使用
%W	曜日名 (Sunday..Saturday)
%w	曜日 (0=Sunday..6=Saturday)
%X	年間の週、日曜日が週の初日、数字、4 桁、%V とともに使用
%x	年間の週、月曜日が週の初日、数字、4 桁、%v とともに使用
%Y	年、数字、4 桁
%y	年、数字 (2 桁)
%%	リテラル 「%」 文字
%x	x (上記にないすべての 「x」)

MySQL では '2014-00-00' などの不完全な日付の格納が許可されるため、月および日の指定子の範囲はゼロから始まります。

日および月の名前と略語に使用される言語は、lc_time_names システム変数 (セクション10.7「MySQL Server の口ケールサポート」) の値で制御されます。

%U、%u、%V、および %v 指定子のモード値については、WEEK() 関数の説明を参照してください。モードによって、週番号が付与される方法が影響を受けます。

DATE_FORMAT() は、ASCII 以外の文字を含む月および週の名前を返すことができるように、character_set_connection および collation_connection で指定された文字セットおよび照合順序を含む文字列を返します。

```
mysql> SELECT DATE_FORMAT('2009-10-04 22:23:00', '%W %M %Y');
    -> 'Sunday October 2009'
mysql> SELECT DATE_FORMAT('2007-10-04 22:23:00', '%H:%i:%s');
    -> '22:23:00'
mysql> SELECT DATE_FORMAT('1900-10-04 22:23:00',
  ->              '%D %y %a %d %m %b %j');
    -> '4th 00 Thu 04 10 Oct 277'
mysql> SELECT DATE_FORMAT('1997-10-04 22:23:00',
  ->              '%H %k %I %r %T %S %w');
    -> '22 22 10 10:23:00 PM 22:23:00 00 6'
mysql> SELECT DATE_FORMAT('1999-01-01', '%X %V');
    -> '1998 52'
mysql> SELECT DATE_FORMAT('2006-06-00', '%d');
    -> '00'
```

- DATE_SUB(date,INTERVAL expr unit)

  DATE_ADD() の説明を参照してください。

- DAY(date)

  DAY() は DAYOFMONTH() のシノニムです。

- DAYNAME(date)

  date に対応する曜日の名前を返します。名前に使用される言語は、lc_time_names システム変数 (セクション10.7「MySQL Server の口ケールサポート」) の値で制御されます。

```
mysql> SELECT DAYNAME('2007-02-03');
    -> 'Saturday'
```

- DAYOFMONTH(date)

  1 から 31 までの範囲内で date に対応する日を返します。'0000-00-00' や '2008-00-00' のように日の部分がゼロの場合は、0 を返します。

```
mysql> SELECT DAYOFMONTH('2007-02-03');
```

```
    -> 3
```

- DAYOFWEEK(date)

  date の曜日インデックス (1 = Sunday、2 = Monday、…、7 = Saturday) を返します。これらのインデックス値は、ODBC 標準に対応しています。

```
mysql> SELECT DAYOFWEEK('2007-02-03');
    -> 7
```

- DAYOFYEAR(date)

  1 から 366 までの範囲内で date に対応する通日を返します。

```
mysql> SELECT DAYOFYEAR('2007-02-03');
    -> 34
```

- EXTRACT(unit FROM date)

  EXTRACT() 関数では、DATE_ADD() または DATE_SUB() と同じ単位指定子が使用されますが、データ演算が実行されるのではなく、データから一部が抽出されます。

```
mysql> SELECT EXTRACT(YEAR FROM '2009-07-02');
    -> 2009
mysql> SELECT EXTRACT(YEAR_MONTH FROM '2009-07-02 01:02:03');
    -> 200907
mysql> SELECT EXTRACT(DAY_MINUTE FROM '2009-07-02 01:02:03');
    -> 20102
mysql> SELECT EXTRACT(MICROSECOND
  ->              FROM '2003-01-02 10:30:00.000123');
    -> 123
```

- FROM_DAYS(N)

  日数 N が指定され、DATE 値を返します。

```
mysql> SELECT FROM_DAYS(730669);
    -> '2007-07-03'
```

  古い日付では、FROM_DAYS() を慎重に使用してください。グレゴリオ暦 (1582) の出現よりも前の値とともに使用することを目的に設計されていません。セクション12.8「MySQL で使用されるカレンダー」を参照してください。

- FROM_UNIXTIME(unix_timestamp), FROM_UNIXTIME(unix_timestamp,format)

  関数が文字列と数値のどちらのコンテキストで使用されたのかに応じて、unix_timestamp 引数の表現を 'YYYY-MM-DD HH:MM:SS' または YYYYMMDDHHMMSS 書式の値として返します。値は、現在のタイムゾーンで表現されています。unix_timestamp は、UNIX_TIMESTAMP() 関数で生成されるような内部タイムスタンプ値です。

  format が指定されている場合は、DATE_FORMAT() 関数のエントリで一覧表示される場合と同じ方法で使用される format 文字列に従って、結果が書式設定されます。

```
mysql> SELECT FROM_UNIXTIME(1196440219);
    -> '2007-11-30 10:30:19'
mysql> SELECT FROM_UNIXTIME(1196440219) + 0;
    -> 20071130103019.000000
mysql> SELECT FROM_UNIXTIME(UNIX_TIMESTAMP(),
  ->              '%Y %D %M %h:%i:%s %x');
    -> '2007 30th November 10:30:59 2007'
```

  注: UNIX_TIMESTAMP() および FROM_UNIXTIME() を使用して TIMESTAMP 値と Unix タイムスタンプ値とを変換する場合は、どちらの方向でもマッピングが 1 対 1 ではないため、変換は不可逆です。詳細は UNIX_TIMESTAMP() 関数の説明を参照してください。

- GET_FORMAT({DATE|TIME|DATETIME}, {'EUR'|'USA'|'JIS'|'ISO'|'INTERNAL'})

  書式文字列を返します。この関数は、DATE_FORMAT() およびSTR_TO_DATE() 関数と組み合わせて使用すると便利です。

  1 番目と 2 番目の引数に値を指定できるため、複数の書式文字列を生成できます (使用される指定子については、DATE_FORMAT() 関数の説明で示す表を参照してください)。ISO 書式は ISO 8601 ではなく、ISO 9075 を参照しています。

関数呼び出し	結果
GET_FORMAT(DATE,'USA')	'%m.%d.%Y'
GET_FORMAT(DATE,'JIS')	'%Y-%m-%d'
GET_FORMAT(DATE,'ISO')	'%Y-%m-%d'
GET_FORMAT(DATE,'EUR')	'%d.%m.%Y'
GET_FORMAT(DATE,'INTERNAL')	'%Y%m%d'
GET_FORMAT(DATETIME,'USA')	'%Y-%m-%d %H.%i.%s'
GET_FORMAT(DATETIME,'JIS')	'%Y-%m-%d %H:%i:%s'
GET_FORMAT(DATETIME,'ISO')	'%Y-%m-%d %H:%i:%s'
GET_FORMAT(DATETIME,'EUR')	'%Y-%m-%d %H.%i.%s'
GET_FORMAT(DATETIME,'INTERNAL')	'%Y%m%d%H%i%s'
GET_FORMAT(TIME,'USA')	'%h:%i:%s %p'
GET_FORMAT(TIME,'JIS')	'%H:%i:%s'
GET_FORMAT(TIME,'ISO')	'%H:%i:%s'
GET_FORMAT(TIME,'EUR')	'%H.%i.%s'
GET_FORMAT(TIME,'INTERNAL')	'%H%i%s'

TIMESTAMP は、GET_FORMAT() への 1 番目の引数としても使用できます。その場合、関数は DATETIME の場合と同じ値を返します。

```
mysql> SELECT DATE_FORMAT('2003-10-03',GET_FORMAT(DATE,'EUR'));
       -> '03.10.2003'
mysql> SELECT STR_TO_DATE('10.31.2003',GET_FORMAT(DATE,'USA'));
       -> '2003-10-31'
```

- HOUR(time)

  time に対応する時を返します。戻り値の範囲は、日付時間値の 0 から 23 までです。ただし、TIME 値の範囲は実際にはもっと大きいため、HOUR は 23 よりも大きい値を返すことができます。

```
mysql> SELECT HOUR('10:05:03');
       -> 10
mysql> SELECT HOUR('272:59:59');
       -> 272
```

- LAST_DAY(date)

  日付または日付時間の値が指定され、月の最終日に対応する値を返します。引数が無効である場合は、NULL を返します。

```
mysql> SELECT LAST_DAY('2003-02-05');
       -> '2003-02-28'
mysql> SELECT LAST_DAY('2004-02-05');
       -> '2004-02-29'
mysql> SELECT LAST_DAY('2004-01-01 01:01:01');
       -> '2004-01-31'
mysql> SELECT LAST_DAY('2003-03-32');
       -> NULL
```

- LOCALTIME, LOCALTIME([fsp])

  LOCALTIME および LOCALTIME() は NOW() のシノニムです。

- LOCALTIMESTAMP, LOCALTIMESTAMP([fsp])

  LOCALTIMESTAMP および LOCALTIMESTAMP() は NOW() のシノニムです。

- MAKEDATE(year,dayofyear)

  指定された年と年間通算値から、日付を返します。dayofyear は 0 よりも大きくする必要があり、それ以外の場合は結果が NULL になります。

```
mysql> SELECT MAKEDATE(2011,31), MAKEDATE(2011,32);
       -> '2011-01-31', '2011-02-01'
```

```
mysql> SELECT MAKEDATE(2011,365), MAKEDATE(2014,365);
    -> '2011-12-31', '2014-12-31'
mysql> SELECT MAKEDATE(2011,0);
    -> NULL
```

- MAKETIME(hour,minute,second)

  hour、minute、および second 引数から計算された時間値を返します。

  MySQL 5.6.4 の時点では、second 引数に小数部を含めることができます。

```
mysql> SELECT MAKETIME(12,15,30);
    -> '12:15:30'
```

- MICROSECOND(expr)

  0 から 999999 までの範囲内の数値として、時間または日付時間式 expr からのマイクロ秒を返します。

```
mysql> SELECT MICROSECOND('12:00:00.123456');
    -> 123456
mysql> SELECT MICROSECOND('2009-12-31 23:59:59.000010');
    -> 10
```

- MINUTE(time)

  0 から 59 までの範囲内で、time に対応する分を返します。

```
mysql> SELECT MINUTE('2008-02-03 10:05:03');
    -> 5
```

- MONTH(date)

  1 (1 月) から 12 (12 月) の範囲内で、date に対応する月を返します。'0000-00-00' や '2008-00-00' のように月の部分がゼロの場合は、0 を返します。

```
mysql> SELECT MONTH('2008-02-03');
    -> 2
```

- MONTHNAME(date)

  date に対応する月の完全名を返します。名前に使用される言語は、lc_time_names システム変数 (セクション 10.7「MySQL Server のロケールサポート」) の値で制御されます。

```
mysql> SELECT MONTHNAME('2008-02-03');
    -> 'February'
```

- NOW([fsp])

  関数が文字列と数値のどちらのコンテキストで使用されているのかに応じて、現在の日付と時間を 'YYYY-MM-DD HH:MM:SS' または YYYYMMDDHHMMSS 書式の値として返します。値は、現在のタイムゾーンで表現されています。

  MySQL 5.6.4 の時点では、0 から 6 までの小数秒の精度を指定するために fsp 引数が指定されている場合は、その桁数の小数秒部分が戻り値に含まれます。5.6.4 よりも前では、すべての引数が無視されます。

```
mysql> SELECT NOW();
    -> '2007-12-15 23:50:26'
mysql> SELECT NOW() + 0;
    -> 20071215235026.000000
```

  NOW() は、ステートメントが実行を開始する時刻を示す定数時間を返します。(ストアドファンクションまたはトリガーでは、NOW() は関数またはトリガーステートメントが実行を開始する時間を返します。)これは、正確な実行時間を返す SYSDATE() の動作とは異なります。

```
mysql> SELECT NOW(), SLEEP(2), NOW();
+---------------------+----------+---------------------+
| NOW()               | SLEEP(2) | NOW()               |
+---------------------+----------+---------------------+
| 2006-04-12 13:47:36 |        0 | 2006-04-12 13:47:36 |
+---------------------+----------+---------------------+

mysql> SELECT SYSDATE(), SLEEP(2), SYSDATE();
+---------------------+----------+---------------------+
| SYSDATE()           | SLEEP(2) | SYSDATE()           |
+---------------------+----------+---------------------+
```

```
| 2006-04-12 13:47:44 |        0 | 2006-04-12 13:47:46 |
+---------------------+----------+---------------------+
```

さらに、SET TIMESTAMP ステートメントによって、NOW() で返された値は影響を受けますが、SYSDATE() で返された値は影響を受けません。つまり、バイナリログのタイムスタンプ設定は、SYSDATE() の呼び出しに影響しないことを意味します。タイムスタンプをゼロ以外の値に設定すると、後続の NOW() が起動されるたびに、その値が返されます。タイムスタンプをゼロに設定すると、この効果が取り消され、再度 NOW() が現在の日付と時間を返すようになります。

2 つの関数の相違点についての詳細は、SYSDATE() の説明を参照してください。

- PERIOD_ADD(P,N)

  N 月を期間 P に (YYMM または YYYYMM の書式で) 加算します。YYYYMM の書式で値を返します。

  > **注記**
  >
  > 期間引数 P は、日付値ではありません。

```
mysql> SELECT PERIOD_ADD(200801,2);
    -> 200803
```

- PERIOD_DIFF(P1,P2)

  期間 P1 と P2 間の月数を返します。P1 および P2 は、YYMM または YYYYMM の書式にする必要があります。期間引数 P1 および P2 は日付値ではないことに注意してください。

```
mysql> SELECT PERIOD_DIFF(200802,200703);
    -> 11
```

- QUARTER(date)

  1 から 4 までの範囲内で date に対応する四半期を返します。

```
mysql> SELECT QUARTER('2008-04-01');
    -> 2
```

- SECOND(time)

  0 から 59 までの範囲内で、time に対応する秒数を返します。

```
mysql> SELECT SECOND('10:05:03');
    -> 3
```

- SEC_TO_TIME(seconds)

  TIME 値として、時、分、秒に変換された seconds 引数を返します。結果の範囲は、TIME データ型の範囲に制約されます。引数がその範囲外の値に対応している場合は、警告が発行されます。

```
mysql> SELECT SEC_TO_TIME(2378);
    -> '00:39:38'
mysql> SELECT SEC_TO_TIME(2378) + 0;
    -> 3938
```

- STR_TO_DATE(str,format)

  これは DATE_FORMAT() 関数の逆です。文字列 str と書式文字列 format が指定されます。STR_TO_DATE() は、書式文字列に日付と時間の両方の部分が含まれる場合は DATETIME 値を返し、文字列に日付と時間の部分の一方のみが含まれる場合は DATE または TIME 値を返します。str から抽出された日付値、時間値、または日付時間値が不正な場合は、STR_TO_DATE() によって NULL が返され、警告が発行されます。

  サーバーは str をスキャンすることで、format の一致を試みます。書式文字列には、リテラル文字と % で始まる書式指定子を含めることができます。format 内のリテラル文字は、str 内と完全に一致する必要があります。format 内の書式指定子は、str 内の日付または時間の部分に一致する必要があります。format で使用できる指定子については、DATE_FORMAT() 関数の説明を参照してください。

```
mysql> SELECT STR_TO_DATE('01,5,2013','%d,%m,%Y');
    -> '2013-05-01'
mysql> SELECT STR_TO_DATE('May 1, 2013','%M %d,%Y');
```

```
    -> '2013-05-01'
```

str の先頭からスキャンが開始され、一致しない format が見つかった場合は失敗します。str の末尾にある余分な文字は、無視されます。

```
mysql> SELECT STR_TO_DATE('a09:30:17','a%h:%i:%s');
    -> '09:30:17'
mysql> SELECT STR_TO_DATE('a09:30:17','%h:%i:%s');
    -> NULL
mysql> SELECT STR_TO_DATE('09:30:17a','%h:%i:%s');
    -> '09:30:17'
```

指定されていない日付または時間の部分の値は 0 になるため、str に指定された値が不完全な場合は、結果の一部または全部が 0 に設定されます。

```
mysql> SELECT STR_TO_DATE('abc','abc');
    -> '0000-00-00'
mysql> SELECT STR_TO_DATE('9','%m');
    -> '0000-09-00'
mysql> SELECT STR_TO_DATE('9','%s');
    -> '00:00:09'
```

日付値の部分をチェックする範囲は、セクション11.3.1「DATE、DATETIME、および TIMESTAMP 型」の説明どおりです。たとえば、「ゼロ」の日付または部分値が 0 の日付は、このような値が許可されないように SQL モードが設定されていなければ、許可されます。

```
mysql> SELECT STR_TO_DATE('00/00/0000', '%m/%d/%Y');
    -> '0000-00-00'
mysql> SELECT STR_TO_DATE('04/31/2004', '%m/%d/%Y');
    -> '2004-04-31'
```

> **注記**
>
> 週が月の境界を越えた場合は、年と週の組み合わせでは年と月を一意に識別できないため、"%X%V" の書式を使用しても、年と週の文字列を日付に変換できません。年と週を日付に変換するには、曜日も指定する必要があります。
>
> ```
> mysql> SELECT STR_TO_DATE('200442 Monday', '%X%V %W');
>     -> '2004-10-18'
> ```

- SUBDATE(date,INTERVAL expr unit), SUBDATE(expr,days)

  INTERVAL 形式で 2 番目の引数を付けて呼び出されると、SUBDATE() は DATE_SUB() のシノニムになります。INTERVAL unit 引数については、DATE_ADD() の説明を参照してください。

```
mysql> SELECT DATE_SUB('2008-01-02', INTERVAL 31 DAY);
    -> '2007-12-02'
mysql> SELECT SUBDATE('2008-01-02', INTERVAL 31 DAY);
    -> '2007-12-02'
```

  2 番目の形式では、days の整数値を使用できます。このような場合は、日付または日付時間式 expr から日数が減算されると解釈されます。

```
mysql> SELECT SUBDATE('2008-01-02 12:00:00', 31);
    -> '2007-12-02 12:00:00'
```

- SUBTIME(expr1,expr2)

  SUBTIME() は、expr1 と同じ書式で表現される expr1 と expr2 を返します。expr1 は時間または日付時間式であり、expr2 は時間式です。

```
mysql> SELECT SUBTIME('2007-12-31 23:59:59.999999','1 1:1:1.000002');
    -> '2007-12-30 22:58:58.999997'
mysql> SELECT SUBTIME('01:00:00.999999', '02:00:00.999998');
    -> '-00:59:59.999999'
```

- SYSDATE([fsp])

  関数が文字列と数値のどちらのコンテキストで使用されているのかに応じて、現在の日付と時間を 'YYYY-MM-DD HH:MM:SS' または YYYYMMDDHHMMSS 書式の値として返します。

  MySQL 5.6.4 の時点では、0 から 6 までの小数秒の精度を指定するために fsp 引数が指定されている場合は、その桁数の小数秒部分が戻り値に含まれます。5.6.4 よりも前では、すべての引数が無視されます。

  SYSDATE() は、実行された時間を返します。これは、ステートメントが実行を開始する時間を示す定数時間を返す NOW() の動作とは異なります。(ストアドファンクションまたはトリガーでは、NOW() は関数またはトリガーステートメントが実行を開始する時間を返します。)

```
mysql> SELECT NOW(), SLEEP(2), NOW();
+---------------------+----------+---------------------+
| NOW()               | SLEEP(2) | NOW()               |
+---------------------+----------+---------------------+
| 2006-04-12 13:47:36 |        0 | 2006-04-12 13:47:36 |
+---------------------+----------+---------------------+

mysql> SELECT SYSDATE(), SLEEP(2), SYSDATE();
+---------------------+----------+---------------------+
| SYSDATE()           | SLEEP(2) | SYSDATE()           |
+---------------------+----------+---------------------+
| 2006-04-12 13:47:44 |        0 | 2006-04-12 13:47:46 |
+---------------------+----------+---------------------+
```

  さらに、SET TIMESTAMP ステートメントによって、NOW() で返された値は影響を受けますが、SYSDATE() で返された値は影響を受けません。つまり、バイナリログのタイムスタンプ設定は、SYSDATE() の呼び出しに影響しないことを意味します。

  SYSDATE() は、同じステートメント内でもさまざまな値を返すことができ、SET TIMESTAMP による影響も受けないため、非決定的です。そのため、ステートメントベースのバイナリロギングが使用されている場合は、レプリケーションで安全に使用できません。これが問題となる場合は、行ベースのロギングを使用できます。

  また、--sysdate-is-now オプションを使用すると、SYSDATE() を NOW() のエイリアスにすることができます。これは、オプションがマスターとスレーブの両方で使用される場合に機能します。

  SYSDATE() に非決定的な特性があるということは、それを参照する式を評価する際にインデックスを使用できないことも意味します。

- TIME(expr)

  時間または日付時間式 expr の時部分を抽出し、文字列として返します。

  この関数は、ステートメントベースのレプリケーションでは安全に使用できません。binlog_format が STATEMENT に設定されているときに、この関数を使用すると、警告のログが記録されます。

```
mysql> SELECT TIME('2003-12-31 01:02:03');
    -> '01:02:03'
mysql> SELECT TIME('2003-12-31 01:02:03.000123');
    -> '01:02:03.000123'
```

- TIMEDIFF(expr1,expr2)

  TIMEDIFF() は、時間値として表現された expr1 − expr2 を返します。expr1 および expr2 は時間または日付時間式ですが、両方とも同じ型にする必要があります。

  TIMEDIFF() で返される結果は、TIME 値で許可される範囲に制限されています。また、TIMESTAMPDIFF() および UNIX_TIMESTAMP() 関数のいずれかを使用することもできます。両方とも整数を返します。

```
mysql> SELECT TIMEDIFF('2000:01:01 00:00:00',
    ->              '2000:01:01 00:00:00.000001');
    -> '-00:00:00.000001'
mysql> SELECT TIMEDIFF('2008-12-31 23:59:59.000001',
    ->              '2008-12-30 01:01:01.000002');
    -> '46:58:57.999999'
```

- TIMESTAMP(expr), TIMESTAMP(expr1,expr2)

  引数を 1 つ付けると、この関数は日付または日付時間式 expr を日付時間値として返します。引数を 2 つ付けると、時間式 expr2 を日付または日付時間式 expr1 に加算し、その結果を日付時間値として返します。

```
mysql> SELECT TIMESTAMP('2003-12-31');
    -> '2003-12-31 00:00:00'
mysql> SELECT TIMESTAMP('2003-12-31 12:00:00','12:00:00');
    -> '2004-01-01 00:00:00'
```

- TIMESTAMPADD(unit,interval,datetime_expr)

  整数式 interval を日付または日付時間式 datetime_expr に加算します。interval の単位は、unit 引数で指定されます。この引数は、MICROSECOND (マイクロ秒)、SECOND、MINUTE、HOUR、DAY、WEEK、MONTH、QUARTER、YEAR 値のいずれかにする必要があります。

  unit 値を指定するには、ここで示したキーワードのいずれかを使用するか、SQL_TSI_ をプリフィクスとして付けます。たとえば、DAY と SQL_TSI_DAY は両方とも有効です。

```
mysql> SELECT TIMESTAMPADD(MINUTE,1,'2003-01-02');
    -> '2003-01-02 00:01:00'
mysql> SELECT TIMESTAMPADD(WEEK,1,'2003-01-02');
    -> '2003-01-09'
```

- TIMESTAMPDIFF(unit,datetime_expr1,datetime_expr2)

  datetime_expr2 − datetime_expr1 を返します。datetime_expr1 と datetime_expr2 は、日付または日付時間式です。式の一方が日付で、他方が日付時間にすることもできます。日付値は、必要に応じて時間部分が '00:00:00' の日付時間として処理されます。結果 (整数) の単位は、unit 引数で指定されます。unit の有効な値は、TIMESTAMPADD() 関数の説明で一覧表示された値と同じです。

```
mysql> SELECT TIMESTAMPDIFF(MONTH,'2003-02-01','2003-05-01');
    -> 3
mysql> SELECT TIMESTAMPDIFF(YEAR,'2002-05-01','2001-01-01');
    -> -1
mysql> SELECT TIMESTAMPDIFF(MINUTE,'2003-02-01','2003-05-01 12:05:55');
    -> 128885
```

> **注記**
>
> この関数の日付または日付時間引数の順序は、TIMESTAMP() 関数を 2 つの引数を指定して呼び出す場合の順序と逆になります。

- TIME_FORMAT(time,format)

  これは DATE_FORMAT() 関数と同様に使用されますが、format 文字列には時間、分、秒、マイクロ秒の書式指定子のみを含めることができます。その他の指定子では、NULL 値または 0 が生成されます。

  time 値に 23 よりも大きな時間部分が含まれる場合は、%H および %k 時間書式指定子によって、0..23 の通常の範囲よりも大きな値が生成されます。その他の時間書式指定子では、時間値モジュロ 12 が生成されます。

```
mysql> SELECT TIME_FORMAT('100:00:00', '%H %k %h %I %l');
    -> '100 100 04 04 4'
```

- TIME_TO_SEC(time)

  秒に変換された time 引数を返します。

```
mysql> SELECT TIME_TO_SEC('22:23:00');
    -> 80580
mysql> SELECT TIME_TO_SEC('00:39:38');
    -> 2378
```

- TO_DAYS(date)

  日付 date が指定され、日数 (0 年以降の日数) を返します。

```
mysql> SELECT TO_DAYS(950501);
    -> 728779
mysql> SELECT TO_DAYS('2007-10-07');
```

```
    -> 733321
```

TO_DAYS() は、カレンダーが変更された際に失われた日が考慮されないため、グレゴリオ暦 (1582) の出現よりも前の値とともに使用するために設計されていません。日付が 1582 よりも前の場合は (ほかの口ケールでは、さらにあとの年になる可能性があります)、この関数の結果は信頼できません。詳細は セクション12.8「MySQL で使用されるカレンダー」を参照してください。

MySQL では セクション11.3「日付と時間型」 のルールを使用して、日付の 2 桁の年の値が 4 桁の形式に変換されることを忘れないでください。たとえば、'2008-10-07' と '08-10-07' は同じ日付と認識されます。

```
mysql> SELECT TO_DAYS('2008-10-07'), TO_DAYS('08-10-07');
    -> 733687, 733687
```

MySQL では、ゼロの日付は '0000-00-00' として定義されます。ただし、このデータ自体は無効とみなされます。つまり、'0000-00-00' および '0000-01-01' の場合、TO_DAYS() は次に示す値を返します。

```
mysql> SELECT TO_DAYS('0000-00-00');
+-----------------------+
| to_days('0000-00-00') |
+-----------------------+
|                  NULL |
+-----------------------+
1 row in set, 1 warning (0.00 sec)

mysql> SHOW WARNINGS;
+---------+------+---------------------------------------+
| Level   | Code | Message                               |
+---------+------+---------------------------------------+
| Warning | 1292 | Incorrect datetime value: '0000-00-00' |
+---------+------+---------------------------------------+
1 row in set (0.00 sec)


mysql> SELECT TO_DAYS('0000-01-01');
+-----------------------+
| to_days('0000-01-01') |
+-----------------------+
|                     1 |
+-----------------------+
1 row in set (0.00 sec)
```

このことは、ALLOW_INVALID_DATES SQL サーバーモードが有効であるかどうかに関係なく当てはまります。

- TO_SECONDS(expr)

  日付または日付時間の expr が指定され、0 年以降の日数を返します。expr が有効な日付または日付時間の値ではない場合は、NULL を返します。

```
mysql> SELECT TO_SECONDS(950501);
    -> 62966505600
mysql> SELECT TO_SECONDS('2009-11-29');
    -> 63426672000
mysql> SELECT TO_SECONDS('2009-11-29 13:43:32');
    -> 63426721412
mysql> SELECT TO_SECONDS( NOW() );
    -> 63426721458
```

TO_DAYS() と同様に、TO_SECONDS() は、カレンダーが変更された際に失われた日が考慮されないため、グレゴリオ暦 (1582) の出現よりも前の値とともに使用する目的で設計されていません。日付が 1582 よりも前の場合は (ほかの口ケールでは、さらにあとの年になる可能性があります)、この関数の結果は信頼できません。詳細は セクション12.8「MySQL で使用されるカレンダー」を参照してください。

TO_DAYS() と同様に、TO_SECONDS() は セクション11.3「日付と時間型」 のルールを使用して日付の 2 桁の年の値を 4 桁の形式に変換します。

MySQL では、ゼロの日付は '0000-00-00' として定義されます。ただし、このデータ自体は無効とみなされます。つまり、'0000-00-00' および '0000-01-01' の場合、TO_SECONDS() は次に示す値を返します。

```
mysql> SELECT TO_SECONDS('0000-00-00');
+--------------------------+
| TO_SECONDS('0000-00-00') |
+--------------------------+
|                     NULL |
+--------------------------+
```

```
1 row in set, 1 warning (0.00 sec)

mysql> SHOW WARNINGS;
+---------+------+----------------------------------------+
| Level   | Code | Message                                |
+---------+------+----------------------------------------+
| Warning | 1292 | Incorrect datetime value: '0000-00-00' |
+---------+------+----------------------------------------+
1 row in set (0.00 sec)


mysql> SELECT TO_SECONDS('0000-01-01');
+--------------------------+
| TO_SECONDS('0000-01-01') |
+--------------------------+
|                    86400 |
+--------------------------+
1 row in set (0.00 sec)
```

このことは、ALLOW_INVALID_DATES SQL サーバーモードが有効であるかどうかに関係なく当てはまります。

- UNIX_TIMESTAMP()、UNIX_TIMESTAMP(date)

  引数なしで呼び出された場合は、Unix タイムスタンプ ('1970-01-01 00:00:00' UTC 以降の秒数) を符号なし整数として返します。date 引数を付けて UNIX_TIMESTAMP() が呼び出された場合は、その引数の値が '1970-01-01 00:00:00' UTC 以降の秒数として返されます。date には、DATE 文字列、DATETIME 文字列、TIMESTAMP、YYMMDD または YYYYMMDD 書式の数値を指定できます。サーバーは date を現在のタイムゾーンの値として解釈し、UTC の内部値に変換します。クライアントは、セクション10.6「MySQL Server でのタイムゾーンのサポート」で説明するとおりに、独自のタイムゾーンを設定できます。

```
mysql> SELECT UNIX_TIMESTAMP();
       -> 1196440210
mysql> SELECT UNIX_TIMESTAMP('2007-11-30 10:30:19');
       -> 1196440219
```

UNIX_TIMESTAMP() が TIMESTAMP カラムで使用されると、暗黙的に 「string-to-Unix-timestamp」 に変換されずに、内部タイムスタンプ値が直接返されます。UNIX_TIMESTAMP() に範囲外の日付を渡すと、0 が返されます。

注: UNIX_TIMESTAMP() および FROM_UNIXTIME() を使用して TIMESTAMP 値と Unix タイムスタンプ値とを変換する場合は、どちらの方向でもマッピングが 1 対 1 ではないため、変換は不可逆です。たとえば、ローカルタイムゾーンを変更するための変換が原因で、2 つの UNIX_TIMESTAMP() で 2 つの TIMESTAMP 値が同じ Unix タイムスタンプ値にマップされる可能性があります。FROM_UNIXTIME() は、その値を元の TIMESTAMP 値のいずれかにのみマップし直します。次に、CET タイムゾーンでの TIMESTAMP 値の使用例を示します。

```
mysql> SELECT UNIX_TIMESTAMP('2005-03-27 03:00:00');
+---------------------------------------+
| UNIX_TIMESTAMP('2005-03-27 03:00:00') |
+---------------------------------------+
|                            1111885200 |
+---------------------------------------+
mysql> SELECT UNIX_TIMESTAMP('2005-03-27 02:00:00');
+---------------------------------------+
| UNIX_TIMESTAMP('2005-03-27 02:00:00') |
+---------------------------------------+
|                            1111885200 |
+---------------------------------------+
mysql> SELECT FROM_UNIXTIME(1111885200);
+---------------------------+
| FROM_UNIXTIME(1111885200) |
+---------------------------+
| 2005-03-27 03:00:00       |
+---------------------------+
```

UNIX_TIMESTAMP() カラムを減算する場合は、結果を符号付きの整数にキャストします。セクション12.10「キャスト関数と演算子」を参照してください。

- UTC_DATE、UTC_DATE()

  関数が文字列と数値のどちらのコンテキストで使用されているのかに応じて、現在の UTC 日付を 'YYYY-MM-DD' または YYYYMMDD 書式の値で返します。

```
mysql> SELECT UTC_DATE(), UTC_DATE() + 0;
```

```
    -> '2003-08-14', 20030814
```

- UTC_TIME, UTC_TIME([fsp])

  関数が文字列と数値のどちらのコンテキストで使用されているのかに応じて、現在の UTC 時間を 'HH:MM:SS' または HHMMSS 書式の値で返します。

  MySQL 5.6.4 の時点では、0 から 6 までの小数秒の精度を指定するために fsp 引数が指定されている場合は、その桁数の小数秒部分が戻り値に含まれます。5.6.4 よりも前では、すべての引数が無視されます。

```
mysql> SELECT UTC_TIME(), UTC_TIME() + 0;
    -> '18:07:53', 180753.000000
```

- UTC_TIMESTAMP、UTC_TIMESTAMP([fsp])

  関数が文字列と数値のどちらのコンテキストで使用されているのかに応じて、現在の UTC 日付と時間を 'YYYY-MM-DD HH:MM:SS' または YYYYMMDDHHMMSS 書式の値として返します。

  MySQL 5.6.4 の時点では、0 から 6 までの小数秒の精度を指定するために fsp 引数が指定されている場合は、その桁数の小数秒部分が戻り値に含まれます。5.6.4 よりも前では、すべての引数が無視されます。

```
mysql> SELECT UTC_TIMESTAMP(), UTC_TIMESTAMP() + 0;
    -> '2003-08-14 18:08:04', 20030814180804.000000
```

- WEEK(date[,mode])

  この関数は、date に対応する週番号を返します。2 つの引数を取る形式の WEEK() を使用すると、週が日曜日と月曜日のどちらから始まるのか、および戻り値が 0 から 53までと 1 から 53 までのどちらの範囲内であるのかを指定できます。mode 引数が省略された場合は、default_week_format システム変数の値が使用されます。セクション5.1.4「サーバーシステム変数」を参照してください。

  次の表では、mode 引数がどのように機能するのかについて説明します。

モード	週の 1 日目	範囲	第 1 週は次の条件を満たす最初の週
0	日曜日	0-53	本年の日曜日を含む
1	月曜日	0-53	本年の 4 日以上を含む
2	日曜日	1-53	本年の日曜日を含む
3	月曜日	1-53	本年の 4 日以上を含む
4	日曜日	0-53	本年の 4 日以上を含む
5	月曜日	0-53	本年の月曜日を含む
6	日曜日	1-53	本年の 4 日以上を含む
7	月曜日	1-53	本年の月曜日を含む

  「本年の 4 日以上を含む」という意味を持つ mode 値では、ISO 8601:1988 に従って週番が付けられます。

  - 1 月 1 日を含む週に新年の 4 日以上が含まれる場合は、その週が第 1 週です。
  - それ以外の場合は、前年の最終週となり、次の週が第 1 週です。

```
mysql> SELECT WEEK('2008-02-20');
    -> 7
mysql> SELECT WEEK('2008-02-20',0);
    -> 7
mysql> SELECT WEEK('2008-02-20',1);
    -> 8
mysql> SELECT WEEK('2008-12-31',1);
    -> 53
```

  日付が前年の最終週に入っている場合は、オプションの mode 引数として2、3、6、または 7 を使用しなければ、MySQL によって 0 が返されます。

```
mysql> SELECT YEAR('2000-01-01'), WEEK('2000-01-01',0);
    -> 2000, 0
```

  指定された日付は実際には 1999 年の第 52 週に発生するため、WEEK() は 52 を返す必要があると議論されることもあります。代わりに WEEK() は、戻り値が「指定された年の週番号」となるように 0 を返します。これ

により、日付から日付部分を抽出するその他の関数と組み合わせると、WEEK() 関数を信頼して使用できるようになります。

指定された日付に対応する週の 1 日目を含む年について評価された結果を優先する場合は、オプションの mode 引数として 0、2、5、または 7 を使用します。

```
mysql> SELECT WEEK('2000-01-01',2);
    -> 52
```

または、YEARWEEK() 関数を使用します。

```
mysql> SELECT YEARWEEK('2000-01-01');
    -> 199952
mysql> SELECT MID(YEARWEEK('2000-01-01'),5,2);
    -> '52'
```

- WEEKDAY(date)

  date に対応する曜日インデックス (0 = Monday、1 = Tuesday、...6 = Sunday) を返します。

```
mysql> SELECT WEEKDAY('2008-02-03 22:23:00');
    -> 6
mysql> SELECT WEEKDAY('2007-11-06');
    -> 1
```

- WEEKOFYEAR(date)

  1 から 53 までの範囲内で、日付の暦週を返します。WEEKOFYEAR() は WEEK(date,3) に同等の互換性のある関数です。

```
mysql> SELECT WEEKOFYEAR('2008-02-20');
    -> 8
```

- YEAR(date)

  1000 から 9999 までの範囲内で、date に対応する年を返します。日付が「ゼロ」の場合は、0 を返します。

```
mysql> SELECT YEAR('1987-01-01');
    -> 1987
```

- YEARWEEK(date)、YEARWEEK(date,mode)

  日付に対応する年と週を返します。mode 引数は、WEEK() への mode 引数とまったく同様に機能します。結果の年と日付引数の年では、その年の最初と最後の週が異なる可能性があります。

```
mysql> SELECT YEARWEEK('1987-01-01');
    -> 198653
```

  WEEK() はその後、指定された年のコンテキストで週を返すため、週番号はオプションの引数 0 または 1 を付けた場合に WEEK() 関数で返される数字 (0) とは異なります。

12.8 MySQL で使用されるカレンダー

MySQL では、先発グレゴリオ暦と呼ばれるものが使用されています。

ユリウス暦からグレゴリオ暦に切り替えた国はすべて、切り替え時に少なくとも 10 日間を破棄する必要がありました。この動作を確認するために、最初にユリウス暦からグレゴリオ暦への切り替えが発生した 1582 年 10 月を考えてください。

月曜日	火曜日	水曜日	木曜日	金曜日	土曜日	日曜日
1	2	3	4	15	16	17
18	19	20	21	22	23	24
25	26	27	28	29	30	31

10 月 4 日から 10 月 15 日の間には日付がありません。この不連続性はカットオーバーと呼ばれます。カットオーバー前の日付はユリウス暦で、カットオーバー後の日付はグレゴリオ暦です。カットオーバー中の日付は存在しません。

実際に使用されていなかった日付に適用されるカレンダーは、先発と呼ばれます。したがって、カットオーバーが発生せず、常にグレゴリオ暦のルールで制御されていると考えられる場合は、先発グレゴリオ暦が使用され

ています。これが MySQL で使用されるものであり、標準 SQL でも必須です。そのため、MySQL DATE または DATETIME 値として格納されたカットオーバー前の日付は、その違いが補正されるように調整する必要があります。すべての国で同時にカットオーバーが発生しなかったこと、および発生が遅くなるほど失われる日数も多かったことに気付くことが重要です。たとえば、イギリスでは 1752 年に発生し、9 月 2 日水曜日の翌日が 9 月 14 日木曜日になりました。ロシアでは 1918 年までユリウス暦のままでしたが、その過程で 13 日間が失われました。「10 月革命」として知られる有名な事件は、グレゴリオ暦に従うと 11 月に発生しました。

12.9 全文検索関数

```
MATCH (col1,col2,...) AGAINST (expr [search_modifier])

search_modifier:
 {
     IN NATURAL LANGUAGE MODE
   | IN NATURAL LANGUAGE MODE WITH QUERY EXPANSION
   | IN BOOLEAN MODE
   | WITH QUERY EXPANSION
 }
```

MySQL では、次のような全文インデックス設定および検索がサポートされています。

- MySQL の全文インデックスは、型 FULLTEXT のインデックスです。
- 全文インデックスは、InnoDB または MyISAM テーブルでのみ使用でき、CHAR、VARCHAR、または TEXT カラムにのみ作成できます。
- FULLTEXT インデックスの定義は、テーブルの作成時に CREATE TABLE ステートメントで指定することも、あとで ALTER TABLE または CREATE INDEX を使用して追加することもできます。
- データセットが大きい場合は、FULLTEXT インデックスが付いていないテーブルにロードしてから、そのあとでインデックスを作成した方が、既存のFULLTEXT インデックスが付いているテーブルにロードするよりも断然速いです。

全文検索は、MATCH() ... AGAINST 構文を使用して実行されます。MATCH() には、検索対象のカラム名をカンマで区切ったリストを指定します。AGAINST には、検索する文字列と、実行する検索のタイプを示すオプションの修飾子を指定します。検索文字列は、クエリー評価時に定数である文字列値にする必要があります。たとえば、テーブルカラムは、行ごとに異なる可能性があるため除外されます。

全文検索には、次の 3 つの種類があります。

- 自然言語の検索では、検索文字列が人間の自然な言語でのフレーズ (フリーテキストのフレーズ) として解釈されます。特別な演算子はありません。ストップワードリストが適用されます。これらは、InnoDB 検索インデックスの場合は innodb_ft_enable_stopword、innodb_ft_server_stopword_table、および innodb_ft_user_stopword_table、MyISAM 検索インデックスの場合は ft_stopword_file によって制御されます。詳細は、セクション12.9.4「全文ストップワード」を参照してください。

  IN NATURAL LANGUAGE MODE 修飾子が指定されている場合または修飾子がまったく指定されていない場合は、全文検索が自然言語検索になります。詳細は、セクション12.9.1「自然言語全文検索」を参照してください。
- ブール検索では、特別なクエリー言語のルールを使用して検索文字列が解釈されます。文字列には、検索対象の単語が含まれます。また、一致する行に単語が存在しなければならない、または存在してはならないように、あるいは通常よりも単語の重みが高くまたは低くなるように、要件を指定する演算子を含めることもできます。特定の共通単語 (ストップワード) は、検索インデックスから省略され、検索文字列に存在しない場合は一致が行われません。IN BOOLEAN MODE 修飾子は、ブール検索を指定します。詳細は、セクション12.9.2「ブール全文検索」を参照してください。
- クエリー拡張検索は、自然言語検索を改善したものです。自然言語検索を実行する際は、検索文字列が使用されます。その後、検索で返されたもっとも関連性の高い行からの単語が検索文字列に追加され、再度検索が実行されます。クエリーでは、2 回目の検索からの行が返されます。IN NATURAL LANGUAGE MODE WITH QUERY EXPANSION または WITH QUERY EXPANSION 修飾子は、クエリー拡張検索を指定します。詳細は、セクション12.9.3「クエリー拡張を使用した全文検索」を参照してください。

FULLTEXT クエリーのパフォーマンスについては、セクション8.3.4「カラムインデックス」を参照してください。

InnoDB FULLTEXT インデックスの処理に関するより技術的な詳細は、セクション14.2.13.3「FULLTEXT インデックス」を参照してください。

全文検索上の制約については、セクション12.9.5「全文制限」に一覧表示されています。

myisam_ftdump ユーティリティーは、MyISAM 全文インデックスの内容をダンプします。これは、全文クエリーのデバッグ時に役立つことがあります。セクション4.6.2「myisam_ftdump — 全文インデックス情報の表示」を参照してください。

12.9.1 自然言語全文検索

デフォルトの場合や IN NATURAL LANGUAGE MODE 修飾子が指定された場合は、MATCH() 関数は、テキストコレクションに対して文字列の自然言語検索を実行します。コレクションは、FULLTEXT インデックスに含まれる 1 つ以上のカラムのセットです。検索文字列は、AGAINST() への引数として指定されます。MATCH() は、テーブルの行ごとに関連性の値を返します。つまり、検索文字列と、MATCH() リストで名前が指定されたカラムの該当行のテキスト間で類似性が評価されます。

```
mysql> CREATE TABLE articles (
    id INT UNSIGNED AUTO_INCREMENT NOT NULL PRIMARY KEY,
    title VARCHAR(200),
    body TEXT,
    FULLTEXT (title,body)
  ) ENGINE=InnoDB;
Query OK, 0 rows affected (0.00 sec)

mysql> INSERT INTO articles (title,body) VALUES
  ('MySQL Tutorial','DBMS stands for DataBase ...'),
  ('How To Use MySQL Well','After you went through a ...'),
  ('Optimizing MySQL','In this tutorial we will show ...'),
  ('1001 MySQL Tricks','1. Never run mysqld as root. 2. ...'),
  ('MySQL vs. YourSQL','In the following database comparison ...'),
  ('MySQL Security','When configured properly, MySQL ...');
Query OK, 6 rows affected (0.00 sec)
Records: 6  Duplicates: 0  Warnings: 0

mysql> SELECT * FROM articles
  WHERE MATCH (title,body)
  AGAINST ('database' IN NATURAL LANGUAGE MODE);
+----+-------------------+------------------------------------------+
| id | title             | body                                     |
+----+-------------------+------------------------------------------+
|  1 | MySQL Tutorial    | DBMS stands for DataBase ...             |
|  5 | MySQL vs. YourSQL | In the following database comparison ... |
+----+-------------------+------------------------------------------+
2 rows in set (0.00 sec)
```

デフォルトでは、大文字と小文字が区別される方法で検索が実行されます。大文字と大文字が区別された全文検索を実行するには、インデックス付きカラムのバイナリ照合順序を使用します。たとえば、全文検索で大文字と小文字が区別されるように、latin1 文字セットが使用されているカラムに latin1_bin の照合順序を割り当てることができます。

以前に例で示したように、MATCH() が WHERE 句で使用されると、もっとも関連性の高い行が 1 番目に返されるように、自動的にソートされます。関連性の値は、負ではない浮動小数点数です。ゼロの関連性は、類似性がないという意味です。関連性は、行内の単語の数、該当行内の一意な単語の数、コレクション内の単語の合計数、および特定の単語を含むドキュメント (行) の数に基づいて計算されます。

単に一致をカウントするには、次のようなクエリーを使用してください。

```
mysql> SELECT COUNT(*) FROM articles
  WHERE MATCH (title,body)
  AGAINST ('database' IN NATURAL LANGUAGE MODE);
+----------+
| COUNT(*) |
+----------+
|        2 |
+----------+
1 row in set (0.00 sec)
```

次のように、クエリーを再作成した方が早い場合もあります。

```
mysql> SELECT
  COUNT(IF(MATCH (title,body) AGAINST ('database' IN NATURAL LANGUAGE MODE), 1, NULL))
  AS count
  FROM articles;
+-------+
| count |
+-------+
|     2 |
+-------+
1 row in set (0.03 sec)
```

1 つ目のクエリーでは、いくつかの追加作業 (関連性別の結果のソート) が実行されますが、WHERE 句に基づくインデックス検索を使用することもできます。インデックス検索では、検索でほとんど行が一致しない場合に、最初のクエリーが速くなる可能性があります。2 つ目のクエリーではテーブルの完全スキャンが実行され、ほとんどの行に検索語句が存在した場合に、インデックス検索よりも高速になる可能性があります。

自然言語による全文検索では、MATCH() 関数で名前が指定されたカラムは、テーブル内の一部の FULLTEXT インデックスに含まれるカラムと同じである必要があります。上記のクエリーでは、MATCH() 関数で名前が指定されたカラム (title と body) は、article テーブルの FULLTEXT インデックスの定義で名前が指定されたカラムと同じです。title または body を個別に検索するには、カラムごとに個別の FULLTEXT インデックスを作成します。

ブール検索やクエリー拡張を使用した検索を実行することもできます。これらの検索タイプについては、セクション12.9.2「ブール全文検索」およびセクション12.9.3「クエリー拡張を使用した全文検索」で説明されています。

インデックスを使用した全文検索では、インデックスが複数のテーブルに及ぶ可能性はないため、MATCH() 句の単一テーブルにあるカラムにしか名前を付けることができません。MyISAM テーブルでは、インデックスが存在しない場合でも、ブール検索を実行できます (ただし、低速になります)。この場合、複数のテーブルのカラムに名前を指定できます。

上記の例では、関連性の降順で行が返される MATCH() 関数の使用方法について簡単に説明しました。次の例では、関連性の値を明示的に取得する方法を示します。SELECT ステートメントには WHERE 句も ORDER BY 句も含まれていないため、返される行は順序付けられません。

```
mysql> SELECT id, MATCH (title,body)
    AGAINST ('Tutorial' IN NATURAL LANGUAGE MODE) AS score
    FROM articles;
+----+--------------------+
| id | score              |
+----+--------------------+
|  1 | 0.22764469683170319 |
|  2 |                  0 |
|  3 | 0.22764469683170319 |
|  4 |                  0 |
|  5 |                  0 |
|  6 |                  0 |
+----+--------------------+
6 rows in set (0.00 sec)
```

次の例はさらに複雑です。このクエリーでは関連性の値が返され、関連性の降順での行のソートも行われます。この結果を実現するには、MATCH() を 2 回 (1 回は SELECT リストに、もう 1 回は WHERE 句に) 指定します。MySQL オプティマイザによって、2 回の MATCH() 呼び出しが同じであり、全文検索コードが 1 回のみ起動されることが検出されるため、追加のオーバーヘッドは発生しません。

```
mysql> SELECT id, body, MATCH (title,body) AGAINST
    ('Security implications of running MySQL as root'
    IN NATURAL LANGUAGE MODE) AS score
    FROM articles WHERE MATCH (title,body) AGAINST
    ('Security implications of running MySQL as root'
    IN NATURAL LANGUAGE MODE);
+----+-------------------------------------+-----------------+
| id | body                                | score           |
+----+-------------------------------------+-----------------+
|  4 | 1. Never run mysqld as root. 2. ... | 1.5219271183014 |
|  6 | When configured properly, MySQL ... | 1.3114095926285 |
+----+-------------------------------------+-----------------+
2 rows in set (0.00 sec)
```

MySQL FULLTEXT の実装では、トゥルーワード文字 (文字、数字、およびアンダースコア) のシーケンスが単語とみなされます。そのシーケンスには、アポストロフィー (「'」) も含めることはできますが、1 行に 1 つまでです。つまり、aaa'bbb は 1 語とみなされますが、aaa''bbb は 2 語とみなされます。単語の先頭または末尾のアポストロフィーは、FULLTEXT パーサーによって削除されます。'aaa'bbb' は、aaa'bbb として解析されます。

FULLTEXT パーサーは、特定の区切り文字 (「 」(空白)、「,」(カンマ)、「.」(ピリオド) など) を検索することで、単語の先頭と末尾を特定します。単語が区切り文字で区切られていない場合 (たとえば中国語の場合)、FULLTEXT パーサーでは単語の先頭と末尾を特定できません。このような言語で単語やその他のインデックス付きの語句を FULLTEXT インデックスに追加するには、「"」 などの任意の区切り文字で区切られるように前処理を行う必要があります。

MySQL 5.6 では、組み込みの全文パーサーから置き換えられるプラグインを作成できます (MyISAM でのみサポートされています)。詳細は、セクション24.2「MySQL プラグイン API」を参照してください。パーサープラグインのサンプルソースコードについては、MySQL のソース配布の plugin/fulltext ディレクトリを参照してください。

全文検索では、一部の単語が無視されます。

- 短すぎる単語は無視されます。全文検索で見つかった単語のデフォルトの最小長は、InnoDB 検索インデックスの場合は 3 文字、MyISAM の場合は 4 文字です。インデックスを作成する前に、構成オプション (InnoDB 検索インデックスの場合は innodb_ft_min_token_size 構成オプション、MyISAM の場合は ft_min_word_len) を設定すると、カットオフを制御できます。
- ストップワードリスト内の単語は無視されます。ストップワードは、セマンティクス値がゼロであると考えられるほど一般的な単語 (「the」 や 「some」 など) です。組み込みのストップワードリストもありますが、ユーザー定義のリストでオーバーライドできます。ストップワードリストおよび関連する構成オプションは、InnoDB 検索インデックスと MyISAM 検索インデックスとで異なります。ストップワードの処理は、InnoDB 検索インデックスの場合は構成オプション innodb_ft_enable_stopword、innodb_ft_server_stopword_table、および innodb_ft_user_stopword_table、MyISAM 検索インデックスの場合は ft_stopword_file によって制御されます。

デフォルトのストップワードリストおよびそれを変更する方法を表示する方法については、セクション12.9.4「全文ストップワード」を参照してください。単語のデフォルト最小長は、セクション12.9.6「MySQL の全文検索の微調整」で説明したように変更できます。

コレクションおよびクエリー内のすべての正確な単語は、コレクションまたはクエリーでの重要性に従って重み付けられます。したがって、多くのドキュメント内に存在する単語では、この特定のコレクション内のセマンティクス値が低くなるため、重みも低くなります。反対に、まれな単語には、高い重みが付けられます。単語の重みを組み合わせることで、行の関連性が計算されます。この技術は、大きなコレクションで最適に機能します。

> **MyISAM の制限**
>
> 非常に小さなテーブルでは、単語の配布が適切にセマンティクス値に反映されないため、このモデルでは、MyISAM テーブル上の検索インデックスに対して異常な結果が生成される可能性があります。たとえば、以前に示した articles テーブルのすべての行には、「MySQL」 という単語が存在しますが、MyISAM 検索インデックス内の単語を検索しても結果が生成されません。
>
> ```
> mysql> SELECT * FROM articles
>    WHERE MATCH (title,body)
>    AGAINST ('MySQL' IN NATURAL LANGUAGE MODE);
> Empty set (0.00 sec)
> ```
>
> 「MySQL」 という単語は 50% 以上の行に存在するため、検索の結果が空になります。そのため、事実上ストップワードとして処理されます。このフィルタ処理技術は、よくある語句に対して不適切な結果が生成される可能性のある小さなデータセットよりも、1G バイトのテーブルから 1 行おきに結果セットが返される可能性のない大きなデータセットに適しています。
>
> 全文検索がどのように動作するかを確認するために最初に試すと、しきい値が 50% であることに驚くかもしれませんが、これによって InnoDB テーブルが全文検索での実験によりふさわしいことがわかります。MyISAM テーブルを作成し、それに 1、2 行のテキストのみを挿入する場合は、テキスト内のすべての単語が 50% 以上の行に出現します。その結果、テーブルにより多くの行が含まれるまで、検索の結果が返されません。50% の制限を回避する必要のあるユーザーは、InnoDB テーブル上に検索インデックスを構築したり、セクション12.9.2「ブール全文検索」で説明したブール検索モードを使用したりできます。

12.9.2 ブール全文検索

MySQL では、IN BOOLEAN MODE 修飾子を使用することでブール全文検索を実行できます。この修飾子を使用すると、検索文字列の先頭または末尾にある特定の文字が特別な意味を持ちます。次のクエリーでは、+ および - 演算子は、一致が発生するために単語が存在しなければならないことと、単語が存在してはならないことをそれぞれ示します。したがって、このクエリーでは、「MySQL」 という単語は含まれるが、「YourSQL」 という単語は含まれないすべての行が取得されます。

```
mysql> SELECT * FROM articles WHERE MATCH (title,body)
   AGAINST ('+MySQL -YourSQL' IN BOOLEAN MODE);
+----+-----------------------+-------------------------------------+
| id | title                 | body                                |
+----+-----------------------+-------------------------------------+
|  1 | MySQL Tutorial        | DBMS stands for DataBase ...        |
|  2 | How To Use MySQL Well | After you went through a ...        |
|  3 | Optimizing MySQL      | In this tutorial we will show ...   |
```

```
|  4 | 1001 MySQL Tricks     | 1. Never run mysqld as root. 2. ... |
|  6 | MySQL Security        | When configured properly, MySQL ... |
+----+-----------------------+-------------------------------------+
```

> **注記**
>
> この機能を実装すると、MySQL では暗黙的ブール論理とも呼ばれる次のようなものが使用されます。
>
> - + は AND を表します
> - - は NOT を表します
> - [演算子なし] は暗黙的に OR を表します

ブール全文検索には、次のような特徴があります。

- 行は自動的に関連性の降順にソートされません。
- InnoDB テーブルでブールクエリーを実行するには、MATCH() 式のすべてのカラム上に FULLTEXT インデックスが必要です。MyISAM 検索インデックスに対するブールクエリーは、FULLTEXT インデックスなしでも機能します。ただし、この方法で実行される検索の速度は、非常に遅くなります。
- 単語の最小長および最大長の全文パラメータは、InnoDB 検索インデックスの場合は innodb_ft_min_token_size および innodb_ft_max_token_size、MyISAM 検索インデックスの場合は ft_min_word_len および ft_max_word_len に適用されます。
- ストップワードリストが適用されます。これらは、InnoDB 検索インデックスの場合は innodb_ft_enable_stopword、innodb_ft_server_stopword_table、および innodb_ft_user_stopword_table、MyISAM 検索インデックスの場合は ft_stopword_file によって制御されます。
- InnoDB の全文検索では、'++apple' の例と同様に、単一の検索単語で複数の演算子を使用するようサポートされていません。MyISAM の全文検索では、同じ検索が正常に処理され、検索単語に隣接する演算子を除くすべての演算子が無視されます。
- InnoDB の全文検索では、先頭のプラス記号またはマイナス記号のみがサポートされています。たとえば、InnoDB では '+apple' がサポートされますが、'apple+' はサポートされていません。末尾にプラス記号またはマイナス記号を指定すると、InnoDB で構文エラーがレポートされます。
- InnoDB の全文検索では、ワイルドカード ('+*') を使用した先頭のプラス記号、プラス記号とマイナス記号の組み合わせ ('+-')、または先頭のプラス記号とマイナス記号の組み合わせ ('+-apple') はサポートされていません。このような無効なクエリーでは、構文エラーが返されます。
- MyISAM 検索インデックスに適用される 50% のしきい値は使用されません。

ブール全文検索機能では、次の演算子がサポートされています。

- +

  先頭または末尾のプラス記号は、この単語が返される各行に存在しなければならないことを示します。InnoDB では、先頭のプラス記号のみがサポートされています。

- -

  先頭または末尾のマイナス記号は、この単語が返される行のいずれにも存在してはならないことを示します。InnoDB では、先頭のマイナス記号のみがサポートされています。

  注: - 演算子は、本来ならほかの検索語句で一致が行われる行を除外することのみに使用します。したがって、- の前にある検索語句のみを含むブールモードの検索では、空の結果が返されます。「除外された検索語句のいずれかを含む行を除いたすべての行」が返されるわけではありません。

- (演算子なし)

  デフォルトでは (+ と - のどちらも指定されてない場合)、この単語はオプションですが、それを含む行の評価は高くなります。これは、IN BOOLEAN MODE 修飾子なしの MATCH() ... AGAINST() の動作と似ています。

- @distance

  この演算子は、InnoDB テーブルでのみ機能します。2 つ以上の単語がすべて、相互に指定された距離内で始まっているかどうかが単語単位でテストされます。@distance 演算子の直前に、二重引用符で囲まれた文字

列内の検索単語を指定します (たとえば、MATCH(col1) AGAINST('"word1 word2 word3" @8' IN BOOLEAN MODE))。

- > <

  これらの 2 つの演算子は、行に割り当てられた関連性の値への単語の貢献度を変更する際に使用されます。> 演算子は貢献度を上げ、< 演算子は貢献度を下げます。次のリストのあとに示す例を参照してください。

- ( )

  丸括弧は、単語を部分式にグループ化します。丸括弧で囲まれたグループはネストできます。

- ~

  先頭のチルダは否定演算子として機能するため、行の関連性への単語の貢献度がマイナスになります。これは、「ノイズ」単語にマークを付ける際に便利です。このような単語を含む行は、その他よりも低く評価されますが、- 演算子を使用した場合のように、完全に除外されることはありません。

- *

  アスタリスクは、切り捨て (またはワイルドカード) 演算子として機能します。その他の演算子とは異なり、影響を受ける単語に追加されます。* 演算子の前の単語で始まれば、単語が一致します。

  切り捨て演算子を付けて単語が指定されている場合は、その単語が短すぎたり、ストップワードであったりしても、ブールクエリーから削除されません。単語が短すぎるかどうかは、InnoDB テーブルの場合は innodb_ft_min_token_size 設定、MyISAM テーブルの場合は ft_min_word_len によって判断されます。ワイルドカード単語は、1 つ以上の単語の先頭に存在しなければならないプリフィクスとみなされます。単語の最小長が 4 である場合は、'+word +the*' の検索では、2 番目のクエリーで短すぎる検索語句 the が無視されるため、'+word +the' の検索よりも少ない行が返される可能性があります。

- "

  二重引用符 (「"」) 文字内で囲まれたフレーズは、入力されたそのままのフレーズを含む行にのみ一致します。全文エンジンでは、フレーズが複数の単語に分割され、それらの単語の FULLTEXT インデックス内で検索が実行されます。単語以外の文字は、正確に一致する必要がありません。フレーズ検索では、そのフレーズとまったく同じ単語が同じ順序で一致に含まれることのみが必要です。たとえば、"test phrase" は "test, phrase" と一致します。

  フレーズにインデックス内にある単語が含まれない場合は、結果が空になります。単語がテキスト内に存在しない場合、ストップワードである場合、またはインデックス付きの単語の最小長よりも短い場合の組み合わせが原因で、単語がインデックス内に存在しない可能性があります。

次の例では、ブール全文演算子を使用する一部の検索文字列を実演します。

- 'apple banana'

  2 つの単語の 1 つ以上を含む行を検索します。

- '+apple +juice'

  両方の単語を含む行を検索します。

- '+apple macintosh'

  単語 「apple」 を含む行を検索しますが、行に 「macintosh」 も含まれる場合は行を高く評価されます。

- '+apple -macintosh'

  単語 「apple」 を含むが、「macintosh」 は含まない行を検索します。

- '+apple ~macintosh'

  単語 「apple」 を含む行を検索しますが、行に単語 「macintosh」 も含まれる場合は、行に含まれない場合よりも低く評価されます。これは、「macintosh」 が存在すると、行がまったく返されない '+apple -macintosh' の検索よりも「ソフト」です。

- '+apple +(>turnover <strudel)'

  単語 「apple」 と 「turnover」、または 「apple」 と 「strudel」 (順序は不問) を含む行を検索しますが、「apple turnover」 を 「apple strudel」 よりも高く評価します。

- 'apple*'

  「apple」、「apples」、「applesauce」、「applet」 などの単語を含む行を検索します。

- "'some words"'

  「some words」 とまったく同じフレーズを含む行を検索します (たとえば、「some words of wisdom」 を含む行は検索しますが、「some noise words」 は検索しません)。

  > **注記**
  >
  > フレーズを囲む 「"」 文字は、フレーズを区切る演算子文字です。検索文字列自体を囲む引用符ではありません。

InnoDB ブールモード検索の関連性ランキング

InnoDB の全文検索は、Sphinx の全文検索エンジンをモデルにし、使用されるアルゴリズムは、BM25 および TF-IDF のランキングアルゴリズムに基づいています。このような理由のため、InnoDB のブール全文検索の関連性ランキングは、MyISAM の関連性ランキングと異なる場合があります。

InnoDB では、「term frequency-inverse document frequency」 (TF-IDF) 重み付けシステムの偏差を使用して、指定された全文検索クエリーのドキュメントの関連性にランクが付けられます。TF-IDF の重み付けは、ドキュメントで単語が出現する頻度に基づき、コレクション内のすべてのドキュメントで単語が出現する頻度によってオフセットされます。言い換えると、ある単語がドキュメントで出現する頻度が高くなるほど、その単語がドキュメントコレクションで出現する頻度が低くなり、ドキュメントのランクが高くなります。

関連性ランキングの計算方法

単語の出現頻度 (TF) 値は、単語がドキュメントで出現する回数です。単語の逆文書頻度 (IDF) 値は、次の公式を使用して計算されます。ここで、total_records はコレクション内のレコード数、matching_records は検索語句が表示されるレコード数です。

```
${IDF} = log10( ${total_records} / ${matching_records} )
```

ドキュメントに単語が複数回含まれる場合は、IDF 値が TF 値で乗算されます。

```
${TF} * ${IDF}
```

TF および IDF 値を使用する場合は、ドキュメントの関連性ランキングが次の公式を使用して計算されます。

```
${rank} = ${TF} * ${IDF} * ${IDF}
```

公式については、次の例で実演されています。

単一単語検索の関連性ランキング

この例では、単一単語検索の関連性ランキングの計算を実演します。

```
mysql> CREATE TABLE articles (
id INT UNSIGNED AUTO_INCREMENT NOT NULL PRIMARY KEY,
title VARCHAR(200),
body TEXT,
FULLTEXT (title,body)
) ENGINE=InnoDB;
Query OK, 0 rows affected (1.04 sec)

mysql> INSERT INTO articles (title,body) VALUES
('MySQL Tutorial','This database tutorial ...'),
("How To Use MySQL",'After you went through a ...'),
('Optimizing Your Database','In this database tutorial ...'),
('MySQL vs. YourSQL','When comparing databases ...'),
('MySQL Security','When configured properly, MySQL ...'),
('Database, Database, Database','database database database'),
('1001 MySQL Tricks','1. Never run mysqld as root. 2. ...'),
('MySQL Full-Text Indexes', 'MySQL fulltext indexes use a ..');
Query OK, 8 rows affected (0.06 sec)
Records: 8  Duplicates: 0  Warnings: 0

mysql> SELECT id, title, body, MATCH (title,body)  AGAINST ('database' IN BOOLEAN MODE)
AS score FROM articles ORDER BY score DESC;
+----+-----------------------------+---------------------------------------+-------------------+
| id | title                       | body                                  | score             |
+----+-----------------------------+---------------------------------------+-------------------+
```

```
|  6 | Database, Database, Database | database database database         |  1.0886961221694946 |
|  3 | Optimizing Your Database     | In this database tutorial ...      | 0.36289870738983154 |
|  1 | MySQL Tutorial               | This database tutorial ...         | 0.18144935369491577 |
|  2 | How To Use MySQL             | After you went through a ...       |                   0 |
|  4 | MySQL vs. YourSQL            | When comparing databases ...       |                   0 |
|  5 | MySQL Security               | When configured properly, MySQL ... |                   0 |
|  7 | 1001 MySQL Tricks            | 1. Never run mysqld as root. 2. ... |                   0 |
|  8 | MySQL Full-Text Indexes      | MySQL fulltext indexes use a ..    |                   0 |
+----+------------------------------+------------------------------------+---------------------+
8 rows in set (0.00 sec)
```

合計で 8 つのレコードがあり、そのうち 3 つが 「database」 という検索語句に一致します。1 つ目のレコード (id 6) には、検索語句が 6 回含まれ、関連性ランキングは 1.0886961221694946 です。このランキング値は、6 の TF 値 (レコード id 6 には 「database」 という検索語句が 6 回出現します)、および次のように計算される 0.42596873216370745 の IDF 値 (ここで、8 はレコードの合計数、3 は検索語句が出現するレコードの数) を使用して計算されます。

```
${IDF} = log10( 8 / 3 ) = 0.42596873216370745
```

その後、TF および IDF 値はランキング公式に入力されます。

```
${rank} = ${TF} * ${IDF} * ${IDF}
```

MySQL コマンド行クライアントで計算を実行すると、1.088696164686938 のランキング値が返されます。

```
mysql> SELECT 6*log10(8/3)*log10(8/3);
+--------------------------+
| 6*log10(8/3)*log10(8/3) |
+--------------------------+
|       1.088696164686938 |
+--------------------------+
1 row in set (0.00 sec)
```

> **注記**
>
> SELECT ... MATCH ... AGAINST ステートメントと MySQL コマンド行クライアントで返されるランキング値 (1.0886961221694946 と 1.088696164686938) に、わずかな相違がある場合があります。この相違は、整数と浮動小数点/倍精度間のキャストが (関連する精度および丸めの決定とともに) InnoDB によって内部で実行される方法、およびその他の場所 (MySQL コマンド行クライアントやその他のタイプの計算機など) で実行される方法が原因で発生します。

複数単語検索の関連性ランキング

この例では、以前の例で使用された articles テーブルおよびデータに基づいて、複数単語の全文検索の関連性ランキングの計算を実演します。

複数の単語で検索する場合は、次の公式に示すように、関連性ランキングの値が各単語の関連性ランキングの合計になります。

```
${rank} = ${TF} * ${IDF} * ${IDF} + ${TF} * ${IDF} * ${IDF}
```

2 つの語句 ('mysql tutorial') で検索を実行すると、次の結果が返されます。

```
mysql> SELECT id, title, body, MATCH (title,body)  AGAINST ('mysql tutorial' IN BOOLEAN MODE)
   AS score FROM articles ORDER BY score DESC;
+----+------------------------------+------------------------------------+----------------------+
| id | title                        | body                               | score                |
+----+------------------------------+------------------------------------+----------------------+
|  1 | MySQL Tutorial               | This database tutorial ...         |   0.7405621409416199 |
|  3 | Optimizing Your Database     | In this database tutorial ...      |   0.3624762296676636 |
|  5 | MySQL Security               | When configured properly, MySQL ... | 0.031219376251101494 |
|  8 | MySQL Full-Text Indexes      | MySQL fulltext indexes use a ..    | 0.031219376251101494 |
|  2 | How To Use MySQL             | After you went through a ...       | 0.015609688125550747 |
|  4 | MySQL vs. YourSQL            | When comparing databases ...       | 0.015609688125550747 |
|  7 | 1001 MySQL Tricks            | 1. Never run mysqld as root. 2. ... | 0.015609688125550747 |
|  6 | Database, Database, Database | database database database         |                    0 |
+----+------------------------------+------------------------------------+----------------------+
8 rows in set (0.00 sec)
```

1 つ目のレコード (id 8) では、'mysql' が 1 回出現し、'tutorial' が 2 回出現します。'mysql' に一致するレコードは 6 つ、'tutorial' に一致するレコードは 2 つあります。MySQL コマンド行クライアントでは、これらの値を複数単語検索のランキング公式に挿入するときに、予期されるランキング値が返されます。

```
mysql> SELECT (1*log10(8/6)*log10(8/6)) + (2*log10(8/2)*log10(8/2));
```

```
+-----------------------------------------------------+
| (1*log10(8/6)*log10(8/6)) + (2*log10(8/2)*log10(8/2)) |
+-----------------------------------------------------+
|                                0.7405621541938003 |
+-----------------------------------------------------+
1 row in set (0.00 sec)
```

> **注記**
>
> 上記の例では、SELECT ... MATCH ... AGAINST ステートメントと MySQL コマンド行クライアントで返されるランキング値に、わずかな相違があることについて説明しました。

12.9.3 クエリー拡張を使用した全文検索

全文検索では、クエリー拡張 (特に、そのバリアントの 「ブラインドクエリー拡張」) がサポートされています。一般に、これは検索フレーズが短すぎるときに役立ちます。つまり、ユーザーは多くの場合暗黙的な知識に依存しますが、全文検索エンジンにはこれが不足していることを意味します。たとえば、ユーザーが 「database」 を検索することは、実際は 「MySQL」、「Oracle」、「DB2」、および 「RDBMS」 がすべて、「databases」 に一致して返されるはずのフレーズであることを意味する場合があります。これが暗黙的な知識です。

ブラインドクエリー拡張 (自動関連性フィードバックとも呼ばれる) は、検索フレーズのあとに WITH QUERY EXPANSION または IN NATURAL LANGUAGE MODE WITH QUERY EXPANSION を追加することで有効になります。これは、検索を 2 回実行することで機能します。2 回目の検索での検索フレーズは、1 回目の検索でのもっとも関連性の高い数個のドキュメントと連結されたオリジナルの検索フレーズです。したがって、これらのドキュメントのいずれかに単語 「databases」 および単語 「MySQL」 が含まれている場合は、単語 「database」 が含まれていなくても、2 回目の検索で単語 「MySQL」 を含むドキュメントが検索されます。次の例では、この相違点を示します。

```
mysql> SELECT * FROM articles
    WHERE MATCH (title,body)
    AGAINST ('database' IN NATURAL LANGUAGE MODE);
+----+-------------------+----------------------------------------+
| id | title             | body                                   |
+----+-------------------+----------------------------------------+
|  1 | MySQL Tutorial    | DBMS stands for DataBase ...           |
|  5 | MySQL vs. YourSQL | In the following database comparison ... |
+----+-------------------+----------------------------------------+
2 rows in set (0.00 sec)

mysql> SELECT * FROM articles
    WHERE MATCH (title,body)
    AGAINST ('database' WITH QUERY EXPANSION);
+----+-----------------------+----------------------------------------+
| id | title                 | body                                   |
+----+-----------------------+----------------------------------------+
|  5 | MySQL vs. YourSQL     | In the following database comparison ... |
|  1 | MySQL Tutorial        | DBMS stands for DataBase ...           |
|  3 | Optimizing MySQL      | In this tutorial we will show ...      |
|  6 | MySQL Security        | When configured properly, MySQL ...    |
|  2 | How To Use MySQL Well | After you went through a ...           |
|  4 | 1001 MySQL Tricks     | 1. Never run mysqld as root. 2. ...    |
+----+-----------------------+----------------------------------------+
6 rows in set (0.00 sec)
```

もう 1 つの例では、ユーザーが 「Maigret」 のスペルに自信がないときに、Georges Simenon 著の Maigret に関する書籍を検索しています。クエリー拡張を使用しなければ、「Megre and the reluctant witnesses」 を検索しても 「Maigret and the Reluctant Witnesses」 のみが検索されます。クエリー拡張を使用すれば、2 回目の検索で単語 「Maigret」 を含むすべての書籍が検索されます。

> **注記**
>
> ブラインドクエリー拡張には、関連性のないドキュメントが返されるとノイズが大幅に増加する傾向があるため、検索フレーズが短すぎる場合にのみ使用してください。

12.9.4 全文ストップワード

サーバー文字セットおよび照合順序 (character_set_server および collation_server システム変数の値) を使用すると、全文クエリー用のストップワードリストがロードおよび検索されます。全文インデックス作成または検索で使用されるストップワードファイルまたはカラムに、character_set_server または collation_server とは異なる文字セットまたは照合順序が含まれている場合は、ストップワード検索で誤ったヒットまたはミスが発生する可能性があります。

ストップワード検索で大文字と小文字が区別されるかどうかは、サーバー照合順序によって異なります。たとえば、照合順序が latin1_swedish_ci の場合は検索で大文字と小文字が区別されませんが、照合順序が latin1_general_cs または latin1_bin の場合は検索で大文字と小文字が区別されます。

InnoDB 検索インデックスのストップワード

技術的、文学的、およびその他のソースからのドキュメントでは、キーワードとしてまたは重要なフレーズで短い単語が使用されることが多いため、InnoDB ではデフォルトのストップワードリストが比較的短くなります。たとえば、「to be or not to be」 を検索し、これらの単語がすべて無視されるのではなく、適切な結果が取得されることを期待するとします。

デフォルトの InnoDB ストップワードリストを確認するには、INFORMATION_SCHEMA.INNODB_FT_DEFAULT_STOPWORD テーブルを問い合わせます。

```
mysql> SELECT * FROM INFORMATION_SCHEMA.INNODB_FT_DEFAULT_STOPWORD;
+-------+
| value |
+-------+
| a     |
| about |
| an    |
| are   |
| as    |
| at    |
| be    |
| by    |
| com   |
| de    |
| en    |
| for   |
| from  |
| how   |
| i     |
| in    |
| is    |
| it    |
| la    |
| of    |
| on    |
| or    |
| that  |
| the   |
| this  |
| to    |
| was   |
| what  |
| when  |
| where |
| who   |
| will  |
| with  |
| und   |
| the   |
| www   |
+-------+
36 rows in set (0.00 sec)
```

すべての InnoDB テーブルで独自のストップワードリストを定義するには、INNODB_FT_DEFAULT_STOPWORD テーブルと同じ構造を持つテーブルを定義し、それにストップワードを移入し、innodb_ft_server_stopword_table オプションの値を db_name/table_name 形式の値に設定してから、全文インデックスを作成します。ストップワードテーブルには、value という名前の単一の VARCHAR カラムが含まれている必要があります。次の例では、InnoDB 用に新しいグローバルストップワードテーブルを作成および構成するよう実演します。

```
-- Create a new stopword table

mysql> CREATE TABLE my_stopwords(value VARCHAR(30)) ENGINE = INNODB;
Query OK, 0 rows affected (0.01 sec)

-- Insert stopwords (for simplicity, a single stopword is used in this example)

mysql> INSERT INTO my_stopwords(value) VALUES ('Ishmael');
Query OK, 1 row affected (0.00 sec)

-- Create the table
```

```
mysql> CREATE TABLE opening_lines (
id INT UNSIGNED AUTO_INCREMENT NOT NULL PRIMARY KEY,
opening_line TEXT(500),
author VARCHAR(200),
title VARCHAR(200)
) ENGINE=InnoDB;
Query OK, 0 rows affected (0.01 sec)

-- Insert data into the table

mysql> INSERT INTO opening_lines(opening_line,author,title) VALUES
('Call me Ishmael.','Herman Melville','Moby-Dick'),
('A screaming comes across the sky.','Thomas Pynchon','Gravity\'s Rainbow'),
('I am an invisible man.','Ralph Ellison','Invisible Man'),
('Where now? Who now? When now?','Samuel Beckett','The Unnamable'),
('It was love at first sight.','Joseph Heller','Catch-22'),
('All this happened, more or less.','Kurt Vonnegut','Slaughterhouse-Five'),
('Mrs. Dalloway said she would buy the flowers herself.','Virginia Woolf','Mrs. Dalloway'),
('It was a pleasure to burn.','Ray Bradbury','Fahrenheit 451');
Query OK, 8 rows affected (0.00 sec)
Records: 8  Duplicates: 0  Warnings: 0

-- Set the innodb_ft_server_stopword_table option to the new stopword table

mysql> SET GLOBAL innodb_ft_server_stopword_table = 'test/my_stopwords';
Query OK, 0 rows affected (0.00 sec)

-- Create the full-text index (which rebuilds the table if no FTS_DOC_ID column is defined)

mysql> CREATE FULLTEXT INDEX idx ON opening_lines(opening_line);
Query OK, 0 rows affected, 1 warning (1.17 sec)
Records: 0  Duplicates: 0  Warnings: 1
```

INFORMATION_SCHEMA.INNODB_FT_INDEX_TABLE で単語を問い合わせて、指定したストップワード ('Ishmael') が表示されないことを確認します。

> **注記**
>
> デフォルトでは、長さが 3 文字よりも少ない単語または 84 文字よりも多い単語は、InnoDB の全文検索インデックスに表示されません。単語の最大長および最小長の値は、innodb_ft_max_token_size および innodb_ft_min_token_size 変数を使用して構成できます。

```
mysql> SET GLOBAL innodb_ft_aux_table='test/opening_lines';
Query OK, 0 rows affected (0.00 sec)

mysql> SELECT word FROM INFORMATION_SCHEMA.INNODB_FT_INDEX_TABLE LIMIT 15;
+-----------+
| word      |
+-----------+
| across    |
| all       |
| burn      |
| buy       |
| call      |
| comes     |
| dalloway  |
| first     |
| flowers   |
| happened  |
| herself   |
| invisible |
| less      |
| love      |
| man       |
+-----------+
15 rows in set (0.00 sec)
```

ストップワードリストをテーブルごとに作成するには、その他のストップワードテーブルを作成し、innodb_ft_user_stopword_table オプションを使用して使用されるストップワードテーブルを指定してから、全文インデックスを作成します。

MyISAM 検索インデックスのストップワード

MySQL 5.6 では、character_set_server が ucs2、utf16、utf16le、または utf32 の場合、ストップワードファイルが latin1 を使用してロードおよび検索されます。

MyISAM テーブル用のデフォルトのストップワードリストをオーバーライドするには、ft_stopword_file システム変数を設定します。(セクション5.1.4「サーバーシステム変数」を参照してください。)変数の値は、ストップワードリストを含むファイルのパス名、またはストップワードのフィルタ処理が無効になる空の文字列になるようにしてください。サーバーは、別のディレクトリを指定する絶対パス名が指定されないかぎり、データディレクトリ内のファイルを検索します。この変数の値またはストップワードファイルの内容を変更したら、サーバーを再起動し、FULLTEXT インデックスを再構築してください。

ストップワードリストは自由形式で、改行、空白、カンマなどの英数字以外の文字でストップワードが区切られます。例外として、下線文字 (「_」) と単一アポストロフィー (「'」) は単語の一部として処理されます。ストップワードリストの文字セットは、サーバーのデフォルト文字セットです。セクション10.1.3.1「サーバー文字セットおよび照合順序」を参照してください。

次の表には、MyISAM 検索インデックスのデフォルトのストップワードリストを示します。このリストは、MySQL ソース配布の storage/myisam/ft_static.c ファイルで検索できます。

a's	able	about	above	according
accordingly	across	actually	after	afterwards
again	against	ain't	all	allow
allows	almost	alone	along	already
also	although	always	am	among
amongst	an	and	another	any
anybody	anyhow	anyone	anything	anyway
anyways	anywhere	apart	appear	appreciate
appropriate	are	aren't	around	as
aside	ask	asking	associated	at
available	away	awfully	be	became
because	become	becomes	becoming	been
before	beforehand	behind	being	believe
below	beside	besides	best	better
between	beyond	both	brief	but
by	c'mon	c's	came	can
can't	cannot	cant	cause	causes
certain	certainly	changes	clearly	co
com	come	comes	concerning	consequently
consider	considering	contain	containing	contains
corresponding	could	couldn't	course	currently
definitely	described	despite	did	didn't
different	do	does	doesn't	doing
don't	done	down	downwards	during
each	edu	eg	eight	either
else	elsewhere	enough	entirely	especially
et	etc	even	ever	every
everybody	everyone	everything	everywhere	ex
exactly	example	except	far	few
fifth	first	five	followed	following
follows	for	former	formerly	forth
four	from	further	furthermore	get
gets	getting	given	gives	go
goes	going	gone	got	gotten
greetings	had	hadn't	happens	hardly

has	hasn't	have	haven't	having
he	he's	hello	help	hence
her	here	here's	hereafter	hereby
herein	hereupon	hers	herself	hi
him	himself	his	hither	hopefully
how	howbeit	however	i'd	i'll
i'm	i've	ie	if	ignored
immediate	in	inasmuch	inc	indeed
indicate	indicated	indicates	inner	insofar
instead	into	inward	is	isn't
it	it'd	it'll	it's	its
itself	just	keep	keeps	kept
know	known	knows	last	lately
later	latter	latterly	least	less
lest	let	let's	like	liked
likely	little	look	looking	looks
ltd	mainly	many	may	maybe
me	mean	meanwhile	merely	might
more	moreover	most	mostly	much
must	my	myself	name	namely
nd	near	nearly	necessary	need
needs	neither	never	nevertheless	new
next	nine	no	nobody	non
none	noone	nor	normally	not
nothing	novel	now	nowhere	obviously
of	off	often	oh	ok
okay	old	on	once	one
ones	only	onto	or	other
others	otherwise	ought	our	ours
ourselves	out	outside	over	overall
own	particular	particularly	per	perhaps
placed	please	plus	possible	presumably
probably	provides	que	quite	qv
rather	rd	re	really	reasonably
regarding	regardless	regards	relatively	respectively
right	said	same	saw	say
saying	says	second	secondly	see
seeing	seem	seemed	seeming	seems
seen	self	selves	sensible	sent
serious	seriously	seven	several	shall
she	should	shouldn't	since	six
so	some	somebody	somehow	someone
something	sometime	sometimes	somewhat	somewhere
soon	sorry	specified	specify	specifying
still	sub	such	sup	sure

t's	take	taken	tell	tends
th	than	thank	thanks	thanx
that	that's	thats	the	their
theirs	them	themselves	then	thence
there	there's	thereafter	thereby	therefore
therein	theres	thereupon	these	they
they'd	they'll	they're	they've	think
third	this	thorough	thoroughly	those
though	three	through	throughout	thru
thus	to	together	too	took
toward	towards	tried	tries	truly
try	trying	twice	two	un
under	unfortunately	unless	unlikely	until
unto	up	upon	us	use
used	useful	uses	using	usually
value	various	very	via	viz
vs	want	wants	was	wasn't
way	we	we'd	we'll	we're
we've	welcome	well	went	were
weren't	what	what's	whatever	when
whence	whenever	where	where's	whereafter
whereas	whereby	wherein	whereupon	wherever
whether	which	while	whither	who
who's	whoever	whole	whom	whose
why	will	willing	wish	with
within	without	won't	wonder	would
wouldn't	yes	yet	you	you'd
you'll	you're	you've	your	yours
yourself	yourselves	zero		

12.9.5 全文制限

- 全文検索は、InnoDB および MyISAM テーブルでのみサポートされています。InnoDB テーブルで FULLTEXT インデックスがサポートされるようにするには、MySQL 5.6.4 以上が必要です。
- パーティション化されたテーブルでは、全文検索がサポートされていません。セクション19.6「パーティショニングの制約と制限」を参照してください。
- 全文検索は、ほとんどのマルチバイト文字セットで使用できます。例外として、Unicode では、utf8 文字セットは使用できますが、ucs2 文字セットは使用できません。ucs2 カラム上では FULLTEXT インデックスを使用できませんが、このようなインデックスが含まれない ucs2 上で IN BOOLEAN MODE 検索を実行することはできます。

  utf8 に関する備考は utf8mb4 にも適用され、ucs2 に関する備考は utf16、utf16le、および utf32 にも適用されます。
- 中国語や日本語のような表意文字を使用する言語には、単語の区切り文字がありません。したがって、FULLTEXT パーサーでは、このような言語で単語の初めと終わりを特定できません。この意味および問題の一部の回避策については、セクション12.9「全文検索関数」で説明されています。
- 単一テーブル内で複数の文字セットを使用することはサポートされていますが、FULLTEXT インデックス内のすべてのカラムで同じ文字セットおよび照合順序が使用される必要があります。

- MATCH() カラムリストは、この MATCH() が MyISAM テーブル上の IN BOOLEAN MODE である場合を除いて、テーブルの一部の FULLTEXT インデックス定義に含まれるカラムリストに完全に一致する必要があります。MyISAM テーブルでは、インデックスが付いていないカラムでもブールモード検索を実行できます。ただし、検索が遅くなる可能性があります。
- AGAINST() への引数は、クエリー評価時に定数である文字列値にする必要があります。たとえば、テーブルカラムは、行ごとに異なる可能性があるため除外されます。
- FULLTEXT 以外の検索の場合よりも FULLTEXT 検索の場合の方が、インデックスヒントに対する制限が多くなります。セクション13.2.9.3「インデックスヒントの構文」を参照してください。
- InnoDB では、全文インデックスを含むカラムを必要とするすべての DML 演算子 (INSERT、UPDATE、DELETE) は、トランザクションのコミット時に処理されます。たとえば、INSERT 演算では、挿入された文字列がトークン化され、個々の単語に分解されます。その後、個々の単語は、トランザクションのコミット時に全文インデックステーブルに追加されます。その結果、全文検索ではコミットされたデータのみが返されます。

12.9.6 MySQL の全文検索の微調整

MySQL の全文検索機能には、ユーザーが調整できるパラメータがほとんどありません。一部の変更でソースコードを変更する必要があるために、MySQL ソース配布を持っている場合は、全文検索の動作をさらに制御できます。セクション2.9「ソースから MySQL をインストールする」を参照してください。

全文検索の有効性は、慎重に調整されます。ほとんど場合、デフォルトの動作を変更すると、実際には有効性が低くなる可能性があります。使用方法を理解していない場合は、MySQL ソースは変更しないでください。

このセクションで説明するほとんどの全文変数は、サーバーの起動時に設定する必要があります。変更するには、サーバーの再起動が必要です。サーバーが動作しているときは、変更できません。

一部の変数を変更するには、テーブル内の FULLTEXT インデックスを再構築する必要があります。これを行う手順については、このセクションの後半で説明します。

単語の最小長と最大長の構成

インデックスが付けられる単語の最小長および最大長は、InnoDB 検索インデックスの場合は innodb_ft_min_token_size および innodb_ft_max_token_size、MyISAM 検索インデックスの場合は ft_min_word_len および ft_max_word_len で定義されます。これらのオプションのいずれかを変更したら、変更を有効にするために FULLTEXT インデックスを再構築してください。たとえば、2 文字の単語を検索可能にするには、オプションファイルに次の行を配置します。

```
[mysqld]
innodb_ft_min_token_size=2
ft_min_word_len=2
```

次に、サーバーを再起動し、FULLTEXT インデックスを再構築します。MyISAM テーブルについては、MyISAM の全文インデックスを再構築する際に従う手順で、myisamchk に関する備考に注意してください。

自然言語検索のしきい値の構成

MyISAM 検索インデックスでは、選択された特定の重み付けスキームによって、自然言語検索で 50% のしきい値が決定されます。これを無効にするには、storage/myisam/ftdefs.h で次の行を検索してください。

```
#define GWS_IN_USE GWS_PROB
```

この行を次のように変更します。

```
#define GWS_IN_USE GWS_FREQ
```

次に、MySQL を再コンパイルします。この場合、インデックスを再構築する必要はありません。

> 注記
>
> このように変更すると、MATCH() 関数に適切な関連性値を提供する MySQL の能力が大幅に低下します。このような一般的な単語を検索する必要がある場合は、代わりに、50% のしきい値に従わない IN BOOLEAN MODE を使用して検索する方が適切です。

ブール全文検索演算子の変更

MyISAM テーブル上でブール全文検索に使用される演算子を変更するには、ft_boolean_syntax システム変数を設定します。(InnoDB には同等の設定がありません。)この変数はサーバーの実行中に変更できますが、そのためには SUPER 権限を持っている必要があります。この場合は、インデックスを再構築する必要はありません。この変数を設定する方法を制御するルールについては、セクション5.1.4「サーバーシステム変数」を参照してください。

文字セットの変更

次のリストで説明するように、単語文字とみなされる文字セットは複数の方法で変更できます。変更が完了したら、任意の FULLTEXT インデックスを含むテーブルごとにインデックスを再構築します。ハイフン文字 (' - ') を単語文字として処理すると仮定します。次の方法のいずれかを使用します。

- MySQL ソースを変更します。storage/innobase/handler/ha_innodb.cc (InnoDB の場合) または storage/myisam/ftdefs.h (MyISAM の場合) で、true_word_char() および misc_word_char() マクロを参照してください。それらのマクロのいずれかに '-' を追加し、MySQL を再コンパイルします。
- 文字セットファイルを変更します。再コンパイルする必要はありません。true_word_char() マクロでは、英数字とその他の文字を区別するために 「character type」 テーブルが使用されます。文字セット XML ファイルのいずれかで <ctype><map> 配列の内容を編集すると、'-' が「英字」になるように指定できます。次に、FULLTEXT インデックスに指定された文字セットを使用します。<ctype><map> 配列の書式については、セクション10.3.1「文字定義配列」を参照してください。
- インデックス付きのカラムで使用される文字セットに新しい照合順序を追加し、その照合順序が使用されるようにカラムを変更します。照合順序の追加に関する一般的な情報については、セクション10.4「文字セットへの照合順序の追加」を参照してください。全文インデックス作成に固有の例については、セクション12.9.7「全文インデックス作成用の照合順序の追加」を参照してください。

InnoDB 全文インデックスの再構築

インデックス作成に影響を与える全文変数 (innodb_ft_min_token_size、innodb_ft_max_token_size、innodb_ft_server_stopword_table、innodb_ft_user_stopword_table) を変更する場合は、変更したあとに FULLTEXT インデックスを再構築する必要があります。innodb_ft_min_token_size および innodb_ft_max_token_size 変数は動的に設定できないため、変更するにはサーバーを再起動し、インデックスを再構築する必要があります。

InnoDB テーブルの FULLTEXT インデックスを再構築するには、DROP INDEX および ADD INDEX オプションを付けて ALTER TABLE を使用して、各インデックスを削除してから再作成します。

InnoDB 全文インデックスの最適化

全文インデックス付きのテーブル上で OPTIMIZE TABLE を実行すると、全文インデックスが再構築され、削除済みのドキュメント ID が削除され、同じ単語に対応する複数のエントリが連結されます (可能な場合)。

全文インデックスを最適化するには、innodb_optimize_fulltext_only を有効にして、OPTIMIZE TABLE を実行します。

```
mysql> set GLOBAL innodb_optimize_fulltext_only=ON;
Query OK, 0 rows affected (0.01 sec)

mysql> OPTIMIZE TABLE opening_lines;
+--------------------+----------+----------+----------+
| Table              | Op       | Msg_type | Msg_text |
+--------------------+----------+----------+----------+
| test.opening_lines | optimize | status   | OK       |
+--------------------+----------+----------+----------+
1 row in set (0.01 sec)
```

大きなテーブルで全文インデックスの再構築時間が長くなることを回避するには、innodb_ft_num_word_optimize オプションを使用すれば、最適化を段階的に実行できます。innodb_ft_num_word_optimize オプションでは、OPTIMIZE TABLE が実行されるたびに最適化される単語の数が定義されます。デフォルト設定は 2000 です。これは、OPTIMIZE TABLE が実行されるたびに 2000 個の単語が最適化されることを表します。後続の OPTIMIZE TABLE 演算は、先行する OPTIMIZE TABLE 演算が終了した場所から続行されます。

MyISAM 全文インデックスの再構築

インデックス作成に影響を与える全文変数 (ft_min_word_len、ft_max_word_len、または ft_stopword_file) を変更する場合や、ストップワードファイル自体を変更する場合は、変更して、サーバーを再起動したあとに、FULLTEXT インデックスを再構築する必要があります。

MyISAM テーブルの FULLTEXT インデックスを再構築するには、QUICK 修復演算を実行すれば十分です。

```
mysql> REPAIR TABLE tbl_name QUICK;
```

または、先ほど説明した ALTER TABLE を使用します。これは、修復演算よりも高速になる可能性もあります。

任意の FULLTEXT インデックスを含む各テーブルは、上記のように修復する必要があります。それ以外の場合は、テーブルのクエリーで不正な結果が生成される可能性があり、テーブルを変更すると、サーバーでは、テーブルが破損していて修復が必要であるとみなされます。

myisamchk を使用して、MyISAM テーブルインデックスを変更する演算 (修復や分析など) を実行する場合は、ほかに指定がなければ、単語の最小長、単語の最大長、およびストップワードファイルのデフォルトの全文パラメータ値を使用して、FULLTEXT インデックスが再構築されます。これにより、クエリーに失敗する可能性があります。

この問題は、これらのパラメータがサーバーでのみ認識されていることが原因で発生します。MyISAM インデックスファイルには格納されていません。サーバーで使用される単語の最小長や最大長、またはストップワードファイルの値を変更した場合の問題を回避するには、mysqld で使用される myisamchk と同じ ft_min_word_len、ft_max_word_len、および ft_stopword_file 値を指定します。たとえば、単語の最小長を 3 に設定した場合は、次のように myisamchk を使用してテーブルを修復できます。

```
shell> myisamchk --recover --ft_min_word_len=3 tbl_name.MYI
```

myisamchk およびサーバーで全文パラメータに必ず同じ値が使用されるようにするには、オプションファイルの [mysqld] と [myisamchk] の両方のセクションにそれぞれを配置してください。

```
[mysqld]
ft_min_word_len=3

[myisamchk]
ft_min_word_len=3
```

MyISAM テーブルインデックスの変更に myisamchk を使用する方法の代替として、REPAIR TABLE、ANALYZE TABLE、OPTIMIZE TABLE、または ALTER TABLE ステートメントを使用します。これらのステートメントは、適切に使用される全文パラメータ値が認識されているサーバーで実行されます。

12.9.7 全文インデックス作成用の照合順序の追加

このセクションでは、全文検索用に新しい照合順序を追加する方法について説明します。サンプルの照合順序は latin1_swedish_ci と似ていますが、'-' 文字は、単語文字としてインデックスを付けることができるように、句読文字としてではなく英字として処理されます。照合順序の追加に関する一般的な情報については、セクション 10.4「文字セットへの照合順序の追加」で説明されています。この情報を参照し、関与するファイルをよく理解することが前提となっています。

全文インデックス作成用に照合順序を追加するには、次の手順を使用します。

1. 照合順序を Index.xml ファイルに追加します。照合順序 ID は未使用にする必要があるため、その ID がすでにシステムで取得されている場合は、1000 以外の値を選択してください。

   ```
   <charset name="latin1">
   ...
   <collation name="latin1_fulltext_ci" id="1000"/>
   </charset>
   ```

2. latin1.xml ファイルで照合順序のソート順序を宣言します。この場合、latin1_swedish_ci から順序をコピーできます。

   ```
   <collation name="latin1_fulltext_ci">
   <map>
   00 01 02 03 04 05 06 07 08 09 0A 0B 0C 0D 0E 0F
   10 11 12 13 14 15 16 17 18 19 1A 1B 1C 1D 1E 1F
   20 21 22 23 24 25 26 27 28 29 2A 2B 2C 2D 2E 2F
   30 31 32 33 34 35 36 37 38 39 3A 3B 3C 3D 3E 3F
   40 41 42 43 44 45 46 47 48 49 4A 4B 4C 4D 4E 4F
   50 51 52 53 54 55 56 57 58 59 5A 5B 5C 5D 5E 5F
   60 41 42 43 44 45 46 47 48 49 4A 4B 4C 4D 4E 4F
   50 51 52 53 54 55 56 57 58 59 5A 7B 7C 7D 7E 7F
   80 81 82 83 84 85 86 87 88 89 8A 8B 8C 8D 8E 8F
   90 91 92 93 94 95 96 97 98 99 9A 9B 9C 9D 9E 9F
   A0 A1 A2 A3 A4 A5 A6 A7 A8 A9 AA AB AC AD AE AF
   B0 B1 B2 B3 B4 B5 B6 B7 B8 B9 BA BB BC BD BE BF
   41 41 41 41 5C 5B 5C 43 45 45 45 45 49 49 49 49
   44 4E 4F 4F 4F 4F 5D D7 D8 55 55 55 59 59 DE DF
   41 41 41 41 5C 5B 5C 43 45 45 45 45 49 49 49 49
   ```

```
44 4E 4F 4F 4F 4F 5D F7 D8 55 55 55 59 59 DE FF
</map>
</collation>
```

3. latin1.xml で ctype 配列を変更します。0x2D ('-' 文字のコード) に対応する値を 10 (句読点) から 01 (小文字) に変更します。次の配列では、これは 4 行目の要素で、最後から 3 番目の値です。

```
<ctype>
<map>
00
20 20 20 20 20 20 20 20 20 28 28 28 28 28 20 20
20 20 20 20 20 20 20 20 20 20 20 20 20 20 20 20
48 10 10 10 10 10 10 10 10 10 10 10 01 10 10
84 84 84 84 84 84 84 84 84 84 10 10 10 10 10 10
10 81 81 81 81 81 81 01 01 01 01 01 01 01 01 01
01 01 01 01 01 01 01 01 01 01 10 10 10 10 10
10 82 82 82 82 82 82 02 02 02 02 02 02 02 02 02
02 02 02 02 02 02 02 02 02 02 10 10 10 10 20
10 00 10 02 10 10 10 10 10 10 01 10 01 00 01 00
00 10 10 10 10 10 10 10 10 10 02 10 02 00 02 01
48 10 10 10 10 10 10 10 10 10 10 10 10 10 10
10 10 10 10 10 10 10 10 10 10 10 10 10 10 10
01 01 01 01 01 01 01 01 01 01 01 01 01 01 01
01 01 01 01 01 01 01 10 01 01 01 01 01 01 02
02 02 02 02 02 02 02 02 02 02 02 02 02 02 02
02 02 02 02 02 02 02 10 02 02 02 02 02 02 02
</map>
</ctype>
```

4. サーバーを再起動します。

5. 新しい照合順序を使用するには、使用されるカラムの定義に追加します。

```
mysql> DROP TABLE IF EXISTS t1;
Query OK, 0 rows affected (0.13 sec)

mysql> CREATE TABLE t1 (
    a TEXT CHARACTER SET latin1 COLLATE latin1_fulltext_ci,
    FULLTEXT INDEX(a)
    ) ENGINE=InnoDB;
Query OK, 0 rows affected (0.47 sec)
```

6. 照合順序をテストして、ハイフンが単語文字としてみなされることを確認します。

```
mysql> INSERT INTO t1 VALUEs ('----'),('....'),('abcd');
Query OK, 3 rows affected (0.22 sec)
Records: 3  Duplicates: 0  Warnings: 0

mysql> SELECT * FROM t1 WHERE MATCH a AGAINST ('----' IN BOOLEAN MODE);
+------+
| a    |
+------+
| ---- |
+------+
1 row in set (0.00 sec)
```

12.10 キャスト関数と演算子

表 12.14 キャスト関数

名前	説明
BINARY	文字列をバイナリ文字列にキャストします
CAST()	値を特定の型としてキャストします
CONVERT()	値を特定の型としてキャストします

- BINARY

  BINARY 演算子は、それに続く文字列をバイナリ文字列にキャストします。これは、カラムの比較を文字ごとでなく、バイトごとに強制的に実行させる簡単な方法です。これにより、カラムが BINARY または BLOB として定義されていない場合でも、比較で大文字と小文字が区別されます。また、BINARY では末尾の空白文字が重要になります。

```
mysql> SELECT 'a' = 'A';
       -> 1
```

```
mysql> SELECT BINARY 'a' = 'A';
    -> 0
mysql> SELECT 'a' = 'a ';
    -> 1
mysql> SELECT BINARY 'a' = 'a ';
    -> 0
```

比較時に、BINARY によって演算全体が影響を受けます。これは、同じ結果を持ついずれかのオペランドの前に指定できます。

BINARY str は、CAST(str AS BINARY) の略です。

一部のコンテキストでは、インデックス付きのカラムを BINARY にキャストすると、MySQL はインデックスを効率的に使用できなくなります。

- CAST(expr AS type)

  CONVERT() と同様に、CAST() 関数には任意の型の式が指定され、指定された型の結果値が生成されます。詳細は、CONVERT() の説明を参照してください。

- CONVERT(expr,type), CONVERT(expr USING transcoding_name)

  CONVERT() および CAST() 関数には、任意の型の式が指定され、指定された型の結果値が生成されます。

  結果の type には、次の値のいずれかを指定できます。

  - BINARY[(N)]
  - CHAR[(N)]
  - DATE
  - DATETIME
  - DECIMAL[(M[,D])]
  - SIGNED [INTEGER]
  - TIME
  - UNSIGNED [INTEGER]

  BINARY は、BINARY データ型の文字列を生成します。これによる比較への影響度については、セクション 11.4.2「BINARY および VARBINARY 型」を参照してください。オプションの長さ N が指定されている場合に、BINARY(N) を使用すると、キャストで N バイトの引数しか使用されなくなります。値が N バイトよりも短い場合は、N の長さになるまで 0x00 バイトでパディングされます。

  CHAR(N) 句を使用すると、キャストで N 文字の引数しか使用されなくなります。

  CAST() および CONVERT(... USING ...) は標準 SQL の構文です。USING 形式以外の CONVERT() は ODBC の構文です。

  USING とともに CONVERT() を使用すると、さまざまな文字セット間でデータが変換されます。MySQL では、トランスコーディング名は対応する文字セット名と同じです。たとえば、このステートメントは、デフォルトの文字セットの文字列 'abc' を utf8 文字セットの対応する文字列に変換します。

```
SELECT CONVERT('abc' USING utf8);
```

バイナリ文字列には文字セットがなく、大文字と小文字の区別の概念もないため、通常は、大文字と小文字が区別されない方法で、BLOB 値またはその他のバイナリ文字列を比較できません。大文字と小文字が区別されない比較を実行するには、CONVERT() 関数を使用して、値を非バイナリ文字列に変換します。結果の比較には、文字列照合順序が使用されます。たとえば、結果の文字セットに大文字と小文字が区別されない照合順序が含まれる場合は、LIKE 演算子でも大文字と小文字が区別されません。

```
SELECT 'A' LIKE CONVERT(blob_col USING latin1) FROM tbl_name;
```

別の文字セットを使用するには、上記のステートメントの latin1 にその名前を代入します。変換された文字列用に特定の照合順序を指定するには、セクション10.1.9.2「CONVERT() と CAST()」で説明するように、CONVERT() の呼び出しのあとに COLLATE 句を使用します。たとえば、latin1_german1_ci を使用する場合は次のとおりです。

```
SELECT 'A' LIKE CONVERT(blob_col USING latin1) COLLATE latin1_german1_ci
  FROM tbl_name;
```

CONVERT() は、さまざま文字セットで表される文字列を比較する際に、より一般的に使用できます。

LOWER() (および UPPER()) をバイナリ文字列 (BINARY、VARBINARY、BLOB) に適用しても、何の効果もありません。大文字/小文字の変換を実行するには、バイナリ文字列を非バイナリ文字列に変換します。

```
mysql> SET @str = BINARY 'New York';
mysql> SELECT LOWER(@str), LOWER(CONVERT(@str USING latin1));
+-------------+-----------------------------------+
| LOWER(@str) | LOWER(CONVERT(@str USING latin1)) |
+-------------+-----------------------------------+
| New York    | new york                          |
+-------------+-----------------------------------+
```

キャスト関数は、CREATE TABLE ... SELECT ステートメントで特定の型を持つカラムを作成する際に役立ちます。

```
CREATE TABLE new_table SELECT CAST('2000-01-01' AS DATE);
```

この関数は、ENUM カラムを語彙順にソートする際に役立つこともあります。通常、ENUM カラムのソートは、内部数値を使用して実行されます。値を CHAR にキャストすると、語彙順にソートされます。

```
SELECT enum_col FROM tbl_name ORDER BY CAST(enum_col AS CHAR);
```

CAST(str AS BINARY) は BINARY str と同じです。CAST(expr AS CHAR) では、式はデフォルトの文字セットを持つ文字列として処理されます。

CONCAT('Date: ',CAST(NOW() AS DATE)) のように、より複雑な式の一部として使用する場合でも、CAST() の結果が変わります。

データを別の書式で抽出するには、CAST() ではなく、代わりに LEFT() または EXTRACT() のような文字列関数を使用してください。セクション12.7「日付および時間関数」を参照してください。

数値のコンテキストで文字列を数値にキャストするには、通常は、文字列値を数字と同様に使用する以外には何もする必要はありません。

```
mysql> SELECT 1+'1';
    -> 2
```

算術演算で文字列を使用する場合は、式の評価時に浮動小数点数に変換されます。

文字列のコンテキストで数字を使用する場合は、自動的に数字が文字列に変換されます。

```
mysql> SELECT CONCAT('hello you ',2);
    -> 'hello you 2'
```

MySQL 5.6.4 よりも前では、どのテーブルからも選択されないステートメントの TIMESTAMP 値で明示的に CAST() を使用すると、変換が実行される前に、その値が MySQL 5.6 で文字列として処理されます。これにより、値を数値型にキャストすると、次に示すように値が切り捨てられます。

```
mysql> SELECT CAST(TIMESTAMP '2014-09-08 18:07:54' AS SIGNED);
+-----------------------------------------------+
| CAST(TIMESTAMP '2014-09-08 18:07:54' AS SIGNED) |
+-----------------------------------------------+
|                                          2014 |
+-----------------------------------------------+
1 row in set, 1 warning (0.00 sec)

mysql> SHOW WARNINGS;
+---------+------+-------------------------------------------------------------+
| Level   | Code | Message                                                     |
+---------+------+-------------------------------------------------------------+
| Warning | 1292 | Truncated incorrect INTEGER value: '2014-09-08 18:07:54' |
+---------+------+-------------------------------------------------------------+
1 row in set (0.00 sec)
```

次に示すように、これはテーブルから行を選択するときには適用されません。

```
mysql> USE test;

Database changed
mysql> CREATE TABLE c_test (col TIMESTAMP);
Query OK, 0 rows affected (0.07 sec)
```

```
mysql> INSERT INTO c_test VALUES ('2014-09-08 18:07:54');
Query OK, 1 row affected (0.05 sec)

mysql> SELECT col, CAST(col AS UNSIGNED) AS c_col FROM c_test;
+---------------------+----------------+
| col                 | c_col          |
+---------------------+----------------+
| 2014-09-08 18:07:54 | 20140908180754 |
+---------------------+----------------+
1 row in set (0.00 sec)
```

MySQL 5.6.4 以降では、次に示すように、どの行も選択されていないクエリーでも、選択されている場合と同様に CAST() で TIMESTAMP 値が処理されます。

```
mysql> SELECT CAST(TIMESTAMP '2014-09-08 18:07:54' AS SIGNED);
+-------------------------------------------------+
| CAST(TIMESTAMP '2014-09-08 18:05:07' AS SIGNED) |
+-------------------------------------------------+
|                                  20140908180754 |
+-------------------------------------------------+
1 row in set (0.00 sec)
```

数字から文字列への暗黙的な変換については、セクション12.2「式評価での型変換」を参照してください。

MySQL では、符号付きと符号なしの両方の 64 ビット値を使用した演算がサポートされています。数値演算子 (+ や - など) を使用していて、オペランドのいずれかが符号なし整数である場合は、デフォルトの結果が符号なしになります (セクション12.6.1「算術演算子」を参照してください)。これは、SIGNED または UNSIGNED キャスト演算子を使用して、値をそれぞれ符号付きまたは符号なしの 64 ビット整数にキャストすることでオーバーライドできます。

```
mysql> SELECT CAST(1-2 AS UNSIGNED)
    -> 18446744073709551615
mysql> SELECT CAST(CAST(1-2 AS UNSIGNED) AS SIGNED);
    -> -1
```

オペランドのいずれかが浮動小数点値である場合は、結果が浮動小数点値になり、上記のルールによる影響を受けません。(このコンテキストでは、DECIMAL カラム値は浮動小数点値とみなされます。)

```
mysql> SELECT CAST(1 AS UNSIGNED) - 2.0;
    -> -1.0
```

SQL モードは、変換演算の結果に影響を与えます。例:

- 「zero」 日付文字列を日付に変換する場合、CONVERT() と CAST() は NULL を返し、NO_ZERO_DATE SQL モードが有効になると警告を発行します。
- 整数の減算では、NO_UNSIGNED_SUBTRACTION SQL モードが有効になっている場合は、オペランドのいずれかが符号なしでも、結果が符号付きになります。

詳細は、セクション5.1.7「サーバー SQL モード」を参照してください。

12.11 XML 関数

表 12.15 XML 関数

名前	説明
ExtractValue()	XPath 表記法を使用して、XML 文字列から値を抽出します
UpdateXML()	置換後 XML フラグメントを返します

このセクションでは、MySQL での XML および関連する機能について説明します。

> **注記**
>
> --xml オプションを付けて呼び出すと、mysql および mysqldump クライアントで XML 書式の出力を MySQL から取得できます。セクション4.5.1「mysql — MySQL コマンド行ツール」およびセクション4.5.4「mysqldump — データベースバックアッププログラム」を参照してください。

基本的な XPath 1.0 (XML Path Language、バージョン 1.0) の機能を提供する 2 つの関数が使用可能です。XPath の構文および使用法に関する一部の基本情報については、このセクションの後半で提供しますが、このようなトピックの詳しい説明は、このマニュアルのスコープ外です。最終的な情報については、「XML Path Language

(XPath) 1.0 標準」を参照してください。XPath がはじめてのユーザーや基本の復習を希望するユーザーに役立つリソースは、複数の言語で入手できる「Zvon.org XPath Tutorial」です。

> **注記**
>
> これらの関数はまだ開発中です。XML および XPath 機能のこれらの側面やその他の側面については、MySQL 5.6 以降で引き続き改善します。これらについて議論したり、質問したり、MySQL XML ユーザーフォーラムでほかのユーザーからの支援を得たりすることもできます。

これらの関数で使用される XPath の式では、ユーザー変数およびローカルストアドプログラム変数がサポートされています。ユーザー変数は簡単にチェックされます。ストアドプログラムへのローカル変数は厳密にチェックされます (Bug#26518 も参照してください)。

- ユーザー変数 (簡単なチェック)　構文 $@variable_name を使用する変数 (つまり、ユーザー変数) はチェックされません。変数の型が間違っている場合や、変数に値が事前に割り当てられていない場合でも、サーバーから警告やエラーが発行されません。これは、(たとえば) $@myvariable が使用されても警告が発行されないため ($@myvariable は故意に間違えたものです)、入力ミスについてはユーザーが全責任を負うことも意味します。

  例:

```
mysql> SET @xml = '<a><b>X</b><b>Y</b></a>';
Query OK, 0 rows affected (0.00 sec)

mysql> SET @i =1, @j = 2;
Query OK, 0 rows affected (0.00 sec)

mysql> SELECT @i, ExtractValue(@xml, '//b[$@i]');
+------+--------------------------------+
| @i   | ExtractValue(@xml, '//b[$@i]') |
+------+--------------------------------+
|    1 | X                              |
+------+--------------------------------+
1 row in set (0.00 sec)

mysql> SELECT @j, ExtractValue(@xml, '//b[$@j]');
+------+--------------------------------+
| @j   | ExtractValue(@xml, '//b[$@j]') |
+------+--------------------------------+
|    2 | Y                              |
+------+--------------------------------+
1 row in set (0.00 sec)

mysql> SELECT @k, ExtractValue(@xml, '//b[$@k]');
+------+--------------------------------+
| @k   | ExtractValue(@xml, '//b[$@k]') |
+------+--------------------------------+
| NULL |                                |
+------+--------------------------------+
1 row in set (0.00 sec)
```

- ストアドプログラム内の変数 (厳密なチェック)　これらの関数をストアドプログラム内部で呼び出すときに、構文 $variable_name を使用する変数を宣言し、これらの関数で使用できます。このような変数は、定義されているストアドプログラムへのローカル変数であり、型および値について厳密にチェックされます。

  例:

```
mysql> DELIMITER |

mysql> CREATE PROCEDURE myproc ()
    -> BEGIN
    ->   DECLARE i INT DEFAULT 1;
    ->   DECLARE xml VARCHAR(25) DEFAULT '<a>X</a><a>Y</a><a>Z</a>';
    ->
    ->   WHILE i < 4 DO
    ->     SELECT xml, i, ExtractValue(xml, '//a[$i]');
    ->     SET i = i+1;
    ->   END WHILE;
    -> END |
Query OK, 0 rows affected (0.01 sec)

mysql> DELIMITER ;

mysql> CALL myproc();
+-------------------------+---+-------------------------------+
| xml                     | i | ExtractValue(xml, '//a[$i]') |
```

```
+-------------------------+---+-------------------------------+
| <a>X</a><a>Y</a><a>Z</a> | 1 | X                             |
+-------------------------+---+-------------------------------+
1 row in set (0.00 sec)

+-------------------------+---+-------------------------------+
| xml                     | i | ExtractValue(xml, '//a[$i]') |
+-------------------------+---+-------------------------------+
| <a>X</a><a>Y</a><a>Z</a> | 2 | Y                             |
+-------------------------+---+-------------------------------+
1 row in set (0.01 sec)

+-------------------------+---+-------------------------------+
| xml                     | i | ExtractValue(xml, '//a[$i]') |
+-------------------------+---+-------------------------------+
| <a>X</a><a>Y</a><a>Z</a> | 3 | Z                             |
+-------------------------+---+-------------------------------+
1 row in set (0.01 sec)
```

パラメータ　ストアドルーチン内部の XPath 式で使用され、パラメータとして渡される変数も、厳密なチェックの対象です。

ユーザー変数やストアドプログラムへのローカル変数を含む式は、その他の点 (表記法は除く) では、XPath 1.0 仕様で規定されている変数を含む XPath 式のルールに準拠する必要があります。

> **注記**
>
> 現時点では、XPath 式の格納に使用されたユーザー変数は、空の文字列として処理されます。このため、ユーザー変数として XPath 式を格納することはできません。(Bug #32911)

- ExtractValue(xml_frag, xpath_expr)

  ExtractValue() には、XML マークアップ xml_frag のフラグメントと XPath 式 xpath_expr (locator とも呼ばれます) の 2 つの文字列引数が指定され、要素の子または XPath 式で一致された要素である 1 番目のテキストノードのテキスト (CDATA) が返されます。MySQL 5.6.6 以前では、XPath 式に最大でも 127 文字しか含めることができませんでした。この制限は、MySQL 5.6.7 で解除されました。(Bug #13007062、Bug #62429)

  この関数を使用することは、/text() を追加したあとに xpath_expr を使用して一致を実行することと同等です。言い換えると、ExtractValue('<a>Sakila', '/a/b') と ExtractValue('<a>Sakila', '/a/b/text()') では同じ結果が生成されます。

  複数の一致が見つかった場合は、一致する各要素の 1 番目の子テキストノードの内容が空白文字で区切られた単一文字列として (一致した順序で) 返されます。

  式に一致するテキストノード (暗黙的な /text() を含む) が見つからない場合は、どのような理由でも、xpath_expr が有効で、xml_frag が適切にネストされ、閉じられた要素で構成されていれば、空の文字列が返されます。空の要素で一致することと、まったく一致しないこととは区別されません。これは意図的なものです。

  xml_frag で一致する要素が見つからなかったのか、またはこのような要素は見つかったが、子テキストノードが含まれていなかったのかを判断する必要がある場合は、XPath count() 関数を使用する式の結果をテストしてください。たとえば、次に示すように、これらのステートメントの両方で空の文字列が返されます。

```
mysql> SELECT ExtractValue('<a><b/></a>', '/a/b');
+-------------------------------------+
| ExtractValue('<a><b/></a>', '/a/b') |
+-------------------------------------+
|                                     |
+-------------------------------------+
1 row in set (0.00 sec)

mysql> SELECT ExtractValue('<a><c/></a>', '/a/b');
+-------------------------------------+
| ExtractValue('<a><c/></a>', '/a/b') |
+-------------------------------------+
|                                     |
+-------------------------------------+
1 row in set (0.00 sec)
```

  ただし、次のコマンドを使用すれば、実際に一致する要素があったかどうかを判断できます。

```
mysql> SELECT ExtractValue('<a><b/></a>', 'count(/a/b)');
+-------------------------------------+
```

```
| ExtractValue('<a><b/></a>', 'count(/a/b)') |
+--------------------------------------------+
| 1                                          |
+--------------------------------------------+
1 row in set (0.00 sec)

mysql> SELECT ExtractValue('<a><c/></a>', 'count(/a/b)');
+--------------------------------------------+
| ExtractValue('<a><c/></a>', 'count(/a/b)') |
+--------------------------------------------+
| 0                                          |
+--------------------------------------------+
1 row in set (0.01 sec)
```

> **重要**
>
> ExtractValue() では CDATA のみが返され、一致するタグ内に含まれるタグや、それらの内容は返されません (次の例で、val1 として返された結果を参照してください)。

```
mysql> SELECT
    ->   ExtractValue('<a>ccc<b>ddd</b></a>', '/a') AS val1,
    ->   ExtractValue('<a>ccc<b>ddd</b></a>', '/a/b') AS val2,
    ->   ExtractValue('<a>ccc<b>ddd</b></a>', '//b') AS val3,
    ->   ExtractValue('<a>ccc<b>ddd</b></a>', '/b') AS val4,
    ->   ExtractValue('<a>ccc<b>ddd</b><b>eee</b></a>', '//b') AS val5;

+------+------+------+------+---------+
| val1 | val2 | val3 | val4 | val5    |
+------+------+------+------+---------+
| ccc  | ddd  | ddd  |      | ddd eee |
+------+------+------+------+---------+
```

この関数では、contains() との比較を実行し、その他の文字列関数 (CONCAT() など) と同じ照合順序アグリゲーションを実行し、それらの引数の照合順序強制性を考慮に入れる際に、現在の SQL 照合順序が使用されます。この動作を制御するルールの説明については、セクション10.1.7.5「式の照合順序」を参照してください。

(以前は、大文字と小文字が区別されるバイナリが常に使用されていました。)

次の例に示すように、xml_frag に、適切にネストされていない要素や閉じられていない要素が含まれ、警告が生成された場合は、NULL が返されます。

```
mysql> SELECT ExtractValue('<a>c</a><b', '//a');
+-----------------------------------+
| ExtractValue('<a>c</a><b', '//a') |
+-----------------------------------+
| NULL                              |
+-----------------------------------+
1 row in set, 1 warning (0.00 sec)

mysql> SHOW WARNINGS\G
*************************** 1. row ***************************
  Level: Warning
   Code: 1525
Message: Incorrect XML value: 'parse error at line 1 pos 11:
        END-OF-INPUT unexpected ('>' wanted)'
1 row in set (0.00 sec)

mysql> SELECT ExtractValue('<a>c</a><b/>', '//a');
+-------------------------------------+
| ExtractValue('<a>c</a><b/>', '//a') |
+-------------------------------------+
| c                                   |
+-------------------------------------+
1 row in set (0.00 sec)
```

- UpdateXML(xml_target, xpath_expr, new_xml)

  この関数は、XML マークアップ xml_target の特定のフラグメントの一部を新しい XML フラグメント new_xml に置き換えてから、変更された XML を返します。置換された xml_target の一部は、ユーザーが指定した XPath 式 xpath_expr に一致します。MySQL 5.6.6 以前では、XPath 式に最大でも 127 文字しか含めることができませんでした。この制限は、MySQL 5.6.7 で解除されました。(Bug #13007062、Bug #62429)

  xpath_expr に一致する式が見つからない場合、または複数の一致が見つかった場合、この関数は元の xml_target XML フラグメントを返します。3 つの引数はすべて文字列にする必要があります。

```
mysql> SELECT
```

```
    ->   UpdateXML('<a><b>ccc</b><d></d></a>', '/a', '<e>fff</e>') AS val1,
    ->   UpdateXML('<a><b>ccc</b><d></d></a>', '/b', '<e>fff</e>') AS val2,
    ->   UpdateXML('<a><b>ccc</b><d></d></a>', '//b', '<e>fff</e>') AS val3,
    ->   UpdateXML('<a><b>ccc</b><d></d></a>', '/a/d', '<e>fff</e>') AS val4,
    ->   UpdateXML('<a><d></d><b>ccc</b><d></d></a>', '/a/d', '<e>fff</e>') AS val5
    -> \G

*************************** 1. row ***************************
val1: <e>fff</e>
val2: <a><b>ccc</b><d></d></a>
val3: <a><e>fff</e><d></d></a>
val4: <a><b>ccc</b><e>fff</e></a>
val5: <a><d></d><b>ccc</b><d></d></a>
```

> **注記**
>
> XPath 構文のさらに詳しい説明や使用法については、このマニュアルのスコープ外です。最終的な情報については、「XML Path Language (XPath) 1.0 仕様」を参照してください。XPath がはじめてのユーザーや基本の復習を希望するユーザーに役立つリソースは、複数の言語で入手できる「Zvon.org XPath Tutorial」です。

一部の基本的な XPath 式の説明および例は、次のとおりです。

- /tag

  <tag/> がルート要素の場合にかぎり、<tag/> に一致します。

  例: /a はいちばん外側の (ルート) タグに一致するため、<a> には一致があります。この例では、別の要素の子であるため、<a/> の内側の a 要素には一致しません。

- /tag1/tag2

  <tag1/> の子であり、<tag1/> がルート要素である場合にかぎり、<tag2/> に一致します。

  例: /a/b はルート要素 a の子であるため、XML フラグメント <a> の b 要素に一致します。この場合、b はルート要素 (その他の要素の子) であるため、<a/> には一致がありません。XPath 式でも <a><c></c> に一致がありません。ここで、b は a の子孫ですが、実際には a の子ではありません。

  この構成は、3 つ以上の要素に拡張できます。たとえば、XPath 式 /a/b/c は、フラグメント <a><c/> 内の c 要素に一致します。

- //tag

  <tag> の任意のインスタンスに一致します。

  例: //a は、<a><c/>、<c><a>、<c><a/></c> のいずれかの a 要素に一致します。

  // は / と組み合わせることができます。たとえば、//a/b は、フラグメント <a> または <a><c/> 内の b 要素に一致します。

> **注記**
>
> //tag は /descendant-or-self::*/tag と同等です。よく見られる誤りは、これと /descendant-or-self::tag とを混同することです。次に示すように、実際には後者の式ではまったく異なる結果が生成される可能性があります。

```
mysql> SET @xml = '<a><b><c>w</c><b>x</b><d>y</d>z</b></a>';
Query OK, 0 rows affected (0.00 sec)

mysql> SELECT @xml;
+---------------------------------------+
| @xml                                  |
+---------------------------------------+
| <a><b><c>w</c><b>x</b><d>y</d>z</b></a> |
+---------------------------------------+
1 row in set (0.00 sec)

mysql> SELECT ExtractValue(@xml, '//b[1]');
+-------------------------------+
| ExtractValue(@xml, '//b[1]') |
+-------------------------------+
| x z                           |
+-------------------------------+
```

```
1 row in set (0.00 sec)

mysql> SELECT ExtractValue(@xml, '//b[2]');
+-------------------------------+
| ExtractValue(@xml, '//b[2]') |
+-------------------------------+
|                          |
+-------------------------------+
1 row in set (0.01 sec)

mysql> SELECT ExtractValue(@xml, '/descendant-or-self::*/b[1]');
+-------------------------------------------------+
| ExtractValue(@xml, '/descendant-or-self::*/b[1]') |
+-------------------------------------------------+
| x z                                             |
+-------------------------------------------------+
1 row in set (0.06 sec)

mysql> SELECT ExtractValue(@xml, '/descendant-or-self::*/b[2]');
+-------------------------------------------------+
| ExtractValue(@xml, '/descendant-or-self::*/b[2]') |
+-------------------------------------------------+
|                                                 |
+-------------------------------------------------+
1 row in set (0.00 sec)


mysql> SELECT ExtractValue(@xml, '/descendant-or-self::b[1]');
+-----------------------------------------------+
| ExtractValue(@xml, '/descendant-or-self::b[1]') |
+-----------------------------------------------+
| z                                             |
+-----------------------------------------------+
1 row in set (0.00 sec)

mysql> SELECT ExtractValue(@xml, '/descendant-or-self::b[2]');
+-----------------------------------------------+
| ExtractValue(@xml, '/descendant-or-self::b[2]') |
+-----------------------------------------------+
| x                                             |
+-----------------------------------------------+
1 row in set (0.00 sec)
```

- * 演算子は、任意の要素に一致する「ワイルドカード」として機能します。たとえば、式 /*/b は、XML フラグメント <a> または <c></c> 内の b 要素に一致します。ただし、b はその他の要素の子である必要があるため、この式ではフラグメント <a/> 内の一致は生成されません。ワイルドカードは、任意の位置で使用できます。式 /*/b/* は、それ自体がルート要素でない b 要素の任意の子に一致します。

- | (UNION) 演算子を使用すれば、複数の口ケータのいずれかに一致できます。たとえば、式 //b|//c は、XML ターゲット内のすべての b および c 要素に一致します。

- その属性の 1 つ以上の値に基づいて、要素に一致することもできます。これは、構文 tag[@attribute="value"] を使用して実行されます。たとえば、式 //b[@id="idB"] は、フラグメント <a><c/> 内の 2 番目の b 要素に一致します。attribute="value" を含む任意の要素に対して一致を行うには、XPath 式 //*[attribute="value"] を使用します。

  複数の属性値をフィルタ処理するには、単に複数の属性比較句を連続して使用するだけです。たとえば、式 //b[@c="x"][@d="y"] は、指定した XML フラグメントの任意の場所で発生した要素 に一致します。

  同じ属性が複数の値のいずれかに一致する要素を見つけるには、| 演算子で結合された複数の口ケータを使用します。たとえば、c 属性の値が 23 または 17 であるすべての b 要素に一致するには、式 //b[@c="23"]|//b[@c="17"] を使用します。この目的のために、//b[@c="23" or @c="17"] のように論理 or 演算子を使用することもできます。

  > **注記**
  >
  > or と | の相違点として、or は条件を結合するのに対し、| は結果セットを結合します。

XPath の制限　現在、これらの関数でサポートされている XPath 構文は、次の制限の対象となっています。

- ノードセット間の比較 ('/a/b[@c=@d]' など) はサポートされていません。

- 標準の XPath 比較演算子はすべてサポートされています。(Bug #22823)

- 相対ロケータ式は、ルートノードのコンテキストで解決されます。たとえば、次のようなクエリーと結果を考えてみます。

```
mysql> SELECT ExtractValue(
    ->   '<a><b c="1">X</b><b c="2">Y</b></a>',
    ->    'a/b'
    -> ) AS result;
+--------+
| result |
+--------+
| X Y    |
+--------+
1 row in set (0.03 sec)
```

  この場合は、ロケータ a/b が /a/b に解決されています。

  相対ロケータは、述語内でもサポートされています。次の例では、d[../@c="1"] が /a/b[@c="1"]/d として解決されています。

```
mysql> SELECT ExtractValue(
    ->     '<a>
    ->       <b c="1"><d>X</d></b>
    ->       <b c="2"><d>X</d></b>
    ->     </a>',
    ->     'a/b/d[../@c="1"]')
    -> AS result;
+--------+
| result |
+--------+
| X      |
+--------+
1 row in set (0.00 sec)
```

- スカラー値 (変数参照、リテラル、数字、およびスカラー関数の呼び出しを含む) として評価する式が前に付けられたロケータは許可されず、使用すると エラーが発生します。

- :: 演算子を次のようなノード型と組み合わせることは、サポートされていません。

  - axis::comment()
  - axis::text()
  - axis::processing-instructions()
  - axis::node()

  ただし、次の例に示すように、名前のテスト (axis::name や axis::* など) はサポートされています。

```
mysql> SELECT ExtractValue('<a><b>x</b><c>y</c></a>','/a/child::b');
+-----------------------------------------------------+
| ExtractValue('<a><b>x</b><c>y</c></a>','/a/child::b') |
+-----------------------------------------------------+
| x                                                   |
+-----------------------------------------------------+
1 row in set (0.02 sec)

mysql> SELECT ExtractValue('<a><b>x</b><c>y</c></a>','/a/child::*');
+-----------------------------------------------------+
| ExtractValue('<a><b>x</b><c>y</c></a>','/a/child::*') |
+-----------------------------------------------------+
| x y                                                 |
+-----------------------------------------------------+
1 row in set (0.01 sec)
```

- パスがルート要素を「上方向」に導いている場合は、「上下」の移動がサポートされていません。つまり、現在の要素の 1 つ以上の祖先がルート要素の祖先でもある場合は、指定された要素の祖先の子孫で一致する式を使用できません (Bug #16321 を参照してください)。

- 次の XPath 関数はサポートされていないか、または説明したような既知の問題があります。

  - id()
  - lang()
  - local-name()

  - name()
  - namespace-uri()
  - normalize-space()
  - starts-with()
  - string()
  - substring-after()
  - substring-before()
  - translate()
- 次の軸はサポートされていません。
  - following-sibling
  - following
  - preceding-sibling
  - preceding

ExtractValue() および UpdateXML() への引数として渡される XPath 式の要素セレクタ内には、コロン文字 (「:」) が含まれている可能性があります。これにより、XML 名前空間の表記法を使用しているマークアップとの使用が有効になります。例:

```
mysql> SET @xml = '<a>111<b:c>222<d>333</d><e:f>444</e:f></b:c></a>';
Query OK, 0 rows affected (0.00 sec)

mysql> SELECT ExtractValue(@xml, '//e:f');
+-----------------------------+
| ExtractValue(@xml, '//e:f') |
+-----------------------------+
| 444                         |
+-----------------------------+
1 row in set (0.00 sec)

mysql> SELECT UpdateXML(@xml, '//b:c', '<g:h>555</g:h>');
+-------------------------------------------+
| UpdateXML(@xml, '//b:c', '<g:h>555</g:h>') |
+-------------------------------------------+
| <a>111<g:h>555</g:h></a>                  |
+-------------------------------------------+
1 row in set (0.00 sec)
```

これは、いくつかの点で Apache Xalan およびその他の一部のパーサーで許可されているものと似ていますが、名前空間の制限や namespace-uri() および local-name() 関数の使用を必要とするよりも大幅に単純です。

エラー処理　ExtractValue() と UpdateXML() のどちらの場合でも、使用される XPath 口ケータが有効であり、検索対象の XML が適切にネストされ、閉じられた要素で構成されている必要があります。口ケータが無効な場合は、次のようなエラーが生成されます。

```
mysql> SELECT ExtractValue('<a>c</a><b/>', '/&a');
ERROR 1105 (HY000): XPATH syntax error: '&a'
```

xml_frag が適切にネストされ、閉じられている要素で構成されていない場合は、次の例に示すように、NULL が返され、警告が生成されます。

```
mysql> SELECT ExtractValue('<a>c</a><b', '//a');
+----------------------------------+
| ExtractValue('<a>c</a><b', '//a') |
+----------------------------------+
| NULL                             |
+----------------------------------+
1 row in set, 1 warning (0.00 sec)

mysql> SHOW WARNINGS\G
*************************** 1. row ***************************
  Level: Warning
   Code: 1525
```

```
Message: Incorrect XML value: 'parse error at line 1 pos 11:
        END-OF-INPUT unexpected ('>' wanted)'
1 row in set (0.00 sec)

mysql> SELECT ExtractValue('<a>c</a><b/>', '//a');
+-------------------------------------+
| ExtractValue('<a>c</a><b/>', '//a') |
+-------------------------------------+
| c                                   |
+-------------------------------------+
1 row in set (0.00 sec)
```

> **重要**
>
> UpdateXML() への第 3 引数として使用される置換用の XML は、適切にネストされ、閉じられている要素のみで構成されているかどうかを判断するためにチェックされません。

XPath インジェクション　コードインジェクションは、権限やデータへの不正アクセス権を取得するために、悪意のあるコードがシステムに導入された場合に発生します。これは、ユーザーが入力したデータの型や内容について開発者が行なった想定の悪用に基づいています。これに関しては、XPath も例外ではありません。

この問題が発生する可能性のある一般的なシナリオは、次のような XPath 式を使用して、ログイン名とパスワードの組み合わせを XML ファイル内で見つかったものと一致させることで承認を処理するアプリケーションのケースです。

```
//user[login/text()='neapolitan' and password/text()='1c3cr34m']/attribute::id
```

この XPath 式は、次のような SQL ステートメントと同等です。

```
SELECT id FROM users WHERE login='neapolitan' AND password='1c3cr34m';
```

XPath を使用している PHP アプリケーションでは、次のようにログインプロセスが処理される可能性があります。

```
<?php

 $file     =   "users.xml";

 $login    =   $POST["login"];
 $password =   $POST["password"];

 $xpath = "//user[login/text()=$login and password/text()=$password]/attribute::id";

 if( file_exists($file) )
 {
   $xml = simplexml_load_file($file);

   if($result = $xml->xpath($xpath))
     echo "You are now logged in as user $result[0].";
   else
     echo "Invalid login name or password.";
 }
 else
   exit("Failed to open $file.");

?>
```

入力時にはチェックが実行されません。これは、悪意のあるユーザーがログイン名とパスワードの両方に ' or 1=1 と入力することで、テストを「回避」できることを意味します。その結果、$xpath が次のように評価されます。

```
//user[login/text()='' or 1=1 and password/text()='' or 1=1]/attribute::id
```

角括弧内の式は常に true と評価されるため、事実上、XML ドキュメント内のすべての user 要素の id 属性に一致する次の式と同じです。

```
//user/attribute::id
```

この攻撃を回避する方法の 1 つは、$xpath の定義内に挿入される変数名を単に引用符で囲むだけです。これにより、Web フォームから渡された値が強制的に文字列に変換されます。

```
$xpath = "//user[login/text()='$login' and password/text()='$password']/attribute::id";
```

これは、SQL インジェクション攻撃を回避する際に推奨されることの多い方法と同じです。一般に、XPath インジェクション攻撃を回避するために従うべき方法は、SQL インジェクションを回避するための方法と同じです。

- アプリケーションでは、テストされていないユーザーデータは許可されません。
- ユーザーが送信したすべてのデータの型をチェックします。不正な型のデータは拒否または変換します。
- 数値データに範囲外の値が含まれていないかをテストします。範囲外の値は切り捨てるか、丸めるか、拒否します。文字列に不正な文字が含まれていないかをテストし、不正な文字が含まれる入力は削除するか拒否します。
- 明示的なエラーメッセージは、システムを危険にさらすために使用できる手がかりを未承認ユーザーに与える可能性があるため、出力しないでください。その代わりに、ファイルやデータベーステーブルにログを記録してください。

SQL インジェクション攻撃を使用すればデータベーススキーマに関する情報を取得できるように、XPath インジェクションを使用すれば、Amit Klein 氏の論文『Blind XPath Injection』(PDF ファイル、46K バイト) で説明されているように、XML ファイルをスキャンして構造を明らかにできます。

クライアントに返送される出力をチェックすることも重要です。MySQL の ExtractValue() 関数を使用すると何が発生する可能性があるのかを検討します。

```
mysql> SELECT ExtractValue(
    ->     LOAD_FILE('users.xml'),
    ->     '//user[login/text()="" or 1=1 and password/text()="" or 1=1]/attribute::id'
    -> ) AS id;
+--------------------------------+
| id                             |
+--------------------------------+
| 00327 13579 02403 42354 28570 |
+--------------------------------+
1 row in set (0.01 sec)
```

ExtractValue() は複数の一致を空白で区切られた単一の文字列として返すため、このインジェクション攻撃によって、users.xml 内に含まれるすべての有効な ID が単一の出力行としてユーザーに提供されます。追加の保護手段として、ユーザーに返される前に出力のテストも行うべきです。次に、単純な例を示します。

```
mysql> SELECT @id = ExtractValue(
    ->     LOAD_FILE('users.xml'),
    ->     '//user[login/text()="" or 1=1 and password/text()="" or 1=1]/attribute::id'
    -> );
Query OK, 0 rows affected (0.00 sec)

mysql> SELECT IF(
    ->     INSTR(@id, ' ') = 0,
    ->     @id,
    ->     'Unable to retrieve user ID')
    -> AS singleID;
+---------------------------+
| singleID                  |
+---------------------------+
| Unable to retrieve user ID |
+---------------------------+
1 row in set (0.00 sec)
```

一般に、ユーザーにデータをセキュアに返すためのガイドラインは、ユーザー入力を受け入れるためのガイドラインと同じです。それらは、次のように要約できます。

- 常に、出力データの型および許可される値をテストします。
- 未承認ユーザーがエラーメッセージを表示することを許可しないでください。アプリケーションに関する情報が提供される可能性があり、それを悪用されるおそれがあります。

12.12 ビット関数

表 12.16 ビット単位の関数

名前	説明
BIT_COUNT()	設定されているビット数を返します
&	ビット単位の AND
~	ビットを反転します
|	ビット単位の OR
^	ビット単位の XOR
<<	左シフト

名前	説明
>>	右シフト

MySQL では、演算子の最大範囲が 64 ビットになるように、ビット演算に BIGINT (64 ビット) 算術が使用されます。

- |

  ビット単位の OR:

```
mysql> SELECT 29 | 15;
    -> 31
```

  結果は符合なしの 64 ビット整数です。

- &

  ビット単位の AND:

```
mysql> SELECT 29 & 15;
    -> 13
```

  結果は符合なしの 64 ビット整数です。

- ^

  ビット単位の XOR:

```
mysql> SELECT 1 ^ 1;
    -> 0
mysql> SELECT 1 ^ 0;
    -> 1
mysql> SELECT 11 ^ 3;
    -> 8
```

  結果は符合なしの 64 ビット整数です。

- <<

  longlong (BIGINT) 数値を左にシフトします。

```
mysql> SELECT 1 << 2;
    -> 4
```

  結果は符合なしの 64 ビット整数です。値は 64 ビットに切り捨てられます。特に、シフト数が符号なし 64 ビット数値の幅以上の大きさの場合は、結果がゼロになります。

- >>

  longlong (BIGINT) 数値を右にシフトします。

```
mysql> SELECT 4 >> 2;
    -> 1
```

  結果は符合なしの 64 ビット整数です。値は 64 ビットに切り捨てられます。特に、シフト数が符号なし 64 ビット数値の幅以上の大きさの場合は、結果がゼロになります。

- ~

  すべてのビットを反転します。

```
mysql> SELECT 5 & ~1;
    -> 4
```

  結果は符合なしの 64 ビット整数です。

- BIT_COUNT(N)

  引数 N で設定されているビットの数を返します。

```
mysql> SELECT BIT_COUNT(29), BIT_COUNT(b'101010');
    -> 4, 3
```

12.13 暗号化関数と圧縮関数

表 12.17 暗号化関数

名前	説明
AES_DECRYPT()	AES を使用して復号化します
AES_ENCRYPT()	AES を使用して暗号化します
ASYMMETRIC_DECRYPT() (導入 5.6.21)	秘密鍵または公開鍵を使用して暗号文を復号化します
ASYMMETRIC_DERIVE() (導入 5.6.21)	非対称鍵から対称鍵を導出します
ASYMMETRIC_ENCRYPT() (導入 5.6.21)	秘密鍵または公開鍵を使用してプレーンテキストを暗号化します
ASYMMETRIC_SIGN() (導入 5.6.21)	ダイジェストから署名を生成します
ASYMMETRIC_VERIFY() (導入 5.6.21)	署名がダイジェストと一致することを確認します
COMPRESS()	バイナリ文字列として結果を返します
CREATE_ASYMMETRIC_PRIV_KEY() (導入 5.6.21)	秘密鍵を作成します
CREATE_ASYMMETRIC_PUB_KEY() (導入 5.6.21)	公開鍵を作成します
CREATE_DH_PARAMETERS() (導入 5.6.21)	共有 DH シークレットを生成します
CREATE_DIGEST() (導入 5.6.21)	文字列からダイジェストを生成します
DECODE()	ENCODE() を使用して暗号化された文字列をデコードします
DES_DECRYPT()	文字列を復号化します
DES_ENCRYPT()	文字列を暗号化します
ENCODE()	文字列をエンコードします
ENCRYPT()	文字列を暗号化します
MD5()	MD5 チェックサムを計算します
OLD_PASSWORD() (非推奨 5.6.5)	4.1 より前の PASSWORD 実装の値を返します
PASSWORD()	パスワード文字列を計算して返します
RANDOM_BYTES() (導入 5.6.17)	ランダムなバイトベクトルを返します
SHA1(), SHA()	SHA-1 160 ビットチェックサムを計算します
SHA2()	SHA-2 チェックサムを計算します
UNCOMPRESS()	圧縮された文字列を圧縮解除します
UNCOMPRESSED_LENGTH()	圧縮前の文字列長を返します
VALIDATE_PASSWORD_STRENGTH() (導入 5.6.6)	パスワードの強度を判断します

多くの暗号化関数および圧縮関数では、結果に任意のバイト値が含まれている可能性のある文字列が返されます。これらの結果を格納する場合は、VARBINARY または BLOB バイナリ文字列データ型のカラムを使用します。これにより、末尾の空白を削除したり、文字セットを変換したりするとデータ値が変更される可能性があるという問題を回避できます。たとえば、非バイナリ文字列のデータ型 (CHAR、VARCHAR、TEXT) を使用した場合に、これが発生する可能性があります。

一部の暗号化関数では、ASCII 文字 (MD5()、OLD_PASSWORD()、PASSWORD()、SHA()、SHA1()、SHA2()) の文字列が返されます。MySQL 5.6 では、戻り値は、character_set_connection および collation_connection システム変数で決定された文字セットおよび照合順序を含む非バイナリ文字列です。

MD5() や SHA1() などの関数がバイナリ文字列として 16 進数の文字列を返すバージョンでは、戻り値を大文字に変換したり、大文字と小文字が区別されない方法でそのまま比較したりできません。値を非バイナリ文字列に変換する必要があります。セクション12.10「キャスト関数と演算子」で、バイナリ文字列の変換の説明を参照してください。

アプリケーションで 16 進数の文字列を返す関数 (MD5() や SHA1() など) からの値を格納する場合は、UNHEX() を使用して 16 進表現をバイナリに変換し、その結果を BINARY(N) カラムに格納すれば、より効率的な格納および比較を実現できます。16 進数の各ペアには、バイナリ形式での 1 バイトが必要であるため、N の値は、16 進

文字列の長さによって異なります。N は、MD5() 値の場合は 16、SHA1() の場合は 20 です。SHA2() の場合、N の範囲は、結果の目的のビット長を指定する引数に応じて 28 から 32 までです。

utf8 文字セット (文字ごとに 4 バイト使用されます) が使用されるカラムに値が格納される場合に、16 進文字列を CHAR カラムに格納する際のサイズのペナルティーは最小で 2 回、最大で 8 回です。また、文字列を格納すると、値が大きくなり、文字セットの照合順序ルールを考慮に入れる必要があるため、比較が遅くなります。

アプリケーションで MD5() 文字列値が CHAR(32) カラムに格納されると仮定します。

```
CREATE TABLE md5_tbl (md5_val CHAR(32), ...);
INSERT INTO md5_tbl (md5_val, ...) VALUES(MD5('abcdef'), ...);
```

16 進文字列をよりコンパクトな形式に変換するには、次のように、代わりに UNHEX() および BINARY(16) が使用されるようにアプリケーションを変更します。

```
CREATE TABLE md5_tbl (md5_val BINARY(16), ...);
INSERT INTO md5_tbl (md5_val, ...) VALUES(UNHEX(MD5('abcdef')), ...);
```

ハッシュ関数が 2 つの異なる入力値に同じ値を生成するという非常にまれなケースに対応できるように、アプリケーションが準備されるはずです。競合を検出可能にする方法の 1 つは、ハッシュカラムを主キーにすることです。

> **注記**
>
> MD5 および SHA-1 アルゴリズムの悪用が知られています。代わりに、このセクションで説明するその他の暗号化関数 (SHA2() など) のいずれか使用することを検討してください。

> **注意**
>
> SSL 接続が使用されていなければ、暗号化関数への引数として指定されたパスワードやその他の機密の値は、プレーンテキストで MySQL サーバーに送信されます。また、このような値は、書き込み先の MySQL ログにも表示されます。このようなタイプの露出を回避するために、アプリケーションはクライアント側で機密の値を暗号化してから、サーバーに送信できます。同じ考慮事項が暗号化鍵にも適用されます。これらの露出を回避するために、アプリケーションはストアドプロシージャーを使用して、サーバー側で値を暗号化および復号化できます。

- AES_DECRYPT(crypt_str,key_str[,init_vector])

  この関数は、公式の AES (Advanced Encryption Standard) アルゴリズムを使用してデータを復号化します。詳細は、AES_ENCRYPT() の説明を参照してください。

  MySQL 5.6.17 の時点では、オプションの初期化ベクトル引数 init_vector を使用できます。そのバージョンでは、AES_DECRYPT() が使用されるステートメントはステートメントベースのレプリケーションに対して安全ではなく、クエリーキャッシュ内に格納できません。

- AES_ENCRYPT(str,key_str[,init_vector])

  AES_ENCRYPT() および AES_DECRYPT() では、AES (Advanced Encryption Standard) アルゴリズム (以前は「Rijndael」と呼ばれていました) を使用したデータの暗号化および復号化が実装されます。AES の標準では、さまざまな鍵の長さが許可されます。デフォルトでは、これらの関数で鍵の長さが 128 ビットの AES が実装されます。MySQL 5.6.17 の時点では、あとで説明するように、使用できる鍵の長さは 196 または 256 ビットです。鍵の長さは、パフォーマンスとセキュリティーの間でのトレードオフです。

  AES_ENCRYPT() は、鍵文字列 key_str を使用して文字列 str を暗号化し、暗号化された出力を含むバイナリ文字列を返します。AES_DECRYPT() は、鍵文字列 key_str を使用して暗号化された文字列 crypt_str を復号化し、元のプレーンテキスト文字列を返します。関数引数のいずれかが NULL の場合は、関数で NULL が返されます。

  str および crypt_str 引数は任意の長さにすることができ、AES などのブロックベースのアルゴリズムによって、必要に応じてブロックの倍数になるように、自動的にパディングが str に追加されます。このパディングは、AES_DECRYPT() 関数によって自動的に削除されます。crypt_str の長さは、次の公式を使用して計算できます。

  ```
  16 * (trunc(string_length / 16) + 1)
  ```

  鍵の長さが 128 ビットの場合、key_str 引数に鍵を渡すもっともセキュアな方法は、完全にランダムな 128 ビット値を作成し、それをバイナリ値として渡すことです。例:

```
INSERT INTO t
VALUES (1,AES_ENCRYPT('text',UNHEX('F3229A0B371ED2D9441B830D21A390C3')));
```

パスフレーズを使用すると、パスフレーズをハッシュ化することで AES 鍵を生成できます。例:

```
INSERT INTO t VALUES (1,AES_ENCRYPT('text', SHA2('My secret passphrase',512)));
```

パスワードまたはパスフレーズは直接 crypt_str に渡さず、最初にハッシュ化してください。このドキュメントの以前のバージョンでは、従来のアプローチが提案されていましたが、ここで示す例の方がセキュアであるため、推奨されなくなりました。

AES_DECRYPT() で無効な日付または不正なパディングが検出された場合は、NULL が返されます。ただし、入力データまたは鍵が無効になっている場合は、AES_DECRYPT() で NULL 以外の値 (ごみの可能性もあります) が返される可能性があります。

MySQL 5.6.17 の時点では、AES_ENCRYPT() および AES_DECRYPT() でブロック暗号化モードの制御が許可され、オプションの init_vector 初期化ベクトル引数が指定されます。

- block_encryption_mode システム変数は、ブロックベースの暗号化アルゴリズムのモードを制御します。そのデフォルト値は、128 ビットの鍵の長さと ECB モードを使用した暗号化を表す aes-128-ecb です。この変数で許可されている値については、セクション5.1.4「サーバーシステム変数」を参照してください。
- オプションの init_vector 引数では、必要とするブロック暗号化モードに対応する初期化ベクトルが提供されます。

オプションの init_vector 引数が必要なモードでは、16 バイト以上の長さにする必要があります (16 を超えるバイトは無視されます)。init_vector が欠落している場合は、エラーが発生します。

init_vector が必要ないモードでは、これが無視され、指定されている場合は警告が生成されます。

RANDOM_BYTES(16) を呼び出すと、初期化ベクトルに使用されるバイトのランダム文字列を生成できます。初期化ベクトルが必要な暗号化モードでは、暗号化および復号化でも同じベクトルを使用する必要があります。

```
mysql> SET block_encryption_mode = 'aes-256-cbc';
mysql> SET @key_str = SHA2('My secret passphrase',512);
mysql> SET @init_vector = RANDOM_BYTES(16);
mysql> SET @crypt_str = AES_ENCRYPT('text',@key_str,@init_vector);
mysql> SELECT AES_DECRYPT(@crypt_str,@key_str,@init_vector);
+-------------------------------------------------+
| AES_DECRYPT(@crypt_str,@key_str,@init_vector) |
+-------------------------------------------------+
| text                                            |
+-------------------------------------------------+
```

次の表には、許可されている各ブロック暗号化モード、サポートされている SSL ライブラリ、および初期化ベクトル引数が必須であるかどうかを一覧表示します。

ブロック暗号化モード	モードがサポートされている SSL ライブラリ	初期化ベクトルが必要
ECB	OpenSSL、yaSSL	いいえ
CBC	OpenSSL、yaSSL	はい
CFB1	OpenSSL	はい
CFB8	OpenSSL	はい
CFB128	OpenSSL	はい
OFB	OpenSSL	はい

MySQL 5.6.17 では、AES_ENCRYPT() または AES_DECRYPT() が使用されるステートメントはステートメントベースのレプリケーションに対して安全ではなく、クエリーキャッシュ内に格納できません。

- COMPRESS(string_to_compress)

  文字列を圧縮し、その結果をバイナリ文字列として返します。この関数を使用するには、MySQL が zlib などの圧縮ライブラリを使用してコンパイルされている必要があります。そうでない場合、戻り値は常に NULL になります。圧縮された文字列は、UNCOMPRESS() を使用して圧縮解除できます。

```
mysql> SELECT LENGTH(COMPRESS(REPEAT('a',1000)));
    -> 21
mysql> SELECT LENGTH(COMPRESS(''));
    -> 0
mysql> SELECT LENGTH(COMPRESS('a'));
    -> 13
mysql> SELECT LENGTH(COMPRESS(REPEAT('a',16)));
    -> 15
```

  圧縮された文字列の内容は、次の方法で格納されます。

  - 空の文字列は、空の文字列として格納されます。
  - 空以外の文字列は、4 バイトの長さの圧縮されていない文字列として格納され (低いバイトが 1 番目)、そのあとに圧縮された文字列が続きます。文字列が空白文字で終わる場合は、結果が CHAR または VARCHAR カラムに格納されていても、末尾の空白文字が削除されるという問題が回避されるように、「.」 文字が追加されます。(ただし、CHAR や VARCHAR などの非バイナリ文字列のデータ型を使用して圧縮された文字列を格納すると、文字セットの変換が発生する可能性があるため、いずれにしても推奨されません。代わりに、VARBINARY または BLOB バイナリ文字列のカラムを使用してください。)

- DECODE(crypt_str,pass_str)

  暗号化された文字列 crypt_str は、パスワードとして pass_str を使用して復号化します。crypt_str は、ENCODE() から返された文字列にするようにしてください。

- DES_DECRYPT(crypt_str[,key_str])

  DES_ENCRYPT() を使用して暗号化された文字列を復号化します。エラーが発生した場合、この関数は NULL を返します。

  この関数は、MySQL が SSL サポートで構成されている場合のみ機能します。セクション6.3.10「セキュアな接続のための SSL の使用」を参照してください。

  key_str 引数が指定されていない場合、DES_DECRYPT() は暗号化された文字列の最初のバイトを調査して、元の文字列を暗号化したときに使用した DES 鍵番号を特定してから、DES 鍵ファイルからメッセージを復号化するための鍵を読み取ります。これが機能するには、ユーザーが SUPER 権限を持っている必要があります。鍵ファイルは、--des-key-file サーバーオプションを使用して指定できます。

  この関数を key_str 引数に渡すと、その文字列がメッセージを復号化するための鍵として使用されます。

  crypt_str 引数が暗号化された文字列として表示されない場合は、MySQL では指定された crypt_str が返されます。

- DES_ENCRYPT(str[,{key_num|key_str}])

  Triple-DES アルゴリズムを使用して、指定された鍵で文字列を暗号化します。

  この関数は、MySQL が SSL サポートで構成されている場合のみ機能します。セクション6.3.10「セキュアな接続のための SSL の使用」を参照してください。

  使用する暗号化鍵は、指定されていれば、DES_ENCRYPT() への 2 番目の引数に基づいて選択されます。引数を付けない場合は、DES 鍵ファイルの最初の鍵が使用されます。key_num 引数を付けると、DES 鍵ファイルの指定された鍵番号 (0 - 9) が使用されます。key_str 引数を付けた場合は、指定された鍵文字列を使用して str が暗号化されます。

  鍵ファイルは、--des-key-file サーバーオプションを使用して指定できます。

  返される文字列は、最初の文字が CHAR(128 | key_num) であるバイナリ文字列です。エラーが発生した場合、DES_ENCRYPT() は NULL を返します。

  暗号化された鍵を認識しやすくするために、128 が追加されます。文字列鍵を使用する場合、key_num は 127 です。

  結果の文字列の長さは、次の公式で指定されます。

```
new_len = orig_len + (8 - (orig_len % 8)) + 1
```

DES 鍵ファイルの各行の書式は、次のとおりです。

```
key_num des_key_str
```

各 key_num 値は、0 から 9 までの範囲内の数字にする必要があります。ファイル内の行は、任意の順序で指定できます。des_key_str は、メッセージを暗号化する際に使用される文字列です。数字と鍵の間には、少なくとも 1 つの空白文字を入れるようにしてください。最初の鍵は、DES_ENCRYPT() への鍵引数を指定しなかった場合に使用されるデフォルトのキーです。

FLUSH DES_KEY_FILE ステートメントで鍵ファイルから新しいキー値を読み取るように、MySQL に指示できます。これには、RELOAD 権限が必須です。

デフォルト鍵のセットを持つ利点は、これらの値を復号化する権利をエンドユーザーに付与せずに、暗号化されたカラム値の有無をチェックする方法がアプリケーションに提供されることです。

```
mysql> SELECT customer_address FROM customer_table
    > WHERE crypted_credit_card = DES_ENCRYPT('credit_card_number');
```

- ENCODE(str,pass_str)

  パスワードとして pass_str を使用して、str を暗号化します。結果は、str と同じ長さのバイナリ文字列です。結果を復号化するには、DECODE() を使用します。

  ENCODE() 関数は使用されるべきではなくなっています。まだ ENCODE() を使用する必要がある場合は、リスクを軽減するために、一緒に salt 値を使用する必要があります。例:

  ```
  ENCODE('plaintext', CONCAT('my_random_salt','my_secret_password'))
  ```

  パスワードを更新するたびに、新しいランダムな salt 値を使用する必要があります。

- ENCRYPT(str[,salt])

  Unix crypt() システム呼び出しを使用して str を暗号化し、バイナリ文字列を返します。salt 引数は、2 つ以上の文字を含む文字列にする必要があります。それ以外の場合は、結果が NULL になります。salt 引数が指定されていない場合は、ランダムな値が使用されます。

  ```
  mysql> SELECT ENCRYPT('hello');
        -> 'VxuFAJXVARROc'
  ```

  少なくとも一部のシステムでは、ENCRYPT() は str の最初の 8 文字以外のすべてを無視します。この動作は、ベースとなる crypt() システム呼び出しの実装によって決まります。

  システム呼び出しではゼロバイトで終了する文字列が要求されるため、ucs2、utf16、utf16le、または utf32 マルチバイト文字セットを含む ENCRYPT() の使用は推奨されません。

  システムで crypt() を使用できない場合 (Windows の場合など)、ENCRYPT() は常に NULL を返します。

- MD5(str)

  文字列の MD5 128 ビットチェックサムを計算します。この値は、32 桁の 16 進数の文字列として返されます。引数が NULL だった場合は、NULL になります。たとえば、戻り値をハッシュ鍵として使用できます。効率的なハッシュ値の格納については、このセクションの冒頭で示した注記を参照してください。

  戻り値は、接続文字セット内の非バイナリ文字列です。

  ```
  mysql> SELECT MD5('testing');
        -> 'ae2b1fca515949e5d54fb22b8ed95575'
  ```

  これは、「RSA Data Security, Inc. MD5 Message-Digest Algorithm」 です。

  このセクションの冒頭で示した MD5 アルゴリズムに関する注記を参照してください。

- OLD_PASSWORD(str)

  OLD_PASSWORD() は、セキュリティーを改善するために、MySQL 4.1 で PASSWORD() の実装が変更されたときに追加されました。OLD_PASSWORD() は、PASSWORD() の 4.1 よりも前の実装の値を文字列として返します。これは、バージョン 5.6 の MySQL サーバーに接続する必要のある 4.1 よりも前のクライアントを

ロックアウトせずに、それらのクライアントのパスワードをリセットするよう許可するためです。セクション 6.1.2.4「MySQL でのパスワードハッシュ」を参照してください。

戻り値は、接続文字セット内の非バイナリ文字列です。

> **注記**
>
> 4.1 より前のハッシュ方式を使用するパスワードはネイティブのパスワードハッシュ方式を使用するパスワードよりもセキュアでないため、使用しないようにしてください。4.1 よりも前のパスワードは非推奨であり、これらのサポートは今後の MySQL リリースで削除される予定です。最終的に、OLD_PASSWORD() も非推奨になりました。

- PASSWORD(str)

平文のパスワード str から計算してハッシュ化されたパスワード文字列を返します。戻り値は、接続文字セット内の非バイナリ文字列です。引数が NULL の場合は、NULL になります。この関数は、mysql.user 付与テーブルに格納する MySQL パスワードを暗号化する際に、サーバーで使用されるアルゴリズムへの SQL インタフェースです。

old_passwords システム変数は、PASSWORD() 関数で使用されるパスワードのハッシュ化方法を制御します。これは、IDENTIFIED BY 句を使用してパスワードを指定する CREATE USER および GRANT ステートメントによって実行されるパスワードハッシュにも影響します。

次の表は、old_passwords の許可される値、それぞれの値に対するパスワードハッシュ方式、およびそれぞれの方式でハッシュされたパスワードを使用する認証プラグインを示します。これらの値は MySQL 5.6.6 以降で許可されます。5.6.6 より前では、許可される値は 0 (または OFF) および 1 (または ON) です。

値	パスワードハッシュ方式	関連付けられた認証プラグイン
0	MySQL 4.1 ネイティブハッシュ	mysql_native_password
1	4.1 以前の (「古い」) ハッシュ	mysql_old_password
2	SHA-256 ハッシュ	sha256_password

old_passwords=1 の場合、PASSWORD(str) は OLD_PASSWORD(str) と同じ値を返します。後者の関数は old_passwords の値によって影響を受けません。

```
mysql> SET old_passwords = 0;
mysql> SELECT PASSWORD('mypass'), OLD_PASSWORD('mypass');
+-------------------------------------------+--------------------------+
| PASSWORD('mypass')                        | OLD_PASSWORD('mypass') |
+-------------------------------------------+--------------------------+
| *6C8989366EAF75BB670AD8EA7A7FC1176A95CEF4 | 6f8c114b58f2ce9e       |
+-------------------------------------------+--------------------------+

mysql> SET old_passwords = 1;
mysql> SELECT PASSWORD('mypass'), OLD_PASSWORD('mypass');
+--------------------+--------------------------+
| PASSWORD('mypass') | OLD_PASSWORD('mypass') |
+--------------------+--------------------------+
| 6f8c114b58f2ce9e   | 6f8c114b58f2ce9e       |
+--------------------+--------------------------+
```

SHA-256 パスワードのハッシュ化 (old_passwords=2) では、PASSWORD() からの結果を非決定的にするランダムな salt 値が使用されます。その結果、この関数が使用されるステートメントはステートメントベースのレプリケーションで安全ではなく、クエリーキャッシュ内に格納できません。

PASSWORD() で実行される暗号化は一方向です (可逆ではありません)。暗号化のタイプは、Unix パスワードに使用されるものと同じではありません。そのタイプを使用するには、ENCRYPT() を使用します。

> **注記**
>
> PASSWORD() 関数は、MySQL サーバーの認証システムで使用されます。独自のアプリケーションでは、使用しないようにしてください。そのためには、代わりに MD5() または SHA2() を使用してください。アプリケーションでのセキュアなパスワードの処理および認証セキュリティーについての詳細は、RFC 2195 のセクショ

> ン 2 (「Challenge-Response Authentication Mechanism (CRAM)」)も参照してください。

> 注記
>
> 4.1 より前のハッシュ方式を使用するパスワードはネイティブのパスワードハッシュ方式を使用するパスワードよりもセキュアでないため、使用しないようにしてください。4.1 よりも前のパスワードは非推奨であり、これらのサポートは今後の MySQL リリースで削除される予定です。その結果、PASSWORD() で 4.1 以前のパスワードハッシュを生成する old_passwords=1 も非推奨となります。アカウントのアップグレード手順については、セクション6.3.8.3「4.1 よりも前のパスワードハッシュ方式と mysql_old_password プラグインからの移行」を参照してください。

> 注意
>
> 状況によっては、サーバーログまたはクライアント側の ~/.mysql_history などの履歴ファイルに、PASSWORD() を呼び出すステートメントが記録される可能性があります。これは、その情報への読み取りアクセス権を持っているユーザーならだれでも、平文のパスワードを読み取る可能性があることを意味します。これがサーバーログで発生する条件およびこれを制御する方法については、セクション6.1.2.3「パスワードおよびロギング」を参照してください。クライアント側のロギングに関する同様の情報については、セクション4.5.1.3「mysql のロギング」を参照してください。

- RANDOM_BYTES(len)

  この関数は、SSL ライブラリ (OpenSSL または yaSSL) の乱数ジェネレータを使用して生成されたランダムな len バイトのバイナリ文字列を返します。len で許可されている値は、1 から 1024 までの範囲内です。値がこの範囲外の場合、RANDOM_BYTES() は警告を生成し、NULL を返します。

  RANDOM_BYTES() を使用すると、AES_DECRYPT() および AES_ENCRYPT() 関数に初期化ベクトルを提供できます。このコンテキストで使用するには、len を 16 以上にする必要があります。さらに大きい値も許可されますが、16 を超えるバイトは無視されます。

  RANDOM_BYTES() は、その結果を非決定的にするランダムな値を生成します。その結果、この関数が使用されるステートメントはステートメントベースのレプリケーションに対して安全ではなく、クエリーキャッシュ内に格納できません。

  この関数は、MySQL 5.6.17 の時点で使用可能です。

- SHA1(str)、SHA(str)

  RFC 3174 (Secure Hash Algorithm) で説明されているように、文字列の SHA-1 160 ビットチェックサムを計算します。この値は、40 桁の 16 進数の文字列として返されます。引数が NULL だった場合は、NULL になります。この関数を使用する一例として、ハッシュ鍵が考えられます。効率的なハッシュ値の格納については、このセクションの冒頭で示した注記を参照してください。SHA1() は、パスワードを格納する暗号化関数としても使用できます。SHA() は SHA1() のシノニムです。

  戻り値は、接続文字セット内の非バイナリ文字列です。

```
mysql> SELECT SHA1('abc');
    -> 'a9993e364706816aba3e25717850c26c9cd0d89d'
```

  SHA1() は MD5() と同等ですが、暗号化に関してはよりセキュアであると考えられます。ただし、このセクションの冒頭で示した MD5 と SHA-1 アルゴリズムに関する注記を参照してください。

- SHA2(str, hash_length)

  SHA-2 ファミリのハッシュ関数 (SHA-224、SHA-256、SHA-384、および SHA-512) を計算します。1 番目の引数は、ハッシュ化される平文の文字列です。2 番目の引数には、結果の目的のビット長が指定されます。値は、224、256、384、512、または 0 (256 と同等です) にする必要があります。引数のいずれかが NULL の場合や、ハッシュの長さが許可される値のいずれでもない場合は、戻り値が NULL になります。それ以外の場合は、関数の結果が目的のビット数が含まれるハッシュ値になります。効率的なハッシュ値の格納については、このセクションの冒頭で示した注記を参照してください。

  戻り値は、接続文字セット内の非バイナリ文字列です。

```
mysql> SELECT SHA2('abc', 224);
```

```
    -> '23097d223405d8228642a477bda255b32aadbce4bda0b3f7e36c9da7'
```

この関数は、MySQL が SSL サポートで構成されている場合のみ機能します。セクション6.3.10「セキュアな接続のための SSL の使用」を参照してください。

SHA2() は MD5() や SHA1() よりも、暗号化に関してはよりセキュアであると考えられます。

- UNCOMPRESS(string_to_uncompress)

  COMPRESS() 関数で圧縮された文字列を圧縮解除します。引数が圧縮された値でない場合は、結果が NULL になります。この関数を使用するには、MySQL が zlib などの圧縮ライブラリを使用してコンパイルされている必要があります。そうでない場合、戻り値は常に NULL になります。

```
mysql> SELECT UNCOMPRESS(COMPRESS('any string'));
    -> 'any string'
mysql> SELECT UNCOMPRESS('any string');
    -> NULL
```

- UNCOMPRESSED_LENGTH(compressed_string)

  圧縮された文字列が圧縮される前の長さを返します。

```
mysql> SELECT UNCOMPRESSED_LENGTH(COMPRESS(REPEAT('a',30)));
    -> 30
```

- VALIDATE_PASSWORD_STRENGTH(str)

  平文のパスワードを表す引数が指定された場合、この関数はパスワードの強度を示す整数を返します。戻り値は、0 (弱) から 100 (強) までの範囲内です。

  パスワードは、一段と厳密になったテストの対象であり、戻り値は、次の表に示すように、どのテストに合格したのかを示します。

パスワードテスト	戻り値
長さ < 4	0
長さ ≥ 4 および < validate_password_length	25
ポリシー 1 を満たす (LOW)	50
ポリシー 2 を満たす (MEDIUM)	75
ポリシー 3 を満たす (STRONG)	100

  VALIDATE_PASSWORD_STRENGTH() によるパスワードの評価は、validate_password プラグインで実行されます。このプラグインがインストールされていない場合、関数は常に 0 を返します。validate_password プラグインのインストールについては、セクション6.1.2.6「パスワード検証プラグイン」を参照してください。パスワードのテストに影響を与えるパラメータを確認または構成するには、validate_password プラグインで実装されているシステム変数をチェックまたは設定します。パスワード検証プラグインのオプションおよび変数を参照してください。

  この関数は、MySQL 5.6.6 で追加されました。

12.14 情報関数

表 12.18 情報関数

名前	説明
BENCHMARK()	式を繰り返し実行します
CHARSET()	引数の文字セットを返します
COERCIBILITY()	文字列引数の照合順序強制性値を返します
COLLATION()	文字列引数の照合順序を返します
CONNECTION_ID()	接続のための接続 ID (スレッド ID) を返します
CURRENT_USER(), CURRENT_USER	認証済みユーザー名とホスト名
DATABASE()	デフォルト (現在) のデータベース名を返します
FOUND_ROWS()	LIMIT 句付き SELECT で、LIMIT 句がない場合に戻される可能性がある行の数です

名前	説明
LAST_INSERT_ID()	前回の INSERT での AUTOINCREMENT カラムの値です
ROW_COUNT()	更新された行数
SCHEMA()	DATABASE() のシノニムです
SESSION_USER()	USER() のシノニムです
SYSTEM_USER()	USER() のシノニムです
USER()	ユーザー名と、クライアントによって提供されるホスト名です
VERSION()	MySQL サーバーのバージョンを示す文字列を返します

- BENCHMARK(count,expr)

  BENCHMARK() 関数は、式 expr を count の回数だけ繰り返し実行します。MySQL による式の処理速度を計測する際に使用される場合もあります。結果の値は常に 0 になります。この使用目的は、mysql クライアント内から、クエリーの実行時間をレポートすることです。

```
mysql> SELECT BENCHMARK(1000000,ENCODE('hello','goodbye'));
+----------------------------------------------+
| BENCHMARK(1000000,ENCODE('hello','goodbye')) |
+----------------------------------------------+
|                                            0 |
+----------------------------------------------+
1 row in set (4.74 sec)
```

  レポートされる時間は、クライアント側での経過時間であり、サーバー側での CPU 時間ではありません。BENCHMARK() を複数回実行し、サーバーマシン上の負荷量について結果を解釈することをお勧めします。

  BENCHMARK() の目的は、スカラー式の実行時パフォーマンスを測定することです。これにより、その使用方法や結果の解釈方法について、重要ないくつかの推測が提供されます。

  - スカラー式しか使用できません。式をサブクエリーにすることはできますが、単一のカラムおよび最大でも単一の行が返される必要があります。たとえば、テーブル t に複数のカラムや複数の行が含まれていると、BENCHMARK(10, (SELECT * FROM t)) は失敗します。
  - SELECT expr ステートメントを N 回実行する場合と、SELECT BENCHMARK(N, expr) を実行する場合とでは、発生するオーバーヘッドの量が異なります。この 2 つは非常に異なる実行プロファイルを持つため、両者の所要時間は同一になりません。前者では、パーサー、オプティマイザ、テーブルロック、および実行時評価がそれぞれ N 回ずつ発生します。後者では、実行時評価のみが N 回発生し、その他のすべてのコンポーネントは 1 回だけ発生します。割り当て済みのメモリー構造体は再使用され、集約関数で評価済みの結果をローカルキャッシュに入れるなどの実行時最適化によって、結果が変わる可能性もあります。したがって、BENCHMARK() を使用して、実行時コンポーネントに高い重みを付加し、ネットワーク、パーサー、オプティマイザなどで導入された「ノイズ」を削除することで、そのコンポーネントのパフォーマンスが測定されます。

- CHARSET(str)

  文字列引数の文字セットを返します。

```
mysql> SELECT CHARSET('abc');
    -> 'latin1'
mysql> SELECT CHARSET(CONVERT('abc' USING utf8));
    -> 'utf8'
mysql> SELECT CHARSET(USER());
    -> 'utf8'
```

- COERCIBILITY(str)

  文字列引数の照合順序強制性値を返します。

```
mysql> SELECT COERCIBILITY('abc' COLLATE latin1_swedish_ci);
    -> 0
mysql> SELECT COERCIBILITY(USER());
    -> 3
mysql> SELECT COERCIBILITY('abc');
```

```
    -> 4
```

戻り値の意味は、次の表に示すとおりです。値が低いほど、優先順位は高くなります。

型変換属性	意味	例
0	明示的な照合順序	COLLATE 句の値
1	照合順序なし	さまざまな照合順序との文字列の連結
2	暗黙的な照合順序	カラム値、ストアドルーチンパラメータ、またはローカル変数
3	系統定数	USER() の戻り値
4	型変換可能	リテラル文字列
5	無視可能	NULL または NULL から派生した式

- COLLATION(str)

  文字列引数の照合順序を返します。

```
mysql> SELECT COLLATION('abc');
    -> 'latin1_swedish_ci'
mysql> SELECT COLLATION(_utf8'abc');
    -> 'utf8_general_ci'
```

- CONNECTION_ID()

  接続用の接続 ID (スレッド ID) を返します。すべての接続は、現在接続されているクライアントのセット間で一意の ID を持っています。

  CONNECTION_ID() で返される値の型は、INFORMATION_SCHEMA.PROCESSLIST テーブルの ID カラム、SHOW PROCESSLIST 出力の Id カラム、およびパフォーマンススキーマ threads テーブルの PROCESSLIST_ID カラムに表示される値と同じです。

```
mysql> SELECT CONNECTION_ID();
    -> 23786
```

- CURRENT_USER, CURRENT_USER()

  現在のクライアントを認証する際にサーバーで使用された MySQL アカウントを表すユーザー名とホスト名の組み合わせを返します。このアカウントで、アクセス権限が決まります。戻り値は、utf8 文字セット内の文字列です。

  CURRENT_USER() の値は、USER() の値とは異なる可能性があります。

```
mysql> SELECT USER();
    -> 'davida@localhost'
mysql> SELECT * FROM mysql.user;
ERROR 1044: Access denied for user ''@'localhost' to
database 'mysql'
mysql> SELECT CURRENT_USER();
    -> '@localhost'
```

この例は、クライアントが davida のユーザー名を指定 (USER() 関数の値で指定されます) したが、サーバーは匿名のユーザーアカウント (CURRENT_USER() 値の空のユーザー名部分に表示されます) を使用してクライア

ントを認証したことを示しています。これが発生する原因として、davida の付与テーブルにアカウントが一覧表示されていないことが考えられます。

ストアドプログラムまたはビューでは、SQL SECURITY INVOKER 特性で定義されていなければ、CURRENT_USER() はオブジェクトを定義したユーザー (その DEFINER 値で指定されます) のアカウントを返します。後者の場合、CURRENT_USER() はオブジェクトを呼び出したユーザーを返します。

トリガーおよびイベントには、SQL SECURITY 特性を定義するためのオプションがありません。したがって、このようなオブジェクトの場合、CURRENT_USER() はオブジェクトを定義したユーザーのアカウントを返します。呼び出したユーザーを返すには、USER() または SESSION_USER() を使用します。

次のステートメントでは、影響を受けるユーザーや定義したユーザーの名前 (ホストの可能性もあります) の代わりに、CURRENT_USER() 関数を使用することがサポートされています。このような場合、必要に応じて CURRENT_USER() が拡張されます。

- DROP USER
- RENAME USER
- GRANT
- REVOKE
- CREATE FUNCTION
- CREATE PROCEDURE
- CREATE TRIGGER
- CREATE EVENT
- CREATE VIEW
- ALTER EVENT
- ALTER VIEW
- SET PASSWORD

このような CURRENT_USER() の拡張がさまざまな MySQL 5.6 リリースのレプリケーションで持つ意味については、セクション17.4.1.7「CURRENT_USER() のレプリケーション」を参照してください。

- DATABASE()

デフォルト (現在) のデータベース名を utf8 文字セット内の文字列として返します。デフォルトのデータベースがない場合は、DATABASE() は NULL を返します。ストアドルーチン内では、デフォルトのデータベースはルーチンが関連付けられたデータベースですが、これは呼び出し元のコンテキストでのデフォルトのデータベースと同じであるとはかぎりません。

```
mysql> SELECT DATABASE();
       -> 'test'
```

デフォルトのデータベースがない場合は、DATABASE() は NULL を返します。

- FOUND_ROWS()

サーバーからクライアントに返される行の数を制限するために、SELECT ステートメントに LIMIT 句が含まれている場合があります。場合によっては、ステートメントを再度実行せずに、LIMIT を付けなかった場合にステートメントで返されるはずの行数を知っておくことが望ましいことがあります。この行数を取得するには、SELECT ステートメントに SQL_CALC_FOUND_ROWS オプションを付けてから、FOUND_ROWS() を呼び出します。

```
mysql> SELECT SQL_CALC_FOUND_ROWS * FROM tbl_name
    -> WHERE id > 100 LIMIT 10;
```

```
mysql> SELECT FOUND_ROWS();
```

2 番目の SELECT は、1 番目の SELECT を LIMIT 句なしで記述した場合に返される行数を示す数字を返します。

最近成功した SELECT ステートメントに SQL_CALC_FOUND_ROWS オプションを付けなければ、FOUND_ROWS() は、そのステートメントで返された結果セットの行数を返します。ステートメントに LIMIT 句が含まれている場合、FOUND_ROWS() はその制限値以下の行数を返します。たとえば、ステートメントに LIMIT 10 または LIMIT 50, 10 が含まれている場合、FOUND_ROWS() はそれぞれ 10 と 60 を返します。

FOUND_ROWS() から取得できる行数は一時的なもので、SELECT SQL_CALC_FOUND_ROWS ステートメントのあとに、このステートメントを発行しても取得できるようには設計されていません。あとで値を参照する必要がある場合は、保存してください。

```
mysql> SELECT SQL_CALC_FOUND_ROWS * FROM ... ;
mysql> SET @rows = FOUND_ROWS();
```

SELECT SQL_CALC_FOUND_ROWS を使用している場合は、MySQL では完全な結果セット内の行数を計算する必要があります。ただし、結果セットはクライアントに送信される必要がないため、LIMIT なしでクエリーを再度実行するよりも速くなります。

SQL_CALC_FOUND_ROWS および FOUND_ROWS() は、クエリーで返される行数を制限するが、クエリーを再度実行しないで完全な結果セット内の行数を確認する必要がある状況でも役立ちます。例として、検索結果のほかのセクションを表示するページへのリンクを含むページが表示される Web スクリプトがあります。FOUND_ROWS() を使用すると、残りの結果を表示するために必要なその他のページ数を確認できます。

SQL_CALC_FOUND_ROWS および FOUND_ROWS() を使用すると、UNION の複数箇所で LIMIT が発生する可能性があるため、単純な SELECT ステートメントよりも、UNION ステートメントで使用した方が複雑になります。これは、UNION 内の個々の SELECT ステートメントに適用される場合と、UNION の結果全体にグローバルに適用される場合があります。

UNION で SQL_CALC_FOUND_ROWS を使用する目的は、グローバルな LIMIT なしで返される行数を返すことです。UNION で SQL_CALC_FOUND_ROWS を使用する条件は、次のとおりです。

- UNION の 1 番目の SELECT に、SQL_CALC_FOUND_ROWS キーワードが表示される必要があります。
- FOUND_ROWS() の値は、UNION ALL が使用されている場合にのみ正確です。ALL なしで UNION が使用される場合は、重複の削除が発生し、FOUND_ROWS() の値が単なる近似値になります。
- UNION で LIMIT が表示されない場合は、SQL_CALC_FOUND_ROWS が無視され、UNION を処理するために作成された一時テーブル内の行数が返されます。

ここで説明した以外のケースでは、FOUND_ROWS() の動作 (エラーが発生して SELECT ステートメントに失敗したあとの値など) が定義されません。

> **重要**
>
> ステートメントベースのレプリケーションでは、確実に FOUND_ROWS() をレプリケートすることはできません。行ベースのレプリケーションを使用すると、この関数は自動的にレプリケートされます。

- LAST_INSERT_ID()、LAST_INSERT_ID(expr)

LAST_INSERT_ID() に引数を付けない場合は、最近実行された INSERT ステートメントの結果として、最初に自動的に生成され、正常に AUTO_INCREMENT カラムに挿入された値を表す 64 ビット値が返されます。この値の型は、MySQL 5.6.9 の時点では BIGINT UNSIGNED、それよりも前では BIGINT (符号付き) です。正常に挿入された行がない場合は、LAST_INSERT_ID() の値は未変更のままです。

LAST_INSERT_ID() に引数を付けない場合は、MySQL 5.6.9 の時点では符号なし整数、それよりも前では符号付き整数が返されます。

たとえば、AUTO_INCREMENT 値を生成する行を挿入したあとは、次のようにして値を取得できます。

```
mysql> SELECT LAST_INSERT_ID();
```

```
    -> 195
```

現在実行中のステートメントによって、LAST_INSERT_ID() の値は影響を受けません。1 つのステートメントで AUTO_INCREMENT 値を生成してから、独自の AUTO_INCREMENT カラムを含むテーブルに行を挿入する複数行の INSERT ステートメントで LAST_INSERT_ID() を参照すると仮定します。LAST_INSERT_ID() の値は、2 番目のステートメントでも未変更のままです。2 番目以降の行でも、その値は以前に行われた行の挿入による影響を受けません。(ただし、LAST_INSERT_ID() と LAST_INSERT_ID(expr) への参照を混在させると、効果は定義されません。)

以前のステートメントでエラーが返された場合は、LAST_INSERT_ID() の値が定義されません。トランザクションテーブルでは、エラーによってステートメントがロールバックされる場合、LAST_INSERT_ID() の値は未定義のままです。手動の ROLLBACK では、LAST_INSERT_ID() の値はトランザクション前の値にリストアされず、ROLLBACK 時点と同じままです。

MySQL 5.6.15 よりも前では、レプリケーションのフィルタ処理ルールが使用されている場合に、この関数が正常にレプリケートされませんでした。(Bug #17234370、Bug #69861)

ストアドルーチン (プロシージャーや関数) またはトリガーの本文内では、LAST_INSERT_ID() の値は、このような種類のオブジェクトの本文外で実行されたステートメントと同様に変更されます。あとに続くステートメントで参照される LAST_INSERT_ID() の値でのストアドルーチンまたはトリガーの効果は、ルーチンの種類によって異なります。

- ストアドプロシージャーで LAST_INSERT_ID() の値を変更するステートメントが実行される場合は、プロシージャー呼び出しが続くステートメントで変更された値が参照されます。
- 値を変更するストアドファンクションおよびトリガーでは、値は関数やトリガーが終了したときにリストアされ、あとに続くステートメントでは変更された値が参照されません。

生成された ID は、接続ごとにサーバー内に保持されます。つまり、関数によって指定されたクライアントに返された値は、そのクライアントによって AUTO_INCREMENT カラムに影響を与える最近のステートメント用に最初に生成された AUTO_INCREMENT 値です。この値は、ほかのクライアントが独自の AUTO_INCREMENT 値を生成した場合でも影響を受ける可能性はありません。この動作によって、各クライアントはほかのクライアントのアクティビティーを気にすることなく、ロックやトランザクションを実行しないで独自の ID を取得できます。

行の AUTO_INCREMENT カラムを非「マジック」値 (つまり、NULL でも 0 でもない値) に設定する場合は、LAST_INSERT_ID() の値が変更されません。

> **重要**
>
> 単一の INSERT ステートメントを使用して複数の行を挿入する場合、LAST_INSERT_ID() は、最初に挿入された行のみに対して生成された値を返します。この理由は、ほかの一部のサーバーに対して同じ INSERT ステートメントを簡単に再現できるようにするためです。

例:

```
mysql> USE test;
Database changed
mysql> CREATE TABLE t (
    ->   id INT AUTO_INCREMENT NOT NULL PRIMARY KEY,
    ->   name VARCHAR(10) NOT NULL
    -> );
Query OK, 0 rows affected (0.09 sec)

mysql> INSERT INTO t VALUES (NULL, 'Bob');
Query OK, 1 row affected (0.01 sec)

mysql> SELECT * FROM t;
+----+------+
| id | name |
+----+------+
|  1 | Bob  |
+----+------+
1 row in set (0.01 sec)

mysql> SELECT LAST_INSERT_ID();
+------------------+
| LAST_INSERT_ID() |
+------------------+
|                1 |
```

```
+------------------+
1 row in set (0.00 sec)

mysql> INSERT INTO t VALUES
    -> (NULL, 'Mary'), (NULL, 'Jane'), (NULL, 'Lisa');
Query OK, 3 rows affected (0.00 sec)
Records: 3  Duplicates: 0  Warnings: 0

mysql> SELECT * FROM t;
+----+------+
| id | name |
+----+------+
|  1 | Bob  |
|  2 | Mary |
|  3 | Jane |
|  4 | Lisa |
+----+------+
4 rows in set (0.01 sec)

mysql> SELECT LAST_INSERT_ID();
+------------------+
| LAST_INSERT_ID() |
+------------------+
|                2 |
+------------------+
1 row in set (0.00 sec)
```

2 番目の INSERT ステートメントで 3 つの新しい行が t に挿入されましたが、これらの行の 1 番目に生成された ID は 2 であり、あとに続く SELECT ステートメントでも、この値が LAST_INSERT_ID() によって返されます。

INSERT IGNORE を使用し、その行が無視された場合、LAST_INSERT_ID() は現在の値から未変更のままです (接続で正常な INSERT が実行されていない場合は、0 が返されます)。トランザクショナル以外のテーブルでは、AUTO_INCREMENT カウンタが増分されません。InnoDB テーブルでは、innodb_autoinc_lock_mode が 1 または 2 に設定されている場合は、次の例で示すように AUTO_INCREMENT が増分されます。

```
mysql> USE test;
Database changed

mysql> SELECT @@innodb_autoinc_lock_mode;
+----------------------------+
| @@innodb_autoinc_lock_mode |
+----------------------------+
|                          1 |
+----------------------------+
1 row in set (0.00 sec)

mysql> CREATE TABLE `t` (
`id` INT(11) NOT NULL AUTO_INCREMENT,
`val` INT(11) DEFAULT NULL,
PRIMARY KEY (`id`),
UNIQUE KEY `i1` (`val`)
) ENGINE=InnoDB DEFAULT CHARSET=latin1;
Query OK, 0 rows affected (0.02 sec)

-- Insert two rows

mysql> INSERT INTO t (val) VALUES (1),(2);
Query OK, 2 rows affected (0.00 sec)
Records: 2  Duplicates: 0  Warnings: 0

-- With auto_increment_offset=1, the inserted rows
-- result in an AUTO_INCREMENT value of 3

mysql> SHOW CREATE TABLE t\G
*************************** 1. row ***************************
       Table: t
Create Table: CREATE TABLE `t` (
  `id` int(11) NOT NULL AUTO_INCREMENT,
  `val` int(11) DEFAULT NULL,
  PRIMARY KEY (`id`),
  UNIQUE KEY `i1` (`val`)
) ENGINE=MyISAM AUTO_INCREMENT=3 DEFAULT CHARSET=latin1
1 row in set (0.00 sec)

-- LAST_INSERT_ID() returns the first automatically generated
-- value that is successfully inserted for the AUTO_INCREMENT column
```

```
mysql> SELECT LAST_INSERT_ID();
+------------------+
| LAST_INSERT_ID() |
+------------------+
|                1 |
+------------------+
1 row in set (0.00 sec)

-- The attempted insertion of duplicate rows fail but errors are ignored

mysql> INSERT IGNORE INTO t (val) VALUES (1),(2);
Query OK, 0 rows affected (0.00 sec)
Records: 2  Duplicates: 2  Warnings: 0

-- With innodb_autoinc_lock_mode=1, the AUTO_INCREMENT counter
-- is incremented for the ignored rows

mysql> SHOW CREATE TABLE t\G
*************************** 1. row ***************************
       Table: t
Create Table: CREATE TABLE `t` (
  `id` int(11) NOT NULL AUTO_INCREMENT,
  `val` int(11) DEFAULT NULL,
  PRIMARY KEY (`id`),
  UNIQUE KEY `i1` (`val`)
) ENGINE=MyISAM AUTO_INCREMENT=5 DEFAULT CHARSET=latin1
1 row in set (0.00 sec)

-- The LAST_INSERT_ID is unchanged becuase the previous insert was unsuccessful

mysql> SELECT LAST_INSERT_ID();
+------------------+
| LAST_INSERT_ID() |
+------------------+
|                1 |
+------------------+
1 row in set (0.00 sec)
```

詳細は、セクション14.6.5「InnoDB での AUTO_INCREMENT 処理」を参照してください。

expr が LAST_INSERT_ID() への引数として指定されている場合は、その引数の値が関数によって返され、LAST_INSERT_ID() によって次に返される値として記憶されます。これを使用すると、シーケンスのシミュレーションを行うことができます。

1. シーケンスカウンタを保持するテーブルを作成し、それを初期化します。

   ```
   mysql> CREATE TABLE sequence (id INT NOT NULL);
   mysql> INSERT INTO sequence VALUES (0);
   ```

2. そのテーブルを使用して、次のようにシーケンス番号を生成します。

   ```
   mysql> UPDATE sequence SET id=LAST_INSERT_ID(id+1);
   mysql> SELECT LAST_INSERT_ID();
   ```

   UPDATE ステートメントは、シーケンスカウンタを増分し、LAST_INSERT_ID() への次の呼び出しで更新された値が返されるようにします。SELECT ステートメントは、その値を取得します。mysql_insert_id() C API 関数を使用して、値を取得することもできます。セクション23.7.7.37「mysql_insert_id()」を参照してください。

LAST_INSERT_ID() を呼び出さずに、シーケンスを生成できます。このように関数を使用する有用性は、ID 値が最後に自動的に生成された値として保持されることです。独自のシーケンス値を生成するほかのクライアントと互いに影響しあうことなく、複数のクライアントが UPDATE ステートメントを発行し、SELECT ステートメント (または mysql_insert_id()) で独自のシーケンス値を取得できるため、マルチユーザーでも安全です。

mysql_insert_id() 関数は、INSERT および UPDATE ステートメントのあとにしか更新されないため、SELECT や SET などのその他の SQL ステートメントを実行したあとに、この C API 関数を使用しても、LAST_INSERT_ID(expr) の値を取得できません。

- ROW_COUNT()

  MySQL 5.6 では、ROW_COUNT() は次のように値を返します。

  - DDL ステートメント: 0。これは、CREATE TABLE や DROP TABLE などのステートメントに適用されます。
  - SELECT 以外の DML ステートメント: 影響を受ける行数です。これは、(以前と同様に) UPDATE、INSERT、DELETE などのステートメントに適用されますが、ALTER TABLE や LOAD DATA INFILE などのステートメントにも適用されるようになりました。
  - SELECT: ステートメントで結果セットが返される場合は -1、そうでない場合は「影響を受ける」行数。たとえば、SELECT * FROM t1 の場合、ROW_COUNT() は -1 を返します。SELECT * FROM t1 INTO OUTFILE 'file_name' の場合、ROW_COUNT() はファイルに書き込まれた行の数を返します。
  - SIGNAL ステートメント: 0。

  UPDATE ステートメントの場合、デフォルトで影響を受けた行の値は実際に変更された行の数です。mysqld への接続時に CLIENT_FOUND_ROWS フラグを mysql_real_connect() に指定した場合、影響を受けた行の値は「見つかった」、つまり WHERE 句に一致した行数です。

  REPLACE ステートメントの場合、影響を受けた行の値は、新しい行が古い行に置き換わった場合 2 です。この場合、重複が削除されたあとに行が挿入されたためです。

  INSERT ... ON DUPLICATE KEY UPDATE ステートメントの場合、行ごとの影響を受けた行の値は、その行が新しい行として挿入された場合は 1、既存の行が更新された場合は 2、既存の行がその現在の値に設定された場合は 0 です。CLIENT_FOUND_ROWS フラグを指定した場合、影響を受けた行の値は、既存の行がその現在の値に設定された場合は (0 ではなく) 1 になります。

  ROW_COUNT() は、mysql_affected_rows() C API 関数から取得される値と同様で、ステートメントの実行後に mysql クライアントに表示される行数です。

```
mysql> INSERT INTO t VALUES(1),(2),(3);
Query OK, 3 rows affected (0.00 sec)
Records: 3  Duplicates: 0  Warnings: 0

mysql> SELECT ROW_COUNT();
+-------------+
| ROW_COUNT() |
+-------------+
|           3 |
+-------------+
1 row in set (0.00 sec)

mysql> DELETE FROM t WHERE i IN(1,2);
Query OK, 2 rows affected (0.00 sec)

mysql> SELECT ROW_COUNT();
+-------------+
| ROW_COUNT() |
+-------------+
|           2 |
+-------------+
1 row in set (0.00 sec)
```

  > **重要**
  >
  > ステートメントベースのレプリケーションでは、確実に ROW_COUNT() をレプリケートすることはできません。行ベースのレプリケーションを使用すると、この関数は自動的にレプリケートされます。

- SCHEMA()

  この関数は DATABASE() のシノニムです。

- SESSION_USER()

  SESSION_USER() は USER() のシノニムです。

- SYSTEM_USER()

  SYSTEM_USER() は USER() のシノニムです。

- USER()

  現在の MySQL ユーザー名とホスト名を utf8 文字セット内の文字列として返します。

```
mysql> SELECT USER();
    -> 'davida@localhost'
```

  この値は、サーバーへの接続時に指定したユーザー名および接続元のクライアントホストを示します。CURRENT_USER() の値とは異なる可能性があります。

  次のように、ユーザー名の部分のみを抽出できます。

```
mysql> SELECT SUBSTRING_INDEX(USER(),'@',1);
    -> 'davida'
```

- VERSION()

  MySQL サーバーのバージョンを示す文字列を返します。この文字列では、utf8 文字セットが使用されます。値にはバージョン番号に加えて、サフィクスが付いている場合もあります。セクション5.1.4「サーバーシステム変数」内の version システム変数の説明を参照してください。

  この関数は、ステートメントベースのレプリケーションでは安全に使用できません。binlog_format が STATEMENT に設定されているときに、この関数を使用すると、警告の口グが記録されます。

```
mysql> SELECT VERSION();
    -> '5.6.23-standard'
```

12.15 空間分析関数

MySQL には、空間データに対してさまざまな操作を実行する関数が用意されています。これらの関数は、実行する操作の種類に従って、次のいくつかの主なカテゴリにグループ化できます。

- さまざまな形式 (WKT、WKB、内部) の幾何図形を作成する関数
- 幾何図形の形式を変換する関数
- 幾何図形の定性的または定量的なプロパティーにアクセスする関数
- 2 つの幾何図形間の関係を記述する関数
- 既存の幾何図形から新しい幾何図形を作成する関数

空間データを使用するための MySQL のサポートに関する一般的なバックグラウンドについては、セクション11.5「空間データの拡張」を参照してください。

12.15.1 空間関数のリファレンス

表 12.19 空間関数

名前	説明
Area()	Polygon または MultiPolygon 領域を返します
AsBinary(), AsWKB()	内部幾何形式から WKB に変換します
AsText(), AsWKT()	内部幾何形式から WKT に変換します
Buffer() (導入 5.6.1)	幾何図形から指定された距離内にある点の幾何図形を返します
Centroid()	重心を Point として返します
Contains()	ある幾何図形に別の幾何図形が含まれているかどうか
Crosses()	ある幾何図形が別の幾何図形と交差しているかどうか
Dimension()	幾何図形の次元
Disjoint()	ある幾何図形が別の幾何図形から切り離されているかどうか
EndPoint()	LineString の終点
Envelope()	幾何図形の MBR を返します
Equals()	ある幾何図形が別の幾何図形に等しいかどうか
ExteriorRing()	Polygon の外側のリングを返します

名前	説明
GeomCollFromText(), GeometryCollectionFromText()	WKT からジオメトリコレクションを返します
GeomCollFromWKB(), GeometryCollectionFromWKB()	WKB からジオメトリコレクションを返します
GeometryCollection()	幾何図形からジオメトリコレクションを構築します
GeometryN()	ジオメトリコレクションから N 番目の幾何図形を返します
GeometryType()	幾何型の名前を返します
GeomFromText(), GeometryFromText()	WKT から幾何図形を返します
GeomFromWKB()	WKB から幾何図形を返します
GLength()	LineString の長さを返します
InteriorRingN()	Polygon の N 番目の内側のリングを返します
Intersects()	ある幾何図形が別の幾何図形と交差しているかどうか
IsClosed()	幾何図形が閉じていて単純かどうか
IsEmpty()	プレースホルダ関数
IsSimple()	幾何図形が単純かどうか
LineFromText()	WKT から LineString を構築します
LineFromWKB(), LineStringFromWKB()	WKB から LineString を構築します
LineString()	Point 値から LineString を構築します
MBRContains()	ある幾何図形の MBR に、別の幾何図形の MBR が含まれているかどうか
MBRDisjoint()	2 つの幾何図形の MBR が切り離されているかどうか
MBREqual()	2 つの幾何図形の MBR が等しいかどうか
MBRIntersects()	2 つの幾何図形の MBR が交差しているかどうか
MBROverlaps()	2 つの幾何図形の MBR がオーバーラップしているかどうか
MBRTouches()	2 つの幾何図形の MBR が接しているかどうか
MBRWithin()	ある幾何図形の MBR が、別の幾何図形の MBR の内部にあるかどうか
MLineFromText(), MultiLineStringFromText()	WKT から MultiLineString を構築します
MLineFromWKB(), MultiLineStringFromWKB()	WKB から MultiLineString を構築します
MPointFromText(), MultiPointFromText()	WKT から MultiPoint を構築します
MPointFromWKB(), MultiPointFromWKB()	WKB から MultiPoint を構築します
MPolyFromText(), MultiPolygonFromText()	WKT から MultiPolygon を構築します
MPolyFromWKB(), MultiPolygonFromWKB()	WKB から MultiPolygon を構築します
MultiLineString()	LineString 値から MultiLineString を構築します
MultiPoint()	Point 値から MultiPoint を構築します
MultiPolygon()	Polygon 値から MultiPolygon を構築します
NumGeometries()	ジオメトリコレクション内の幾何図形数を返します
NumInteriorRings()	Polygon 内の内側のリング数を返します
NumPoints()	LineString 内の Point の数を返します
Overlaps()	ある幾何図形が別の幾何図形とオーバーラップしているかどうか
Point()	座標から Point を構築します
PointFromText()	WKT から Point を構築します
PointFromWKB()	WKB から Point を構築します
PointN()	LineString から N 番目の Point を返します
PolyFromText(), PolygonFromText()	WKT から Polygon を構築します

名前	説明
PolyFromWKB(), PolygonFromWKB()	WKB から Polygon を構築します
Polygon()	LineString 引数から Polygon を構築します
SRID()	幾何図形の空間参照システム ID を返します
ST_Area() (導入 5.6.1)	Polygon または MultiPolygon 領域を返します
ST_Centroid() (導入 5.6.1)	重心を Point として返します
ST_Contains() (導入 5.6.1)	ある幾何図形に別の幾何図形が含まれているかどうか
ST_Crosses() (導入 5.6.1)	ある幾何図形が別の幾何図形と交差しているかどうか
ST_Difference() (導入 5.6.1)	2 つの幾何図形の点集合の差集合を返します
ST_Disjoint() (導入 5.6.1)	ある幾何図形が別の幾何図形から切り離されているかどうか
ST_Distance() (導入 5.6.1)	ある幾何図形の別の幾何図形からの距離
ST_Envelope() (導入 5.6.1)	幾何図形の MBR を返します
ST_Equals() (導入 5.6.1)	ある幾何図形が別の幾何図形に等しいかどうか
ST_Intersection() (導入 5.6.1)	2 つの幾何図形の点集合の積集合を返します
ST_Intersects() (導入 5.6.1)	ある幾何図形が別の幾何図形と交差しているかどうか
ST_Overlaps() (導入 5.6.1)	ある幾何図形が別の幾何図形とオーバーラップしているかどうか
ST_SymDifference() (導入 5.6.1)	2 つの幾何図形の点集合の対称差を返します
ST_Touches() (導入 5.6.1)	ある幾何図形が別の幾何図形に接しているかどうか
ST_Union() (導入 5.6.1)	2 つの幾何図形の点集合の和集合を返します
ST_Within() (導入 5.6.1)	ある幾何図形が別の幾何図形の内部にあるかどうか
StartPoint()	LineString の始点
Touches()	ある幾何図形が別の幾何図形に接しているかどうか
Within()	ある幾何図形が別の幾何図形の内部にあるかどうか
X()	Point の X 座標を返します
Y()	Point の Y 座標を返します

12.15.2 空間関数による引数処理

空間値 (または幾何図形) は、セクション11.5.2.2「Geometry クラス」で説明されているプロパティーを持っています。次の説明では、空間関数の一般的な引数処理特性を一覧表示します。特定の関数または関数のグループには、追加の引数処理特性がある場合があります。これらについては、これらの関数の説明が行われるセクションで説明されています。

空間関数は、有効な幾何値に対してのみ定義されています。無効な幾何図形が空間関数に渡された場合の結果は定義されていません。

幾何図形の空間参照識別子 (SRID) は、その幾何図形が定義されている座標空間を識別します。MySQL の SRID 値は、幾何値に関連付けられた整数です。ただし、すべての計算が実際の SRID 値には関係なく、デカルト (平面) 座標を表す SRID 0 を想定して実行されます。将来は、指定された SRID 値が計算で使用される可能性があります。SRID 0 の動作を保証するには、SRID 0 を使用して幾何図形を作成します。SRID 0 は、SRID が指定されていない場合の新しい幾何図形のためのデフォルトです。

使用可能な SRID の最大値は $2^{32}-1$ です。より大きな値が指定されると、低位の 32 ビットだけが使用されます。

どの空間関数によって生成された幾何値も、その幾何図形の引数の SRID を継承します。

12.15.3 WKT 値から幾何値を作成する関数

これらの関数は、引数として Well-Known Text (WKT) 表現と、オプションで空間参照システム識別子 (SRID) を受け取ります。これらは、対応する幾何図形を返します。

GeomFromText() は、その最初の引数として任意の幾何型の WKT 値を受け入れます。その他の関数は、各幾何型の幾何値の構築のための型固有の構築関数を提供します。

WKT 形式については、WKT (Well-Known Text) 形式を参照してください。

- GeomCollFromText(wkt[,srid])、GeometryCollectionFromText(wkt[,srid])

  WKT 表現と SRID を使用して GeometryCollection 値を構築します。

- GeomFromText(wkt[,srid])、GeometryFromText(wkt[,srid])

  WKT 表現と SRID を使用して任意の型の幾何値を構築します。

- LineFromText(wkt[,srid])、LineStringFromText(wkt[,srid])

  WKT 表現と SRID を使用して LineString 値を構築します。

- MLineFromText(wkt[,srid])、MultiLineStringFromText(wkt[,srid])

  WKT 表現と SRID を使用して MultiLineString 値を構築します。

- MPointFromText(wkt[,srid])、MultiPointFromText(wkt[,srid])

  WKT 表現と SRID を使用して MultiPoint 値を構築します。

- MPolyFromText(wkt[,srid])、MultiPolygonFromText(wkt[,srid])

  WKT 表現と SRID を使用して MultiPolygon 値を構築します。

- PointFromText(wkt[,srid])

  WKT 表現と SRID を使用して Point 値を構築します。

- PolyFromText(wkt[,srid])、PolygonFromText(wkt[,srid])

  WKT 表現と SRID を使用して Polygon 値を構築します。

12.15.4 WKB 値から幾何値を作成する関数

これらの関数は、引数として Well-Known Binary (WKB) 表現を含む BLOB と、オプションで空間参照システム識別子 (SRID) を受け取ります。これらは、対応する幾何図形を返します。

これらの関数はまた、セクション12.15.5「幾何値を作成する MySQL 固有の関数」にある関数の戻り値との互換性のために幾何オブジェクトも受け入れます。そのため、これらの関数を使用すると、このセクションにある関数への最初の引数を提供できます。

GeomFromWKB() は、その最初の引数として任意の幾何型の WKB 値を受け入れます。その他の関数は、各幾何型の幾何値の構築のための型固有の構築関数を提供します。

WKB 形式については、WKB (Well-Known Binary) 形式を参照してください。

- GeomCollFromWKB(wkb[,srid])、GeometryCollectionFromWKB(wkb[,srid])

  WKB 表現と SRID を使用して GeometryCollection 値を構築します。

- GeomFromWKB(wkb[,srid])、GeometryFromWKB(wkb[,srid])

  WKB 表現と SRID を使用して任意の型の幾何値を構築します。

- LineFromWKB(wkb[,srid])、LineStringFromWKB(wkb[,srid])

  WKB 表現と SRID を使用して LineString 値を構築します。

- MLineFromWKB(wkb[,srid])、MultiLineStringFromWKB(wkb[,srid])

  WKB 表現と SRID を使用して MultiLineString 値を構築します。

- MPointFromWKB(wkb[,srid])、MultiPointFromWKB(wkb[,srid])

  WKB 表現と SRID を使用して MultiPoint 値を構築します。

- MPolyFromWKB(wkb[,srid])、MultiPolygonFromWKB(wkb[,srid])

  WKB 表現と SRID を使用して MultiPolygon 値を構築します。

- PointFromWKB(wkb[,srid])

WKB 表現と SRID を使用して Point 値を構築します。

- PolyFromWKB(wkb[,srid])、PolygonFromWKB(wkb[,srid])

  WKB 表現と SRID を使用して Polygon 値を構築します。

12.15.5 幾何値を作成する MySQL 固有の関数

MySQL には、幾何値を作成するために役立つ一連の非標準関数が用意されています。このセクションで説明されている関数は、OpenGIS 仕様への MySQL 拡張です。

これらの関数は、引数としての WKB 値または幾何オブジェクトから幾何オブジェクトを生成します。いずれかの引数が適切な WKB でも、適切なオブジェクト型の幾何表現でもない場合、戻り値は NULL になります。

たとえば、Point() からの幾何図形の戻り値を POINT カラムに直接挿入できます。

```
INSERT INTO t1 (pt_col) VALUES(Point(1,2));
```

- GeometryCollection(g1,g2,...)

  GeometryCollection を構築します。

  引数にサポートされていない幾何図形が含まれている場合、戻り値は NULL になります。

- LineString(pt1,pt2,...)

  複数の Point または WKB Point 引数から LineString 値を構築します。引数の数が 2 未満の場合、戻り値は NULL になります。

- MultiLineString(ls1,ls2,...)

  LineString または WKB LineString 引数を使用して MultiLineString 値を構築します。

- MultiPoint(pt1,pt2,...)

  Point または WKB Point 引数を使用して MultiPoint 値を構築します。

- MultiPolygon(poly1,poly2,...)

  一連の Polygon または WKB Polygon 引数から MultiPolygon 値を構築します。

- Point(x,y)

  座標を使用して Point を構築します。

- Polygon(ls1,ls2,...)

  複数の LineString または WKB LineString 引数から Polygon 値を構築します。いずれかの引数が LinearRing を表していない (つまり、閉じた単純な LineString でない) 場合、戻り値は NULL になります。

12.15.6 幾何形式変換関数

MySQL は、内部形式と WKT または WKB 形式の間で幾何値を変換するための次の関数をサポートします。

- AsBinary(g)、AsWKB(g)

  内部幾何形式の値を WKB 表現に変換し、そのバイナリの結果を返します。

```
SELECT AsBinary(g) FROM geom;
```

- AsText(g)、AsWKT(g)

  内部幾何形式の値を WKT 表現に変換し、その文字列の結果を返します。

```
mysql> SET @g = 'LineString(1 1,2 2,3 3)';
mysql> SELECT AsText(GeomFromText(@g));
+---------------------------+
| AsText(GeomFromText(@g)) |
+---------------------------+
| LINESTRING(1 1,2 2,3 3)  |
+---------------------------+
```

- GeomFromText(wkt[,srid])

  文字列値を WKT 表現から内部幾何形式に変換し、その結果を返します。また、PointFromText() や LineFromText() などの型固有の関数もいくつかサポートされます。セクション12.15.3「WKT 値から幾何値を作成する関数」を参照してください。

- GeomFromWKB(wkb[,srid])

  バイナリ値を WKB 表現から内部幾何形式に変換し、その結果を返します。また、PointFromWKB() や LineFromWKB() などの型固有の関数もいくつかサポートされます。セクション12.15.4「WKB 値から幾何値を作成する関数」を参照してください。

12.15.7 幾何プロパティー関数

このグループに属する各関数は、引数として幾何値を受け取り、その幾何図形の何らかの定量的または定性的なプロパティーを返します。一部の関数では、その引数の型が制限されます。このような関数は、その引数の幾何型が正しくない場合は NULL を返します。たとえば、Area() 多角形関数は、オブジェクト型が Polygon でも MultiPolygon でもない場合は NULL を返します。

12.15.7.1 一般的な幾何プロパティー関数

このセクションに示されている関数では引数が制限されず、任意の型の幾何値が受け入れられます。

- Dimension(g)

  幾何値 g の固有の次元を返します。結果は、-1、0、1、2 のいずれかです。これらの値の意味は、セクション11.5.2.2「Geometry クラス」で指定されています。

```
mysql> SELECT Dimension(GeomFromText('LineString(1 1,2 2)'));
+---------------------------------------------+
| Dimension(GeomFromText('LineString(1 1,2 2)')) |
+---------------------------------------------+
|                                           1 |
+---------------------------------------------+
```

- Envelope(g)

  ST_Envelope() と Envelope() はシノニムです。詳細は、ST_Envelope() の説明を参照してください。

- GeometryType(g)

  幾何インスタンス g がメンバーになっている幾何型の名前を示すバイナリ文字列を返します。この名前は、インスタンス化可能な Geometry サブクラスのいずれかに対応します。

```
mysql> SELECT GeometryType(GeomFromText('POINT(1 1)'));
+---------------------------------------+
| GeometryType(GeomFromText('POINT(1 1)')) |
+---------------------------------------+
| POINT                                 |
+---------------------------------------+
```

- IsEmpty(g)

  この関数は、すべての有効な幾何値に対しては 0、すべての無効な幾何値または NULL に対しては 1 を返すプレースホルダです。

  MySQL は、POINT EMPTY などの GIS の EMPTY 値をサポートしていません。

- IsSimple(g)

  幾何値 g に自己交差や自己接触などの異常な幾何点が含まれていない場合は 1 を返します。IsSimple() は、その引数が単純でない場合は 0 を、NULL である場合は NULL を返します。

  この章の最初の方で指定されているインスタンス化可能な各幾何クラスの説明には、そのクラスのインスタンスが単純でないとして分類される具体的な条件が含まれています。(セクション11.5.2.1「幾何クラスの階層」を参照してください。)

  MySQL 5.6.1 より前は、この関数は常に 0 を返します。

- SRID(g)

幾何値 g の空間参照システム ID を示す整数を返します。

MySQL では、SRID 値は幾何値に関連付けられた整数にすぎません。すべての計算はユークリッド (平面) 幾何学を前提にして実行されます。

```
mysql> SELECT SRID(GeomFromText('LineString(1 1,2 2)',101));
+---------------------------------------------+
| SRID(GeomFromText('LineString(1 1,2 2)',101)) |
+---------------------------------------------+
|                                         101 |
+---------------------------------------------+
```

- ST_Envelope(g)

幾何値 g の最小外接矩形 (MBR) を返します。結果は、その外接矩形の角の点によって定義された Polygon 値として返されます。

```
POLYGON((MINX MINY, MAXX MINY, MAXX MAXY, MINX MAXY, MINX MINY))
```

```
mysql> SELECT AsText(ST_Envelope(GeomFromText('LineString(1 1,2 2)')));
+---------------------------------------------------------+
| AsText(ST_Envelope(GeomFromText('LineString(1 1,2 2)'))) |
+---------------------------------------------------------+
| POLYGON((1 1,2 1,2 2,1 2,1 1))                          |
+---------------------------------------------------------+
```

ST_Envelope() と Envelope() はシノニムです。

ST_Envelope() は、MySQL 5.6.1 で追加されました。

12.15.7.2 点プロパティー関数

Point は、次の関数を使用して取得できる X 座標と Y 座標で構成されます。

- X(p)

Point オブジェクト p の X 座標値を倍精度数値として返します。

```
mysql> SELECT X(POINT(56.7, 53.34));
+----------------------+
| X(POINT(56.7, 53.34)) |
+----------------------+
|                 56.7 |
+----------------------+
```

- Y(p)

Point オブジェクト p の Y 座標値を倍精度数値として返します。

```
mysql> SELECT Y(POINT(56.7, 53.34));
+----------------------+
| Y(POINT(56.7, 53.34)) |
+----------------------+
|                53.34 |
+----------------------+
```

12.15.7.3 LineString プロパティー関数

LineString は、Point 値で構成されます。LineString の特定の点を抽出したり、そこに含まれている点の数をカウントしたり、その長さを取得したりできます。

- EndPoint(ls)

LineString 値 ls の終点である Point を返します。

```
mysql> SET @ls = 'LineString(1 1,2 2,3 3)';
mysql> SELECT AsText(EndPoint(GeomFromText(@ls)));
+-------------------------------------+
| AsText(EndPoint(GeomFromText(@ls))) |
+-------------------------------------+
| POINT(3 3)                          |
+-------------------------------------+
```

- GLength(ls)

LineString 値 ls の関連付けられた空間参照内の長さを示す倍精度数値を返します。

```
mysql> SET @ls = 'LineString(1 1,2 2,3 3)';
mysql> SELECT GLength(GeomFromText(@ls));
+---------------------------+
| GLength(GeomFromText(@ls)) |
+---------------------------+
|          2.8284271247462 |
+---------------------------+
```

GLength() は非標準名です。これは、OpenGIS の Length() 関数に対応します。(文字列値の長さを計算する既存の SQL 関数 Length() が存在します。)

- IsClosed(ls)

  LineString 値 ls が閉じており (つまり、その StartPoint() 値と EndPoint() 値が同じであり)、かつ単純である (同じ点を複数回通っていない) 場合は 1 を返します。ls が閉じていない場合は 0 を、NULL である場合は -1 を返します。

```
mysql> SET @ls1 = 'LineString(1 1,2 2,3 3,2 2)';
Query OK, 0 rows affected (0.00 sec)

mysql> SET @ls2 = 'LineString(1 1,2 2,3 3,1 1)';
Query OK, 0 rows affected (0.00 sec)

mysql> SELECT IsClosed(GeomFromText(@ls1));
+------------------------------+
| IsClosed(GeomFromText(@ls1)) |
+------------------------------+
|                            0 |
+------------------------------+
1 row in set (0.00 sec)

mysql> SELECT IsClosed(GeomFromText(@ls2));
+------------------------------+
| IsClosed(GeomFromText(@ls2)) |
+------------------------------+
|                            1 |
+------------------------------+
1 row in set (0.00 sec)
```

- NumPoints(ls)

  LineString 値 ls 内の Point オブジェクトの数を返します。

```
mysql> SET @ls = 'LineString(1 1,2 2,3 3)';
mysql> SELECT NumPoints(GeomFromText(@ls));
+------------------------------+
| NumPoints(GeomFromText(@ls)) |
+------------------------------+
|                            3 |
+------------------------------+
```

- PointN(ls,N)

  Linestring 値 ls 内の N 番目の Point を返します。点の番号は 1 から始まります。

```
mysql> SET @ls = 'LineString(1 1,2 2,3 3)';
mysql> SELECT AsText(PointN(GeomFromText(@ls),2));
+-------------------------------------+
| AsText(PointN(GeomFromText(@ls),2)) |
+-------------------------------------+
| POINT(2 2)                          |
+-------------------------------------+
```

- StartPoint(ls)

  LineString 値 ls の始点である Point を返します。

```
mysql> SET @ls = 'LineString(1 1,2 2,3 3)';
mysql> SELECT AsText(StartPoint(GeomFromText(@ls)));
+---------------------------------------+
| AsText(StartPoint(GeomFromText(@ls))) |
+---------------------------------------+
| POINT(1 1)                            |
+---------------------------------------+
```

12.15.7.4 MultiLineString プロパティー関数

これらの関数は、MultiLineString 値のプロパティーを返します。

- GLength(mls)

  MultiLineString 値 mls の長さを示す倍精度数値を返します。mls の長さは、その要素の長さの合計に等しくなります。

```
mysql> SET @mls = 'MultiLineString((1 1,2 2,3 3),(4 4,5 5))';
mysql> SELECT GLength(GeomFromText(@mls));
+---------------------------+
| GLength(GeomFromText(@mls)) |
+---------------------------+
|          4.2426406871193 |
+---------------------------+
```

  GLength() は非標準名です。これは、OpenGIS の Length() 関数に対応します。

- IsClosed(mls)

  MultiLineString 値 mls が閉じている (つまり、mls 内の各 LineString で StartPoint() 値と EndPoint() 値が同じである) 場合は 1 を返します。mls が閉じていない場合は 0 を、NULL である場合は -1 を返します。

```
mysql> SET @mls = 'MultiLineString((1 1,2 2,3 3),(4 4,5 5))';
mysql> SELECT IsClosed(GeomFromText(@mls));
+---------------------------+
| IsClosed(GeomFromText(@mls)) |
+---------------------------+
|                         0 |
+---------------------------+
```

12.15.7.5 Polygon プロパティー関数

これらの関数は、Polygon 値のプロパティーを返します。

- Area(poly)

  ST_Area() と Area() はシノニムです。詳細は、ST_Area() の説明を参照してください。

- ExteriorRing(poly)

  Polygon 値 poly の外側のリングを LineString として返します。

```
mysql> SET @poly =
    -> 'Polygon((0 0,0 3,3 3,3 0,0 0),(1 1,1 2,2 2,2 1,1 1))';
mysql> SELECT AsText(ExteriorRing(GeomFromText(@poly)));
+-------------------------------------------+
| AsText(ExteriorRing(GeomFromText(@poly))) |
+-------------------------------------------+
| LINESTRING(0 0,0 3,3 3,3 0,0 0)           |
+-------------------------------------------+
```

- InteriorRingN(poly,N)

  Polygon 値 poly の N 番目の内側のリングを LineString として返します。リングの番号は 1 から始まります。

```
mysql> SET @poly =
    -> 'Polygon((0 0,0 3,3 3,3 0,0 0),(1 1,1 2,2 2,2 1,1 1))';
mysql> SELECT AsText(InteriorRingN(GeomFromText(@poly),1));
+-----------------------------------------------+
| AsText(InteriorRingN(GeomFromText(@poly),1)) |
+-----------------------------------------------+
| LINESTRING(1 1,1 2,2 2,2 1,1 1)               |
+-----------------------------------------------+
```

- NumInteriorRings(poly)

  Polygon 値 poly 内の内側のリングの数を返します。

```
mysql> SET @poly =
    -> 'Polygon((0 0,0 3,3 3,3 0,0 0),(1 1,1 2,2 2,2 1,1 1))';
mysql> SELECT NumInteriorRings(GeomFromText(@poly));
+--------------------------------------+
| NumInteriorRings(GeomFromText(@poly)) |
```

```
+--------------------------------------+
|                                    1 |
+--------------------------------------+
```

- ST_Area(poly)

  空間参照システムで測定された引数の面積を示す倍精度数値を返します。次元 0 または 1 の引数の場合、結果は 0 です。

```
mysql> SET @poly = 'Polygon((0 0,0 3,3 0,0 0),(1 1,1 2,2 1,1 1))';
mysql> SELECT ST_Area(GeomFromText(@poly));
+------------------------------+
| ST_Area(GeomFromText(@poly)) |
+------------------------------+
|                            4 |
+------------------------------+

mysql> SET @mpoly =
    -> 'MultiPolygon(((0 0,0 3,3 3,3 0,0 0),(1 1,1 2,2 2,2 1,1 1)))';
mysql> SELECT ST_Area(GeomFromText(@mpoly));
+-------------------------------+
| ST_Area(GeomFromText(@mpoly)) |
+-------------------------------+
|                             8 |
+-------------------------------+
```

  ST_Area() と Area() はシノニムです。

  ST_Area() は、MySQL 5.6.1 で追加されました。

12.15.7.6 MultiPolygon プロパティー関数

これらの関数は、MultiPolygon 値のプロパティーを返します。

- Area(mpoly)

  セクション12.15.7.5「Polygon プロパティー関数」にある Area() の説明を参照してください。

- Centroid(mpoly)

  ST_Centroid() と Centroid() はシノニムです。詳細は、ST_Centroid() の説明を参照してください。

- ST_Area(mpoly)

  セクション12.15.7.5「Polygon プロパティー関数」にある Area() の説明を参照してください。

- ST_Centroid(mpoly)

  MultiPolygon 値 mpoly の数学的な重心を Point として返します。結果が MultiPolygon 上にある保証はありません。

```
mysql> SET @poly =
    -> GeomFromText('POLYGON((0 0,10 0,10 10,0 10,0 0),(5 5,7 5,7 7,5 7,5 5))');
mysql> SELECT GeometryType(@poly),AsText(ST_Centroid(@poly));
+--------------------+-----------------------------------------+
| GeometryType(@poly) | AsText(ST_Centroid(@poly))              |
+--------------------+-----------------------------------------+
| POLYGON            | POINT(4.958333333333333 4.958333333333333) |
+--------------------+-----------------------------------------+
```

  ST_Centroid() と Centroid() はシノニムです。

  ST_Centroid() は、MySQL 5.6.1 で追加されました。

12.15.7.7 GeometryCollection プロパティー関数

これらの関数は、GeometryCollection 値のプロパティーを返します。

- GeometryN(gc,N)

  GeometryCollection 値 gc 内の N 番目の幾何図形を返します。幾何図形の番号は 1 から始まります。

```
mysql> SET @gc = 'GeometryCollection(Point(1 1),LineString(2 2, 3 3))';
mysql> SELECT AsText(GeometryN(GeomFromText(@gc),1));
```

```
+----------------------------------------+
| AsText(GeometryN(GeomFromText(@gc),1)) |
+----------------------------------------+
| POINT(1 1)                             |
+----------------------------------------+
```

- NumGeometries(gc)

  GeometryCollection 値 gc 内の幾何図形の数を返します。

```
mysql> SET @gc = 'GeometryCollection(Point(1 1),LineString(2 2, 3 3))';
mysql> SELECT NumGeometries(GeomFromText(@gc));
+----------------------------------+
| NumGeometries(GeomFromText(@gc)) |
+----------------------------------+
|                                2 |
+----------------------------------+
```

12.15.8 空間演算子関数

OpenGIS では、幾何図形を生成できる関数がいくつか提案されています。これらは、空間演算子を実装するように設計されています。

- Buffer(g,d)

  幾何値 g からの距離が d の距離以下であるすべての点を表す幾何図形を返します。

  Buffer() は、多角形、複数多角形、および多角形または複数多角形を含む幾何図形コレクションに対して負の距離をサポートしています。点、複数点、ライン文字列、複数ライン文字列、および多角形または複数多角形を含まない幾何図形コレクションに対して、負の距離を含む Buffer() は NULL を返します。

  Buffer() は、MySQL 5.6.1 で追加されました。

- ST_Difference(g1, g2)

  幾何値 g1 と g2 の点集合の差集合を表す幾何図形を返します。

```
mysql> SET @g1 = POINT(1,1), @g2 = POINT(2,2);
mysql> SELECT AsText(ST_Difference(@g1, @g2));
+-----------------------------------+
| AsText(ST_Difference(@g1, @g2)) |
+-----------------------------------+
| POINT(1 1)                        |
+-----------------------------------+
```

  ST_Difference() は、MySQL 5.6.1 で追加されました。

- ST_Intersection(g1, g2)

  幾何値 g1 と g2 の点集合の共通集合を表す幾何図形を返します。

```
mysql> SET @g1 = GeomFromText('LineString(1 1, 3 3)');
mysql> SET @g2 = GeomFromText('LineString(1 3, 3 1)');
mysql> SELECT AsText(ST_Intersection(@g1, @g2));
+-----------------------------------+
| AsText(ST_Intersection(@g1, @g2)) |
+-----------------------------------+
| POINT(2 2)                        |
+-----------------------------------+
```

  ST_Intersection() は、MySQL 5.6.1 で追加されました。

- ST_SymDifference(g1, g2)

  幾何値 g1 と g2 の点集合の対称差を表す幾何図形を返します。これは、次のように定義されます。

```
g1 symdifference g2 := (g1 union g2) difference (g1 intersection g2)
```

  または、関数呼び出しの表記では次のようになります。

```
ST_SymDifference(g1, g2) = ST_Difference(ST_Union(g1, g2), ST_Intersection(g1, g2))
```

```
mysql> SET @g1 = POINT(1,1), @g2 = POINT(2,2);
mysql> SELECT AsText(ST_SymDifference(@g1, @g2));
```

```
+-------------------------------------------+
| AsText(ST_SymDifference(@g1, @g2))        |
+-------------------------------------------+
| GEOMETRYCOLLECTION(POINT(1 1),POINT(2 2)) |
+-------------------------------------------+
```

ST_SymDifference() は、MySQL 5.6.1 で追加されました。

- ST_Union(g1, g2)

  幾何値 g1 と g2 の点集合の和集合を表す幾何図形を返します。

```
mysql> SET @g1 = GeomFromText('LineString(1 1, 3 3)');
mysql> SET @g2 = GeomFromText('LineString(1 3, 3 1)');
mysql> SELECT AsText(ST_Union(@g1, @g2));
+--------------------------------------+
| AsText(ST_Union(@g1, @g2))           |
+--------------------------------------+
| MULTILINESTRING((1 1,3 3),(3 1,1 3)) |
+--------------------------------------+
```

ST_Union() は、MySQL 5.6.1 で追加されました。

さらに、セクション12.15.7「幾何プロパティー関数」では、既存の幾何図形から新しい幾何図形を構築するいくつかの関数について説明しています。これらの関数の説明については、そのセクションを参照してください。

- Envelope(g)
- StartPoint(ls)
- EndPoint(ls)
- PointN(ls,N)
- ExteriorRing(poly)
- InteriorRingN(poly,N)
- GeometryN(gc,N)

12.15.9 幾何オブジェクト間の空間関係をテストする関数

このセクションで説明されている関数は、引数として 2 つの幾何図形を受け取り、それらの間の定性的または定量的な関係を返します。

- OpenGIS 仕様では、2 つの幾何値間の関係をテストする一連の関数が定義されています。MySQL は最初、これらの関数を実装することにより、オブジェクトの最小外接矩形 (MBR) を使用し、対応する MBR ベースの関数と同じ結果を返していました。たとえば、Contains() と MBRContains() は同じ結果を返します。(例外: Crosses() は存在しますが、MBRCrosses() は存在しません。)

  MySQL 5.6.1 の時点では、正確なオブジェクト形状を使用する対応するバージョンを使用できます。これらのバージョンの名前には ST_ プリフィクスが付けられています。たとえば、Contains() が最小外接矩形を使用するのに対して、ST_Contains() はオブジェクト形状を使用します。

  MySQL 5.6.1 の時点では、すでに正確であった既存の空間関数のための ST_ エイリアスも存在します。たとえば、ST_IsEmpty() は IsEmpty() のエイリアスです。さらに、ST_Distance() も使用できます。

- 2 つの幾何値間の関係をテストするには、MySQL 固有の一連の MBR ベースの関数を使用できます。

12.15.9.1 オブジェクト形状を使用する空間関係関数

OpenGIS 仕様では、次の関数が定義されています。これらは、正確なオブジェクト形状を使用して、2 つの幾何値 g1 と g2 の間の関係をテストします。戻り値 1 と 0 は、それぞれ true と false を示します。

- ST_Contains(g1,g2)

  g1 が g2 を完全に含んでいるかどうかを示す 1 または 0 を返します。これは、ST_Within() とは逆の関係をテストします。

- ST_Crosses(g1,g2)

g1 が g2 と空間的に交差している場合は 1 を返します。g1 が Polygon または MultiPolygon である場合、あるいは g2 が Point または MultiPoint である場合は NULL を返します。それ以外の場合は、0 を返します。

空間的に交差しているという用語は、2 つの指定された幾何図形間の空間関係が次の性質を持っていることを示します。

- 2 つの幾何図形が交差している
- それらの交差によって、2 つの指定された幾何図形の最大の次元より 1 つ小さい次元を持つ幾何図形が生成される
- それらの交差が 2 つの指定された幾何図形のどちらとも等しくない

- ST_Disjoint(g1,g2)

  g1 が g2 と空間的に切り離されている (交差していない) かどうかを示す 1 または 0 を返します。

- ST_Distance(g1,g2)

  g1 と g2 の間の距離を返します。

```
mysql> SET @g1 = POINT(1,1), @g2 = POINT(2,2);
mysql> SELECT ST_Distance(@g1, @g2);
+----------------------+
| ST_Distance(@g1, @g2) |
+----------------------+
|    1.4142135623730951 |
+----------------------+
```

  ST_Distance() は、MySQL 5.6.1 で追加されました。

- ST_Equals(g1,g2)

  g1 が g2 と空間的に等しいかどうかを示す 1 または 0 を返します。

- ST_Intersects(g1,g2)

  g1 が g2 と空間的に交差しているかどうかを示す 1 または 0 を返します。

- ST_Overlaps(g1,g2)

  g1 が g2 と空間的にオーバーラップしているかどうかを示す 1 または 0 を返します。空間的にオーバーラップしているという用語が使用されるのは、2 つの幾何図形が交差しており、かつそれらの交差によって同じ次元の幾何図形が生成されるが、指定された幾何図形のどちらとも等しくない場合です。

- ST_Touches(g1,g2)

  g1 が g2 に空間的に接しているかどうかを示す 1 または 0 を返します。2 つの幾何図形が空間的に接しているのは、これらの幾何図形の内部は交差していないが、一方の幾何図形の境界が他方の幾何図形の境界または内部と交差している場合です。

- ST_Within(g1,g2)

  g1 が空間的に g2 の内部にあるかどうかを示す 1 または 0 を返します。これは、ST_Contains() とは逆の関係をテストします。

12.15.9.2 最小外接矩形 (MBR) を使用する空間関係関数

OpenGIS 仕様では、次の関数が定義されています。これらは、2 つの幾何値 g1 と g2 の間の関係をテストします。MySQL 実装では最小外接矩形を使用するため、これらの関数は、対応する MBR ベースの関数と同じ結果を返します。戻り値 1 と 0 は、それぞれ true と false を示します。

- Contains(g1,g2)

  g1 が g2 を完全に含んでいるかどうかを示す 1 または 0 を返します。これは、Within() とは逆の関係をテストします。

- Crosses(g1,g2)

  g1 が g2 と空間的に交差している場合は 1 を返します。g1 が Polygon または MultiPolygon である場合、あるいは g2 が Point または MultiPoint である場合は NULL を返します。それ以外の場合は、0 を返します。

空間的に交差しているという用語は、2 つの指定された幾何図形間の空間関係が次の性質を持っていることを示します。

- 2 つの幾何図形が交差している
- それらの交差によって、2 つの指定された幾何図形の最大の次元より 1 つ小さい次元を持つ幾何図形が生成される
- それらの交差が 2 つの指定された幾何図形のどちらとも等しくない

- Disjoint(g1,g2)

  g1 が g2 と空間的に切り離されている (交差していない) かどうかを示す 1 または 0 を返します。

- Equals(g1,g2)

  g1 が g2 と空間的に等しいかどうかを示す 1 または 0 を返します。

- Intersects(g1,g2)

  g1 が g2 と空間的に交差しているかどうかを示す 1 または 0 を返します。

- Overlaps(g1,g2)

  g1 が g2 と空間的にオーバーラップしているかどうかを示す 1 または 0 を返します。空間的にオーバーラップしているという用語が使用されるのは、2 つの幾何図形が交差しており、かつそれらの交差によって同じ次元の幾何図形が生成されるが、指定された幾何図形のどちらとも等しくない場合です。

- Touches(g1,g2)

  g1 が g2 に空間的に接しているかどうかを示す 1 または 0 を返します。2 つの幾何図形が空間的に接しているのは、これらの幾何図形の内部は交差していないが、一方の幾何図形の境界が他方の幾何図形の境界または内部と交差している場合です。

- Within(g1,g2)

  g1 が空間的に g2 の内部にあるかどうかを示す 1 または 0 を返します。これは、Contains() とは逆の関係をテストします。

12.15.9.3 最小外接矩形 (MBR) を使用する MySQL 固有の空間関係関数

MySQL には、2 つの幾何図形 g1 と g2 の最小外接矩形間の関係をテストする関数がいくつか用意されています。戻り値 1 と 0 は、それぞれ true と false を示します。

- MBRContains(g1,g2)

  g1 の最小外接矩形が g2 の最小外接矩形を含んでいるかどうかを示す 1 または 0 を返します。これは、MBRWithin() とは逆の関係をテストします。

```
mysql> SET @g1 = GeomFromText('Polygon((0 0,0 3,3 3,3 0,0 0))');
mysql> SET @g2 = GeomFromText('Point(1 1)');
mysql> SELECT MBRContains(@g1,@g2), MBRWithin(@g2,@g1);
+---------------------+--------------------+
| MBRContains(@g1,@g2) | MBRWithin(@g2,@g1) |
+---------------------+--------------------+
|                    1 |                  1 |
+---------------------+--------------------+
```

- MBRDisjoint(g1,g2)

  2 つの幾何図形 g1 と g2 の最小外接矩形が切り離されている (交差していない) かどうかを示す 1 または 0 を返します。

- MBREqual(g1,g2)

  2 つの幾何図形 g1 と g2 の最小外接矩形が同じであるかどうかを示す 1 または 0 を返します。

- MBRIntersects(g1,g2)

  2 つの幾何図形 g1 と g2 の最小外接矩形が交差しているかどうかを示す 1 または 0 を返します。

- MBROverlaps(g1,g2)

  2 つの幾何図形 g1 と g2 の最小外接矩形がオーバーラップしているかどうかを示す 1 または 0 を返します。空間的にオーバーラップしているという用語が使用されるのは、2 つの幾何図形が交差しており、かつそれらの交差によって同じ次元の幾何図形が生成されるが、指定された幾何図形のどちらとも等しくない場合です。

- MBRTouches(g1,g2)

  2 つの幾何図形 g1 と g2 の最小外接矩形が接しているかどうかを示す 1 または 0 を返します。2 つの幾何図形が空間的に接しているのは、これらの幾何図形の内部は交差していないが、一方の幾何図形の境界が他方の幾何図形の境界または内部と交差している場合です。

- MBRWithin(g1,g2)

  g1 の最小外接矩形が g2 の最小外接矩形の内部にあるかどうかを示す 1 または 0 を返します。これは、MBRContains() とは逆の関係をテストします。

```
mysql> SET @g1 = GeomFromText('Polygon((0 0,0 3,3 3,3 0,0 0))');
mysql> SET @g2 = GeomFromText('Polygon((0 0,0 5,5 5,5 0,0 0))');
mysql> SELECT MBRWithin(@g1,@g2), MBRWithin(@g2,@g1);
+-------------------+-------------------+
| MBRWithin(@g1,@g2) | MBRWithin(@g2,@g1) |
+-------------------+-------------------+
|                 1 |                 0 |
+-------------------+-------------------+
```

12.16 グローバルトランザクション ID とともに使用される関数

このセクションで説明する関数は、GTID ベースのレプリケーション (MySQL 5.6.5 以降で使用可能) で使用されます。これらのすべての関数には、引数として GTID セットの文字列表現が指定されるため、これらの関数とともに GTID セットを使用する際は、常に引用符で囲む必要があることを忘れないことが重要です。

2 つの GTID セットの結合は、単にカンマを挿入して結合された文字列として表現されたものです。言い換えると、ここで作成した関数と同様に、非常に単純な関数を定義すれば、GTID セットの結合を取得できます。

```
CREATE FUNCTION GTID_UNION(g1 TEXT, g2 TEXT)
  RETURNS TEXT DETERMINISTIC
  RETURN CONCAT(g1,',',g2);
```

GTID の詳細およびこれらの GTID 関数を実際に使用する方法については、セクション17.1.3「グローバルトランザクション識別子を使用したレプリケーション」を参照してください。

表 12.20 GTID 関数

名前	説明
GTID_SUBSET() (導入 5.6.5)	サブセット内のすべての GTID がセット内にもある場合は、true を返します。そうでない場合は、false を返します。
GTID_SUBTRACT() (導入 5.6.5)	セット内の GTID のうち、サブセット内にないものをすべてを返します。
SQL_THREAD_WAIT_AFTER_GTIDS() (導入 5.6.5, 非推奨 5.6.9)	廃止: WAIT_UNTIL_SQL_THREAD_AFTER_GTIDS() に置き換わりました
WAIT_UNTIL_SQL_THREAD_AFTER_GTIDS (導入 5.6.9)	指定された GTID がスレーブで実行されるまで待ちます。

- GTID_SUBSET(subset,set)

  2 セットのグローバルトランザクション ID subset と set が指定された場合、subset 内のすべての GTID が set にも存在すれば true (1) を返します。それ以外の場合は、false (0) を返します。

  この関数で使用される GTID セットは、次の例で示すように文字列で表現されます。

```
mysql> SELECT GTID_SUBSET('3E11FA47-71CA-11E1-9E33-C80AA9429562:23',
    ->     '3E11FA47-71CA-11E1-9E33-C80AA9429562:21-57')\G
*************************** 1. row ***************************
GTID_SUBSET('3E11FA47-71CA-11E1-9E33-C80AA9429562:23',
    '3E11FA47-71CA-11E1-9E33-C80AA9429562:21-57'): 1
1 row in set (0.00 sec)
```

```
mysql> SELECT GTID_SUBSET('3E11FA47-71CA-11E1-9E33-C80AA9429562:23-25',
    ->     '3E11FA47-71CA-11E1-9E33-C80AA9429562:21-57')\G
*************************** 1. row ***************************
GTID_SUBSET('3E11FA47-71CA-11E1-9E33-C80AA9429562:23-25',
    '3E11FA47-71CA-11E1-9E33-C80AA9429562:21-57'): 1
1 row in set (0.00 sec)

mysql> SELECT GTID_SUBSET('3E11FA47-71CA-11E1-9E33-C80AA9429562:20-25',
    ->     '3E11FA47-71CA-11E1-9E33-C80AA9429562:21-57')\G
*************************** 1. row ***************************
GTID_SUBSET('3E11FA47-71CA-11E1-9E33-C80AA9429562:20-25',
    '3E11FA47-71CA-11E1-9E33-C80AA9429562:21-57'): 0
1 row in set (0.00 sec)
```

この関数は、MySQL 5.6.5 で追加されました。

- GTID_SUBTRACT(set,subset)

  2 セットのグローバルトランザクション ID subset と set が指定された場合、set 内の GTID で、subset に存在しないものだけを返します。

  この関数で使用される GTID セットはすべて文字列で表現されるため、次の例で示すように引用符で囲む必要があります。

```
mysql> SELECT GTID_SUBTRACT('3E11FA47-71CA-11E1-9E33-C80AA9429562:21-57',
    ->     '3E11FA47-71CA-11E1-9E33-C80AA9429562:21')\G
*************************** 1. row ***************************
GTID_SUBTRACT('3E11FA47-71CA-11E1-9E33-C80AA9429562:21-57',
    '3E11FA47-71CA-11E1-9E33-C80AA9429562:21'): 3e11fa47-71ca-11e1-9e33-c80aa9429562:22-57
1 row in set (0.00 sec)

mysql> SELECT GTID_SUBTRACT('3E11FA47-71CA-11E1-9E33-C80AA9429562:21-57',
    ->     '3E11FA47-71CA-11E1-9E33-C80AA9429562:20-25')\G
*************************** 1. row ***************************
GTID_SUBTRACT('3E11FA47-71CA-11E1-9E33-C80AA9429562:21-57',
    '3E11FA47-71CA-11E1-9E33-C80AA9429562:20-25'): 3e11fa47-71ca-11e1-9e33-c80aa9429562:26-57
1 row in set (0.00 sec)

mysql> SELECT GTID_SUBTRACT('3E11FA47-71CA-11E1-9E33-C80AA9429562:21-57',
    ->     '3E11FA47-71CA-11E1-9E33-C80AA9429562:23-24')\G
*************************** 1. row ***************************
GTID_SUBTRACT('3E11FA47-71CA-11E1-9E33-C80AA9429562:21-57',
    '3E11FA47-71CA-11E1-9E33-C80AA9429562:23-24'): 3e11fa47-71ca-11e1-9e33-c80aa9429562:21-22:25-57
1 row in set (0.01 sec)
```

この関数は、MySQL 5.6.5 で追加されました。

- SQL_THREAD_WAIT_AFTER_GTIDS(gtid_set[, timeout])

  SQL_THREAD_WAIT_AFTER_GTIDS() は MySQL 5.6.5 で追加され、MySQL 5.6.9 で WAIT_UNTIL_SQL_THREAD_AFTER_GTIDS() に置き換えられました。(Bug #14775984)

  詳細は、セクション17.1.3「グローバルトランザクション識別子を使用したレプリケーション」を参照してください。

- WAIT_UNTIL_SQL_THREAD_AFTER_GTIDS(gtid_set[, timeout])

  スレーブ SQL スレッドで gtid_set 内にグローバルトランザクション ID が含まれるすべてのトランザクションが実行されるか (「GTID セット」の定義については、セクション17.1.3.1「GTID の概念」を参照してください)、timeout 秒が経過するかのどちらかが先に発生するまで待機します。timeout はオプションで、デフォルトのタイムアウトは 0 です。この場合、マスターは単に、GTID セット内のすべてのトランザクションが実行されるまで待機するだけです。

  MySQL 5.6.9 よりも前では、WAIT_UNTIL_SQL_THREAD_AFTER_GTIDS() は SQL_THREAD_WAIT_AFTER_GTIDS() という名前でした。(Bug #14775984)

  詳細は、セクション17.1.3「グローバルトランザクション識別子を使用したレプリケーション」を参照してください。

  この関数で使用される GTID セットは、次の例で示すように文字列で表現されるため、引用符で囲む必要があります。

```
mysql> SELECT WAIT_UNTIL_SQL_THREAD_AFTER_GTIDS('3E11FA47-71CA-11E1-9E33-C80AA9429562:1-5');
    -> 5
```

戻り値は、実行されたトランザクションイベントの数です。MySQL 5.6.8 よりも前では、この関数は、タイムアウトが設定されていない場合は予想不可能な動作となり、GTID ベースのレプリケーションがアクティブでない場合でも呼び出されていました。MySQL 5.6.8 以降では、gtid_mode が OFF である場合は、常に NULL を返します。(Bug #14640065)

12.17 MySQL Enterprise Encryption の関数

MySQL 5.6.21 の時点で MySQL Enterprise Edition には、SQL レベルで OpenSSL 機能を表示する OpenSSL ライブラリに基づいた一連の暗号化機能が含まれています。これらの関数を使用することによって、エンタープライズアプリケーションが次の操作を実行できるようになります。

- 公開鍵非対称暗号方式を使用した、追加のデータ保護の実装
- 公開鍵、秘密鍵、およびデジタル署名の作成
- 非対称暗号化および非対称復号化の実行
- デジタル署名およびデータの検証や妥当性検査に対する暗号化ハッシュの使用

Enterprise Encryption では、RSA、DSA、および DH の暗号アルゴリズムがサポートされています。

Enterprise Encryption は、個々の関数を個別にインストールできるユーザー定義関数 (UDF) ライブラリとして提供されます。

12.17.1 Enterprise Encryption のインストール

Enterprise Encryption の関数は、プラグインディレクトリ (plugin_dir システム変数で名前が指定されたディレクトリ) にインストールされたユーザー定義関数 (UDF) ライブラリファイルに配置されています。UDF ライブラリのベース名は openssl_udf であり、サフィクスはプラットフォームに依存します。たとえば、ファイル名は Linux では openssl_udf.so、Windows では openssl_udf.dll です。

ライブラリファイルから関数をインストールするには、CREATE FUNCTION ステートメントを使用します。ライブラリからすべての関数をロードするには、次のステートメントセットを使用します (必要に応じて、ファイル名サフィクスを調整します)。

```
CREATE FUNCTION asymmetric_decrypt RETURNS STRING
  SONAME 'openssl_udf.so';
CREATE FUNCTION asymmetric_derive RETURNS STRING
  SONAME 'openssl_udf.so';
CREATE FUNCTION asymmetric_encrypt RETURNS STRING
  SONAME 'openssl_udf.so';
CREATE FUNCTION asymmetric_sign RETURNS STRING
  SONAME 'openssl_udf.so';
CREATE FUNCTION asymmetric_verify RETURNS INTEGER
  SONAME 'openssl_udf.so';
CREATE FUNCTION create_asymmetric_priv_key RETURNS STRING
  SONAME 'openssl_udf.so';
CREATE FUNCTION create_asymmetric_pub_key RETURNS STRING
  SONAME 'openssl_udf.so';
CREATE FUNCTION create_dh_parameters RETURNS STRING
  SONAME 'openssl_udf.so';
CREATE FUNCTION create_digest RETURNS STRING
  SONAME 'openssl_udf.so';
```

一度インストールすれば、何回サーバーを再起動しても UDF はインストールされたままです。UDF をアンロードするには、DROP FUNCTION ステートメントを使用します。たとえば、鍵生成の関数をアンロードするには、次のように実行します。

```
DROP FUNCTION create_asymmetric_priv_key;
DROP FUNCTION create_asymmetric_pub_key;
```

CREATE FUNCTION および DROP FUNCTION ステートメントでは、関数名を小文字で指定する必要があります。これは、大文字と小文字のどちらでも使用できる関数の呼び出し時での使用とは異なります。

mysql データベースの場合、CREATE FUNCTION および DROP FUNCTION ステートメントでは、それぞれ INSERT および DROP 権限が必要です。

12.17.2 Enterprise Encryption の使用法と例

アプリケーションで Enterprise Encryption を使用するには、実行する演算に適した関数を呼び出します。このセクションでは、一部の典型的なタスクを実行する方法を示します。

タスク: RSA 暗号化を使用して秘密鍵と公開鍵のペアを作成します。

```
-- Encryption algorithm; can be 'DSA' or 'DH' instead
SET @algo = 'RSA';
-- Minimum key length in bits; make larger for stronger keys
SET @key_len = 1024;

-- Create private key
SET @priv = CREATE_ASYMMETRIC_PRIV_KEY(@algo, @key_len);
-- Derive corresponding public key from private key, using same algorithm
SET @pub = CREATE_ASYMMETRIC_PUB_KEY(@algo, @priv);
```

鍵のペアを使用すると、データを暗号化および復号化したり、データを署名および検証したり、対称鍵を生成したりできるようになりました。

タスク: 秘密鍵を使用してデータを暗号化し、公開鍵を使用して復号化します。これには、鍵ペアのメンバーが RSA 鍵である必要があります。

```
SET @ciphertext = ASYMMETRIC_ENCRYPT(@algo, 'My secret text', @priv);
SET @plaintext = ASYMMETRIC_DECRYPT(@algo, @ciphertext, @pub);
```

反対に、公開鍵を使用して暗号化し、秘密鍵を使用して復号化できます。

```
SET @ciphertext = ASYMMETRIC_ENCRYPT(@algo, 'My secret text', @pub);
SET @plaintext = ASYMMETRIC_DECRYPT(@algo, @ciphertext, @priv);
```

いずれの場合でも、暗号化関数および復号化関数用に指定されたアルゴリズムは、鍵を生成したときに使用されたアルゴリズムと一致する必要があります。

タスク: 文字列からダイジェストを生成します。

```
-- Digest type; can be 'SHA256', 'SHA384', or 'SHA512' instead
SET @dig_type = 'SHA224';

-- Generate digest string
SET @dig = CREATE_DIGEST(@dig_type, 'My text to digest');
```

タスク: 鍵ペアを含むダイジェストを使用してデータを署名してから、その署名がダイジェストと一致することを確認します。

```
-- Encryption algorithm; could be 'DSA' instead; keys must
-- have been created using same algorithm
SET @algo = 'RSA';

-- Generate signature for digest and verify signature against digest
SET @sig = ASYMMETRIC_SIGN(@algo, @dig, @priv, @dig_type);
-- Verify signature against digest
SET @verf = ASYMMETRIC_VERIFY(@algo, @dig, @sig, @pub, @dig_type);
```

タスク: 対称鍵を作成します。これには、共有対称シークレットを使用して作成される DH 秘密鍵/公開鍵が入力として必要です。鍵の長さを CREATE_DH_PARAMETERS() に渡してシークレットを作成してから、そのシークレットを「鍵の長さ」として CREATE_ASYMMETRIC_PRIV_KEY() に渡します。

```
-- Generate DH shared symmetric secret
SET @dhp = CREATE_DH_PARAMETERS(1024);
-- Generate DH key pairs
SET @algo = 'DH';
SET @priv1 = CREATE_ASYMMETRIC_PRIV_KEY(@algo, @dhp);
SET @pub1 = CREATE_ASYMMETRIC_PUB_KEY(@algo, @priv1);
SET @priv2 = CREATE_ASYMMETRIC_PRIV_KEY(@algo, @dhp);
SET @pub2 = CREATE_ASYMMETRIC_PUB_KEY(@algo, @priv2);

-- Generate symmetric key using public key of first party,
-- private key of second party
SET @sym1 = ASYMMETRIC_DERIVE(@pub1, @priv2);

-- Or use public key of second party, private key of first party
SET @sym2 = ASYMMETRIC_DERIVE(@pub2, @priv1);
```

鍵文字列の値は、SET、SELECT、または INSERT を使用することで実行時に作成し、変数やテーブルに格納できます。

```
SET @priv1 = CREATE_ASYMMETRIC_PRIV_KEY('RSA', 1024);
SELECT CREATE_ASYMMETRIC_PRIV_KEY('RSA', 1024) INTO @priv2;
```

```
INSERT INTO t (key_col) VALUES(CREATE_ASYMMETRIC_PRIV_KEY('RSA', 1024));
```

ファイルに格納されている鍵文字列の値は、FILE 権限を持つユーザーが LOAD_FILE() 関数を使用することで読み取ることができます。

ダイジェストと署名の文字列は、同様に処理できます。

12.17.3 Enterprise Encryption 関数のリファレンス

表 12.21 MySQL Enterprise Encryption の関数

名前	説明
ASYMMETRIC_DECRYPT() (導入 5.6.21)	秘密鍵または公開鍵を使用して暗号文を復号化します
ASYMMETRIC_DERIVE() (導入 5.6.21)	非対称鍵から対称鍵を導出します
ASYMMETRIC_ENCRYPT() (導入 5.6.21)	秘密鍵または公開鍵を使用してプレーンテキストを暗号化します
ASYMMETRIC_SIGN() (導入 5.6.21)	ダイジェストから署名を生成します
ASYMMETRIC_VERIFY() (導入 5.6.21)	署名がダイジェストと一致することを確認します
CREATE_ASYMMETRIC_PRIV_KEY() (導入 5.6.21)	秘密鍵を作成します
CREATE_ASYMMETRIC_PUB_KEY() (導入 5.6.21)	公開鍵を作成します
CREATE_DH_PARAMETERS() (導入 5.6.21)	共有 DH シークレットを生成します
CREATE_DIGEST() (導入 5.6.21)	文字列からダイジェストを生成します

12.17.4 Enterprise Encryption 関数の説明

Enterprise Encryption の関数には、次のような一般的な特性があります。

- 引数の型が不正な場合や引数の数が間違っている場合は、各関数でエラーが返されます。
- 要求された演算を実行することを関数に許可するのに引数が適していない場合は、必要に応じて NULL または 0 が返されます。これは、指定されたアルゴリズムが関数でサポートされていない場合、鍵の長さが短すぎたり長すぎたりする場合、PEM 書式の鍵文字列として要求される文字列が有効な鍵でない場合などに発生します。
- ベースとなる SSL ライブラリでは、ランダム度の初期化が処理されます。

関数の一部には、暗号化アルゴリズムの引数が指定されます。次の表には、サポートされているアルゴリズムのサマリーを関数別に示します。

表 12.22 関数でサポートされているアルゴリズム

関数	サポートされているアルゴリズム
ASYMMETRIC_DECRYPT()	RSA
ASYMMETRIC_DERIVE()	DH
ASYMMETRIC_ENCRYPT()	RSA
ASYMMETRIC_SIGN()	RSA、DSA
ASYMMETRIC_VERIFY()	RSA、DSA
CREATE_ASYMMETRIC_PRIV_KEY()	RSA、DSA、DH
CREATE_ASYMMETRIC_PUB_KEY()	RSA、DSA、DH

> **注記**
>
> RSA、DSA、または DH のいずれかの暗号化アルゴリズムを使用すれば鍵を作成できますが、鍵引数が指定されるその他の関数では、特定のタイプの鍵のみ

が許可される可能性があります。たとえば、ASYMMETRIC_ENCRYPT() および ASYMMETRIC_DECRYPT() では RSA 鍵のみが許可されます。

次の説明では、Enterprise Encryption 関数を呼び出す手順を示します。追加の例と説明については、セクション 12.17.2「Enterprise Encryption の使用法と例」を参照してください。

- ASYMMETRIC_DECRYPT(algorithm, crypt_str, key_str)

  指定されたアルゴリズムおよび鍵文字列を使用して、暗号化された文字列を復号化し、結果として生成されるプレーンテキストをバイナリ文字列として返します。復号化に失敗した場合は、結果が NULL になります。

  key_str は、PEM 書式の有効な鍵文字列である必要があります。復号化に成功した場合は、公開鍵または秘密鍵の文字列が、暗号化された文字列を生成する際に ASYMMETRIC_ENCRYPT() で使用された公開鍵または秘密鍵の文字列に対応している必要があります。algorithm は、鍵を作成する際に使用された暗号化アルゴリズムを示します。

  サポートされている algorithm 値: 'RSA'

  使用例については、ASYMMETRIC_ENCRYPT() の説明を参照してください。

- ASYMMETRIC_DERIVE(pub_key_str, priv_key_str)

  あるパーティーの秘密鍵と別のパーティーの公開鍵を使用して対称鍵を導出し、結果として生成される鍵をバイナリ文字列として返します。鍵の抽出に失敗した場合は、結果が NULL になります。

  pub_key_str および priv_key_str は、PEM 書式の有効な鍵文字列である必要があります。これらは、DH アルゴリズムを使用して作成する必要があります。

  公開鍵と秘密鍵の 2 つのペアを持っていると仮定します。

```
SET @dhp = CREATE_DH_PARAMETERS(1024);
SET @priv1 = CREATE_ASYMMETRIC_PRIV_KEY('DH', @dhp);
SET @pub1 = CREATE_ASYMMETRIC_PUB_KEY('DH', @priv1);
SET @priv2 = CREATE_ASYMMETRIC_PRIV_KEY('DH', @dhp);
SET @pub2 = CREATE_ASYMMETRIC_PUB_KEY('DH', @priv2);
```

  さらに、1 つのペアから秘密鍵を使用し、もう 1 つのペアから公開鍵を使用して、対称鍵文字列を作成すると仮定します。その後、この対称鍵の ID 関係が次のように保持されます。

```
ASYMMETRIC_DERIVE(@pub1, @priv2) = ASYMMETRIC_DERIVE(@pub2, @priv1)
```

- ASYMMETRIC_ENCRYPT(algorithm, str, key_str)

  指定されたアルゴリズムおよび鍵文字列を使用して文字列を暗号化し、結果として生成される暗号化テキストをバイナリ文字列として返します。暗号化に失敗した場合は、結果が NULL になります。

  str の長さは、バイト単位で key_str の長さ − 11 よりも大きくすることができません。

  key_str は、PEM 書式の有効な鍵文字列にする必要があります。algorithm は、鍵を作成する際に使用された暗号化アルゴリズムを示します。

  サポートされている algorithm 値: 'RSA'

  文字列を暗号化するには、秘密鍵または公開鍵の文字列を ASYMMETRIC_ENCRYPT() に渡します。元の非暗号化文字列をリカバリするには、暗号化に使用された秘密鍵または公開鍵の文字列に対応する公開鍵または秘密鍵の文字列とともに、暗号化された文字列を ASYMMETRIC_DECRYPT() に渡します。

```
-- Generate private/public key pair
SET @priv = CREATE_ASYMMETRIC_PRIV_KEY('RSA', 1024);
SET @pub = CREATE_ASYMMETRIC_PUB_KEY('RSA', @priv);

-- Encrypt using private key, decrypt using public key
SET @ciphertext = ASYMMETRIC_ENCRYPT('RSA', 'The quick brown fox', @priv);
SET @plaintext = ASYMMETRIC_DECRYPT('RSA', @ciphertext, @pub);

-- Encrypt using public key, decrypt using private key
SET @ciphertext = ASYMMETRIC_ENCRYPT('RSA', 'The quick brown fox', @pub);
SET @plaintext = ASYMMETRIC_DECRYPT('RSA', @ciphertext, @priv);
```

  次のように仮定します。

```
SET @s = a string to be encrypted
```

```
SET @priv = a valid private RSA key string in PEM format
SET @pub = the corresponding public RSA key string in PEM format
```

その後、これらの ID 関係が次のように保持されます。

```
ASYMMETRIC_DECRYPT('RSA', ASYMMETRIC_ENCRYPT('RSA', @s, @priv), @pub) = @s
ASYMMETRIC_DECRYPT('RSA', ASYMMETRIC_ENCRYPT('RSA', @s, @pub), @priv) = @s
```

- ASYMMETRIC_SIGN(algorithm, digest_str, priv_key_str, digest_type)

  秘密鍵文字列を使用してダイジェスト文字列に署名し、その署名をバイナリ文字列として返します。署名に失敗した場合は、結果が NULL になります。

  digest_str はダイジェスト文字列です。これは、CREATE_DIGEST() を呼び出すことで生成できます。digest_type は、ダイジェスト文字列を生成する際に使用されたダイジェストアルゴリズムを示します。

  priv_key_str は、ダイジェスト文字列に著名する際に使用される秘密鍵文字列です。これは、PEM 書式の有効な鍵文字列にする必要があります。algorithm は、鍵を作成する際に使用された暗号化アルゴリズムを示します。

  サポートされている algorithm 値: 'RSA'、'DSA'

  サポートされている digest_type 値: 'SHA224'、'SHA256'、'SHA384'、'SHA512'

  使用例については、ASYMMETRIC_VERIFY() の説明を参照してください。

- ASYMMETRIC_VERIFY(algorithm, digest_str, sig_str, pub_key_str, digest_type)

  署名文字列がダイジェスト文字列と一致するかどうかを確認し、確認に成功したのか失敗したのかを示す 1 または 0 を返します。

  digest_str はダイジェスト文字列です。これは、CREATE_DIGEST() を呼び出すことで生成できます。digest_type は、ダイジェスト文字列を生成する際に使用されたダイジェストアルゴリズムを示します。

  sig_str は署名文字列です。これは、ASYMMETRIC_SIGN() を呼び出すことで生成できます。

  pub_key_str は、署名者の公開鍵文字列です。これは、署名文字列を生成する ASYMMETRIC_SIGN() に渡される秘密鍵に対応し、PEM 書式の有効な鍵文字列にする必要があります。algorithm は、鍵を作成する際に使用された暗号化アルゴリズムを示します。

  サポートされている algorithm 値: 'RSA'、'DSA'

  サポートされている digest_type 値: 'SHA224'、'SHA256'、'SHA384'、'SHA512'

```
-- Set the encryption algorithm and digest type
SET @algo = 'RSA';
SET @dig_type = 'SHA224';

-- Create private/public key pair
SET @priv = CREATE_ASYMMETRIC_PRIV_KEY(@algo, 1024);
SET @pub = CREATE_ASYMMETRIC_PUB_KEY(@algo, @priv);

-- Generate digest from string
SET @dig = CREATE_DIGEST(@dig_type, 'The quick brown fox');

-- Generate signature for digest and verify signature against digest
SET @sig = ASYMMETRIC_SIGN(@algo, @dig, @priv, @dig_type);
SET @verf = ASYMMETRIC_VERIFY(@algo, @dig, @sig, @pub, @dig_type);
```

- CREATE_ASYMMETRIC_PRIV_KEY(algorithm, {key_len|dh_secret})

  指定されたアルゴリズムおよび鍵の長さまたは DH シークレットを使用して秘密鍵を作成し、その鍵を PEM 書式のバイナリ文字列として返します。鍵の生成に失敗した場合は、結果が NULL になります。

  サポートされている algorithm 値: 'RSA'、'DSA'、'DH'

  サポートされている key_len 値: 最小の鍵の長さは 1024 ビットです。最大の鍵の長さはアルゴリズムによって異なり、RSA の場合は 16,384、DSA の場合は 10,000 です。これらの長さは、OpenSSL によって課された制約です。

  DH 鍵の場合は、キーの長さの代わりに、共有 DH シークレットを渡します。シークレットを作成するには、鍵の長さを CREATE_DH_PARAMETERS() に渡します。

この例では、2,048 ビットの DSA 秘密鍵を作成してから、その秘密鍵から公開鍵を導出します。

```
SET @priv = CREATE_ASYMMETRIC_PRIV_KEY('DSA', 2048);
SET @pub = CREATE_ASYMMETRIC_PUB_KEY('DSA', @priv);
```

DH 鍵の生成を示す例については、ASYMMETRIC_DERIVE() の説明を参照してください。

鍵の長さと暗号化アルゴリズムを選択する際の一般的ないくつかの考慮事項は、次のとおりです。

- 鍵のサイズとともに、公開鍵と秘密鍵の暗号化強度が増加しますが、鍵の生成時間も同様に増加します。
- DH 鍵の生成時間は、RSA 鍵または RSA 鍵よりも大幅に長くなります。
- 非対称暗号化関数は、対称関数よりも遅くなります。パフォーマンスが重要な要素であり、その関数が非常に頻繁に使用される場合は、対称暗号化を使用した方が適切です。たとえば、AES_ENCRYPT() および AES_DECRYPT() を使用することを検討してください。

- CREATE_ASYMMETRIC_PUB_KEY(algorithm, priv_key_str)

  指定されたアルゴリズムを使用して、指定された秘密鍵から公開鍵を導出し、その鍵を PEM 書式のバイナリ文字列として返します。鍵の抽出に失敗した場合は、結果が NULL になります。

  priv_key_str は、PEM 書式の有効な鍵文字列にする必要があります。algorithm は、鍵を作成する際に使用された暗号化アルゴリズムを示します。

  サポートされている algorithm 値: 'RSA'、'DSA'、'DH'

  使用例については、CREATE_ASYMMETRIC_PRIV_KEY() の説明を参照してください。

- CREATE_DH_PARAMETERS(key_len)

  DH 秘密鍵と公開鍵のペアを生成するための共有シークレットを作成し、CREATE_ASYMMETRIC_PRIV_KEY() に渡すことができるバイナリ文字列を返します。シークレットの生成に失敗した場合は、結果が NULL になります。

  サポートされている key_len 値: 最小および最大の鍵の長さは、1024 ビットおよび 10,000 ビットです。これらの長さは、OpenSSL によって課された制約です。

  対称鍵を生成するための戻り値を使用する方法を示す例については、ASYMMETRIC_DERIVE() の説明を参照してください。

  ```
  SET @dhp = CREATE_DH_PARAMETERS(1024);
  ```

- CREATE_DIGEST(digest_type, str)

  指定されたダイジェストタイプを使用して、指定された文字列からダイジェストを作成し、そのダイジェストをバイナリ文字列として返します。ダイジェストの生成に失敗した場合は、結果が NULL になります。

  サポートされている digest_type 値: 'SHA224'、'SHA256'、'SHA384'、'SHA512'

  ```
  SET @dig = CREATE_DIGEST('SHA512', The quick brown fox');
  ```

  結果として生成されるダイジェスト文字列は、ASYMMETRIC_SIGN() および ASYMMETRIC_VERIFY() との使用に適しています。

12.18 その他の関数

表 12.23 その他の関数

名前	説明
DEFAULT()	テーブルカラムのデフォルト値を返します
GET_LOCK()	名前付きロックを取得します
INET_ATON()	IP アドレスの数値を返します
INET_NTOA()	数値から IP アドレスを返します
INET6_ATON() (導入 5.6.3)	IPv6 アドレスの数値を返します
INET6_NTOA() (導入 5.6.3)	数値から IPv6 アドレスを返します

名前	説明
IS_FREE_LOCK()	名前付きロックが解放されているかどうかを確認します
IS_IPV4() (導入 5.6.3)	引数が IPv4 アドレスの場合、true を返します
IS_IPV4_COMPAT() (導入 5.6.3)	引数が IPv4 互換アドレスの場合、true を返します
IS_IPV4_MAPPED() (導入 5.6.3)	引数が IPv4 マップアドレスの場合、true を返します
IS_IPV6() (導入 5.6.3)	引数が IPv6 アドレスの場合、true を返します
IS_USED_LOCK()	名前付きロックが使用中かどうかを確認します。true の場合は接続識別子を返します。
MASTER_POS_WAIT()	スレーブが指定された位置まですべての更新を読み取って適用するまで、ブロックします
NAME_CONST()	指定された名前がカラムに付けられます
RELEASE_LOCK()	名前付きロックを解放します
SLEEP()	ある秒数間スリープ状態にします
UUID()	ユニバーサル固有識別子 (UUID) を返します
UUID_SHORT()	整数値のユニバーサル識別子を返します
VALUES()	INSERT で使用される値を定義します

- DEFAULT(col_name)

  テーブルカラムのデフォルト値を返します。カラムにデフォルト値がない場合は、エラーが発生します。

```
mysql> UPDATE t SET i = DEFAULT(i)+1 WHERE id < 100;
```

- FORMAT(X,D)

  数値 X を '#,###,###.##' のような書式に変換し、小数点第 D 位に丸めて、その結果を文字列として返します。詳細は、セクション12.5「文字列関数」を参照してください。

- GET_LOCK(str,timeout)

  timeout 秒のタイムアウトを使用して、文字列 str で指定された名前でロックの取得を試みます。負の timeout 値は、無限のタイムアウトを表します。

  ロックの取得に成功した場合は 1 を返し、試行がタイムアウトになった場合 (たとえば、ほかのクライアントがすでにその名前をロックしている場合) は 0 を返し、エラー (メモリー不足や mysqladmin kill によるスレッドの停止など) が発生した場合は NULL を返します。GET_LOCK() を使用してロックを取得した場合は、RELEASE_LOCK() を実行したとき、新しい GET_LOCK() を実行したとき、または接続が切断されたとき (正常または異常を問わず) に解除されます。GET_LOCK() を使用して取得されたロックは、トランザクションとやりとりを行いません。つまり、トランザクションをコミットしても、トランザクション中に取得されたロックは解除されません。

  GET_LOCK() を使用すると、アプリケーションロックを実装したり、レコードロックのシミュレーションを行なったりできます。名前はサーバー全体にわたってロックされます。1 つのセッション内で名前がロックされた場合は、GET_LOCK() によって、別のセッションによる同じ名前を持つロックのリクエストがブロックされます。これにより、指定されたロック名について合意したクライアントは、その名前を使用すると共同のアドバイザリロックを実行できます。ただし、共同するクライアントのセットに属さないクライアントも、不注意と故意のどちらででも、名前をロックできることに注意してください。したがって、共同するクライアントがその名前をロックできないようにしてください。この可能性を減らす方法の 1 つは、データベース固有またはアプリケーション固有のロック名を使用することです。たとえば、db_name.str または app_name.str 形式のロック名を使用します。

```
mysql> SELECT GET_LOCK('lock1',10);
    -> 1
mysql> SELECT IS_FREE_LOCK('lock2');
    -> 1
mysql> SELECT GET_LOCK('lock2',10);
    -> 1
mysql> SELECT RELEASE_LOCK('lock2');
    -> 1
mysql> SELECT RELEASE_LOCK('lock1');
```

```
    -> NULL
```

ロック 'lock1' は 2 番目の GET_LOCK() 呼び出しによって自動的に解除されるため、2 番目の RELEASE_LOCK() 呼び出しは NULL を返します。

複数のクライアントがロックを待機している場合は、ロックを取得する順序が定義されません。アプリケーションは、ロックリクエストを発行したときと同じ順序でクライアントがロックを取得すると仮定しません。

この関数は、ステートメントベースのレプリケーションでは安全に使用できません。binlog_format が STATEMENT に設定されているときに、この関数を使用すると、警告のログが記録されます。

- INET_ATON(expr)

  IPv4 ネットワークアドレスのドット区切り表現が文字列として指定された場合、アドレスの数値を表す整数をネットワークバイト順序 (ビッグエンディアン) で返します。引数が認識されない場合、INET_ATON() は NULL を返します。

```
mysql> SELECT INET_ATON('10.0.5.9');
    -> 167773449
```

  この例では、戻り値は $10\times256^3 + 0\times256^2 + 5\times256 + 9$ として計算されます。

  INET_ATON() は、短い形式の IP アドレス ('127.0.0.1' の表現として '127.1' など) の場合に、非 NULL を返す場合と返さない場合があります。このため、このようなアドレスには INET_ATON() を使用しないでください。

  > **注記**
  >
  > INET_ATON() で生成された値を格納するには、署名される INT ではなく、INT UNSIGNED カラムを使用します。署名付きのカラムを使用する場合は、第 1 オクテットが 127 よりも大きい IP アドレスに対応する値を正常に格納できません。セクション11.2.6「範囲外およびオーバーフローの処理」を参照してください。

- INET_NTOA(expr)

  数値の IPv4 ネットワークアドレスがネットワークバイト順序で指定された場合、アドレスのドット区切り文字列表現を接続文字セット内の非バイナリ文字列として返します。引数が認識されない場合、INET_NTOA() は NULL を返します。

```
mysql> SELECT INET_NTOA(167773449);
    -> '10.0.5.9'
```

- INET6_ATON(expr)

  IPv6 または IPv4 ネットワークアドレスが文字列として指定された場合、アドレスの数値を表すバイナリ文字列をネットワークバイト順序 (ビッグエンディアン) で返します。数値書式の IPv6 アドレスでは最大の整数型よりも大きいバイトが必要であるため、この関数で返される表現のデータ型は、VARBINARY (IPv6 アドレスの場合は VARBINARY(16)、IPv4 アドレスの場合は VARBINARY(4)) です。引数が有効なアドレスでない場合、INET6_ATON() は NULL を返します。

  次の例では、HEX() を使用して INET6_ATON() の結果を出力可能な形式で表示します。

```
mysql> SELECT HEX(INET6_ATON('fdfe::5a55:caff:fefa:9089'));
    -> 'FDFE0000000000005A55CAFFFEFA9089'
mysql> SELECT HEX(INET6_ATON('10.0.5.9'));
    -> '0A000509'
```

  INET6_ATON() は、有効な引数上の複数の制約を監視します。これらについては、次のリストに例とともに示します。

  - fe80::3%1 や fe80::3%eth0 のような末尾のゾーン ID は許可されません。
  - 2001:45f:3:ba::/64 や 192.168.1.0/24 のような末尾のネットワークマスクは許可されません。
  - IPv4 アドレスを表す値では、クラスレスアドレスのみがサポートされます。192.168.1 などのクラスフルアドレスは拒否されます。192.168.1.2:8080 のような末尾のポート番号は許可されません。192.0xa0.1.2 のようなアドレスコンポーネント内の 16 進数は許可されません。8 進数はサポートされていません。192.168.010.1 は 192.168.8.1 ではなく、192.168.10.1 として処理されます。このような IPv4 の制約

は、IPv4 アドレス部分を持つ IPv6 アドレス (IPv4 互換アドレスや IPv4 マップアドレスなど) にも適用されます。

INT 値として数値形式で表現された IPv4 アドレス expr を、VARBINARY 値として数値形式で表現された IPv6 アドレスに変換するには、次の式を使用します。

```
INET6_ATON(INET_NTOA(expr))
```

例:

```
mysql> SELECT HEX(INET6_ATON(INET_NTOA(167773449)));
    -> '0A000509'
```

この関数は、MySQL 5.6.3 で追加されました。

- INET6_NTOA(expr)

  バイナリ文字列として数値形式で表現された IPv6 または IPv4 ネットワークアドレスが指定された場合、アドレスの文字列表現を接続文字セット内の非バイナリ文字列として返します。引数が有効なアドレスでない場合、INET6_NTOA() は NULL を返します。

  INET6_NTOA() には、次のようなプロパティーがあります。

  - 変換はオペレーティングシステムの機能を使用して実行されないため、出力文字列はプラットフォームに依存しません。
  - 戻り文字列の最大長は 39 (4 x 8 + 7) です。次のステートメントが指定された場合、

    ```
    CREATE TABLE t AS SELECT INET6_NTOA(expr) AS c1;
    ```

    結果として生成されるテーブルに次の定義が含まれます。

    ```
    CREATE TABLE t (c1 VARCHAR(39) CHARACTER SET utf8 DEFAULT NULL);
    ```

  - 戻り文字列では、IPv6 アドレスを表す小文字が使用されます。

  ```
  mysql> SELECT INET6_NTOA(INET6_ATON('fdfe::5a55:caff:fefa:9089'));
      -> 'fdfe::5a55:caff:fefa:9089'
  mysql> SELECT INET6_NTOA(INET6_ATON('10.0.5.9'));
      -> '10.0.5.9'

  mysql> SELECT INET6_NTOA(UNHEX('FDFE0000000000005A55CAFFFEFA9089'));
      -> 'fdfe::5a55:caff:fefa:9089'
  mysql> SELECT INET6_NTOA(UNHEX('0A000509'));
      -> '10.0.5.9'
  ```

  この関数は、MySQL 5.6.3 で追加されました。

- IS_FREE_LOCK(str)

  str という名前が付けられたロックが使用可能であるか (つまり、ロックされていないか) どうかをチェックします。ロックが使用可能である (だれもロックを使用していない) 場合は 1 を返し、使用中である場合は 0 を返し、エラー (不正な引数など) が発生した場合は NULL を返します。

  この関数は、ステートメントベースのレプリケーションでは安全に使用できません。binlog_format が STATEMENT に設定されているときに、この関数を使用すると、警告のログが記録されます。

- IS_IPV4(expr)

  引数が文字列として指定された有効な IPv4 アドレスの場合は 1 を返し、それ以外の場合は 0 を返します。

  ```
  mysql> SELECT IS_IPV4('10.0.5.9'), IS_IPV4('10.0.5.256');
      -> 1, 0
  ```

  特定の引数では、IS_IPV4() が 1 を返すと、INET_ATON() (および INET6_ATON()) は NULL 以外を返します。逆のステートメントは該当しません。一部のケースでは、IS_IPV4() が 0 を返すと、INET_ATON() は NULL 以外を返します。

  上記の備考で示したように、IS_IPV4() は、有効な IPv4 アドレスを構成するものに関して、INET_ATON() よりも厳密であるため、無効な値に対して強固なチェックを実行する必要のあるアプリケーションで役立つことがあります。または、INET6_ATON() を使用して、IPv4 アドレスを内部形式に変換し、(無効なアドレスを示す)

NULL の結果をチェックします。INET6_ATON() は、IPv4 アドレスのチェックという点では、IS_IPV4() と同等の強固さです。

この関数は、MySQL 5.6.3 で追加されました。

- IS_IPV4_COMPAT(expr)

この関数には、INET6_ATON() で返されるように、バイナリ文字列として数値形式で表現された IPv6 アドレスが指定されます。引数が有効な IPv4 互換 IPv6 アドレスの場合は 1 を返し、それ以外の場合は 0 を返します。IPv4 互換アドレスの形式は、::ipv4_address です。

```
mysql> SELECT IS_IPV4_COMPAT(INET6_ATON('::10.0.5.9'));
    -> 1
mysql> SELECT IS_IPV4_COMPAT(INET6_ATON('::ffff:10.0.5.9'));
    -> 0
```

IPv4 互換アドレスの IPv4 部分は、16 進表記法を使用して表現することもできます。たとえば、192.168.0.1 には、次のような生の 16 進値が含まれます。

```
mysql> SELECT HEX(INET6_ATON('192.168.0.1'));
    -> 'C0A80001'
```

IPv4 互換形式で表現された ::192.168.0.1 は、::c0a8:0001 または (先頭のゼロなしの) ::c0a8:1 と同等です。

```
mysql> SELECT
   ->   IS_IPV4_COMPAT(INET6_ATON('::192.168.0.1')),
   ->   IS_IPV4_COMPAT(INET6_ATON('::c0a8:0001')),
   ->   IS_IPV4_COMPAT(INET6_ATON('::c0a8:1'));
    -> 1, 1, 1
```

この関数は、MySQL 5.6.3 で追加されました。

- IS_IPV4_MAPPED(expr)

この関数には、INET6_ATON() で返されるように、バイナリ文字列として数値形式で表現された IPv6 アドレスが指定されます。引数が有効な IPv4 マップ IPv6 アドレスの場合は 1 を返し、それ以外の場合は 0 を返します。IPv4 マップアドレスの形式は、::ffff:ipv4_address です。

```
mysql> SELECT IS_IPV4_MAPPED(INET6_ATON('::10.0.5.9'));
    -> 0
mysql> SELECT IS_IPV4_MAPPED(INET6_ATON('::ffff:10.0.5.9'));
    -> 1
```

IS_IPV4_COMPAT() と同様に、IPv4 マップアドレスの IPv4 部分は、16 進表記法を使用して表現することもできます。

```
mysql> SELECT
   ->   IS_IPV4_MAPPED(INET6_ATON('::ffff:192.168.0.1')),
   ->   IS_IPV4_MAPPED(INET6_ATON('::ffff:c0a8:0001')),
   ->   IS_IPV4_MAPPED(INET6_ATON('::ffff:c0a8:1'));
    -> 1, 1, 1
```

この関数は、MySQL 5.6.3 で追加されました。

- IS_IPV6(expr)

引数が文字列として指定された有効な IPv6 アドレスの場合は 1 を返し、それ以外の場合は 0 を返します。この関数では、IPv4 アドレスが有効な IPv6 アドレスとみなされません。

```
mysql> SELECT IS_IPV6('10.0.5.9'), IS_IPV6('::1');
    -> 0, 1
```

特定の引数では、IS_IPV6() が 1 を返すと、INET6_ATON() は NULL 以外を返します。

この関数は、MySQL 5.6.3 で追加されました。

- IS_USED_LOCK(str)

  str という名前が付けられたロックが使用中であるか (つまり、ロックされているか) どうかをチェックします。その場合は、ロックを保持しているクライアントの接続識別子を返します。そうでない場合は、NULL を返します。

  この関数は、ステートメントベースのレプリケーションでは安全に使用できません。binlog_format が STATEMENT に設定されているときに、この関数を使用すると、警告のログが記録されます。

- MASTER_POS_WAIT(log_name,log_pos[,timeout])

  この関数は、マスター/スレーブの同期化を制御する際に役立ちます。スレーブがマスターログで指定された位置までのすべての更新を読み取り、適用するまでブロックします。戻り値は、指定された位置まで進むまでスレーブが待機する必要のあるログイベントの数です。この関数は、スレーブ SQL スレッドが開始されていない場合、スレーブのマスター情報が初期化されていない場合、引数が正しくない場合、またはエラーが発生した場合に NULL を返します。タイムアウトを超えた場合は、-1 を返します。MASTER_POS_WAIT() の待機中にスレーブ SQL スレッドが停止すると、関数は NULL を返します。スレーブが指定された位置を過ぎると、関数は即座に返します。

  timeout 値が指定されている場合、MASTER_POS_WAIT() は、timeout 秒が経過した時点で待機を停止します。timeout は 0 よりも大きい値にする必要があります。timeout がゼロまたは負の値である場合は、タイムアウトがないことを意味します。

  この関数は、ステートメントベースのレプリケーションでは安全に使用できません。binlog_format が STATEMENT に設定されているときに、この関数を使用すると、警告のログが記録されます。

- NAME_CONST(name,value)

  指定された値を返します。結果セットのカラムを生成する際に NAME_CONST() を使用すると、指定された名前がカラムに付けられます。引数には定数を指定してください。

```
mysql> SELECT NAME_CONST('myname', 14);
+--------+
| myname |
+--------+
|     14 |
+--------+
```

  この関数は、内部でのみ使用されます。セクション20.7「ストアドプログラムのバイナリロギング」で説明したように、ローカルプログラム変数への参照を含むストアドプログラムからステートメントを記述する際に、サーバーで使用されます。この関数は、mysqlbinlog からの出力に表示されることがあります。

  アプリケーションで次のような単純なエイリアスを使用しても、上記で示した例とまったく同じ結果を取得できます。

```
mysql> SELECT 14 AS myname;
+--------+
| myname |
+--------+
|     14 |
+--------+
1 row in set (0.00 sec)
```

  カラムのエイリアスについての詳細は、セクション13.2.9「SELECT 構文」を参照してください。

- RELEASE_LOCK(str)

  GET_LOCK() を使用して取得された文字列 str によって名前が付けられたロックを解除します。ロックが解除された場合は 1 を返し、このスレッドによってロックが確立されなかった場合 (その場合、ロックは解除されません) は 0 を返し、名前付きのロックが存在しない場合は NULL を返します。GET_LOCK() を呼び出しても取得されなかった場合や、事前に解除された場合は、ロックが存在しません。

  DO ステートメントは、RELEASE_LOCK() とともに使用すると便利です。セクション13.2.3「DO 構文」を参照してください。

  この関数は、ステートメントベースのレプリケーションでは安全に使用できません。binlog_format が STATEMENT に設定されているときに、この関数を使用すると、警告のログが記録されます。

- SLEEP(duration)

  duration 引数で指定された秒数間スリープ状態に (一時停止) してから、0 を返します。SLEEP() が中断された場合は、1 を返します。この期間には、小数部分が含まれている場合もあります。

  この関数は、ステートメントベースのレプリケーションでは安全に使用できません。binlog_format が STATEMENT に設定されているときに、この関数を使用すると、警告のログが記録されます。

- UUID()

  1997 年 10 月に The Open Group によって公開された 「DCE 1.1: Remote Procedure Call」 (付録 A) 「CAE (Common Applications Environment) Specifications」(ドキュメント番号 C706、http://www.opengroup.org/public/pubs/catalog/c706.htm)に従って生成されたユニバーサル固有識別子 (UUID) を返します。

  UUID は、空間と時間においてグローバルに一意の数字として設計されています。UUID() の 2 回の呼び出しが、相互に接続されていない 2 台の異なるコンピュータ上で実行された場合でも、これらの呼び出しでは 2 つの異なる値が生成されると想定されます。

  UUID は、aaaaaaaa-bbbb-cccc-dddd-eeeeeeeeeeee という書式で、5 つの 16 進数で構成される utf8 文字列で表現された 128 ビットの数値です。

  - 最初の 3 つの数字は、タイムスタンプから生成されます。
  - 4 番目の数字には、(たとえば、サマータイムが原因で) タイムスタンプ値の単調性が失われる場合に備えて、時間の一意性が保持されます。
  - 5 番目の数字は、空間の一意性を提供する IEEE 802 ノード番号です。(たとえば、ホストコンピュータに Ethernet カードが搭載されていないことや、オペレーティングシステム上のインタフェースのハードウェアアドレスを見つける方法がわからないことが原因で) 後者を使用できない場合は、ランダムな数字で置き換えられます。この場合、空間の一意性は保証できません。しかしそれでも、競合が発生する可能性は非常に低くなります。

    現在、インタフェースの MAC アドレスは、FreeBSD と Linux でのみ考慮されています。その他のオペレーティングシステムでは、ランダムに生成された 48 ビットの数字が MySQL で使用されます。

```
mysql> SELECT UUID();
    -> '6ccd780c-baba-1026-9564-0040f4311e29'
```

  > **警告**
  >
  > UUID() 値の目的は一意性を保つことですが、必ずしも推測不可能または予測不可能であるとはかぎりません。予測不可能性が必要である場合は、UUID 値を何か別の方法で生成してください。

  > **注記**
  >
  > UUID() は、ステートメントベースのレプリケーションでは正しく動作しません。

- UUID_SHORT()

  (UUID() 関数で返されるような文字列形式の 128 ビット識別子ではなく)、「短い」ユニバーサル識別子を 64 ビットの符号なし整数として返します。

  次の条件を満たす場合は、UUID_SHORT() の値が一意であることが保証されます。

  - 一連のマスター/スレーブサーバーの中で現在のホストの server_id が一意である
  - server_id の範囲が 0 から 255 である
  - mysqld の再起動間にサーバーのシステム時間を戻さない
  - mysqld の再起動間に、UUID_SHORT() を 1 秒あたり平均 1600 万を上回る回数で呼び出さない

  UUID_SHORT() の戻り値は、次のように構成されます。

```
  (server_id & 255) << 56
+ (server_startup_time_in_seconds << 24)
+ incremented_variable++;
```

```
mysql> SELECT UUID_SHORT();
```

```
    -> 92395783831158784
```

> **注記**
>
> UUID_SHORT() は、ステートメントベースのレプリケーションでは正しく動作しません。

- VALUES(col_name)

  INSERT ... ON DUPLICATE KEY UPDATE ステートメントでは、UPDATE 句の VALUES(col_name) 関数を使用すると、ステートメントの INSERT 部分からカラム値を参照できます。つまり、UPDATE 句の VALUES(col_name) は、挿入される col_name 値を参照するため、重複キーの競合が発生しなくなります。この関数は、複数の行を挿入する際に特に役立ちます。VALUES() 関数は、INSERT ステートメントの ON DUPLICATE KEY UPDATE 句でのみ有効であり、それ以外の場合は NULL を返します。セクション 13.2.5.3「INSERT ... ON DUPLICATE KEY UPDATE 構文」を参照してください。

```
mysql> INSERT INTO table (a,b,c) VALUES (1,2,3),(4,5,6)
    -> ON DUPLICATE KEY UPDATE c=VALUES(a)+VALUES(b);
```

12.19 GROUP BY 句で使用される関数と修飾子

12.19.1 GROUP BY (集約) 関数

表 12.24 集約 (GROUP BY) 関数

名前	説明
AVG()	引数の平均値を返します
BIT_AND()	ビット単位の And を返します
BIT_OR()	ビット単位の OR を返します
BIT_XOR()	ビット単位の XOR を返します
COUNT()	返された行数のカウントを返します
COUNT(DISTINCT)	異なる値のカウントを返します
GROUP_CONCAT()	連結された文字列を返します
MAX()	最大値を返します
MIN()	最小値を返します
STD()	母標準偏差を返します
STDDEV()	母標準偏差を返します
STDDEV_POP()	母標準偏差を返します
STDDEV_SAMP()	標本標準偏差を返します
SUM()	集計を返します
VAR_POP()	母標準分散を返します
VAR_SAMP()	標本分散を返します
VARIANCE()	母標準分散を返します

このセクションでは、値のセットを演算するグループ (集約) 関数について説明します。特に指定されていなければ、グループ関数では NULL 値が無視されます。

GROUP BY 句を含まないステートメントでグループ関数を使用する場合は、すべての行をグループ化することと同等になります。詳細は、セクション12.19.3「MySQL での GROUP BY の処理」を参照してください。

数値の引数の場合、分散および標準偏差関数が DOUBLE 値を返します。SUM() および AVG() 関数は、正確な値の引数 (整数または DECIMAL) の場合は DECIMAL 値を返し、近似値の引数 (FLOAT または DOUBLE) の場合は DOUBLE 値を返します。

SUM() および AVG() 集計関数は時間値を扱いません。(これらは値を数字に変換するので、最初の数字以外の文字のあとのすべての情報が失われます。)この問題を回避するには、数値単位に変換し、集計操作を実行してから、時間値に戻します。例:

```
SELECT SEC_TO_TIME(SUM(TIME_TO_SEC(time_col))) FROM tbl_name;
SELECT FROM_DAYS(SUM(TO_DAYS(date_col))) FROM tbl_name;
```

数値引数を取る SUM() や AVG() などの関数は、必要に応じて引数を数値にキャストします。SET や ENUM 値の場合、キャスト演算でベースとなる数値が使用されます。

- AVG([DISTINCT] expr)

  expr の平均値を返します。DISTINCT オプションを使用すると、個別の expr 値の平均を返すことができます。

  一致する行がなかった場合、AVG() は NULL を返します。

```
mysql> SELECT student_name, AVG(test_score)
    ->       FROM student
    ->       GROUP BY student_name;
```

- BIT_AND(expr)

  expr 内のすべてのビットのビット単位の AND を返します。計算は 64 ビット (BIGINT) の精度で実行されます。

  一致する行がなかった場合、この関数は 18446744073709551615 を返します。(これは、すべてのビットが 1 に設定された符号なし BIGINT 値の値です。)

- BIT_OR(expr)

  expr 内のすべてのビットのビット単位の OR を返します。計算は 64 ビット (BIGINT) の精度で実行されます。

  一致する行がなかった場合、この関数は 0 を返します。

- BIT_XOR(expr)

  expr 内のすべてのビットのビット単位の XOR を返します。計算は 64 ビット (BIGINT) の精度で実行されます。

  一致する行がなかった場合、この関数は 0 を返します。

- COUNT(expr)

  SELECT ステートメントで取得された行に含まれる expr の非 NULL 値の数を返します。結果は BIGINT 値になります。

  一致する行がなかった場合、COUNT() は 0 を返します。

```
mysql> SELECT student.student_name,COUNT(*)
    ->       FROM student,course
    ->       WHERE student.student_id=course.student_id
    ->       GROUP BY student_name;
```

  COUNT(*) は、NULL 値が含まれるかどうかに関係なく、取得された行の数を返すという点で少し異なります。

  COUNT(*) は、SELECT が 1 つのテーブルから取得し、その他のカラムは取得されず、WHERE 句がない場合に、非常に迅速に返すように最適化されています。例:

```
mysql> SELECT COUNT(*) FROM student;
```

  この最適化は、MyISAM テーブルにのみ適用されます。その理由は、正確な行数は、このストレージエンジンで格納されることで、非常にすばやくアクセスできるためです。InnoDB などのトランザクショナルストレージエンジンで正確な行数を格納すると、複数のトランザクションが発生し、それぞれが数に影響を与える可能性があるため、問題が発生する可能性が高くなります。

- COUNT(DISTINCT expr,[expr...])

  さまざまな非 NULL expr 値を含む行の数を返します。

  一致する行がなかった場合、COUNT(DISTINCT) は 0 を返します。

```
mysql> SELECT COUNT(DISTINCT results) FROM student;
```

  MySQL では、式のリストを指定することで、NULL が含まれない個別の式の組み合わせ数を取得できます。標準 SQL では、COUNT(DISTINCT ...) 内部で、すべての式の連結を行う必要があります。

- GROUP_CONCAT(expr)

  この関数は、グループから連結された非 NULL 値を含む文字列の結果を返します。非 NULL 値がない場合は、NULL を返します。完全な構文は次のとおりです。

```
GROUP_CONCAT([DISTINCT] expr [,expr ...]
          [ORDER BY {unsigned_integer | col_name | expr}
              [ASC | DESC] [,col_name ...]]
          [SEPARATOR str_val])
```

```
mysql> SELECT student_name,
    ->     GROUP_CONCAT(test_score)
    ->     FROM student
    ->     GROUP BY student_name;
```

  または:

```
mysql> SELECT student_name,
    ->     GROUP_CONCAT(DISTINCT test_score
    ->             ORDER BY test_score DESC SEPARATOR ' ')
    ->     FROM student
    ->     GROUP BY student_name;
```

  MySQL では、式の組み合わせを連結した値を取得できます。重複する値を除去するには、DISTINCT 句を使用します。結果の値をソートするには、ORDER BY 句を使用します。逆順でソートするには、ORDER BY 句のソートするカラムの名前に DESC (降順) キーワードを追加します。デフォルトは昇順です。これは、ASC キーワードを使用することで明示的に指定できます。グループ内の値間のデフォルトの区切り文字は、カンマ (「,」) です。区切り文字を明示的に指定するには、SEPARATOR に続けて、グループ値の間に挿入される文字列リテラル値を指定します。区切り文字を完全に除去するには、SEPARATOR '' を指定します。

  結果は、group_concat_max_len システム変数で指定された最大長まで切り捨てられます。その変数のデフォルト値は 1024 です。さらに高い値にも設定できますが、戻り値の有効な最大長は、max_allowed_packet の値によって制約されます。実行時に group_concat_max_len の値を変更するための構文は、次のとおりです。ここで、val は符号なし整数です。

```
SET [GLOBAL | SESSION] group_concat_max_len = val;
```

  戻り値は、引数が非バイナリとバイナリのどちらの文字列であるのかに応じて、非バイナリ文字列またはバイナリ文字列になります。結果の型は、group_concat_max_len が 512 以下の場合 (この場合、結果の型は VARCHAR または VARBINARY です) を除いて、TEXT または BLOB です。

  CONCAT() および CONCAT_WS(): セクション12.5「文字列関数」も参照してください。

- MAX([DISTINCT] expr)

  expr の最大値を返します。MAX() には、文字列の引数を指定できます。このような場合は、最大の文字列値が返されます。セクション8.3.1「MySQL のインデックスの使用の仕組み」を参照してください。DISTINCT キーワードを使用すると、個別の expr 値の最大を検索できます。ただし、DISTINCT を省略した場合と同じ結果が生成されます。

  一致する行がなかった場合、MAX() は NULL を返します。

```
mysql> SELECT student_name, MIN(test_score), MAX(test_score)
    ->       FROM student
    ->       GROUP BY student_name;
```

  現在、MySQL の MAX() では、ENUM と SET カラムが、セット内の文字列の相対位置ではなく、文字列値について比較されます。これは、ORDER BY による比較方法とは異なります。これは、今後の MySQL リリースで反映される予定です。

- MIN([DISTINCT] expr)

  expr の最小値を返します。MIN() には、文字列の引数を指定できます。このような場合は、最小の文字列値が返されます。セクション8.3.1「MySQL のインデックスの使用の仕組み」を参照してください。DISTINCT キーワードを使用すると、個別の expr 値の最小を検索できます。ただし、DISTINCT を省略した場合と同じ結果が生成されます。

  一致する行がなかった場合、MIN() は NULL を返します。

```
mysql> SELECT student_name, MIN(test_score), MAX(test_score)
    ->       FROM student
    ->       GROUP BY student_name;
```

現在、MySQL の MIN() では、ENUM と SET カラムが、セット内の文字列の相対位置ではなく、文字列値について比較されます。これは、ORDER BY による比較方法とは異なります。これは、今後の MySQL リリースで反映される予定です。

- STD(expr)

  expr の母標準偏差を返します。これは、標準 SQL の拡張です。代わりに、標準 SQL 関数 STDDEV_POP() を使用できます。

  一致する行がなかった場合、この関数は NULL を返します。

- STDDEV(expr)

  expr の母標準偏差を返します。この関数は、Oracle との互換性を確保するために提供されています。代わりに、標準 SQL 関数 STDDEV_POP() を使用できます。

  一致する行がなかった場合、この関数は NULL を返します。

- STDDEV_POP(expr)

  expr の母標準偏差 (VAR_POP() の平方根) を返します。STD() または STDDEV() を使用することもできます。これらは同等ですが、標準 SQL ではありません。

  一致する行がなかった場合、STDDEV_POP() は NULL を返します。

- STDDEV_SAMP(expr)

  expr の標本標準偏差 (VAR_SAMP() の平方根) を返します。

  一致する行がなかった場合、STDDEV_SAMP() は NULL を返します。

- SUM([DISTINCT] expr)

  expr の集計を返します。戻り値のセットに行が含まれていない場合、SUM() は NULL を返します。DISTINCT キーワードを使用すると、個別の expr 値のみを集計できます。

  一致する行がなかった場合、SUM() は NULL を返します。

- VAR_POP(expr)

  expr の母標準分散を返します。行は標本ではなく、母集団全体とみなされるため、行の数が分母とみなされます。また、VARIANCE() を使用することもできます。これは同等ですが、標準 SQL ではありません。

  一致する行がなかった場合、VAR_POP() は NULL を返します。

- VAR_SAMP(expr)

  expr の標本分散を返します。つまり、分母は行の数から 1 を引いたものです。

  一致する行がなかった場合、VAR_SAMP() は NULL を返します。

- VARIANCE(expr)

  expr の母標準分散を返します。これは、標準 SQL の拡張です。代わりに、標準 SQL 関数 VAR_POP() を使用できます。

  一致する行がなかった場合、VARIANCE() は NULL を返します。

12.19.2 GROUP BY 修飾子

GROUP BY 句では、追加の行がサマリー出力に追加される WITH ROLLUP 修飾子が許可されます。これらの行は、高レベル (または超集約) のサマリー演算を表します。したがって、ROLLUP で単一のクエリーを使用すれば、複数レベルの分析で質問に回答できます。たとえば、これを使用すれば、OLAP (Online Analytical Processing) 演算をサポートできます。

sales という名前が付けられたテーブルに、売り上げの収益性を記録するための year、country、product、および profit カラムが含まれていると仮定します。

```
CREATE TABLE sales
```

```
(
   year    INT NOT NULL,
   country VARCHAR(20) NOT NULL,
   product VARCHAR(32) NOT NULL,
   profit  INT
);
```

次のような単純な GROUP BY を使用すれば、テーブルの内容を年ごとにまとめることができます。

```
mysql> SELECT year, SUM(profit) FROM sales GROUP BY year;
+------+-------------+
| year | SUM(profit) |
+------+-------------+
| 2000 |        4525 |
| 2001 |        3010 |
+------+-------------+
```

この出力には、年ごとの合計収益が表示されますが、すべての年にわたって集計された合計収益を確認する場合は、各値を自分で合計するか、追加のクエリーを実行する必要があります。

または、単一のクエリーで両方のレベルの分析を提供する ROLLUP も使用できます。GROUP BY 句に WITH ROLLUP 修飾子を追加すると、クエリーによってすべての年にわたる総合計を示す別の行が生成されます。

```
mysql> SELECT year, SUM(profit) FROM sales GROUP BY year WITH ROLLUP;
+------+-------------+
| year | SUM(profit) |
+------+-------------+
| 2000 |        4525 |
| 2001 |        3010 |
| NULL |        7535 |
+------+-------------+
```

総合計の超集約行は、year カラムの NULL 値で識別されます。

複数の GROUP BY カラムがある場合は、ROLLUP の効果がより複雑になります。この場合、最後のグループ化カラム以外で「ブレイク」 (値の変更) が発生するたびに、クエリーによって追加の超集約サマリー行が生成されます。

たとえば、ROLLUP を使用しない場合は、year、country、および product に基づいた sales テーブルのサマリーが次のように表示されます。

```
mysql> SELECT year, country, product, SUM(profit)
    -> FROM sales
    -> GROUP BY year, country, product;
+------+---------+------------+-------------+
| year | country | product    | SUM(profit) |
+------+---------+------------+-------------+
| 2000 | Finland | Computer   |        1500 |
| 2000 | Finland | Phone      |         100 |
| 2000 | India   | Calculator |         150 |
| 2000 | India   | Computer   |        1200 |
| 2000 | USA     | Calculator |          75 |
| 2000 | USA     | Computer   |        1500 |
| 2001 | Finland | Phone      |          10 |
| 2001 | USA     | Calculator |          50 |
| 2001 | USA     | Computer   |        2700 |
| 2001 | USA     | TV         |         250 |
+------+---------+------------+-------------+
```

この出力には、year/country/product の分析レベルでのみサマリー値が示されます。ROLLUP が追加されると、クエリーによって複数の追加行が生成されます。

```
mysql> SELECT year, country, product, SUM(profit)
    -> FROM sales
    -> GROUP BY year, country, product WITH ROLLUP;
+------+---------+------------+-------------+
| year | country | product    | SUM(profit) |
+------+---------+------------+-------------+
| 2000 | Finland | Computer   |        1500 |
| 2000 | Finland | Phone      |         100 |
| 2000 | Finland | NULL       |        1600 |
| 2000 | India   | Calculator |         150 |
| 2000 | India   | Computer   |        1200 |
| 2000 | India   | NULL       |        1350 |
| 2000 | USA     | Calculator |          75 |
| 2000 | USA     | Computer   |        1500 |
```

```
| 2000 | USA     | NULL       |       1575 |
| 2000 | NULL    | NULL       |       4525 |
| 2001 | Finland | Phone      |         10 |
| 2001 | Finland | NULL       |         10 |
| 2001 | USA     | Calculator |         50 |
| 2001 | USA     | Computer   |       2700 |
| 2001 | USA     | TV         |        250 |
| 2001 | USA     | NULL       |       3000 |
| 2001 | NULL    | NULL       |       3010 |
| NULL | NULL    | NULL       |       7535 |
+------+---------+------------+------------+
```

このクエリーでは、ROLLUP 句を追加すると、出力に 1 つだけでなく、4 つの分析レベルでのサマリー情報が含まれます。次に、ROLLUP 出力の解釈方法を示します。

- 指定された year と country に対応する product 行の各セットに続いて、すべての product の合計を示す追加のサマリー行が生成されます。これらの行には、NULL に設定された product カラムが含まれています。
- 指定された year に対応する行の各セットに続いて、すべての country と product の合計を示す追加のサマリー行が生成されます。これらの行には、NULL に設定された country および products カラムが含まれています。
- 最後に、その他のすべての行に続いて、すべての year、country、および product の総合計を示す追加のサマリー行が生成されます。この行には、NULL に設定された year、country、および products カラムが含まれています。

ROLLUP 使用時のその他の考慮事項

次の項目には、MySQL での ROLLUP の実装に固有の動作の一部を一覧表示します。

ROLLUP を使用する場合は、ORDER BY 句を同時に使用して結果をソートできません。つまり、ROLLUP と ORDER BY は相互に排他的です。ただし、ソート順序を一部制御することはできます。MySQL の GROUP BY によって結果がソートされます。GROUP BY で名前が指定されたカラムで明示的な ASC および DESC キーワードを使用すると、カラムごとにソート順序を指定できます。(ROLLUP で追加された高レベルのサマリー行は、ソート順序には関係なく、計算された行の後ろに表示されます。)

LIMIT を使用すると、クライアントに返される行の数を制限できます。LIMIT は ROLLUP のあとに適用されるため、ROLLUP で追加された追加の行に対して制限が適用されます。例:

```
mysql> SELECT year, country, product, SUM(profit)
    -> FROM sales
    -> GROUP BY year, country, product WITH ROLLUP
    -> LIMIT 5;
+------+---------+------------+-------------+
| year | country | product    | SUM(profit) |
+------+---------+------------+-------------+
| 2000 | Finland | Computer   |        1500 |
| 2000 | Finland | Phone      |         100 |
| 2000 | Finland | NULL       |        1600 |
| 2000 | India   | Calculator |         150 |
| 2000 | India   | Computer   |        1200 |
+------+---------+------------+-------------+
```

LIMIT と ROLLUP を同時に使用すると、超集約行を理解するためのコンテキストが少ないため、解釈がより難しい結果が生成される可能性があります。

各超集約行の NULL 指示子は、行がクライアントに送信されるときに生成されます。サーバーでは、変更された値を持つ左端のカラムに続いて、GROUP BY 句で名前が指定されたカラムが調査されます。これらの名前のいずれかに語彙が一致する名前を持つ結果セット内のカラムでは、その値が NULL に設定されます。(カラム番号別にグループ化カラムを指定すると、サーバーでは NULL に設定するカラムが番号で識別されます。)

超集約行の NULL 値は、クエリー処理のかなり後半の段階で結果セットに配置されるため、クエリー自体で NULL 値としてテストすることはできません。たとえば、クエリーに HAVING product IS NULL を追加しても、超集約行以外のすべてを出力から除去できません。

一方で、NULL 値はクライアント側で NULL として表示され、MySQL クライアントプログラミングインタフェースのいずれかを使用してテストできます。

MySQL では、GROUP BY リストに表示されないカラムを、選択リストで名前を指定することが許可されます。この場合、サーバーはサマリー行内のこのような非集約カラムから任意の値を自由に選択できます。これには、WITH ROLLUP で追加された追加の行も含まれます。たとえば、次のクエリーでは、country は GROUP BY リストに表示されない非集約カラムであり、このカラムで選択された値は不確定です。

```
mysql> SELECT year, country, SUM(profit)
  -> FROM sales GROUP BY year WITH ROLLUP;
+------+---------+-------------+
| year | country | SUM(profit) |
+------+---------+-------------+
| 2000 | India   |        4525 |
| 2001 | USA     |        3010 |
| NULL | USA     |        7535 |
+------+---------+-------------+
```

この動作は、ONLY_FULL_GROUP_BY SQL モードが無効になっている場合に発生します。このモードが有効になっている場合は、country が GROUP BY 句に一覧表示されないため、サーバーはそのクエリーを不正として拒否します。非集約カラムおよび GROUP BY についての詳細は、セクション12.19.3「MySQL での GROUP BY の処理」を参照してください。

12.19.3 MySQL での GROUP BY の処理

標準 SQL では、GROUP BY 句を含むクエリーは、GROUP BY 句で名前が指定されていない選択リスト内の非集約カラムを参照できません。たとえば、このクエリーは、選択リスト内の name カラムが GROUP BY に表示されていないため、標準 SQL では不正です。

```
SELECT o.custid, c.name, MAX(o.payment)
 FROM orders AS o, customers AS c
 WHERE o.custid = c.custid
 GROUP BY o.custid;
```

このクエリーを正当にするには、name カラムを選択リストから削除するか、GROUP BY 句で名前を指定する必要があります。

MySQL では、選択リストが GROUP BY 句で名前が指定されていない非集約カラムを参照できないように、GROUP BY の使用が拡張されています。つまり、上記のクエリーは MySQL では正当です。この機能を使用すると、不要なカラムのソートおよびグループ化が回避されるため、パフォーマンスを改善できます。ただし、これは主に、GROUP BY で名前が指定されていない各非集約カラム内のすべての値がグループごとに同じである場合に役立ちます。サーバーは各グループから任意の値を自由に選択できるため、同じ値でなければ、選択した値は不確定です。さらに、ORDER BY 句を追加しても、各グループからの値の選択が影響を受ける可能性はありません。値が選択されたあとに結果セットのソートが発生しますが、ORDER BY によって、サーバーで選択された各グループ内の値は影響を受けません。

同様の MySQL 拡張が HAVING 句に適用されます。標準 SQL では、GROUP BY 句を含むクエリーは、GROUP BY 句で名前が指定されていない HAVING 句の非集約カラムを参照できません。MySQL 拡張では、計算を簡単にするために、このようなカラムへの参照が許可されます。この拡張では、グループ化されていないカラムに同じグループに関する値が含まれると仮定されます。それ以外の場合は、結果が不確定です。

MySQL GROUP BY の拡張を無効にするには、ONLY_FULL_GROUP_BY SQL モードを有効にします。これにより、標準 SQL の動作が有効になります。GROUP BY 句で名前が指定されていないカラムは、集約関数で囲まなければ、選択リストまたは HAVING 句で使用できません。

また、ONLY_FULL_GROUP_BY によって、HAVING 句のエイリアスの使用も影響を受けません。たとえば、次のクエリーは、orders テーブルで 1 回だけ発生する name 値を返します。

```
SELECT name, COUNT(name) FROM orders
 GROUP BY name
 HAVING COUNT(name) = 1;
```

MySQL では、集約カラム用に HAVING 句でエイリアスを使用することが許可されるように、この動作が拡張されています。

```
SELECT name, COUNT(name) AS c FROM orders
 GROUP BY name
 HAVING c = 1;
```

ONLY_FULL_GROUP_BY を有効にすると、この MySQL 拡張が無効になり、HAVING 句の c カラムが集約関数で囲まれていないため (これは、集約関数です)、「non-grouping field 'c' is used in HAVING clause」というエラーが発生します。

選択リストの拡張は、ORDER BY にも適用されます。つまり、GROUP BY 句に表示されていない ORDER BY 句の非集約カラムは参照できません。(ただし、すでに説明したように、ORDER BY によって、非集約カラムからどの値が選択されるのかは影響を受けません。選択されたあとにはじめてソートされます。)ONLY_FULL_GROUP_BY SQL モードが有効になっている場合は、この拡張が適用されません。

一部のケースでは、MIN() および MAX() を使用すると、一意でない場合でも特定のカラム値を取得できます。sort カラムに 6 桁以内の整数が含まれている場合、次のクエリーは、最小の sort 値を含む行から column の値を取得します。

```
SUBSTR(MIN(CONCAT(LPAD(sort,6,'0'),column)),7)
```

セクション3.6.4「特定のカラムのグループごとの最大値が格納されている行」を参照してください。

標準 SQL に従おうとすると、GROUP BY 句で式を使用できません。回避策として、式のエイリアスを使用します。

```
SELECT id, FLOOR(value/100) AS val
  FROM tbl_name
  GROUP BY id, val;
```

MySQL では、GROUP BY 句で式を使用することが許可されているため、エイリアスは必要ありません。

```
SELECT id, FLOOR(value/100)
  FROM tbl_name
  GROUP BY id, FLOOR(value/100);
```

12.20 高精度計算

MySQL 5.6 は、高精度計算、つまり、きわめて正確な結果が得られ、さらに無効な値を高いレベルで制御できる数値処理をサポートしています。高精度計算は、次の 2 つの機能に基づいています。

- サーバーが無効なデータの受け入れまたは拒否に関してどの程度厳密かを制御する SQL モード。
- 固定小数点数演算のための MySQL ライブラリ。

これらの機能は数値操作にいくつかの影響を与えるとともに、標準 SQL への高度の準拠を実現します。

- 正確な計算: 厳密値数値の場合、計算に浮動小数点エラーは導入されません。代わりに、正確な精度が使用されます。たとえば、MySQL は .0001 などの数値を近似値ではなく厳密値として処理し、それを 10,000 回合計すると 1 に「近い」だけの値ではなく、正確に 1 の結果が生成されます。
- 適切に定義された丸め動作: 厳密値数値の場合、ROUND() の結果は、ベースとなる C ライブラリの動作方法などの環境要因ではなく、その引数に依存します。
- プラットフォームからの独立: 厳密値数値に対する操作は、Windows や Unix などの各プラットフォームにわたって同じです。
- 無効な値の処理に対する制御: オーバーフローおよび 0 による除算が検出可能であり、これらはエラーとして処理できます。たとえば、あるカラムに対して大きすぎる値は、そのカラムのデータ型の範囲内に収まるように値が切り捨てられるようにするのではなく、エラーとして処理できます。同様に、0 による除算は、NULL の結果を生成する操作としてではなく、エラーとして処理できます。どちらのアプローチをとるかの選択は、サーバー SQL モードの設定によって決定されます。

次の説明では、古いアプリケーションとの非互換性の可能性を含め、高精度計算の動作方法のいくつかの側面について触れています。最後に、MySQL 5.6 が数値操作をどのように処理するかを正確に示すいくつかの例を示します。SQL モードの制御については、セクション5.1.7「サーバー SQL モード」を参照してください。

12.20.1 数値の型

厳密値の操作に関する高精度計算のスコープには、厳密値データ型 (整数および DECIMAL 型) と厳密値数値リテラルが含まれます。近似値データ型および数値リテラルは、浮動小数点数として処理されます。

正確値の数値リテラルには、整数部または小数部、あるいはその両方が含まれています。これらには符号を付けることができます。例: 1、.2、3.4、-5、-6.78、+9.10。

近似値の数値リテラルは仮数と指数による指数表現で表されます。一方または両方の部分に符号を付けることができます。例: 1.2E3、1.2E-3、-1.2E3、-1.2E-3。

2 つの数値が同じように見えても、別々に扱われることがあります。たとえば、2.34 は (固定小数点の) 正確値ですが、2.34E0 は (浮動小数点の) 近似値です。

DECIMAL データ型は固定小数点型で、計算は正確です。MySQL では、DECIMAL 型には、NUMERIC、DEC、FIXED という複数のシノニムがあります。整数型も正確値型です。

FLOAT データ型および DOUBLE データ型は浮動小数点型で、計算によって近似値が得られます。MySQL では、FLOAT または DOUBLE のシノニムである型は DOUBLE PRECISION および REAL です。

12.20.2 DECIMAL データ型の特性

このセクションでは、MySQL 5.6 での DECIMAL データ型 (およびそのシノニム) の特性について、特に次のトピックに注意を払いながら説明します。

- 最大桁数
- ストレージフォーマット
- ストレージ要件
- DECIMAL カラムの上限への非標準の MySQL 拡張。

このセクション全体を通して、古いバージョン (5.0.3 より前) の MySQL のために記述されたアプリケーションとの非互換性の可能性が記載されています。

DECIMAL カラムの宣言構文は、DECIMAL(M,D) です。MySQL 5.6 での引数の値の範囲は次のとおりです。

- M は、最大桁数 (精度) です。その範囲は 1 から 65 までです。(古いバージョンの MySQL では 1 から 254 までの範囲が許可されました。)
- D は、小数点の右側の桁数 (スケール) です。その範囲は 0 から 30 までであり、M より大きくすることはできません。

M が 65 の最大値である場合は、DECIMAL 値に対する計算が 65 桁まで正確であることを示します。この 65 桁の精度の制限は厳密値数値リテラルにも適用されるため、このようなリテラルの最大範囲は以前とは異なります。(古いバージョンの MySQL では、10 進数値の最大桁数は 254 でした。ただし、計算は浮動小数点を使用して実行されたために厳密値ではなく、近似値でした。)

MySQL 5.6 での DECIMAL カラムの値は、9 桁の 10 進数が 4 バイトにパックされたバイナリ形式を使用して格納されます。各値の整数部と小数部に対するストレージ要件は個別に決定されます。9 桁の繰り返しごとに 4 バイトが必要であり、残りの桁がある場合は、4 バイトのうちの一部が必要になります。残りの桁に必要なストレージは、次の表で指定されます。

余りの桁	バイト数
0	0
1–2	1
3–4	2
5–6	3
7–9	4

たとえば、DECIMAL(18,9) カラムは小数点の両側に 9 桁あるため、整数部と小数部のそれぞれに 4 バイトが必要です。DECIMAL(20,6) カラムは整数部に 14 桁、小数部に 6 桁あります。整数部の桁は、そのうちの 9 桁に 4 バイト、残りの 5 桁に 3 バイトが必要です。小数部の 6 桁には 3 バイトが必要です。

一部の古いバージョンの MySQL とは異なり、MySQL 5.6 での DECIMAL カラムには、先頭の + 文字や - 文字、または先頭の 0 の桁が格納されません。DECIMAL(5,1) カラムに +0003.1 を挿入すると、それは 3.1 として格納されます。負の数の場合、リテラルの - 文字は格納されません。古い動作に依存するアプリケーションは、この変更が反映されるように変更する必要があります。

MySQL 5.6 での DECIMAL カラムでは、カラム定義によって暗黙的に指定された範囲を超える値が許可されません。たとえば、DECIMAL(3,0) カラムは -999 から 999 までの範囲をサポートします。DECIMAL(M,D) カラムでは、小数点の左側に許可されるのは多くても M - D 桁です。これは、+ 記号の代わりに追加の桁の格納が許可されていた古いバージョンの MySQL に依存するアプリケーションとは互換性がありません。

SQL 標準では、NUMERIC(M,D) の精度は正確に M 桁である必要があります。DECIMAL(M,D) の場合、この標準では少なくとも M 桁の精度が必要ですが、それを超える精度が許可されます。MySQL では、DECIMAL(M,D) と NUMERIC(M,D) は同じであり、どちらも正確に M 桁の精度を持っています。

DECIMAL 値の内部形式の完全な説明については、MySQL ソース配布内のファイル strings/decimal.c を参照してください。この形式は、decimal2bin() 関数で (例を使用して) 説明されています。

DECIMAL データ型の古い処理に依存するアプリケーションの移植の詳細は、『MySQL 5.0 リファレンスマニュアル』を参照してください。

12.20.3 式の処理

高精度計算では、可能な場合は常に、厳密値数値は与えられたとおりに使用されます。たとえば、比較での数値は、値を変更することなく正確に指定されたとおりに使用されます。厳密な SQL モードでは、厳密値データ型 (DECIMAL または整数) を持つカラムへの INSERT の場合、数値の厳密値がそのカラムの範囲内にあれば、その数値が厳密値で挿入されます。取得されると、その値は、挿入された値と同じになるはずです。(厳密な SQL モードが有効になっていない場合は、INSERT での切り捨てが許可されます。)

数値式の処理は、その式にどのような種類の値が含まれているかによって異なります。

- 近似値が存在する場合、その式は近似値であるため、浮動小数点演算を使用して評価されます。
- 近似値が存在しない場合、その式には厳密値のみが含まれています。いずれかの厳密値に小数部 (小数点に続く値) が含まれている場合、その式は DECIMAL の正確な演算を使用して評価され、精度は 65 桁になります。「正確な」という用語は、バイナリで表すことができる精度の制限に従います。たとえば、1.0/3.0 は 10 進数表記で .333... と近似できますが、厳密値数値として記述されないため、(1.0/3.0)*3.0 は正確には 1.0 に評価されません。
- それ以外の場合、式には整数値のみが含まれています。その式は厳密値であり、整数演算を使用して評価され、精度は BIGINT (64 ビット) と同じになります。

数値式に文字列が含まれている場合、その文字列は倍精度浮動小数点値に変換され、その式は近似値になります。

数値カラムへの挿入は、sql_mode システム変数によって制御される SQL モードによって影響を受けます。(セクション5.1.7「サーバー SQL モード」を参照してください。)次の説明では、厳密モード (STRICT_ALL_TABLES または STRICT_TRANS_TABLES モード値によって選択されます) および ERROR_FOR_DIVISION_BY_ZERO について触れています。すべての制限を有効にするには、単純に、厳密モード値と ERROR_FOR_DIVISION_BY_ZERO の両方を含む TRADITIONAL モードを使用できます。

```
mysql> SET sql_mode='TRADITIONAL';
```

数値が厳密値型のカラム (DECIMAL または整数) に挿入された場合は、数値の厳密値がそのカラムの範囲内にあれば、その数値が厳密値で挿入されます。

その値の小数部に含まれている桁が多すぎる場合は、丸めが発生し、警告が生成されます。丸めは、セクション12.20.4「丸め動作」で説明されているとおりに実行されます。

その値の整数部に含まれている桁が多すぎる場合は、その値が大きすぎるため、次のように処理されます。

- 厳密モードが有効になっていない場合は、その値がもっとも近い正当な値に切り捨てられ、警告が生成されます。
- 厳密モードが有効になっている場合は、オーバーフローエラーが発生します。

アンダーフローは検出されないため、アンダーフローの処理は定義されていません。

数値カラムへの文字列の挿入では、その文字列に非数値の内容が含まれている場合、文字列から数値への変換は次のように処理されます。

- 数値で始まらない文字列を数値として使用することはできないため、この文字列により厳密モードではエラーが生成され、それ以外の場合は警告が生成されます。これには、空の文字列が含まれます。
- 数値で始まる文字列は変換できますが、後続の非数値の部分は切り捨てられます。その切り捨てられた部分にスペース以外の文字が含まれている場合、厳密モードではエラーが生成され、それ以外の場合は警告が生成されます。

デフォルトでは、0 による除算によって NULL の結果が生成され、警告は生成されません。SQL モードを適切に設定することにより、0 による除算を制限できます。

ERROR_FOR_DIVISION_BY_ZERO SQL モードが有効になっている場合、MySQL は、0 による除算を次の異なった方法で処理します。

- 厳密モードが有効になっていない場合は、警告が発生します。
- 厳密モードが有効になっている場合は、0 による除算を含む挿入と更新が禁止され、エラーが発生します。

つまり、0 による除算を実行する式を含む挿入や更新をエラーとして処理できますが、それには、厳密モードに加えて ERROR_FOR_DIVISION_BY_ZERO が必要です。

次のステートメントがあるとします。

```
INSERT INTO t SET i = 1/0;
```

次に、厳密モードと ERROR_FOR_DIVISION_BY_ZERO モードの各組み合わせに対する動作を示します。

sql_mode 値	結果
'' (デフォルト)	警告なし、エラーなし。i は NULL に設定されます。
厳密	警告なし、エラーなし。i は NULL に設定されます。
ERROR_FOR_DIVISION_BY_ZERO	警告あり、エラーなし。i は NULL に設定されます。
厳密、ERROR_FOR_DIVISION_BY_ZERO	エラー状態。行は挿入されません。

12.20.4 丸め動作

このセクションでは、ROUND() 関数、および厳密値型 (DECIMAL と整数) を持つカラムへの挿入での高精度計算の丸めについて説明します。

ROUND() 関数は、その引数が厳密値または近似値のどちらであるかに応じて、異なった方法で丸めます。

- 真値の数値の場合、ROUND() は「四捨五入」ルールを使用します。0.5 以上の小数部を持つ値は、正の場合は次の整数に切り上げられ、負の場合は次の整数に切り下げられます。(つまり、ゼロから遠い方に丸められます。)0.5 未満の小数部を持つ値は、正の場合は次の整数に切り下げられ、負の場合は次の整数に切り上げられます。
- 近似値の数字の場合、結果は C ライブラリによって異なります。つまり、多くのシステムでは、ROUND() は「偶数丸め」ルールを使用します。どのような小数部を持つ値も、もっとも近い偶数に丸められます。

次の例では、正確な値の丸めと近似値の丸めの相違点を示します。

```
mysql> SELECT ROUND(2.5), ROUND(25E-1);
+------------+--------------+
| ROUND(2.5) | ROUND(25E-1) |
+------------+--------------+
| 3          |            2 |
+------------+--------------+
```

DECIMAL または整数カラムへの挿入では、ターゲットが厳密値データ型であるため、挿入される値が厳密値または近似値のどちらであるかには関係なく、丸めでは「四捨五入」が使用されます。

```
mysql> CREATE TABLE t (d DECIMAL(10,0));
Query OK, 0 rows affected (0.00 sec)

mysql> INSERT INTO t VALUES(2.5),(2.5E0);
Query OK, 2 rows affected, 2 warnings (0.00 sec)
Records: 2  Duplicates: 0  Warnings: 2

mysql> SELECT d FROM t;
+------+
| d    |
+------+
| 3    |
| 3    |
+------+
```

12.20.5 高精度計算の例

このセクションでは、MySQL 5.6 での高精度計算のクエリー結果を示すいくつかの例について説明します。これらの例は、セクション12.20.3「式の処理」およびセクション12.20.4「丸め動作」で説明されている原則を示しています。

例 1。数値は、可能であれば、指定されたとおりにその厳密値で使用されます。

```
mysql> SELECT (.1 + .2) = .3;
+----------------+
| (.1 + .2) = .3 |
+----------------+
|              1 |
```

```
+---------------+
```

浮動小数点値の場合、結果は不正確です。

```
mysql> SELECT (.1E0 + .2E0) = .3E0;
+---------------------+
| (.1E0 + .2E0) = .3E0 |
+---------------------+
|                   0 |
+---------------------+
```

厳密値と近似値の処理の違いを確認するための別の方法として、合計値に小さい数値を何回も加える方法があります。ある変数に .0001 を 1,000 回加える次のストアドプロシージャーを考えてみます。

```
CREATE PROCEDURE p ()
BEGIN
  DECLARE i INT DEFAULT 0;
  DECLARE d DECIMAL(10,4) DEFAULT 0;
  DECLARE f FLOAT DEFAULT 0;
  WHILE i < 10000 DO
    SET d = d + .0001;
    SET f = f + .0001E0;
    SET i = i + 1;
  END WHILE;
  SELECT d, f;
END;
```

論理的には、d と f の両方の合計値が 1 になるはずですが、それは 10 進数の計算にしか当てはまりません。浮動小数点の計算では、小さな エラーが発生します。

```
+--------+------------------+
| d      | f                |
+--------+------------------+
| 1.0000 | 0.99999999999991 |
+--------+------------------+
```

例 2。乗算は、標準 SQL に必要なスケールで実行されます。つまり、スケール S1 と S2 を持つ 2 つの数値 X1 と X2 の場合、結果のスケールは S1 + S2 になります。

```
mysql> SELECT .01 * .01;
+-----------+
| .01 * .01 |
+-----------+
| 0.0001    |
+-----------+
```

例 3。厳密値数値に対する丸め動作は、適切に定義されています。

丸め動作 (たとえば、ROUND() 関数を使用した動作) は、ベースとなる C ライブラリの実装には依存しません。つまり、その結果は各プラットフォームにわたって一貫性があります。

- 厳密値カラム (DECIMAL と整数) および厳密値数値に対する丸めでは、「四捨五入」ルールが使用されます。0.5 以上の小数部を持つ値は、次に示すように、0 から遠いもっとも近い整数に丸められます。

  ```
  mysql> SELECT ROUND(2.5), ROUND(-2.5);
  +------------+-------------+
  | ROUND(2.5) | ROUND(-2.5) |
  +------------+-------------+
  | 3          | -3          |
  +------------+-------------+
  ```

- 浮動小数点値に対する丸めでは、C ライブラリが使用されます。これは、多くのシステムでは「偶数丸め」ルールを使用します。このようなシステムでは、どのような小数部を持つ値も、もっとも近い偶数に丸められます。

  ```
  mysql> SELECT ROUND(2.5E0), ROUND(-2.5E0);
  +--------------+---------------+
  | ROUND(2.5E0) | ROUND(-2.5E0) |
  +--------------+---------------+
  |            2 |            -2 |
  +--------------+---------------+
  ```

例 4。厳密モードでは、カラムの範囲を外れている値を挿入すると、正当な値への切り捨てではなく、エラーが発生します。

MySQL が厳密モードで実行されていない場合は、正当な値への切り捨てが発生します。

```
mysql> SET sql_mode='';
Query OK, 0 rows affected (0.00 sec)

mysql> CREATE TABLE t (i TINYINT);
Query OK, 0 rows affected (0.01 sec)

mysql> INSERT INTO t SET i = 128;
Query OK, 1 row affected, 1 warning (0.00 sec)

mysql> SELECT i FROM t;
+------+
| i    |
+------+
|  127 |
+------+
1 row in set (0.00 sec)
```

ただし、厳密モードが有効になっている場合は、エラーが発生します。

```
mysql> SET sql_mode='STRICT_ALL_TABLES';
Query OK, 0 rows affected (0.00 sec)

mysql> CREATE TABLE t (i TINYINT);
Query OK, 0 rows affected (0.00 sec)

mysql> INSERT INTO t SET i = 128;
ERROR 1264 (22003): Out of range value adjusted for column 'i' at row 1

mysql> SELECT i FROM t;
Empty set (0.00 sec)
```

例 5: 厳密モードで ERROR_FOR_DIVISION_BY_ZERO が設定されている場合は、0 による除算によって NULL の結果ではなく、エラーが発生します。

非厳密モードでは、0 による除算によって NULL の結果が生成されます。

```
mysql> SET sql_mode='';
Query OK, 0 rows affected (0.01 sec)

mysql> CREATE TABLE t (i TINYINT);
Query OK, 0 rows affected (0.00 sec)

mysql> INSERT INTO t SET i = 1 / 0;
Query OK, 1 row affected (0.00 sec)

mysql> SELECT i FROM t;
+------+
| i    |
+------+
| NULL |
+------+
1 row in set (0.03 sec)
```

ただし、適切な SQL モードが有効になっている場合、0 による除算はエラーになります。

```
mysql> SET sql_mode='STRICT_ALL_TABLES,ERROR_FOR_DIVISION_BY_ZERO';
Query OK, 0 rows affected (0.00 sec)

mysql> CREATE TABLE t (i TINYINT);
Query OK, 0 rows affected (0.00 sec)

mysql> INSERT INTO t SET i = 1 / 0;
ERROR 1365 (22012): Division by 0

mysql> SELECT i FROM t;
Empty set (0.01 sec)
```

例 6。厳密値リテラルは、厳密値として評価されます。

MySQL 5.0.3 より前は、厳密値リテラルと近似値リテラルはどちらも、倍精度浮動小数点値として評価されます。

```
mysql> SELECT VERSION();
+------------+
| VERSION()  |
+------------+
| 4.1.18-log |
+------------+
```

```
1 row in set (0.01 sec)

mysql> CREATE TABLE t SELECT 2.5 AS a, 25E-1 AS b;
Query OK, 1 row affected (0.07 sec)
Records: 1  Duplicates: 0  Warnings: 0

mysql> DESCRIBE t;
+-------+-------------+------+-----+---------+-------+
| Field | Type        | Null | Key | Default | Extra |
+-------+-------------+------+-----+---------+-------+
| a     | double(3,1) |      |     | 0.0     |       |
| b     | double      |      |     | 0       |       |
+-------+-------------+------+-----+---------+-------+
2 rows in set (0.04 sec)
```

MySQL 5.0.3 の時点では、近似値リテラルは浮動小数点を使用して評価されますが、厳密値リテラルは DECIMAL として処理されます。

```
mysql> SELECT VERSION();
+----------------+
| VERSION()      |
+----------------+
| 5.1.6-alpha-log |
+----------------+
1 row in set (0.11 sec)

mysql> CREATE TABLE t SELECT 2.5 AS a, 25E-1 AS b;
Query OK, 1 row affected (0.01 sec)
Records: 1  Duplicates: 0  Warnings: 0

mysql> DESCRIBE t;
+-------+----------------------+------+-----+---------+-------+
| Field | Type                 | Null | Key | Default | Extra |
+-------+----------------------+------+-----+---------+-------+
| a     | decimal(2,1) unsigned | NO   |     | 0.0     |       |
| b     | double               | NO   |     | 0       |       |
+-------+----------------------+------+-----+---------+-------+
2 rows in set (0.01 sec)
```

例 7。集約関数への引数が厳密値数値型である場合は、その結果も、少なくとも引数と同じスケールを持つ厳密値数値型になります。

これらのステートメントを考慮してください。

```
mysql> CREATE TABLE t (i INT, d DECIMAL, f FLOAT);
mysql> INSERT INTO t VALUES(1,1,1);
mysql> CREATE TABLE y SELECT AVG(i), AVG(d), AVG(f) FROM t;
```

MySQL 5.0.3 より前は、引数の型にかかわらず、結果が倍精度値になります。

```
mysql> DESCRIBE y;
+--------+--------------+------+-----+---------+-------+
| Field  | Type         | Null | Key | Default | Extra |
+--------+--------------+------+-----+---------+-------+
| AVG(i) | double(17,4) | YES  |     | NULL    |       |
| AVG(d) | double(17,4) | YES  |     | NULL    |       |
| AVG(f) | double       | YES  |     | NULL    |       |
+--------+--------------+------+-----+---------+-------+
```

MySQL 5.0.3 の時点では、結果が倍精度値になるのは浮動小数点引数の場合だけです。厳密値型引数の場合は、結果も厳密値型になります。

```
mysql> DESCRIBE y;
+--------+---------------+------+-----+---------+-------+
| Field  | Type          | Null | Key | Default | Extra |
+--------+---------------+------+-----+---------+-------+
| AVG(i) | decimal(14,4) | YES  |     | NULL    |       |
| AVG(d) | decimal(14,4) | YES  |     | NULL    |       |
| AVG(f) | double        | YES  |     | NULL    |       |
+--------+---------------+------+-----+---------+-------+
```

結果が倍精度値になるのは浮動小数点引数の場合だけです。厳密値型引数の場合は、結果も厳密値型になります。

第 13 章 SQL ステートメントの構文

目次

13.1 データ定義ステートメント 1222
13.1.1 ALTER DATABASE 構文 1222
13.1.2 ALTER EVENT 構文 1223
13.1.3 ALTER LOGFILE GROUP 構文 1224
13.1.4 ALTER FUNCTION 構文 1225
13.1.5 ALTER PROCEDURE 構文 1226
13.1.6 ALTER SERVER 構文 1226
13.1.7 ALTER TABLE 構文 1226
13.1.8 ALTER TABLESPACE 構文 1244
13.1.9 ALTER VIEW 構文 1245
13.1.10 CREATE DATABASE 構文 1245
13.1.11 CREATE EVENT 構文 1246
13.1.12 CREATE FUNCTION 構文 1250
13.1.13 CREATE INDEX 構文 1250
13.1.14 CREATE LOGFILE GROUP 構文 1253
13.1.15 CREATE PROCEDURE および CREATE FUNCTION 構文 1254
13.1.16 CREATE SERVER 構文 1259
13.1.17 CREATE TABLE 構文 1260
13.1.18 CREATE TABLESPACE 構文 1285
13.1.19 CREATE TRIGGER 構文 1287
13.1.20 CREATE VIEW 構文 1289
13.1.21 DROP DATABASE 構文 1292
13.1.22 DROP EVENT 構文 1293
13.1.23 DROP FUNCTION 構文 1293
13.1.24 DROP INDEX 構文 1294
13.1.25 DROP LOGFILE GROUP 構文 1294
13.1.26 DROP PROCEDURE および DROP FUNCTION 構文 1295
13.1.27 DROP SERVER 構文 1295
13.1.28 DROP TABLE 構文 1295
13.1.29 DROP TABLESPACE 構文 1296
13.1.30 DROP TRIGGER 構文 1296
13.1.31 DROP VIEW 構文 1296
13.1.32 RENAME TABLE 構文 1296
13.1.33 TRUNCATE TABLE 構文 1297
13.2 データ操作ステートメント 1298
13.2.1 CALL 構文 1298
13.2.2 DELETE 構文 1300
13.2.3 DO 構文 1303
13.2.4 HANDLER 構文 1304
13.2.5 INSERT 構文 1305
13.2.6 LOAD DATA INFILE 構文 1313
13.2.7 LOAD XML 構文 1321
13.2.8 REPLACE 構文 1326
13.2.9 SELECT 構文 1328
13.2.10 サブクエリー構文 1344
13.2.11 UPDATE 構文 1353
13.3 MySQL トランザクションおよびロックステートメント 1355
13.3.1 START TRANSACTION、COMMIT、および ROLLBACK 構文 1355
13.3.2 ロールバックできないステートメント 1358
13.3.3 暗黙的なコミットを発生させるステートメント 1358
13.3.4 SAVEPOINT、ROLLBACK TO SAVEPOINT、および RELEASE SAVEPOINT 構文 1359
13.3.5 LOCK TABLES および UNLOCK TABLES 構文 1360
13.3.6 SET TRANSACTION 構文 1365
13.3.7 XA トランザクション 1367
13.4 レプリケーションステートメント 1371
13.4.1 マスターサーバーを制御するための SQL ステートメント 1371
13.4.2 スレーブサーバーを制御するための SQL ステートメント 1373
13.5 準備済みステートメントのための SQL 構文 1382

13.5.1 PREPARE 構文 1386
13.5.2 EXECUTE 構文 1386
13.5.3 DEALLOCATE PREPARE 構文 1386
13.6 MySQL 複合ステートメント構文 1387
13.6.1 BEGIN ... END 複合ステートメント構文 1387
13.6.2 ステートメントラベルの構文 1387
13.6.3 DECLARE 構文 1388
13.6.4 ストアドプログラム内の変数 1388
13.6.5 フロー制御ステートメント 1389
13.6.6 カーソル 1393
13.6.7 条件の処理 1394
13.7 データベース管理ステートメント 1414
13.7.1 アカウント管理ステートメント 1414
13.7.2 テーブル保守ステートメント 1430
13.7.3 プラグインおよびユーザー定義関数ステートメント 1438
13.7.4 SET 構文 1440
13.7.5 SHOW 構文 1443
13.7.6 その他の管理ステートメント 1480
13.8 MySQL ユーティリティーステートメント 1488
13.8.1 DESCRIBE 構文 1488
13.8.2 EXPLAIN 構文 1488
13.8.3 HELP 構文 1489
13.8.4 USE 構文 1491

この章では、MySQL によってサポートされる SQL ステートメントの構文について説明します。

13.1 データ定義ステートメント

13.1.1 ALTER DATABASE 構文

```
ALTER {DATABASE | SCHEMA} [db_name]
    alter_specification ...
ALTER {DATABASE | SCHEMA} db_name
    UPGRADE DATA DIRECTORY NAME

alter_specification:
    [DEFAULT] CHARACTER SET [=] charset_name
  | [DEFAULT] COLLATE [=] collation_name
```

ALTER DATABASE を使用すると、データベースの全体的な特性を変更できます。これらの特性は、データベースディレクトリ内の db.opt ファイルに格納されます。ALTER DATABASE を使用するには、そのデータベースに対する ALTER 権限が必要です。ALTER SCHEMA は ALTER DATABASE のシノニムです。

最初の構文からはデータベース名を省略できます。その場合、このステートメントはデフォルトデータベースに適用されます。

各国語に関する特性

CHARACTER SET 句は、デフォルトのデータベース文字セットを変更します。COLLATE 句は、デフォルトのデータベース照合順序を変更します。セクション10.1「文字セットのサポート」では、文字セットと照合順序名について説明しています。

どのような文字セットと照合順序を使用できるかは、それぞれ SHOW CHARACTER SET および SHOW COLLATION ステートメントを使用して確認できます。詳細は、セクション13.7.5.4「SHOW CHARACTER SET 構文」およびセクション13.7.5.5「SHOW COLLATION 構文」を参照してください。

データベースのデフォルトの文字セットまたは照合順序を変更する場合、データベースのデフォルトを使用するストアドルーチンを削除して、新しいデフォルトを使用するように再作成する必要があります。(ストアドルーチンでは、文字セットまたは照合順序が明示的に指定されていない場合、文字データ型を伴う変数は、データベースのデフォルトを使用します。セクション13.1.15「CREATE PROCEDURE および CREATE FUNCTION 構文」を参照してください。)

MySQL 5.1 より古いバージョンからのアップグレード

UPGRADE DATA DIRECTORY NAME 句を含む構文は、データベースに関連付けられたディレクトリの名前を、データベース名をデータベースディレクトリ名にマップするための MySQL 5.1 で実装されたエンコーディングを

使用するように更新します (セクション9.2.3「識別子とファイル名のマッピング」を参照してください)。この句は、次の条件の下で使用されることを目的にしています。

- MySQL を古いバージョンから 5.1 以降にアップグレードすることを目的にしている。
- データベースディレクトリ名にエンコーディングが必要な特殊文字が含まれている場合は、その名前を現在のエンコーディング形式に更新することを目的にしている。
- このステートメントが (mysql_upgrade から呼び出された) mysqlcheck によって使用されている。

たとえば、MySQL 5.0 でのデータベース名が a-b-c である場合、その名前には - (ダッシュ) 文字のインスタンスが含まれています。MySQL 5.0 では、データベースディレクトリの名前も、必ずしもすべてのファイルシステムで安全ではない a-b-c になります。MySQL 5.1 以降では、ファイルシステムに依存しないディレクトリ名を生成するために、同じデータベース名が a@002db@002dc としてエンコードされます。

MySQL インストールが古いバージョンから MySQL 5.1 以降にアップグレードされると、サーバーでは a-b-c などの (古い形式の) 名前が #mysql50#a-b-c として表示されるため、#mysql50# プリフィクスを使用して名前を参照する必要があります。この場合は、UPGRADE DATA DIRECTORY NAME を使用して、データベースディレクトリ名を現在のエンコーディング形式に再エンコーディングするようサーバーに明示的に指示します。

```
ALTER DATABASE `#mysql50#a-b-c` UPGRADE DATA DIRECTORY NAME;
```

このステートメントを実行したあとは、特殊な #mysql50# プリフィクスなしで、そのデータベースを a-b-c として参照できます。

13.1.2 ALTER EVENT 構文

```
ALTER
    [DEFINER = { user | CURRENT_USER }]
    EVENT event_name
    [ON SCHEDULE schedule]
    [ON COMPLETION [NOT] PRESERVE]
    [RENAME TO new_event_name]
    [ENABLE | DISABLE | DISABLE ON SLAVE]
    [COMMENT 'comment']
    [DO event_body]
```

ALTER EVENT ステートメントは、既存のイベントの 1 つ以上の特性を、そのイベントを削除して再作成することなく変更します。DEFINER、ON SCHEDULE、ON COMPLETION、COMMENT、ENABLE/DISABLE、DO の各句の構文は、CREATE EVENT で使用される場合とまったく同じです。(セクション13.1.11「CREATE EVENT 構文」を参照してください。)

どのユーザーも、そのユーザーが EVENT 権限を持っているデータベースで定義されたイベントを変更できます。ユーザーが正常な ALTER EVENT ステートメントを実行すると、そのユーザーは、影響を受けるイベントの定義者になります。

ALTER EVENT は、既存のイベントでのみ機能します。

```
mysql> ALTER EVENT no_such_event
    >     ON SCHEDULE
    >       EVERY '2:3' DAY_HOUR;
ERROR 1517 (HY000): Unknown event 'no_such_event'
```

次の各例では、myevent という名前のイベントが次に示すように定義されていることを前提にしています。

```
CREATE EVENT myevent
    ON SCHEDULE
      EVERY 6 HOUR
    COMMENT 'A sample comment.'
    DO
      UPDATE myschema.mytable SET mycol = mycol + 1;
```

次のステートメントは、myevent のスケジュールを、ただちに開始して 6 時間ごとに 1 回から、ステートメントが実行された 4 時間後から開始して 12 時間ごとに 1 回に変更します。

```
ALTER EVENT myevent
    ON SCHEDULE
      EVERY 12 HOUR
    STARTS CURRENT_TIMESTAMP + INTERVAL 4 HOUR;
```

イベントの複数の特性を 1 つのステートメントで変更できます。この例では、myevent によって実行される SQL ステートメントを、mytable のすべてのレコードを削除する SQL ステートメントに変更します。また、イベント

のスケジュールも、この ALTER EVENT ステートメントが実行された 1 日あとに 1 回実行されるように変更します。

```
ALTER EVENT myevent
    ON SCHEDULE
      AT CURRENT_TIMESTAMP + INTERVAL 1 DAY
    DO
      TRUNCATE TABLE myschema.mytable;
```

ALTER EVENT ステートメントでは、変更したい特性のオプションのみを指定します。省略されたオプションでは、その既存の値が保持されます。これには、ENABLE などの、CREATE EVENT のデフォルト値もすべて含まれます。

myevent を無効にするには、この ALTER EVENT ステートメントを使用します。

```
ALTER EVENT myevent
    DISABLE;
```

ON SCHEDULE 句では、組み込みの MySQL 関数やユーザー変数を含む式を使用して、そこに含まれているすべての timestamp または interval 値を取得できます。このような式でストアドルーチンやユーザー定義関数を使用したり、テーブル参照を使用したりすることはできません。ただし、SELECT FROM DUAL は使用できます。これは、ALTER EVENT ステートメントと CREATE EVENT ステートメントの両方に当てはまります。このような場合のストアドルーチン、ユーザー定義関数、およびテーブルへの参照は明確に禁止されており、エラーで失敗します (Bug #22830 を参照してください)。

DO 句に別の ALTER EVENT ステートメントを含む ALTER EVENT ステートメントは成功したように見えますが、結果として得られるスケジュールされたイベントをサーバーが実行しようとすると、その実行はエラーで失敗します。

イベントの名前を変更するには、ALTER EVENT ステートメントの RENAME TO 句を使用します。このステートメントは、イベント myevent の名前を yourevent に変更します。

```
ALTER EVENT myevent
    RENAME TO yourevent;
```

次に示すように、ALTER EVENT ... RENAME TO ... と db_name.event_name 表記を使用して、イベントを別のデータベースに移動することもできます。

```
ALTER EVENT olddb.myevent
    RENAME TO newdb.myevent;
```

前のステートメントを実行するには、それを実行するユーザーが、olddb および newdb データベースの両方に対する EVENT 権限を持っている必要があります。

> **注記**
>
> RENAME EVENT ステートメントはありません。

値 DISABLE ON SLAVE は、マスター上で作成されてスレーブにレプリケートされたが、まだスレーブ上で実行されていないイベントを示すために、ENABLE または DISABLE の代わりにレプリケーションスレーブに対して使用されます。通常、DISABLE ON SLAVE は必要に応じて自動的に設定されます。ただし、手動で変更することが必要になる場合もあります。詳細は、セクション17.4.1.11「呼び出される機能のレプリケーション」を参照してください。

13.1.3 ALTER LOGFILE GROUP 構文

```
ALTER LOGFILE GROUP logfile_group
    ADD UNDOFILE 'file_name'
    [INITIAL_SIZE [=] size]
    [WAIT]
    ENGINE [=] engine_name
```

このステートメントは、'file_name' という名前の UNDO ファイルを既存のログファイルグループ logfile_group に追加します。ALTER LOGFILE GROUP ステートメントには、ADD UNDOFILE 句が 1 つだけ存在します。DROP UNDOFILE 句は、現在サポートされていません。

> **注記**
>
> すべての MySQL Cluster ディスクデータオブジェクトが同じ名前空間を共有します。つまり、各ディスクデータオブジェクトは (単に、特定の型の各ディスクデータオブジェ

> クトというだけでなく)、一意の名前が付けられている必要があります。たとえば、テーブルスペースと Undo ログファイルを同じ名前にしたり、Undo ログファイルとデータファイルを同じ名前にしたりすることはできません。

オプションの INITIAL_SIZE パラメータは、UNDO ファイルの初期サイズをバイト単位で設定します。指定されていない場合、初期サイズはデフォルトで 134217728 (128M バイト) になります。MySQL Cluster NDB 7.3.2 より前は、この値は数字で指定する必要がありました。MySQL Cluster NDB 7.3.2 以降では、size のあとにオプションで、my.cnf で使用されるのと同様の、オーダーを示す 1 文字の略語を指定できます。一般に、これは M (M バイト) または G (G バイト) のどちらかの文字です。(Bug #13116514、Bug #16104705、Bug #62858)

32 ビットシステム上では、INITIAL_SIZE のサポートされる最大値は 4294967296 (4G バイト) です。(Bug #29186)

INITIAL_SIZE の許可される最小値は 1048576 (1M バイト) です。(Bug #29574)

> 注記
>
> WAIT は解析されますが、それ以外は無視されます。このキーワードは現在何の効果もなく、将来の拡張のために用意されています。

ENGINE パラメータ (必須) は、このログファイルグループによって使用されるストレージエンジンを決定します。ここで、engine_name はそのストレージエンジンの名前です。現在、engine_name として受け入れられる値は「NDBCLUSTER」と「NDB」だけです。この 2 つの値は同等です。

次の例では、ログファイルグループ lg_3 がすでに CREATE LOGFILE GROUP を使用して作成されていることを前提にしています (セクション13.1.14「CREATE LOGFILE GROUP 構文」を参照してください)。

```
ALTER LOGFILE GROUP lg_3
   ADD UNDOFILE 'undo_10.dat'
   INITIAL_SIZE=32M
   ENGINE=NDBCLUSTER;
```

ALTER LOGFILE GROUP が ENGINE = NDBCLUSTER (または、ENGINE = NDB) とともに使用された場合は、UNDO ログファイルが各 MySQL Cluster データノード上に作成されます。INFORMATION_SCHEMA.FILES テーブルをクエリーすることによって、UNDO ファイルが作成されたことを確認したり、それらに関する情報を取得したりできます。例:

```
mysql> SELECT FILE_NAME, LOGFILE_GROUP_NUMBER, EXTRA
    -> FROM INFORMATION_SCHEMA.FILES
    -> WHERE LOGFILE_GROUP_NAME = 'lg_3';
+------------+--------------------+----------------+
| FILE_NAME  | LOGFILE_GROUP_NUMBER | EXTRA          |
+------------+--------------------+----------------+
| newdata.dat |                  0 | CLUSTER_NODE=3 |
| newdata.dat |                  0 | CLUSTER_NODE=4 |
| undo_10.dat |                 11 | CLUSTER_NODE=3 |
| undo_10.dat |                 11 | CLUSTER_NODE=4 |
+------------+--------------------+----------------+
4 rows in set (0.01 sec)
```

(セクション21.30.1「INFORMATION_SCHEMA FILES テーブル」を参照してください。)

UNDO_BUFFER_SIZE に使用されるメモリーは、サイズが SharedGlobalMemory データノード構成パラメータの値によって決定されるグローバルプールから取得されます。これには、InitialLogFileGroup データノード構成パラメータの設定により、このオプションに暗黙的に指定されるデフォルト値もすべて含まれます。

ALTER LOGFILE GROUP は、MySQL Cluster のディスクデータストレージでのみ有効です。詳細は、セクション18.5.12「MySQL Cluster ディスクデータテーブル」を参照してください。

13.1.4 ALTER FUNCTION 構文

```
ALTER FUNCTION func_name [characteristic ...]

characteristic:
   COMMENT 'string'
 | LANGUAGE SQL
 | { CONTAINS SQL | NO SQL | READS SQL DATA | MODIFIES SQL DATA }
 | SQL SECURITY { DEFINER | INVOKER }
```

このステートメントを使用すると、ストアドファンクションの特性を変更できます。ALTER FUNCTION ステートメントでは、複数の変更を指定できます。ただし、このステートメントを使用して、ストアドファンクショ

ンのパラメータまたは本体を変更することはできません。このような変更を行うには、DROP FUNCTION と CREATE FUNCTION を使用して、この関数を削除および再作成する必要があります。

この関数に対する ALTER ROUTINE 権限が必要です。(その権限は、関数作成者に自動的に付与されます。)バイナリロギングが有効になっている場合は、セクション20.7「ストアドプログラムのバイナリロギング」で説明されているように、ALTER FUNCTION ステートメントに SUPER 権限も必要になる可能性があります。

13.1.5 ALTER PROCEDURE 構文

```
ALTER PROCEDURE proc_name [characteristic ...]

characteristic:
    COMMENT 'string'
  | LANGUAGE SQL
  | { CONTAINS SQL | NO SQL | READS SQL DATA | MODIFIES SQL DATA }
  | SQL SECURITY { DEFINER | INVOKER }
```

このステートメントを使用すると、ストアドプロシージャーの特性を変更できます。ALTER PROCEDURE ステートメントでは、複数の変更を指定できます。ただし、このステートメントを使用して、ストアドプロシージャーのパラメータまたは本体を変更することはできません。このような変更を行うには、DROP PROCEDURE と CREATE PROCEDURE を使用して、このプロシージャーを削除および再作成する必要があります。

このプロシージャーに対する ALTER ROUTINE 権限が必要です。デフォルトでは、その権限は、プロシージャー作成者に自動的に付与されます。この動作は、automatic_sp_privileges システム変数を無効にすることによって変更できます。セクション20.2.2「ストアドルーチンと MySQL 権限」を参照してください。

13.1.6 ALTER SERVER 構文

```
ALTER SERVER  server_name
    OPTIONS (option [, option] ...)
```

server_name のサーバー情報を変更して、CREATE SERVER ステートメント内で許可されているオプションのいずれかを調整します。それに応じて、mysql.servers テーブル内の対応するフィールドが更新されます。このステートメントには、SUPER 権限が必要です。

たとえば、USER オプションを更新するには、次のようにします。

```
ALTER SERVER s OPTIONS (USER 'sally');
```

ALTER SERVER では、自動コミットは実行されません。

MySQL 5.6 では、使用されている口ギング形式には関係なく、ALTER SERVER はバイナリログに書き込まれません。

MySQL 5.6.11 でのみ、このステートメントを発行する前に、gtid_next を AUTOMATIC に設定する必要があります。(Bug #16062608、Bug #16715809、Bug #69045)

13.1.7 ALTER TABLE 構文

```
ALTER [ONLINE|OFFLINE] [IGNORE] TABLE tbl_name
    [alter_specification [, alter_specification] ...]
    [partition_options]

alter_specification:
    table_options
  | ADD [COLUMN] col_name column_definition
        [FIRST | AFTER col_name ]
  | ADD [COLUMN] (col_name column_definition,...)
  | ADD {INDEX|KEY} [index_name]
        [index_type] (index_col_name,...) [index_option] ...
  | ADD [CONSTRAINT [symbol]] PRIMARY KEY
        [index_type] (index_col_name,...) [index_option] ...
  | ADD [CONSTRAINT [symbol]]
        UNIQUE [INDEX|KEY] [index_name]
        [index_type] (index_col_name,...) [index_option] ...
  | ADD FULLTEXT [INDEX|KEY] [index_name]
        (index_col_name,...) [index_option] ...
  | ADD SPATIAL [INDEX|KEY] [index_name]
        (index_col_name,...) [index_option] ...
  | ADD [CONSTRAINT [symbol]]
        FOREIGN KEY [index_name] (index_col_name,...)
```

```
      reference_definition
  | ALGORITHM [=] {DEFAULT|INPLACE|COPY}
  | ALTER [COLUMN] col_name {SET DEFAULT literal | DROP DEFAULT}
  | CHANGE [COLUMN] old_col_name new_col_name column_definition
      [FIRST|AFTER col_name]
  | LOCK [=] {DEFAULT|NONE|SHARED|EXCLUSIVE}
  | MODIFY [COLUMN] col_name column_definition
      [FIRST | AFTER col_name]
  | DROP [COLUMN] col_name
  | DROP PRIMARY KEY
  | DROP {INDEX|KEY} index_name
  | DROP FOREIGN KEY fk_symbol
  | DISABLE KEYS
  | ENABLE KEYS
  | RENAME [TO|AS] new_tbl_name
  | ORDER BY col_name [, col_name] ...
  | CONVERT TO CHARACTER SET charset_name [COLLATE collation_name]
  | [DEFAULT] CHARACTER SET [=] charset_name [COLLATE [=] collation_name]
  | DISCARD TABLESPACE
  | IMPORT TABLESPACE
  | FORCE
  | ADD PARTITION (partition_definition)
  | DROP PARTITION partition_names
  | TRUNCATE PARTITION {partition_names | ALL}
  | COALESCE PARTITION number
  | REORGANIZE PARTITION partition_names INTO (partition_definitions)
  | EXCHANGE PARTITION partition_name WITH TABLE tbl_name
  | ANALYZE PARTITION {partition_names | ALL}
  | CHECK PARTITION {partition_names | ALL}
  | OPTIMIZE PARTITION {partition_names | ALL}
  | REBUILD PARTITION {partition_names | ALL}
  | REPAIR PARTITION {partition_names | ALL}
  | REMOVE PARTITIONING

index_col_name:
    col_name [(length)] [ASC | DESC]

index_type:
    USING {BTREE | HASH}

index_option:
    KEY_BLOCK_SIZE [=] value
  | index_type
  | WITH PARSER parser_name
  | COMMENT 'string'

table_options:
    table_option [[,] table_option] ...  (see CREATE TABLE options)

partition_options:
    (see CREATE TABLE options)
```

ALTER TABLE は、テーブルの構造を変更します。たとえば、カラムを追加または削除したり、インデックスを作成または破棄したり、既存のカラムの型を変更したり、カラムまたはテーブル自体の名前を変更したりできます。また、テーブルに使用されているストレージエンジンやテーブルのコメントなどの特性を変更することもできます。

テーブル名のあとに、行う変更を指定します。何も指定されていない場合、ALTER TABLE は何もしません。

許可される多くの変更の構文は、CREATE TABLE ステートメントの各句に似ています。詳細は、セクション 13.1.17「CREATE TABLE 構文」を参照してください。

table_options は、ENGINE、AUTO_INCREMENT、AVG_ROW_LENGTH、MAX_ROWS、ROW_FORMAT などの、CREATE TABLE ステートメントで使用できる種類のテーブルオプションを示します。すべてのテーブルオプションのリストと各オプションの説明については、セクション13.1.17「CREATE TABLE 構文」を参照してください。ただし、ALTER TABLE は、DATA DIRECTORY および INDEX DIRECTORY テーブルオプションを無視します。

partition_options は、再パーティション化、パーティションの追加、削除、マージ、および分割、パーティション化の保守の実行などのために、パーティション化されたテーブルで使用できるオプションを示します。ALTER TABLE ステートメントには、ほかの変更指定に加えて、PARTITION BY または REMOVE PARTITIONING 句を含めることができますが、PARTITION BY または REMOVE PARTITIONING 句は、ほかのどの指定よりもあとの最後に指定する必要があります。ADD PARTITION、DROP PARTITION、COALESCE PARTITION、REORGANIZE PARTITION、EXCHANGE PARTITION、ANALYZE PARTITION、CHECK PARTITION、および REPAIR PARTITION オプションは、個々のパーティションに対して機能するため、1 つ

の ALTER TABLE 内でほかの変更指定と組み合わせることはできません。パーティションのオプションの詳細は、セクション13.1.17「CREATE TABLE 構文」およびセクション13.1.7.1「ALTER TABLE パーティション操作」を参照してください。ALTER TABLE ... EXCHANGE PARTITION ステートメントの詳細および例については、セクション19.3.1「RANGE および LIST パーティションの管理」を参照してください。

操作によっては、ストレージエンジンがその操作をサポートしていないテーブルに対して試行されると、警告が生成される場合があります。これらの警告は、SHOW WARNINGS で表示できます。セクション13.7.5.41「SHOW WARNINGS 構文」を参照してください。

ALTER TABLE のトラブルシューティングについては、セクションB.5.7.1「ALTER TABLE での問題」を参照してください。

ストレージ、パフォーマンス、および並列性に関する考慮事項

ほとんどの場合、ALTER TABLE は元のテーブルの一時的なコピーを作成します。MySQL は、そのテーブルを変更しているほかの操作を待ってから、処理を続行します。そのコピーに変更を組み込み、元のテーブルを削除したあと、新しいテーブルの名前を変更します。ALTER TABLE の実行中、ほかのセッションは元のテーブルを読み取ることができます (例外については、すぐあとに説明します)。ALTER TABLE 操作の開始後に開始されたテーブルへの更新や書き込みは、新しいテーブルの準備ができるまで停止されてから、どの更新も失敗することなく、新しいテーブルに自動的にリダイレクトされます。元のテーブルの一時的なコピーは、新しいテーブルのデータベースディレクトリ内に作成されます。これは、テーブルの名前を別のデータベースに変更した ALTER TABLE 操作のための元のテーブルのデータベースディレクトリとは異なる場合があります。

前に触れた例外とは、ALTER TABLE が、テーブルの .frm ファイルの新しいバージョンをインストールし、古いファイルを破棄して、テーブルおよびテーブル定義キャッシュから古くなったテーブル構造をクリアする準備ができた時点で (書き込みだけでなく) 読み取りをブロックすることです。この時点で、このステートメントは排他的ロックを取得する必要があります。それを行うために、現在の読み取り側が完了するのを待って、新しい読み取り (および書き込み) をブロックします。

MyISAM テーブルの場合は、myisam_sort_buffer_size システム変数を大きな値に設定することによって、インデックスの再作成 (変更プロセスのもっとも遅い部分) を高速化できます。

一部の操作では、一時テーブルを必要としないインプレース ALTER TABLE が可能です。

- ALTER TABLE tbl_name RENAME TO new_tbl_name をほかのオプションなしで実行すると、MySQL は単純に、コピーを作成することなく、テーブル tbl_name に対応するすべてのファイルの名前を変更します。(RENAME TABLE ステートメントを使用してテーブルの名前を変更することもできます。セクション13.1.32「RENAME TABLE 構文」を参照してください。)名前変更されたテーブル専用に付与された権限は、どれも新しい名前には移行されません。それらは、手動で変更する必要があります。

- サーバーが変更する必要があるのはテーブルの内容ではなく、テーブルの .frm ファイルだけであるため、テーブルデータを変更せず、テーブルメタデータだけを変更する変更はただちに実行されます。次の変更は、この方法で実行できる迅速な変更です。

  - カラムの名前変更。ただし、MySQL 5.6.6 より前の InnoDB ストレージエンジンを除きます。

  - カラムのデフォルト値の変更 (NDB テーブルを除きます。「MySQL Cluster オンライン操作の制限」を参照してください)。

  - 有効なメンバー値のリストの最後に新しい列挙またはセットメンバーを追加することによる、ENUM または SET カラムの定義の変更。ただし、データ型のストレージサイズが変更される場合を除きます。たとえば、メンバー数が 8 の SET カラムにメンバーを追加すると、値ごとに必要なストレージが 1 バイトから 2 バイトに変更されます。これには、テーブルコピーが必要になります。リストの途中にメンバーを追加すると、既存のメンバーの番号が変更されます。これには、テーブルコピーが必要になります。

- ADD PARTITION、DROP PARTITION、COALESCE PARTITION、REBUILD PARTITION、または REORGANIZE PARTITION を含む ALTER TABLE は、一時テーブルを作成しません (NDB テーブルで使用される場合を除きます)。ただし、これらの操作では、一時的なパーティションファイルが作成されます。

  RANGE または LIST パーティションに対する ADD または DROP 操作は即座の操作か、ほぼ即座の操作です。HASH または KEY パーティションに対する ADD または COALESCE 操作では、LINEAR HASH または LINEAR KEY が使用されていないかぎり、すべてのパーティション間でデータがコピーされます。ADD または COALESCE 操作はパーティションごとに実行されますが、これは実質的に、新しいテーブルの作成と同じです。REORGANIZE 操作では変更されたパーティションのみがコピーされ、変更されていないものはそのままです。

- インデックスの名前変更。InnoDB を除きます。

- InnoDB と NDB に対するインデックスの追加または削除。

old_alter_table システム変数を ON に設定するか、または alter_specification 句の 1 つとして ALGORITHM=COPY を指定することによって、通常であればテーブルコピーを必要としない ALTER TABLE 操作で強制的に (MySQL 5.0 でサポートされている) 一時テーブルの方法を使用するようにできます。old_alter_table 設定と、DEFAULT 以外の値を持つ ALGORITHM 句の間に矛盾がある場合は、ALGORITHM 句が優先されます。(ALGORITHM = DEFAULT は、ALGORITHM 句をまったく指定しないのと同じです。)

ALGORITHM=INPLACE を指定すると、それをサポートする句やストレージエンジンに対してインプレースの手法が使用されるようになり、サポートされていない場合はエラーで失敗します。そのため、予測していたものとは異なるストレージエンジンを使用するテーブルを変更しようとした場合の非常に長いテーブルコピーが回避されます。InnoDB テーブルに対するオンライン DDL については、セクション14.11「InnoDB とオンライン DDL」を参照してください。

MySQL 5.6.16 では、ALTER TABLE は、ADD COLUMN、CHANGE COLUMN、MODIFY COLUMN、ADD INDEX、および FORCE 操作に関して、古い時間カラムを 5.6 形式にアップグレードします。テーブルを再構築しなければならないため、この変換は INPLACE アルゴリズムを使用して実行することはできません。そのため、これらの場合に ALGORITHM=INPLACE を指定すると エラーになります。必要であれば、ALGORITHM=COPY を指定します。

MySQL 5.6.22 から、テーブルを KEY によってパーティション化するために使用されるマルチカラムインデックスに対する ALTER TABLE 操作は、この操作によってカラムの順序が変更される場合はオンラインで実行できません。このような場合は、代わりに、コピーする ALTER TABLE を使用する必要があります。(Bug #17896265)

MySQL Cluster はオンライン ALTER TABLE 操作もサポートしていますが、ALGORITHM=INPLACE 構文を受け入れないため、代わりに ONLINE キーワードが使用されます。ONLINE および OFFLINE キーワードは、MySQL Cluster でのみサポートされます。これらのキーワードは、MySQL Cluster NDB 7.3 から非推奨です。MySQL Cluster NDB 7.4 では引き続きサポートされますが、将来のバージョンの MySQL Cluster で削除される可能性があります。正確な構文およびその他の詳細については、セクション13.1.7.2「MySQL Cluster での ALTER TABLE オンライン操作」を参照してください。

LOCK 句を使用すると、変更中のテーブルの並列読み取りおよび書き込みのレベルを制御できます。この句にデフォルト以外の値を指定すると、変更操作中に特定の量の並列アクセスまたは排他性が要求されるようにすることが可能であり、要求されたレベルのロックを使用できない場合は操作が停止されます。LOCK 句のパラメータは次のとおりです。

- LOCK = DEFAULT

  指定された ALGORITHM 句 (存在する場合) および ALTER TABLE 操作に対する最大レベルの並列性: サポートされている場合は、並列読み取りおよび書き込みを許可します。そうでない場合、サポートされている場合は、並列読み取りを許可します。そうでない場合は、排他的アクセスを適用します。

- LOCK = NONE

  サポートされている場合は、並列読み取りおよび書き込みを許可します。そうでない場合は、エラーメッセージを返します。

- LOCK = SHARED

  サポートされている場合は、並列読み取りを許可しますが、書き込みはブロックします。書き込みは、指定された ALGORITHM 句 (存在する場合) および ALTER TABLE 操作に対して、並列書き込みがストレージエンジンによってサポートされている場合でもブロックされます。並列読み取りがサポートされていない場合は、エラーメッセージを返します。

- LOCK = EXCLUSIVE

  排他的アクセスを適用します。これは、指定された ALGORITHM 句 (存在する場合) および ALTER TABLE 操作に対して、並列読み取り/書き込みがストレージエンジンによってサポートされている場合でも実行されます。

MySQL 5.6.3 の時点では、ALTER TABLE tbl_name FORCE を使用して、テーブルを再構築する「null」変更操作を実行することもできます。以前は、FORCE オプションは認識されましたが、無視されました。MySQL 5.6.17 の時点では、オンライン DDL のサポートは FORCE オプションに対して提供されます。詳細は、セクション14.11.1「オンライン DDL の概要」を参照してください。

NDB テーブルの場合、可変幅カラム上のインデックスを追加および削除する操作は、そのほとんどの期間、テーブルコピーを行なったり、並列 DML アクションをブロックしたりすることなくオンラインで実行されます。セクション13.1.7.2「MySQL Cluster での ALTER TABLE オンライン操作」を参照してください。

使用上の注意

- ALTER TABLE を使用するには、このテーブルに対する ALTER、CREATE、および INSERT 権限が必要です。テーブルを名前変更するには、古いテーブル側で ALTER および DROP と、新しいテーブル側で ALTER、CREATE、および INSERT 権限が必要です。
- IGNORE は、標準 SQL への MySQL 拡張です。これは、新しいテーブル内の一意のキーに関して重複が存在する場合や、厳密モードが有効になっているときに警告が発生した場合の ALTER TABLE の動作を制御します。IGNORE が指定されていない場合は、重複キーエラーが発生すると、コピーは中止され、ロールバックされます。IGNORE が指定されている場合は、一意のキーが重複している行のうちの 1 行だけが使用されます。その他の競合している行は削除されます。正しくない値は、もっとも近い一致する許容可能な値に切り捨てられます。

  MySQL 5.6.17 の時点では、IGNORE 句は非推奨であり、使用すると警告が生成されます。MySQL 5.7 では、IGNORE は削除されます。
- テーブルが書き込みロックされているときに、そのテーブル構造を変更するために ALTER TABLE が使用されると、保留中の INSERT DELAYED ステートメントは失われます。
- table_option は、ENGINE、AUTO_INCREMENT、AVG_ROW_LENGTH、MAX_ROWS、ROW_FORMAT などの、CREATE TABLE ステートメントで使用できる種類のテーブルオプションを示します。すべてのテーブルオプションのリストと各オプションの説明については、セクション13.1.17「CREATE TABLE 構文」を参照してください。ただし、ALTER TABLE は、DATA DIRECTORY および INDEX DIRECTORY テーブルオプションを無視します。

  たとえば、テーブルを InnoDB テーブルになるように変換するには、次のステートメントを使用します。

  ```
  ALTER TABLE t1 ENGINE = InnoDB;
  ```

  テーブルを InnoDB ストレージエンジンに切り替えるときの考慮事項については、セクション14.6.4「MyISAM から InnoDB へのテーブルの変換」を参照してください。

  ENGINE 句を指定すると、ALTER TABLE はテーブルを再構築します。これは、そのテーブルに指定されたストレージエンジンがすでに存在する場合にも当てはまります。

  既存の InnoDB テーブルに対して ALTER TABLE tbl_name ENGINE=INNODB を実行すると、「null」 ALTER TABLE 操作が実行されます。これは、セクション14.10.4「テーブルのデフラグ」で説明されているように、InnoDB テーブルのデフラグに使用できます。InnoDB テーブルに対して ALTER TABLE tbl_name FORCE を実行しても、同じ機能が実行されます。

  MySQL 5.6.17 の時点では、ALTER TABLE tbl_name ENGINE=INNODB と ALTER TABLE tbl_name FORCE の両方がオンライン DDL (ALGORITHM=COPY) を使用します。詳細は、セクション14.11.1「オンライン DDL の概要」を参照してください。

  テーブルのストレージエンジンを変更しようとすると、セクション5.1.7「サーバー SQL モード」で説明されているように、その結果は目的のストレージエンジンが使用可能かどうかや、NO_ENGINE_SUBSTITUTION SQL モードの設定によって影響を受けます。

  データが誤って失われることのないように、ALTER TABLE を使用して、テーブルのストレージエンジンを MERGE または BLACKHOLE に変更することはできません。

  新しい行に使用される AUTO_INCREMENT カウンタの値を変更するには、次のようにします。

  ```
  ALTER TABLE t2 AUTO_INCREMENT = value;
  ```

  このカウンタを、現在使用されている値以下の値にリセットすることはできません。InnoDB と MyISAM のどちらの場合も、この値が現在 AUTO_INCREMENT カラム内にある最大値以下である場合、この値は現在の AUTO_INCREMENT カラムの最大値に 1 を加えた値にリセットされます。
- 1 つの ALTER TABLE ステートメントで、カンマで区切られた複数の ADD、ALTER、DROP、および CHANGE 句を発行できます。これは、ALTER TABLE ステートメントごとに各句が 1 つしか許可されない標準 SQL への MySQL 拡張です。たとえば、1 つのステートメントで複数のカラムを削除するには、次のようにします。

  ```
  ALTER TABLE t2 DROP COLUMN c, DROP COLUMN d;
  ```
- CHANGE col_name、DROP col_name、および DROP INDEX は、標準 SQL への MySQL 拡張です。
- ワード COLUMN はオプションであり、省略できます。

- column_definition 句は、ADD と CHANGE に対して、CREATE TABLE に対するのと同じ構文を使用します。セクション13.1.17「CREATE TABLE 構文」を参照してください。
- CHANGE old_col_name new_col_name column_definition 句を使用して、カラムの名前を変更できます。それを行うには、古いカラム名と新しいカラム名、およびそのカラムの現在の定義を指定します。たとえば、INTEGER カラムの名前を a から b に変更するには、次のように行うことができます。

```
ALTER TABLE t1 CHANGE a b INTEGER;
```

カラムの名前を変更せずに、その型を変更するには、古いカラム名と新しいカラム名が同じ場合でも、CHANGE 構文にはそれらの名前が引き続き必要です。例:

```
ALTER TABLE t1 CHANGE b b BIGINT NOT NULL;
```

MODIFY を使用しても、カラムの名前を変更せずに、その型を変更できます。

```
ALTER TABLE t1 MODIFY b BIGINT NOT NULL;
```

MODIFY は、Oracle の互換性のための ALTER TABLE への拡張です。

CHANGE または MODIFY を使用する場合は、column_definition に、データ型および PRIMARY KEY や UNIQUE などのインデックス属性以外の、新しいカラムに適用されるすべての属性を含める必要があります。元の定義には存在するが、新しい定義として指定されていない属性は引き継がれません。カラム col1 が INT UNSIGNED DEFAULT 1 COMMENT 'my column' として定義されているときに、このカラムを次のように変更するとします。

```
ALTER TABLE t1 MODIFY col1 BIGINT;
```

結果として得られるカラムは BIGINT として定義されますが、属性 UNSIGNED DEFAULT 1 COMMENT 'my column' は含まれません。これらを保持するには、ステートメントを次のようにしてください。

```
ALTER TABLE t1 MODIFY col1 BIGINT UNSIGNED DEFAULT 1 COMMENT 'my column';
```

- CHANGE または MODIFY を使用してデータ型を変更すると、MySQL は、既存のカラム値を新しい型にできるだけ変換しようとします。

> **警告**
>
> この変換によって、データが変更される可能性があります。たとえば、文字列カラムを短くすると、値が切り捨てられる可能性があります。新しいデータ型への変換によってデータが失われる場合は操作が成功しないようにするには、ALTER TABLE を使用する前に厳密な SQL モードを有効にします (セクション5.1.7「サーバー SQL モード」を参照してください)。

- テーブル行内の特定の位置にカラムを追加するには、FIRST または AFTER col_name を使用します。デフォルトでは、そのカラムを最後に追加します。また、CHANGE または MODIFY 操作で FIRST と AFTER を使用して、テーブル内のカラムを並べ替えることもできます。
- ALTER ... SET DEFAULT または ALTER ... DROP DEFAULT は、それぞれ、カラムの新しいデフォルト値を指定するか、または古いデフォルト値を削除します。古いデフォルトが削除され、かつカラムを NULL にできる場合、新しいデフォルトは NULL です。カラムを NULL にできない場合、MySQL は、セクション11.6「データ型デフォルト値」で説明されているようにデフォルト値を割り当てます。
- DROP INDEX は、インデックスを削除します。これは、標準 SQL への MySQL 拡張です。セクション13.1.24「DROP INDEX 構文」を参照してください。インデックス名に確信がない場合は、SHOW INDEX FROM tbl_name を使用します。
- テーブルからカラムが削除された場合、そのカラムは、それが含まれているすべてのインデックスからも削除されます。インデックスを構成するすべてのカラムが削除された場合は、そのインデックスも削除されます。CHANGE または MODIFY を使用して、インデックスが存在するカラムを短くしたときに、結果として得られるカラムの長さがインデックスの長さより短くなった場合、MySQL は自動的にそのインデックスを短くします。
- テーブルに 1 つのカラムしか含まれていない場合は、そのカラムを削除できません。テーブルを削除することが目的である場合は、代わりに DROP TABLE を使用します。
- DROP PRIMARY KEY は、主キーを削除します。主キーが存在しない場合は、エラーが発生します。主キーのパフォーマンス特性 (特に InnoDB テーブルの場合) については、セクション8.3.2「主キーの使用」を参照してください。

テーブルに UNIQUE INDEX または PRIMARY KEY を追加すると、重複キーをできるだけ早く検出できるようにするために、MySQL はそれを一意でないどのインデックスよりも前に格納します。

- 一部のストレージエンジンでは、インデックスの作成時にインデックスタイプを指定できます。index_type 指定子の構文は、USING type_name です。USING の詳細は、セクション13.1.13「CREATE INDEX 構文」を参照してください。推奨される位置は、カラムリストのあとです。このオプションをカラムリストの前に使用するためのサポートは、将来の MySQL リリースで削除される予定です。

  index_option 値は、インデックスの追加オプションを指定します。USING はそのようなオプションの 1 つです。許可される index_option 値の詳細は、セクション13.1.13「CREATE INDEX 構文」を参照してください。

- ALTER TABLE ステートメントのあとに、インデックスカーディナリティー情報を更新するために ANALYZE TABLE の実行が必要になることがあります。セクション13.7.5.23「SHOW INDEX 構文」を参照してください。

- ORDER BY を使用すると、特定の順序で行が含まれる新しいテーブルを作成できます。このオプションは主に、ほとんどは特定の順序で行をクエリーすることがわかっている場合に役立ちます。このオプションをテーブルの大幅な変更のあとに使用すると、パフォーマンスの向上が得られる可能性があります。場合によっては、テーブルが、あとでその並べ替えに使用するカラムごとの順番になっていれば、MySQL でのソートが簡単になることがあります。

  > **注記**
  >
  > 挿入や削除を行うと、このテーブルは指定された順序のままではなくなります。

  ORDER BY 構文では、ソートのためのカラム名を 1 つ以上指定できます。その各カラム名に続けて、オプションで、それぞれ昇順または降順のソート順序を示す ASC または DESC を指定できます。デフォルトは昇順です。ソート条件として許可されるのはカラム名だけです。任意の式は許可されていません。この句は、ほかのどの句よりもあとの最後に指定するようにしてください。

  InnoDB は常に、クラスタ化されたインデックスに従ってテーブル行を並べ替えるため、ORDER BY は InnoDB テーブルでは意味がありません。

  > **注記**
  >
  > パーティション化されたテーブルに対して使用されている場合、ALTER TABLE ... ORDER BY は、各パーティション内でのみ行を並べ替えます。

- MyISAM テーブルに対して ALTER TABLE を使用した場合、一意でないインデックスはすべて (REPAIR TABLE として) 別のバッチに作成されます。多くのインデックスがあるときは、この方法で ALTER TABLE がはるかに早くなります。

  MyISAM テーブルの場合は、キーの更新を明示的に制御できます。ALTER TABLE ... DISABLE KEYS を使用して、一意でないインデックスの更新を停止するよう MySQL に指示します。次に、ALTER TABLE ... ENABLE KEYS を使用して、不足しているインデックスを再作成します。MyISAM はこれを、キーを 1 つずつ挿入するのに比べてはるかに高速な特殊なアルゴリズムで実行するため、一括挿入操作を実行する前にキーを無効にすると大幅な高速化が得られます。ALTER TABLE ... DISABLE KEYS を使用するには、先に説明した権限に加えて INDEX 権限が必要です。

  一意でないインデックスは、無効になっている間、有効なときにはこのインデックスを使用する SELECT や EXPLAIN などのステートメントで無視されます。

- MySQL 5.6.7 より前は、ALTER TABLE を使用して外部キーカラムの定義を変更すると、参照整合性が失われる可能性がありました。たとえば、NULL 値を含む外部キーカラムを NOT NULL になるように変更すると、NULL 値が空の文字列になりました。同様に、親テーブル内の行を削除する ALTER TABLE IGNORE によって、参照整合性が破壊される可能性がありました。

  5.6.7 の時点では、参照整合性が失われる可能性のある外部キーカラムへの変更がサーバーによって禁止されます。また、安全でない可能性のある、このようなカラムのデータ型への変更も禁止されます。たとえば、VARCHAR(20) の VARCHAR(30) への変更は許可されますが、それを VARCHAR(1024) に変更することは、それによって個々の値を格納するために必要なバイト長の数が変更されるため許可されません。回避方法として、カラム定義を変更する前に ALTER TABLE ... DROP FOREIGN KEY を使用し、あとで ALTER TABLE ... ADD FOREIGN KEY を使用します。

- FOREIGN KEY および REFERENCES 句は、ADD [CONSTRAINT [symbol]] FOREIGN KEY [index_name] (...) REFERENCES ... (...) を実装している InnoDB および NDB ストレージエンジンによってサポートされます。セクション14.6.6「InnoDB と FOREIGN KEY 制約」を参照してください。その他のストレージエンジンでは、

これらの句は解析されますが、無視されます。CHECK 句は、すべてのストレージエンジンによって解析されますが、無視されます。セクション13.1.17「CREATE TABLE 構文」を参照してください。構文の句を受け入れるが、無視する理由は互換性のためです。つまり、ほかの SQL サーバーからコードを移植し、参照によってテーブルを作成するアプリケーションを実行することを容易にするためです。セクション1.7.2「MySQL と標準 SQL との違い」を参照してください。

ALTER TABLE では、CREATE TABLE とは異なり、ADD FOREIGN KEY は index_name (指定されている場合) を無視し、自動的に生成された外部キー名を使用します。回避方法として、外部キー名を指定する CONSTRAINT 句を含めます。

```
ADD CONSTRAINT name FOREIGN KEY (....) ...
```

> **重要**
>
> 参照がカラム指定の一部として定義されているインラインの REFERENCES 指定は、暗黙のうちに無視されます。MySQL は、個別の FOREIGN KEY 指定の一部として定義されている REFERENCES 句のみを受け入れます。

> **注記**
>
> パーティション化された InnoDB テーブルは、外部キーをサポートしていません。この制限は、NDB テーブル ([LINEAR] KEY によって明示的にパーティション化されたテーブルを含む) には適用されません。詳細は、セクション19.6.2「ストレージエンジンに関連するパーティショニング制限」を参照してください。

- InnoDB および NDB ストレージエンジンは、外部キーを削除するための ALTER TABLE の使用をサポートします。

```
ALTER TABLE tbl_name DROP FOREIGN KEY fk_symbol;
```

詳細は、セクション14.6.6「InnoDB と FOREIGN KEY 制約」を参照してください。

- MySQL 5.6.6 より前は、同じ ALTER TABLE ステートメントでの外部キーの追加と削除は、問題が発生する場合があるためサポートされていません。操作ごとに個別のステートメントを使用するようにしてください。MySQL 5.6.6 の時点では、同じ ALTER TABLE ステートメントでの外部キーの追加と削除は ALTER TABLE ... ALGORITHM=INPLACE ではサポートされますが、ALTER TABLE ... ALGORITHM=COPY では未サポートのままです。

- .ibd ファイル内の独自のテーブルスペースで作成された InnoDB テーブルの場合は、そのファイルを破棄したり、インポートしたりできます。.ibd ファイルを破棄するには、次のステートメントを使用します。

```
ALTER TABLE tbl_name DISCARD TABLESPACE;
```

これにより、現在の .ibd ファイルが削除されるため、最初にバックアップがあることを確認してください。テーブルスペースファイルが破棄されている間にテーブルの内容を変更しようとすると、エラーが発生します。テーブルスペースファイルが破棄されている間に、セクション14.11「InnoDB とオンライン DDL」に示されている DDL 操作を実行できます。

バックアップ .ibd ファイルを元のテーブルにインポートするには、それをデータベースディレクトリにコピーしてから、次のステートメントを発行します。

```
ALTER TABLE tbl_name IMPORT TABLESPACE;
```

テーブルスペースファイルは、必ずしも、あとでインポートされるサーバー上に作成されている必要はありません。MySQL 5.6 では、テーブルスペースファイルの別のサーバーからのインポートが機能するのは、両方のサーバーのステータスが GA (General Availablility) であり、かつそれらのバージョンが同じシリーズ内にある場合です。そうでない場合、そのファイルはインポートされるサーバー上に作成されている必要があります。

> **注記**
>
> ALTER TABLE ... IMPORT TABLESPACE 機能は、インポートされたデータに対して外部キー制約を課しません。

セクション14.5.2「InnoDB File-Per-Table モード」を参照してください。

- テーブルのデフォルトの文字セットおよびすべての文字カラム (CHAR、VARCHAR、TEXT) を新しい文字セットに変更するには、次のようなステートメントを使用します。

```
ALTER TABLE tbl_name
```

```
CONVERT TO CHARACTER SET charset_name [COLLATE collation_name];
```

このステートメントでは、すべての文字カラムの照合順序も変更されます。使用する照合順序を示す COLLATE 句を指定しない場合、このステートメントは、その文字セットのデフォルトの照合順序を使用します。この照合順序が目的とするテーブル使用に適していない (たとえば、大文字と小文字が区別される照合順序から大文字と小文字が区別されない照合順序に変更されてしまう) 場合は、照合順序を明示的に指定します。

データ型が VARCHAR か、またはいずれかの TEXT 型であるカラムに対して、CONVERT TO CHARACTER SET は、新しいカラムが確実に元のカラムと同じ数の文字を格納できる十分な長さになるように、必要に応じてデータ型を変更します。たとえば、TEXT カラムには、そのカラム内の値のバイト長 (最大 65,535) を格納するための 2 バイト長があります。latin1 TEXT カラムの場合は、各文字に 1 バイトが必要なため、このカラムには最大 65,535 文字を格納できます。このカラムが utf8 に変換された場合は、各文字に最大 3 バイトが必要になる可能性があるため、可能性のある最大の長さは 3 × 65,535 = 196,605 バイトになります。その長さは TEXT カラムのバイト長には収まらないため、MySQL はそのデータ型を、バイト長に 196,605 の値を記録できる最小の文字列型である MEDIUMTEXT に変換します。同様に、VARCHAR カラムは MEDIUMTEXT に変換される可能性があります。

今説明した型のデータ型の変更を回避するには、CONVERT TO CHARACTER SET を使用しないでください。代わりに、MODIFY を使用して個々のカラムを変更します。例:

```
ALTER TABLE t MODIFY latin1_text_col TEXT CHARACTER SET utf8;
ALTER TABLE t MODIFY latin1_varchar_col VARCHAR(M) CHARACTER SET utf8;
```

CONVERT TO CHARACTER SET binary を指定した場合、CHAR、VARCHAR、および TEXT カラムは、それぞれ対応するバイナリ文字列型 (BINARY、VARBINARY、BLOB) に変換されます。つまり、これらのカラムには文字セットが含まれなくなるため、以降の CONVERT TO 操作は適用されません。

charset_name が DEFAULT である場合は、データベース文字セットが使用されます。

> **警告**
>
> CONVERT TO 操作は、文字セット間でカラム値を変換します。これは、ある文字セット (latin1 など) のカラムがあるが、格納された値が実際には、ほかの何らかの互換性のない文字セット (utf8 など) を使用している場合に必要なものではありません。この場合は、このようなカラムごとに、次を実行する必要があります。
>
> ```
> ALTER TABLE t1 CHANGE c1 c1 BLOB;
> ALTER TABLE t1 CHANGE c1 c1 TEXT CHARACTER SET utf8;
> ```
>
> これが機能する理由は、BLOB カラムとの間で変換する場合は変換が発生しないためです。

テーブルのデフォルトの文字セットのみを変更するには、次のステートメントを使用します。

```
ALTER TABLE tbl_name DEFAULT CHARACTER SET charset_name;
```

ワード DEFAULT はオプションです。デフォルトの文字セットは、あとで (たとえば、ALTER TABLE ... ADD column で) テーブルに追加するカラムの文字セットを指定しない場合に使用される文字セットです。

mysql_info() C API 関数を使用すると、ALTER TABLE によってコピーされた行数、および (IGNORE が使用されている場合は) 一意のキー値の重複のために削除された行数を確認できます。セクション 23.7.7.35「mysql_info()」を参照してください。

13.1.7.1 ALTER TABLE パーティション操作

ALTER TABLE のパーティション化関連の句は、再パーティション化、パーティションの追加、削除、マージ、および分割、パーティション化の保守の実行などのために、パーティション化されたテーブルで使用できます。

- 単に、パーティション化されたテーブルに対して ALTER TABLE で partition_options 句を使用するだけで、partition_options で定義されたパーティション化スキームに従って、そのテーブルが再パーティション化されます。この句は、常に PARTITION BY で始まり、CREATE TABLE の partition_options 句に適用されるのと同じ構文およびその他のルールに従います (詳細は、セクション13.1.17「CREATE TABLE 構文」を参照してください)。また、まだパーティション化されていない既存のテーブルのパーティション化にも使用できます。たとえば、次に示すように定義された (パーティション化されていない) テーブルを考えてみます。

```
CREATE TABLE t1 (
    id INT,
    year_col INT
);
```

このテーブルは、次のステートメントを使用し、id カラムをパーティション化キーとして使用して HASH によって 8 つのパーティションにパーティション化できます。

```
ALTER TABLE t1
    PARTITION BY HASH(id)
    PARTITIONS 8;
```

MySQL 5.6.11 以降では、[SUB]PARTITION BY [LINEAR] KEY で ALGORITHM オプションがサポートされます。ALGORITHM=1 を指定すると、サーバーは、パーティション内の行の配置を計算するときに MySQL 5.1 と同じキーハッシュ関数を使用します。ALGORITHM=2 は、サーバーが、MySQL 5.5 以降で実装され、KEY によってパーティション化された新しいテーブルに対してデフォルトで使用されるキーハッシュ関数を使用することを示します。(MySQL 5.5 以降で採用されたキーハッシュ関数によって作成されたパーティション化されたテーブルを MySQL 5.1 サーバーで使用することはできません。)このオプションを指定しない場合は、ALGORITHM=2 を使用するのと同じ効果があります。このオプションは、主に [LINEAR] KEY によってパーティション化されたテーブルを MySQL 5.1 以降の MySQL バージョン間でアップグレードまたはダウングレードするときに使用するか、または MySQL 5.5 以降のサーバー上で、MySQL 5.1 サーバー上で使用できる KEY または LINEAR KEY によってパーティション化されたテーブルを作成することを目的にしています。

MySQL 5.1 で作成された、KEY によってパーティション化されたテーブルをアップグレードするには、最初に SHOW CREATE TABLE を実行し、表示される正確なカラムおよびパーティションの数をメモします。次に、CREATE TABLE ステートメントとまったく同じカラムリストおよびパーティションの数を使用して ALTER TABLE ステートメントを実行しますが、そのとき PARTITION BY キーワードの直後に ALGORITHM=2 を追加します。(元のテーブル定義に LINEAR キーワードが使用されていた場合は、そのキーワードも含めるようにしてください。)mysql クライアントでのセッションの例を次に示します。

```
mysql> SHOW CREATE TABLE p\G
*************************** 1. row ***************************
       Table: p
Create Table: CREATE TABLE `p` (
  `id` int(11) NOT NULL AUTO_INCREMENT,
  `cd` datetime NOT NULL,
  PRIMARY KEY (`id`)
) ENGINE=InnoDB DEFAULT CHARSET=latin1
/*!50100 PARTITION BY LINEAR KEY (id)
PARTITIONS 32 */
1 row in set (0.00 sec)

mysql> ALTER TABLE p PARTITION BY LINEAR KEY ALGORITHM=2 (id) PARTITIONS 32;
Query OK, 0 rows affected (5.34 sec)
Records: 0  Duplicates: 0  Warnings: 0

mysql> SHOW CREATE TABLE p\G
*************************** 1. row ***************************
       Table: p
Create Table: CREATE TABLE `p` (
  `id` int(11) NOT NULL AUTO_INCREMENT,
  `cd` datetime NOT NULL,
  PRIMARY KEY (`id`)
) ENGINE=InnoDB DEFAULT CHARSET=latin1
/*!50100 PARTITION BY LINEAR KEY (id)
PARTITIONS 32 */
1 row in set (0.00 sec)
```

MySQL 5.5 以降で使用されているデフォルトのキーハッシュを使用して作成されたテーブルを、MySQL 5.1 サーバーで使用できるようにダウングレードする場合も同様です。ただし、この場合は、そのテーブルのパーティションが強制的に MySQL 5.1 のキーハッシュ関数を使用して再構築されるように、ALGORITHM=1 を使用するようにしてください。MySQL 5.5 以降ではデフォルトで使用される改善された KEY ハッシュ関数により、古い実装で見つかった多くの問題に対する修正が提供されるため、MySQL 5.1 サーバーとの互換性のために必要な場合を除き、これは行わないようにすることをお勧めします。

> **注記**
>
> ALTER TABLE ... PARTITION BY ALGORITHM=2 [LINEAR] KEY ... を使用してアップグレードされたテーブルは、MySQL 5.1 サーバーでは使用できなくなります。(このようなテーブルを MySQL 5.1 サーバーでふたたび使用できるようにするには、ALTER TABLE ... PARTITION BY ALGORITHM=1 [LINEAR] KEY ... を使用してダウングレードする必要があります。)

ALTER TABLE ... PARTITION BY ステートメントを使用して作成されたテーブルは、CREATE TABLE ... PARTITION BY を使用して作成されたテーブルと同じルールに従う必要があります。これには、そのテーブル

に含まれている可能性のあるすべての一意のキー (すべての主キーを含む) と、パーティショニング式で使用されている 1 つまたは複数のカラムの間の関係を管理するルールが含まれます。これについては、セクション19.6.1「パーティショニングキー、主キー、および一意キー」で説明されています。また、パーティションの数を指定するための CREATE TABLE ... PARTITION BY のルールも ALTER TABLE ... PARTITION BY に適用されます。

ALTER TABLE ADD PARTITION の partition_definition 句は、CREATE TABLE ステートメントの同じ名前の句と同じオプションをサポートしています。(構文と説明については、セクション13.1.17「CREATE TABLE 構文」を参照してください。)次に示すように作成されたパーティション化されたテーブルがあるとします。

```
CREATE TABLE t1 (
    id INT,
    year_col INT
)
PARTITION BY RANGE (year_col) (
    PARTITION p0 VALUES LESS THAN (1991),
    PARTITION p1 VALUES LESS THAN (1995),
    PARTITION p2 VALUES LESS THAN (1999)
);
```

このテーブルに、2002 より小さい値を格納するための新しいパーティション p3 を次のように追加できます。

```
ALTER TABLE t1 ADD PARTITION (PARTITION p3 VALUES LESS THAN (2002));
```

DROP PARTITION を使用すると、1 つ以上の RANGE または LIST パーティションを削除できます。このステートメントを HASH または KEY パーティションに使用することはできません。代わりに COALESCE PARTITION を使用します (下を参照してください)。partition_names リストで名前が指定されている削除されたパーティションに格納されていたデータはすべて破棄されます。たとえば、前に定義されたテーブル t1 の場合は、p0 および p1 という名前のパーティションを次に示すように削除できます。

```
ALTER TABLE t1 DROP PARTITION p0, p1;
```

> **注記**
>
> DROP PARTITION は、NDB ストレージエンジンを使用するテーブルでは機能しません。セクション19.3.1「RANGE および LIST パーティションの管理」およびセクション18.1.6「MySQL Cluster の既知の制限」を参照してください。

ADD PARTITION と DROP PARTITION は現在、IF [NOT] EXISTS をサポートしていません。

パーティション化されたテーブルの名前変更がサポートされています。ALTER TABLE ... REORGANIZE PARTITION を使用して、間接的に個々のパーティションの名前を変更できます。ただし、この操作によってパーティションのデータのコピーが作成されます。

MySQL 5.6 では、TRUNCATE PARTITION オプションを使用して、選択したパーティションの行を削除できます。このオプションは、1 つ以上のパーティション名のカンマ区切りリストを受け取ります。たとえば、次のように定義されたテーブル t1 を考えてみます。

```
CREATE TABLE t1 (
    id INT,
    year_col INT
)
PARTITION BY RANGE (year_col) (
    PARTITION p0 VALUES LESS THAN (1991),
    PARTITION p1 VALUES LESS THAN (1995),
    PARTITION p2 VALUES LESS THAN (1999),
    PARTITION p3 VALUES LESS THAN (2003),
    PARTITION p4 VALUES LESS THAN (2007)
);
```

パーティション p0 のすべての行を削除するには、次のステートメントを使用できます。

```
ALTER TABLE t1 TRUNCATE PARTITION p0;
```

今示したステートメントには、次の DELETE ステートメントと同じ効果があります。

```
DELETE FROM t1 WHERE year_col < 1991;
```

複数のパーティションを切り詰める場合、パーティションが連続している必要はありません。これにより、通常、DELETE ステートメントで実行された場合は非常に複雑な WHERE 条件が必要になる、パーティション

化されたテーブルでの削除操作が大幅に簡素化される可能性があります。たとえば、次のステートメントは、パーティション p1 と p3 のすべての行を削除します。

```
ALTER TABLE t1 TRUNCATE PARTITION p1, p3;
```

同等の DELETE ステートメントを次に示します。

```
DELETE FROM t1 WHERE
   (year_col >= 1991 AND year_col < 1995)
   OR
   (year_col >= 2003 AND year_col < 2007);
```

パーティション名のリストの代わりに ALL キーワードを使用することもできます。この場合、このステートメントはテーブル内のすべてのパーティションに対して機能します。

TRUNCATE PARTITION は行を削除するだけです。そのテーブル自体や、どのパーティションの定義も変更されません。

> **注記**
>
> TRUNCATE PARTITION は、サブパーティションでは機能しません。

次のようなクエリーを使用して INFORMATION_SCHEMA.PARTITIONS テーブルをチェックすることにより、行が削除されたことを確認できます。

```
SELECT PARTITION_NAME, TABLE_ROWS
   FROM INFORMATION_SCHEMA.PARTITIONS
   WHERE TABLE_NAME = 't1';
```

TRUNCATE PARTITION は、MyISAM、InnoDB、または MEMORY ストレージエンジンを使用するパーティション化されたテーブルでのみサポートされます。また、BLACKHOLE テーブルに対しても機能します (ただし、何の効果もありません)。ARCHIVE テーブルではサポートされません。

HASH または KEY によってパーティション化されたテーブルで COALESCE PARTITION を使用すると、そのパーティションの数を number だけ減らすことができます。次の定義を使用して、テーブル t2 を作成したとします。

```
CREATE TABLE t2 (
   name VARCHAR (30),
   started DATE
)
PARTITION BY HASH( YEAR(started) )
PARTITIONS 6;
```

t2 によって使用されるパーティションの数を、次のステートメントを使用して 6 から 4 に減らすことができます。

```
ALTER TABLE t2 COALESCE PARTITION 2;
```

最後の number 個のパーティションに含まれているデータは、残りのパーティションにマージされます。この場合は、パーティション 4 と 5 が最初の 4 つのパーティション (0、1、2、および 3 の番号を持つ各パーティション) にマージされます。

パーティション化されたテーブルで使用される (すべてではなく) 一部のパーティションを変更するには、REORGANIZE PARTITION を使用できます。このステートメントは、次のいくつかの方法で使用できます。

- 一連のパーティションを単一パーティションにマージします。これは、partition_names リストにいくつかのパーティションの名前を指定し、partition_definition に 1 つの定義を指定することによって実行できます。
- 既存のパーティションをいくつかのパーティションに分割します。これは、partition_names に単一パーティションを指定し、複数の partition_definitions を指定することによって実行できます。
- VALUES LESS THAN を使用して、定義されたパーティションのサブセットの範囲を変更するか、または VALUES IN を使用して、定義されたパーティションのサブセットの値リストを変更します。
- このステートメントはまた、HASH パーティション化を使用して自動的にパーティション化されるテーブルに対して partition_names INTO (partition_definitions) オプションなしで使用すると、データを強制的に再配布できます。(現在、このように自動的にパーティション化されるのは NDB テーブルだけです。)これは、既存の MySQL Cluster に新しい MySQL Cluster データノードをオンラインで追加したあと、既存の MySQL

Cluster テーブルデータを新しいデータノードに再配布する必要がある MySQL Cluster で役立ちます。このような場合は、ONLINE オプションを使用してこのステートメントを呼び出すようにしてください。つまり、次に示すようにします。

```
ALTER ONLINE TABLE table REORGANIZE PARTITION;
```

オンラインのテーブル再編成と同時にほかの DDL を実行することはできません。つまり、ALTER ONLINE TABLE ... REORGANIZE PARTITION ステートメントの実行中は、ほかの DDL ステートメントを発行できません。MySQL Cluster データノードをオンラインで追加する方法の詳細は、セクション18.5.13「MySQL Cluster データノードのオンライン追加」を参照してください。

ALTER ONLINE TABLE ... REORGANIZE PARTITION は、MAX_ROWS オプションを使用して作成されたテーブルでは機能しません。このステートメントは、元の CREATE TABLE ステートメントで指定された一定の MAX_ROWS 値を使用して必要なパーティションの数を決定するため、新しいパーティションが作成されないためです。テーブルの行の最大数を増やすには、ALTER ONLINE TABLE ... MAX_ROWS=rows を使用できます。このあと、ALTER ONLINE TABLE ... REORGANIZE PARTITION は、この大きくなった新しい値を使用してパーティションの数を増やすことができます。これが機能するには、rows の値を、元の CREATE TABLE ステートメントで MAX_ROWS に指定された値より大きくする必要があります。

明示的にパーティション化されたテーブルに対して、REORGANIZE PARTITION を partition_names INTO (partition_definitions) オプションなしで使用しようとすると、REORGANIZE PARTITION without parameters can only be used on auto-partitioned tables using HASH partitioningエラーが発生します。

> **注記**
>
> 明示的に名前が付けられていないパーティションに対して、MySQL は自動的に p0、p1、p2 などのデフォルト名を付けます。同じことがサブパーティションにも当てはまります。

ALTER TABLE ... REORGANIZE PARTITION ステートメントの詳細および例については、セクション19.3.1「RANGE および LIST パーティションの管理」を参照してください。

- MySQL 5.6 では、ALTER TABLE ... EXCHANGE PARTITION ステートメントを使用して、テーブルパーティションまたはサブパーティションを別のテーブルと交換できます。つまり、パーティションまたはサブパーティション内の既存の任意の行をパーティション化されていないテーブルに移動したり、パーティション化されていないテーブル内の既存の任意の行をテーブルパーティションまたはサブパーティションに移動したりできます。

  使用方法および例については、セクション19.3.3「パーティションとサブパーティションをテーブルと交換する」を参照してください。

- いくつかの追加オプションによって、パーティション化されていないテーブルのために CHECK TABLE や REPAIR TABLE などのステートメントによって実装されている機能 (これは、パーティション化されたテーブルでもサポートされます。詳細は、セクション13.7.2「テーブル保守ステートメント」を参照してください) に類似したパーティションの保守および修復機能が提供されます。これには、ANALYZE PARTITION、CHECK PARTITION、OPTIMIZE PARTITION、REBUILD PARTITION、および REPAIR PARTITION が含まれます。これらの各オプションは、1 つ以上のパーティション名から成るカンマで区切られた partition_names 句を受け取ります。これらのパーティションは、変更されるテーブル内にすでに存在する必要があります。partition_names の代わりに ALL キーワードを使用することもできます。その場合、このステートメントはテーブル内のすべてのパーティションに対して機能します。詳細および例については、セクション19.3.4「パーティションの保守」を参照してください。

  InnoDB などの一部の MySQL ストレージエンジンは、パーティションごとの最適化をサポートしていません。このようなストレージエンジンを使用したパーティション化されたテーブルに対して、ALTER TABLE ... OPTIMIZE PARTITION はテーブル全体を再構築します。これは既知の問題です。MySQL 5.6.9 から、このようなテーブルに対してこのステートメントを実行するとテーブル全体が再構築および分析され、該当する警告が発行されます。(Bug #11751825、Bug #42822)

  この問題を回避するには、代わりに、ステートメント ALTER TABLE ... REBUILD PARTITION および ALTER TABLE ... ANALYZE PARTITION を使用します。

  ANALYZE PARTITION、CHECK PARTITION、OPTIMIZE PARTITION、および REPAIR PARTITION オプションは、パーティション化されていないテーブルには許可されません。

- REMOVE PARTITIONING を使用すると、テーブルのパーティション化を削除でき、そのテーブルやデータはそれ以外の影響を受けません。このオプションは、カラムやインデックスの追加、削除、名前変更などのために使用されるその他の ALTER TABLE オプションと組み合わせることができます。

- ALTER TABLE で ENGINE オプションを使用すると、パーティション化に影響を与えることなく、テーブルで使用されるストレージエンジンが変更されます。

MySQL 5.6.6 より前は、MyISAM (または、テーブルレベルのロックを使用する別のストレージエンジン) を使用するパーティション化されたテーブルに対して ALTER TABLE ... EXCHANGE PARTITION または ALTER TABLE ... TRUNCATE PARTITION が実行されると、そのパーティション化されたテーブル全体がロックされました。MySQL 5.6.6 以降では、このような場合、実際に読み取られたパーティションのみがロックされます。これは、行レベルロックを採用している (InnoDB などの) ストレージエンジンを使用するパーティション化されたテーブルには影響しませんでした (現在も影響しません)。セクション19.6.4「パーティショニングとロック」を参照してください。

ALTER TABLE ステートメントには、ほかの変更指定に加えて、PARTITION BY または REMOVE PARTITIONING 句を含めることができますが、PARTITION BY または REMOVE PARTITIONING 句は、ほかのどの指定よりもあとの最後に指定する必要があります。

ADD PARTITION、DROP PARTITION、COALESCE PARTITION、REORGANIZE PARTITION、ANALYZE PARTITION、CHECK PARTITION、および REPAIR PARTITION オプションは、個々のパーティションに対して機能するため、1 つの ALTER TABLE 内でほかの変更指定と組み合わせることはできません。詳細は、セクション13.1.7.1「ALTER TABLE パーティション操作」を参照してください。

特定の ALTER TABLE ステートメントでは、次のいずれか 1 つのオプションの単一インスタンスのみを使用できます。PARTITION BY、ADD PARTITION、DROP PARTITION、TRUNCATE PARTITION、EXCHANGE PARTITION、REORGANIZE PARTITION、または COALESCE PARTITION、ANALYZE PARTITION、CHECK PARTITION、OPTIMIZE PARTITION、REBUILD PARTITION、REMOVE PARTITIONING。

たとえば、次の 2 つのステートメントは無効です。

```
ALTER TABLE t1 ANALYZE PARTITION p1, ANALYZE PARTITION p2;

ALTER TABLE t1 ANALYZE PARTITION p1, CHECK PARTITION p2;
```

最初のケースでは、次のように、分析される両方のパーティションを一覧表示した 1 つの ANALYZE PARTITION オプションを含む 1 つのステートメントを使用して、テーブル t1 のパーティション p1 と p2 を同時に分析できます。

```
ALTER TABLE t1 ANALYZE PARTITION p1, p2;
```

2 番目のケースでは、同じテーブルの別のパーティションに対する ANALYZE 操作と CHECK 操作を同時に実行することはできません。代わりに、次のように、2 つの個別のステートメントを発行する必要があります。

```
ALTER TABLE t1 ANALYZE PARTITION p1;
ALTER TABLE t1 CHECK PARTITION p2;
```

ANALYZE、CHECK、OPTIMIZE、REBUILD、REPAIR、および TRUNCATE 操作は、サブパーティションではサポートされません。

13.1.7.2 MySQL Cluster での ALTER TABLE オンライン操作

このセクションでは、MySQL Cluster で実装されたオンラインのテーブルスキーマ変更について説明します。InnoDB ストレージエンジンもまた、MySQL Cluster によってサポートされるものとは異なる構文を使用して、このような操作をオンラインで実行できます。詳細は、セクション14.11「InnoDB とオンライン DDL」を参照してください。

NDB テーブルの可変幅カラム上のインデックスを追加および削除する操作はオンラインで実行されます。オンライン操作はコピーなしです。つまり、インデックスを再作成する必要はありません。変更されるテーブルが、MySQL Cluster 内のほかの API ノードによるアクセスからロックされることはありません (ただし、このセクションのあとの方にある「制限」を参照してください)。このような操作では、複数の API ノードを含むクラスタで行われる NDB テーブルの変更にシングルユーザーモードは必要ありません。オンライン DDL 操作中も、トランザクションは中断なく続行できます。

ONLINE キーワードを使用すると、NDB テーブルに対してオンライン ADD COLUMN、ADD INDEX (CREATE INDEX ステートメントを含む)、および DROP INDEX 操作を実行できます。また、NDB テーブルのオンラインでの名前変更もサポートされます。

> **注記**
>
> ONLINE および OFFLINE キーワードは、MySQL Cluster でのみサポートされます。標準の MySQL Server 5.6 リリースでは、ONLINE または OFFLINE キーワードを ALTER TABLE、CREATE INDEX、または DROP INDEX ステートメントで使用しようとすると、エラーが発生します。

> ONLINE および OFFLINE キーワードは、MySQL Cluster NDB 7.3 から非推奨です。MySQL Cluster NDB 7.4 では引き続きサポートされますが、将来のバージョンの MySQL Cluster で削除される可能性があります。

現在、ディスクベースのカラムを NDB テーブルにオンラインで追加することはできません。つまり、テーブルレベルの STORAGE DISK オプションを使用する NDB テーブルにインメモリーカラムを追加する場合は、新しいカラムをメモリーベースのストレージの使用として明示的に宣言する必要があります。たとえば、すでにテーブルスペース ts1 を作成していると仮定して、テーブル t1 を次のように作成するとします。

```
mysql> CREATE TABLE t1 (
    >     c1 INT NOT NULL PRIMARY KEY,
    >     c2 VARCHAR(30)
    >     )
    >     TABLESPACE ts1 STORAGE DISK
    >     ENGINE NDB;
Query OK, 0 rows affected (1.73 sec)
Records: 0  Duplicates: 0  Warnings: 0
```

次に示すように、新しいインメモリーカラムをこのテーブルにオンラインで追加できます。

```
mysql> ALTER ONLINE TABLE t1 ADD COLUMN c3 INT COLUMN_FORMAT DYNAMIC STORAGE MEMORY;
Query OK, 0 rows affected (1.25 sec)
Records: 0  Duplicates: 0  Warnings: 0
```

STORAGE MEMORY オプションが省略されている場合、このステートメントは失敗します。

```
mysql> ALTER ONLINE TABLE t1 ADD COLUMN c3 INT COLUMN_FORMAT DYNAMIC;
ERROR 1235 (42000): This version of MySQL doesn't yet support
'ALTER ONLINE TABLE t1 ADD COLUMN c3 INT COLUMN_FORMAT DYNAMIC'
```

COLUMN_FORMAT DYNAMIC オプションを省略した場合は、動的なカラムフォーマットが自動的に使用されますが、次に示すような警告が発行されます。

```
mysql> ALTER ONLINE TABLE t1 ADD COLUMN c3 INT STORAGE MEMORY;
Query OK, 0 rows affected, 1 warning (1.17 sec)
Records: 0  Duplicates: 0  Warnings: 0

mysql> SHOW WARNINGS;
+---------+------+-----------------------------------------------------------------+
| Level   | Code | Message                                                         |
+---------+------+-----------------------------------------------------------------+
| Warning | 1478 | Converted FIXED field to DYNAMIC to enable on-line ADD COLUMN |
+---------+------+-----------------------------------------------------------------+
1 row in set (0.00 sec)

mysql> SHOW CREATE TABLE t1\G
*************************** 1. row ***************************
       Table: t1
Create Table: CREATE TABLE `t1` (
  `c1` int(11) NOT NULL,
  `c2` varchar(30) DEFAULT NULL,
  `c3` int(11) /*!50120 STORAGE MEMORY */ /*!50120 COLUMN_FORMAT DYNAMIC */ DEFAULT NULL,
  `t4` int(11) /*!50120 STORAGE MEMORY */ DEFAULT NULL,
  PRIMARY KEY (`c1`)
) /*!50100 TABLESPACE ts_1 STORAGE DISK */ ENGINE=ndbcluster DEFAULT CHARSET=latin1
1 row in set (0.03 sec)
```

> 注記
>
> STORAGE および COLUMN_FORMAT キーワードは、MySQL Cluster でのみサポートされます。ほかのどのバージョンの MySQL でも、このどちらかのキーワードを CREATE TABLE または ALTER TABLE ステートメントで使用しようとすると、エラーが発生します。

NDB テーブルに対して、ステートメント ALTER ONLINE TABLE ... REORGANIZE PARTITION を partition_names INTO (partition_definitions) オプションなしで使用することもできます。これを使用すると、オンラインでクラスタに追加された新しいデータノードの間で MySQL Cluster データを再配布できます。このステートメントの詳細は、セクション13.1.7.1「ALTER TABLE パーティション操作」を参照してください。MySQL Cluster にデータノードをオンラインで追加する方法の詳細は、セクション18.5.13「MySQL Cluster データノードのオンライン追加」を参照してください。

MySQL Cluster オンライン操作の制限

オンライン DROP COLUMN 操作はサポートされていません。

カラムを追加するか、あるいはインデックスを追加または削除するオンライン ALTER TABLE、CREATE INDEX、または DROP INDEX ステートメントは、次の制限に従います。

- 特定のオンライン ALTER TABLE では、ADD COLUMN、ADD INDEX、DROP INDEX のいずれか 1 つのみを使用できます。1 つのステートメントで、1 つ以上のカラムをオンラインで追加できます。1 つのステートメントで、1 つのインデックスのみをオンラインで作成または削除できます。
- 変更されるテーブルは、オンライン ALTER TABLE ADD COLUMN、ADD INDEX、または DROP INDEX 操作 (あるいは CREATE INDEX または DROP INDEX ステートメント) が実行されている API ノード以外の API ノードに対してロックされません。ただし、オンライン操作が実行されている間、このテーブルは同じ API ノードから発信されているほかのすべての操作に対してロックされます。
- 変更されるテーブルには明示的な主キーが存在する必要があります。NDB ストレージエンジンによって作成された非表示の主キーは、この目的には不十分です。
- テーブルで使用されるストレージエンジンをオンラインで変更することはできません。
- MySQL Cluster ディスクデータテーブルで使用された場合、カラムのストレージ型 (DISK または MEMORY) をオンラインで変更することはできません。つまり、操作がオンラインで実行されるような方法でインデックスを追加または削除するときに、1 つまたは複数のカラムのストレージ型が変更されるようにする場合は、インデックスを追加または削除するステートメントで OFFLINE キーワードを使用する必要があります。

オンラインで追加されるカラムは BLOB または TEXT 型を使用できず、次の条件を満たす必要があります。

- このカラムは動的である必要があります。つまり、COLUMN_FORMAT DYNAMIC を使用して作成できる必要があります。COLUMN_FORMAT DYNAMIC オプションを省略した場合は、動的なカラムフォーマットが自動的に使用されます。
- このカラムは NULL 値を許可する必要があり、NULL 以外の明示的なデフォルト値があってはなりません。オンラインで追加されるカラムは、次に示すように、DEFAULT NULL として自動的に作成されます。

```
mysql> CREATE TABLE t1 (
    >     c1 INT NOT NULL AUTO_INCREMENT PRIMARY KEY
    >     ) ENGINE=NDB;
Query OK, 0 rows affected (1.44 sec)

mysql> ALTER ONLINE TABLE t1
    >     ADD COLUMN c2 INT,
    >     ADD COLUMN c3 INT;
Query OK, 0 rows affected, 2 warnings (0.93 sec)

mysql> SHOW CREATE TABLE t1\G
*************************** 1. row ***************************
       Table: t1
Create Table: CREATE TABLE `t1` (
  `c1` int(11) NOT NULL AUTO_INCREMENT,
  `c2` int(11) DEFAULT NULL,
  `c3` int(11) DEFAULT NULL,
  PRIMARY KEY (`c1`)
) ENGINE=ndbcluster DEFAULT CHARSET=latin1
1 row in set (0.00 sec)
```

- このカラムは、既存のすべてのカラムのあとに追加する必要があります。既存のいずれかのカラムの前に、または FIRST キーワードを使用してオンラインでカラムを追加しようとすると、このステートメントはエラーで失敗します。
- 既存のテーブルカラムをオンラインで並べ替えることはできません。

前の制限は、テーブルまたはカラムの名前を変更するだけの操作には適用されません。

NDB テーブルに対するオンライン ALTER TABLE 操作の場合、固定フォーマットのカラムは次に示すように、オンラインで追加されるか、またはインデックスがオンラインで作成または削除されたときに動的なカラムに変換されます。

```
mysql> CREATE TABLE t1 (
    >     c1 INT NOT NULL AUTO_INCREMENT PRIMARY KEY
    >     ) ENGINE=NDB;
Query OK, 0 rows affected (1.44 sec)

mysql> ALTER ONLINE TABLE t1 ADD COLUMN c2 INT, ADD COLUMN c3 INT;
Query OK, 0 rows affected, 2 warnings (0.93 sec)
Records: 0  Duplicates: 0  Warnings: 0

mysql> SHOW WARNINGS;
```

```
+---------+------+-----------------------------------------------------------------+
| Level   | Code | Message                                                         |
+---------+------+-----------------------------------------------------------------+
| Warning | 1475 | Converted FIXED field to DYNAMIC to enable on-line ADD COLUMN |
| Warning | 1475 | Converted FIXED field to DYNAMIC to enable on-line ADD COLUMN |
+---------+------+-----------------------------------------------------------------+
2 rows in set (0.00 sec)
```

> **注記**
>
> 既存のカラム (テーブルの主キーを含む) は、動的である必要はありません。動的である必要があるのは、オンラインで追加される 1 つまたは複数のカラムだけです。

```
mysql> CREATE TABLE t2 (
    >     c1 INT NOT NULL AUTO_INCREMENT PRIMARY KEY COLUMN_FORMAT FIXED
    >     ) ENGINE=NDB;
Query OK, 0 rows affected (2.10 sec)

mysql> ALTER ONLINE TABLE t2 ADD COLUMN c2 INT;
Query OK, 0 rows affected, 1 warning (0.78 sec)
Records: 0  Duplicates: 0  Warnings: 0

mysql> SHOW WARNINGS;
+---------+------+-----------------------------------------------------------------+
| Level   | Code | Message                                                         |
+---------+------+-----------------------------------------------------------------+
| Warning | 1475 | Converted FIXED field to DYNAMIC to enable on-line ADD COLUMN |
+---------+------+-----------------------------------------------------------------+
1 row in set (0.00 sec)
```

名前の変更操作によって、カラムが FIXED から DYNAMIC のカラムフォーマットに変換されることはありません。COLUMN_FORMAT の詳細は、セクション13.1.17「CREATE TABLE 構文」を参照してください。

KEY、CONSTRAINT、および IGNORE キーワードは、ONLINE キーワードを使用している ALTER TABLE ステートメントでサポートされます。

13.1.7.3 ALTER TABLE の例

次に示すように作成されているテーブル t1 から始めます。

```
CREATE TABLE t1 (a INTEGER,b CHAR(10));
```

テーブルの名前を t1 から t2 に変更するには、次のようにします。

```
ALTER TABLE t1 RENAME t2;
```

カラム a を INTEGER から TINYINT NOT NULL に変更し (名前は同じままにします)、またカラム b を CHAR(10) から CHAR(20) に変更し、さらにその名前を b から c に変更するには、次のようにします。

```
ALTER TABLE t2 MODIFY a TINYINT NOT NULL, CHANGE b c CHAR(20);
```

d という名前の新しい TIMESTAMP カラムを追加するには、次のようにします。

```
ALTER TABLE t2 ADD d TIMESTAMP;
```

カラム d にインデックスを、またカラム a に UNIQUE インデックスを追加するには、次のようにします。

```
ALTER TABLE t2 ADD INDEX (d), ADD UNIQUE (a);
```

カラム c を削除するには、次のようにします。

```
ALTER TABLE t2 DROP COLUMN c;
```

c という名前の新しい AUTO_INCREMENT 整数カラムを追加するには、次のようにします。

```
ALTER TABLE t2 ADD c INT UNSIGNED NOT NULL AUTO_INCREMENT,
  ADD PRIMARY KEY (c);
```

AUTO_INCREMENT カラムにはインデックスを設定する必要があるため c に (PRIMARY KEY として) インデックスを設定し、また主キーカラムは NULL にできないため c を NOT NULL として宣言します。

NDB テーブルの場合は、テーブルまたはカラムに使用されるストレージ型を変更することもできます。たとえば、次に示すように作成された NDB テーブルを考えてみます。

```
mysql> CREATE TABLE t1 (c1 INT) TABLESPACE ts_1 ENGINE NDB;
Query OK, 0 rows affected (1.27 sec)
```

このテーブルをディスクベースのストレージに変換するには、次の ALTER TABLE ステートメントを使用できます。

```
mysql> ALTER TABLE t1 TABLESPACE ts_1 STORAGE DISK;
Query OK, 0 rows affected (2.99 sec)
Records: 0  Duplicates: 0  Warnings: 0

mysql> SHOW CREATE TABLE t1\G
*************************** 1. row ***************************
       Table: t1
Create Table: CREATE TABLE `t1` (
  `c1` int(11) DEFAULT NULL
) /*!50100 TABLESPACE ts_1 STORAGE DISK */
ENGINE=ndbcluster DEFAULT CHARSET=latin1
1 row in set (0.01 sec)
```

テーブルが最初に作成されたときにテーブルスペースが参照されている必要はありませんが、テーブルスペースは ALTER TABLE によって参照される必要があります。

```
mysql> CREATE TABLE t2 (c1 INT) ts_1 ENGINE NDB;
Query OK, 0 rows affected (1.00 sec)

mysql> ALTER TABLE t2 STORAGE DISK;
ERROR 1005 (HY000): Can't create table 'c.#sql-1750_3' (errno: 140)
mysql> ALTER TABLE t2 TABLESPACE ts_1 STORAGE DISK;
Query OK, 0 rows affected (3.42 sec)
Records: 0  Duplicates: 0  Warnings: 0
mysql> SHOW CREATE TABLE t2\G
*************************** 1. row ***************************
       Table: t1
Create Table: CREATE TABLE `t2` (
  `c1` int(11) DEFAULT NULL
) /*!50100 TABLESPACE ts_1 STORAGE DISK */
ENGINE=ndbcluster DEFAULT CHARSET=latin1
1 row in set (0.01 sec)
```

個々のカラムのストレージ型を変更するには、ALTER TABLE ... MODIFY [COLUMN] を使用できます。たとえば、次の CREATE TABLE ステートメントを使用して、2 つのカラムを含む MySQL Cluster ディスクデータテーブルを作成するとします。

```
mysql> CREATE TABLE t3 (c1 INT, c2 INT)
    ->     TABLESPACE ts_1 STORAGE DISK ENGINE NDB;
Query OK, 0 rows affected (1.34 sec)
```

カラム c2 をディスクベースのストレージからインメモリーストレージに変更するには、次に示すように、ALTER TABLE ステートメントで使用されるカラム定義に STORAGE MEMORY 句を含めます。

```
mysql> ALTER TABLE t3 MODIFY c2 INT STORAGE MEMORY;
Query OK, 0 rows affected (3.14 sec)
Records: 0  Duplicates: 0  Warnings: 0
```

同様の方法で STORAGE DISK を使用して、インメモリーカラムをディスクベースのカラムにすることができます。

カラム c1 は、ディスクベースのストレージを使用します。これが (CREATE TABLE ステートメント内のテーブルレベルの STORAGE DISK 句によって決定される) テーブルのデフォルトであるためです。ただし、次の SHOW CREATE TABLE の出力に示すように、カラム c2 はインメモリーストレージを使用します。

```
mysql> SHOW CREATE TABLE t3\G
*************************** 1. row ***************************
       Table: t3
Create Table: CREATE TABLE `t3` (
  `c1` int(11) DEFAULT NULL,
  `c2` int(11) /*!50120 STORAGE MEMORY */ DEFAULT NULL
) /*!50100 TABLESPACE ts_1 STORAGE DISK */ ENGINE=ndbcluster DEFAULT CHARSET=latin1
1 row in set (0.02 sec)
```

AUTO_INCREMENT カラムを追加すると、カラム値には、自動的にシーケンス番号が入力されます。MyISAM テーブルの場合は、ALTER TABLE の前に SET INSERT_ID=value を実行するか、または AUTO_INCREMENT=value テーブルオプションを使用することによって、最初のシーケンス番号を設定できます。セクション5.1.4「サーバーシステム変数」を参照してください。

MyISAM テーブルでは、AUTO_INCREMENT カラムを変更しない場合、シーケンス番号は影響を受けません。AUTO_INCREMENT カラムを削除してから、別の AUTO_INCREMENT カラムを追加した場合、シーケンス番号は 1 から付け直されます。

レプリケーションが使用されている場合は、テーブルに AUTO_INCREMENT カラムを追加しても、スレーブとマスターで行の順序が同じにならない可能性があります。これが発生するのは、行が番号付けされる順序が、テーブルに使用される固有のストレージエンジンおよび行が挿入された順序に依存するためです。マスターとスレーブで同じ順序を持つことが重要である場合は、行を並べ替えてから AUTO_INCREMENT 番号を割り当てる必要があります。テーブル t1 に AUTO_INCREMENT カラムを追加すると仮定した場合、次のステートメントは、t1 と同一であるが、AUTO_INCREMENT カラムを含む新しいテーブル t2 を生成します。

```
CREATE TABLE t2 (id INT AUTO_INCREMENT PRIMARY KEY)
SELECT * FROM t1 ORDER BY col1, col2;
```

ここでは、テーブル t1 にカラム col1 と col2 が存在することを前提にしています。

この一連のステートメントではまた、t1 と同一であるが、AUTO_INCREMENT カラムが追加された新しいテーブル t2 も生成されます。

```
CREATE TABLE t2 LIKE t1;
ALTER TABLE t2 ADD id INT AUTO_INCREMENT PRIMARY KEY;
INSERT INTO t2 SELECT * FROM t1 ORDER BY col1, col2;
```

> **重要**
>
> マスターとスレーブの両方で順序が同じになることを保証するには、ORDER BY 句で t1 のすべてのカラムを参照する必要があります。

AUTO_INCREMENT カラムを持つコピーを作成および移入するために使用する方法にかかわらず、最終手順は元のテーブルを削除してコピーの名前を変更することです。

```
DROP TABLE t1;
ALTER TABLE t2 RENAME t1;
```

13.1.8 ALTER TABLESPACE 構文

```
ALTER TABLESPACE tablespace_name
    {ADD|DROP} DATAFILE 'file_name'
    [INITIAL_SIZE [=] size]
    [WAIT]
    ENGINE [=] engine_name
```

このステートメントを使用すると、テーブルスペースへの新しいデータファイルの追加、またはテーブルスペースからのデータファイルの削除を実行できます。

ADD DATAFILE バリアントでは、INITIAL_SIZE 句を使用して初期サイズを指定できます。ここで、size はバイト単位で測定されます。デフォルト値は 134217728 (128M バイト) です。MySQL Cluster NDB 7.3.2 より前は、この値は数字で指定する必要がありました。(Bug #13116514、Bug #16104705、Bug #62858)。MySQL Cluster NDB 7.3.2 以降では、size のあとにオプションで、my.cnf で使用されるのと同様の、オーダーを示す 1 文字の略語を指定できます。一般に、これは M (M バイト) または G (G バイト) のどちらかの文字です。

> **注記**
>
> すべての MySQL Cluster ディスクデータオブジェクトが同じ名前空間を共有します。つまり、各ディスクデータオブジェクトは (単に、特定の型の各ディスクデータオブジェクトというだけでなく)、一意の名前が付けられている必要があります。たとえば、テーブルスペースとデータファイルを同じ名前にしたり、Undo ログファイルとテーブルスペースを同じ名前にしたりすることはできません。

32 ビットシステム上では、INITIAL_SIZE のサポートされる最大値は 4294967296 (4G バイト) です。(Bug #29186)

INITIAL_SIZE は、CREATE TABLESPACE と同様に明示的に丸められます。

データファイルが作成されたあと、そのサイズを変更することはできません。ただし、追加の ALTER TABLESPACE ... ADD DATAFILE ステートメントを使用して、テーブルスペースにさらに多くのデータファイルを追加できます。

DROP DATAFILE を ALTER TABLESPACE とともに使用すると、テーブルスペースからデータファイル 'file_name' が削除されます。いずれかのテーブルが使用しているテーブルスペースからはデータファイルを削除できません。つまり、そのデータファイルが空である (エクステントが使用されていない) ことが必要です。セクション18.5.12.1「MySQL Cluster ディスクデータオブジェクト」を参照してください。さらに、削除されるデータファイルはすべて、CREATE TABLESPACE または ALTER TABLESPACE で以前にそのテーブルスペースに追加されている必要があります。

ALTER TABLESPACE ... ADD DATAFILE と ALTER TABLESPACE ... DROP DATAFILE のどちらにも、そのテーブルスペースによって使用されるストレージエンジンを指定する ENGINE 句が必要です。現在、engine_name として受け入れられる値は NDB と NDBCLUSTER だけです。

WAIT は解析されますが、それ以外は無視されるため、MySQL 5.6 では何の効果もありません。これは将来の拡張のために用意されています。

ALTER TABLESPACE ... ADD DATAFILE が ENGINE = NDB とともに使用された場合は、データファイルが各クラスタデータノード上に作成されます。INFORMATION_SCHEMA.FILES テーブルをクエリーすることによって、データファイルが作成されたことを確認したり、それらに関する情報を取得したりできます。たとえば、次のクエリーは、newts という名前のテーブルスペースに属するすべてのデータファイルを表示します。

```
mysql> SELECT LOGFILE_GROUP_NAME, FILE_NAME, EXTRA
    -> FROM INFORMATION_SCHEMA.FILES
    -> WHERE TABLESPACE_NAME = 'newts' AND FILE_TYPE = 'DATAFILE';
+--------------------+--------------+----------------+
| LOGFILE_GROUP_NAME | FILE_NAME    | EXTRA          |
+--------------------+--------------+----------------+
| lg_3               | newdata.dat  | CLUSTER_NODE=3 |
| lg_3               | newdata.dat  | CLUSTER_NODE=4 |
| lg_3               | newdata2.dat | CLUSTER_NODE=3 |
| lg_3               | newdata2.dat | CLUSTER_NODE=4 |
+--------------------+--------------+----------------+
2 rows in set (0.03 sec)
```

セクション21.30.1「INFORMATION_SCHEMA FILES テーブル」を参照してください。

ALTER TABLESPACE は、MySQL Cluster のディスクデータストレージでのみ有効です。セクション18.5.12「MySQL Cluster ディスクデータテーブル」を参照してください。

13.1.9 ALTER VIEW 構文

```
ALTER
    [ALGORITHM = {UNDEFINED | MERGE | TEMPTABLE}]
    [DEFINER = { user | CURRENT_USER }]
    [SQL SECURITY { DEFINER | INVOKER }]
    VIEW view_name [(column_list)]
    AS select_statement
    [WITH [CASCADED | LOCAL] CHECK OPTION]
```

このステートメントは、ビューの定義を変更します。このビューは存在する必要があります。この構文は CREATE VIEW の構文に似ていますが、その効果は CREATE OR REPLACE VIEW と同じです。セクション13.1.20「CREATE VIEW 構文」を参照してください。このステートメントには、このビューに対する CREATE VIEW および DROP 権限と、SELECT ステートメントで参照される各カラムに対する何らかの権限が必要です。ALTER VIEW は、定義者または SUPER 権限を持つユーザーにのみ許可されます。

13.1.10 CREATE DATABASE 構文

```
CREATE {DATABASE | SCHEMA} [IF NOT EXISTS] db_name
    [create_specification] ...

create_specification:
    [DEFAULT] CHARACTER SET [=] charset_name
  | [DEFAULT] COLLATE [=] collation_name
```

CREATE DATABASE は、指定された名前を持つデータベースを作成します。このステートメントを使用するには、このデータベースに対する CREATE 権限が必要です。CREATE SCHEMA は CREATE DATABASE のシノニムです。

そのデータベースが存在するときに IF NOT EXISTS を指定しなかった場合は、エラーが発生します。

MySQL 5.6 では、アクティブな LOCK TABLES ステートメントが存在するセッション内では CREATE DATABASE が許可されません。

create_specification オプションは、データベースの特性を指定します。データベースの特性は、データベースディレクトリ内の db.opt ファイルに格納されます。CHARACTER SET 句は、デフォルトのデータベース文字セットを指定します。COLLATE 句は、デフォルトのデータベース照合順序を指定します。セクション10.1「文字セットのサポート」では、文字セットと照合順序名について説明しています。

MySQL でのデータベースは、そのデータベース内のテーブルに対応するファイルを含むディレクトリとして実装されます。データベースが最初に作成されたとき、その中にはテーブルが存在しないため、CREATE DATABASE ステートメントは、MySQL データディレクトリの下のディレクトリと db.opt ファイルのみを作成します。許可されるデータベース名のルールは、セクション9.2「スキーマオブジェクト名」に示されています。データベース

名に特殊文字が含まれている場合は、セクション9.2.3「識別子とファイル名のマッピング」で説明されているように、その文字のエンコードされたバージョンがデータベースディレクトリの名前に含まれます。

データディレクトリの下に (たとえば、mkdir で) ディレクトリを手動で作成すると、サーバーはそれをデータベースディレクトリと見なし、SHOW DATABASES の出力に表示します。

mysqladmin プログラムを使用してデータベースを作成することもできます。セクション4.5.2「mysqladmin — MySQL サーバーの管理を行うクライアント」を参照してください。

13.1.11 CREATE EVENT 構文

```
CREATE
    [DEFINER = { user | CURRENT_USER }]
    EVENT
    [IF NOT EXISTS]
    event_name
    ON SCHEDULE schedule
    [ON COMPLETION [NOT] PRESERVE]
    [ENABLE | DISABLE | DISABLE ON SLAVE]
    [COMMENT 'comment']
    DO event_body;

schedule:
    AT timestamp [+ INTERVAL interval] ...
  | EVERY interval
    [STARTS timestamp [+ INTERVAL interval] ...]
    [ENDS timestamp [+ INTERVAL interval] ...]

interval:
    quantity {YEAR | QUARTER | MONTH | DAY | HOUR | MINUTE |
              WEEK | SECOND | YEAR_MONTH | DAY_HOUR | DAY_MINUTE |
              DAY_SECOND | HOUR_MINUTE | HOUR_SECOND | MINUTE_SECOND}
```

このステートメントは、新しいイベントを作成してスケジュールします。このイベントは、イベントスケジューラが有効になっていないかぎり実行されません。イベントスケジューラのステータスをチェックし、必要に応じてそれを有効にする方法については、セクション20.4.2「イベントスケジューラの構成」を参照してください。

CREATE EVENT には、イベントが作成されるスキーマに対する EVENT 権限が必要です。このセクションのあとの方で説明されているように、DEFINER 値によっては SUPER 権限も必要になる可能性があります。

有効な CREATE EVENT ステートメントの最小要件は次のとおりです。

- キーワード CREATE EVENT に加えて、データベーススキーマ内のイベントを一意に識別するイベント名。
- イベントが実行される時期と頻度を決定する ON SCHEDULE 句。
- イベントによって実行される SQL ステートメントを含む DO 句。

最小限の CREATE EVENT ステートメントの例を次に示します。

```
CREATE EVENT myevent
    ON SCHEDULE AT CURRENT_TIMESTAMP + INTERVAL 1 HOUR
    DO
      UPDATE myschema.mytable SET mycol = mycol + 1;
```

前のステートメントは、myevent という名前のイベントを作成します。このイベントは、myschema.mytable テーブルの mycol カラムの値を 1 増分する SQL ステートメントを実行することによって (その作成の 1 時間後に) 1 回実行されます。

event_name は、最大長が 64 文字の有効な MySQL 識別子である必要があります。イベント名は大文字と小文字が区別されないため、myevent と MyEvent という名前の 2 つのイベントを同じスキーマ内に含めることはできません。一般に、イベント名を管理するルールは、ストアドルーチンの名前の場合と同じです。セクション9.2「スキーマオブジェクト名」を参照してください。

イベントはスキーマに関連付けられています。event_name の一部としてスキーマが示されていない場合は、デフォルトの (現在の) スキーマと見なされます。イベントを特定のスキーマ内に作成するには、schema_name.event_name 構文を使用して、そのイベント名をスキーマで修飾します。

DEFINER 句は、イベントの実行時にアクセス権限を確認するときに使用される MySQL アカウントを指定します。user 値を指定する場合は、'user_name'@'host_name' (GRANT ステートメントで使用されるのと同じ形式)、CURRENT_USER、または CURRENT_USER() として指定された MySQL アカウントにしてください。DEFINER のデフォルト値は、CREATE EVENT ステートメントを実行するユーザーです。これは、明示的に DEFINER = CURRENT_USER を指定するのと同じです。

DEFINER 句を指定した場合は、次のルールによって有効な DEFINER ユーザーの値が決定されます。

- SUPER 権限がない場合、許可される唯一の user 値は、リテラルで指定するか、または CURRENT_USER を使用して指定した自分のアカウントです。定義者をほかのアカウントに設定することはできません。
- SUPER 権限がある場合は、構文として有効な任意のアカウント名を指定できます。そのアカウントが実際に存在しない場合は、警告が生成されます。
- 存在しない DEFINER アカウントでイベントを作成することはできますが、そのアカウントが存在しない場合は、イベント実行時にエラーが発生します。

イベントのセキュリティーの詳細は、セクション20.6「ストアドプログラムおよびビューのアクセスコントロール」を参照してください。

イベント内では、CURRENT_USER() 関数が、イベント実行時に権限を確認するために使用されるアカウント (DEFINER ユーザー) を返します。イベント内のユーザー監査については、セクション6.3.13「SQL ベースの MySQL アカウントアクティビティーの監査」を参照してください。

CREATE EVENT での IF NOT EXISTS には、CREATE TABLE での場合と同じ意味があります。event_name という名前のイベントが同じスキーマ内にすでに存在する場合、アクションは実行されず、エラーも発生しません。(ただし、このような場合は警告が生成されます。)

ON SCHEDULE 句は、そのイベントに対して定義された event_body を繰り返す時期、頻度、および期間を決定します。この句は、次の 2 つの形式のいずれかを取ります。

- 1 回限りのイベントには、AT timestamp が使用されます。これは、そのイベントが timestamp で指定された日付と時間に 1 回だけ実行されることを指定します。この値は、日付と時間の両方を含んでいるか、または datetime 値に解決される式である必要があります。この目的には、DATETIME または TIMESTAMP 型のどちらかの値を使用できます。日付が過去の日付である場合は、次に示すように、警告が発生します。

```
mysql> SELECT NOW();
+---------------------+
| NOW()               |
+---------------------+
| 2006-02-10 23:59:01 |
+---------------------+
1 row in set (0.04 sec)

mysql> CREATE EVENT e_totals
    ->     ON SCHEDULE AT '2006-02-10 23:59:00'
    ->     DO INSERT INTO test.totals VALUES (NOW());
Query OK, 0 rows affected, 1 warning (0.00 sec)

mysql> SHOW WARNINGS\G
*************************** 1. row ***************************
  Level: Note
   Code: 1588
Message: Event execution time is in the past and ON COMPLETION NOT
         PRESERVE is set. The event was dropped immediately after
         creation.
```

どのような理由であれ、それ自体が無効な CREATE EVENT ステートメントはエラーで失敗します。

現在の日付と時間を指定するには、CURRENT_TIMESTAMP を使用できます。このような場合、イベントは、作成されるとすぐに機能します。

現在の日付と時間を基準にした将来のある時点 (「今から 3 週間後」というフレーズで表される時点など) に発生するイベントを作成するには、オプションの句 + INTERVAL interval を使用できます。interval 部分は、数量と時間単位の 2 つの部分で構成され、DATE_ADD() 関数で使用される間隔を管理するのと同じ構文ルールに従います (セクション12.7「日付および時間関数」を参照してください)。また、単位のキーワードも、イベントを定義する場合はマイクロ秒を含む単位を使用できない点を除いて同じです。一部の間隔型では、複合の時間単位を使用できます。たとえば、「2 分と 10 秒」は、+ INTERVAL '2:10' MINUTE_SECOND として表すことができます。

また、間隔を組み合わせることもできます。たとえば、AT CURRENT_TIMESTAMP + INTERVAL 3 WEEK + INTERVAL 2 DAY は、「今から 3 週間と 2 日後」と同等です。このような句の各部分は、+ INTERVAL で始まる必要があります。

- アクションを定期的に繰り返すには、EVERY 句を使用します。EVERY キーワードのあとに、前の AT キーワードの説明に示されている interval を指定します。(EVERY では + INTERVAL は使用されません。)たとえば、EVERY 6 WEEK は「6 週間ごと」を示します。

EVERY 句では + INTERVAL 句は許可されていませんが、+ INTERVAL 内で許可されているのと同じ複合の時間単位を使用できます。

EVERY 句には、オプションの STARTS 句を含めることができます。STARTS のあとに、このアクションがいつ繰り返しを開始するかを示す timestamp 値を指定します。また、+ INTERVAL interval を使用して、「今からの」時間を指定することもできます。たとえば、EVERY 3 MONTH STARTS CURRENT_TIMESTAMP + INTERVAL 1 WEEK は、「今から 1 週間後に開始して 3 か月ごと」を示します。同様に、「今から 6 時間と 15 分後から開始して 2 週ごと」を、EVERY 2 WEEK STARTS CURRENT_TIMESTAMP + INTERVAL '6:15' HOUR_MINUTE として表すことができます。STARTS を指定しないことは、STARTS CURRENT_TIMESTAMP を使用することと同じです。つまり、イベントに対して指定されたアクションは、そのイベントが作成されるとただちに繰り返しを開始します。

EVERY 句には、オプションの ENDS 句を含めることができます。ENDS キーワードのあとに、このイベントがいつ繰り返しを停止するかを MySQL に指示する timestamp 値を指定します。また、ENDS とともに + INTERVAL interval を使用することもできます。たとえば、EVERY 12 HOUR STARTS CURRENT_TIMESTAMP + INTERVAL 30 MINUTE ENDS CURRENT_TIMESTAMP + INTERVAL 4 WEEK は、「今から 30 分後に開始し、今から 4 週間後に終了するまで 12 時間ごと」と同等です。ENDS を使用しないことは、このイベントがいつまでも実行を続行することを示します。

ENDS は、複合の時間単位に対して STARTS と同じ構文をサポートします。

EVERY 句では、STARTS または ENDS、あるいはその両方を使用できます。また、どちらも使用しないことも可能です。

繰り返しイベントがスケジュール間隔内に終了しない場合は、イベントの複数のインスタンスが同時に実行される可能性があります。これが好ましくない場合は、同時インスタンスを回避するためのメカニズムを設けてください。たとえば、GET_LOCK() 関数や、行またはテーブルのロックを使用できます。

ON SCHEDULE 句では、組み込みの MySQL 関数やユーザー変数を含む式を使用して、そこに含まれているすべての timestamp または interval 値を取得できます。このような式でストアドファンクションやユーザー定義関数を使用したり、テーブル参照を使用したりすることはできません。ただし、SELECT FROM DUAL は使用できます。これは、CREATE EVENT ステートメントと ALTER EVENT ステートメントの両方に当てはまります。このような場合のストアドファンクション、ユーザー定義関数、およびテーブルへの参照は明確に禁止されており、エラーで失敗します (Bug #22830 を参照してください)。

ON SCHEDULE 句の時間は、現在のセッションの time_zone 値を使用して解釈されます。これがイベントのタイムゾーン、つまり、イベントのスケジューリングに使用され、イベントが実行されるとそのイベント内で有効になるタイムゾーンになります。これらの時間は UTC に変換され、イベントのタイムゾーンとともに mysql.event テーブル内に格納されます。これにより、サーバータイムゾーンまたはサマータイムの影響に対し生じた変更とは無関係に、定義されたとおりにイベントの実行を処理できます。イベントの時間の表現の詳細は、セクション20.4.4「イベントメタデータ」を参照してください。セクション13.7.5.19「SHOW EVENTS 構文」およびセクション21.7「INFORMATION_SCHEMA EVENTS テーブル」も参照してください。

通常は、イベントの期限が切れると、そのイベントはただちに削除されます。この動作は、ON COMPLETION PRESERVE を指定することによってオーバーライドできます。ON COMPLETION NOT PRESERVE を使用すると、単にデフォルトの非持続性の動作が明示的になるだけです。

DISABLE キーワードを使用すると、イベントは作成するが、それがアクティブにならないようにすることができます。あるいは、ENABLE を使用して、デフォルトステータス (アクティブ) を明示的にすることもできます。これは、ALTER EVENT と組み合わせるともっとも有効です (セクション13.1.2「ALTER EVENT 構文」を参照してください)。

ENABLE や DISABLE の代わりに 3 番目の値を使用することもできます。DISABLE ON SLAVE は、イベントがマスター上で作成されてスレーブにレプリケートされたが、まだスレーブ上で実行されていないことを示すために、レプリケーションスレーブ上のイベントのステータスに対して設定されます。セクション17.4.1.11「呼び出される機能のレプリケーション」を参照してください。

COMMENT 句を使用して、イベントに対するコメントを指定できます。comment には、イベントの説明に使用する、最大 64 文字の任意の文字列を指定できます。コメントテキストは文字列リテラルであるため、引用符で囲む必要があります。

DO 句は、イベントによって実行されるアクションを指定するものであり、SQL ステートメントで構成されます。ストアドルーチンで使用できる有効な MySQL ステートメントのほぼすべてを、スケジュールされたイベントのアクションステートメントとしても使用できます。(セクションD.1「ストアドプログラムの制約」を参照してください。)たとえば、次のイベント e_hourly は、sessions テーブルのすべての行を 1 時間に 1 回削除します。ここで、このテーブルは site_activity スキーマの一部です。

```
CREATE EVENT e_hourly
  ON SCHEDULE
    EVERY 1 HOUR
  COMMENT 'Clears out sessions table each hour.'
  DO
    DELETE FROM site_activity.sessions;
```

MySQL は、イベントが作成または変更されたときの有効な sql_mode システム変数の設定を格納し、イベントが実行を開始したときの現在のサーバー SQL モードには関係なく、常にそのイベントを強制的にこの設定で実行します。

DO 句に ALTER EVENT ステートメントを含む CREATE EVENT ステートメントは成功したように見えます。ただし、結果として得られるスケジュールされたイベントをサーバーが実行しようとすると、その実行はエラーで失敗します。

> **注記**
>
> 単に結果セットを返す SELECT や SHOW などのステートメントは、イベントで使用されても何の効果もありません。これらのステートメントからの出力は MySQL モニターに送信されず、またどこにも格納されません。ただし、結果を格納する SELECT ... INTO や INSERT INTO ... SELECT などのステートメントは使用できます。(後者の例については、このセクションにある次の例を参照してください。)

イベントが属するスキーマは、DO 句でのテーブル参照のためのデフォルトスキーマです。ほかのスキーマでのテーブルへの参照はすべて、正しいスキーマ名で修飾する必要があります。

次に示すように、ストアドルーチンと同様に、BEGIN および END キーワードを使用して DO 句で複合ステートメントの構文を使用できます。

```
delimiter |

CREATE EVENT e_daily
  ON SCHEDULE
    EVERY 1 DAY
  COMMENT 'Saves total number of sessions then clears the table each day'
  DO
    BEGIN
      INSERT INTO site_activity.totals (time, total)
        SELECT CURRENT_TIMESTAMP, COUNT(*)
          FROM site_activity.sessions;
      DELETE FROM site_activity.sessions;
    END |

delimiter ;
```

この例では、delimiter コマンドを使用して、ステートメント区切り文字を変更します。セクション20.1「ストアドプログラムの定義」を参照してください。

イベントでは、ストアドルーチンで使用されているような、より複雑な複合ステートメントを使用できます。この例では、ローカル変数、エラーハンドラ、およびフロー制御構造構文を使用しています。

```
delimiter |

CREATE EVENT e
  ON SCHEDULE
    EVERY 5 SECOND
  DO
    BEGIN
      DECLARE v INTEGER;
      DECLARE CONTINUE HANDLER FOR SQLEXCEPTION BEGIN END;

      SET v = 0;

      WHILE v < 5 DO
        INSERT INTO t1 VALUES (0);
        UPDATE t2 SET s1 = s1 + 1;
        SET v = v + 1;
      END WHILE;
  END |

delimiter ;
```

イベントに、またはイベントから直接パラメータを渡す方法はありませんが、パラメータを持つストアドルーチンをイベント内で呼び出すことは可能です。

```
CREATE EVENT e_call_myproc
```

```
  ON SCHEDULE
    AT CURRENT_TIMESTAMP + INTERVAL 1 DAY
  DO CALL myproc(5, 27);
```

イベントの定義者に SUPER 権限がある場合、そのイベントはグローバル変数の読み取りおよび書き込みが可能です。この権限を付与すると悪用される可能性があるため、これを行う場合は十分に注意する必要があります。

一般に、ストアドルーチンで有効なすべてのステートメントを、イベントによって実行されるアクションステートメントに使用できます。ストアドルーチン内で許可されるステートメントの詳細は、セクション20.2.1「ストアドルーチンの構文」を参照してください。ストアドルーチンの一部としてイベントを作成できますが、イベントを別のイベントで作成することはできません。

13.1.12 CREATE FUNCTION 構文

CREATE FUNCTION ステートメントは、ストアドファンクションやユーザー定義関数 (UDF) を作成するために使用されます。

- ストアドファンクションの作成については、セクション13.1.15「CREATE PROCEDURE および CREATE FUNCTION 構文」を参照してください。
- ユーザー定義関数の作成については、セクション13.7.3.1「ユーザー定義関数のための CREATE FUNCTION 構文」を参照してください。

13.1.13 CREATE INDEX 構文

```
CREATE [ONLINE|OFFLINE] [UNIQUE|FULLTEXT|SPATIAL] INDEX index_name
    [index_type]
    ON tbl_name (index_col_name,...)
    [index_option]
    [algorithm_option | lock_option] ...

index_col_name:
    col_name [(length)] [ASC | DESC]

index_type:
    USING {BTREE | HASH}

index_option:
    KEY_BLOCK_SIZE [=] value
  | index_type
  | WITH PARSER parser_name
  | COMMENT 'string'

algorithm_option:
    ALGORITHM [=] {DEFAULT|INPLACE|COPY}

lock_option:
    LOCK [=] {DEFAULT|NONE|SHARED|EXCLUSIVE}
```

CREATE INDEX は、インデックスを作成するために ALTER TABLE ステートメントにマップされます。セクション13.1.7「ALTER TABLE 構文」を参照してください。CREATE INDEX を使用して PRIMARY KEY を作成することはできません。代わりに ALTER TABLE を使用します。インデックスの詳細は、セクション8.3.1「MySQL のインデックスの使用の仕組み」を参照してください。

通常、テーブル上のすべてのインデックスは、そのテーブル自体が CREATE TABLE で作成された時点で作成します。セクション13.1.17「CREATE TABLE 構文」を参照してください。このガイドラインは、主キーによってデータファイル内の行の物理配列が決定される InnoDB テーブルの場合に特に重要です。CREATE INDEX では、既存のテーブルにインデックスを追加できます。

(col1,col2,...) という形式のカラムリストは、マルチカラムインデックスを作成します。インデックスキーの値は、特定のカラムの値を連結することによって形成されます。

col_name(length) 構文を使用してインデックスプリフィクス長を指定することにより、カラム値の先頭の部分のみを使用するインデックスを作成できます。

- プリフィクスは、CHAR、VARCHAR、BINARY、および VARBINARY カラムに対して指定できます。
- BLOB および TEXT カラムにもインデックスを設定できますが、プリフィクス長を指定する必要があります。
- プリフィクス長は、バイナリ以外の文字列型の場合は文字数で、バイナリ文字列型の場合はバイト単位で指定されます。つまり、インデックスエントリは、CHAR、VARCHAR、および TEXT カラムの場合は各カラム値の最初の length 文字、BINARY、VARBINARY、および BLOB カラムの場合は各カラム値の最初の length バイトで構成されます。

- このセクションのあとの方で説明されているように、空間カラムの場合はプリフィクス値を指定できません。

次に示すステートメントは、name カラムの最初の 10 文字を使用してインデックスを作成します。

```
CREATE INDEX part_of_name ON customer (name(10));
```

カラム内の名前が一般に最初の 10 文字と異なっている場合は、このインデックスが、name カラム全体から作成されたインデックスよりはるかに遅いことはないはずです。また、インデックスにカラムプリフィクスを使用するとインデックスファイルをはるかに小さくできるため、多くのディスク領域が節約されるだけでなく、INSERT 操作も高速化される可能性があります。

プリフィクスのサポートやプリフィクスの長さ (サポートされている場合) は、ストレージエンジンに依存します。たとえば、InnoDB テーブルではプリフィクスの長さを最大 767 バイトに、また innodb_large_prefix オプションが有効になっている場合は 3072 バイトにすることができます。MyISAM テーブルの場合、プリフィクスの制限は 1000 バイトです。

> **注記**
>
> プリフィクスの制限がバイト単位で測定されるのに対して、CREATE INDEX ステートメントでのプリフィクス長は、バイナリ以外のデータ型 (CHAR、VARCHAR、TEXT) では文字数として解釈されます。複数バイトの文字セットを使用するカラムのプリフィクス長を指定する場合は、この点を考慮に入れてください。

NDBCLUSTER テーブルの可変幅カラム上のインデックスはオンラインで、つまり、テーブルコピーを行うことなく作成されます。このテーブルは、この操作の期間中、同じ API ノードに対するほかの操作に対してロックされますが、ほかの MySQL Cluster API ノードからのアクセスに対してはロックされません。これは、サーバーが実行できると判断した場合は常に、そのサーバーによって自動的に実行されます。これを実行するために、特殊な SQL 構文やサーバーオプションを使用する必要はありません。

標準の MySQL 5.6 リリースでは、サーバーがテーブルコピーなしでインデックスを作成すると決定したとき、そのサーバーをオーバーライドすることはできません。MySQL Cluster では、OFFLINE キーワードを使用してインデックスをオフラインで作成できます (これにより、そのテーブルはクラスタ内のすべての API ノードに対してロックされます)。CREATE OFFLINE INDEX および CREATE ONLINE INDEX を管理するルールや制限は、ALTER OFFLINE TABLE ... ADD INDEX および ALTER ONLINE TABLE ... ADD INDEX の場合と同じです。ONLINE キーワードを使用して、通常はオフラインで作成されるインデックスのコピーなし作成が実行されるようにすることはできません。CREATE INDEX 操作をテーブルコピーなしで実行できない場合、サーバーは ONLINE キーワードを無視します。詳細は、セクション13.1.7.2「MySQL Cluster での ALTER TABLE オンライン操作」を参照してください。

ONLINE および OFFLINE キーワードは、MySQL Cluster でのみ使用できます。これらのキーワードを標準の MySQL Server 5.6 リリースで使用しようとすると、構文エラーが発生します。ONLINE および OFFLINE キーワードは、MySQL Cluster NDB 7.3 では非推奨です。MySQL Cluster NDB 7.4 では引き続きサポートされますが、将来の MySQL Cluster リリースで削除される可能性があります。

UNIQUE インデックスは、そのインデックス内のすべての値が異なっている必要があるという制約を作成します。既存の行に一致するキー値を持つ新しい行を追加しようとすると、エラーが発生します。すべてのエンジンについて、UNIQUE インデックスは、NULL を含むことができるカラムでの複数の NULL 値を許可します。UNIQUE インデックス内のカラムのプリフィクス値を指定する場合は、カラム値がプリフィクス内で一意である必要があります。

FULLTEXT インデックスは InnoDB および MyISAM テーブルでのみサポートされ、CHAR、VARCHAR、および TEXT カラムのみを含めることができます。インデックス設定は常に、カラム全体に対して実行されます。カラムプリフィクスのインデックス設定はサポートされていないため、プリフィクス長が指定されてもすべて無視されます。操作の詳細は、セクション12.9「全文検索関数」を参照してください。

MyISAM、InnoDB、NDB、および ARCHIVE ストレージエンジンは、POINT や GEOMETRY などの空間カラムをサポートしています。(セクション11.5「空間データの拡張」では、空間データ型について説明しています。)ただし、空間カラムのインデックス設定に対するサポートはエンジンによって異なります。空間および非空間インデックスは、次のルールに従って使用できます。

(SPATIAL INDEX を使用して作成された) 空間インデックスには、次の特性があります。

- MyISAM テーブルでのみ使用できます。その他のストレージエンジンに対して SPATIAL INDEX を指定すると、エラーが発生します。
- インデックス付きカラムは NOT NULL である必要があります。
- MySQL 5.6 では、カラムプリフィクス長は禁止されています。各カラムの幅全体にインデックスが設定されます。

INDEX、UNIQUE、または PRIMARY KEY で作成された非空間インデックスの特性は次のとおりです。

- ARCHIVE を除く空間カラムをサポートするすべてのストレージエンジンに対して許可されます。
- インデックスが主キーでないかぎり、カラムを NULL にすることができます。
- POINT カラムを除く非 SPATIAL インデックス内の空間カラムごとに、カラムプリフィクス長を指定する必要があります。(これは、インデックス付き BLOB カラムの場合と同じ要件です。)プリフィクス長は、バイト単位で指定されます。
- 非 SPATIAL インデックスのインデックスタイプは、ストレージエンジンによって異なります。現在は、B ツリーが使用されます。

MySQL 5.6 では次のとおりです。

- NULL 値を持つことができるカラムにインデックスを追加できるのは、InnoDB、MyISAM、または MEMORY ストレージエンジンを使用している場合だけです。
- BLOB または TEXT カラムにインデックスを追加できるのは、InnoDB または MyISAM ストレージエンジンを使用している場合だけです。
- innodb_stats_persistent 設定が有効になっている場合は、InnoDB テーブル上でインデックスを作成したあと、そのテーブルに対して ANALYZE TABLE ステートメントを実行します。

index_col_name の指定を ASC または DESC で終了させることができます。これらのキーワードは、インデックス値の昇順または降順での格納を指定する将来の拡張のために許可されています。現在、これらは解析されますが、無視されます。インデックス値は、常に昇順で格納されます。

インデックスカラムリストのあとに、インデックスオプションを指定できます。index_option 値には、次のいずれかを指定できます。

- KEY_BLOCK_SIZE [=] value

  オプションで、インデックスキーのブロックに使用するサイズをバイト単位で指定します。この値はヒントとして扱われます。必要に応じて、異なるサイズが使用される可能性があります。

  > **注記**
  >
  > KEY_BLOCK_SIZE は、InnoDB に対してテーブルレベルでのみサポートされます。セクション13.1.17「CREATE TABLE 構文」を参照してください。

- index_type

  一部のストレージエンジンでは、インデックスの作成時にインデックスタイプを指定できます。ストレージエンジンごとにサポートされている許可されるインデックスタイプ値を次の表に示します。複数のインデックスタイプが示されている場合は、最初のものが、インデックスタイプ指示子が指定されないときのデフォルトになります。

ストレージエンジン	許可されるインデックスタイプ
InnoDB	BTREE
MyISAM	BTREE
MEMORY/HEAP	HASH、BTREE
NDB	HASH、BTREE (テキストの注を参照してください)

  例:

```
CREATE TABLE lookup (id INT) ENGINE = MEMORY;
CREATE INDEX id_index ON lookup (id) USING BTREE;
```

  BTREE インデックスは、NDBCLUSTER ストレージエンジンによって T ツリーインデックスとして実装されます。

  > **注記**
  >
  > NDB テーブルカラム上のインデックスの場合、USING オプションは、一意のインデックスまたは主キーに対してのみ指定できます。USING HASH は、暗黙的な順序付けされたインデックスが作成されないようにします。それ以外の場合は、NDB テーブル上に一意のインデックスまたは主キーを作成すると、順序付けされたインデック

> スとハッシュインデックスの両方が自動的に作成され、それぞれが同じ一連のカラムにインデックスを設定します。
>
> つまり、NULL カラム上の一意のインデックスまたは主キーを使用するクエリーは常に、NDB によってテーブルのフルスキャンで処理されます。特に、NDB テーブルの一意のインデックスまたは主キーカラムに関連した IS NULL または IS NOT NULL 条件を使用する予定がある場合は、このようなインデックスをすべて USING HASH なしで作成するようにしてください。

index_type 句を SPATIAL INDEX とともに使用することはできません。

特定のストレージエンジンに対して有効でないインデックスタイプを指定したが、そのエンジンがクエリー結果に影響を与えることなく使用できる使用可能な別のインデックスタイプが存在する場合、エンジンはその使用可能なタイプを使用します。パーサーは RTREE をタイプ名として認識しますが、現在、これはどのストレージエンジンに対しても指定できません。

このオプションを ON tbl_name 句の前に使用することは非推奨です。このオプションをこの位置で使用するためのサポートは、将来の MySQL リリースで削除される予定です。index_type オプションが前とあとの両方の位置で指定された場合は、最後のオプションが適用されます。

TYPE type_name は、USING type_name のシノニムとして認識されます。ただし、推奨される形式は USING です。

- WITH PARSER parser_name

  このオプションは、FULLTEXT インデックスとともにのみ使用できます。これは、全文インデックス設定および検索操作に特殊な処理が必要な場合に、パーサープラグインをインデックスに関連付けます。プラグインの作成の詳細は、セクション24.2「MySQL プラグイン API」を参照してください。

- COMMENT 'string'

  インデックス定義には、最大 1024 文字のオプションのコメントを含めることができます。

MySQL 5.6.6 の時点では、ALGORITHM および LOCK 句を指定できます。これらは、テーブルコピーの方法や、インデックスが変更されている間のテーブルの読み取りと書き込みの並列性のレベルに影響を与えます。これらには、ALTER TABLE ステートメントの場合と同じ意味があります。詳細は、セクション13.1.7「ALTER TABLE 構文」を参照してください。

13.1.14 CREATE LOGFILE GROUP 構文

```
CREATE LOGFILE GROUP logfile_group
    ADD UNDOFILE 'undo_file'
    [INITIAL_SIZE [=] initial_size]
    [UNDO_BUFFER_SIZE [=] undo_buffer_size]
    [REDO_BUFFER_SIZE [=] redo_buffer_size]
    [NODEGROUP [=] nodegroup_id]
    [WAIT]
    [COMMENT [=] comment_text]
    ENGINE [=] engine_name
```

このステートメントは、'undo_file' という名前の 1 つの UNDO ファイルを持つ logfile_group という名前の新しいログファイルグループを作成します。CREATE LOGFILE GROUP ステートメントには、ADD UNDOFILE 句が 1 つだけ存在します。ログファイルグループの命名を管理するルールについては、セクション9.2「スキーマオブジェクト名」を参照してください。

> 注記
>
> すべての MySQL Cluster ディスクデータオブジェクトが同じ名前空間を共有します。つまり、各ディスクデータオブジェクトは (単に、特定の型の各ディスクデータオブジェクトというだけでなく)、一意の名前が付けられている必要があります。たとえば、テーブルスペースとログファイルグループを同じ名前にしたり、テーブルスペースとデータファイルを同じ名前にしたりすることはできません。

MySQL Cluster NDB 7.3 以降では、クラスタあたり、常に 1 つのログファイルグループしか作成できません。(Bug #16386 を参照してください)

オプションの INITIAL_SIZE パラメータは、UNDO ファイルの初期サイズを設定します。指定されていない場合は、デフォルトで 128M (128M バイト) になります。オプションの UNDO_BUFFER_SIZE パラメータは、ログファイルグループの UNDO バッファーで使用されるサイズを設定します。UNDO_BUFFER_SIZE のデフォルト値は 8M (8M バイト) です。この値が、使用可能なシステムメモリーの量を超えることはできません。これらのパ

ラメータは、どちらもバイト単位で指定されます。MySQL Cluster NDB 7.3.2 以降では、これらの両方またはどちらか一方のあとにオプションで、my.cnf で使用されるのと同様の、桁を示す 1 文字の略語を指定できます。一般に、これは M (M バイト) または G (G バイト) のどちらかの文字です。MySQL Cluster NDB 7.3.2 より前は、これらのオプションの値は数字でしか指定できませんでした。(Bug #13116514、Bug #16104705、Bug #62858)

INITIAL_SIZE と UNDO_BUFFER_SIZE の両方に使用されるメモリーは、サイズが SharedGlobalMemory データノード構成パラメータの値によって決定されるグローバルプールから取得されます。これには、InitialLogFileGroup データノード構成パラメータの設定により、これらのオプションに暗黙的に指定されるデフォルト値もすべて含まれます。

UNDO_BUFFER_SIZE に許可される最大値は 629145600 (600M バイト) です。

32 ビットシステム上では、INITIAL_SIZE のサポートされる最大値は 4294967296 (4G バイト) です。(Bug #29186)

INITIAL_SIZE の許可される最小値は 1048576 (1M バイト) です。

ENGINE オプションは、このログファイルグループによって使用されるストレージエンジンを決定します。ここで、engine_name はそのストレージエンジンの名前です。MySQL 5.6 では、これは NDB (または NDBCLUSTER) である必要があります。ENGINE が設定されていない場合、MySQL は、default_storage_engine サーバーシステム変数 (以前の storage_engine) で指定されたエンジンを使用しようとします。いずれにしても、エンジンが NDB または NDBCLUSTER として指定されていない場合、CREATE LOGFILE GROUP ステートメントは成功したように見えますが、次に示すように、実際にはログファイルグループの作成に失敗します。

```
mysql> CREATE LOGFILE GROUP lg1
    ->     ADD UNDOFILE 'undo.dat' INITIAL_SIZE = 10M;
Query OK, 0 rows affected, 1 warning (0.00 sec)

mysql> SHOW WARNINGS;
+-------+------+-----------------------------------------------------------------------------------------------+
| Level | Code | Message                                                                                       |
+-------+------+-----------------------------------------------------------------------------------------------+
| Error | 1478 | Table storage engine 'InnoDB' does not support the create option 'TABLESPACE or LOGFILE GROUP' |
+-------+------+-----------------------------------------------------------------------------------------------+
1 row in set (0.00 sec)

mysql> DROP LOGFILE GROUP lg1 ENGINE = NDB;
ERROR 1529 (HY000): Failed to drop LOGFILE GROUP

mysql> CREATE LOGFILE GROUP lg1
    ->     ADD UNDOFILE 'undo.dat' INITIAL_SIZE = 10M
    ->     ENGINE = NDB;
Query OK, 0 rows affected (2.97 sec)
```

NDB 以外のストレージエンジンが指定されたときに CREATE LOGFILE GROUP ステートメントが実際にはエラーを返さず、成功したように見えるという事実は、MySQL Cluster の将来のリリースで対処したいと考えている既知の問題です。

REDO_BUFFER_SIZE、NODEGROUP、WAIT、および COMMENT は解析されますが、無視されるため、MySQL 5.6 では何の効果もありません。これらのオプションは、将来の拡張のために用意されています。

ENGINE [=] NDB とともに使用された場合は、ログファイルグループとそれに関連付けられた UNDO ログファイルが各クラスタデータノード上に作成されます。INFORMATION_SCHEMA.FILES テーブルをクエリーすることによって、UNDO ファイルが作成されたことを確認したり、それらに関する情報を取得したりできます。例:

```
mysql> SELECT LOGFILE_GROUP_NAME, LOGFILE_GROUP_NUMBER, EXTRA
    -> FROM INFORMATION_SCHEMA.FILES
    -> WHERE FILE_NAME = 'undo_10.dat';
+--------------------+----------------------+----------------+
| LOGFILE_GROUP_NAME | LOGFILE_GROUP_NUMBER | EXTRA          |
+--------------------+----------------------+----------------+
| lg_3               |                   11 | CLUSTER_NODE=3 |
| lg_3               |                   11 | CLUSTER_NODE=4 |
+--------------------+----------------------+----------------+
2 rows in set (0.06 sec)
```

CREATE LOGFILE GROUP は、MySQL Cluster のディスクデータストレージでのみ有効です。セクション 18.5.12「MySQL Cluster ディスクデータテーブル」を参照してください。

13.1.15 CREATE PROCEDURE および CREATE FUNCTION 構文

```
CREATE
    [DEFINER = { user | CURRENT_USER }]
    PROCEDURE sp_name ([proc_parameter[,...]])
```

```
  [characteristic ...] routine_body

CREATE
  [DEFINER = { user | CURRENT_USER }]
  FUNCTION sp_name ([func_parameter[,...]])
  RETURNS type
  [characteristic ...] routine_body

proc_parameter:
  [ IN | OUT | INOUT ] param_name type

func_parameter:
  param_name type

type:
  Any valid MySQL data type

characteristic:
  COMMENT 'string'
 | LANGUAGE SQL
 | [NOT] DETERMINISTIC
 | { CONTAINS SQL | NO SQL | READS SQL DATA | MODIFIES SQL DATA }
 | SQL SECURITY { DEFINER | INVOKER }

routine_body:
  Valid SQL routine statement
```

これらのステートメントは、ストアドルーチンを作成します。デフォルトでは、ルーチンはデフォルトデータベースに関連付けられます。ルーチンを明示的に特定のデータベースに関連付けるには、そのルーチンの作成時に、その名前を db_name.sp_name として指定します。

CREATE FUNCTION ステートメントはまた、UDF (ユーザー定義関数) をサポートするために MySQL でも使用されます。セクション24.3「MySQL への新しい関数の追加」を参照してください。UDF は、外部のストアドファンクションと見なすことができます。ストアドファンクションは、その名前空間を UDF と共有します。各種の関数への参照をサーバーが解釈する方法を記述したルールについては、セクション9.2.4「関数名の構文解析と解決」を参照してください。

ストアドプロシージャーを呼び出すには、CALL ステートメントを使用します (セクション13.2.1「CALL 構文」を参照してください)。ストアドファンクションを呼び出すには、式でその関数を参照します。その関数は、式の評価中に値を返します。

CREATE PROCEDURE および CREATE FUNCTION には、CREATE ROUTINE 権限が必要です。このセクションのあとの方で説明されているように、DEFINER 値によっては SUPER 権限も必要になる可能性があります。バイナリロギングが有効になっている場合は、セクション20.7「ストアドプログラムのバイナリロギング」で説明されているように、CREATE FUNCTION に SUPER 権限が必要になることがあります。

デフォルトでは、MySQL は、ルーチン作成者に ALTER ROUTINE および EXECUTE 権限を自動的に付与します。この動作は、automatic_sp_privileges システム変数を無効にすることによって変更できます。セクション20.2.2「ストアドルーチンと MySQL 権限」を参照してください。

DEFINER および SQL SECURITY 句は、このセクションのあとの方で説明されているように、ルーチンの実行時にアクセス権限を確認するときに使用されるセキュリティーコンテキストを指定します。

ルーチン名が組み込みの SQL 関数の名前と同じである場合は、そのルーチンを定義するか、またはあとで呼び出すときに名前とそれに続く括弧の間にスペースを使用しないかぎり、構文エラーが発生します。このため、ユーザー独自のストアドルーチンに既存の SQL 関数の名前を使用することは避けてください。

IGNORE_SPACE SQL モードは、ストアドルーチンではなく、組み込み関数に適用されます。ストアドルーチン名のあとのスペースは、IGNORE_SPACE が有効になっているかどうかには関係なく、常に許可されます。

括弧で囲まれたパラメータリストは、常に存在する必要があります。パラメータが存在しない場合は、() の空のパラメータリストを使用するようにしてください。パラメータ名は大文字と小文字が区別されません。

各パラメータは、デフォルトでは IN パラメータです。それ以外のパラメータを指定するには、パラメータ名の前にキーワード OUT または INOUT を使用します。

> **注記**
>
> IN、OUT、または INOUT としてのパラメータの指定は、PROCEDURE に対してのみ有効です。FUNCTION の場合、パラメータは常に IN パラメータと見なされます。

IN パラメータは、プロシージャーへの値を渡します。プロシージャーはその値を変更する可能性がありますが、そのプロシージャーから戻ったとき、その変更は呼び出し元に表示されません。OUT パラメータは、プロシー

ジャーから呼び出し元に値を渡します。その初期値はプロシージャー内では NULL であり、そのプロシージャーから戻ったとき、その値は呼び出し元に表示されます。INOUT パラメータは呼び出し元によって初期化され、プロシージャーで変更できます。そのプロシージャーから戻ったとき、プロシージャーによって行われた変更はすべて呼び出し元に表示されます。

OUT または INOUT パラメータごとに、プロシージャーを呼び出す CALL ステートメントでユーザー定義変数を渡して、プロシージャーから戻ったときにその値を取得できるようにします。そのプロシージャーを別のストアドプロシージャーまたはストアドファンクション内から呼び出している場合は、IN または INOUT パラメータとしてルーチンパラメータまたはローカルルーチン変数を渡すこともできます。

ルーチン内に準備されたステートメントでルーチンパラメータを参照することはできません。セクションD.1「ストアドプログラムの制約」を参照してください。

次の例は、OUT パラメータを使用する単純なストアドプロシージャーを示しています。

```
mysql> delimiter //

mysql> CREATE PROCEDURE simpleproc (OUT param1 INT)
    -> BEGIN
    ->   SELECT COUNT(*) INTO param1 FROM t;
    -> END//
Query OK, 0 rows affected (0.00 sec)

mysql> delimiter ;

mysql> CALL simpleproc(@a);
Query OK, 0 rows affected (0.00 sec)

mysql> SELECT @a;
+------+
| @a   |
+------+
| 3    |
+------+
1 row in set (0.00 sec)
```

この例では、プロシージャーの定義中に mysql クライアントの delimiter コマンドを使用して、ステートメント区切り文字を ; から // に変更しています。これにより、プロシージャー本体で使用される ; 区切り文字を、mysql 自体が解釈するのではなく、サーバーに渡すようにすることができます。セクション20.1「ストアドプログラムの定義」を参照してください。

RETURNS 句は、FUNCTION (これには必須です) に対してのみ指定できます。これは関数の戻り型を示すものであり、関数本体には RETURN value ステートメントが含まれている必要があります。RETURN ステートメントが異なる型の値を返した場合、その値は正しい型に強制的に変更されます。たとえば、ある関数が RETURNS 句で ENUM または SET 値を指定しているが、RETURN ステートメントが整数を返した場合、その関数から返される値は SET メンバーのセットの対応する ENUM メンバーを示す文字列になります。

次の関数例はパラメータを受け取り、SQL 関数を使用して操作を実行したあと、結果を返します。この場合は、関数定義に内部の ; ステートメント区切り文字が含まれていないため、delimiter を使用する必要はありません。

```
mysql> CREATE FUNCTION hello (s CHAR(20))
mysql> RETURNS CHAR(50) DETERMINISTIC
    -> RETURN CONCAT('Hello, ',s,'!');
Query OK, 0 rows affected (0.00 sec)

mysql> SELECT hello('world');
+----------------+
| hello('world') |
+----------------+
| Hello, world!  |
+----------------+
1 row in set (0.00 sec)
```

パラメータ型と関数の戻り型は、任意の有効なデータ型を使用するように宣言できます。前に CHARACTER SET 属性がある場合は、COLLATE 属性を使用できます。

routine_body は、有効な SQL ルーチンステートメントで構成されます。これは SELECT や INSERT などの単純なステートメントでも、BEGIN と END を使用して記述された複合ステートメントでもかまいません。複合ステートメントには、宣言、ループ、およびその他の制御構造ステートメントを含めることができます。これらのステートメントの構文については、セクション13.6「MySQL 複合ステートメント構文」で説明されています。

MySQL では、ルーチンに CREATE や DROP などの DDL ステートメントを含めることが許可されます。MySQL ではまた、ストアドプロシージャーに COMMIT などの SQL トランザクションステートメントを含めることも許

可されます (ただし、ストアドファンクションには許可されません)。ストアドファンクションには、明示的または暗黙的なコミットまたはロールバックを実行するステートメントを含めることはできません。これらのステートメントのサポートは、SQL 標準では必要ありません。SQL 標準では、各 DBMS ベンダーがこれらのステートメントを許可するかどうかを決められると定めています。

結果セットを返すステートメントはストアドプロシージャー内で使用できますが、ストアドファンクション内では使用できません。この禁止には、INTO var_list 句を含まない SELECT ステートメントや、SHOW、EXPLAIN、CHECK TABLE などのその他のステートメントが含まれます。結果セットを返すことを関数の定義時に判定できるステートメントの場合は、Not allowed to return a result set from a function エラーが発生します (ER_SP_NO_RETSET)。結果セットを返すことを実行時にしか判定できないステートメントの場合は、PROCEDURE %s can't return a result set in the given context エラーが発生します (ER_SP_BADSELECT)。

ストアドルーチン内での USE ステートメントは許可されていません。ルーチンが呼び出されると、暗黙的な USE db_name が実行されます (また、そのルーチンが終了すると元に戻されます)。これにより、そのルーチンには実行中、特定のデフォルトデータベースが割り当てられます。ルーチンのデフォルトデータベース以外のデータベース内のオブジェクトへの参照は、適切なデータベース名で修飾するようにしてください。

ストアドルーチン内では許可されないステートメントの詳細は、セクションD.1「ストアドプログラムの制約」を参照してください。

MySQL インタフェースを備える言語で記述されたプログラム内からのストアドプロシージャーの呼び出しについては、セクション13.2.1「CALL 構文」を参照してください。

MySQL は、ルーチンが作成または変更されたときの有効な sql_mode システム変数の設定を格納し、ルーチンが実行を開始したときの現在のサーバー SQL モードには関係なく、常にそのルーチンを強制的にこの設定で実行します。

呼び出し元の SQL モードからそのルーチンの SQL モードへの切り替えは、引数を評価し、結果として得られる値をルーチンパラメータに割り当てたあとに実行されます。あるルーチンを厳密な SQL モードで定義したが、その呼び出しを非厳密モードで行なった場合は、引数のルーチンパラメータへの割り当てが厳密モードで実行されません。ルーチンに渡される式を厳密な SQL モードで割り当てる必要がある場合は、そのルーチンを厳密モードが有効な状態で呼び出すようにしてください。

COMMENT 特性は MySQL 拡張であり、そのストアドルーチンの説明のために使用できます。この情報は、SHOW CREATE PROCEDURE および SHOW CREATE FUNCTION ステートメントによって表示されます。

LANGUAGE 特性は、そのルーチンが記述されている言語を示します。サーバーはこの特性を無視します。SQL ルーチンのみがサポートされています。

ルーチンは、同じ入力パラメータに対して常に同じ結果を生成する場合は「決定的」と見なされ、それ以外の場合は「非決定的」と見なされます。ルーチン定義で DETERMINISTIC と NOT DETERMINISTIC のどちらも指定されていない場合、デフォルトは NOT DETERMINISTIC になります。関数が決定的であることを宣言するには、明示的に DETERMINISTIC を指定する必要があります。

ルーチンの性質の評価は、作成者の「誠実さ」に基づいています。MySQL は、DETERMINISTIC と宣言されたルーチンに非決定的な結果を生成するステートメントが含まれていないかどうかをチェックしません。ただし、ルーチンの誤った宣言は、その結果やパフォーマンスに影響を与える可能性があります。非決定的なルーチンを DETERMINISTIC として宣言すると、オプティマイザが正しくない実行計画を選択するために、予期しない結果を招くことがあります。決定的なルーチンを NONDETERMINISTIC として宣言すると、使用可能な最適化が使用されなくなるために、パフォーマンスが低下することがあります。

バイナリロギングが有効になっている場合、DETERMINISTIC 特性は、MySQL がどのルーチン定義を受け入れるかに影響を与えます。セクション20.7「ストアドプログラムのバイナリロギング」を参照してください。

NOW() 関数 (または、そのシノニム) あるいは RAND() を含むルーチンは非決定的ですが、引き続きレプリケーションに対して安全である可能性があります。NOW() の場合、バイナリログにはタイムスタンプが含まれ、正しくレプリケートされます。RAND() もまた、ルーチンの実行中に 1 回だけ呼び出されるかぎり、正しくレプリケートされます。(ルーチン実行のタイムスタンプや乱数シードは、マスターとスレーブ上で同一の暗黙的な入力と見なすことができます。)

いくつかの特性によって、ルーチンによるデータ使用の性質に関する情報が提供されます。MySQL では、これらの特性はアドバイザリにすぎません。サーバーがこれらを使用して、あるルーチンにどのような種類のステートメントの実行を許可するかを制約することはありません。

- CONTAINS SQL は、そのルーチンに、データの読み取りや書き込みを行うステートメントが含まれていないことを示します。これは、これらのどの特性も明示的に指定されていない場合のデフォルトです。このようなステートメントの例として、実行されてもデータの読み取りや書き込みを行わない SET @x = 1 または DO RELEASE_LOCK('abc') があります。

- NO SQL は、そのルーチンに SQL ステートメントが含まれていないことを示します。
- READS SQL DATA は、そのルーチンに、データを読み取るステートメント (SELECT など) が含まれているが、データを書き込むステートメントは含まれていないことを示します。
- MODIFIES SQL DATA は、そのルーチンに、データを書き込む可能性のあるステートメント (INSERT や DELETE など) が含まれていることを示します。

SQL SECURITY 特性は、セキュリティーコンテキストを指定する DEFINER または INVOKER のどちらかです。これは、そのルーチンがルーチンの DEFINER 句で指定されたアカウント、またはそのルーチンを呼び出すユーザーのどちらの権限を使用して実行されるかを示します。このアカウントには、そのルーチンが関連付けられているデータベースにアクセスするためのアクセス権が必要です。デフォルト値は DEFINER です。そのルーチンを呼び出すユーザーには、それに対する EXECUTE 権限が必要です。また、そのルーチンが定義者のセキュリティーコンテキストで実行される場合は、DEFINER アカウントにもその権限が必要です。

DEFINER 句は、SQL SECURITY DEFINER 特性を持つルーチンのルーチン実行時にアクセス権限を確認するときに使用される MySQL アカウントを指定します。

DEFINER 句に user 値を指定する場合は、'user_name'@'host_name' (GRANT ステートメントで使用されるのと同じ形式)、CURRENT_USER、または CURRENT_USER() として指定された MySQL アカウントにするようにしてください。DEFINER のデフォルト値は、CREATE PROCEDURE または CREATE FUNCTION ステートメントを実行するユーザーです。これは、明示的に DEFINER = CURRENT_USER を指定するのと同じです。

DEFINER 句を指定した場合は、次のルールによって有効な DEFINER ユーザーの値が決定されます。

- SUPER 権限がない場合、許可される唯一の user 値は、リテラルで指定するか、または CURRENT_USER を使用して指定した自分のアカウントです。定義者をほかのアカウントに設定することはできません。
- SUPER 権限がある場合は、構文として有効な任意のアカウント名を指定できます。そのアカウントが実際に存在しない場合は、警告が生成されます。
- 存在しない DEFINER アカウントでルーチンを作成することはできますが、SQL SECURITY 値が DEFINER であるが、定義者アカウントが存在しない場合は、ルーチン実行時にエラーが発生します。

ストアドルーチンのセキュリティーの詳細は、セクション20.6「ストアドプログラムおよびビューのアクセスコントロール」を参照してください。

SQL SECURITY DEFINER 特性を使用して定義されたストアドルーチン内で、CURRENT_USER は、そのルーチンの DEFINER 値を返します。ストアドルーチン内のユーザー監査については、セクション6.3.13「SQL ベースの MySQL アカウントアクティビティーの監査」を参照してください。

mysql.user テーブルにリストされている MySQL アカウントの数を表示する次のプロシージャーを考えてみます。

```
CREATE DEFINER = 'admin'@'localhost' PROCEDURE account_count()
BEGIN
  SELECT 'Number of accounts:', COUNT(*) FROM mysql.user;
END;
```

このプロシージャーには、それがどのユーザーによって定義されている場合でも、'admin'@'localhost' の DEFINER アカウントが割り当てられます。また、それがどのユーザーから呼び出された場合でも、そのアカウントの権限で実行されます (デフォルトのセキュリティー特性は DEFINER であるため)。このプロシージャーは、呼び出し元にそれに対する EXECUTE 権限があり、かつ 'admin'@'localhost' に mysql.user テーブルに対する SELECT 権限があるかどうかに応じて成功または失敗します。

ここで、このプロシージャーが SQL SECURITY INVOKER 特性を使用して定義されているとします。

```
CREATE DEFINER = 'admin'@'localhost' PROCEDURE account_count()
SQL SECURITY INVOKER
BEGIN
  SELECT 'Number of accounts:', COUNT(*) FROM mysql.user;
END;
```

このプロシージャーは、依然として 'admin'@'localhost' の DEFINER を持っていますが、この場合は呼び出し元ユーザーの権限で実行されます。そのため、このプロシージャーは、呼び出し元にそれに対する EXECUTE 権限と、mysql.user テーブルに対する SELECT 権限があるかどうかに応じて成功または失敗します。

サーバーは、ルーチンパラメータ、DECLARE を使用して作成されたローカルルーチン変数、または関数の戻り値のデータ型を次のように処理します。

- データ型の不一致やオーバーフローがないかどうか割り当てがチェックされます。変換やオーバーフローの問題によって警告が発生するか、または厳密な SQL モードではエラーが発生します。

- スカラー値のみを割り当てることができます。たとえば、SET x = (SELECT 1, 2) などのステートメントは無効です。
- 文字データ型で、宣言内に CHARACTER SET 属性が存在する場合は、指定された文字セットとそのデフォルトの照合順序が使用されます。COLLATE 属性も存在する場合は、デフォルトの照合順序ではなく、その照合順序が使用されます。

  CHARACTER SET および COLLATE 属性が存在しない場合は、ルーチンの作成時に有効なデータベース文字セットおよび照合順序が使用されます。サーバーでデータベース文字セットおよび照合順序が使用されないようにするには、文字データパラメータとして明示的な CHARACTER SET および COLLATE 属性を指定します。

  データベースのデフォルトの文字セットまたは照合順序を変更する場合は、データベースのデフォルトを使用するストアドルーチンを削除および再作成して、それらが新しいデフォルトを使用するようにする必要があります。

  データベース文字セットおよび照合順序は、character_set_database および collation_database システム変数の値で指定されます。詳細は、セクション10.1.3.2「データベース文字セットおよび照合順序」を参照してください。

13.1.16 CREATE SERVER 構文

```
CREATE SERVER server_name
   FOREIGN DATA WRAPPER wrapper_name
   OPTIONS (option [, option] ...)

option:
 { HOST character-literal
 | DATABASE character-literal
 | USER character-literal
 | PASSWORD character-literal
 | SOCKET character-literal
 | OWNER character-literal
 | PORT numeric-literal }
```

このステートメントは、FEDERATED ストレージエンジンで使用するためのサーバーの定義を作成します。CREATE SERVER ステートメントは、mysql データベース内の servers テーブルに新しい行を作成します。このステートメントには、SUPER 権限が必要です。

server_name は、そのサーバーへの一意の参照にしてください。サーバー定義は、そのサーバーのスコープ内ではグローバルであるため、サーバー定義を特定のデータベースに対して修飾することはできません。server_name の最大長は 64 文字であり (64 文字より長い名前は暗黙のうちに切り捨てられます)、大文字小文字を区別しません。この名前は、引用符で囲まれた文字列として指定できます。

wrapper_name は mysql にしてください。また、それを単一引用符で囲むことができます。wrapper_name に対するその他の値は現在、サポートされていません。

各 option について、文字リテラルまたは数値リテラルのどちらかを指定する必要があります。文字リテラルは UTF-8 であり、64 文字の最大長をサポートし、デフォルトではブランク (空) の文字列になります。文字列リテラルは、暗黙のうちに 64 文字に切り捨てられます。数値リテラルは 0 から 9999 までの数字である必要があり、デフォルト値は 0 です。

> **注記**
>
> OWNER オプションは現在、適用されず、作成されるサーバー接続の所有権または操作には影響を与えません。

CREATE SERVER ステートメントは、mysql.servers テーブル内にエントリを作成します。これは、あとで FEDERATED テーブルを作成するときに CREATE TABLE ステートメントで使用できます。指定したオプションは、mysql.servers テーブル内のカラムを移入するために使用されます。テーブルカラムは、Server_name、Host、Db、Username、Password、Port、および Socket です。

例:

```
CREATE SERVER s
FOREIGN DATA WRAPPER mysql
OPTIONS (USER 'Remote', HOST '192.168.1.106', DATABASE 'test');
```

サーバーへの接続を確立するために必要なすべてのオプションを指定する必要があります。ユーザー名、ホスト名、およびデータベース名は必須です。パスワードなどの、その他のオプションも必要になる可能性があります。

このテーブルに格納されたデータは、FEDERATED テーブルへの接続を作成するときに使用できます。

```
CREATE TABLE t (s1 INT) ENGINE=FEDERATED CONNECTION='s';
```

詳細は、セクション15.8「FEDERATED ストレージエンジン」を参照してください。

CREATE SERVER では、自動コミットが実行されます。

MySQL 5.6 では、使用されている口ギング形式には関係なく、CREATE SERVER はバイナリログに書き込まれません。

MySQL 5.6.11 でのみ、このステートメントを発行する前に、gtid_next を AUTOMATIC に設定する必要があります。(Bug #16062608、Bug #16715809、Bug #69045)

13.1.17 CREATE TABLE 構文

```
CREATE [TEMPORARY] TABLE [IF NOT EXISTS] tbl_name
    (create_definition,...)
    [table_options]
    [partition_options]

CREATE [TEMPORARY] TABLE [IF NOT EXISTS] tbl_name
    [(create_definition,...)]
    [table_options]
    [partition_options]
    select_statement

CREATE [TEMPORARY] TABLE [IF NOT EXISTS] tbl_name
    { LIKE old_tbl_name | (LIKE old_tbl_name) }

create_definition:
    col_name column_definition
  | [CONSTRAINT [symbol]] PRIMARY KEY [index_type] (index_col_name,...)
      [index_option] ...
  | {INDEX|KEY} [index_name] [index_type] (index_col_name,...)
      [index_option] ...
  | [CONSTRAINT [symbol]] UNIQUE [INDEX|KEY]
      [index_name] [index_type] (index_col_name,...)
      [index_option] ...
  | {FULLTEXT|SPATIAL} [INDEX|KEY] [index_name] (index_col_name,...)
      [index_option] ...
  | [CONSTRAINT [symbol]] FOREIGN KEY
      [index_name] (index_col_name,...) reference_definition
  | CHECK (expr)

column_definition:
    data_type [NOT NULL | NULL] [DEFAULT default_value]
      [AUTO_INCREMENT] [UNIQUE [KEY] | [PRIMARY] KEY]
      [COMMENT 'string']
      [COLUMN_FORMAT {FIXED|DYNAMIC|DEFAULT}]
      [STORAGE {DISK|MEMORY|DEFAULT}]
      [reference_definition]

data_type:
    BIT[(length)]
  | TINYINT[(length)] [UNSIGNED] [ZEROFILL]
  | SMALLINT[(length)] [UNSIGNED] [ZEROFILL]
  | MEDIUMINT[(length)] [UNSIGNED] [ZEROFILL]
  | INT[(length)] [UNSIGNED] [ZEROFILL]
  | INTEGER[(length)] [UNSIGNED] [ZEROFILL]
  | BIGINT[(length)] [UNSIGNED] [ZEROFILL]
  | REAL[(length,decimals)] [UNSIGNED] [ZEROFILL]
  | DOUBLE[(length,decimals)] [UNSIGNED] [ZEROFILL]
  | FLOAT[(length,decimals)] [UNSIGNED] [ZEROFILL]
  | DECIMAL[(length[,decimals])] [UNSIGNED] [ZEROFILL]
  | NUMERIC[(length[,decimals])] [UNSIGNED] [ZEROFILL]
  | DATE
  | TIME[(fsp)]
  | TIMESTAMP[(fsp)]
  | DATETIME[(fsp)]
  | YEAR
  | CHAR[(length)]
      [CHARACTER SET charset_name] [COLLATE collation_name]
  | VARCHAR(length)
      [CHARACTER SET charset_name] [COLLATE collation_name]
  | BINARY[(length)]
  | VARBINARY(length)
  | TINYBLOB
  | BLOB
```

```
 | MEDIUMBLOB
 | LONGBLOB
 | TINYTEXT [BINARY]
     [CHARACTER SET charset_name] [COLLATE collation_name]
 | TEXT [BINARY]
     [CHARACTER SET charset_name] [COLLATE collation_name]
 | MEDIUMTEXT [BINARY]
     [CHARACTER SET charset_name] [COLLATE collation_name]
 | LONGTEXT [BINARY]
     [CHARACTER SET charset_name] [COLLATE collation_name]
 | ENUM(value1,value2,value3,...)
     [CHARACTER SET charset_name] [COLLATE collation_name]
 | SET(value1,value2,value3,...)
     [CHARACTER SET charset_name] [COLLATE collation_name]
 | spatial_type

index_col_name:
    col_name [(length)] [ASC | DESC]

index_type:
    USING {BTREE | HASH}

index_option:
    KEY_BLOCK_SIZE [=] value
  | index_type
  | WITH PARSER parser_name
  | COMMENT 'string'

reference_definition:
    REFERENCES tbl_name (index_col_name,...)
      [MATCH FULL | MATCH PARTIAL | MATCH SIMPLE]
      [ON DELETE reference_option]
      [ON UPDATE reference_option]

reference_option:
    RESTRICT | CASCADE | SET NULL | NO ACTION

table_options:
    table_option [[,] table_option] ...

table_option:
    ENGINE [=] engine_name
  | AUTO_INCREMENT [=] value
  | AVG_ROW_LENGTH [=] value
  | [DEFAULT] CHARACTER SET [=] charset_name
  | CHECKSUM [=] {0 | 1}
  | [DEFAULT] COLLATE [=] collation_name
  | COMMENT [=] 'string'
  | CONNECTION [=] 'connect_string'
  | DATA DIRECTORY [=] 'absolute path to directory'
  | DELAY_KEY_WRITE [=] {0 | 1}
  | INDEX DIRECTORY [=] 'absolute path to directory'
  | INSERT_METHOD [=] { NO | FIRST | LAST }
  | KEY_BLOCK_SIZE [=] value
  | MAX_ROWS [=] value
  | MIN_ROWS [=] value
  | PACK_KEYS [=] {0 | 1 | DEFAULT}
  | PASSWORD [=] 'string'
  | ROW_FORMAT [=] {DEFAULT|DYNAMIC|FIXED|COMPRESSED|REDUNDANT|COMPACT}
  | STATS_AUTO_RECALC [=] {DEFAULT|0|1}
  | STATS_PERSISTENT [=] {DEFAULT|0|1}
  | STATS_SAMPLE_PAGES [=] value
  | TABLESPACE tablespace_name [STORAGE {DISK|MEMORY|DEFAULT}]
  | UNION [=] (tbl_name[,tbl_name]...)

partition_options:
    PARTITION BY
        { [LINEAR] HASH(expr)
        | [LINEAR] KEY [ALGORITHM={1|2}] (column_list)
        | RANGE{(expr) | COLUMNS(column_list)}
        | LIST{(expr) | COLUMNS(column_list)} }
    [PARTITIONS num]
    [SUBPARTITION BY
        { [LINEAR] HASH(expr)
        | [LINEAR] KEY [ALGORITHM={1|2}] (column_list) }
      [SUBPARTITIONS num]
    ]
    [(partition_definition [, partition_definition] ...)]

partition_definition:
```

```
  PARTITION partition_name
    [VALUES
      {LESS THAN {(expr | value_list) | MAXVALUE}
      |
      IN (value_list)}]
    [[STORAGE] ENGINE [=] engine_name]
    [COMMENT [=] 'comment_text' ]
    [DATA DIRECTORY [=] 'data_dir']
    [INDEX DIRECTORY [=] 'index_dir']
    [MAX_ROWS [=] max_number_of_rows]
    [MIN_ROWS [=] min_number_of_rows]
    [TABLESPACE [=] tablespace_name]
    [NODEGROUP [=] node_group_id]
    [(subpartition_definition [, subpartition_definition] ...)]

subpartition_definition:
  SUBPARTITION logical_name
    [[STORAGE] ENGINE [=] engine_name]
    [COMMENT [=] 'comment_text' ]
    [DATA DIRECTORY [=] 'data_dir']
    [INDEX DIRECTORY [=] 'index_dir']
    [MAX_ROWS [=] max_number_of_rows]
    [MIN_ROWS [=] min_number_of_rows]
    [TABLESPACE [=] tablespace_name]
    [NODEGROUP [=] node_group_id]

select_statement:
  [IGNORE | REPLACE] [AS] SELECT ...  (Some valid select statement)
```

CREATE TABLE は、指定された名前を持つテーブルを作成します。このテーブルに対する CREATE 権限が必要です。

許可されるテーブル名のルールは、セクション9.2「スキーマオブジェクト名」に示されています。デフォルトでは、テーブルは InnoDB ストレージエンジンを使用して、デフォルトデータベース内に作成されます。テーブルがすでに存在する場合、デフォルトデータベースが存在しない場合、またはデータベースが存在しない場合は エラーが発生します。

特定のデータベース内にテーブルを作成するには、テーブル名を db_name.tbl_name として指定できます。そのデータベースが存在すると仮定すると、これは、デフォルトデータベースが存在するかどうかには関係なく機能します。引用符で囲まれた識別子を使用する場合は、データベース名とテーブル名を個別に引用符で囲みます。たとえば、`mydb.mytbl` ではなく、`mydb`.`mytbl` と記述します。

一時テーブル

テーブルの作成時に TEMPORARY キーワードを使用できます。TEMPORARY テーブルは現在のセッションにのみ表示され、そのセッションが閉じられると自動的に削除されます。つまり、2 つの異なるセッションが同じ一時テーブル名を使用することができ、互いに、または同じ名前の既存の TEMPORARY 以外のテーブルと競合することはありません。(既存のテーブルは、一時テーブルが削除されるまで非表示になります。)一時テーブルを作成するには、CREATE TEMPORARY TABLES 権限が必要です。

> 注記
>
> TEMPORARY キーワードを使用した場合、CREATE TABLE は、現在のアクティブなトランザクションを自動的にはコミットしません。

> 注記
>
> TEMPORARY テーブルは、データベース (スキーマ) と非常に疎な関係を持っています。データベースを削除しても、そのデータベース内で作成されたどの TEMPORARY テーブルも自動的には削除されません。また、CREATE TABLE ステートメントでテーブル名をデータベース名で修飾した場合は、存在しないデータベース内に TEMPORARY テーブルを作成することもできます。この場合は、そのテーブルへの以降のすべての参照をデータベース名で修飾する必要があります。

同じ名前を持つ既存のテーブル

キーワード IF NOT EXISTS は、テーブルがすでに存在する場合にエラーが発生しないようにします。ただし、既存のテーブルの構造が CREATE TABLE ステートメントによって示されている構造と同一であることの検証は行われません。

物理表現

MySQL は、各テーブルを、データベースディレクトリ内にある .frm テーブル形式 (定義) ファイルで表します。そのテーブルのストレージエンジンによって、ほかのファイルが作成されることもあります。

InnoDB テーブルの場合、ファイルストレージは、innodb_file_per_table 構成オプションによって制御されます。このオプションがオフになっている場合、InnoDB テーブルおよびインデックスはすべて、1 つ以上の .ibd ファイルによって表されるシステムテーブルスペースに格納されます。このオプションがオンになっているときに作成された各 InnoDB テーブルでは、テーブルデータとそれに関連付けられたすべてのインデックスは、データベースディレクトリ内にある .ibd ファイルに格納されます。

MyISAM テーブルの場合は、ストレージエンジンがデータおよびインデックスファイルを作成します。そのため、MyISAM テーブル tbl_name ごとに 3 つのディスクファイルが存在します。

ファイル	目的
tbl_name.frm	テーブル形式 (定義) ファイル
tbl_name.MYD	データファイル
tbl_name.MYI	インデックスファイル

第15章「代替ストレージエンジン」では、テーブルを表すために各ストレージエンジンがどのようなファイルを作成するかについて説明しています。テーブル名に特殊文字が含まれている場合は、セクション9.2.3「識別子とファイル名のマッピング」で説明されているように、その文字のエンコードされたバージョンがテーブルファイルの名前に含まれます。

カラムのデータ型および属性

data_type は、カラム定義内のデータ型を表します。spatial_type は、空間データ型を表します。示されているデータ型の構文は代表的な例にすぎません。カラムデータ型を指定するために使用できる構文の完全な説明や、各型のプロパティーに関する情報については、第11章「データ型」およびセクション11.5「空間データの拡張」を参照してください。

属性の中には、すべてのデータ型には適用されないものがあります。AUTO_INCREMENT は、整数型と浮動小数点型にのみ適用されます。DEFAULT は、BLOB または TEXT 型には適用されません。

- NULL と NOT NULL のどちらも指定されていない場合、そのカラムは NULL が指定されたかのように処理されます。
- 整数または浮動小数点のカラムには、追加の属性 AUTO_INCREMENT を指定できます。インデックスが設定された AUTO_INCREMENT カラムに NULL (推奨) または 0 の値を挿入すると、カラムは次のシーケンス値に設定されます。通常、これは value+1 です。ここで value は現在テーブルにあるカラムの最大値です。AUTO_INCREMENT シーケンスは 1 で始まります。

  行を挿入したあとに AUTO_INCREMENT 値を取得するには、LAST_INSERT_ID() SQL 関数または mysql_insert_id() C API 関数を使用します。セクション12.14「情報関数」およびセクション23.7.7.37「mysql_insert_id()」を参照してください。

  NO_AUTO_VALUE_ON_ZERO SQL モードが有効になっている場合は、新しいシーケンス値を生成することなく、0 を AUTO_INCREMENT カラム内に 0 として格納できます。セクション5.1.7「サーバー SQL モード」を参照してください。

  > **注記**
  >
  > テーブルごとに存在できる AUTO_INCREMENT カラムは 1 つだけです。このカラムはインデックス付きである必要があり、DEFAULT 値を割り当てることはできません。AUTO_INCREMENT カラムは、正の値だけが含まれている場合にのみ正しく機能します。負の数を挿入すると、非常に大きな正の数を挿入したと見なされます。これは、数字が正から負に「ラップする」ときの精度の問題を回避すると同時に、0 を含む AUTO_INCREMENT カラムを誤って取得してしまわないようにするために行われます。

  MyISAM テーブルの場合は、マルチカラムキー内の AUTO_INCREMENT セカンダリカラムを指定できます。セクション3.6.9「AUTO_INCREMENT の使用」を参照してください。

  MySQL を一部の ODBC アプリケーションと互換性があるようにするために、次のクエリーを使用して、最後に挿入された行の AUTO_INCREMENT 値を見つけることができます。

```
SELECT * FROM tbl_name WHERE auto_col IS NULL
```

InnoDB と AUTO_INCREMENT については、セクション14.6.5「InnoDB での AUTO_INCREMENT 処理」を参照してください。AUTO_INCREMENT と MySQL レプリケーションについては、セクション17.4.1.1「レプリケーションと AUTO_INCREMENT」を参照してください。

- 文字データ型 (CHAR、VARCHAR、TEXT) には、そのカラムの文字セットと照合順序を指定するための CHARACTER SET および COLLATE 属性を含めることができます。詳細は、セクション10.1「文字セットのサポート」を参照してください。CHARSET は CHARACTER SET のシノニムです。例:

```
CREATE TABLE t (c CHAR(20) CHARACTER SET utf8 COLLATE utf8_bin);
```

MySQL 5.6 は、文字カラム定義内の長さの指定を文字数で解釈します。(MySQL 4.1 より前のバージョンでは、バイト単位で解釈されました。)BINARY と VARBINARY の長さはバイト単位です。

- DEFAULT 句は、カラムのデフォルト値を指定します。例外が 1 つあります。デフォルト値は定数である必要があるので、関数または式にはできません。これは、たとえば日付カラムのデフォルト値に NOW() や CURRENT_DATE などの関数の値を設定できないことを意味します。例外として、TIMESTAMP または (MySQL 5.6.5 の時点では) DATETIME カラムのデフォルトとして CURRENT_TIMESTAMP を指定できることがあります。セクション11.3.5「TIMESTAMP および DATETIME の自動初期化および更新機能」を参照してください。

  カラム定義に明示的な DEFAULT 値が含まれていない場合、MySQL は、セクション11.6「データ型デフォルト値」で説明されているようにデフォルト値を決定します。

  BLOB および TEXT カラムにはデフォルト値を割り当てられません。

  NO_ZERO_DATE または NO_ZERO_IN_DATE SQL モードが有効になっているときに、日付の値のデフォルトがそのモードに従って正しくない場合、CREATE TABLE では厳密な SQL モードが有効になっていない場合は警告を、厳密モードが有効になっている場合はエラーを生成します。たとえば、NO_ZERO_IN_DATE が有効になっている場合は、c1 DATE DEFAULT '2010-00-00' によって警告が生成されます。(MySQL 5.6.6 より前は、厳密モードが有効になっていない場合でも、このステートメントはエラーを生成します。)

- COMMENT オプションを使用して、カラムのコメントを最大 1024 文字の長さで指定できます。このコメントは、SHOW CREATE TABLE および SHOW FULL COLUMNS ステートメントによって表示されます。

- MySQL Cluster では、COLUMN_FORMAT を使用して、NDB テーブルの個々のカラムのデータストレージフォーマットを指定することもできます。許可されるカラムフォーマットは、FIXED、DYNAMIC、および DEFAULT です。FIXED は固定幅のストレージを指定するために使用され、DYNAMIC はカラムが可変幅になることを許可し、DEFAULT はカラムで、そのカラムのデータ型によって決定される固定幅または可変幅のストレージが使用されるようにします (ROW_FORMAT 指定子によってオーバーライドされる可能性があります)。

  NDB テーブルの場合、COLUMN_FORMAT のデフォルト値は DEFAULT です。

  COLUMN_FORMAT は現在、NDB 以外のストレージエンジンを使用しているテーブルのカラムには影響を与えません。MySQL 5.6 以降では、COLUMN_FORMAT は暗黙のうちに無視されます。

- NDB テーブルの場合は、STORAGE 句を使用して、カラムがディスク上またはメモリー内のどちらに格納されるかを指定することもできます。STORAGE DISK を指定するとカラムはディスク上に格納され、STORAGE MEMORY を指定するとインメモリーストレージが使用されます。使用される CREATE TABLE ステートメントには、引き続き TABLESPACE 句が含まれている必要があります。

```
mysql> CREATE TABLE t1 (
    ->     c1 INT STORAGE DISK,
    ->     c2 INT STORAGE MEMORY
    -> ) ENGINE NDB;
ERROR 1005 (HY000): Can't create table 'c.t1' (errno: 140)

mysql> CREATE TABLE t1 (
    ->     c1 INT STORAGE DISK,
    ->     c2 INT STORAGE MEMORY
    -> ) TABLESPACE ts_1 ENGINE NDB;
Query OK, 0 rows affected (1.06 sec)
```

NDB テーブルの場合、STORAGE DEFAULT は STORAGE MEMORY と同等です。

STORAGE 句は、NDB 以外のストレージエンジンを使用しているテーブルには影響を与えません。STORAGE キーワードは、MySQL Cluster に付属の mysqld の構築でのみサポートされます。ほかのどのバージョンの MySQL でも認識されません。その場合は、STORAGE キーワードを使用しようとすると、必ず構文エラーが発生します。

- KEY は通常、INDEX のシノニムです。キー属性 PRIMARY KEY もまた、カラム定義内で指定する場合は、単に KEY として指定できます。これは、ほかのデータベースシステムとの互換性のために実装されました。

- UNIQUE インデックスは、そのインデックス内のすべての値が異なっている必要があるという制約を作成します。既存の行に一致するキー値を持つ新しい行を追加しようとすると、エラーが発生します。すべてのエンジンについて、UNIQUE インデックスは、NULL を含むことができるカラムでの複数の NULL 値を許可します。

- PRIMARY KEY は、すべてのキーカラムを NOT NULL として定義する必要のある一意のインデックスです。それらが NOT NULL として明示的に宣言されていない場合、MySQL は、それらを暗黙的に (かつ警告なしで) そのように宣言します。テーブルに存在できる PRIMARY KEY は 1 つだけです。PRIMARY KEY の名前は、常に PRIMARY です。そのため、これをその他のどの種類のインデックスの名前としても使用できません。

  PRIMARY KEY が存在しないときに、アプリケーションがテーブル内の PRIMARY KEY を要求した場合、MySQL は、NULL カラムのない最初の UNIQUE インデックスを PRIMARY KEY として返します。

  InnoDB テーブルでは、セカンダリインデックスのためのストレージのオーバーヘッドを最小限に抑えるために、PRIMARY KEY を短い値に維持してください。各セカンダリインデックスエントリには、対応する行の主キーカラムのコピーが含まれています。(セクション14.2.13「InnoDB テーブルおよびインデックスの構造」を参照してください。)

- 作成されたテーブルでは、PRIMARY KEY が最初に配置され、そのあとにすべての UNIQUE インデックス、さらに一意でないインデックスが続きます。これは、MySQL オプティマイザが、使用するインデックスに優先順位を付けたり、重複した UNIQUE キーをよりすばやく検出したりするのに役立ちます。

- PRIMARY KEY をマルチカラムインデックスにすることができます。ただし、カラム指定で PRIMARY KEY キー属性を使用してマルチカラムインデックスを作成することはできません。それを行なっても、その単一カラムがプライマリとしてマークされるだけです。個別の PRIMARY KEY(index_col_name, ...) 句を使用する必要があります。

- PRIMARY KEY または UNIQUE インデックスが、整数型を含む 1 つのカラムのみで構成されている場合は、SELECT ステートメントでそのカラムを _rowid として参照することもできます。

- MySQL では、PRIMARY KEY の名前は PRIMARY です。その他のインデックスでは、名前を割り当てなかった場合、そのインデックスには最初のインデックス付きカラムと同じ名前が割り当てられ、それを一意にするためにオプションのサフィクス (_2、_3、...) が付けられます。テーブルのインデックス名は、SHOW INDEX FROM tbl_name を使用して確認できます。セクション13.7.5.23「SHOW INDEX 構文」を参照してください。

- 一部のストレージエンジンでは、インデックスの作成時にインデックスタイプを指定できます。index_type 指定子の構文は、USING type_name です。

  例:

```
CREATE TABLE lookup
  (id INT, INDEX USING BTREE (id))
  ENGINE = MEMORY;
```

  USING の推奨される位置は、インデックスカラムリストのあとです。カラムリストの前にも指定できますが、このオプションをその位置で使用するためのサポートは非推奨であり、将来の MySQL リリースで削除される予定です。

  index_option 値は、インデックスの追加オプションを指定します。USING はそのようなオプションの 1 つです。許可される index_option 値の詳細は、セクション13.1.13「CREATE INDEX 構文」を参照してください。

  インデックスの詳細は、セクション8.3.1「MySQL のインデックスの使用の仕組み」を参照してください。

- MySQL 5.6 では、NULL 値を持つことができるカラム上のインデックスをサポートするのは InnoDB、MyISAM、および MEMORY だけです。それ以外の場合は、インデックス付きカラムを NOT NULL として宣言する必要があります。そうしないと、エラー結果が発生します。

- CHAR、VARCHAR、BINARY、および VARBINARY カラムの場合は、col_name(length) 構文を使用してインデックスプリフィクス長を指定することにより、カラム値の先頭の部分のみを使用するインデックスを作成できます。BLOB および TEXT カラムにもインデックスを設定できますが、プリフィクス長を指定する必要があります。プリフィクス長は、バイナリ以外の文字列型の場合は文字数で、バイナリ文字列型の場合はバイト単位で指定されます。つまり、インデックスエントリは、CHAR、VARCHAR、および TEXT カラムの場合は各カラム値の最初の length 文字、BINARY、VARBINARY、および BLOB カラムの場合は各カラム値の最初

の length バイトで構成されます。このようにカラム値のプリフィクスのみにインデックスを設定すると、インデックスファイルをはるかに小さくできます。セクション8.3.4「カラムインデックス」を参照してください。

BLOB および TEXT カラム上のインデックス設定をサポートするのは、InnoDB および MyISAM ストレージエンジンだけです。例:

```
CREATE TABLE test (blob_col BLOB, INDEX(blob_col(10)));
```

InnoDB テーブルではプリフィクスの長さを最大 767 バイトに、また innodb_large_prefix オプションが有効になっている場合は 3072 バイトにすることができます。プリフィクスの制限がバイト単位で測定されるのに対して、CREATE TABLE ステートメントでのプリフィクス長は、バイナリ以外のデータ型 (CHAR、VARCHAR、TEXT) では文字数として解釈されます。複数バイトの文字セットを使用するカラムのプリフィクス長を指定する場合は、この点を考慮に入れてください。

- index_col_name の指定を ASC または DESC で終了させることができます。これらのキーワードは、インデックス値の昇順または降順での格納を指定する将来の拡張のために許可されています。現在、これらは解析されますが、無視されます。インデックス値は、常に昇順で格納されます。
- SELECT 内でカラムに対して ORDER BY または GROUP BY を使用すると、サーバーは、max_sort_length システム変数によって示されている初期のバイト数のみを使用して値をソートします。
- 全文検索に使用される特殊な FULLTEXT インデックスを作成できます。FULLTEXT インデックスをサポートするのは、InnoDB および MyISAM だけです。これらは、CHAR、VARCHAR、および TEXT カラムからのみ作成できます。インデックス設定は常に、カラム全体に対して実行されます。カラムプリフィクスのインデックス設定はサポートされていないため、プリフィクス長が指定されてもすべて無視されます。操作の詳細は、セクション12.9「全文検索関数」を参照してください。WITH PARSER 句は、全文インデックス設定および検索操作に特殊な処理が必要な場合にパーサープラグインをインデックスに関連付けるために、index_option 値として指定できます。この句は、FULLTEXT インデックスに対してのみ有効です。プラグインの作成の詳細は、セクション24.2「MySQL プラグイン API」を参照してください。
- 空間データ型に SPATIAL インデックスを作成できます。空間型は MyISAM テーブルでのみサポートされ、インデックス付きカラムを NOT NULL として宣言する必要があります。セクション11.5「空間データの拡張」を参照してください。
- MySQL 5.6 では、インデックス定義に最大 1024 文字のオプションのコメントを含めることができます。
- InnoDB および NDB テーブルは、外部キー制約のチェックをサポートしています。参照されるテーブルのカラムには、常に明示的に名前を付ける必要があります。外部キーに対しては ON DELETE と ON UPDATE の両方のアクションがサポートされています。詳細および例については、セクション13.1.17.2「外部キー制約の使用」を参照してください。InnoDB での外部キーに固有の情報については、セクション14.6.6「InnoDB と FOREIGN KEY 制約」を参照してください。

  その他のストレージエンジンの場合、MySQL Server は、CREATE TABLE ステートメント内の FOREIGN KEY および REFERENCES 構文を解析して無視します。CHECK 句は、すべてのストレージエンジンによって解析されますが、無視されます。セクション1.7.2.4「外部キーの違い」を参照してください。

  > **重要**
  >
  > ANSI/ISO SQL 標準に精通しているユーザーの場合は、参照整合性の制約定義で使用される MATCH 句を認識または適用するストレージエンジンは (InnoDB を含め) 存在しません。明示的な MATCH 句を使用しても、指定された効果が得られないだけでなく、ON DELETE および ON UPDATE 句が無視される原因にもなります。これらの理由により、MATCH の指定は避けるようにしてください。
  >
  > SQL 標準での MATCH 句は、複合 (マルチカラム) 外部キー内の NULL 値が、主キーとの比較時にどのように処理されるかを制御します。InnoDB は基本的に、外部キーをすべてまたは部分的に NULL にすることが許可される、MATCH SIMPLE で定義されるセマンティクスを実装しています。その場合は、このような外部キーを含む (子テーブルの) 行の挿入が許可され、その行は参照される (親) テーブル内のどの行にも一致しません。トリガーを使用して、ほかのセマンティクスを実装できます。
  >
  > さらに、MySQL ではパフォーマンスのために、参照されるカラムにインデックスを設定する必要があります。ただし、参照されるカラムを UNIQUE または NOT NULL として宣言するという要件は適用されません。一意でないキーまたは NULL 値を含むキーへの外部キー参照の処理は、UPDATE や DELETE CASCADE などの操作に対し

て適切に定義されていません。UNIQUE (または PRIMARY) と NOT NULL の両方であるキーのみを参照する外部キーを使用することをお勧めします。

MySQL は、参照がカラム指定の一部として定義されている (SQL 標準で定義された)「インラインの REFERENCES 指定」を認識せず、またサポートもしていません。MySQL は、個別の FOREIGN KEY 指定の一部として指定されている場合にのみ REFERENCES 句を受け入れます。

注記

InnoDB ストレージエンジンを使用するパーティション化されたテーブルは、外部キーをサポートしていません。KEY または LINEAR KEY によってパーティション化された NDB テーブルは、この制限によって影響を受けません。詳細については、セクション19.6「パーティショニングの制約と制限」を参照してください。

- テーブルあたり 4096 カラムという強い制限値がありますが、特定のテーブルでは、実際の最大数がこれより少なくなる可能性があります。実際の最大数は、セクションD.10.4「テーブルカラム数と行サイズの制限」で説明されている要因によって異なります。

TABLESPACE および STORAGE テーブルオプションは、NDB テーブルでのみ使用されます。tablespace_name という名前のテーブルスペースが、すでに CREATE TABLESPACE を使用して作成されている必要があります。STORAGE は、使用されるストレージのタイプ (ディスクまたはメモリー) を決定するものであり、DISK、MEMORY、DEFAULT のいずれかです。

TABLESPACE ... STORAGE DISK は、MySQL Cluster ディスクデータテーブルスペースにテーブルを割り当てます。詳細は、セクション18.5.12「MySQL Cluster ディスクデータテーブル」を参照してください。

重要

STORAGE 句を、TABLESPACE 句のない CREATE TABLE ステートメントで使用することはできません。

Storage Engines (ストレージエンジン)

ENGINE テーブルオプションは、次の表に示されている名前のいずれかを使用して、テーブルのストレージエンジンを指定します。エンジン名は、引用符で囲んでも囲まなくてもかまいません。引用符で囲まれた名前 'DEFAULT' は認識されますが、無視されます。

ストレージエンジン	説明
InnoDB	行ロックと外部キーを備えたトランザクションセーフテーブル。新しいテーブルのためのデフォルトのストレージエンジン。MySQL は経験しているが、InnoDB がはじめてである場合は、第14章「InnoDB ストレージエンジン」、そのなかでも特にセクション14.1.1「デフォルトの MySQL ストレージエンジンとしての InnoDB」を参照してください。
MyISAM	主に読み取り専用または読み取りが大半のワークロードに使用される、バイナリの移植可能なストレージエンジン。セクション15.2「MyISAM ストレージエンジン」を参照してください。
MEMORY	このストレージエンジンのデータは、メモリー内にのみ格納されます。セクション15.3「MEMORY ストレージエンジン」を参照してください。
CSV	カンマ区切り値形式で行を格納するテーブル。セクション15.4「CSV ストレージエンジン」を参照してください。
ARCHIVE	アーカイブストレージエンジン。セクション15.5「ARCHIVE ストレージエンジン」を参照してください。
EXAMPLE	サンプルのエンジン。セクション15.9「EXAMPLE ストレージエンジン」を参照してください。
FEDERATED	リモートテーブルにアクセスするストレージエンジン。セクション15.8「FEDERATED ストレージエンジン」を参照してください。
HEAP	これは MEMORY のシノニムです。
MERGE	1 つのテーブルとして使用される MyISAM テーブルのコレクション。MRG_MyISAM とも呼ばれます。セクション15.7「MERGE ストレージエンジン」を参照してください。

ストレージエンジン	説明
NDB	トランザクションと外部キーをサポートする、クラスタ化された、耐障害の、メモリーベースのテーブル。NDBCLUSTER とも呼ばれます。第18章「MySQL Cluster NDB 7.3 および MySQL Cluster NDB 7.4」を参照してください。

使用できないストレージエンジンが指定されている場合、MySQL は、代わりにデフォルトのエンジンを使用します。通常、これは MyISAM です。たとえば、テーブル定義に ENGINE=INNODB オプションが含まれているが、MySQL サーバーが INNODB テーブルをサポートしていない場合、テーブルは MyISAM テーブルとして作成されます。これにより、マスター上にはトランザクションテーブルが存在するが、スレーブ上に作成されるテーブルは (高速化のために) 非トランザクションであるようなレプリケーションセットアップを行うことが可能になります。MySQL 5.6 では、ストレージエンジンの指定が受け付けられない場合は警告が発生します。

セクション5.1.7「サーバー SQL モード」で説明されているように、NO_ENGINE_SUBSTITUTION SQL モードの設定によってエンジンの置換を制御できます。

> 注記
>
> ENGINE のシノニムであった古い TYPE オプションは、MySQL 5.5 で削除されました。MySQL 5.5 以降にアップグレードする場合は、TYPE に依存する既存のアプリケーションを、代わりに ENGINE を使用するように変換する必要があります。

パフォーマンスの最適化

その他のテーブルオプションは、テーブルの動作を最適化するために使用されます。ほとんどの場合は、それらのうちのどれも指定する必要はありません。特に示されていないかぎり、これらのオプションはすべてのストレージエンジンに適用されます。特定のストレージエンジンに適用されないオプションは、テーブル定義の一部として受け入れられ、記憶される可能性があります。それにより、あとで ALTER TABLE を使用して、別のストレージエンジンを使用するようにテーブルを変換した場合に、このようなオプションが適用されます。

- AUTO_INCREMENT

  テーブルの初期の AUTO_INCREMENT 値。MySQL 5.6 では、これは MyISAM、MEMORY、InnoDB、および ARCHIVE テーブルに対して機能します。AUTO_INCREMENT テーブルオプションをサポートしていないエンジンの最初の自動インクリメント値を設定するには、テーブルを作成したあとに目的の値より 1 小さい値を持つ「ダミーの」行を挿入してから、そのダミーの行を削除します。

  CREATE TABLE ステートメント内の AUTO_INCREMENT テーブルオプションをサポートするエンジンの場合は、ALTER TABLE tbl_name AUTO_INCREMENT = N を使用して AUTO_INCREMENT 値をリセットすることもできます。この値を、現在カラム内にある最大値より小さく設定することはできません。

- AVG_ROW_LENGTH

  テーブルの平均の行の長さの近似値。これを設定する必要があるのは、可変サイズの行を持つ大きなテーブルの場合だけです。

  MyISAM テーブルを作成すると、MySQL は MAX_ROWS および AVG_ROW_LENGTH オプションの積を使用して、結果として得られるテーブルがどれくらいの大きさになるかを判定します。どちらのオプションも指定しない場合、MyISAM データおよびインデックスファイルの最大サイズは、デフォルトで 256T バイトになります。(オペレーティングシステムでその大きさのファイルがサポートされていない場合、テーブルサイズはファイルサイズ制限によって制約されます。)インデックスをより小さく、かつ高速にするためにポインタサイズを小さく維持したいと考えており、実際に大きなファイルが必要でない場合は、myisam_data_pointer_size システム変数を設定することによってデフォルトのポインタサイズを小さくすることができます。(セクション5.1.4「サーバーシステム変数」を参照してください。)すべてのテーブルをデフォルトの制限を超えて拡張できるようにしたいと考えており、テーブルが必要以上に少し遅く、かつ大きくなってもかまわない場合は、この変数を設定することによってデフォルトのポインタサイズを大きくすることができます。この値を 7 に設定すると、最大 65,536T バイトのテーブルサイズが許可されます。

- [DEFAULT] CHARACTER SET

  テーブルのデフォルトの文字セットを指定します。CHARSET は CHARACTER SET のシノニムです。文字セット名が DEFAULT である場合は、データベース文字セットが使用されます。

- CHECKSUM

  MySQL ですべての行のライブチェックサム (つまり、テーブルが変更されると MySQL が自動的に更新するチェックサム) が保持されるようにする場合は、これを 1 に設定します。これにより、テーブルの更新が少し遅

くなりますが、破損したテーブルを見つけることが容易になります。CHECKSUM TABLE ステートメントは、このチェックサムをレポートします。(MyISAM のみ。)

- [DEFAULT] COLLATE

  テーブルのデフォルトの照合順序を指定します。

- COMMENT

  テーブルのコメントであり、長さは最大 2048 文字です。

- CONNECTION

  FEDERATED テーブルの接続文字列。

  > **注記**
  >
  > 古いバージョンの MySQL は、接続文字列に COMMENT オプションを使用していました。

- DATA DIRECTORY、INDEX DIRECTORY

  InnoDB では、DATA DIRECTORY='directory' オプションを使用すると、MySQL データディレクトリ以外の場所に新しい InnoDB file-per-table テーブルスペースを作成できます。MySQL は、指定されたディレクトリ内にデータベース名に対応するサブディレクトリを作成し、さらにその中に新しいテーブルの .ibd ファイルを作成します。InnoDB テーブルで DATA DIRECTORY オプションを使用するには、innodb_file_per_table 構成オプションを有効にする必要があります。このディレクトリは、ディレクトリへの (相対パスではなく) フルパス名である必要があります。詳細は、セクション14.5.4「テーブルスペースの位置の指定」を参照してください。

  MyISAM テーブルを作成する場合は、DATA DIRECTORY='directory' 句、INDEX DIRECTORY='directory' 句、またはその両方を使用できます。これらは、それぞれ MyISAM テーブルのデータファイルとインデックスファイルを配置する場所を指定します。InnoDB テーブルとは異なり、DATA DIRECTORY または INDEX DIRECTORY オプションで MyISAM テーブルを作成する場合、MySQL はデータベース名に対応するサブディレクトリを作成しません。各ファイルは、指定されたディレクトリ内に作成されます。

  > **重要**
  >
  > テーブルレベルの DATA DIRECTORY および INDEX DIRECTORY オプションは、パーティション化されたテーブルでは無視されます。(Bug #32091)

  これらのオプションは、--skip-symbolic-links オプションを使用していない場合にのみ機能します。また、オペレーティングシステムにも、機能するスレッドに対して安全な realpath() 呼び出しが存在する必要があります。詳細は、Unix 上の MyISAM へのシンボリックリンクの使用を参照してください。

  MyISAM テーブルが DATA DIRECTORY オプションなしで作成される場合、.MYD ファイルがデータベースディレクトリ内に作成されます。デフォルトでは、MyISAM が既存の .MYD ファイルを検出した場合、そのファイルを上書きします。INDEX DIRECTORY オプションを指定せずに作成されたテーブルについて、.MYI ファイルに同じことが当てはまります。この動作を抑制するには、--keep_files_on_create オプションを使用してサーバーを起動します。その場合、MyISAM は既存のファイルを上書きせず、代わりにエラーを返します。

  MyISAM テーブルが DATA DIRECTORY または INDEX DIRECTORY オプションを使用して作成され、既存の .MYD または .MYI ファイルが見つかった場合、MyISAM は常にエラーを返します。指定されたディレクトリ内のファイルは上書きされません。

  > **重要**
  >
  > DATA DIRECTORY または INDEX DIRECTORY では、MySQL データディレクトリを含むパス名を使用できません。これには、パーティション化されたテーブルや個々のテーブルパーティションが含まれます。(Bug #32167 を参照してください。)

- DELAY_KEY_WRITE

  テーブルのキー更新をテーブルが閉じられるまで遅らせる場合は、これを 1 に設定します。セクション5.1.4「サーバーシステム変数」にある delay_key_write システム変数の説明を参照してください。(MyISAM のみ。)

- INSERT_METHOD

  MERGE テーブルにデータを挿入する場合は、INSERT_METHOD を使用して、行を挿入するテーブルを指定する必要があります。INSERT_METHOD は、MERGE テーブルにのみ役立つオプションです。最初または最後のテーブルに挿入するには FIRST または LAST の値を、挿入されないようにするには NO の値を使用します。セクション15.7「MERGE ストレージエンジン」を参照してください。

- KEY_BLOCK_SIZE

  圧縮された InnoDB テーブルでは、オプションで、ページに使用するサイズをバイト単位で指定します。この値はヒントとして扱われます。InnoDB では、必要に応じて異なるサイズが使用される可能性があります。値 0 は、innodb_page_size 値の半分であるデフォルトの圧縮済みページサイズを表します。KEY_BLOCK_SIZE 値は、innodb_page_size 値以下にしかできません。innodb_page_size 値を超える値を指定した場合は、その値が無視され、警告が発行されます。また、KEY_BLOCK_SIZE は innodb_page_size 値の半分に設定されます。innodb_strict_mode=ON の場合、無効な KEY_BLOCK_SIZE 値を指定するとエラーが返されます。使用法の詳細は、セクション14.7「InnoDB 圧縮テーブル」を参照してください。

  個々のインデックス定義では、テーブルの値をオーバーライドする独自の KEY_BLOCK_SIZE 値を指定できます。

  > **注記**
  >
  > InnoDB テーブルに対して KEY_BLOCK_SIZE 句を使用している場合は、innodb_strict_mode を有効にすることをお勧めします。

- MAX_ROWS

  テーブル内に格納することを予定している行の最大数。これは強い制限値ではなく、どちらかと言うと、テーブルが少なくともこの行数を格納できる必要があるという、ストレージエンジンへのヒントです。

  NDB ストレージエンジンは、この値を最大値として扱います。非常に大きな (数百万行を含む) MySQL Cluster テーブルを作成する予定がある場合は、このオプションを使用して MAX_ROWS = 2 * rows を設定することにより、テーブルの主キーのハッシュを格納するために使用されるハッシュテーブル内に NDB によって十分な数のインデックススロットが割り当てられることを保証するようにしてください。ここで、rows はテーブルに挿入することが予測される行数です。

  MAX_ROWS の最大値は 4294967295 です。これを超える値は、この制限に切り捨てられます。

- MIN_ROWS

  テーブル内に格納することを予定している行の最小数。MEMORY ストレージエンジンは、このオプションをメモリー使用に関するヒントとして使用します。

- PACK_KEYS

  PACK_KEYS は、MyISAM テーブルでのみ有効になります。インデックスを小さくする場合は、このオプションを 1 に設定します。通常は、これによって更新は遅く、読み取りは高速になります。このオプションを 0 に設定すると、キーのすべてのパッキングが無効になります。これを DEFAULT に設定すると、長い CHAR、VARCHAR、BINARY、または VARBINARY カラムのみをパックするようストレージエンジンに指示します。

  PACK_KEYS を使用しない場合、デフォルトでは文字列をパックしますが、数値はパックしません。PACK_KEYS=1 を使用した場合は、数値もパックされます。

  2 進数のキーをパックする場合、MySQL は次のプリフィクス圧縮を使用します。

  - 前のキーの何バイトが次のキーと同じであるかを示すために、すべてのキーに 1 バイトが余分に必要になります。
  - 行へのポインタは、圧縮率を向上させるために、キーの直後に高位バイトが先に来る順序で格納されます。

  つまり、2 つの連続した行に等しいキーが多数存在する場合は、次の「同じ」キーはすべて、通常 (行へのポインタを含め) 2 バイトしか占有しません。これを、次のキーが storage_size_for_key + pointer_size (ここで、ポインタサイズは通常 4) を占有する通常のケースと比較してください。逆に言うと、プリフィクス圧縮から大きな利点が得られるのは、同じ数値が多数存在する場合だけです。すべてのキーが完全に異なっている場合は、そのキーが NULL 値を持つことができるキーでないかぎり、キーあたり 1 バイト多く使用されます。(この場合、パックされたキーの長さは、キーが NULL であるかどうかをマークするために使用されるのと同じバイトに格納されます。)

- PASSWORD

  このオプションは使用されません。.frm ファイルを暗号化し、ほかのどの MySQL サーバーからも使用できないようにする必要がある場合は、当社の販売部門に問い合わせてください。

- ROW_FORMAT

  行が格納される物理フォーマットを定義します。これらの選択は、テーブルに使用されているストレージエンジンによって異なります。

  InnoDB テーブルの場合:

  - デフォルトでは、行は圧縮形式 (ROW_FORMAT=COMPACT) で格納されます。
  - 古いバージョンの MySQL で使用されていた非圧縮形式は、ROW_FORMAT=REDUNDANT を指定することによって引き続き要求できます。
  - InnoDB テーブルの圧縮を有効にするには、ROW_FORMAT=COMPRESSED を指定し、セクション14.7「InnoDB 圧縮テーブル」の手順に従います。
  - データ型 (特に BLOB 型) の InnoDB ストレージの効率を向上させるには、ROW_FORMAT=DYNAMIC を指定し、セクション14.9.3「DYNAMIC および COMPRESSED 行フォーマット」の手順に従います。COMPRESSED および DYNAMIC 行フォーマットはどちらも、構成設定 innodb_file_per_table=1 および innodb_file_format=barracuda を使用してテーブルを作成する必要があります。
  - デフォルト以外の ROW_FORMAT 句を指定する場合は、innodb_strict_mode 構成オプションも有効にすることを考慮してください。
  - InnoDB 行フォーマットの詳細は、セクション14.9「InnoDB の行ストレージと行フォーマット」を参照してください。

  MyISAM テーブルの場合は、このオプション値を、静的行フォーマットまたは可変長行フォーマットを示す FIXED または DYNAMIC に設定できます。myisampack は、この型を COMPRESSED に設定します。セクション15.2.3「MyISAM テーブルのストレージフォーマット」を参照してください。

  > 注記
  >
  > CREATE TABLE ステートメントを実行するとき、テーブルに使用しているストレージエンジンでサポートされていない行フォーマットを指定した場合、テーブルはそのストレージエンジンのデフォルトの行フォーマットを使用して作成されます。SHOW TABLE STATUS に応答してこのカラムでレポートされる情報は、使用されている実際の行フォーマットです。作成中は元の CREATE TABLE 定義が保持されているため、これは Create_options カラム内の値とは異なる可能性があります。

- STATS_AUTO_RECALC

  InnoDB テーブルの永続的統計を自動的に再計算するかどうかを指定します。値 DEFAULT を指定すると、テーブルの永続的統計設定は innodb_stats_auto_recalc 構成オプションによって決定されます。値 1 を指定すると、統計は、テーブル内のデータの 10% が変更されたときに再計算されます。値 0 は、このテーブルの自動再計算が行われないようにします。この設定の場合、テーブルへの大幅な変更を行なったあとに統計を再計算するには、ANALYZE TABLE ステートメントを発行します。永続的統計機能の詳細は、セクション14.13.16.1「永続的オプティマイザ統計のパラメータの構成」を参照してください。

- STATS_PERSISTENT

  InnoDB テーブルの永続的統計を有効にするかどうかを指定します。値 DEFAULT を指定すると、テーブルの永続的統計設定は innodb_stats_persistent 構成オプションによって決定されます。値 1 がテーブルの永続的統計を有効にするのに対して、値 0 はこの機能を無効にします。CREATE TABLE または ALTER TABLE ステートメントを使用して永続的統計を有効にしたあと、代表的なデータのテーブルへのロード後に統計を計算するには、ANALYZE TABLE ステートメントを発行します。永続的統計機能の詳細は、セクション14.13.16.1「永続的オプティマイザ統計のパラメータの構成」を参照してください。

- STATS_SAMPLE_PAGES

  インデックス付きカラムのカーディナリティーやその他の統計 (ANALYZE TABLE によって計算される統計など) を推定するときにサンプリングするインデックスページの数。詳細は、セクション14.13.16.1「永続的オプティマイザ統計のパラメータの構成」を参照してください。

- UNION

  UNION は、同一の MyISAM テーブルのコレクションを 1 つのものとしてアクセスする場合に使用されます。これは、MERGE テーブルでのみ機能します。セクション15.7「MERGE ストレージエンジン」を参照してください。

  MERGE テーブルにマップするテーブルに対する SELECT、UPDATE、および DELETE 権限が必要です。

  > **注記**
  >
  > 以前は、使用されるすべてのテーブルが MERGE テーブル自体と同じデータベース内に存在する必要がありました。この制限は適用されなくなりました。

パーティション化

partition_options を使用すると、CREATE TABLE で作成されたテーブルのパーティション化を制御できます。

> **重要**
>
> このセクションの最初にある partition_options の構文に示されているすべてのオプションが、すべてのパーティショニングタイプに使用できるわけではありません。各タイプに固有の情報については、次の個々のタイプのリストを参照してください。また、MySQL でのパーティション化の動作や使用に関するより詳細な情報、および MySQL のパーティション化に関連したテーブル作成やその他のステートメントの追加の例については、第19章「パーティション化」を参照してください。

partition_options 句が使用される場合、この句は PARTITION BY で始まります。この句には、パーティションを決定するために使用される関数が含まれています。この関数は、1 から num までの範囲の整数値を返します。ここで、num はパーティションの数です。(テーブルに含めることのできるユーザー定義パーティションの最大数は 1024 です。この最大数には、このセクションのあとの方で説明されているサブパーティションの数が含まれています。)MySQL 5.6 で、この関数に使用可能な選択肢を次のリストに示します。

- HASH(expr): 行の配置および検索のためのキーを作成するために 1 つ以上のカラムをハッシュします。expr は、1 つ以上のテーブルカラムを使用した式です。これは、1 つの整数値が得られる任意の有効な MySQL 式 (MySQL 関数を含む) にすることができます。たとえば、次はどちらも、PARTITION BY HASH を使用した有効な CREATE TABLE ステートメントです。

  ```
  CREATE TABLE t1 (col1 INT, col2 CHAR(5))
    PARTITION BY HASH(col1);

  CREATE TABLE t1 (col1 INT, col2 CHAR(5), col3 DATETIME)
    PARTITION BY HASH ( YEAR(col3) );
  ```

  PARTITION BY HASH では、VALUES LESS THAN または VALUES IN のどちらの句も使用できません。

  PARTITION BY HASH は、expr をパーティションの数で割った余り (つまり、法) を使用します。例および追加情報については、セクション19.2.4「HASH パーティショニング」を参照してください。

  LINEAR キーワードには、いくぶん異なるアルゴリズムが必要になります。この場合、行が格納されるパーティションの数は、1 つ以上の論理的な AND 演算の結果として計算されます。線形ハッシュの説明および例については、セクション19.2.4.1「LINEAR HASH パーティショニング」を参照してください。

- KEY(column_list): これは HASH に似ていますが、均一なデータ分散を保証するために MySQL がハッシュ関数を提供する点が異なります。column_list 引数は、単純に 1 つ以上のテーブルカラム (最大 16 個) のリストです。この例は、4 つのパーティションを持つ、キーによってパーティション化された単純なテーブルを示しています。

  ```
  CREATE TABLE tk (col1 INT, col2 CHAR(5), col3 DATE)
    PARTITION BY KEY(col3)
    PARTITIONS 4;
  ```

  キーによってパーティション化されたテーブルの場合は、LINEAR キーワードを使用して線形パーティション化を採用できます。これには、HASH によってパーティション化されたテーブルの場合と同じ効果があります。つまり、パーティション番号は法ではなく、& 演算子を使用して見つけられます (詳細は、セクション19.2.4.1「LINEAR HASH パーティショニング」およびセクション19.2.5「KEY パーティショニング」を参照してください)。この例では、キーによる線形パーティション化を使用して 5 つのパーティション間でデータを分散させます。

  ```
  CREATE TABLE tk (col1 INT, col2 CHAR(5), col3 DATE)
    PARTITION BY LINEAR KEY(col3)
  ```

```
    PARTITIONS 5;
```

ALGORITHM={1|2} オプションは、MySQL 5.6.11 から [SUB]PARTITION BY [LINEAR] KEY でサポートされています。ALGORITHM=1 を指定すると、サーバーは MySQL 5.1 と同じキーハッシュ関数を使用します。ALGORITHM=2 は、サーバーが、MySQL 5.5 以降で実装され、KEY によってパーティション化された新しいテーブルに対してデフォルトで使用されるキーハッシュ関数を採用することを示します。(MySQL 5.5 以降で採用されたキーハッシュ関数によって作成されたパーティション化されたテーブルを MySQL 5.1 サーバーで使用することはできません。)このオプションを指定しない場合は、ALGORITHM=2 を使用するのと同じ効果があります。このオプションは、主に [LINEAR] KEY によってパーティション化されたテーブルを MySQL 5.1 以降の MySQL バージョン間でアップグレードまたはダウングレードするときに使用するか、または MySQL 5.5 以降のサーバー上で、MySQL 5.1 サーバー上で使用できる KEY または LINEAR KEY によってパーティション化されたテーブルを作成することを目的にしています。詳細は、セクション13.1.7.1「ALTER TABLE パーティション操作」を参照してください。

MySQL 5.6.11 以降の mysqldump は、このオプションをバージョン管理されたコメント内に次のように書き込みます。

```
CREATE TABLE t1 (a INT)
/*!50100 PARTITION BY KEY */ /*!50611 ALGORITHM = 1 */ /*!50100 ()
    PARTITIONS 3 */
```

これにより、MySQL 5.6.10 以前のサーバーはこのオプションを無視するようになります。これらのバージョンでは、通常であれば構文エラーが発生します。KEY によってパーティション化またはサブパーティション化されたテーブルを使用している MySQL 5.5.31 またはそれ以降の MySQL 5.5 サーバー上で作成されたダンプを、バージョン 5.6.11 より前の MySQL 5.6 サーバーにロードする予定がある場合は、続行する前に、必ずセクション2.11.1.3「MySQL 5.5 から 5.6 へのアップグレード」を参照するようにしてください。(そこで見つかった情報は、MySQL 5.6.11 以降のサーバーから作成された KEY によってパーティション化またはサブパーティション化されたテーブルを含むダンプを、MySQL 5.5.30 以前のサーバーにロードする場合にも適用されます。)

また、MySQL 5.6.11 以降では、ALGORITHM=1 が mysqldump と同じ方法で、バージョン管理されたコメントを使用して SHOW CREATE TABLE の出力に必要に応じて表示されます。ALGORITHM=2 は、元のテーブルを作成するときにこのオプションが指定された場合でも、SHOW CREATE TABLE の出力から常に省略されます。

PARTITION BY KEY では、VALUES LESS THAN または VALUES IN のどちらの句も使用できません。

- RANGE(expr): この場合、expr は、VALUES LESS THAN 演算子のセットを使用して値の範囲を示します。範囲のパーティション化を使用する場合は、VALUES LESS THAN を使用して、少なくとも 1 つのパーティションを定義する必要があります。範囲のパーティション化では VALUES IN を使用できません。

  > **注記**
  >
  > RANGE によってパーティション化されたテーブルでは、VALUES LESS THAN を整数リテラル値、または 1 つの整数値に評価される式のどちらかとともに使用する必要があります。MySQL 5.6 では、このセクションのあとの方で説明されている、PARTITION BY RANGE COLUMNS を使用して定義されたテーブルでこの制限を克服できます。

次のスキームに従って、年の値を含むカラムに関してパーティション化するテーブルがあるとします。

パーティション番号:	年の範囲:
0	1990 以前
1	1991 から 1994 まで
2	1995 から 1998 まで
3	1999 から 2002 まで
4	2003 から 2005 まで
5	2006 以降

このようなパーティション化スキームを実装するテーブルは、次に示す CREATE TABLE ステートメントによって実現できます。

```
CREATE TABLE t1 (
    year_col  INT,
    some_data INT
```

```
)
PARTITION BY RANGE (year_col) (
    PARTITION p0 VALUES LESS THAN (1991),
    PARTITION p1 VALUES LESS THAN (1995),
    PARTITION p2 VALUES LESS THAN (1999),
    PARTITION p3 VALUES LESS THAN (2002),
    PARTITION p4 VALUES LESS THAN (2006),
    PARTITION p5 VALUES LESS THAN MAXVALUE
);
```

PARTITION ... VALUES LESS THAN ... ステートメントは、連続的に機能します。VALUES LESS THAN MAXVALUE は、それ以外で指定されている最大値より大きい「残りの」値を指定するように機能します。

VALUES LESS THAN 句は (C、Java、PHP などの多くのプログラミング言語に見られるような) switch ... case ブロックの case 部分と同様の方法で連続的に機能します。つまり、この句は、連続した各 VALUES LESS THAN で指定されている上限が前の句の上限より大きく、かつ MAXVALUE を参照している句がリスト内のすべての句の最後に来るような方法で配置されている必要があります。

- RANGE COLUMNS(column_list): RANGE に対するこのバリアントは、複数のカラムに関する範囲条件を使用した (つまり、WHERE a = 1 AND b < 10 や WHERE a = 1 AND b = 10 AND c < 10 などの条件を持つ) クエリーに対するパーティションプルーニングを容易にします。これにより、COLUMNS 句内のカラムのリストと、各 PARTITION ... VALUES LESS THAN (value_list) パーティション定義句内のカラム値のセットを使用して、複数のカラム内の値の範囲を指定できるようになります。(もっとも単純なケースでは、このセットは単一カラムで構成されます。)column_list および value_list で参照できるカラムの最大数は 16 です。

  COLUMNS 句で使用される column_list には、カラムの名前のみを含めることができます。リスト内の各カラムは MySQL のデータ型のうち、整数型、文字列型、および時間または日付カラム型のいずれかである必要があります。BLOB、TEXT、SET、ENUM、BIT、または空間データ型を使用したカラムは許可されていません。浮動小数点数型を使用するカラムも許可されていません。また、COLUMNS 句では、関数や演算式も使用できません。

  パーティション定義で使用される VALUES LESS THAN 句は、COLUMNS() 句に現れるカラムごとにリテラル値を指定する必要があります。つまり、各 VALUES LESS THAN 句で使用される値のリストには、COLUMNS 句にリストされているカラムの数と同じ数の値が含まれている必要があります。VALUES LESS THAN 句で COLUMNS 句に存在する数より多いか、または少ない値を使用しようとすると、このステートメントはエラー Inconsistency in usage of column lists for partitioning... で失敗します。VALUES LESS THAN に現れるどの値にも NULL は使用できません。この例に示すように、最初のカラム以外の特定のカラムで MAXVALUE を複数回使用できます。

  ```
  CREATE TABLE rc (
      a INT NOT NULL,
      b INT NOT NULL
  )
  PARTITION BY RANGE COLUMNS(a,b) (
      PARTITION p0 VALUES LESS THAN (10,5),
      PARTITION p1 VALUES LESS THAN (20,10),
      PARTITION p2 VALUES LESS THAN (MAXVALUE,15),
      PARTITION p3 VALUES LESS THAN (MAXVALUE,MAXVALUE)
  );
  ```

  VALUES LESS THAN 値リストで使用されている各値が、対応するカラムの型に正確に一致している必要があります。変換は行われません。たとえば、整数型を使用するカラムに一致する値として文字列 '1' を使用したり (代わりに、数値 1 を使用する必要があります)、文字列型を使用するカラムに一致する値として数値 1 を使用したりすることはできません (このような場合は、引用符で囲まれた文字列 '1' を使用する必要があります)。

  詳細は、セクション19.2.1「RANGE パーティショニング」およびセクション19.4「パーティションプルーニング」を参照してください。

- LIST(expr): これは、州または国コードなどの、制限された指定可能な値のセットを持つテーブルカラムに基づいてパーティションを割り当てる場合に役立ちます。このような場合は、特定の州または国に関連するすべての行を単一パーティションに割り当てたり、特定の州または国のセットのためにパーティションを予約したりできます。これは RANGE に似ていますが、各パーティションに許可される値を指定するために VALUES IN しか使用できない点が異なります。

  VALUES IN は、一致させる値のリストとともに使用されます。たとえば、次のようなパーティション化スキームを作成できます。

  ```
  CREATE TABLE client_firms (
      id   INT,
      name VARCHAR(35)
  )
  ```

```
PARTITION BY LIST (id) (
    PARTITION r0 VALUES IN (1, 5, 9, 13, 17, 21),
    PARTITION r1 VALUES IN (2, 6, 10, 14, 18, 22),
    PARTITION r2 VALUES IN (3, 7, 11, 15, 19, 23),
    PARTITION r3 VALUES IN (4, 8, 12, 16, 20, 24)
);
```

リストのパーティション化を使用する場合は、VALUES IN を使用して、少なくとも 1 つのパーティションを定義する必要があります。PARTITION BY LIST では VALUES LESS THAN を使用できません。

> 注記
>
> LIST によってパーティション化されたテーブルでは、VALUES IN で使用される値リストを整数値のみで構成する必要があります。MySQL 5.6 では、このセクションのあとの方で説明されている、LIST COLUMNS によるパーティション化を使用してこの制限を克服できます。

- LIST COLUMNS(column_list): LIST に対するこのバリアントは、複数のカラムに関する比較条件を使用した (つまり、WHERE a = 5 AND b = 5 や WHERE a = 1 AND b = 10 AND c = 5 などの条件を持つ) クエリーに対するパーティションプルーニングを容易にします。これにより、COLUMNS 句内のカラムのリストと、各 PARTITION ... VALUES IN (value_list) パーティション定義句内のカラム値のセットを使用して、複数のカラム内の値を指定できるようになります。

  LIST COLUMNS(column_list) で使用されるカラムリストと VALUES IN(value_list) で使用される値リストに関連したデータ型を管理するルールは、VALUES IN 句では MAXVALUE が許可されず、NULL を使用できる点を除き、それぞれ RANGE COLUMNS(column_list) で使用されるカラムリストと VALUES LESS THAN(value_list) で使用される値リストの場合のルールと同じです。

  PARTITION BY LIST COLUMNS で VALUES IN に使用される値のリストには、PARTITION BY LIST で使用された場合と比較して重要な違いが 1 つあります。PARTITION BY LIST COLUMNS で使用された場合、VALUES IN 句内の各要素は、カラム値のセットである必要があります。各セット内の値の数は COLUMNS 句で使用されているカラム数と同じである必要があり、これらの値のデータ型はそれらのカラムのデータ型に一致している (しかも、同じ順序で現れる) 必要があります。もっとも単純なケースでは、このセットは単一カラムで構成されます。column_list および value_list を構成する各要素で使用できるカラムの最大数は 16 です。

  次の CREATE TABLE ステートメントで定義されるテーブルは、LIST COLUMNS パーティション化を使用したテーブルの例を示しています。

```
CREATE TABLE lc (
    a INT NULL,
    b INT NULL
)
PARTITION BY LIST COLUMNS(a,b) (
    PARTITION p0 VALUES IN( (0,0), (NULL,NULL) ),
    PARTITION p1 VALUES IN( (0,1), (0,2), (0,3), (1,1), (1,2) ),
    PARTITION p2 VALUES IN( (1,0), (2,0), (2,1), (3,0), (3,1) ),
    PARTITION p3 VALUES IN( (1,3), (2,2), (2,3), (3,2), (3,3) )
);
```

- オプションで、PARTITIONS num 句を使用してパーティションの数を指定できます。ここで、num はパーティションの数です。この句とほかのいずれかの PARTITION 句の両方が使用されている場合、num は、PARTITION 句を使用して宣言されているすべてのパーティションの総数と同じである必要があります。

  > 注記
  >
  > RANGE または LIST によってパーティション化されたテーブルの作成で PARTITIONS 句を使用するかどうかにかかわらず、テーブル定義には引き続き、少なくとも 1 つの PARTITION VALUES 句を含める必要があります (下を参照してください)。

- オプションで、パーティションを複数のサブパーティションに分割できます。これは、オプションの SUBPARTITION BY 句を使用して示すことができます。サブパーティション化は、HASH または KEY によって実行できます。これらのどちらも LINEAR にすることができます。これらは、同等のパーティショニングタイプについて前に説明したのと同じように機能します。(LIST または RANGE によってサブパーティション化することはできません。)

  サブパーティションの数は、SUBPARTITIONS キーワードと、そのあとの整数値を使用して示すことができます。

- PARTITIONS または SUBPARTITIONS 句で使用されている値の厳密なチェックが適用され、この値は次のルールに従っている必要があります。
  - この値は 0 以外の正の整数である必要があります。
  - 先頭の 0 は許可されません。
  - この値は整数リテラルである必要があり、式にすることはできません。たとえば、0.2E+01 が 2 に評価されたとしても、PARTITIONS 0.2E+01 は許可されません。(Bug #15890)

    > **注記**
    >
    > PARTITION BY 句で使用される式 (expr) は、作成されているテーブルにはないどのカラムも参照できません。このような参照は明確に禁止されており、そのステートメントがエラーで失敗する原因になります。(Bug #29444)

各パーティションは、partition_definition 句を使用して個別に定義できます。この句を構成する個別の部分は次のとおりです。

- PARTITION partition_name: これはパーティションの論理名を指定します。
- VALUES 句: 範囲のパーティション化では、各パーティションに VALUES LESS THAN 句が含まれている必要があります。リストのパーティション化では、パーティションごとに VALUES IN 句を指定する必要があります。これは、このパーティションにどの行を格納するかを決定するために使用されます。構文の例については、第19章「パーティション化」にあるパーティショニングタイプの説明を参照してください。
- オプションの COMMENT 句を使用すると、このパーティションを説明する文字列を指定できます。例:

  ```
  COMMENT = 'Data for the years previous to 1999'
  ```

  MySQL 5.6.6 から、パーティションのコメントの最大長は 1024 文字です。(以前は、この制限が明示的に定義されていませんでした。)
- DATA DIRECTORY と INDEX DIRECTORY は、それぞれ、このパーティションのデータとインデックスが格納されるディレクトリを示すために使用できます。data_dir と index_dir はどちらも、絶対システムパス名である必要があります。例:

  ```
  CREATE TABLE th (id INT, name VARCHAR(30), adate DATE)
  PARTITION BY LIST(YEAR(adate))
  (
    PARTITION p1999 VALUES IN (1995, 1999, 2003)
      DATA DIRECTORY = '/var/appdata/95/data'
      INDEX DIRECTORY = '/var/appdata/95/idx',
    PARTITION p2000 VALUES IN (1996, 2000, 2004)
      DATA DIRECTORY = '/var/appdata/96/data'
      INDEX DIRECTORY = '/var/appdata/96/idx',
    PARTITION p2001 VALUES IN (1997, 2001, 2005)
      DATA DIRECTORY = '/var/appdata/97/data'
      INDEX DIRECTORY = '/var/appdata/97/idx',
    PARTITION p2002 VALUES IN (1998, 2002, 2006)
      DATA DIRECTORY = '/var/appdata/98/data'
      INDEX DIRECTORY = '/var/appdata/98/idx'
  );
  ```

  DATA DIRECTORY と INDEX DIRECTORY は、MyISAM テーブルに使用される CREATE TABLE ステートメントの table_option 句と同じように動作します。

  パーティションごとに 1 つのデータディレクトリと 1 つのインデックスディレクトリを指定できます。指定されないままになっている場合、データとインデックスは、デフォルトではそのテーブルのデータベースディレクトリ内に格納されます。

  Windows では、DATA DIRECTORY および INDEX DIRECTORY オプションは MyISAM テーブルの個々のパーティションまたはサブパーティションに対してサポートされず、INDEX DIRECTORY オプションは InnoDB テーブルの個々のパーティションまたはサブパーティションに対してサポートされません。これらのオプションは、警告が生成される点を除き、Windows では無視されます。(Bug #30459)

  > **注記**
  >
  > DATA DIRECTORY および INDEX DIRECTORY オプションは、NO_DIR_IN_CREATE が有効になっている場合、パーティション化されたテーブルの作成では無視されます。(Bug #24633)

- MAX_ROWS と MIN_ROWS は、それぞれ、このパーティションに格納される行の最大数と最小数を指定するために使用できます。max_number_of_rows と min_number_of_rows の値は正の整数である必要があります。同じ名前を持つテーブルレベルのオプションと同様に、これらはサーバーへの「提案」としてのみ機能し、強い制限値ではありません。

- オプションの TABLESPACE 句を使用すると、このパーティションのテーブルスペースを指定できます。MySQL Cluster にのみ使用されます。

- パーティション化ハンドラは、PARTITION と SUBPARTITION の両方について [STORAGE] ENGINE オプションを受け入れます。現在、これを使用するには、すべてのパーティションまたはすべてのサブパーティションを同じストレージエンジンに設定するしか方法がなく、同じテーブル内のパーティションまたはサブパーティションに対して異なるストレージエンジンを設定しようとするとエラー ERROR 1469 (HY000): The mix of handlers in the partitions is not permitted in this version of MySQL が発生します。将来の MySQL リリースでは、このパーティション化に関する制限を解消する予定です。

- オプションで、パーティション定義に 1 つ以上の subpartition_definition 句を含めることができます。これらの各句は、少なくとも SUBPARTITION name で構成されます。ここで、name はそのサブパーティションの識別子です。PARTITION キーワードが SUBPARTITION に置き換えられる点を除き、サブパーティション定義の構文はパーティション定義の構文と同じです。

  サブパーティション化は HASH または KEY によって実行する必要があり、RANGE または LIST パーティションに対してのみ実行できます。セクション19.2.6「サブパーティショニング」を参照してください。

パーティションに対しては変更、マージ、テーブルへの追加、およびテーブルからの削除が可能です。これらのタスクを実行するための MySQL ステートメントに関する基本情報については、セクション13.1.7「ALTER TABLE 構文」を参照してください。より詳細な説明および例については、セクション19.3「パーティション管理」を参照してください。

> **重要**
>
> 元の CREATE TABLE ステートメント (すべての指定およびテーブルオプションを含む) は、そのテーブルが作成されるときに MySQL によって格納されます。この情報が保持されるのは、ALTER TABLE ステートメントを使用してストレージエンジン、照合順序、またはその他の設定を変更した場合に、指定された元のテーブルオプションが保持されるようにするためです。これにより、2 つのエンジンによってサポートされる行フォーマットが異なっていても、InnoDB と MyISAM のテーブルタイプ間での変更が可能になります。
>
> 元のステートメントのテキストは保持されますが、特定の値やオプション (ROW_FORMAT など) が暗黙のうちに再構成される可能性があるため、アクティブなテーブル定義 (DESCRIBE または SHOW TABLE STATUS によってアクセス可能) やテーブル作成文字列 (SHOW CREATE TABLE によってアクセス可能) は異なる値をレポートします。

テーブルのクローニングまたはコピー

CREATE TABLE ステートメントの最後に SELECT ステートメントを追加することによって、あるテーブルを別のテーブルから作成できます。

```
CREATE TABLE new_tbl SELECT * FROM orig_tbl;
```

詳細は、セクション13.1.17.1「CREATE TABLE ... SELECT 構文」を参照してください。

別のテーブルの定義 (元のテーブルで定義されているすべてのカラム属性やインデックスを含む) に基づいて空のテーブルを作成するには、LIKE を使用します。

```
CREATE TABLE new_tbl LIKE orig_tbl;
```

このコピーは、元のテーブルと同じバージョンのテーブルストレージフォーマットを使用して作成されます。元のテーブルに対する SELECT 権限が必要です。

LIKE は、ビューに対してではなく、ベーステーブルに対してのみ機能します。

> **重要**
>
> MySQL 5.6.1 から、LOCK TABLES ステートメントが有効になっている間は CREATE TABLE または CREATE TABLE ... LIKE を実行できません。

> また MySQL 5.6.1 の時点では、CREATE TABLE ... LIKE は CREATE TABLE と同じチェックを行い、単に .frm ファイルをコピーするだけではありません。つまり、現在の SQL モードが、元のテーブルが作成されたときの有効なモードとは異なっている場合、テーブル定義が新しいモードでは無効と見なされる可能性があり、ステートメントは失敗します。

CREATE TABLE ... LIKE は、元のテーブルや、すべての外部キー定義に対して指定されたどの DATA DIRECTORY または INDEX DIRECTORY テーブルオプションも保持しません。

元のテーブルが TEMPORARY テーブルである場合、CREATE TABLE ... LIKE は TEMPORARY を保持しません。TEMPORARY 宛先テーブルを作成するには、CREATE TEMPORARY TABLE ... LIKE を使用します。

13.1.17.1 CREATE TABLE ... SELECT 構文

CREATE TABLE ステートメントの最後に SELECT ステートメントを追加することによって、あるテーブルを別のテーブルから作成できます。

```
CREATE TABLE new_tbl [AS] SELECT * FROM orig_tbl;
```

MySQL は、SELECT 内のすべての要素に対して新しいカラムを作成します。例:

```
mysql> CREATE TABLE test (a INT NOT NULL AUTO_INCREMENT,
    ->       PRIMARY KEY (a), KEY(b))
    ->       ENGINE=MyISAM SELECT b,c FROM test2;
```

これにより、a、b、c の 3 つのカラムを含む MyISAM テーブルが作成されます。ENGINE オプションは CREATE TABLE ステートメントの一部であるため、SELECT のあとに使用してはいけません。これにより、構文エラーが発生します。CHARSET などのその他の CREATE TABLE オプションにも同じことが当てはまります。

SELECT ステートメントからのカラムは、テーブルにオーバーラップされるのではなく、テーブルの右側に付加されます。次の例を考えてみます。

```
mysql> SELECT * FROM foo;
+---+
| n |
+---+
| 1 |
+---+

mysql> CREATE TABLE bar (m INT) SELECT n FROM foo;
Query OK, 1 row affected (0.02 sec)
Records: 1  Duplicates: 0  Warnings: 0

mysql> SELECT * FROM bar;
+------+---+
| m    | n |
+------+---+
| NULL | 1 |
+------+---+
1 row in set (0.00 sec)
```

テーブル foo 内の行ごとに、foo からの値と新しいカラムのデフォルト値を持つ行が bar 内に挿入されます。

CREATE TABLE ... SELECT の結果として得られるテーブルでは、CREATE TABLE 部分でのみ指定されているカラムが最初に来ます。両方の部分で指定されているカラム、または SELECT 部分でのみ指定されているカラムがそのあとに来ます。SELECT カラムのデータ型は、CREATE TABLE 部分にあるカラムも指定することによってオーバーライドできます。

テーブルへのデータのコピー中にエラーが発生した場合、そのデータは自動的に削除され、作成されません。

一意のキー値を複製する行を処理する方法を示すために、SELECT の前に IGNORE または REPLACE を指定できます。IGNORE を指定すると、一意のキー値に関して既存の行を複製する行は破棄されます。REPLACE を指定すると、新しい行によって同じ一意のキー値を持つ行が置き換えられます。IGNORE と REPLACE のどちらも指定されていない場合は、重複した一意のキー値によってエラーが発生します。

ベースとなる SELECT ステートメント内の行の順序を常に特定することはできないため、MySQL 5.6.4 以降では、CREATE TABLE ... IGNORE SELECT および CREATE TABLE ... REPLACE SELECT ステートメントには、ステートメントベースのレプリケーションには安全でないというフラグが付けられます。この変更により、このようなステートメントは、ステートメントベースモードを使用しているときはログ内に警告を生成し、MIXED

モードを使用しているときは行ベース形式を使用して ログに記録されます。セクション17.1.2.1「ステートメントベースおよび行ベースレプリケーションのメリットとデメリット」も参照してください。

CREATE TABLE ... SELECT は、どのインデックスも自動的には作成しません。これは、ステートメントをできるだけ柔軟にするために意図的に行われます。作成されたテーブル内にインデックスを設定する場合は、これらを SELECT ステートメントの前に指定するようにしてください。

```
mysql> CREATE TABLE bar (UNIQUE (n)) SELECT n FROM foo;
```

何らかのデータ型の変換が実行される可能性があります。たとえば、AUTO_INCREMENT 属性が保持されないため、VARCHAR カラムは CHAR カラムになることができます。リトレインされる属性は NULL (または NOT NULL) と、それらを含むカラムの場合は、CHARACTER SET、COLLATION、COMMENT、および DEFAULT 句です。

CREATE TABLE ... SELECT を使用してテーブルを作成する場合は、クエリー内のすべての関数呼び出しまたは式にエイリアスを付けるようにしてください。そうしないと、CREATE ステートメントが失敗するか、または好ましくないカラム名が生成される可能性があります。

```
CREATE TABLE artists_and_works
 SELECT artist.name, COUNT(work.artist_id) AS number_of_works
 FROM artist LEFT JOIN work ON artist.id = work.artist_id
 GROUP BY artist.id;
```

また、生成されるカラムのデータ型を明示的に指定することもできます。

```
CREATE TABLE foo (a TINYINT NOT NULL) SELECT b+1 AS a FROM bar;
```

CREATE TABLE ... SELECT で、IF NOT EXISTS が指定されているときに宛先テーブルがすでに存在する場合、その結果はバージョンに依存します。MySQL 5.5.6 より前は、MySQL はこのステートメントを次のように処理します。

- CREATE TABLE 部分で指定されているテーブル定義は無視されます。その定義が既存のテーブルの定義に一致しない場合でも、エラーは発生しません。MySQL は、いずれにしても SELECT 部分から行を挿入しようとします。
- テーブル内のカラム数と、SELECT 部分によって生成されたカラム数の間に不一致がある場合、選択された値は右端のカラムに割り当てられます。たとえば、テーブルに n 個のカラムが含まれているとき、SELECT によって m 個のカラムが生成される場合 (ここで、m < n)、選択された値はテーブル内の右端の m 個のカラムに割り当てられます。最初の n - m 個の各カラムにはデフォルト値が割り当てられます。このデフォルト値は、カラム定義で明示的に、またはその定義にデフォルトが含まれていない場合は暗黙的なカラムデータ型のデフォルトで指定されます。SELECT 部分によって生成されるカラムが多すぎる場合は (m > n)、エラーが発生します。
- 厳密な SQL モードが有効になっており、かつこれらの最初のカラムのいずれにも明示的なデフォルト値が含まれていない場合、このステートメントはエラーで失敗します。

次の例は、IF NOT EXISTS の処理を示しています。

```
mysql> CREATE TABLE t1 (i1 INT DEFAULT 0, i2 INT, i3 INT, i4 INT);
Query OK, 0 rows affected (0.05 sec)

mysql> CREATE TABLE IF NOT EXISTS t1 (c1 CHAR(10)) SELECT 1, 2;
Query OK, 1 row affected, 1 warning (0.01 sec)
Records: 1  Duplicates: 0  Warnings: 0

mysql> SELECT * FROM t1;
+------+------+------+------+
| i1   | i2   | i3   | i4   |
+------+------+------+------+
|    0 | NULL |    1 |    2 |
+------+------+------+------+
1 row in set (0.00 sec)
```

MySQL 5.5.6 の時点で、CREATE TABLE IF NOT EXISTS ... SELECT ステートメントの処理は、宛先テーブルがすでに存在するケースに関して変更されました。また、この変更には 5.1.51 以降の MySQL 5.1 での変更も含まれています。

- 以前は、CREATE TABLE IF NOT EXISTS ... SELECT の場合、MySQL はテーブルが存在するという警告を生成しましたが、いずれにしてもそれらの行を挿入し、そのステートメントをバイナリログに書き込みました。これに対して、CREATE TABLE ... SELECT (IF NOT EXISTS がありません) はエラーで失敗しましたが、MySQL は行を挿入せず、バイナリログにもステートメントを書き込みませんでした。

- MySQL は現在、宛先テーブルが存在する場合、両方のステートメントを同じ方法で処理します。つまり、どちらのステートメントも行を挿入せず、またバイナリログにも書き込まれません。これらの違いは、IF NOT EXISTS が存在する場合は MySQL が警告を生成し、存在しない場合はエラーを生成することです。

この変更は、前の例で言うと、MySQL 5.5.6 の時点では CREATE TABLE IF NOT EXISTS ... SELECT ステートメントが宛先テーブルに何も挿入しないことを示します。

この IF NOT EXISTS の処理の変更により、元の動作を行う MySQL 5.1 マスターから、新しい動作を行う MySQL 5.5 スレーブへのステートメントベースのレプリケーションで非互換性が発生します。CREATE TABLE IF NOT EXISTS ... SELECT がマスター上で実行されたときに、宛先テーブルが存在しているとします。その結果、マスター上では行が挿入されますが、スレーブ上では挿入されません。(行ベースのレプリケーションにはこの問題はありません。)

この問題に対処するために、MySQL 5.1 では、CREATE TABLE IF NOT EXISTS ... SELECT に対するステートメントベースのバイナリロギングが 5.1.51 の時点で変更されました。

- 宛先テーブルが存在しない場合は、変更ありません。ステートメントはそのまま口グに記録されます。
- 宛先テーブルが存在しない場合、ステートメントは CREATE TABLE IF NOT EXISTS および INSERT ... SELECT ステートメントの同等のペアとして口グに記録されます。(元のステートメント内の SELECT の前に IGNORE または REPLACE が指定されている場合は、INSERT がそれぞれ、INSERT IGNORE または REPLACE になります。)

この変更により、宛先テーブルが存在する場合はマスターとスレーブの両方で行が挿入されるため、MySQL 5.1 から 5.5 へのステートメントベースのレプリケーションでの上位互換性が提供されます。この互換性対策を利用するには、5.1 サーバーは少なくとも 5.1.51 である必要があり、5.5 サーバーは少なくとも 5.5.6 である必要があります。

既存の 5.1 から 5.5 へのレプリケーションシナリオをアップグレードするには、最初にマスターを 5.1.51 以降にアップグレードします。これは、最初にスレーブをアップグレードするという、通常のレプリケーションアップグレードのアドバイスとは異なります。

元の効果 (宛先テーブルが存在するかどうかには関係なく行が挿入される) を実現したいアプリケーションには、CREATE TABLE IF NOT EXISTS ... SELECT ステートメントではなく、CREATE TABLE IF NOT EXISTS および INSERT ... SELECT ステートメントを使用するという回避方法があります。

今説明した変更とともに、次の関連する変更が行われました。以前は、CREATE TABLE IF NOT EXISTS ... SELECT の宛先テーブルとして既存のビューが指定された場合、基礎となるベーステーブルに行が挿入され、このステートメントがバイナリログに書き込まれました。MySQL 5.1.51 および 5.5.6 の時点では、何も挿入されたり、ログに記録されたりしません。

バイナリログを使用して元のテーブルを確実に再作成できるようにするために、MySQL では、CREATE TABLE ... SELECT 中の並列挿入が許可されません。

> **重要**
>
> CREATE TABLE new_table SELECT ... FROM old_table ... などのステートメントで SELECT の一部として FOR UPDATE を使用することはできません。それを行おうとすると、このステートメントは失敗します。これは、CREATE TABLE ... SELECT ステートメントが作成されているテーブル以外のテーブルで変更を行うことを許可していた、MySQL 5.5 およびそれ以前からの動作の変更を表しています。
>
> この変更はまた、古いマスターから MySQL 5.6 以降のスレーブへのステートメントベースのレプリケーションにも影響を与える場合があります。詳細は、セクション 17.4.1.5「CREATE TABLE ... SELECT ステートメントのレプリケーション」を参照してください。

13.1.17.2 外部キー制約の使用

MySQL は、関連データのテーブルにまたがる相互参照を可能にする外部キーと、この分散したデータの整合性を維持するために役立つ外部キー制約をサポートします。CREATE TABLE または ALTER TABLE ステートメントで外部キー制約を定義するための基本的な構文は次のようになります。

```
[CONSTRAINT [symbol]] FOREIGN KEY
    [index_name] (index_col_name, ...)
    REFERENCES tbl_name (index_col_name,...)
    [ON DELETE reference_option]
```

```
  [ON UPDATE reference_option]

reference_option:
  RESTRICT | CASCADE | SET NULL | NO ACTION
```

index_name は、外部キー ID を表します。外部キーをサポートできる子テーブル上に明示的に定義されたインデックスがすでに存在する場合、index_name 値は無視されます。それ以外の場合、MySQL は、次のルールに従って名前が付けられた外部キーのインデックスを暗黙的に作成します。

- 定義されている場合は、CONSTRAINT symbol 値が使用されます。それ以外の場合は、FOREIGN KEY index_name 値が使用されます。
- CONSTRAINT symbol と FOREIGN KEY index_name のどちらも定義されていない場合、外部キーのインデックス名は、参照している外部キーカラムの名前を使用して生成されます。

外部キー定義は、次の条件に従います。

- 外部キー関係には、中央のデータ値を保持している親テーブルと、その元の親を指す同一の値を持っている子テーブルが含まれます。FOREIGN KEY 句は、子テーブルで指定されます。親テーブルと子テーブルは、同じストレージエンジンを使用する必要があります。これらは、TEMPORARY テーブルであってはいけません。
- 外部キー内の対応するカラムと、参照されるキーは同様のデータ型を持っている必要があります。整数型のサイズと符号が同じである必要があります。文字列型の長さが同じである必要はありません。バイナリ以外の (文字の) 文字列カラムの場合、文字セットと照合順序が同じである必要があります。
- MySQL では、外部キーチェックを高速に実行でき、かつテーブルスキャンが必要なくなるように、外部キーおよび参照されるキーに関するインデックスが必要です。参照しているテーブルには、外部キーカラムが同じ順序で最初のカラムとしてリストされているインデックスが存在する必要があります。このようなインデックスが存在しない場合は、参照しているテーブル上に自動的に作成されます。このインデックスは、外部キー制約を適用するために使用できる別のインデックスを作成した場合、あとで暗黙のうちに削除される可能性があります。index_name (指定されている場合) は、前に説明したとおりに使用されます。
- InnoDB では、外部キーが任意のインデックスカラムまたはカラムのグループを参照することが許可されます。ただし、参照されるテーブルには、参照されるカラムが同じ順序で最初のカラムとして一覧表示されているインデックスが存在する必要があります。

  NDB には、外部キーとして参照されるいずれかのカラム上の明示的な一意のキー (または主キー) が必要です。
- 外部キーカラム上のインデックスプリフィクスはサポートされていません。この 1 つの影響として、BLOB および TEXT カラム上のインデックスには常にプリフィクス長が含まれている必要があるため、それらのカラムを外部キーに含めることができない点があります。
- CONSTRAINT symbol 句が指定されている場合、symbol 値 (使用されている場合) はデータベース内で一意である必要があります。symbol が重複している場合は、次のような エラーが発生します: ERROR 1022 (2300): 書き込めません。テーブル '#sql- 464_1' に重複するキーがあります'。この句が指定されていない場合や、CONSTRAINT キーワードのあとに symbol が含まれていない場合は、制約の名前が自動的に作成されます。
- 現在、InnoDB ではユーザー定義のパーティションを持つテーブルの外部キーがサポートされていません。これには、親テーブルと子テーブルの両方が含まれます。

  この制限は、KEY または LINEAR KEY によってパーティション化された NDB テーブル (NDB ストレージエンジンによってサポートされる唯一のユーザーパーティショニングタイプ) には適用されません。これらは外部キー参照を含むか、またはこのような参照のターゲットになることができます。
- NDB テーブルでは、参照先が親テーブルの主キーである場合、ON UPDATE CASCADE はサポートされません。

参照アクション

このセクションでは、外部キーが参照整合性の保証にどのように役立つかについて説明します。

外部キーをサポートするストレージエンジンで、親テーブル内に一致する候補のキー値が存在しない場合、MySQL は、子テーブル内に外部キー値を作成しようとするすべての INSERT または UPDATE 操作を拒否します。

UPDATE または DELETE 操作が、子テーブル内に一致する行を持つ親テーブル内のキー値に影響を与える場合、その結果は、FOREIGN KEY 句の ON UPDATE および ON DELETE サブ句を使用して指定された参照アクショ

ンによって異なります。MySQL は、実行されるアクションに関連した次の 5 つのオプションをサポートしています。

- CASCADE: 親テーブルの行を削除または更新し、子テーブル内の一致する行を自動的に削除または更新します。ON DELETE CASCADE と ON UPDATE CASCADE の両方がサポートされています。2 つのテーブル間で、親テーブルまたは子テーブル内の同じカラムに対して機能する複数の ON UPDATE CASCADE 句を定義しないでください。

  > **注記**
  >
  > 現在、カスケードされた外部キーのアクションではトリガーがアクティブになっていません。

- SET NULL: 親テーブルの行を削除または更新し、子テーブル内の 1 つまたは複数の外部キーカラムを NULL に設定します。ON DELETE SET NULL 句と ON UPDATE SET NULL 句の両方がサポートされています。

  SET NULL アクションを指定する場合は、子テーブル内のカラムを NOT NULL として宣言していないことを確認してください。

- RESTRICT: 親テーブルに対する削除または更新操作を拒否します。RESTRICT (または NO ACTION) を指定することは、ON DELETE または ON UPDATE 句を省略することと同じです。

- NO ACTION: 標準 SQL のキーワード。MySQL では、RESTRICT と同等です。MySQL Server は、参照されるテーブル内に関連する外部キー値が存在する場合、親テーブルに対する削除または更新操作を拒否します。一部のデータベースシステムは遅延チェックを備えており、その場合、NO ACTION は遅延チェックです。MySQL では、外部キー制約はただちにチェックされるため、NO ACTION は RESTRICT と同じです。

- SET DEFAULT: このアクションは MySQL パーサーによって認識されますが、InnoDB と NDB はどちらも、ON DELETE SET DEFAULT または ON UPDATE SET DEFAULT 句を含むテーブル定義を拒否します。

指定されていない ON DELETE または ON UPDATE では、デフォルトのアクションは常に RESTRICT です。

MySQL は、1 つのテーブル内のあるカラムと別のカラムの間の外部キー参照をサポートしています。(あるカラムが、それ自体への外部キー参照を持つことはできません。)これらの場合、「子テーブルのレコード」は、実際に同じテーブル内の依存レコードを参照します。

外部キー句の例

単一カラム外部キーを使用して parent および child テーブルを関連付ける単純な例を次に示します。

```
CREATE TABLE parent (
    id INT NOT NULL,
    PRIMARY KEY (id)
) ENGINE=INNODB;

CREATE TABLE child (
    id INT,
    parent_id INT,
    INDEX par_ind (parent_id),
    FOREIGN KEY (parent_id)
        REFERENCES parent(id)
        ON DELETE CASCADE
) ENGINE=INNODB;
```

product_order テーブルにほかの 2 つのテーブルへの外部キーが存在する、より複雑な例。1 つの外部キーが、product テーブル内の 2 カラムのインデックスを参照しています。もう一方の外部キーは、customer テーブル内の単一カラムインデックスを参照しています。

```
CREATE TABLE product (
    category INT NOT NULL, id INT NOT NULL,
    price DECIMAL,
    PRIMARY KEY(category, id)
)   ENGINE=INNODB;

CREATE TABLE customer (
    id INT NOT NULL,
    PRIMARY KEY (id)
)   ENGINE=INNODB;

CREATE TABLE product_order (
    no INT NOT NULL AUTO_INCREMENT,
    product_category INT NOT NULL,
```

```
  product_id INT NOT NULL,
  customer_id INT NOT NULL,

  PRIMARY KEY(no),
  INDEX (product_category, product_id),
  INDEX (customer_id),

  FOREIGN KEY (product_category, product_id)
    REFERENCES product(category, id)
    ON UPDATE CASCADE ON DELETE RESTRICT,

  FOREIGN KEY (customer_id)
    REFERENCES customer(id)
)  ENGINE=INNODB;
```

外部キーの追加

ALTER TABLE を使用して、既存のテーブルに新しい外部キー制約を追加できます。このステートメントの外部キーに関連した構文を次に示します。

```
ALTER TABLE tbl_name
  ADD [CONSTRAINT [symbol]] FOREIGN KEY
  [index_name] (index_col_name, ...)
  REFERENCES tbl_name (index_col_name,...)
  [ON DELETE reference_option]
  [ON UPDATE reference_option]
```

外部キーは、自己参照型にする (同じテーブルを参照する) ことができます。ALTER TABLE を使用してテーブルに外部キー制約を追加する場合は、まず必要なインデックスを作成することを忘れないでください。

外部キーの削除

次に示す構文を使用して、ALTER TABLE で外部キーを削除することもできます。

```
ALTER TABLE tbl_name DROP FOREIGN KEY fk_symbol;
```

外部キーを作成したときに FOREIGN KEY 句に CONSTRAINT の名前が含まれていた場合は、その名前を参照して外部キーを削除できます。それ以外の場合は、外部キーが作成されるときに fk_symbol 値が内部的に生成されます。外部キーを削除するときにシンボル値を見つけるには、次に示すように、SHOW CREATE TABLE ステートメントを使用します。

```
mysql> SHOW CREATE TABLE ibtest11c\G
*************************** 1. row ***************************
       Table: ibtest11c
Create Table: CREATE TABLE `ibtest11c` (
 `A` int(11) NOT NULL auto_increment,
 `D` int(11) NOT NULL default '0',
 `B` varchar(200) NOT NULL default '',
 `C` varchar(175) default NULL,
 PRIMARY KEY  (`A`,`D`,`B`),
 KEY `B` (`B`,`C`),
 KEY `C` (`C`),
 CONSTRAINT `0_38775` FOREIGN KEY (`A`, `D`)
REFERENCES `ibtest11a` (`A`, `D`)
ON DELETE CASCADE ON UPDATE CASCADE,
 CONSTRAINT `0_38776` FOREIGN KEY (`B`, `C`)
REFERENCES `ibtest11a` (`B`, `C`)
ON DELETE CASCADE ON UPDATE CASCADE
) ENGINE=INNODB CHARSET=latin1
1 row in set (0.01 sec)

mysql> ALTER TABLE ibtest11c DROP FOREIGN KEY `0_38775`;
```

MySQL 5.6.6 より前は、同じ ALTER TABLE ステートメントでの外部キーの追加と削除は、問題が発生する場合があるためサポートされていません。操作ごとに個別のステートメントを使用するようにしてください。MySQL 5.6.6 の時点では、同じ ALTER TABLE ステートメントでの外部キーの追加と削除は ALTER TABLE ... ALGORITHM=INPLACE ではサポートされますが、ALTER TABLE ... ALGORITHM=COPY では未サポートのままです。

MySQL 5.6.7 より前は、ALTER TABLE を使用して外部キーカラムの定義を変更すると、参照整合性が失われる可能性がありました。たとえば、NULL 値を含む外部キーカラムを NOT NULL になるように変更すると、NULL 値が空の文字列になりました。同様に、親テーブル内の行を削除する ALTER TABLE IGNORE によって、参照整合性が破壊される可能性がありました。

5.6.7 の時点では、参照整合性が失われる可能性のある外部キーカラムへの変更がサーバーによって禁止されます。回避方法として、カラム定義を変更する前に ALTER TABLE ... DROP FOREIGN KEY を使用し、あとで ALTER TABLE ... ADD FOREIGN KEY を使用します。

外部キーおよびその他の MySQL ステートメント

FOREIGN KEY ... REFERENCES ... 句内のテーブルとカラムの識別子は、逆引用符 (`) で囲むことができます。あるいは、ANSI_QUOTES SQL モードが有効になっている場合は、二重引用符 (") を使用できます。また、lower_case_table_names システム変数の設定も考慮に入れられます。

SHOW CREATE TABLE ステートメントの出力の一部として、子テーブルの外部キー定義を表示できます。

```
SHOW CREATE TABLE tbl_name;
```

INFORMATION_SCHEMA.KEY_COLUMN_USAGE テーブルをクエリーすることによって、外部キーに関する情報を取得することもできます。

INNODB_SYS_FOREIGN および INNODB_SYS_FOREIGN_COLS テーブル、さらには INFORMATION_SCHEMA データベース内の InnoDB テーブルによって使用される外部キーに関する情報を検索できます。

mysqldump は、テーブルの正しい定義 (子テーブルへの外部キーを含む) をダンプファイル内に生成します。

外部キー関係を持つテーブルのダンプファイルのリロードを容易にするために、mysqldump は、foreign_key_checks を 0 に設定するステートメントをダンプ出力内に自動的に含めます。これにより、ダンプがリロードされるときに特定の順序でリロードする必要のあるテーブルに関する問題が回避されます。また、この変数を手動で設定することもできます。

```
mysql> SET foreign_key_checks = 0;
mysql> SOURCE dump_file_name;
mysql> SET foreign_key_checks = 1;
```

これにより、外部キーに関して正しく順序付けられていないテーブルがダンプファイルに含まれている場合でも、そのテーブルを任意の順序でインポートできます。また、インポート操作も高速化されます。foreign_key_checks を 0 に設定することは、LOAD DATA および ALTER TABLE 操作中に外部キー制約を無視するためにも役立つ場合があります。ただし、foreign_key_checks = 0 の場合でも、MySQL では、カラムが一致しないカラム型を参照している外部キー制約の作成は許可されません。また、テーブルに外部キー制約が存在する場合は、ALTER TABLE を使用して、そのテーブルを別のストレージエンジンを使用するように変更することはできません。ストレージエンジンを変更するには、まず外部キー制約をすべて削除する必要があります。

SET foreign_key_checks = 0 を実行しないかぎり、FOREIGN KEY 制約によって参照されるテーブルに対して DROP TABLE を発行できません。テーブルを削除すると、そのテーブルを作成するために使用されたステートメントで定義されていた制約もすべて削除されます。

削除されたテーブルを再作成する場合は、そのテーブルに、それを参照している外部キー制約に準拠する定義が存在する必要があります。また、カラムの正しい名前と型、および前に説明した参照されるキーに関するインデックスが存在する必要があります。これらが満たされていない場合、MySQL はエラー 1005 を返し、エラーメッセージでエラー 150 を示します。これは、外部キー制約が正しく形成されなかったことを示します。同様に、ALTER TABLE がエラー 150 で失敗した場合、これは、変更されたテーブルのために外部キー定義が誤って形成されることを示します。

InnoDB テーブルの場合は、SHOW ENGINE INNODB STATUS の出力をチェックすることによって、MySQL Server で最新の InnoDB 外部キーエラーの詳細な説明を取得できます。

> **重要**
>
> ANSI/ISO SQL 標準に精通しているユーザーの場合は、参照整合性の制約定義で使用される MATCH 句を認識または適用するストレージエンジンは (InnoDB を含め) 存在しません。明示的な MATCH 句を使用しても、指定された効果が得られないだけでなく、ON DELETE および ON UPDATE 句が無視される原因にもなります。これらの理由により、MATCH の指定は避けるようにしてください。
>
> SQL 標準での MATCH 句は、複合 (マルチカラム) 外部キー内の NULL 値が、主キーとの比較時にどのように処理されるかを制御します。MySQL は基本的に、外部キーをすべてまたは部分的に NULL にすることが許可される、MATCH SIMPLE で定義されるセマンティクスを実装しています。その場合は、このような外部キーを含む (子テーブル

> の) 行の挿入が許可され、その行は参照される (親) テーブル内のどの行にも一致しません。トリガーを使用して、ほかのセマンティクスを実装できます。
>
> さらに、MySQL ではパフォーマンス上の理由から、参照されるカラムにインデックスを設定する必要があります。ただし、システムでは、参照されるカラムを UNIQUE にするか、または NOT NULL として宣言するという要件は適用されません。一意でないキーまたは NULL 値を含むキーへの外部キー参照の処理は、UPDATE や DELETE CASCADE などの操作に対して適切に定義されていません。UNIQUE (PRIMARY を含む) および NOT NULL キーのみを参照する外部キーを使用することをお勧めします。
>
> さらに、MySQL は、参照がカラム指定の一部として定義されている (SQL 標準で定義された)「インラインの REFERENCES 指定」を認識せず、またサポートもしていません。MySQL は、個別の FOREIGN KEY 指定の一部として指定されている場合にのみ REFERENCES 句を受け入れます。外部キーをサポートしていない (MyISAM などの) ストレージエンジンの場合、MySQL Server は、外部キーの指定を解析して無視します。

13.1.17.3 暗黙のカラム指定の変更

MySQL は場合によって、カラム指定を CREATE TABLE または ALTER TABLE ステートメントで指定されたものから暗黙のうちに変更することがあります。これらの変更は、データ型、データ型に関連付けられた属性、またはインデックス指定に対して行われる可能性があります。

すべての変更は 65,535 バイトの内部の行サイズ制限に従うため、データ型を変更しようとする一部の試みが失敗する可能性があります。セクションD.10.4「テーブルカラム数と行サイズの制限」を参照してください。

- PRIMARY KEY の一部であるカラムは、そのように宣言されていない場合でも、NOT NULL にされます。
- テーブルが作成されたとき、ENUM および SET メンバー値から末尾のスペースが自動的に削除されます。
- MySQL は、ほかの SQL データベースベンダーによって使用されている特定のデータ型を MySQL 型にマップします。セクション11.9「その他のデータベースエンジンのデータ型の使用」を参照してください。
- 特定のストレージエンジンには許可されないインデックスタイプを指定するために USING 句を含めたが、そのエンジンがクエリー結果に影響を与えることなく使用できる使用可能な別のインデックスタイプが存在する場合、エンジンはその使用可能なタイプを使用します。
- 厳密な SQL モードが有効になっていない場合、長さ指定が 65535 より大きい VARCHAR カラムは TEXT に変換され、長さ指定が 65535 より大きい VARBINARY カラムは BLOB に変換されます。そうでない場合は、これらのいずれの場合にも エラーが発生します。
- 文字データ型に CHARACTER SET binary 属性を指定すると、カラムは対応するバイナリデータ型として作成されます。つまり、CHAR は BINARY になり、VARCHAR は VARBINARY になり、TEXT は BLOB になります。ENUM および SET データ型では、これは行われず、宣言されたとおりに作成されます。この定義を使用して、テーブルを指定したとします。

```
CREATE TABLE t
(
  c1 VARCHAR(10) CHARACTER SET binary,
  c2 TEXT CHARACTER SET binary,
  c3 ENUM('a','b','c') CHARACTER SET binary
);
```

結果のテーブルには、この定義が含まれています。

```
CREATE TABLE t
(
  c1 VARBINARY(10),
  c2 BLOB,
  c3 ENUM('a','b','c') CHARACTER SET binary
);
```

MySQL が、指定したもの以外のデータ型を使用したかどうかを確認するには、テーブルを作成または変更したあとに、DESCRIBE または SHOW CREATE TABLE ステートメントを発行します。

myisampack を使用してテーブルを圧縮する場合は、その他の特定のデータ型の変更が発生する場合があります。セクション15.2.3.3「圧縮テーブルの特徴」を参照してください。

13.1.18 CREATE TABLESPACE 構文

```
CREATE TABLESPACE tablespace_name
```

```
  ADD DATAFILE 'file_name'
  USE LOGFILE GROUP logfile_group
  [EXTENT_SIZE [=] extent_size]
  [INITIAL_SIZE [=] initial_size]
  [AUTOEXTEND_SIZE [=] autoextend_size]
  [MAX_SIZE [=] max_size]
  [NODEGROUP [=] nodegroup_id]
  [WAIT]
  [COMMENT [=] comment_text]
  ENGINE [=] engine_name
```

このステートメントは、テーブルスペースを作成するために使用されます。テーブルスペースは 1 つ以上のデータファイルを含むことができ、テーブルのストレージ領域を提供します。このステートメントを使用して 1 つのデータファイルが作成され、テーブルスペースに追加されます。ALTER TABLESPACE ステートメントを使用して、テーブルスペースにデータファイルを追加できます (セクション13.1.8「ALTER TABLESPACE 構文」を参照してください)。テーブルスペースの命名を管理するルールについては、セクション9.2「スキーマオブジェクト名」を参照してください。

> **注記**
>
> すべての MySQL Cluster ディスクデータオブジェクトが同じ名前空間を共有します。つまり、各ディスクデータオブジェクトは (単に、特定の型の各ディスクデータオブジェクトというだけでなく)、一意の名前が付けられている必要があります。たとえば、テーブルスペースとログファイルグループを同じ名前にしたり、テーブルスペースとデータファイルを同じ名前にしたりすることはできません。

作成されるテーブルスペースには、USE LOGFILE GROUP 句を使用して、1 つ以上の UNDO ログファイルのログファイルグループを割り当てる必要があります。logfile_group は、CREATE LOGFILE GROUP で作成された既存のログファイルグループである必要があります (セクション13.1.14「CREATE LOGFILE GROUP 構文」を参照してください)。複数のテーブルスペースが UNDO ログギングのために同じログファイルグループを使用できます。

EXTENT_SIZE は、そのテーブルスペースに属するすべてのファイルによって使用されるエクステントのサイズ (バイト単位) を設定します。デフォルト値は 1M です。最小サイズは 32K であり、理論的な最大サイズは 2G です。ただし、実際的な最大サイズはいくつかの要因によって異なります。ほとんどの場合は、エクステントサイズを変更してもパフォーマンスに測定可能な影響を与えることはないため、特別な状況を除き、常にデフォルト値を使用することをお勧めします。

エクステントは、ディスク領域の割り当ての単位です。1 つのエクステントが、そのエクステントに収容できる量のデータでいっぱいになってから、別のエクステントが使用されます。理論上は、データファイルあたり最大 65,535 (64K) 個のエクステントを使用できます。ただし、推奨される最大数は 32,768 (32K) です。1 つのデータファイルの推奨される最大サイズは 32G (つまり、32K 個のエクステント × エクステントあたり 1M バイト) です。さらに、エクステントを特定のパーティションに割り当てたあと、そのエクステントを使用して別のパーティションのデータを格納することはできません。エクステントには、複数のパーティションのデータを格納できません。つまり、たとえば、INITIAL_SIZE が 256M バイトで、EXTENT_SIZE が 128M である 1 つのデータファイルを含むテーブルスペースにはエクステントが 2 つしか存在しないため、このテーブルスペースを使用して最大 2 つの異なるディスクデータテーブルパーティションのデータを格納できます。

INFORMATION_SCHEMA.FILES テーブルをクエリーすることによって、特定のデータファイルに未使用のまま残っているエクステントの数を確認できるため、ファイル内の空き容量の概算値を導き出すことができます。それ以上の説明および例については、セクション21.30.1「INFORMATION_SCHEMA FILES テーブル」を参照してください。

INITIAL_SIZE パラメータは、データファイルの合計サイズをバイト単位で設定します。ファイルが作成されたあと、そのサイズを変更することはできません。ただし、ALTER TABLESPACE ... ADD DATAFILE を使用して、テーブルスペースにさらに多くのデータファイルを追加できます。セクション13.1.8「ALTER TABLESPACE 構文」を参照してください。

INITIAL_SIZE はオプションです。そのデフォルト値は 134217728 (128M バイト) です。

32 ビットシステム上では、INITIAL_SIZE のサポートされる最大値は 4294967296 (4G バイト) です。(Bug #29186)

EXTENT_SIZE を設定する場合は、数値のあとにオプションで、my.cnf で使用されるのと同様の、桁を示す 1 文字の略語を指定できます。一般に、これは M (M バイト) または G (G バイト) のどちらかの文字です。MySQL Cluster NDB 7.3.2 以降では、これらの略語は INITIAL_SIZE を指定する場合もサポートされます。(Bug #13116514、Bug #16104705、Bug #62858)

INITIAL_SIZE、EXTENT_SIZE、および UNDO_BUFFER_SIZE は、次のような丸めに従います。

- EXTENT_SIZE と UNDO_BUFFER_SIZE はそれぞれ、32K のもっとも近い整数倍に切り上げられます。
- INITIAL_SIZE は、32K のもっとも近い整数倍に切り下げられます。

  データファイルの場合は、INITIAL_SIZE に対してさらに丸め処理が行われます。今得られた結果が (すべての丸めのあと) EXTENT_SIZE のもっとも近い整数倍に切り上げられます。

今説明した丸めは明示的に実行され、このような丸めのいずれかが実行された場合は MySQL Server によって警告が発行されます。丸められた値はまた、INFORMATION_SCHEMA.FILES カラム値の計算やその他の目的のために、NDB カーネルでも使用されます。ただし、予期しない結果が発生しないようにするために、これらのオプションの指定では常に 32K の整数倍を使用することをお勧めします。

AUTOEXTEND_SIZE、MAX_SIZE、NODEGROUP、WAIT、および COMMENT は解析されますが、無視されるため、現在は何の効果もありません。これらのオプションは、将来の拡張のために用意されています。

ENGINE パラメータは、このテーブルスペースが使用するストレージエンジンを決定します。ここで、engine_name はそのストレージエンジンの名前です。現在、engine_name は、値 NDB または NDBCLUSTER のいずれかである必要があります。

CREATE TABLESPACE が ENGINE = NDB とともに使用された場合は、テーブルスペースとそれに関連付けられたデータファイルが各クラスタデータノード上に作成されます。INFORMATION_SCHEMA.FILES テーブルをクエリーすることによって、データファイルが作成されたことを確認したり、それらに関する情報を取得したりできます。例:

```
mysql> SELECT LOGFILE_GROUP_NAME, FILE_NAME, EXTRA
    -> FROM INFORMATION_SCHEMA.FILES
    -> WHERE TABLESPACE_NAME = 'newts' AND FILE_TYPE = 'DATAFILE';
+--------------------+-------------+----------------+
| LOGFILE_GROUP_NAME | FILE_NAME   | EXTRA          |
+--------------------+-------------+----------------+
| lg_3               | newdata.dat | CLUSTER_NODE=3 |
| lg_3               | newdata.dat | CLUSTER_NODE=4 |
+--------------------+-------------+----------------+
2 rows in set (0.01 sec)
```

(セクション21.30.1「INFORMATION_SCHEMA FILES テーブル」を参照してください。)

CREATE TABLESPACE は、MySQL Cluster のディスクデータストレージでのみ有効です。セクション18.5.12「MySQL Cluster ディスクデータテーブル」を参照してください。

13.1.19 CREATE TRIGGER 構文

```
CREATE
    [DEFINER = { user | CURRENT_USER }]
    TRIGGER trigger_name
    trigger_time trigger_event
    ON tbl_name FOR EACH ROW
    trigger_body

trigger_time: { BEFORE | AFTER }

trigger_event: { INSERT | UPDATE | DELETE }
```

このステートメントは、新しいトリガーを作成します。トリガーとは、テーブルに関連付けられ、そのテーブルに対して特定のイベントが発生するとアクティブ化される名前付きデータベースオブジェクトのことです。トリガーは、tbl_name という名前のテーブルに関連付けられます。これは、永続的なテーブルを指す必要があります。トリガーを TEMPORARY テーブルまたはビューに関連付けることはできません。

トリガー名はスキーマの名前空間内に存在します。つまり、すべてのトリガーがスキーマ内で一意の名前を持つ必要があります。異なるスキーマ内のトリガーは同じ名前を持つことができます。

このセクションでは、CREATE TRIGGER 構文について説明します。詳細は、セクション20.3.1「トリガーの構文と例」を参照してください。

CREATE TRIGGER には、このトリガーに関連付けられたテーブルに対する TRIGGER 権限が必要です。このセクションのあとの方で説明されているように、DEFINER 値によっては、このステートメントに SUPER 権限も必要になる可能性があります。バイナリロギングが有効になっている場合は、セクション20.7「ストアドプログラ

ムのバイナリロギング」で説明されているように、CREATE TRIGGER に SUPER 権限が必要になることがあります。

DEFINER 句は、このセクションのあとの方で説明されているように、トリガーのアクティブ化時にアクセス権限を確認するときに使用されるセキュリティーコンテキストを決定します。

trigger_time は、このトリガーのアクション時間です。これは、トリガーが各行の変更の前またはあとにアクティブ化されることを示す BEFORE または AFTER にすることができます。

trigger_event は、このトリガーをアクティブ化する操作の種類を示します。次の trigger_event 値が許可されます。

- INSERT: このトリガーは (たとえば、INSERT、LOAD DATA、および REPLACE ステートメントを使用して) 新しい行がテーブルに挿入されると常にアクティブ化されます。
- UPDATE: このトリガーは (たとえば、UPDATE ステートメントを使用して) 行が変更されると常にアクティブ化されます。
- DELETE: このトリガーは (たとえば、DELETE および REPLACE ステートメントを使用して) 行がテーブルから削除されると常にアクティブ化されます。テーブルに対する DROP TABLE および TRUNCATE TABLE ステートメントは、DELETE を使用しないため、このトリガーをアクティブ化しません。また、パーティションを削除しても DELETE トリガーはアクティブ化されません。

trigger_event は、トリガーをアクティブ化する SQL ステートメントのリテラル型を表しているのではなく、テーブル操作の種類を表しています。たとえば、INSERT トリガーは、INSERT ステートメントだけでなく、LOAD DATA ステートメントでもアクティブ化されます。それは、どちらのステートメントもテーブルに行を挿入するためです。

この混乱を招く可能性がある例として、INSERT INTO ... ON DUPLICATE KEY UPDATE ... 構文があります。すべての行で BEFORE INSERT トリガーがアクティブ化されたあと、その行に重複キーが存在したかどうかに応じて、AFTER INSERT トリガーだけか、または BEFORE UPDATE トリガーと AFTER UPDATE トリガーの両方がアクティブ化されます。

> **注記**
>
> カスケードされた外部キーアクションはトリガーをアクティブ化しません。

特定のテーブルに対して、トリガーイベントとアクション時間が同じ複数のトリガーが存在していてはいけません。たとえば、1 つのテーブルに対して 2 つの BEFORE UPDATE トリガーを定義することはできません。ただし、BEFORE UPDATE および BEFORE INSERT トリガー、または BEFORE UPDATE および AFTER UPDATE トリガーは設定できます。

trigger_body は、トリガーがアクティブ化されるときに実行されるステートメントです。複数のステートメントを実行するには、BEGIN ... END 複合ステートメント構造構文を使用します。また、これにより、ストアドルーチン内で許可されるのと同じステートメントを使用することもできます。セクション13.6.1「BEGIN ... END 複合ステートメント構文」を参照してください。一部のステートメントは、トリガー内では許可されません。セクションD.1「ストアドプログラムの制約」を参照してください。

トリガー本体内では、エイリアス OLD と NEW を使用して、対象テーブル (そのトリガーに関連付けられたテーブル) 内のカラムを参照できます。OLD.col_name は、更新または削除される前の既存の行のカラムを示します。NEW.col_name は、挿入された新しい行、または更新されたあとの既存の行のカラムを示します。

MySQL は、トリガーが作成されたときの有効な sql_mode システム変数の設定を格納し、トリガーが実行を開始したときの現在のサーバー SQL モードには関係なく、常にそのトリガー本体を強制的にこの設定で実行します。

DEFINER 句は、トリガーのアクティブ化時にアクセス権限を確認するときに使用される MySQL アカウントを指定します。user 値を指定する場合は、'user_name'@'host_name' (GRANT ステートメントで使用されるのと同じ形式)、CURRENT_USER、または CURRENT_USER() として指定された MySQL アカウントにしてください。DEFINER のデフォルト値は、CREATE TRIGGER ステートメントを実行するユーザーです。これは、明示的に DEFINER = CURRENT_USER を指定するのと同じです。

DEFINER 句を指定した場合は、次のルールによって有効な DEFINER ユーザーの値が決定されます。

- SUPER 権限がない場合、許可される唯一の user 値は、リテラルで指定するか、または CURRENT_USER を使用して指定した自分のアカウントです。定義者をほかのアカウントに設定することはできません。
- SUPER 権限がある場合は、構文として有効な任意のアカウント名を指定できます。そのアカウントが実際に存在しない場合は、警告が生成されます。

- 存在しない DEFINER アカウントでトリガーを作成することはできますが、そのアカウントが実際に存在するようになるまで、このようなトリガーをアクティブ化することはお勧めできません。それ以外の権限確認に関する動作は定義されていません。

MySQL は、トリガー権限を確認するときに、DEFINER ユーザーを次のように考慮します。

- CREATE TRIGGER の時点で、このステートメントを発行するユーザーには TRIGGER 権限が必要です。
- トリガーのアクティブ化時、権限は DEFINER ユーザーに対して確認されます。このユーザーには、次の権限が必要です。
  - 対象テーブルに対する TRIGGER 権限。
  - テーブルカラムへの参照がトリガー本体内の OLD.col_name または NEW.col_name を使用して発生した場合は、対象テーブルに対する SELECT 権限。
  - テーブルカラムがトリガー本体内の SET NEW.col_name = value 割り当てのターゲットである場合は、対象テーブルに対する UPDATE 権限。
  - その他のどのような権限も、通常、そのトリガーによって実行されるステートメントに必要です。

トリガーのセキュリティーの詳細は、セクション20.6「ストアドプログラムおよびビューのアクセスコントロール」を参照してください。

トリガー本体内で、CURRENT_USER() 関数は、トリガーのアクティブ化時に権限を確認するために使用されるアカウントを返します。これは、そのトリガーがアクティブ化される原因となるアクションを実行したユーザーではなく、DEFINER ユーザーです。トリガー内のユーザー監査については、セクション6.3.13「SQL ベースの MySQL アカウントアクティビティーの監査」を参照してください。

LOCK TABLES を使用してトリガーを含むテーブルをロックした場合は、セクション13.3.5.2「LOCK TABLES とトリガー」で説明されているように、そのトリガー内で使用されているテーブルもロックされます。

トリガーの使用の詳細は、セクション20.3.1「トリガーの構文と例」を参照してください。

13.1.20 CREATE VIEW 構文

```
CREATE
    [OR REPLACE]
    [ALGORITHM = {UNDEFINED | MERGE | TEMPTABLE}]
    [DEFINER = { user | CURRENT_USER }]
    [SQL SECURITY { DEFINER | INVOKER }]
    VIEW view_name [(column_list)]
    AS select_statement
    [WITH [CASCADED | LOCAL] CHECK OPTION]
```

CREATE VIEW ステートメントは、新しいビューを作成するか、または OR REPLACE 句が指定されている場合は既存のビューを置き換えます。そのビューが存在しない場合、CREATE OR REPLACE VIEW は CREATE VIEW と同じです。そのビューが存在する場合、CREATE OR REPLACE VIEW は ALTER VIEW と同じです。

select_statement は、そのビューの定義を提供する SELECT ステートメントです。(ビューから選択すると、事実上、SELECT ステートメントを使用して選択したことになります。) select_statement は、ベーステーブルまたはほかのビューから選択できます。

ビュー定義は作成時に「固定される」ため、ベースとなるテーブルへのそれ以降の変更はビュー定義に影響を与えません。たとえば、ビューがテーブル上で SELECT * として定義されている場合、あとでテーブルに追加された新しいカラムはそのビューの一部になりません。

ALGORITHM 句は、MySQL によるビューの処理方法に影響を与えます。DEFINER および SQL SECURITY 句は、ビューの呼び出し時にアクセス権限を確認するときに使用されるセキュリティーコンテキストを指定します。WITH CHECK OPTION 句を指定すると、ビューによって参照されているテーブル内の行への挿入または更新を制約できます。これらの句については、このセクションのあとの方で説明されています。

CREATE VIEW ステートメントには、このビューに対する CREATE VIEW 権限と、SELECT ステートメントによって選択される各カラムに対する何らかの権限が必要です。SELECT ステートメント内の別の場所で使用されているカラムに対しては、SELECT 権限が必要です。OR REPLACE 句が存在する場合は、このビューに対する DROP 権限も必要です。このセクションのあとの方で説明されているように、DEFINER 値によっては、CREATE VIEW に SUPER 権限も必要になる可能性があります。

ビューが参照されると、このセクションのあとの方で説明されている権限確認が発生します。

ビューはデータベースに属します。デフォルトでは、新しいビューはデフォルトデータベース内に作成されます。ビューを明示的に特定のデータベース内に作成するには、そのビューの作成時に、その名前を db_name.view_name として指定します。

```
mysql> CREATE VIEW test.v AS SELECT * FROM t;
```

データベース内で、ベーステーブルとビューは同じ名前空間を共有するため、ベーステーブルとビューが同じ名前を持つことはできません。

SELECT ステートメントによって取得されるカラムは、テーブルカラムへの単純な参照にすることができます。また、関数、定数値、演算子などを使用した式にすることもできます。

ビューは、ベーステーブルと同様に、重複のない一意のカラム名を持つ必要があります。デフォルトでは、SELECT ステートメントによって取得されるカラムの名前はビューカラム名に使用されます。ビューカラムの明示的な名前を定義するには、オプションの column_list 句をカンマで区切られた識別子のリストとして指定できます。column_list 内の名前の数は、SELECT ステートメントによって取得されるカラムの数と同じである必要があります。

SELECT ステートメント内の修飾されていないテーブルまたはビュー名は、デフォルトデータベースを基準にして解釈されます。ビューは、テーブルまたはビュー名を適切なデータベース名で修飾することによって、ほかのデータベース内のテーブルまたはビューを参照できます。

ビューは、多くの種類の SELECT ステートメントから作成できできます。ベーステーブルまたはほかのビューを参照できます。結合、UNION、およびサブクエリーを使用できます。SELECT がテーブルをまったく参照しなくてもかまいません。次の例では、別のテーブルからの 2 つのカラムに加え、それらのカラムから計算される式を選択するビューを定義しています。

```
mysql> CREATE TABLE t (qty INT, price INT);
mysql> INSERT INTO t VALUES(3, 50);
mysql> CREATE VIEW v AS SELECT qty, price, qty*price AS value FROM t;
mysql> SELECT * FROM v;
+------+-------+-------+
| qty  | price | value |
+------+-------+-------+
|    3 |    50 |   150 |
+------+-------+-------+
```

ビュー定義は、次の制限に従います。

- SELECT ステートメントに FROM 句内のサブクエリーを含めることはできません。
- SELECT ステートメントは、システムまたはユーザー変数を参照できません。
- ストアドプログラム内で、この定義は、プログラムパラメータまたはローカル変数を参照できません。
- SELECT ステートメントは、準備済みステートメントのパラメータを参照できません。
- この定義で参照されているテーブルまたはビューは、すべて存在する必要があります。ただし、ビューが作成されたあとは、この定義で参照されているテーブルまたはビューを削除できます。この場合は、このビューを使用すると、エラーが発生します。この種類の問題に関してビュー定義を確認するには、CHECK TABLE ステートメントを使用します。
- この定義は TEMPORARY テーブルを参照できないため、TEMPORARY ビューは作成できません。
- ビュー定義で指定されているテーブルは、すべて定義時に存在する必要があります。
- トリガーをビューに関連付けることはできません。
- SELECT ステートメント内のカラム名のエイリアスは (256 文字の別名の最大の長さではなく) 64 文字のカラムの最大の長さに対してチェックされます。

ORDER BY はビュー定義内で許可されていますが、独自の ORDER BY を含むステートメントを使用しているビューから選択した場合は無視されます。

この定義内のその他のオプションまたは句の場合は、そのビューを参照しているステートメントのオプションまたは句に追加されますが、その効果は定義されていません。たとえば、ビュー定義に LIMIT 句が含まれているときに、独自の LIMIT 句を含むステートメントを使用しているビューから選択した場合、どの制限が適用されるか

は未定義です。この同じ原則は、SELECT キーワードに続く ALL、DISTINCT、SQL_SMALL_RESULT などのオプションや、INTO、FOR UPDATE、LOCK IN SHARE MODE、PROCEDURE などの句にも適用されます。

ビューを作成したあとに、システム変数の変更によってクエリー処理環境を変更すると、そのビューから得られる結果に影響を与える可能性があります。

```
mysql> CREATE VIEW v (mycol) AS SELECT 'abc';
Query OK, 0 rows affected (0.01 sec)

mysql> SET sql_mode = '';
Query OK, 0 rows affected (0.00 sec)

mysql> SELECT "mycol" FROM v;
+-------+
| mycol |
+-------+
| mycol |
+-------+
1 row in set (0.01 sec)

mysql> SET sql_mode = 'ANSI_QUOTES';
Query OK, 0 rows affected (0.00 sec)

mysql> SELECT "mycol" FROM v;
+-------+
| mycol |
+-------+
| abc   |
+-------+
1 row in set (0.00 sec)
```

DEFINER および SQL SECURITY 句は、そのビューを参照しているステートメントの実行時に、そのビューに対するアクセス権限を確認するときにどの MySQL アカウントを使用するかを決定します。SQL SECURITY 特性の有効な値は、DEFINER と INVOKER です。これらは、それぞれ、そのビューを定義したユーザーまたは呼び出したユーザーが必要な権限を持っている必要があることを示します。SQL SECURITY のデフォルト値は DEFINER です。

DEFINER 句に user 値を指定する場合は、'user_name'@'host_name' (GRANT ステートメントで使用されるのと同じ形式)、CURRENT_USER、または CURRENT_USER() として指定された MySQL アカウントにするようにしてください。DEFINER のデフォルト値は、CREATE VIEW ステートメントを実行するユーザーです。これは、明示的に DEFINER = CURRENT_USER を指定するのと同じです。

DEFINER 句を指定した場合は、次のルールによって有効な DEFINER ユーザーの値が決定されます。

- SUPER 権限がない場合、有効な唯一の user 値は、リテラルで指定するか、または CURRENT_USER を使用して指定した自分のアカウントです。定義者をほかのアカウントに設定することはできません。
- SUPER 権限がある場合は、構文として有効な任意のアカウント名を指定できます。そのアカウントが実際に存在しない場合は、警告が生成されます。
- 存在しない DEFINER アカウントでビューを作成することはできますが、SQL SECURITY 値が DEFINER であるが、定義者アカウントが存在しない場合は、そのビューが参照されたときにエラーが発生します。

ビューのセキュリティーの詳細は、セクション20.6「ストアドプログラムおよびビューのアクセスコントロール」を参照してください。

ビュー定義内で、CURRENT_USER は、デフォルトではそのビューの DEFINER 値を返します。SQL SECURITY INVOKER 特性を使用して定義されたビューの場合、CURRENT_USER は、そのビューの呼び出し元のアカウントを返します。ビュー内のユーザー監査については、セクション6.3.13「SQL ベースの MySQL アカウントアクティビティーの監査」を参照してください。

SQL SECURITY DEFINER 特性を使用して定義されたストアドルーチン内で、CURRENT_USER は、そのルーチンの DEFINER 値を返します。ビュー定義に CURRENT_USER の DEFINER 値が含まれている場合は、これにより、このようなルーチン内で定義されたビューも影響を受けます。

ビューの権限は、次のように確認されます。

- ビューの定義時に、ビュー作成者は、そのビューによってアクセスされるトップレベルのオブジェクトを使用するために必要な権限を持っている必要があります。たとえば、ビュー定義がテーブルカラムを参照している場合、作成者は、その定義の選択リスト内の各カラムに対する何らかの権限と、その定義内の別の場所で使用されている各カラムに対する SELECT 権限を持っている必要があります。この定義がストアドファンクション

を参照している場合は、その関数を呼び出すために必要な権限のみを確認できます。関数呼び出し時に必要な権限は、その関数が実行されるときにしか確認できません。別の呼び出しでは、その関数内の別の実行パスが選択される可能性があります。

- ビューを参照するユーザーは、そのビューにアクセスするための適切な権限 (そのビューから選択するための SELECT や、そのビューに挿入するための INSERT など) を持っている必要があります。
- ビューが参照されると、そのビューによってアクセスされるオブジェクトに対する権限が、SQL SECURITY 特性が DEFINER または INVOKER のどちらであるかに応じて、それぞれ、そのビューの DEFINER アカウントによって保持されている権限または呼び出し元に対して確認されます。
- ビューへの参照によってストアドファンクションが実行される場合、その関数内で実行されるステートメントの権限確認は、その関数の SQL SECURITY 特性が DEFINER または INVOKER のどちらであるかによって異なります。セキュリティー特性が DEFINER である場合、その関数は DEFINER アカウントの権限で実行されます。この特性が INVOKER である場合、その関数は、そのビューの SQL SECURITY 特性によって決定される権限で実行されます。

例: あるビューがストアドファンクションに依存する可能性があり、さらにその関数がほかのストアドルーチンを呼び出す可能性があります。たとえば、次のビューはストアドファンクション f() を呼び出します。

```
CREATE VIEW v AS SELECT * FROM t WHERE t.id = f(t.name);
```

f() に次のようなステートメントが含まれているとします。

```
IF name IS NULL then
  CALL p1();
ELSE
  CALL p2();
END IF;
```

f() が実行されるとき、f() 内のステートメントを実行するために必要な権限を確認する必要があります。これは、f() 内の実行パスに応じて、p1() または p2() に対する権限が必要であることを示します。これらの権限は実行時に確認する必要があり、それらの権限を持っている必要のあるユーザーは、ビュー v と関数 f() の SQL SECURITY 値によって決定されます。

ビューの DEFINER および SQL SECURITY 句は、標準 SQL への拡張です。標準 SQL では、ビューは SQL SECURITY DEFINER のルールを使用して処理されます。標準には、ビューの定義者 (これは、ビューのスキーマの所有者と同じです) はそのビューに対する該当する権限 (SELECT など) を取得し、またそれらを付与することができると記載されています。MySQL にはスキーマの「所有者」という概念がないため、MySQL では定義者を識別するための句が追加されています。DEFINER 句は、標準が備えている機能、つまり、だれがそのビューを定義したかについての永続的なレコードを備えることを目的とした拡張です。DEFINER のデフォルト値がビュー作成者のアカウントになっているのはそのためです。

オプションの ALGORITHM 句は、標準 SQL への MySQL 拡張です。これは、MySQL によるビューの処理方法に影響を与えます。ALGORITHM は、MERGE、TEMPTABLE、または UNDEFINED の 3 つの値を受け取ります。ALGORITHM 句が存在しない場合、デフォルトのアルゴリズムは UNDEFINED です。詳細は、セクション 20.5.2「ビュー処理アルゴリズム」を参照してください。

いくつかのビューは更新可能です。つまり、これらのビューを UPDATE、DELETE、INSERT などのステートメントで使用して、ベースとなるテーブルの内容を更新できます。ビューが更新可能であるためには、そのビュー内の行とベースとなるテーブル内の行の間に 1 対 1 の関係が存在する必要があります。また、ビューを更新不可能にするその他の特定の構造構文も存在します。

更新可能なビューに対して WITH CHECK OPTION 句を指定すると、select_statement 内の WHERE 句が true である行を除く行への挿入または更新を回避できます。

更新可能なビューに対する WITH CHECK OPTION 句では、そのビューが別のビューとの関連で定義されている場合、LOCAL および CASCADED キーワードによってチェックテストのスコープが決定されます。LOCAL キーワードは、CHECK OPTION を、定義されているビューのみに制限します。CASCADED を指定すると、ベースとなるビューに対するチェックも評価されます。どちらのキーワードも指定されていない場合、デフォルトは CASCADED になります。

更新可能なビューおよび WITH CHECK OPTION 句の詳細は、セクション20.5.3「更新可能および挿入可能なビュー」を参照してください。

13.1.21 DROP DATABASE 構文

```
DROP {DATABASE | SCHEMA} [IF EXISTS] db_name
```

DROP DATABASE は、データベース内のすべてのテーブルを削除したあと、そのデータベースを削除します。このステートメントには十分に注意してください。DROP DATABASE を使用するには、そのデータベースに対する DROP 権限が必要です。DROP SCHEMA は DROP DATABASE のシノニムです。

> **重要**
>
> データベースが削除されても、そのデータベースに対するユーザー権限が自動的に削除されることはありません。セクション13.7.1.4「GRANT 構文」を参照してください。

IF EXISTS は、データベースが存在しない場合にエラーが発生しないようにするために使用されます。

デフォルトデータベースが削除されると、そのデフォルトデータベースは設定解除されます (DATABASE() 関数が NULL を返します)。

シンボリックリンクされたデータベースに対して DROP DATABASE を使用した場合は、そのリンクと元のデータベースの両方が削除されます。

DROP DATABASE は、削除されたテーブルの数を返します。これは、削除された .frm ファイルの数に対応します。

DROP DATABASE ステートメントは、MySQL 自体が通常の動作中に作成する可能性のあるファイルとディレクトリを特定のデータベースディレクトリから削除します。

- 次の拡張子を持つすべてのファイル。

.BAK	.DAT	.HSH	.MRG
.MYD	.MYI	.TRG	.TRN
.db	.frm	.ibd	.ndb
.par			

- db.opt ファイル (存在する場合)。

今一覧表示されたファイルを MySQL が削除したあとに、このデータベースディレクトリ内にほかのファイルやディレクトリが残っている場合は、そのデータベースディレクトリを削除できません。この場合は、残っているすべてのファイルまたはディレクトリを手動で削除してから、再度 DROP DATABASE ステートメントを発行する必要があります。

データベースを削除しても、そのデータベース内に作成されたどの TEMPORARY テーブルも削除されません。TEMPORARY テーブルは、それらを作成したセッションが終了すると自動的に削除されます。一時テーブルを参照してください。

データベースは mysqladmin でも削除できます。セクション4.5.2「mysqladmin — MySQL サーバーの管理を行うクライアント」を参照してください。

13.1.22 DROP EVENT 構文

```
DROP EVENT [IF EXISTS] event_name
```

このステートメントは、event_name という名前のイベントを削除します。このイベントはただちにアクティブな状態を停止し、サーバーから完全に削除されます。

このイベントが存在しない場合は、エラー ERROR 1517 (HY000): Unknown event 'event_name' が発生します。これをオーバーライドし、代わりに IF EXISTS を使用して、このステートメントで存在しないイベントに対する警告が生成されるようにできます。

このステートメントには、削除されるイベントが属するスキーマに対する EVENT 権限が必要です。

13.1.23 DROP FUNCTION 構文

DROP FUNCTION ステートメントは、ストアドファンクションやユーザー定義関数 (UDF) を削除するために使用されます。

- ストアドファンクションの削除については、セクション13.1.26「DROP PROCEDURE および DROP FUNCTION 構文」を参照してください。
- ユーザー定義関数の削除については、セクション13.7.3.2「DROP FUNCTION 構文」を参照してください。

13.1.24 DROP INDEX 構文

```
DROP INDEX [ONLINE|OFFLINE] index_name ON tbl_name
    [algorithm_option | lock_option] ...

algorithm_option:
    ALGORITHM [=] {DEFAULT|INPLACE|COPY}

lock_option:
    LOCK [=] {DEFAULT|NONE|SHARED|EXCLUSIVE}
```

DROP INDEX は、テーブル tbl_name から index_name という名前のインデックスを削除します。このステートメントは、このインデックスを削除するために ALTER TABLE ステートメントにマップされます。セクション13.1.7「ALTER TABLE 構文」を参照してください。

主キーを削除するには、インデックス名は常に PRIMARY です。これは、PRIMARY が予約語であるため、引用符で囲まれた識別子として指定する必要があります。

```
DROP INDEX `PRIMARY` ON t;
```

NDB テーブルの可変幅カラム上のインデックスはオンラインで、つまり、テーブルコピーを行うことなく削除されます。このテーブルは、この操作の期間中、同じ API ノードに対するほかの操作に対してロックされますが、ほかの MySQL Cluster API ノードからのアクセスに対してはロックされません。これは、サーバーが実行できると判断した場合は常に、そのサーバーによって自動的に実行されます。これを実行するために、特殊な SQL 構文やサーバーオプションを使用する必要はありません。

MySQL Cluster では、OFFLINE キーワードを使用してインデックスをオフラインで削除できます (これにより、そのテーブルはクラスタ内のすべての API ノードに対してロックされます)。DROP OFFLINE INDEX および DROP ONLINE INDEX を管理するルールや制限は、ALTER OFFLINE TABLE ... DROP INDEX および ALTER ONLINE TABLE ... DROP INDEX の場合と同じです。ONLINE キーワードを使用して、通常はオフラインで削除されるインデックスのコピーなし削除が実行されるようにすることはできません。DROP 操作をテーブルコピーなしで実行できない場合、サーバーは ONLINE キーワードを無視します。詳細は、セクション13.1.7.2「MySQL Cluster での ALTER TABLE オンライン操作」を参照してください。

ONLINE および OFFLINE キーワードは、MySQL Cluster でのみ使用できます。これらのキーワードを標準の MySQL Server 5.6 リリースで使用しようとすると、構文エラーが発生します。ONLINE および OFFLINE キーワードは、MySQL Cluster NDB 7.3 では非推奨です。MySQL Cluster NDB 7.4 では引き続きサポートされますが、将来の MySQL Cluster リリースでは削除対象としてスケジュールされています。

MySQL 5.6.6 の時点では、ALGORITHM および LOCK 句を指定できます。これらは、テーブルコピーの方法や、インデックスが変更されている間のテーブルの読み取りと書き込みの並列性のレベルに影響を与えます。これらには、ALTER TABLE ステートメントの場合と同じ意味があります。詳細は、セクション13.1.7「ALTER TABLE 構文」を参照してください。

13.1.25 DROP LOGFILE GROUP 構文

```
DROP LOGFILE GROUP logfile_group
    ENGINE [=] engine_name
```

このステートメントは、logfile_group という名前のログファイルグループを削除します。このログファイルグループがすでに存在する必要があります。そうしないと、エラー結果が発生します。(ログファイルグループの作成については、セクション13.1.14「CREATE LOGFILE GROUP 構文」を参照してください。)

> **重要**
>
> ログファイルグループを削除する前に、そのログファイルグループを UNDO ロギングのために使用しているすべてのテーブルスペースを削除する必要があります。

必須の ENGINE 句は、削除されるログファイルグループによって使用されるストレージエンジンの名前を指定します。現在、engine_name に許可される値は NDB と NDBCLUSTER だけです。

DROP LOGFILE GROUP は、MySQL Cluster のディスクデータストレージでのみ有効です。セクション18.5.12「MySQL Cluster ディスクデータテーブル」を参照してください。

13.1.26 DROP PROCEDURE および DROP FUNCTION 構文

```
DROP {PROCEDURE | FUNCTION} [IF EXISTS] sp_name
```

このステートメントは、ストアドプロシージャーまたはストアドファンクションを削除するために使用されます。つまり、指定されたルーチンがサーバーから削除されます。このルーチンに対する ALTER ROUTINE 権限が必要です。(automatic_sp_privileges システム変数が有効になっている場合は、その権限と EXECUTE が自動的に、そのルーチンが作成されるときはルーチン作成者に付与され、そのルーチンが削除されるときは作成者から削除されます。セクション20.2.2「ストアドルーチンと MySQL 権限」を参照してください。)

IF EXISTS 句は MySQL 拡張です。これは、プロシージャーまたは関数が存在しない場合にエラーが発生しないようにします。SHOW WARNINGS で表示できる警告が生成されます。

DROP FUNCTION はまた、ユーザー定義関数を削除するためにも使用されます (セクション13.7.3.2「DROP FUNCTION 構文」を参照してください)。

13.1.27 DROP SERVER 構文

```
DROP SERVER [ IF EXISTS ] server_name
```

server_name という名前のサーバーのサーバー定義を削除します。mysql.servers テーブル内の対応する行が削除されます。このステートメントには、SUPER 権限が必要です。

テーブルのサーバーを削除しても、作成されるときにこの接続情報を使用したどの FEDERATED テーブルにも影響を与えません。セクション13.1.16「CREATE SERVER 構文」を参照してください。

DROP SERVER では、自動コミットは実行されません。

MySQL 5.6 では、使用されているロギング形式には関係なく、DROP SERVER はバイナリログに書き込まれません。

MySQL 5.6.11 でのみ、このステートメントを発行する前に、gtid_next を AUTOMATIC に設定する必要があります。(Bug #16062608、Bug #16715809、Bug #69045)

13.1.28 DROP TABLE 構文

```
DROP [TEMPORARY] TABLE [IF EXISTS]
  tbl_name [, tbl_name] ...
  [RESTRICT | CASCADE]
```

DROP TABLE は、1 つ以上のテーブルを削除します。各テーブルに対する DROP 権限が必要です。すべてのテーブルデータとテーブル定義が削除されるため、このステートメントには注意してください。引数リストで指定されているいずれかのテーブルが存在しない場合、MySQL は削除できなかった存在しないテーブルを名前で示すエラーを返しますが、リスト内の存在しているすべてのテーブルの削除も行います。

> **重要**
>
> テーブルが削除されても、そのテーブルに対するユーザー権限は自動的には削除されません。セクション13.7.1.4「GRANT 構文」を参照してください。

パーティション化されたテーブルの場合は、DROP TABLE によってテーブル定義、そのすべてのパーティション、およびそれらのパーティションに格納されていたすべてのデータが永続的に削除されることに注意してください。また、削除されたテーブルに関連付けられているパーティション化定義 (.par) ファイルも削除されます。

存在しないテーブルに対してエラーが発生しないようにするには、IF EXISTS を使用します。IF EXISTS を使用している場合は、存在しないテーブルごとに NOTE が生成されます。セクション13.7.5.41「SHOW WARNINGS 構文」を参照してください。

RESTRICT と CASCADE は、移植を容易にするために許可されています。MySQL 5.6 では、これらは何も行いません。

> **注記**
>
> DROP TABLE は、TEMPORARY キーワードが使用されていないかぎり、現在のアクティブなトランザクションを自動的にコミットします。

TEMPORARY キーワードには、次の効果があります。

- このステートメントは、TEMPORARY テーブルのみを削除します。
- このステートメントは、進行中のトランザクションを終了しません。
- アクセス権は確認されません。(TEMPORARY テーブルは、それを作成したセッションにのみ表示されるため、確認は必要ありません。)

TEMPORARY の使用は、TEMPORARY 以外のテーブルを誤って削除してしまわないようにするための適切な方法です。

13.1.29 DROP TABLESPACE 構文

```
DROP TABLESPACE tablespace_name
    ENGINE [=] engine_name
```

このステートメントは、以前に CREATE TABLESPACE を使用して作成されたテーブルスペースを削除します (セクション13.1.18「CREATE TABLESPACE 構文」を参照してください)。

> **重要**
>
> 削除されるテーブルスペースにデータファイルが含まれていてはいけません。つまり、テーブルスペースを削除できるようにするには、まず ALTER TABLESPACE ... DROP DATAFILE を使用してその各データファイルを削除する必要があります (セクション13.1.8「ALTER TABLESPACE 構文」を参照してください)。

ENGINE 句 (必須) は、このテーブルスペースによって使用されるストレージエンジンを指定します。現在、engine_name として受け入れられる値は NDB と NDBCLUSTER だけです。

DROP TABLESPACE は、MySQL Cluster のディスクデータストレージでのみ有効です。セクション18.5.12「MySQL Cluster ディスクデータテーブル」を参照してください。

13.1.30 DROP TRIGGER 構文

```
DROP TRIGGER [IF EXISTS] [schema_name.]trigger_name
```

このステートメントは、トリガーを削除します。スキーマ (データベース) 名はオプションです。スキーマが省略されている場合、このトリガーはデフォルトスキーマから削除されます。DROP TRIGGER には、このトリガーに関連付けられたテーブルに対する TRIGGER 権限が必要です。

存在しないトリガーに対してエラーが発生しないようにするには、IF EXISTS を使用します。IF EXISTS を使用している場合は、存在しないトリガーに対して NOTE が生成されます。セクション13.7.5.41「SHOW WARNINGS 構文」を参照してください。

テーブルを削除すると、そのテーブルのトリガーも削除されます。

13.1.31 DROP VIEW 構文

```
DROP VIEW [IF EXISTS]
    view_name [, view_name] ...
    [RESTRICT | CASCADE]
```

DROP VIEW は、1 つ以上のビューを削除します。各ビューに対する DROP 権限が必要です。引数リストで指定されているいずれかのビューが存在しない場合、MySQL は削除できなかった存在しないビューを名前で示すエラーを返しますが、リスト内の存在しているすべてのビューの削除も行います。

IF EXISTS 句は、存在しないビューに対してエラーが発生しないようにします。この句が指定されている場合は、存在しないビューごとに NOTE が生成されます。セクション13.7.5.41「SHOW WARNINGS 構文」を参照してください。

RESTRICT と CASCADE (指定されている場合) は解析されますが、無視されます。

13.1.32 RENAME TABLE 構文

```
RENAME TABLE tbl_name TO new_tbl_name
```

```
    [, tbl_name2 TO new_tbl_name2] ...
```

このステートメントは、1 つ以上のテーブルの名前を変更します。

名前変更の操作は原子的に実行されます。つまり、名前の変更が実行されている間、ほかのセッションはどのテーブルにもアクセスできません。たとえば、既存のテーブル old_table が存在する場合は、次のように、同じ構造を持っているが空である別のテーブル new_table を作成してから、既存のテーブルをその空のテーブルに置き換えることができます (backup_table はまだ存在していないと仮定します)。

```
CREATE TABLE new_table (...);
RENAME TABLE old_table TO backup_table, new_table TO old_table;
```

このステートメントで複数のテーブルの名前を変更する場合、名前の変更操作は左から右に実行されます。2 つのテーブル名をスワップする場合は、次のように実行できます (tmp_table はまだ存在していないと仮定します)。

```
RENAME TABLE old_table TO tmp_table,
             new_table TO old_table,
             tmp_table TO new_table;
```

2 つのデータベースが同じファイルシステム上に存在するかぎり、RENAME TABLE を使用して、あるデータベースから別のデータベースにテーブルを移動できます。

```
RENAME TABLE current_db.tbl_name TO other_db.tbl_name;
```

RENAME TABLE を使用して別のデータベースに移動されたテーブルに関連付けられたトリガーが存在する場合は、ステートメントがエラー Trigger in wrong schema で失敗します。

ビューの名前を変更して別のデータベースに移動しようとしないかぎり、RENAME TABLE はビューに対しても機能します。

名前変更されたテーブルまたはビュー専用に付与された権限は、どれも新しい名前には移行されません。それらは、手動で変更する必要があります。

RENAME を実行する場合は、ロックされたテーブルやアクティブなトランザクションが存在していてはいけません。また、元のテーブルに対する ALTER および DROP 権限と、新しいテーブルに対する CREATE および INSERT 権限も必要です。

複数テーブルの名前変更で何らかのエラーが発生した場合、MySQL はすべてをその元の状態に戻すために、名前変更されたすべてのテーブルに対して逆方向の名前変更を実行します。

RENAME を使用して TEMPORARY テーブルの名前を変更することはできません。ただし、代わりに ALTER TABLE を使用できます。

```
mysql> ALTER TABLE orig_name RENAME new_name;
```

この名前変更操作によってテーブルが別のファイルシステム上にあるデータベースに移動される場合、結果の成功はプラットフォーム固有であり、テーブルファイルを移動するために使用されるベースとなるオペレーティングシステムコールに依存します。

13.1.33 TRUNCATE TABLE 構文

```
TRUNCATE [TABLE] tbl_name
```

TRUNCATE TABLE は、テーブルを完全に空にします。これには DROP 権限が必要です。

TRUNCATE TABLE は論理的に、すべての行を削除する DELETE ステートメントや、DROP TABLE および CREATE TABLE ステートメントのシーケンスに似ています。高性能を実現するために、データを削除するための DML の方法をバイパスします。そのため、ロールバックすることができず、ON DELETE トリガーが起動されることはなく、さらに親子の外部キー関係を持つ InnoDB テーブルに対して実行することもできません。

TRUNCATE TABLE は DELETE に似ているにもかかわらず、DML ステートメントではなく DDL ステートメントとして分類されます。MySQL 5.6 では、DELETE とは次の点で異なります。

- 切り捨て操作はテーブルを削除して再作成するため、特に大きなテーブルの場合は、行を 1 つずつ削除するよりはるかに高速です。
- 切り捨て操作は暗黙的なコミットを発生させるため、ロールバックできません。

- セッションがアクティブなテーブルロックを保持している場合は、切り詰め操作を実行できません。
- TRUNCATE TABLE は、InnoDB テーブルに対して、このテーブルを参照するほかのテーブルからの何らかの FOREIGN KEY 制約が存在する場合は失敗します。同じテーブルのカラム間の外部キー制約が許可されます。
- 切り詰め操作は、削除された行数に対して、意味のある値を返しません。通常の結果は「0 rows affected」ですが、これは「情報がない」ものとして解釈してください。
- テーブル形式ファイル tbl_name.frm が有効であるかぎり、データまたはインデックスファイルが破損した場合でも、TRUNCATE TABLE を使用してテーブルを空のテーブルとして再作成できます。
- AUTO_INCREMENT 値はすべて、その開始値にリセットされます。これは、通常はシーケンス値を再利用しない MyISAM や InnoDB にも当てはまります。
- パーティション化されたテーブルで使用された場合、TRUNCATE TABLE はそのパーティション化を保持します。つまり、データおよびインデックスファイルが削除されて再作成されるのに対して、パーティション定義 (.par) ファイルは影響を受けません。
- TRUNCATE TABLE ステートメントは、ON DELETE トリガーを起動しません。

テーブルに対する TRUNCATE TABLE は、HANDLER OPEN で開かれたそのテーブルのすべてのハンドラを閉じます。

TRUNCATE TABLE は、バイナリロギングおよびレプリケーション目的のときは、DROP TABLE とそれに続く CREATE TABLE として、つまり、DML ではなく DDL として扱われます。これは、InnoDB またはほかのトランザクションストレージエンジン (そのトランザクション分離レベルがステートメントベースロギングを許可しない (READ COMMITTED または READ UNCOMMITTED)) を使用するときは、STATEMENT または MIXED ロギングモード使用時にステートメントがログに記録されず複製されなかった事実によります。(Bug #36763) ただし、InnoDB を使用するレプリケーションスレーブには依然としてすでに説明した方法で適用されます。

TRUNCATE TABLE はパフォーマンススキーマのサマリーテーブルで使用できますが、その効果は行の削除ではなく、サマリーカラムを 0 または NULL にリセットすることです。セクション22.9.9「パフォーマンススキーマサマリーテーブル」を参照してください。

13.2 データ操作ステートメント

13.2.1 CALL 構文

```
CALL sp_name([parameter[,...]])
CALL sp_name[()]
```

CALL ステートメントは、以前に CREATE PROCEDURE を使用して定義されたストアドプロシージャーを呼び出します。

引数を取らないストアドプロシージャーは、括弧なしで呼び出すことができます。つまり、CALL p() と CALL p は同等です。

CALL は、OUT または INOUT パラメータとして宣言されたパラメータを使用して、その呼び出し元に値を返すことができます。そのプロシージャーから戻るとき、クライアントプログラムは、ルーチン内で実行された最後のステートメントで影響を受けた行数を取得することもできます。SQL レベルでは、ROW_COUNT() 関数を呼び出します。C API からは、mysql_affected_rows() 関数を呼び出します。

OUT または INOUT パラメータを使用してプロシージャーから値を取得するには、ユーザー変数を使用してこのパラメータを渡し、そのプロシージャーから戻ったあとに変数の値をチェックします。(そのプロシージャーを別のストアドプロシージャーまたはストアドファンクション内から呼び出している場合は、IN または INOUT パラメータとしてルーチンパラメータまたはローカルルーチン変数を渡すこともできます。)INOUT パラメータの場合は、プロシージャーに渡す前にその値を初期化してください。次のプロシージャーには、このプロシージャーが現在のサーバーバージョンに設定する OUT パラメータと、このプロシージャーがその現在の値から 1 増分する INOUT 値が含まれています。

```
CREATE PROCEDURE p (OUT ver_param VARCHAR(25), INOUT incr_param INT)
BEGIN
  # Set value of OUT parameter
  SELECT VERSION() INTO ver_param;
  # Increment value of INOUT parameter
  SET incr_param = incr_param + 1;
END;
```

このプロシージャーを呼び出す前に、INOUT パラメータとして渡される変数を初期化します。このプロシージャーを呼び出したあと、これらの 2 つの変数の値は設定または変更されています。

```
mysql> SET @increment = 10;
mysql> CALL p(@version, @increment);
mysql> SELECT @version, @increment;
+--------------+------------+
| @version     | @increment |
+--------------+------------+
| 5.5.3-m3-log |         11 |
+--------------+------------+
```

PREPARE および EXECUTE で使用される準備済み CALL ステートメントでは、IN パラメータにプレースホルダを使用できます。OUT および INOUT パラメータの場合、プレースホルダのサポートは MySQL 5.5.3 以降で使用できます。これらの種類のパラメータは、次のように使用できます。

```
mysql> SET @increment = 10;
mysql> PREPARE s FROM 'CALL p(?, ?)';
mysql> EXECUTE s USING @version, @increment;
mysql> SELECT @version, @increment;
+--------------+------------+
| @version     | @increment |
+--------------+------------+
| 5.5.3-m3-log |         11 |
+--------------+------------+
```

MySQL 5.5.3 より前は、OUT または INOUT パラメータにプレースホルダのサポートは使用できません。OUT および INOUT パラメータに対するこの制限を回避するために、プレースホルダの使用は避けてください。代わりに、ユーザー変数を CALL ステートメント自体で参照し、EXECUTE ステートメントでは指定しないでください。

```
mysql> SET @increment = 10;
mysql> PREPARE s FROM 'CALL p(@version, @increment)';
mysql> EXECUTE s;
mysql> SELECT @version, @increment;
+--------------+------------+
| @version     | @increment |
+--------------+------------+
| 5.5.0-m2-log |         11 |
+--------------+------------+
```

CALL SQL ステートメントを使用して、結果セットを生成するストアドプロシージャーを実行する C プログラムを記述するには、CLIENT_MULTI_RESULTS フラグが有効になっている必要があります。これは、各 CALL によって、プロシージャー内で実行されるステートメントによって返される可能性のある結果セットに加えて、呼び出しステータスを示すための結果が返されるためです。CLIENT_MULTI_RESULTS は、CALL が、準備済みステートメントを含むストアドプロシージャーを実行するために使用される場合にも有効になっている必要があります。このようなプロシージャーがいつロードされるかや、これらのステートメントによって結果セットが生成されるかどうかを特定することはできないため、これらを想定する必要があります。

CLIENT_MULTI_RESULTS は、mysql_real_connect() を呼び出すときに、CLIENT_MULTI_RESULTS フラグ自体を渡すことによって明示的に、または CLIENT_MULTI_STATEMENTS を渡すことによって暗黙的に有効にする (これによって CLIENT_MULTI_RESULTS も有効になります) ことができます。MySQL 5.6 では、CLIENT_MULTI_RESULTS はデフォルトで有効にされています。

mysql_query() または mysql_real_query() を使用して実行された CALL ステートメントの結果を処理するには、それ以上結果が存在するかどうかを判定するために mysql_next_result() を呼び出すループを使用してください。例については、セクション23.7.17「複数ステートメント実行の C API サポート」を参照してください。

MySQL インタフェースを備える言語で記述されたプログラムの場合は、CALL ステートメントからの OUT または INOUT パラメータの結果を直接取得するためのネイティブメソッドが MySQL 5.5.3 より前には存在しません。これらのパラメータ値を取得するには、CALL ステートメントでプロシージャーにユーザー定義変数を渡したあと、SELECT ステートメントを実行して変数値を含む結果セットを生成してください。INOUT パラメータを処理するには、CALL の前に、対応するユーザー変数を、プロシージャーに渡される値に設定するステートメントを実行してください。

次の例は、先に説明した、OUT パラメータと INOUT パラメータを含むストアドプロシージャー p の手法 (エラーチェックはなし) を示しています。

```
mysql_query(mysql, "SET @increment = 10");
mysql_query(mysql, "CALL p(@version, @increment)");
mysql_query(mysql, "SELECT @version, @increment");
result = mysql_store_result(mysql);
```

```
row = mysql_fetch_row(result);
mysql_free_result(result);
```

前のコードが実行されたあと、row[0] と row[1] にはそれぞれ、@version と @increment の値が含まれています。

MySQL 5.6 では、C プログラムは準備済みステートメントインタフェースを使用して CALL ステートメントを実行し、OUT および INOUT パラメータにアクセスできます。これは、それ以上結果が存在するかどうかを判定するために mysql_stmt_next_result() を呼び出すループを使用して CALL ステートメントの結果を処理することにより行われます。例については、セクション23.7.20「C API のプリペアド CALL ステートメントのサポート」を参照してください。MySQL インタフェースを備える言語は、準備済み CALL ステートメントを使用して、OUT および INOUT プロシージャーパラメータを直接取得できます。

MySQL 5.6.6 以降では、ストアドプログラムによって参照されるオブジェクトへのメタデータ変更が検出され、そのプログラムが次に実行されときに、影響を受けるステートメントの自動再解析が行われるようになります。詳細については、セクション8.9.4「プリペアドステートメントおよびストアドプログラムのキャッシュ」を参照してください。

13.2.2 DELETE 構文

DELETE は、テーブルの行を削除する DML ステートメントです。

単一テーブル構文

```
DELETE [LOW_PRIORITY] [QUICK] [IGNORE] FROM tbl_name
    [PARTITION (partition_name,...)]
    [WHERE where_condition]
    [ORDER BY ...]
    [LIMIT row_count]
```

DELETE ステートメントは、tbl_name の行を削除し、削除された行数を返します。削除された行数をチェックするには、セクション12.14「情報関数」で説明されているROW_COUNT() 関数を呼び出します。

メインの句

オプションの WHERE 句内の条件は、どの行を削除するかを識別します。WHERE 句がない場合は、すべての行が削除されます。

where_condition は、削除される各行に対して true に評価される式です。これは、セクション13.2.9「SELECT 構文」で説明されているように指定されます。

ORDER BY 句が指定されている場合は、指定されている順序で行が削除されます。LIMIT 句は、削除できる行数に制限を設定します。これらの句は単一テーブルの削除に適用されますが、複数テーブルの削除には適用されません。

複数テーブル構文

```
DELETE [LOW_PRIORITY] [QUICK] [IGNORE]
    tbl_name[.*] [, tbl_name[.*]] ...
    FROM table_references
    [WHERE where_condition]
```

または:

```
DELETE [LOW_PRIORITY] [QUICK] [IGNORE]
    FROM tbl_name[.*] [, tbl_name[.*]] ...
    USING table_references
    [WHERE where_condition]
```

権限

テーブルから行を削除するには、そのテーブルに対する DELETE 権限が必要です。WHERE 句で指定されているカラムなどの、読み取られるだけのカラムに対しては、SELECT 権限のみが必要です。

パフォーマンス

削除された行数を知る必要がない場合、テーブルを空にするには、WHERE 句のない DELETE ステートメントより TRUNCATE TABLE ステートメントの方が高速です。DELETE とは異なり、TRUNCATE TABLE はトランザク

ション内で、またはそのテーブルがロックされている場合は使用できません。セクション13.1.33「TRUNCATE TABLE 構文」およびセクション13.3.5「LOCK TABLES および UNLOCK TABLES 構文」を参照してください。

削除操作の速度はまた、セクション8.2.2.3「DELETE ステートメントの速度」で説明されている要因によって影響を受ける可能性もあります。

特定の DELETE ステートメントに時間がかかりすぎないようにするために、DELETE の MySQL 固有の LIMIT row_count 句は、削除される行の最大数を指定します。削除する行数がこの制限を超えている場合は、影響を受ける行数が LIMIT 値を下回るまで DELETE ステートメントを繰り返します。

サブクエリー

現在、テーブルから削除し、さらにサブクエリーで同じテーブルから選択することはできません。

パーティション化されたテーブル

MySQL 5.6.2 から、DELETE は、削除される行を選択する 1 つ以上のパーティションまたはサブパーティション (またはその両方) の名前のカンマ区切りリストを含む PARTITION オプションを使用した明示的なパーティション選択をサポートしています。このリストに含まれていないパーティションは無視されます。p0 という名前のパーティションを含むパーティション化されたテーブル t がある場合、ステートメント DELETE FROM t PARTITION (p0) の実行には、このテーブルに対して ALTER TABLE t TRUNCATE PARTITION (p0) を実行するのと同じ効果があります。どちらの場合も、パーティション p0 内のすべての行が削除されます。

PARTITION は、WHERE 条件とともに使用できます。その場合、この条件は、リストされているパーティション内の行に対してのみテストされます。たとえば、DELETE FROM t PARTITION (p0) WHERE c < 5 は、条件 c < 5 が true であるパーティション p0 の行のみを削除します。ほかのパーティション内の行はチェックされないため、DELETE によって影響を受けません。

PARTITION オプションはまた、複数テーブルの DELETE ステートメントでも使用できます。このようなオプションを、FROM オプションで指定されているテーブルごとに最大 1 つ使用できます。

詳細および例については、セクション19.5「パーティション選択」を参照してください。

自動インクリメントカラム

AUTO_INCREMENT カラムに最大値を含む行を削除した場合、その値は、MyISAM または InnoDB テーブルには再利用されません。autocommit モードで DELETE FROM tbl_name (WHERE 句はなし) を使用してテーブル内のすべての行を削除した場合、そのシーケンスは、InnoDB と MyISAM を除くすべてのストレージエンジンに対して開始されます。セクション14.6.5「InnoDB での AUTO_INCREMENT 処理」で説明されているように、InnoDB テーブルに対しては、この動作の例外がいくつかあります。

MyISAM テーブルの場合は、マルチカラムキー内の AUTO_INCREMENT セカンダリカラムを指定できます。この場合は、シーケンスの先頭から削除された値の再利用が MyISAM テーブルに対しても実行されます。セクション3.6.9「AUTO_INCREMENT の使用」を参照してください。

修飾子

DELETE ステートメントは、次の修飾子をサポートします。

- LOW_PRIORITY を指定した場合、サーバーは、ほかのどのクライアントもそのテーブルから読み取らなくなるまで DELETE の実行を遅延させます。これは、テーブルレベルロックのみを使用するストレージエンジン (MyISAM、MEMORY、および MERGE) にのみ影響を与えます。
- MyISAM テーブルでは、QUICK キーワードを使用した場合、ストレージエンジンは削除中にインデックスリーフをマージしません。これにより、一部の種類の削除操作が高速化される可能性があります。
- IGNORE キーワードを指定すると、MySQL は行削除プロセス中のエラーを無視します。(解析の段階で検出されたエラーは、通常の方法で処理されます。)IGNORE の使用のために無視されたエラーは、警告として返されます。

削除の順序

DELETE ステートメントに ORDER BY 句が含まれている場合は、この句で指定されている順序で行が削除されます。これは、主に LIMIT と組み合わせて使用した場合に有効です。たとえば、次のステートメントは WHERE 句に一致する行を見つけ、それらを timestamp_column でソートしたあと、最初の (もっとも古い) 行を削除します。

```
DELETE FROM somelog WHERE user = 'jcole'
ORDER BY timestamp_column LIMIT 1;
```

ORDER BY はまた、参照整合性の違反を回避するために必要な順序で行を削除する場合も役立ちます。

InnoDB テーブル

大きなテーブルから多数の行を削除する場合は、InnoDB テーブルに対するロックテーブルのサイズを超える可能性があります。この問題を回避するために、または単にテーブルがロックされたままになる時間を最小限に抑えるために、DELETE をまったく使用しない次の方法が有効な場合があります。

1. 削除されない行を選択して、元のテーブルと同じ構造を持つ空のテーブルに格納します。

   ```
   INSERT INTO t_copy SELECT * FROM t WHERE ... ;
   ```

2. RENAME TABLE を使用して元のテーブルを原子的に移動したあと、コピーの名前を元の名前に変更します。

   ```
   RENAME TABLE t TO t_old, t_copy TO t;
   ```

3. 元のテーブルを削除します。

   ```
   DROP TABLE t_old;
   ```

RENAME TABLE が実行されている間、関連するテーブルにはほかのどのセッションからもアクセスできないため、名前変更の操作は並列性の問題に制約されません。セクション13.1.32「RENAME TABLE 構文」を参照してください。

MyISAM テーブル

MyISAM テーブルでは、削除された行はリンクリスト内に保持され、以降の INSERT 操作は古い行の位置を再利用します。未使用領域を再利用し、ファイルサイズを減らすには、OPTIMIZE TABLE ステートメントまたは myisamchk ユーティリティーを使用してテーブルを再編成します。OPTIMIZE TABLE の方が使い方は簡単ですが、myisamchk の方が高速です。セクション13.7.2.4「OPTIMIZE TABLE 構文」およびセクション4.6.3「myisamchk — MyISAM テーブルメンテナンスユーティリティー」を参照してください。

QUICK 修飾子は、削除操作でインデックスリーフがマージされるかどうかに影響を与えます。DELETE QUICK は、削除された行のインデックス値が、あとで挿入された行の同様のインデックス値に置き換えられるアプリケーションで、特に役立ちます。この場合、削除された値によって残された穴は再利用されます。

DELETE QUICK は、削除された値によって、新しい挿入が再度発生するインデックス値の範囲全体にわたって空きのあるインデックスブロックが残される場合には役立ちません。この場合は、QUICK を使用すると、再利用されないままのインデックスで領域が浪費される可能性があります。このようなシナリオの例を次に示します。

1. インデックス付き AUTO_INCREMENT カラムを含むテーブルを作成します。
2. このテーブルに多数の行を挿入します。各挿入によって、インデックスの先頭に追加されるインデックス値が生成されます。
3. DELETE QUICK を使用して、カラムの範囲の最後にある行のブロックを削除します。

このシナリオでは、削除されたインデックス値に関連付けられたインデックスブロックに空きができますが、QUICK が使用されているため、ほかのインデックスブロックにはマージされません。新しい挿入が発生したとき、新しい行には削除された範囲内のインデックス値が含まれていないため、これらのインデックスブロックは空きがあるままになります。さらに、削除された一部のインデックス値が偶然に空きのあるブロック内か、またはその隣のインデックスブロックに含まれていないかぎり、あとで QUICK なしで DELETE を使用した場合でも空きがあるままになります。これらの状況で未使用のインデックス領域を再利用するには、OPTIMIZE TABLE を使用します。

テーブルから多数の行を削除しようとしている場合は、DELETE QUICK に続けて OPTIMIZE TABLE を使用した方が高速になることがあります。これにより、インデックスブロックの多数のマージ操作が実行されるのではなく、インデックスが再構築されます。

複数テーブルの削除

WHERE 句内の条件に応じて 1 つ以上のテーブルから行を削除するには、DELETE ステートメントで複数のテーブルを指定できます。複数テーブルの DELETE では、ORDER BY または LIMIT を使用できません。セクション13.2.9.2「JOIN 構文」で説明されているように、table_references 句は、結合に含まれるテーブルをリストします。

最初の複数テーブル構文では、FROM 句の前にリストされているテーブルの一致する行のみが削除されます。2 番目の複数テーブル構文では、USING 句の前にある FROM 句にリストされているテーブルの一致する行のみが削除されます。その効果は、多数のテーブルの行を同時に削除し、さらに検索にのみ使用される追加のテーブルを指定できることです。

```
DELETE t1, t2 FROM t1 INNER JOIN t2 INNER JOIN t3
WHERE t1.id=t2.id AND t2.id=t3.id;
```

または:

```
DELETE FROM t1, t2 USING t1 INNER JOIN t2 INNER JOIN t3
WHERE t1.id=t2.id AND t2.id=t3.id;
```

これらのステートメントは、削除する行を検索するときに 3 つのすべてのテーブルを使用しますが、テーブル t1 と t2 の一致する行のみを削除します。

前の例では INNER JOIN を使用していますが、複数テーブルの DELETE ステートメントは、SELECT ステートメント内で許可されているほかの型の結合 (LEFT JOIN など) を使用できます。たとえば、t1 内に存在する行で t2 内に一致するものがない行を削除するには、LEFT JOIN を使用します。

```
DELETE t1 FROM t1 LEFT JOIN t2 ON t1.id=t2.id WHERE t2.id IS NULL;
```

この構文では、Access との互換性のために、各 tbl_name のあとに .* が許可されます。

外部キー制約が存在する InnoDB テーブルを含む、複数テーブルの DELETE ステートメントを使用した場合は、MySQL オプティマイザが、それらの親子関係の順序とは異なる順序でテーブルを処理する可能性があります。この場合、このステートメントは失敗し、ロールバックされます。代わりに、1 つのテーブルから削除したあと、InnoDB が提供する ON DELETE 機能を使用して、ほかのテーブルがそれに応じて変更されるようにしてください。

> 注記
>
> テーブルのエイリアスを宣言した場合は、テーブルを参照するときにそのエイリアスを使用する必要があります。
>
> ```
> DELETE t1 FROM test AS t1, test2 WHERE ...
> ```

複数テーブルの DELETE 内のテーブルエイリアスは、そのステートメントの table_references 部分でのみ宣言するようにしてください。それ以外の場所では、エイリアス参照が許可されますが、エイリアス宣言は許可されません。

正しい:

```
DELETE a1, a2 FROM t1 AS a1 INNER JOIN t2 AS a2
WHERE a1.id=a2.id;

DELETE FROM a1, a2 USING t1 AS a1 INNER JOIN t2 AS a2
WHERE a1.id=a2.id;
```

正しくない:

```
DELETE t1 AS a1, t2 AS a2 FROM t1 INNER JOIN t2
WHERE a1.id=a2.id;

DELETE FROM t1 AS a1, t2 AS a2 USING t1 INNER JOIN t2
WHERE a1.id=a2.id;
```

13.2.3 DO 構文

```
DO expr [, expr] ...
```

DO は式を実行しますが、結果は何も返しません。ほとんどの点で、DO は SELECT expr, ... の短縮形ですが、その結果に関心がない場合は少し高速であるという利点があります。

DO は主に、副作用がある関数 (RELEASE_LOCK() など) で役立ちます。

例: この SELECT ステートメントは一時停止しますが、結果セットの生成も行います。

```
mysql> SELECT SLEEP(5);
```

```
+----------+
| SLEEP(5) |
+----------+
|        0 |
+----------+
1 row in set (5.02 sec)
```

それに対して、DO は、結果セットを生成することなく一時停止します。

```
mysql> DO SLEEP(5);
Query OK, 0 rows affected (4.99 sec)
```

これは、たとえば、結果セットを生成するステートメントを禁止しているストアドファンクションまたはトリガーで役立つ場合があります。

DO は式を実行するだけです。SELECT を使用できるすべての場合に使用できるわけではありません。たとえば、DO id FROM t1 は、テーブルを参照するため無効です。

13.2.4 HANDLER 構文

```
HANDLER tbl_name OPEN [ [AS] alias]

HANDLER tbl_name READ index_name { = | <= | >= | < | > } (value1,value2,...)
    [ WHERE where_condition ] [LIMIT ... ]
HANDLER tbl_name READ index_name { FIRST | NEXT | PREV | LAST }
    [ WHERE where_condition ] [LIMIT ... ]
HANDLER tbl_name READ { FIRST | NEXT }
    [ WHERE where_condition ] [LIMIT ... ]

HANDLER tbl_name CLOSE
```

HANDLER ステートメントは、テーブルストレージエンジンインタフェースへの直接アクセスを提供します。これは、InnoDB および MyISAM テーブルに使用できます。

HANDLER ... OPEN ステートメントはテーブルを開き、それを以降の HANDLER ... READ ステートメントを使用してアクセス可能にします。このテーブルオブジェクトはほかのセッションによって共有されておらず、このセッションが HANDLER ... CLOSE を呼び出すか、またはこのセッションが終了するまでクローズされません。エイリアスを使用してテーブルを開いた場合は、その開かれたテーブルへのほかの HANDLER ステートメントによるそれ以降の参照は、テーブル名ではなくエイリアスを使用する必要があります。

最初の HANDLER ... READ 構文は、指定されたインデックスが特定の値を満たし、かつ WHERE 条件が満たされている行をフェッチします。マルチカラムインデックスがある場合は、インデックスカラム値をカンマ区切りリストとして指定します。インデックス内のすべてのカラムの値を指定するか、またはインデックスカラムの左端のプリフィクスの値を指定します。インデックス my_idx に、col_a、col_b、および col_c という名前の 3 つのカラムがその順序で含まれているとします。HANDLER ステートメントは、そのインデックス内の 3 つのすべてのカラム、または左端のプリフィクス内のカラムの値を指定できます。例:

```
HANDLER ... READ my_idx = (col_a_val,col_b_val,col_c_val) ...
HANDLER ... READ my_idx = (col_a_val,col_b_val) ...
HANDLER ... READ my_idx = (col_a_val) ...
```

HANDLER インタフェースを使用してテーブルの PRIMARY KEY を参照するには、引用符で囲まれた識別子 `PRIMARY` を使用します。

```
HANDLER tbl_name READ `PRIMARY` ...
```

2 番目の HANDLER ... READ 構文は、WHERE 条件に一致するインデックス順序でテーブルの行をフェッチします。

3 番目の HANDLER ... READ 構文は、WHERE 条件に一致する自然な行順序でテーブルの行をフェッチします。これは、フルテーブルスキャンが望ましい場合は、HANDLER tbl_name READ index_name より高速です。自然な行順序とは、行が MyISAM テーブルデータファイル内に格納されている順序のことです。このステートメントは InnoDB テーブルに対しても機能しますが、個別のデータファイルが存在しないため、このような概念はありません。

LIMIT 句を使用しない場合は、すべての形式の HANDLER ... READ が単一行 (使用可能な場合) をフェッチします。特定の行数を返すには、LIMIT 句を含めます。その構文は、SELECT ステートメントの場合と同じです。セクション13.2.9「SELECT 構文」を参照してください。

HANDLER ... CLOSE は、HANDLER ... OPEN でオープンされたテーブルをクローズします。

通常の SELECT ステートメントの代わりに HANDLER インタフェースを使用する理由として、次のいくつかがあります。

- HANDLER は SELECT より高速です。
  - HANDLER ... OPEN に対して、指定されたストレージエンジンハンドラオブジェクトが割り当てられます。このオブジェクトは、そのテーブルに対する以降の HANDLER ステートメントに再利用されます。ステートメントごとに再初期化する必要はありません。
  - 関連する解析が少なくなります。
  - オプティマイザまたはクエリーチェックのオーバーヘッドがありません。
  - ハンドラインタフェースは (たとえば、ダーティー読み取りが許可されるような) データの整合性のある外観を提供する必要がないため、ストレージエンジンは、SELECT が通常は許可しない最適化を使用できます。
- HANDLER によって、ISAM に似た低レベルのインタフェースを使用する MySQL アプリケーションへの移植が容易になります。(キー値格納パラダイムを使用するアプリケーションを適応させるための代替手段については、セクション14.18「InnoDB と memcached の統合」を参照してください。)
- HANDLER を使用すると、SELECT では実現が困難な (または、不可能でさえある) 方法でデータベースをたどることができます。HANDLER インタフェースは、データベースに対話型ユーザーインタフェースを提供するアプリケーションの操作時にデータを調べるためのより自然な方法です。

HANDLER は、やや低レベルのステートメントです。たとえば、一貫性が提供されません。つまり、HANDLER ... OPEN はテーブルのスナップショットを作成せず、テーブルのロックも行いません。これは、HANDLER ... OPEN ステートメントが発行されたあと、テーブルデータを (現在のセッションまたはその他のセッションで) 変更することができ、これらの変更が HANDLER ... NEXT または HANDLER ... PREV スキャンに部分的にしか表示されない可能性があることを示します。

開かれたハンドラを閉じ、再度開くようにマークすることができます。その場合、このハンドラはテーブル内の位置を失います。これは、次の両方の状況が当てはまる場合に発生します。

- このハンドラのテーブルに対して、いずれかのセッションが FLUSH TABLES または DDL ステートメントを実行している。
- このハンドラを開いているセッションが、テーブルを使用する HANDLER 以外のステートメントを実行している。

テーブルに対する TRUNCATE TABLE は、HANDLER OPEN で開かれたそのテーブルのすべてのハンドラを閉じます。

FLUSH TABLES tbl_name WITH READ LOCK でフラッシュされたテーブルが HANDLER で開かれた場合、そのハンドラは暗黙的にフラッシュされ、その位置を失います。

HANDLER は、パーティション化されたテーブルではサポートされません。

13.2.5 INSERT 構文

```
INSERT [LOW_PRIORITY | DELAYED | HIGH_PRIORITY] [IGNORE]
    [INTO] tbl_name
    [PARTITION (partition_name,...)]
    [(col_name,...)]
    {VALUES | VALUE} ({expr | DEFAULT},...),(...),...
    [ ON DUPLICATE KEY UPDATE
      col_name=expr
        [, col_name=expr] ... ]
```

または:

```
INSERT [LOW_PRIORITY | DELAYED | HIGH_PRIORITY] [IGNORE]
    [INTO] tbl_name
    [PARTITION (partition_name,...)]
    SET col_name={expr | DEFAULT}, ...
    [ ON DUPLICATE KEY UPDATE
      col_name=expr
        [, col_name=expr] ... ]
```

または:

```
INSERT [LOW_PRIORITY | HIGH_PRIORITY] [IGNORE]
    [INTO] tbl_name
    [PARTITION (partition_name,...)]
    [(col_name,...)]
    SELECT ...
    [ ON DUPLICATE KEY UPDATE
      col_name=expr
        [, col_name=expr] ... ]
```

INSERT は、既存のテーブルに新しい行を挿入します。このステートメントの INSERT ... VALUES および INSERT ... SET 形式は、明示的に指定された値に基づいて行を挿入します。INSERT ... SELECT 形式は、別の 1 つまたは複数のテーブルから選択された行を挿入します。INSERT ... SELECT については、セクション 13.2.5.1「INSERT ... SELECT 構文」でさらに詳細に説明されています。

MySQL 5.6.2 以降では、パーティション化されたテーブルに挿入する場合、どのパーティションおよびサブパーティションが新しい行を受け入れるかを制御できます。PARTITION オプションは、テーブルの 1 つ以上のパーティションまたはサブパーティション (またはその両方) の名前のカンマ区切りリストを受け取ります。特定の INSERT ステートメントによって挿入される行がリストされているいずれかのパーティションに一致しない場合、INSERT ステートメントは Found a row not matching the given partition set エラーで失敗します。詳細および例については、セクション19.5「パーティション選択」を参照してください。

古い行を上書きするには、INSERT の代わりに REPLACE を使用できます。REPLACE は、古い行を複製する一意のキー値を含む新しい行の処理において INSERT IGNORE に相当するものです。新しい行は、破棄されるのではなく、古い行を置き換えるために使用されます。セクション13.2.8「REPLACE 構文」を参照してください。

tbl_name は、行が挿入されるテーブルです。このステートメントによって値が提供されるカラムは、次のように指定できます。

- テーブル名のあとにカラム名のカンマ区切りリストを指定できます。この場合は、指定された各カラムの値を VALUES リストまたは SELECT ステートメントで指定する必要があります。
- INSERT ... VALUES または INSERT ... SELECT にカラム名のリストを指定しない場合は、テーブル内のすべてのカラムの値を VALUES リストまたは SELECT ステートメントで指定する必要があります。テーブル内のカラムの順序がわからない場合は、DESCRIBE tbl_name を使用して見つけます。
- SET 句は、カラム名を明示的に示します。

カラム値は、次のいくつかの方法で指定できます。

- 厳密な SQL モードで実行していない場合、値が明示的に指定されていないカラムはすべて、デフォルトの (明示的または暗黙的な) 値に設定されます。たとえば、テーブル内のすべてのカラムを指定していないカラムリストを指定した場合、指定されていないカラムはそのデフォルト値に設定されます。デフォルト値の割り当てについては、セクション11.6「データ型デフォルト値」で説明されています。セクション1.7.3.3「無効データの制約」も参照してください。

  デフォルト値が含まれていないすべてのカラムの値を明示的に指定しないかぎり、INSERT ステートメントでエラーが生成されるようにする場合は、厳密モードを使用するようにしてください。セクション5.1.7「サーバー SQL モード」を参照してください。

- カラムを明示的にそのデフォルト値に設定するには、キーワード DEFAULT を使用します。これにより、テーブル内の各カラムの値が含まれていない不完全な VALUES リストを書かなくても済むため、いくつかのカラムを除くすべてのカラムに値を割り当てる INSERT ステートメントの記述が容易になります。そうでない場合は、VALUES リスト内の各値に対応するカラム名のリストを書き出す必要があります。

  また、特定のカラムのデフォルト値を生成する式で使用できるより一般的な形式として DEFAULT(col_name) を使用することもできます。

- カラムリストと VALUES リストの両方が空である場合、INSERT は、各カラムがそのデフォルト値に設定された行を作成します。

  ```
  INSERT INTO tbl_name () VALUES();
  ```

  厳密モードでは、いずれかのカラムにデフォルト値が含まれていない場合、エラーが発生します。それ以外の場合、MySQL は、明示的に定義されたデフォルト値が含まれていないすべてのカラムに対して暗黙のデフォルト値を使用します。

- 式 expr を指定して、カラム値を指定できます。これには、式の型がカラムの型に一致しない場合は型変換が行われる可能性があり、特定の値の変換によって、データ型に応じて異なる値が挿入されることがあります。たとえば、文字列 '1999.0e-2' を INT、FLOAT、DECIMAL(10,6)、または YEAR カラムに挿入すると、それぞ

れ、値 1999、19.9921、19.992100、および 1999 が挿入されます。INT および YEAR カラムに格納される値が 1999 である理由は、文字列から整数への変換では、その文字列の最初の、有効な整数または年と見なすことができる部分だけが調べられるためです。浮動小数点および固定小数点数カラムの場合、文字列から浮動小数点への変換では、文字列全体を有効な浮動小数点値と見なします。

式 expr は、以前に値リスト内に設定された任意のカラムを参照できます。たとえば、次のステートメントは、col2 の値が、前に割り当てられている col1 を参照しているため実行可能です。

```
INSERT INTO tbl_name (col1,col2) VALUES(15,col1*2);
```

ただし、次のステートメントは、col1 の値が、col1 のあとに割り当てられている col2 を参照しているため正当ではありません。

```
INSERT INTO tbl_name (col1,col2) VALUES(col2*2,15);
```

1 つの例外として、AUTO_INCREMENT 値を含むカラムがあります。AUTO_INCREMENT 値はほかの値の割り当てのあとに生成されるため、割り当て内の AUTO_INCREMENT カラムへの参照はすべて 0 を返します。

VALUES 構文を使用する INSERT ステートメントは複数の行を挿入できます。これを行うには、それぞれが括弧で囲まれ、カンマで区切られた、カラム値の複数のリストを含めます。例:

```
INSERT INTO tbl_name (a,b,c) VALUES(1,2,3),(4,5,6),(7,8,9);
```

各行の値リストは、括弧で囲まれている必要があります。次のステートメントは、リスト内の値の数がカラム名の数に一致しないため不正です。

```
INSERT INTO tbl_name (a,b,c) VALUES(1,2,3,4,5,6,7,8,9);
```

このコンテキストでは、VALUE は VALUES のシノニムです。どちらも、値リストの数については何も示しておらず、値リストが 1 つの場合でも複数の場合でも使用できます。

INSERT に関して影響を受けた行の値は、ROW_COUNT() 関数 (セクション12.14「情報関数」を参照してください) または mysql_affected_rows() C API 関数 (セクション23.7.7.1「mysql_affected_rows()」を参照してください) を使用して取得できます。

INSERT ... VALUES ステートメントを複数の値リストまたは INSERT ... SELECT とともに使用した場合、このステートメントは、次の形式の情報文字列を返します。

```
Records: 100 Duplicates: 0 Warnings: 0
```

Records は、このステートメントによって処理された行数を示します。(これは、Duplicates が 0 以外であることがあるため、必ずしも実際に挿入された行数ではありません。)Duplicates は、何らかの既存の一意のインデックス値を複製しているために挿入できなかった行数を示します。Warnings は、何らかの点で問題があったカラム値を挿入するための試行回数を示します。警告は、次のいずれかの条件で発生する場合があります。

- NOT NULL として宣言されているカラムへの NULL の挿入。複数行の INSERT ステートメントまたは INSERT INTO ... SELECT ステートメントの場合、このカラムは、そのカラムデータ型の暗黙のデフォルト値に設定されます。これは、数値型では 0、文字列型では空の文字列 (''')、および日付と時間型では「0」の値です。サーバーは SELECT からの結果セットを検査して、それが単一行を返すかどうかを確認しないため、INSERT INTO ... SELECT ステートメントは複数行の挿入と同じ方法で処理されます。(単一行の INSERT の場合は、NULL が NOT NULL カラムに挿入されても警告は発生しません。代わりに、このステートメントがエラーで失敗します。)
- 数値カラムの、そのカラムの範囲外にある値への設定。この値は、その範囲のもっとも近い端点にクリップされます。
- 数値カラムへの '10.34 a' などの値の割り当て。後続の非数値のテキストは取り除かれ、残りの数値部分が挿入されます。文字列値に先頭の数値部分が含まれていない場合、このカラムは 0 に設定されます。
- 文字列カラム (CHAR、VARCHAR、TEXT、または BLOB) への、そのカラムの最大長を超える文字列の挿入。この値は、そのカラムの最大長に切り捨てられます。
- 日付または時間カラムへの、そのデータ型として不正な値の挿入。このカラムは、その型の適切な 0 の値に設定されます。

C API を使用している場合は、mysql_info() 関数を呼び出すことによって情報文字列を取得できます。セクション23.7.7.35「mysql_info()」を参照してください。

INSERT で AUTO_INCREMENT カラムを含むテーブルに行を挿入した場合、そのカラムに使用された値は SQL の LAST_INSERT_ID() 関数を使用して検索できます。C API 内からは、mysql_insert_id() 関数を使用します。

> **注記**
>
> これらの 2 つの関数が、必ずしも同じ動作を行うとは限りません。AUTO_INCREMENT カラムに関連した INSERT ステートメントの動作については、セクション12.14「情報関数」およびセクション23.7.7.37「mysql_insert_id()」でさらに詳細に説明されています。

INSERT ステートメントは、次の修飾子をサポートします。

- DELAYED キーワードを使用した場合は、挿入される 1 つまたは複数の行をサーバーがバッファーに配置するため、INSERT DELAYED ステートメントを発行しているクライアントはただちに続行できます。そのテーブルが使用中である場合、サーバーはそれらの行を保持します。そのテーブルが未使用である場合、サーバーは行の挿入を開始する一方、そのテーブルに対する新しい読み取り要求が存在するかどうかを定期的にチェックします。存在する場合は、そのテーブルがふたたび未使用になるまで、遅延された行のキューは中断されます。セクション13.2.5.2「INSERT DELAYED 構文」を参照してください。

  DELAYED は、INSERT ... SELECT または INSERT ... ON DUPLICATE KEY UPDATE では無視されます。

  DELAYED はまた、テーブルやトリガーにアクセスする関数を使用しているか、または関数やトリガーから呼び出された INSERT でも無視されます。

  > **注記**
  >
  > MySQL 5.6.6 現在、INSERT DELAYED は非推奨であり、将来のリリースで削除されます。代わりに INSERT (DELAYED を付けない) を使用してください。

- LOW_PRIORITY キーワードを使用した場合、INSERT の実行は、ほかのどのクライアントもそのテーブルから読み取らなくなるまで遅延されます。これには、既存のクライアントが読み取っている間や、INSERT LOW_PRIORITY ステートメントが待機している間に読み取りを開始したほかのクライアントが含まれます。そのため、読み取り負荷の高い環境では、INSERT LOW_PRIORITY ステートメントを発行したクライアントが非常に長い時間 (場合によっては無期限に) 待機することになるおそれがあります。(これは、クライアントをただちに続行できるようにする INSERT DELAYED とは対照的です。)MyISAM テーブルで LOW_PRIORITY を使用すると、並列挿入が無効になるため、通常はこれを行わないようにしてください。セクション8.10.3「同時挿入」を参照してください。

  HIGH_PRIORITY を指定すると、サーバーが --low-priority-updates オプションで起動されている場合に、その効果がオーバーライドされます。また、同時挿入も使用されなくなります。セクション8.10.3「同時挿入」を参照してください。

  LOW_PRIORITY と HIGH_PRIORITY は、テーブルレベルのロックのみを使用するストレージエンジン (MyISAM、MEMORY、MERGE など) にのみ影響を与えます。

- IGNORE キーワードを使用した場合、INSERT ステートメントの実行中に発生したエラーは無視されます。たとえば、IGNORE を使用しない場合は、テーブル内の既存の UNIQUE インデックスまたは PRIMARY KEY 値を複製する行によって重複キーエラーが発生し、このステートメントは中止されます。IGNORE を指定すると、その行が破棄され、エラーは発生しません。代わりに、無視されたエラーが警告を生成する可能性がありますが、重複キーエラーは生成しません。

  IGNORE には、特定の値に一致するパーティションが見つからないパーティション化されたテーブルへの挿入でも同様の効果があります。IGNORE を指定しない場合、このような INSERT ステートメントはエラーで中止されます。ただし、INSERT IGNORE が使用されている場合は、一致しない値を含む行に対する挿入操作が暗黙のうちに失敗しますが、一致した行はすべて挿入されます。例については、セクション19.2.2「LIST パーティショニング」を参照してください。

  IGNORE が指定されていない場合は、エラーをトリガーするデータ変換によってステートメントが中止されます。IGNORE を指定すると、無効な値はもっとも近い値に調整されて挿入されます。警告は生成されますが、ステートメントは中止されません。mysql_info() C API 関数を使用すると、テーブルに実際に挿入された行数を確認できます。

- ON DUPLICATE KEY UPDATE を指定したとき、UNIQUE インデックスまたは PRIMARY KEY に重複した値を発生させる行が挿入された場合は、古い行の UPDATE が実行されます。行ごとの影響を受けた行の値は、その行が新しい行として挿入された場合は 1、既存の行が更新された場合は 2、既存の行がその現在の値に設定された場合は 0 です。mysqld への接続時に CLIENT_FOUND_ROWS フラグを mysql_real_connect() に指定する

と、既存の行がその現在の値に設定された場合の影響を受けた行の値は (0 ではなく) 1 になります。セクション13.2.5.3「INSERT ... ON DUPLICATE KEY UPDATE 構文」を参照してください。

テーブルに挿入するには、そのテーブルに対する INSERT 権限が必要です。ON DUPLICATE KEY UPDATE 句が使用されていて、重複キーのために代わりに UPDATE が実行される場合、このステートメントには、更新されるカラムに対する UPDATE 権限が必要です。読み取られるが、変更されないカラムの場合は、SELECT 権限のみが必要です (ON DUPLICATE KEY UPDATE 句にある col_name=expr 割り当ての右側でのみ参照されるカラムの場合など)。

MySQL 5.6.6 より前は、テーブルレベルのロックを採用した MyISAM などのストレージエンジンを使用しているパーティション化されたテーブルに影響を与える INSERT によって、そのテーブルのすべてのパーティションがロックされました。これは、INSERT ... PARTITION ステートメントにも当てはまりました。(これは、行レベルロックを採用した InnoDB などのストレージエンジンでは発生しておらず、現在も発生しません。)MySQL 5.6.6 以降では、MySQL はパーティションロックプルーニングを使用します。これにより、行が挿入されるパーティションだけが実際にロックされるようになります。詳細は、セクション19.6.4「パーティショニングとロック」を参照してください。

13.2.5.1 INSERT ... SELECT 構文

```
INSERT [LOW_PRIORITY | HIGH_PRIORITY] [IGNORE]
    [INTO] tbl_name
    [PARTITION (partition_name,...)]
    [(col_name,...)]
    SELECT ...
    [ ON DUPLICATE KEY UPDATE col_name=expr, ... ]
```

INSERT ... SELECT を使用すると、1 つまたは多数のテーブルから多数の行をテーブルにすばやく挿入できます。例:

```
INSERT INTO tbl_temp2 (fld_id)
  SELECT tbl_temp1.fld_order_id
  FROM tbl_temp1 WHERE tbl_temp1.fld_order_id > 100;
```

INSERT ... SELECT ステートメントには、次の条件が適用されます。

- 重複キー違反の原因になる行を無視するには、IGNORE を指定します。
- DELAYED は、INSERT ... SELECT では無視されます。
- INSERT ステートメントのターゲットテーブルが、クエリーの SELECT 部分の FROM 句に現れてもかまいません。(これは、一部の古いバージョンの MySQL では不可能でした。)ただし、テーブルに挿入し、さらにサブクエリーで同じテーブルから選択することはできません。

  テーブルからの選択とそのテーブルへの挿入を同時に行う場合、MySQL は SELECT からの行を保持するための一時テーブルを作成してから、それらの行をターゲットテーブルに挿入します。ただし、TEMPORARY テーブルを同じステートメント内で 2 回参照することはできないため、t が TEMPORARY テーブルのときに INSERT INTO t ... SELECT ... FROM t を使用できない点は引き続き残ります (セクション B.5.7.2「TEMPORARY テーブルに関する問題」を参照してください)。
- AUTO_INCREMENT カラムは、通常どおりに機能します。
- バイナリログを使用して元のテーブルを確実に再作成できるようにするために、MySQL では、INSERT ... SELECT ステートメントでの並列挿入が許可されません。
- SELECT と INSERT が同じテーブルを参照している場合のあいまいなカラム参照の問題を回避するには、SELECT 部分で使用されている各テーブルの一意のエイリアスを指定し、その部分にあるカラム名を適切なエイリアスで修飾します。

MySQL 5.6.2 からは、テーブルの名前に続く PARTITION オプションでソースまたはターゲットテーブル (またはその両方) のどのパーティションまたはサブパーティション (またはその両方) を使用するかを明示的に選択できます。PARTITION がこのステートメントの SELECT 部分にあるソーステーブルの名前とともに使用されている場合は、そのパーティションリストで指定されているパーティションまたはサブパーティションの行のみが選択されます。PARTITION がこのステートメントの INSERT 部分のターゲットテーブルの名前とともに使用されている場合は、選択されたすべての行を、このオプションに続くパーティションリストで指定されているパーティションまたはサブパーティションに挿入できる必要があります。そうでない場合、INSERT ... SELECT ステートメントは失敗します。詳細および例については、セクション19.5「パーティション選択」を参照してください。

ON DUPLICATE KEY UPDATE の値の部分では、SELECT 部分で GROUP BY を使用していないかぎり、ほかのテーブル内のカラムを参照できます。1 つの副作用として、値の部分にある一意でないカラム名を修飾しなければならない点があります。

ORDER BY 句のない SELECT ステートメントが行を返す順序は特定されていません。つまり、レプリケーションを使用している場合、このような SELECT がマスターとスレーブ上で行を同じ順序で返す保証はありません。これにより、マスターとスレーブの間で不整合が発生する場合があります。これが発生しないようにするために、レプリケートされる INSERT ... SELECT ステートメントは常に INSERT ... SELECT ... ORDER BY column として記述するようにしてください。column の選択は、マスターとスレーブの両方で間違いなく行が同じ順序で返されるかぎり問題にはなりません。セクション17.4.1.16「レプリケーションと LIMIT」も参照してください。

この問題のために、MySQL 5.6.4 から、INSERT ... SELECT ON DUPLICATE KEY UPDATE および INSERT IGNORE ... SELECT ステートメントには、ステートメントベースのレプリケーションには安全でないというフラグが付けられます。この変更により、このようなステートメントは、ステートメントベースモードを使用しているときは口グ内に警告を生成し、MIXED モードを使用しているときは行ベース形式を使用して口グに記録されます。(Bug #11758262、Bug #50439)

セクション17.1.2.1「ステートメントベースおよび行ベースレプリケーションのメリットとデメリット」も参照してください。

MySQL 5.6.6 より前は、テーブルレベルのロックを採用した MyISAM などのストレージエンジンを使用しているパーティション化されたテーブルに対して機能した INSERT ... SELECT ステートメントによって、ソースおよびターゲットテーブルのすべてのパーティションがロックされました。(これは、行レベルロックを採用した InnoDB などのストレージエンジンを使用しているテーブルでは発生しておらず、現在も発生しません。)MySQL 5.6.6 以降では、ターゲットテーブルのすべてのパーティションがロックされますが、ソーステーブルは実際に読み取られたパーティションのみがロックされます。詳細は、セクション19.6.4「パーティショニングとロック」を参照してください。

13.2.5.2 INSERT DELAYED 構文

```
INSERT DELAYED ...
```

INSERT ステートメントの DELAYED オプションは、特定の種類のテーブル (MyISAM など) に使用できる、標準 SQL への MySQL 拡張です。クライアントが INSERT DELAYED を使用すると、サーバーからはただちに了解が得られ、行は、そのテーブルがほかのどのスレッドによっても使用されていないときに挿入されるようにキューに入れられます。

> **注記**
>
> INSERT DELAYED は、そのテーブルがほかで使用されていなければ、通常の INSERT より低速です。また、サーバーが、遅延された行が存在するテーブルごとに個別のスレッドを処理するための追加のオーバーヘッドもあります。つまり、INSERT DELAYED は、それが必要なことを実際に確信している場合にのみ使用するようにしてください。
>
> MySQL 5.6.6 現在、INSERT DELAYED は非推奨であり、将来のリリースで削除されます。代わりに INSERT (DELAYED を付けない) を使用してください。

キューに入れられた行は、テーブルに挿入されるまで、メモリー内にのみ保持されます。つまり、mysqld を強制的に (たとえば、kill -9 で) 終了した場合や、mysqld が予期せず終了した場合は、まだディスクに書き込まれていないキューに入れられたすべての行が失われます。

DELAYED の使用には、次のいくつかの制約があります。

- INSERT DELAYED は、MyISAM、MEMORY、ARCHIVE、および BLACKHOLE テーブルでのみ機能します。DELAYED をサポートしていないエンジンの場合は、エラーが発生します。
- 挿入は、ロックを保持するセッションではなく、別のスレッドによって処理される必要があるため、LOCK TABLES を使用してロックされたテーブルで使用された場合は、INSERT DELAYED に対してエラーが発生します。
- MyISAM テーブルでは、データファイルの途中に空きブロックが存在しない場合は、並列 SELECT および INSERT ステートメントがサポートされます。これらの状況では、MyISAM で INSERT DELAYED を使用する必要はほとんどありません。
- INSERT DELAYED は、値リストを指定する INSERT ステートメントでのみ使用するようにしてください。サーバーは、INSERT ... SELECT または INSERT ... ON DUPLICATE KEY UPDATE ステートメントでの DELAYED を無視します。
- INSERT DELAYED ステートメントはただちに復帰するため、行が挿入される前に、LAST_INSERT_ID() を使用して、このステートメントによって生成される可能性のある AUTO_INCREMENT 値を取得することはできません。

- DELAYED 行は、実際に挿入されるまで、SELECT ステートメントには表示されません。
- MySQL 5.6 より前は、ステートメントが複数の行を挿入し、バイナリロギングが有効になっており、かつグローバルロギング形式がステートメントベースである (つまり、binlog_format が STATEMENT に設定されているときは必ず) 場合、INSERT DELAYED は通常の INSERT として処理されました。MySQL 5.6 からは、binlog_format の値が STATEMENT または MIXED である場合は必ず、INSERT DELAYED は常に、単純な (つまり、DELAYED オプションのない) INSERT として処理されます。(後者の場合、このステートメントは行ベースのロギングへの切り替えをトリガーしなくなったため、そのステートメントベースの形式を使用してログに記録されます。)

  これは、行ベースのバイナリロギングモードを使用している (binlog_format が ROW に設定されている) 場合は適用されません。この場合、INSERT DELAYED ステートメントは常に、指定のとおりに DELAYED オプションを使用して実行され、行更新イベントとしてログに記録されます。
- INSERT DELAYED がスレーブ上で通常の INSERT として処理されるように、DELAYED はスレーブレプリケーションサーバー上で無視されます。これは、DELAYED のために、スレーブにマスターとは異なるデータが存在することになる場合があるためです。
- テーブルが書き込みロックされているときに、そのテーブル構造を変更するために ALTER TABLE が使用されると、保留中の INSERT DELAYED ステートメントは失われます。
- INSERT DELAYED は、ビューではサポートされません。
- INSERT DELAYED は、パーティション化されたテーブルではサポートされません。

次に、INSERT または REPLACE に対して DELAYED オプションを使用したときの動作について詳細に説明します。この説明では、「スレッド」は INSERT DELAYED ステートメントを受信したスレッドであり、「ハンドラ」は、特定のテーブルに対するすべての INSERT DELAYED ステートメントを処理するスレッドです。

- スレッドがあるテーブルに対する DELAYED ステートメントを実行すると、そのテーブルに対するすべての DELAYED ステートメントを処理するためのハンドラスレッドが作成されます (このようなハンドラがまだ存在しない場合)。
- このスレッドは、そのハンドラが以前に DELAYED ロックを取得したかどうかをチェックします。取得していない場合は、それを行うようハンドラスレッドに指示します。DELAYED ロックは、ほかのスレッドがそのテーブルに対する READ または WRITE ロックを保持している場合でも取得できます。ただし、ハンドラはテーブル構造を確実に最新の状態にするために、すべての ALTER TABLE ロックまたは FLUSH TABLES ステートメントが完了するのを待機します。
- このスレッドは INSERT ステートメントを実行しますが、行をテーブルに書き込む代わりに、最終行のコピーをそのハンドラスレッドによって管理されているキューに配置します。構文エラーはすべてスレッドによって検出され、クライアントプログラムにレポートされます。
- INSERT は挿入操作が完了する前に復帰するため、クライアントは重複した行の数や、結果として得られる行の AUTO_INCREMENT 値をサーバーから取得できません。(C API を使用している場合も、同じ理由で、mysql_info() 関数は意味のある情報を何も返しません。)
- バイナリログは、行がテーブルに挿入されたときにハンドラスレッドによって更新されます。複数行の挿入の場合、バイナリログは、先頭行が挿入されたときに更新されます。
- delayed_insert_limit 行が書き込まれるたびに、ハンドラは、いずれかの SELECT ステートメントが引き続き保留中かどうかをチェックします。保留中の場合は、続行する前に、これらの実行を許可します。
- ハンドラのキューにそれ以上行がなくなると、テーブルはロック解除されます。新しい INSERT DELAYED ステートメントが delayed_insert_timeout 秒以内に受信されなかった場合、ハンドラは終了します。
- 特定のハンドラキュー内で delayed_queue_size を超える行が保留中である場合、INSERT DELAYED を要求しているスレッドは、そのキューに空きができるまで待機します。これは、遅延されたメモリーキューのすべてのメモリーが mysqld によって使用されてしまわないようにするために行われます。
- このハンドラスレッドは、MySQL プロセスリストの Command カラム内に delayed_insert として現れます。これは、FLUSH TABLES ステートメントを実行するか、または KILL thread_id で強制終了すると強制終了されます。ただし、終了する前に、まずキューに入れられたすべての行をテーブルに格納します。この時間中は、ほかのスレッドからの新しいどの INSERT ステートメントも受け入れません。このあとに INSERT DELAYED ステートメントを実行すると、新しいハンドラスレッドが作成されます。

  つまり、実行中の INSERT DELAYED ハンドラが存在する場合、INSERT DELAYED ステートメントは通常の INSERT ステートメントより高い優先度を持っています。その他の更新ステートメントは、INSERT DELAYED

キューが空になるか、だれかが (KILL thread_id で) このハンドラスレッドを終了するか、またはだれかが FLUSH TABLES を実行するまで待機する必要があります。

- 次のステータス変数は、INSERT DELAYED ステートメントに関する情報を提供します。

ステータス変数	意味
Delayed_insert_threads	ハンドラスレッドの数
Delayed_writes	INSERT DELAYED で書き込まれた行数
Not_flushed_delayed_rows	書き込みを待機している行数

これらの変数は、SHOW STATUS ステートメントを発行するか、または mysqladmin extended-status コマンドを実行することによって表示できます。

13.2.5.3 INSERT ... ON DUPLICATE KEY UPDATE 構文

ON DUPLICATE KEY UPDATE を指定したとき、UNIQUE インデックスまたは PRIMARY KEY に重複した値を発生させる行が挿入された場合は、MySQL によって古い行の UPDATE が実行されます。たとえば、カラム a が UNIQUE として宣言され、値 1 を含んでいる場合、次の 2 つのステートメントには同様の効果があります。

```
INSERT INTO table (a,b,c) VALUES (1,2,3)
  ON DUPLICATE KEY UPDATE c=c+1;

UPDATE table SET c=c+1 WHERE a=1;
```

(これらの効果は、a が自動インクリメントカラムである InnoDB テーブルに対して同じではありません。自動インクリメントカラムを使用した場合、INSERT ステートメントは自動インクリメント値を増やしますが、UPDATE は増やしません。)

ON DUPLICATE KEY UPDATE 句には、カンマで区切られた、複数のカラム割り当てを含めることができます。

ON DUPLICATE KEY UPDATE を使用した場合、行ごとの影響を受けた行の値は、その行が新しい行として挿入された場合は 1、既存の行が更新された場合は 2、既存の行がその現在の値に設定された場合は 0 です。mysqld への接続時に CLIENT_FOUND_ROWS フラグを mysql_real_connect() に指定すると、既存の行がその現在の値に設定された場合の影響を受けた行の値は (0 ではなく) 1 になります。

カラム b も一意である場合、INSERT は、代わりに次の UPDATE ステートメントと同等です。

```
UPDATE table SET c=c+1 WHERE a=1 OR b=2 LIMIT 1;
```

a=1 OR b=2 が複数の行に一致する場合は、1 つの行だけが更新されます。一般に、一意のインデックスが複数含まれているテーブルに対して ON DUPLICATE KEY UPDATE 句を使用することは避けるようにしてください。

UPDATE 句で VALUES(col_name) 関数を使用して、INSERT ... ON DUPLICATE KEY UPDATE ステートメントの INSERT 部分からカラム値を参照できます。つまり、ON DUPLICATE KEY UPDATE 句にある VALUES(col_name) は、重複キーの競合が発生していない場合に挿入される col_name の値を参照します。この関数は、複数の行を挿入する際に特に役立ちます。VALUES() 関数は、INSERT ... UPDATE ステートメントの中でだけ意味を持ち、そうでなければ NULL を返します。例:

```
INSERT INTO table (a,b,c) VALUES (1,2,3),(4,5,6)
  ON DUPLICATE KEY UPDATE c=VALUES(a)+VALUES(b);
```

そのステートメントは、次の 2 つのステートメントと同一です。

```
INSERT INTO table (a,b,c) VALUES (1,2,3)
  ON DUPLICATE KEY UPDATE c=3;
INSERT INTO table (a,b,c) VALUES (4,5,6)
  ON DUPLICATE KEY UPDATE c=9;
```

テーブルに AUTO_INCREMENT カラムが含まれているときに、INSERT ... ON DUPLICATE KEY UPDATE で行を挿入または更新した場合、LAST_INSERT_ID() 関数は AUTO_INCREMENT 値を返します。

ON DUPLICATE KEY UPDATE を使用している場合、DELAYED オプションは無視されます。

INSERT ... SELECT ステートメントの結果は SELECT からの行の順序に依存し、またこの順序を常に保証することはできないため、ロギング時に、INSERT ... SELECT ON DUPLICATE KEY UPDATE ステートメントがマスターとスレーブで異なる可能性があります。そのため、MySQL 5.6.4 以降では、INSERT ... SELECT ON DUPLICATE KEY UPDATE ステートメントには、ステートメントベースのレプリケーションには安全でない

というフラグが付けられます。この変更により、このようなステートメントは、ステートメントベースモードを使用しているときはログ内に警告を生成し、MIXED モードを使用しているときは行ベース形式を使用してログに記録されます。さらに、MySQL 5.6.6 からは、一意のキーまたは主キーが複数含まれているテーブルに対する INSERT ... ON DUPLICATE KEY UPDATE ステートメントも安全ではないとしてマークされます。(Bug #11765650、Bug #58637) セクション17.1.2.1「ステートメントベースおよび行ベースレプリケーションのメリットとデメリット」も参照してください。

MySQL 5.6.6 より前は、テーブルレベルのロックを採用した MyISAM などのストレージエンジンを使用しているパーティション化されたテーブルに対する INSERT ... ON DUPLICATE KEY UPDATE によって、そのテーブルのすべてのパーティションがロックされました。(これは、行レベルロックを採用した InnoDB などのストレージエンジンを使用しているテーブルでは発生しておらず、現在も発生しません。)MySQL 5.6.6 以降では、このようなステートメントでは、パーティション化キーカラムが更新されたパーティションのみがロックされます。詳細は、セクション19.6.4「パーティショニングとロック」を参照してください。

13.2.6 LOAD DATA INFILE 構文

```
LOAD DATA [LOW_PRIORITY | CONCURRENT] [LOCAL] INFILE 'file_name'
    [REPLACE | IGNORE]
    INTO TABLE tbl_name
    [PARTITION (partition_name,...)]
    [CHARACTER SET charset_name]
    [{FIELDS | COLUMNS}
        [TERMINATED BY 'string']
        [[OPTIONALLY] ENCLOSED BY 'char']
        [ESCAPED BY 'char']
    ]
    [LINES
        [STARTING BY 'string']
        [TERMINATED BY 'string']
    ]
    [IGNORE number {LINES | ROWS}]
    [(col_name_or_user_var,...)]
    [SET col_name = expr,...]
```

LOAD DATA INFILE ステートメントは、非常に高速にテキストファイルからテーブルに行を読み取ります。LOAD DATA INFILE は、SELECT ... INTO OUTFILE を補完するものです。(セクション13.2.9.1「SELECT ... INTO 構文」を参照してください。)テーブルからファイルにデータを書き込むには、SELECT ... INTO OUTFILE を使用します。そのファイルをテーブルに読み戻すには、LOAD DATA INFILE を使用します。FIELDS および LINES 句の構文は、両方のステートメントで同じです。どちらの句もオプションですが、両方が指定される場合は、FIELDS を LINES の前に指定する必要があります。

mysqlimport ユーティリティーを使用してデータファイルをロードすることもできます。これは、LOAD DATA INFILE ステートメントをサーバーに送信することによって動作します。--local オプションを指定すると、mysqlimport は、クライアントホストからデータファイルを読み取ります。クライアントとサーバーが圧縮されたプロトコルをサポートしている場合は、--compress オプションを指定すると、低速ネットワーク経由のパフォーマンスを向上させることができます。セクション4.5.5「mysqlimport — データインポートプログラム」を参照してください。

INSERT と LOAD DATA INFILE の効率の比較、および LOAD DATA INFILE の高速化の詳細は、セクション8.2.2.1「INSERT ステートメントの速度」を参照してください。

ファイル名は、リテラル文字列として指定する必要があります。Windows では、パス名内のバックスラッシュをスラッシュまたは二重のバックスラッシュとして指定します。character_set_filesystem システム変数は、ファイル名の解釈を制御します。

MySQL 5.6.2 以降では、LOAD DATA は、パーティション、サブパーティション、またはその両方の 1 つ以上の名前のカンマ区切りリストを含む PARTITION オプションを使用した明示的なパーティション選択をサポートしています。このオプションが使用されているとき、リストで指定されているいずれかのパーティションまたはサブパーティションにファイルからの行を挿入できない場合、このステートメントは Found a row not matching the given partition set エラーで失敗します。詳細は、セクション19.5「パーティション選択」を参照してください。

テーブルロックを採用したストレージエンジン (MyISAM など) を使用しているパーティション化されたテーブルの場合、LOAD DATA はどのパーティションロックも削除できません。これは、行レベルロックを採用したストレージエンジン (InnoDB など) を使用しているテーブルには適用されません。詳細は、セクション19.6.4「パーティショニングとロック」を参照してください。

サーバーは、character_set_database システム変数によって示されている文字セットを使用してファイル内の情報を解釈します。SET NAMES や、character_set_client の設定は入力の解釈に影響を与えません。入力ファイル

の内容にデフォルトとは異なる文字セットが使用されている場合は、通常、CHARACTER SET 句を使用してそのファイルの文字セットを指定することをお勧めします。binary の文字セットは、「変換なし」を指定します。

LOAD DATA INFILE は、フィールド値がロードされるカラムのデータ型には関係なく、ファイル内のすべてのフィールドに同じ文字セットが割り当てられていると解釈します。ファイルの内容が正しく解釈されるように、そのファイルが正しい文字セットで書き込まれていることを確認する必要があります。たとえば、mysqldump -T を使用して、または mysql で SELECT ... INTO OUTFILE ステートメントを発行することによってデータファイルを書き込む場合は、そのファイルが LOAD DATA INFILE でロードされるときに使用される文字セットで出力が書き込まれるように、必ず --default-character-set オプションを使用してください。

> **注記**
>
> ucs2、utf16、utf16le、または utf32 文字セットを使用するデータファイルはロードできません。

LOW_PRIORITY を使用した場合、LOAD DATA ステートメントの実行は、ほかのどのクライアントもそのテーブルから読み取らなくなるまで遅延されます。これは、テーブルレベルロックのみを使用するストレージエンジン (MyISAM、MEMORY、および MERGE) にのみ影響を与えます。

並列挿入の条件を満たす (つまり、途中に空きブロックが含まれていない) MyISAM テーブルで CONCURRENT を指定すると、ほかのスレッドは LOAD DATA の実行中にそのテーブルからデータを取得できます。このオプションは、そのテーブルがほかのスレッドから同時に使用されていない場合でも、LOAD DATA のパフォーマンスに少し影響を与えます。

行ベースのレプリケーションでは、CONCURRENT は MySQL バージョンにかかわらずレプリケートされます。ステートメントベースのレプリケーションでは、CONCURRENT は MySQL 5.5.1 より前ではレプリケートされません (Bug #34628 を参照してください)。詳細は、セクション17.4.1.17「レプリケーションと LOAD DATA INFILE」を参照してください。

LOCAL キーワードは、あとで説明されているように、ファイルの予測される場所やエラー処理に影響を与えます。LOCAL は、サーバーとクライアントの両方がそれを許可するように構成されている場合にのみ機能します。たとえば、mysqld が --local-infile=0 で起動された場合、LOCAL は機能しません。セクション6.1.6「LOAD DATA LOCAL のセキュリティーの問題」を参照してください。

LOCAL キーワードは、ファイルが見つかることが予測される場所に影響を与えます。

- LOCAL が指定されている場合、ファイルはクライアントホスト上のクライアントプログラムによって読み取られ、サーバーに送信されます。このファイルは、その正確な場所を指定するためにフルパス名として指定できます。相対パス名として指定されている場合、その名前は、クライアントプログラムが起動されたディレクトリを基準にして解釈されます。

  LOCAL を LOAD DATA とともに使用している場合は、そのファイルのコピーがサーバーの一時ディレクトリ内に作成されます。これは tmpdir または slave_load_tmpdir の値によって決定されるディレクトリではなく、オペレーティングシステムの一時ディレクトリであり、MySQL Server では構成できません。(システムの一時ディレクトリは通常、Linux システムでは /tmp、Windows では C:\WINDOWS\TEMP です。)このディレクトリ内にコピーのための十分な領域がないと、LOAD DATA LOCAL ステートメントが失敗する場合があります。

- LOCAL が指定されていない場合、ファイルはサーバーホスト上にある必要があり、直接サーバーによって読み取られます。サーバーは、次のルールを使用してファイルを見つけます。

  - ファイル名が絶対パス名である場合、サーバーはそれを指定されたとおりに使用します。
  - ファイル名が 1 つ以上の先行コンポーネントを含む相対パス名である場合、サーバーは、サーバーのデータディレクトリを基準にしてファイルを検索します。
  - 先行コンポーネントを含まないファイル名が指定されている場合、サーバーは、デフォルトデータベースのデータベースディレクトリ内でそのファイルを探します。

LOCAL 以外のケースでは、これらのルールは、./myfile.txt という名前のファイルがサーバーのデータディレクトリから読み取られるのに対して、myfile.txt として指定されたファイルはデフォルトデータベースのデータベースディレクトリから読み取られることを示します。たとえば、db1 がデフォルトデータベースである場合、次の LOAD DATA ステートメントは、このステートメントが明示的に db2 データベース内のテーブルにファイルをロードしているにもかかわらず、db1 のデータベースディレクトリからファイル data.txt を読み取ります。

```
LOAD DATA INFILE 'data.txt' INTO TABLE db2.my_table;
```

セキュリティー上の理由から、サーバー上にあるテキストファイルを読み取る場合、そのファイルはデータベースディレクトリ内に存在するか、またはすべてのユーザーから読み取り可能のどちらかである必要がありま

す。また、サーバーファイルに対して LOAD DATA INFILE を使用するには、FILE 権限が必要です。セクション 6.2.1「MySQL で提供される権限」を参照してください。LOCAL 以外のロード操作では、secure_file_priv システム変数が空以外のディレクトリ名に設定されている場合、ロードされるファイルはそのディレクトリ内に存在する必要があります。

LOCAL を使用すると、クライアントが接続を経由してファイルの内容をサーバーに送信する必要があるため、サーバーが直接ファイルにアクセスできるようにした場合より少し遅くなります。その一方で、ローカルファイルをロードするために FILE 権限は必要ありません。

LOCAL はまた、エラー処理にも影響を与えます。

- LOAD DATA INFILE では、データ解釈や重複キーのエラーによって操作が終了します。
- LOAD DATA LOCAL INFILE では、操作の最中にファイルの転送を停止する方法がサーバーにはないため、データ解釈や重複キーのエラーは警告になり、操作は続行されます。重複キーエラーについては、これは IGNORE が指定されている場合と同じです。IGNORE については、このセクションのあとの方でさらに詳細に説明されています。

REPLACE および IGNORE キーワードは、一意のキー値に関して既存の行を複製する入力行の処理を制御します。

- REPLACE を指定した場合は、入力行によって既存の行が置き換えられます。つまり、主キーまたは一意のインデックスに関して既存の行と同じ値を持つ行のことです。セクション13.2.8「REPLACE 構文」を参照してください。
- IGNORE を指定した場合は、一意のキー値に関して既存の行を複製する行は破棄されます。
- どちらのオプションも指定しない場合、その動作は LOCAL キーワードが指定されているかどうかによって異なります。LOCAL が指定されていない場合は、重複キー値が見つかるとエラーが発生し、テキストファイルの残りは無視されます。LOCAL が指定されている場合、デフォルトの動作は IGNORE が指定されている場合と同じです。これは、操作の最中にファイルの転送を停止する方法がサーバーにはないためです。

ロード操作中に外部キー制約を無視するには、LOAD DATA を実行する前に SET foreign_key_checks = 0 ステートメントを発行します。

空の MyISAM テーブルに対して LOAD DATA INFILE を使用した場合、一意でないインデックスはすべて (REPAIR TABLE として) 別のバッチに作成されます。通常、多くのインデックスがあるときは、この方法で LOAD DATA INFILE がはるかに早くなります。一部の極端なケースでは、ファイルをテーブルにロードする前に ALTER TABLE ... DISABLE KEYS でインデックスを無効にし、ファイルをロードしたあとに ALTER TABLE ... ENABLE KEYS を使用してインデックスを再作成することによって、インデックスをさらに高速に作成できます。セクション8.2.2.1「INSERT ステートメントの速度」を参照してください。

FIELDS および LINES 句の構文は、LOAD DATA INFILE と SELECT ... INTO OUTFILE の両方のステートメントで同じです。どちらの句もオプションですが、両方が指定される場合は、FIELDS を LINES の前に指定する必要があります。

FIELDS 句を指定する場合は、その各サブ句 (TERMINATED BY、[OPTIONALLY] ENCLOSED BY、および ESCAPED BY) もオプションです。ただし、そのうちの少なくとも 1 つを指定する必要があります。

FIELDS または LINES 句を指定しない場合、そのデフォルトは、次を記述した場合と同じです。

```
FIELDS TERMINATED BY '\t' ENCLOSED BY '' ESCAPED BY '\\'
LINES TERMINATED BY '\n' STARTING BY ''
```

(バックスラッシュは SQL ステートメントの文字列内の MySQL のエスケープ文字であるため、リテラルバックスラッシュを指定するには、1 つのバックスラッシュとして解釈される値に対して 2 つのバックスラッシュを指定する必要があります。エスケープシーケンス '\t' と '\n' は、それぞれタブと改行文字を指定します。)

つまり、デフォルトでは、入力を読み取るとき LOAD DATA INFILE は次のように機能します。

- 改行の位置にある行の境界を探します。
- どの行プリフィクスもスキップしません。
- タブの位置で行をフィールドに分割します。
- フィールドが引用文字で囲まれていることを期待しません。

- 前にエスケープ文字「\」がある文字をエスケープシーケンスとして解釈します。たとえば、「\t」、「\n」、および「\\」はそれぞれ、タブ、改行、およびバックスラッシュを示します。エスケープシーケンスの完全なリストについては、あとの FIELDS ESCAPED BY の説明を参照してください。

逆に、デフォルトでは、出力を書き込むとき SELECT ... INTO OUTFILE は次のように機能します。

- フィールド間にタブを書き込みます。
- フィールドを引用文字で囲みません。
- 「\」を使用して、フィールド値の中に現れるタブ、改行、または「\」のインスタンスをエスケープします。
- 行の最後に改行を書き込みます。

> **注記**
>
> Windows システム上でテキストファイルを生成した場合、Windows プログラムは通常、行ターミネータとして 2 文字を使用するため、そのファイルを正しく読み取るには LINES TERMINATED BY '\r\n' の使用が必要になることがあります。WordPad などの一部のプログラムは、ファイルを書き込むときに行ターミネータとして \r を使用する可能性があります。このようなファイルを読み取るには、LINES TERMINATED BY '\r' を使用します。

読み取るすべての行に、無視したい共通のプリフィクスが含まれている場合は、LINES STARTING BY 'prefix_string' を使用して、プリフィクスとその前にあるすべてのものをスキップできます。行にプリフィクスが含まれていない場合は、行全体がスキップされます。たとえば、次のステートメントを発行するとします。

```
LOAD DATA INFILE '/tmp/test.txt' INTO TABLE test
  FIELDS TERMINATED BY ','  LINES STARTING BY 'xxx';
```

データファイルは次のようになっています。

```
xxx"abc",1
something xxx"def",2
"ghi",3
```

結果として得られる行は、("abc",1) および ("def",2) になります。ファイル内の 3 行目は、プリフィクスが含まれていないためスキップされます。

IGNORE number LINES オプションを使用すると、ファイルの先頭にある行を無視できます。たとえば、IGNORE 1 LINES を使用すると、カラム名を含む開始ヘッダー行をスキップできます。

```
LOAD DATA INFILE '/tmp/test.txt' INTO TABLE test IGNORE 1 LINES;
```

SELECT ... INTO OUTFILE と LOAD DATA INFILE を連携して使用してデータベースからファイルにデータを書き込み、あとでそのファイルを元のデータベースに読み取る場合は、両方のステートメントのフィールド処理と行処理のオプションが一致している必要があります。そうしないと、LOAD DATA INFILE は、そのファイルの内容を正しく解釈しません。SELECT ... INTO OUTFILE を使用して、カンマで区切られたフィールドを含むファイルを書き込むとします。

```
SELECT * INTO OUTFILE 'data.txt'
  FIELDS TERMINATED BY ','
  FROM table2;
```

カンマで区切られたファイルを読み戻すための正しいステートメントは次のようになります。

```
LOAD DATA INFILE 'data.txt' INTO TABLE table2
  FIELDS TERMINATED BY ',';
```

代わりに、次に示すステートメントを使用してこのファイルを読み取ろうとしても、これはフィールド間のタブを探すよう LOAD DATA INFILE に指示するため機能しません。

```
LOAD DATA INFILE 'data.txt' INTO TABLE table2
  FIELDS TERMINATED BY '\t';
```

その結果、各入力行が 1 つのフィールドとして解釈される可能性があります。

LOAD DATA INFILE を使用すると、外部ソースから取得されたファイルを読み取ることができます。たとえば、多くのプログラムは、各行にカンマで区切られ、二重引用符で囲まれた複数のフィールドが含まれており、かつ開始行がカラム名になっているようなカンマ区切り値 (CSV) 形式でデータをエクスポートできます。このような

ファイル内の行が復帰改行と改行のペアで終了している場合、次に示すステートメントは、このファイルをロードするために使用するフィールド処理と行処理のオプションを示しています。

```
LOAD DATA INFILE 'data.txt' INTO TABLE tbl_name
  FIELDS TERMINATED BY ',' ENCLOSED BY ""
  LINES TERMINATED BY '\r\n'
  IGNORE 1 LINES;
```

入力値が必ずしも引用符で囲まれていない場合は、ENCLOSED BY キーワードの前に OPTIONALLY を使用します。

フィールド処理または行処理のどのオプションにも、空の文字列 ('') を指定できます。空でない場合、FIELDS [OPTIONALLY] ENCLOSED BY および FIELDS ESCAPED BY 値は単一の文字である必要があります。FIELDS TERMINATED BY、LINES STARTING BY、および LINES TERMINATED BY 値は、複数の文字にすることができます。たとえば、復帰改行と改行のペアで終了する行を書き込むか、またはこのような行を含むファイルを読み取るには、LINES TERMINATED BY '\r\n' 句を指定します。

%% から成る行で区切られたジョークを含むファイルを読み取るには、次のようにできます。

```
CREATE TABLE jokes
  (a INT NOT NULL AUTO_INCREMENT PRIMARY KEY,
  joke TEXT NOT NULL);
LOAD DATA INFILE '/tmp/jokes.txt' INTO TABLE jokes
  FIELDS TERMINATED BY ''
  LINES TERMINATED BY '\n%%\n' (joke);
```

FIELDS [OPTIONALLY] ENCLOSED BY は、フィールドの引用符を制御します。出力 (SELECT ... INTO OUTFILE) でワード OPTIONALLY を省略した場合は、すべてのフィールドが ENCLOSED BY 文字で囲まれます。フィールド区切り文字としてカンマを使用したこのような出力の例を次に示します。

```
"1","a string","100.20"
"2","a string containing a , comma","102.20"
"3","a string containing a \" quote","102.20"
"4","a string containing a \", quote and comma","102.20"
```

OPTIONALLY を指定した場合、ENCLOSED BY 文字は、文字列データ型 (CHAR、BINARY、TEXT、ENUM など) を持つカラムの値を囲むためにのみ使用されます。

```
1,"a string",100.20
2,"a string containing a , comma",102.20
3,"a string containing a \" quote",102.20
4,"a string containing a \", quote and comma",102.20
```

フィールド値の中に ENCLOSED BY 文字が現れると、その文字は、前に ESCAPED BY 文字を付けることによってエスケープされます。また、空の ESCAPED BY 値を指定した場合は、LOAD DATA INFILE で正しく読み取ることができない出力が誤って生成される可能性もあります。たとえば、エスケープ文字が空である場合、今示した前の出力は次のようになります。4 行目の 2 番目のフィールドに含まれる引用符のあとにカンマが続いていることに注目してください。これにより、このフィールドが (誤って) 終了するように見えます。

```
1,"a string",100.20
2,"a string containing a , comma",102.20
3,"a string containing a " quote",102.20
4,"a string containing a ", quote and comma",102.20
```

入力では、ENCLOSED BY 文字 (存在する場合) はフィールド値の最後から取り除かれます。(これは、OPTIONALLY が指定されているかどうかには関係しません。OPTIONALLY は、入力の解釈には影響を与えません。)ENCLOSED BY 文字が ESCAPED BY 文字のあとに現れた場合は、現在のフィールド値の一部として解釈されます。

フィールドが ENCLOSED BY 文字で始まったとき、その文字のインスタンスがフィールド値の終了として認識されるのは、そのあとにフィールドまたは行の TERMINATED BY シーケンスが続いている場合だけです。あいまいさを避けるために、フィールド値の中に ENCLOSED BY 文字が現れるときはそれを 2 文字にすることができ、それがその文字の単一インスタンスとして解釈されます。たとえば、ENCLOSED BY "" が指定されている場合、引用符は次に示すように処理されます。

```
"The ""BIG"" boss"  -> The "BIG" boss
The "BIG" boss      -> The "BIG" boss
The ""BIG"" boss    -> The ""BIG"" boss
```

FIELDS ESCAPED BY は、特殊文字の読み取りまたは書き込みの方法を制御します。

- 入力では、FIELDS ESCAPED BY 文字が空でない場合、その文字が現れると取り除かれ、それに続く文字がフィールド値の一部として文字どおりに解釈されます。最初の文字がエスケープ文字である一部の 2 文字シー

ケンスは例外です。これらのシーケンスを (エスケープ文字に「\」を使用して) 次の表に示します。NULL 処理のルールについては、このセクションのあとの方で説明されています。

文字	エスケープシーケンス
\0	ASCII NUL (0x00) 文字
\b	バックスペース文字
\n	改行 (ラインフィード) 文字
\r	復帰改行文字
\t	タブ文字。
\Z	ASCII 26 (Ctrl+Z)
\N	NULL

「\」でのエスケープ構文の詳細は、セクション9.1.1「文字列リテラル」を参照してください。

FIELDS ESCAPED BY 文字が空である場合、エスケープシーケンスの解釈は実行されません。

- 出力では、FIELDS ESCAPED BY 文字が空でない場合、その文字は、出力上で次の文字の前に付けるために使用されます。
  - FIELDS ESCAPED BY 文字
  - FIELDS [OPTIONALLY] ENCLOSED BY 文字
  - FIELDS TERMINATED BY および LINES TERMINATED BY 値の最初の文字
  - ASCII 0 (エスケープ文字のあとに実際に書き込まれる文字は 0 の値のバイトではなく、ASCII の「0」です)

  FIELDS ESCAPED BY 文字が空である場合は、どの文字もエスケープされず、NULL は \N ではなく、NULL として出力されます。特に、データ内のフィールド値に今指定したリスト内のいずれかの文字が含まれている場合、空のエスケープ文字を指定することはおそらく適切な方法ではありません。

特定のケースでは、フィールド処理と行処理のオプションは相互に作用します。

- LINES TERMINATED BY が空の文字列であり、かつ FIELDS TERMINATED BY が空以外である場合、行は FIELDS TERMINATED BY でも終了します。
- FIELDS TERMINATED BY と FIELDS ENCLOSED BY の値がどちらも空 ('') である場合は、固定行 (区切られていない) フォーマットが使用されます。固定行フォーマットでは、フィールド間に区切り文字は使用されません (ただし、行ターミネータは引き続き存在できます)。代わりに、カラム値は、そのフィールド内のすべての値を保持するために十分に広いフィールド幅を使用して読み取りと書き込みが行われます。TINYINT、SMALLINT、MEDIUMINT、INT、および BIGINT では、宣言されている表示幅にかかわらず、フィールド幅はそれぞれ 4、6、8、11、および 20 です。

  LINES TERMINATED BY は引き続き、行を区切るために使用されます。ある行にすべてのフィールドが含まれていない場合、カラムの残りの部分はそのデフォルト値に設定されます。行ターミネータが存在しない場合は、これを '' に設定してください。この場合は、テキストファイルの各行にすべてのフィールドが含まれている必要があります。

  固定行フォーマットはまた、あとで説明されているように、NULL 値の処理にも影響を与えます。複数バイトの文字セットを使用している場合は、固定サイズフォーマットが機能しません。

NULL 値の処理は、使用されている FIELDS および LINES オプションによって異なります。

- デフォルトの FIELDS および LINES 値では、NULL は出力として \N のフィールド値として書き込まれ、\N のフィールド値は入力として NULL として読み取られます (ESCAPED BY 文字は「\」であると仮定します)。
- FIELDS ENCLOSED BY が空でない場合、リテラルワード NULL をその値として含むフィールドは NULL 値として読み取られます。これは、文字列 'NULL' として読み取られる、FIELDS ENCLOSED BY 文字で囲まれたワード NULL とは異なります。
- FIELDS ESCAPED BY が空である場合、NULL はワード NULL として書き込まれます。
- 固定行フォーマット (これは、FIELDS TERMINATED BY と FIELDS ENCLOSED BY がどちらも空であるときに使用されます) では、NULL は空の文字列として書き込まれます。これにより、NULL 値と空の文字列がどちらも空の文字列として書き込まれるため、ファイルに書き込まれた場合、テーブル内でこの両方を区別できな

くなります。ファイルを読み戻したときにこの 2 つを区別できることが必要な場合は、固定行フォーマットを使用すべきではありません。

NULL を NOT NULL カラムにロードしようとすると、そのカラムのデータ型の暗黙のデフォルト値の割り当てが行われて警告が発生するか、または厳密な SQL モードではエラーが発生します。暗黙のデフォルト値については、セクション11.6「データ型デフォルト値」で説明されています。

次の一部のケースは、LOAD DATA INFILE ではサポートされません。

- 固定サイズ行 (FIELDS TERMINATED BY と FIELDS ENCLOSED BY がどちらも空) および BLOB または TEXT カラム。
- 別の区切り文字と同じか、または別の区切り文字のプリフィクスである区切り文字を指定した場合、LOAD DATA INFILE は入力を正しく解釈できません。たとえば、次の FIELDS 句では問題が発生します。

```
FIELDS TERMINATED BY "" ENCLOSED BY ""
```

- FIELDS ESCAPED BY が空である場合、フィールド値の中に FIELDS ENCLOSED BY または LINES TERMINATED BY に続いて FIELDS TERMINATED BY 値が現れると、LOAD DATA INFILE はフィールドまたは行の読み取りを非常に早く停止します。これは、LOAD DATA INFILE が、フィールドまたは行の値がどこで終了するかを正しく判定できないために発生します。

次の例では、persondata テーブルのすべてのカラムをロードします。

```
LOAD DATA INFILE 'persondata.txt' INTO TABLE persondata;
```

デフォルトでは、LOAD DATA INFILE ステートメントの最後にカラムリストが指定されていないときは、入力行にテーブルカラムごとのフィールドが含まれていることが期待されます。テーブルのカラムの一部のみをロードする場合は、カラムリストを指定します。

```
LOAD DATA INFILE 'persondata.txt' INTO TABLE persondata (col1,col2,...);
```

カラムリストはまた、入力ファイル内のフィールドの順序がテーブル内のカラムの順序と異なる場合にも指定する必要があります。そうしないと、MySQL は、入力フィールドとテーブルカラムを一致させる方法がわかりません。

カラムリストには、カラム名またはユーザー変数のどちらかを含めることができます。ユーザー変数では、SET 句を使用して、各値に対して変換を実行してからその結果をカラムに代入できます。

SET 句内のユーザー変数は、いくつかの方法で使用できます。次の例では、最初の入力カラムを直接 t1.column1 の値に使用し、2 番目の入力カラムを、t1.column2 の値に使用される前に除算演算の対象になるユーザー変数に割り当てます。

```
LOAD DATA INFILE 'file.txt'
 INTO TABLE t1
 (column1, @var1)
 SET column2 = @var1/100;
```

SET 句を使用すると、入力ファイルからは取得されない値を指定できます。次のステートメントは、column3 を現在の日付と時間に設定します。

```
LOAD DATA INFILE 'file.txt'
 INTO TABLE t1
 (column1, column2)
 SET column3 = CURRENT_TIMESTAMP;
```

入力値をユーザー変数に割り当て、その変数をテーブルカラムには代入しないようにして、その値を破棄することもできます。

```
LOAD DATA INFILE 'file.txt'
 INTO TABLE t1
 (column1, @dummy, column2, @dummy, column3);
```

カラム/変数リストと SET 句の使用は、次の制限に従います。

- SET 句の代入では、割り当て演算子の左側にカラム名のみを置くようにしてください。
- SET の代入の右側では、サブクエリーを使用できます。カラムに代入される値を返すサブクエリーとして使用できるのは、スカラーサブクエリーだけです。また、サブクエリーを使用して、ロードされているテーブルから選択することはできません。

- IGNORE 句によって無視された行は、カラム/変数リストや SET 句では処理されません。
- ユーザー変数には表示幅がないため、固定行フォーマットのデータをロードする場合はユーザー変数を使用できません。

入力行を処理する場合、LOAD DATA はそれをフィールドに分割し、カラム/変数リストと SET 句に応じた値 (存在する場合) を使用します。そのあと、結果として得られる行がテーブルに挿入されます。そのテーブルに BEFORE INSERT または AFTER INSERT トリガーが存在する場合、これらのトリガーはそれぞれ、行挿入の前またはあとにアクティブ化されます。

入力行に含まれるフィールドが多すぎる場合は、余分なフィールドが無視され、警告数が 1 増えます。

入力行に含まれるフィールドが少なすぎる場合、入力フィールドがないテーブルカラムはそのデフォルト値に設定されます。デフォルト値の割り当てについては、セクション11.6「データ型デフォルト値」で説明されています。

空のフィールド値はフィールドがないとは見なされず、次のように解釈されます。

- 文字列型の場合、このカラムは空の文字列に設定されます。
- 数値型の場合、このカラムは 0 に設定されます。
- 日付と時間型の場合、このカラムはその型の適切な「0」の値に設定されます。セクション11.3「日付と時間型」を参照してください。

これらは、INSERT または UPDATE ステートメントで空の文字列を文字列、数値、日付または時間の各型に明示的に割り当てた場合の結果と同じ値です。

空のフィールド値や正しくないフィールド値の処理は、SQL モードが制限的な値に設定されていると、今説明した処理とは異なってきます。たとえば、sql_mode='TRADITIONAL である場合は、空の値や 'x' などの値を数値カラムに変換すると 0 に変換されるのではなく、エラーが発生します。(LOCAL が指定されている場合は、操作の最中にファイルの転送を停止する方法がサーバーにはないため、制限的な sql_mode 値が設定されていても、エラーではなく警告が発生します。)

TIMESTAMP カラムが現在の日付と時間に設定されるのは、そのカラムに NULL 値 (つまり、\N) が存在し、かつそのカラムが NULL 値を許可するように宣言されていない場合、または TIMESTAMP カラムのデフォルト値が現在のタイムスタンプであり、かつフィールドリストが指定されたときにこのカラムがフィールドリストから省略されている場合だけです。

LOAD DATA INFILE はすべての入力を文字列と見なすため、ENUM または SET カラムの数値を INSERT ステートメントと同じように使用することはできません。ENUM および SET 値はすべて、文字列として指定する必要があります。

BIT 値を、2 進表記 (b'011010' など) を使用してロードすることはできません。これを回避するには、その値を通常の整数として指定し、SET 句を使用して変換することにより、MySQL で数値型の変換が実行され、それが BIT カラムに正しくロードされるようにします。

```
shell> cat /tmp/bit_test.txt
2
127
shell> mysql test
mysql> LOAD DATA INFILE '/tmp/bit_test.txt'
    -> INTO TABLE bit_test (@var1) SET b = CAST(@var1 AS UNSIGNED);
Query OK, 2 rows affected (0.00 sec)
Records: 2  Deleted: 0  Skipped: 0  Warnings: 0

mysql> SELECT BIN(b+0) FROM bit_test;
+----------+
| bin(b+0) |
+----------+
| 10       |
| 1111111  |
+----------+
2 rows in set (0.00 sec)
```

Unix では、LOAD DATA でパイプから読み取る必要がある場合は次の手法を使用できます (この例では、/ ディレクトリのリストをテーブル db1.t1 にロードします)。

```
mkfifo /mysql/data/db1/ls.dat
chmod 666 /mysql/data/db1/ls.dat
```

```
find / -ls > /mysql/data/db1/ls.dat &
mysql -e "LOAD DATA INFILE 'ls.dat' INTO TABLE t1" db1
```

ロードされるデータを生成するコマンドと mysql コマンドを別の端末で実行するか、または (前の例に示したように) バックグラウンドでデータ生成プロセスを実行する必要があります。これを行わないと、データが mysql プロセスから読み取られる準備ができるまで、パイプがブロックされます。

LOAD DATA INFILE ステートメントは、完了すると、次の形式の情報文字列を返します。

```
Records: 1  Deleted: 0  Skipped: 0  Warnings: 0
```

警告は、INSERT ステートメントを使用して値が挿入されるときと同じ状況で発生します (セクション 13.2.5「INSERT 構文」を参照してください)。ただし、LOAD DATA INFILE では、入力行内のフィールドが少なすぎるか、または多すぎる場合にも警告が生成されます。

SHOW WARNINGS を使用すると、発生した問題に関する情報として最初の max_error_count 警告のリストを取得できます。セクション13.7.5.41「SHOW WARNINGS 構文」を参照してください。

C API を使用している場合は、mysql_info() 関数を呼び出すことによって、そのステートメントに関する情報を取得できます。セクション23.7.7.35「mysql_info()」を参照してください。

13.2.7 LOAD XML 構文

```
LOAD XML [LOW_PRIORITY | CONCURRENT] [LOCAL] INFILE 'file_name'
    [REPLACE | IGNORE]
    INTO TABLE [db_name.]tbl_name
    [PARTITION (partition_name,...)]
    [CHARACTER SET charset_name]
    [ROWS IDENTIFIED BY '<tagname>']
    [IGNORE number {LINES | ROWS}]
    [(column_or_user_var,...)]
    [SET col_name = expr,...]
```

LOAD XML ステートメントは、XML ファイルからテーブルにデータを読み取ります。file_name は、リテラル文字列として指定する必要があります。オプションの ROWS IDENTIFIED BY 句内の tagname もリテラル文字列として指定し、山括弧 (< および >) で囲む必要があります。

LOAD XML は、XML 出力モードでの mysql クライアントの実行 (つまり、--xml オプションを使用したクライアントの起動) を補完するものとして機能します。テーブルから XML ファイルにデータを書き込むには、システムシェルから次のようなコマンドを使用します。

```
shell> mysql --xml -e 'SELECT * FROM mytable' > file.xml
```

そのファイルをテーブルに読み戻すには、LOAD XML INFILE を使用します。デフォルトでは、<row> 要素は、データベーステーブル行と同等であると見なされます。これは、ROWS IDENTIFIED BY 句を使用して変更できます。

このステートメントは、次の 3 つの異なる XML 形式をサポートします。

- 属性としてのカラム名と、属性値としてのカラム値:

  ```
  <row column1="value1" column2="value2" .../>
  ```

- タグとしてのカラム名と、これらのタグの内容としてのカラム値:

  ```
  <row>
    <column1>value1</column1>
    <column2>value2</column2>
  </row>
  ```

- カラム名は <field> タグの name 属性で、値はこれらのタグの内容:

  ```
  <row>
    <field name='column1'>value1</field>
    <field name='column2'>value2</field>
  </row>
  ```

  これは、mysqldump などのほかの MySQL ツールによって使用される形式です。

同じ XML ファイルで 3 つのすべての形式を使用できます。インポートルーチンは各行の形式を自動的に検出し、それを正しく解釈します。タグは、タグまたは属性名とカラム名に基づいて照合されます。

次の句は、基本的に LOAD XML に対して LOAD DATA に対する場合と同じように機能します。

- LOW_PRIORITY または CONCURRENT
- LOCAL
- REPLACE または IGNORE
- PARTITION
- CHARACTER SET
- (column_or_user_var,...)
- SET

これらの句の詳細は、セクション13.2.6「LOAD DATA INFILE 構文」を参照してください。

IGNORE number LINES または IGNORE number ROWS 句を指定すると、XML ファイル内の最初の number 行がスキップされます。これは、LOAD DATA ステートメントの IGNORE ... LINES 句に類似しています。

このステートメントがどのように使用されるかを示すために、次のように作成されたテーブルがあるとします。

```
USE test;

CREATE TABLE person (
    person_id INT NOT NULL PRIMARY KEY,
    fname VARCHAR(40) NULL,
    lname VARCHAR(40) NULL,
    created TIMESTAMP
);
```

さらに、このテーブルが最初は空であるとします。

ここで、次に示すような内容を持つ単純な XML ファイル person.xml があるとします。

```
<?xml version="1.0"?>
<list>
  <person person_id="1" fname="Pekka" lname="Nousiainen"/>
  <person person_id="2" fname="Jonas" lname="Oreland"/>
  <person person_id="3"><fname>Mikael</fname><lname>Ronström</lname></person>
  <person person_id="4"><fname>Lars</fname><lname>Thalmann</lname></person>
  <person><field name="person_id">5</field><field name="fname">Tomas</field>
  <field name="lname">Ulin</field></person>
  <person><field name="person_id">6</field><field name="fname">Martin</field>
  <field name="lname">Sköld</field></person>
</list>
```

例として示したこのファイルには、前に説明した許可される各 XML 形式が表されています。

person.xml 内のデータを person テーブルにインポートするには、次のステートメントを使用できます。

```
mysql> LOAD XML LOCAL INFILE 'person.xml'
    ->   INTO TABLE person
    ->   ROWS IDENTIFIED BY '<person>';

Query OK, 6 rows affected (0.00 sec)
Records: 6  Deleted: 0  Skipped: 0  Warnings: 0
```

ここでは、person.xml が MySQL データディレクトリ内に存在することを前提にしています。このファイルが見つからない場合は、次のエラーが発生します。

```
ERROR 2 (HY000): File '/person.xml' not found (Errcode: 2)
```

ROWS IDENTIFIED BY '<person>' 句は、XML ファイル内の各 <person> 要素が、このデータがインポートされるテーブル内の各行と同等であると見なされることを示します。この場合、これは test データベース内の person テーブルです。

サーバーからの応答でわかるように、test.person テーブルには 6 行がインポートされました。これは、単純な SELECT ステートメントで確認できます。

```
mysql> SELECT * FROM person;
+-----------+--------+------------+---------------------+
```

```
| person_id | fname  | lname      | created             |
+-----------+--------+------------+---------------------+
|         1 | Pekka  | Nousiainen | 2007-07-13 16:18:47 |
|         2 | Jonas  | Oreland    | 2007-07-13 16:18:47 |
|         3 | Mikael | Rönström   | 2007-07-13 16:18:47 |
|         4 | Lars   | Thalmann   | 2007-07-13 16:18:47 |
|         5 | Tomas  | Ulin       | 2007-07-13 16:18:47 |
|         6 | Martin | Sköld      | 2007-07-13 16:18:47 |
+-----------+--------+------------+---------------------+
6 rows in set (0.00 sec)
```

このセクションの前の方で説明したように、許可される 3 つの XML 形式のいずれかまたはすべてを 1 つのファイルに含め、それを LOAD XML を使用して読み取ることができます。

上の操作の逆、つまり、MySQL テーブルデータの XML ファイルへのダンプは、次に示すように、システムシェルから mysql クライアントを使用して実現できます。

> **注記**
>
> --xml オプションを指定すると、mysql クライアントは、その出力として XML 形式を使用します。-e オプションを指定すると、クライアントはそのオプションの直後にある SQL ステートメントを実行します。

```
shell> mysql --xml -e "SELECT * FROM test.person" > person-dump.xml
shell> cat person-dump.xml
<?xml version="1.0"?>

<resultset statement="SELECT * FROM test.person" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
 <row>
      <field name="person_id">1</field>
      <field name="fname">Pekka</field>
      <field name="lname">Nousiainen</field>
      <field name="created">2007-07-13 16:18:47</field>
 </row>

 <row>
      <field name="person_id">2</field>
      <field name="fname">Jonas</field>
      <field name="lname">Oreland</field>
      <field name="created">2007-07-13 16:18:47</field>
 </row>

 <row>
      <field name="person_id">3</field>
      <field name="fname">Mikael</field>
      <field name="lname">Rönström</field>
      <field name="created">2007-07-13 16:18:47</field>
 </row>

 <row>
      <field name="person_id">4</field>
      <field name="fname">Lars</field>
      <field name="lname">Thalmann</field>
      <field name="created">2007-07-13 16:18:47</field>
 </row>

 <row>
      <field name="person_id">5</field>
      <field name="fname">Tomas</field>
      <field name="lname">Ulin</field>
      <field name="created">2007-07-13 16:18:47</field>
 </row>

 <row>
      <field name="person_id">6</field>
      <field name="fname">Martin</field>
      <field name="lname">Sköld</field>
      <field name="created">2007-07-13 16:18:47</field>
 </row>
</resultset>
```

次のように、person のコピーを作成したあと、ダンプファイルを新しいテーブルにインポートすることによって、このダンプが有効であることを確認できます。

```
mysql> USE test;
mysql> CREATE TABLE person2 LIKE person;
Query OK, 0 rows affected (0.00 sec)
```

```
mysql> LOAD XML LOCAL INFILE 'person-dump.xml'
    ->   INTO TABLE person2;
Query OK, 6 rows affected (0.01 sec)
Records: 6  Deleted: 0  Skipped: 0  Warnings: 0

mysql> SELECT * FROM person2;
+-----------+--------+------------+---------------------+
| person_id | fname  | lname      | created             |
+-----------+--------+------------+---------------------+
|         1 | Pekka  | Nousiainen | 2007-07-13 16:18:47 |
|         2 | Jonas  | Oreland    | 2007-07-13 16:18:47 |
|         3 | Mikael | Rönström   | 2007-07-13 16:18:47 |
|         4 | Lars   | Thalmann   | 2007-07-13 16:18:47 |
|         5 | Tomas  | Ulin       | 2007-07-13 16:18:47 |
|         6 | Martin | Sköld      | 2007-07-13 16:18:47 |
+-----------+--------+------------+---------------------+
6 rows in set (0.00 sec)
```

ROWS IDENTIFIED BY '<tagname>' 句を使用すると、同じ XML ファイルのデータを定義の異なるデータベーステーブルにインポートできます。この例では、次の XML を含む address.xml という名前のファイルがあるとします。

```
<?xml version="1.0"?>

<list>
  <person person_id="1">
    <fname>Robert</fname>
    <lname>Jones</lname>
    <address address_id="1" street="Mill Creek Road" zip="45365" city="Sidney"/>
    <address address_id="2" street="Main Street" zip="28681" city="Taylorsville"/>
  </person>

  <person person_id="2">
    <fname>Mary</fname>
    <lname>Smith</lname>
    <address address_id="3" street="River Road" zip="80239" city="Denver"/>
    <!-- <address address_id="4" street="North Street" zip="37920" city="Knoxville"/> -->
  </person>

</list>
```

ここでも、このセクションで前に定義された test.person テーブルを使用できます。テーブルの既存のすべてのレコードをクリアしたあと、次に示すようにその構造を表示します。

```
mysql< TRUNCATE person;
Query OK, 0 rows affected (0.04 sec)

mysql< SHOW CREATE TABLE person\G
*************************** 1. row ***************************
       Table: person
Create Table: CREATE TABLE `person` (
  `person_id` int(11) NOT NULL,
  `fname` varchar(40) DEFAULT NULL,
  `lname` varchar(40) DEFAULT NULL,
  `created` timestamp NOT NULL DEFAULT CURRENT_TIMESTAMP ON UPDATE CURRENT_TIMESTAMP,
  PRIMARY KEY (`person_id`)
) ENGINE=MyISAM DEFAULT CHARSET=latin1
1 row in set (0.00 sec)
```

次に、次の CREATE TABLE ステートメントを使用して、test データベース内に address テーブルを作成します。

```
CREATE TABLE address (
    address_id INT NOT NULL PRIMARY KEY,
    person_id INT NULL,
    street VARCHAR(40) NULL,
    zip INT NULL,
    city VARCHAR(40) NULL,
    created TIMESTAMP
);
```

XML ファイルのデータを person テーブルにインポートするには、次に示すように、行が <person> 要素で指定されるように指定する次の LOAD XML ステートメントを実行します。

```
mysql> LOAD XML LOCAL INFILE 'address.xml'
    ->   INTO TABLE person
    ->   ROWS IDENTIFIED BY '<person>';
```

```
Query OK, 2 rows affected (0.00 sec)
Records: 2  Deleted: 0  Skipped: 0  Warnings: 0
```

SELECT ステートメントを使用して、レコードがインポートされたことを確認できます。

```
mysql> SELECT * FROM person;
+-----------+--------+-------+---------------------+
| person_id | fname  | lname | created             |
+-----------+--------+-------+---------------------+
|         1 | Robert | Jones | 2007-07-24 17:37:06 |
|         2 | Mary   | Smith | 2007-07-24 17:37:06 |
+-----------+--------+-------+---------------------+
2 rows in set (0.00 sec)
```

XML ファイル内の <address> 要素は、person テーブル内に対応するカラムがないためスキップされます。

<address> 要素のデータを address テーブルにインポートするには、次に示す LOAD XML ステートメントを使用します。

```
mysql> LOAD XML LOCAL INFILE 'address.xml'
    ->   INTO TABLE address
    ->   ROWS IDENTIFIED BY '<address>';
Query OK, 3 rows affected (0.00 sec)
Records: 3  Deleted: 0  Skipped: 0  Warnings: 0
```

次のような SELECT ステートメントを使用して、データがインポートされたこと確認できます。

```
mysql> SELECT * FROM address;
+------------+-----------+-----------------+-------+--------------+---------------------+
| address_id | person_id | street          | zip   | city         | created             |
+------------+-----------+-----------------+-------+--------------+---------------------+
|          1 |         1 | Mill Creek Road | 45365 | Sidney       | 2007-07-24 17:37:37 |
|          2 |         1 | Main Street     | 28681 | Taylorsville | 2007-07-24 17:37:37 |
|          3 |         2 | River Road      | 80239 | Denver       | 2007-07-24 17:37:37 |
+------------+-----------+-----------------+-------+--------------+---------------------+
3 rows in set (0.00 sec)
```

<address> 要素のデータのうち、XML コメントで囲まれているものはインポートされません。ただし、address テーブルには person_id カラムがあるため、各 <address> に対する親の <person> 要素の person_id 属性の値は address テーブルにインポートされます。

セキュリティー上の考慮事項　LOAD DATA ステートメントと同様に、クライアントホストからサーバーホストへの XML ファイルの転送は MySQL サーバーによって開始されます。理論上は、LOAD XML ステートメント内でクライアントによって指定されたファイルではなく、サーバーが選択したファイルを転送するようにクライアントプログラムに指示する、パッチが適用されたサーバーを構築できます。そのようなサーバーは、クライアントユーザーが読み取りアクセス権を持つクライアントホスト上のすべてのファイルにアクセスできます。

Web 環境では、クライアントは通常、Web サーバーから MySQL に接続します。MySQL サーバーに対して任意のコマンドを実行できるユーザーは、LOAD XML LOCAL を使用して、Web サーバープロセスが読み取りアクセス権を持つどのファイルでも読み取ることができます。この環境では、そのクライアントは MySQL サーバーに対して、Web サーバーに接続するユーザーによって実行されているリモートプログラムではなく、実際に Web サーバーです。

--local-infile=0 または --local-infile=OFF を使用してサーバーを起動することによって、クライアントからの XML ファイルのロードを無効にすることができます。このオプションはまた、クライアントセッションの期間中は LOAD XML を無効にするように mysql クライアントを起動する場合にも使用できます。

クライアントがサーバーから XML ファイルをロードしないようにするために、対応する MySQL ユーザーアカウントには FILE 権限を付与しないようにするか、またはクライアントユーザーアカウントがすでにこの権限を持っている場合は取り消してください。

> **重要**
>
> FILE 権限を取り消した (または、最初から付与しない) 場合、そのユーザーは LOAD XML INFILE ステートメント (および LOAD_FILE() 関数) を実行できなくなるだけです。LOAD XML LOCAL INFILE の実行は妨げられません。このステートメントを禁止するには、サーバーまたはクライアントを --local-infile=OFF で起動する必要があります。
>
> つまり、FILE 権限は、そのクライアントがサーバー上のファイルを読み取れるかどうかにのみ影響を与えます。そのクライアントがローカルファイルシステム上のファイルを読み取れるかどうかには関係しません。

テーブルロックを採用したストレージエンジン (MyISAM など) を使用しているパーティション化されたテーブルの場合、LOAD XML はどのパーティションロックもプルーニングできません。これは、行レベルロックを採用したストレージエンジン (InnoDB など) を使用しているテーブルには適用されません。詳細は、セクション19.6.4「パーティショニングとロック」を参照してください。

13.2.8 REPLACE 構文

```
REPLACE [LOW_PRIORITY | DELAYED]
    [INTO] tbl_name
    [PARTITION (partition_name,...)]
    [(col_name,...)]
    {VALUES | VALUE} ({expr | DEFAULT},...),(...),...
```

または:

```
REPLACE [LOW_PRIORITY | DELAYED]
    [INTO] tbl_name
    [PARTITION (partition_name,...)]
    SET col_name={expr | DEFAULT}, ...
```

または:

```
REPLACE [LOW_PRIORITY | DELAYED]
    [INTO] tbl_name
    [PARTITION (partition_name,...)]
    [(col_name,...)]
    SELECT ...
```

REPLACE は、INSERT とまったく同じように機能します。ただし、テーブル内の古い行に、PRIMARY KEY または UNIQUE インデックスに関して新しい行と同じ値が含まれている場合、その古い行は新しい行が挿入される前に削除されます。セクション13.2.5「INSERT 構文」を参照してください。

REPLACE は、SQL 標準への MySQL 拡張です。これは挿入を行うか、または削除と挿入を行います。標準 SQL への別の MySQL 拡張 (挿入または更新を行います) については、セクション13.2.5.3「INSERT ... ON DUPLICATE KEY UPDATE 構文」を参照してください。

テーブルに PRIMARY KEY または UNIQUE インデックスが存在しないかぎり、REPLACE ステートメントを使用しても何も意味がありません。新しい行が別の行を複製したかどうかを判定するために使用されるインデックスが存在しないため、それは INSERT と同等になります。

すべてのカラムの値が REPLACE ステートメントで指定されている値から取得されます。カラムがない場合は、INSERT での処理と同様に、そのカラムはそのデフォルト値に設定されます。現在の行の値を参照し、それを新しい行で使用することはできません。SET col_name = col_name + 1 などの代入を使用した場合、右側にあるカラム名への参照は DEFAULT(col_name) として処理されるため、この代入は SET col_name = DEFAULT(col_name) + 1 と同等です。

REPLACE を使用するには、このテーブルに対する INSERT 権限と DELETE 権限の両方が必要です。

MySQL 5.6.2 から、REPLACE は、パーティション、サブパーティション、またはその両方の名前のカンマ区切りリストを含む PARTITION オプションを使用した明示的なパーティション選択をサポートしています。INSERT と同様に、これらのいずれかのパーティションまたはサブパーティションに新しい行を挿入できない場合、REPLACE ステートメントは Found a row not matching the given partition set.エラーで失敗します。詳細は、セクション19.5「パーティション選択」を参照してください。

REPLACE は、影響を受けた行数を示す数を返します。これは、削除された行と挿入された行の合計です。この数が単一行の REPLACE に対して 1 である場合は、行が挿入され、削除された行はありませんでした。この数が 1 より大きい場合は、新しい行が挿入される前に 1 つ以上の古い行が削除されました。テーブルに複数の一意のインデックスが存在するときに、新しい行が異なる一意のインデックス内の別の古い行の値を複製した場合は、単一行が複数の古い行を置き換えることがあります。

影響を受けた行数により、REPLACE が行を追加しただけか、または行の置き換えも行なったかを判定することが容易になります。その数が 1 (追加した) か、またはそれより大きい (置き換えた) かをチェックします。

C API を使用している場合は、mysql_affected_rows() 関数を使用して、影響を受けた行数を取得できます。

現在、テーブルへの置き換えを行い、さらにサブクエリーで同じテーブルから選択することはできません。

MySQL は、REPLACE (および LOAD DATA ... REPLACE) に次のアルゴリズムを使用します。

1. テーブルへの新しい行の挿入を試みます

2. 主キーまたは一意のインデックスに関する重複キーエラーが発生したために挿入が失敗している間、次のことを行います。

   a. 重複キー値を含む競合している行をテーブルから削除します

   b. テーブルへの新しい行の挿入を再試行します

重複キーエラーが発生した場合、ストレージエンジンが削除と挿入ではなく、更新として REPLACE を実行する可能性がありますが、そのセマンティクスは同じです。ストレージエンジンが Handler_xxx ステータス変数を増分する方法が異なる可能性がある以外、ユーザーに見える影響はありません。

REPLACE ... SELECT ステートメントの結果は SELECT からの行の順序に依存し、またこの順序を常に保証することはできないため、ロギング時に、これらのステートメントがマスターとスレーブで異なる可能性があります。このため、MySQL 5.6.4 以降では、REPLACE ... SELECT ステートメントには、ステートメントベースのレプリケーションには安全でないというフラグが付けられます。この変更により、このようなステートメントは、STATEMENT バイナリロギングモードを使用しているときはログ内に警告を生成し、MIXED モードを使用しているときは行ベース形式を使用してログに記録されます。セクション17.1.2.1「ステートメントベースおよび行ベースレプリケーションのメリットとデメリット」も参照してください。

パーティション化されていない既存のテーブルをパーティション化に対応するように変更しているときや、すでにパーティション化されたテーブルのパーティション化を変更しているときに、そのテーブルの主キーの変更を検討する可能性があります (セクション19.6.1「パーティショニングキー、主キー、および一意キー」を参照してください)。これを行うと、パーティション化されていないテーブルの主キーを変更した場合と同様に、REPLACE ステートメントの結果が影響を受ける可能性があります。次の CREATE TABLE ステートメントによって作成されたテーブルを考えてみます。

```
CREATE TABLE test (
  id INT UNSIGNED NOT NULL AUTO_INCREMENT,
  data VARCHAR(64) DEFAULT NULL,
  ts TIMESTAMP NOT NULL DEFAULT CURRENT_TIMESTAMP ON UPDATE CURRENT_TIMESTAMP,
  PRIMARY KEY (id)
);
```

このテーブルを作成し、mysql クライアントに示されているステートメントを実行すると、結果は次のようになります。

```
mysql> REPLACE INTO test VALUES (1, 'Old', '2014-08-20 18:47:00');
Query OK, 1 row affected (0.04 sec)

mysql> REPLACE INTO test VALUES (1, 'New', '2014-08-20 18:47:42');
Query OK, 2 rows affected (0.04 sec)

mysql> SELECT * FROM test;
+----+------+---------------------+
| id | data | ts                  |
+----+------+---------------------+
|  1 | New  | 2014-08-20 18:47:42 |
+----+------+---------------------+
1 row in set (0.00 sec)
```

ここで、次に示すように (強調表示されたテキスト) 主キーが 2 つのカラムになっている点を除き、最初のテーブルとほぼ同一の 2 番目のテーブルを作成します。

```
CREATE TABLE test2 (
  id INT UNSIGNED NOT NULL AUTO_INCREMENT,
  data VARCHAR(64) DEFAULT NULL,
  ts TIMESTAMP NOT NULL DEFAULT CURRENT_TIMESTAMP ON UPDATE CURRENT_TIMESTAMP,
  PRIMARY KEY (id, ts)
);
```

元の test テーブルに対して実行したのと同じ 2 つの REPLACE ステートメントを test2 に対して実行すると、異なる結果が得られます。

```
mysql> REPLACE INTO test2 VALUES (1, 'Old', '2014-08-20 18:47:00');
Query OK, 1 row affected (0.05 sec)

mysql> REPLACE INTO test2 VALUES (1, 'New', '2014-08-20 18:47:42');
Query OK, 1 row affected (0.06 sec)

mysql> SELECT * FROM test2;
+----+------+---------------------+
| id | data | ts                  |
+----+------+---------------------+
|  1 | Old  | 2014-08-20 18:47:00 |
```

```
| 1 | New  | 2014-08-20 18:47:42 |
+---+------+---------------------+
2 rows in set (0.00 sec)
```

これは、test2 に対して実行した場合は id カラムと ts カラムの両方の値が、置き換えられる行に対する既存の行の値に一致している必要があり、そうでないと行が挿入されるためです。

MySQL 5.6.6 より前は、テーブルレベルのロックを採用した MyISAM などのストレージエンジンを使用しているパーティション化されたテーブルに影響を与える REPLACE によって、そのテーブルのすべてのパーティションがロックされました。これは、REPLACE ... PARTITION ステートメントにも当てはまりました。(これは、行レベルロックを採用した InnoDB などのストレージエンジンでは発生しておらず、現在も発生しません。)MySQL 5.6.6 以降では、MySQL はパーティションロックプルーニングを使用します。これにより、そのテーブルのどのパーティション化カラムも更新されないかぎり、REPLACE ステートメントの WHERE 句に一致する行を含むパーティションだけが実際にロックされるようになります。そうでなければ、テーブル全体がロックされます。詳細は、セクション19.6.4「パーティショニングとロック」を参照してください。

13.2.9 SELECT 構文

```
SELECT
    [ALL | DISTINCT | DISTINCTROW ]
      [HIGH_PRIORITY]
      [STRAIGHT_JOIN]
      [SQL_SMALL_RESULT] [SQL_BIG_RESULT] [SQL_BUFFER_RESULT]
      [SQL_CACHE | SQL_NO_CACHE] [SQL_CALC_FOUND_ROWS]
    select_expr [, select_expr ...]
    [FROM table_references
      [PARTITION partition_list]
    [WHERE where_condition]
    [GROUP BY {col_name | expr | position}
      [ASC | DESC], ... [WITH ROLLUP]]
    [HAVING where_condition]
    [ORDER BY {col_name | expr | position}
      [ASC | DESC], ...]
    [LIMIT {[offset,] row_count | row_count OFFSET offset}]
    [PROCEDURE procedure_name(argument_list)]
    [INTO OUTFILE 'file_name'
        [CHARACTER SET charset_name]
        export_options
      | INTO DUMPFILE 'file_name'
      | INTO var_name [, var_name]]
    [FOR UPDATE | LOCK IN SHARE MODE]]
```

SELECT は、1 つ以上のテーブルから選択された行を取得するために使用され、UNION ステートメントとサブクエリーを含めることができます。セクション13.2.9.4「UNION 構文」およびセクション13.2.10「サブクエリー構文」を参照してください。

SELECT ステートメントのもっとも一般的に使用される句は次のとおりです。

- 各 select_expr は、取得するカラムを示します。少なくとも 1 つの select_expr が存在する必要があります。
- table_references は、行を取得する 1 つまたは複数のテーブルを示します。その構文については、セクション13.2.9.2「JOIN 構文」で説明されています。
- MySQL 5.6.2 から、SELECT は、table_reference 内のテーブル名のあとにパーティションまたはサブパーティション (またはその両方) のリストを含む PARTITION キーワードを使用した明示的なパーティション選択をサポートしています (セクション13.2.9.2「JOIN 構文」を参照してください)。この場合、行はリストされているパーティションからのみ選択され、テーブルのほかのパーティションはすべて無視されます。詳細および例については、セクション19.5「パーティション選択」を参照してください。

  MySQL 5.6.6 以降では、テーブルレベルのロック (つまり、パーティションロック) を実行する MyISAM などのストレージエンジンを使用しているテーブルからの SELECT ... PARTITION では、PARTITION オプションで指定されているパーティションまたはサブパーティションのみがロックされます。

  詳細は、セクション19.6.4「パーティショニングとロック」を参照してください。
- WHERE 句 (指定されている場合) は、選択されるために行が満たす必要のある 1 つまたは複数の条件を示します。where_condition は、選択される各行に対して true に評価される式です。WHERE 句がない場合、このステートメントはすべての行を選択します。

  WHERE 式では、集約 (サマリー) 関数を除き、MySQL がサポートするすべての関数および演算子を使用できます。セクション9.5「式の構文」および第12章「関数と演算子」を参照してください。

SELECT を使用して、どのテーブルも参照せずに計算された行を取得することもできます。

例:

```
mysql> SELECT 1 + 1;
    -> 2
```

テーブルが参照されない状況では、ダミーのテーブル名として DUAL を指定することが許可されます。

```
mysql> SELECT 1 + 1 FROM DUAL;
    -> 2
```

DUAL は純粋に、すべての SELECT ステートメントに FROM や、場合によってはその他の句が存在することを要求するユーザーの便宜のために用意されています。MySQL は、これらの句を無視する可能性があります。MySQL では、テーブルが参照されない場合でも FROM DUAL は必要ありません。

一般に、使用される句は、正確に構文の説明で示されている順序で指定する必要があります。たとえば、HAVING 句は、すべての GROUP BY 句のあとで、かつすべての ORDER BY 句の前にある必要があります。例外として、INTO 句は、構文の説明で示されている位置か、または select_expr リストの直後のどちらにも現れることができます。INTO の詳細は、セクション13.2.9.1「SELECT ... INTO 構文」を参照してください。

select_expr 項のリストは、どのカラムを取得するかを示す選択リストで構成されています。これらの項はカラムや式を指定するか、または * の短縮形を使用できます。

- 1 つの修飾されていない * のみから成る選択リストは、すべてのテーブルのすべてのカラムを選択するための短縮形として使用できます。

  ```
  SELECT * FROM t1 INNER JOIN t2 ...
  ```

- tbl_name.* は、指定されたテーブルのすべてのカラムを選択するための修飾された短縮形として使用できます。

  ```
  SELECT t1.*, t2.* FROM t1 INNER JOIN t2 ...
  ```

- 修飾されていない * を選択リスト内のほかの項目とともに使用すると、解析エラーが生成される可能性があります。この問題を回避するには、修飾された tbl_name.* 参照を使用します。

  ```
  SELECT AVG(score), t1.* FROM t1 ...
  ```

次のリストは、その他の SELECT 句に関する追加情報を示しています。

- AS alias_name を使用して、select_expr にエイリアスを指定できます。エイリアスは式のカラム名として使用され、GROUP BY、ORDER BY、または HAVING 句で使用できます。例:

  ```
  SELECT CONCAT(last_name,', ',first_name) AS full_name
    FROM mytable ORDER BY full_name;
  ```

  select_expr にエイリアスとして識別子を指定する場合、AS キーワードはオプションです。前の例は、次のように記述することもできました。

  ```
  SELECT CONCAT(last_name,', ',first_name) full_name
    FROM mytable ORDER BY full_name;
  ```

  ただし、AS はオプションであるため、2 つの select_expr 式の間のカンマを忘れると、軽微な問題が発生する可能性があります。MySQL は、2 番目の式をエイリアスとして解釈します。たとえば、次のステートメントでは、columnb はエイリアスとして処理されます。

  ```
  SELECT columna columnb FROM mytable;
  ```

  このため、カラムのエイリアスを指定するときは AS を明示的に使用するようにすることをお勧めします。

  WHERE 句が実行されるときはまだカラム値が決定されていない可能性があるため、WHERE 句内でカラムのエイリアスを参照することは許可されません。セクションB.5.5.4「カラムエイリアスに関する問題」を参照してください。

- FROM table_references 句は、行を取得する 1 つまたは複数のテーブルを示します。複数のテーブルを指定すると、結合が実行されます。結合構文については、セクション13.2.9.2「JOIN 構文」を参照してください。指定されたテーブルごとに、オプションでエイリアスを指定できます。

  ```
  tbl_name [[AS] alias] [index_hint]
  ```

インデックスヒントを使用すると、クエリー処理中にインデックスを選択する方法に関する情報がオプティマイザに提供されます。これらのヒントを指定するための構文については、セクション13.2.9.3「インデックスヒントの構文」を参照してください。

代わりの方法として SET max_seeks_for_key=value を使用して、MySQL にテーブルスキャンの代わりにキースキャンを強制的に実行させることができます。セクション5.1.4「サーバーシステム変数」を参照してください。

- データベースを明示的に指定するために、デフォルトデータベース内でテーブルを tbl_name または db_name.tbl_name として参照できます。カラムを col_name、tbl_name.col_name または db_name.tbl_name.col_name として参照できます。参照があいまいにならないかぎり、カラム参照のために tbl_name または db_name.tbl_name プリフィクスを指定する必要はありません。より明示的なカラム参照形式を必要とするあいまいさの例については、セクション9.2.1「識別子の修飾子」を参照してください。

- tbl_name AS alias_name または tbl_name alias_name を使用して、テーブル参照に工イリアスを指定できます。

```
SELECT t1.name, t2.salary FROM employee AS t1, info AS t2
  WHERE t1.name = t2.name;

SELECT t1.name, t2.salary FROM employee t1, info t2
  WHERE t1.name = t2.name;
```

- カラム名、カラムの工イリアス、またはカラム位置を使用して、出力のために選択されたカラムを ORDER BY および GROUP BY 句で参照できます。カラム位置は整数であり、1 から始まります。

```
SELECT college, region, seed FROM tournament
  ORDER BY region, seed;

SELECT college, region AS r, seed AS s FROM tournament
  ORDER BY r, s;

SELECT college, region, seed FROM tournament
  ORDER BY 2, 3;
```

逆の順序でソートするには、ソートに使用する ORDER BY 句内のカラムの名前に DESC (降順) キーワードを追加します。デフォルトは昇順です。これは、ASC キーワードを使用して明示的に指定できます。

ORDER BY がサブクエリー内で発生し、外側のクエリー内でも適用される場合は、もっとも外側の ORDER BY が優先されます。たとえば、次のステートメントの結果は昇順ではなく、降順でソートされます。

```
(SELECT ... ORDER BY a) ORDER BY a DESC;
```

カラム位置の使用は、この構文が SQL 標準から削除されたため非推奨です。

- GROUP BY を使用すると、出力行は、同じカラムに対して ORDER BY を指定したかのように GROUP BY カラムに従ってソートされます。GROUP BY によって生成されるソートのオーバーヘッドを回避するには、ORDER BY NULL を追加します。

```
SELECT a, COUNT(b) FROM test_table GROUP BY a ORDER BY NULL;
```

MySQL 5.6 における暗黙の GROUP BY ソートへの依存は、非推奨になっています。グループ化された結果の特定のソート順序を実現するには、明示的な ORDER BY 句を使用することをお勧めします。GROUP BY ソートは、たとえば、オプティマイザがもっとも効率的であると考えるどのような方法でも、グループ化を指示できるようにしたり、ソートオーバーヘッドを回避したりするためなどに、今後のリリースで変更される可能性のある MySQL 拡張機能です。

- MySQL では GROUP BY 句が拡張され、この句で指定されているカラムのあとに ASC と DESC も指定できるようになっています。

```
SELECT a, COUNT(b) FROM test_table GROUP BY a DESC;
```

- MySQL では、GROUP BY の使用が、GROUP BY 句で指定されていないフィールドの選択を許可するように拡張されています。クエリーから期待する結果が得られない場合は、セクション12.19「GROUP BY 句で使用される関数と修飾子」にある GROUP BY の説明を参照してください。

- GROUP BY では、WITH ROLLUP 修飾子が許可されます。セクション12.19.2「GROUP BY 修飾子」を参照してください。

- HAVING 句は、ほぼ最後 (項目がクライアントに送信される直前) に最適化なしで適用されます。(LIMIT は HAVING のあとに適用されます。)

SQL 標準では、HAVING は GROUP BY 句内のカラムか、または集約関数で使用されるカラムしか参照できません。ただし、MySQL ではこの動作への拡張がサポートされており、HAVING が SELECT リスト内のカラムや外側サブクエリー内のカラムを参照することも許可されます。

HAVING 句があいまいなカラムを参照している場合は、警告が発生します。次のステートメントにある col2 は、エイリアスとカラム名の両方として使用されているため、あいまいです。

```
SELECT COUNT(col1) AS col2 FROM t GROUP BY col2 HAVING col2 = 2;
```

標準 SQL の動作の方が優先されるため、HAVING のカラム名が GROUP BY で使用されると同時に、出力カラムリスト内のエイリアスが指定されたカラムとしても使用されている場合は、GROUP BY カラム内のカラムが優先されます。

- WHERE 句に含めるべき項目には HAVING を使用しないでください。たとえば、次のように記述しないでください。

```
SELECT col_name FROM tbl_name HAVING col_name > 0;
```

代わりに、次のように記述してください。

```
SELECT col_name FROM tbl_name WHERE col_name > 0;
```

- HAVING 句は、WHERE 句が参照できない集約関数を参照できます。

```
SELECT user, MAX(salary) FROM users
  GROUP BY user HAVING MAX(salary) > 10;
```

(これは、一部の古いバージョンの MySQL では機能しませんでした。)

- MySQL では、重複したカラム名が許可されます。つまり、同じ名前を持つ複数の select_expr が存在できます。これは、標準 SQL の拡張です。MySQL では GROUP BY や HAVING が select_expr 値を参照することも許可されるため、これにより、あいまいさが発生する場合があります。

```
SELECT 12 AS a, a FROM t GROUP BY a;
```

このステートメントでは、どちらのカラムの名前も a です。グループ化のために正しいカラムが使用されるようにするために、select_expr ごとに異なる名前を使用してください。

- MySQL は、ORDER BY 句内の修飾されていないカラムまたはエイリアス参照を、まず select_expr 値、次に FROM 句内のテーブルのカラム内を検索することによって解決します。GROUP BY または HAVING 句の場合は、select_expr 値内を検索する前に FROM 句を検索します。(GROUP BY と HAVING について、これは、ORDER BY) の場合と同じルールを使用していた MySQL 5.0 より前の動作とは異なります。

- LIMIT 句を使用すると、SELECT ステートメントによって返される行数を制約できます。LIMIT は 1 つまたは 2 つの数値引数を受け取ります。これは、どちらも負ではない整定数である必要があります。ただし、次の例外があります。

  - 準備済みステートメント内では、? プレースホルダマーカーを使用して LIMIT パラメータを指定できます。
  - ストアドプログラム内では、整数値のルーチンパラメータまたはローカル変数を使用して LIMIT パラメータを指定できます。

  引数が 2 つの場合、最初の引数は返す先頭行のオフセットを指定し、2 番目の引数は返す行の最大数を指定します。最初の行のオフセットは (1 ではなく) 0 です。

```
SELECT * FROM tbl LIMIT 5,10;  # Retrieve rows 6-15
```

特定のオフセットから結果セットの最後までのすべての行を取得するために、2 番目のパラメータにある程度大きい数字を使用できます。次のステートメントは、96 行目から最後の行までのすべての行を取得します。

```
SELECT * FROM tbl LIMIT 95,18446744073709551615;
```

引数が 1 つの場合、この値は、結果セットの先頭から返す行数を指定します。

```
SELECT * FROM tbl LIMIT 5;     # Retrieve first 5 rows
```

つまり、LIMIT row_count は LIMIT 0, row_count と同等です。

準備済みステートメントでは、プレースホルダを使用できます。次のステートメントは、tbl テーブルの 1 行を返します。

```
SET @a=1;
PREPARE STMT FROM 'SELECT * FROM tbl LIMIT ?';
EXECUTE STMT USING @a;
```

次のステートメントは、tbl テーブルの 2 行目から 6 行目までを返します。

```
SET @skip=1; SET @numrows=5;
PREPARE STMT FROM 'SELECT * FROM tbl LIMIT ?, ?';
EXECUTE STMT USING @skip, @numrows;
```

PostgreSQL との互換性のために、MySQL は LIMIT row_count OFFSET offset 構文もサポートしています。

LIMIT がサブクエリー内に現れ、また外部クエリーでも適用される場合は、もっとも外側の LIMIT が優先されます。たとえば、次のステートメントは、1 行ではなく 2 行を生成します。

```
(SELECT ... LIMIT 1) LIMIT 2;
```

- PROCEDURE 句は、結果セット内のデータを処理するプロシージャーを指定します。例については、セクション8.4.2.4「PROCEDURE ANALYSE の使用」を参照してください。ここでは、テーブルサイズの削減に役立つ可能性のある最適なカラムデータ型に関する提案を得るために使用できるプロシージャーである ANALYSE について説明しています。

- SELECT の SELECT ... INTO 形式を使用すると、クエリー結果をファイルに書き込んだり、変数に格納したりできます。詳細は、セクション13.2.9.1「SELECT ... INTO 構文」を参照してください。

- ページまたは行ロックを使用するストレージエンジンで FOR UPDATE を使用した場合、クエリーによって検査される行は、現在のトランザクションの最後まで書き込みがロックされます。LOCK IN SHARE MODE を使用すると、その検査される行のほかのトランザクションによる読み取りは許可するが、その更新または削除を許可しない共有ロックが設定されます。セクション14.2.5「ロック読み取り (SELECT ... FOR UPDATE および SELECT ... LOCK IN SHARE MODE)」を参照してください。

  さらに、CREATE TABLE new_table SELECT ... FROM old_table ... などのステートメントで SELECT の一部として FOR UPDATE を使用することはできません。(それを行おうとすると、このステートメントはエラー Can't update table 'old_table' while 'new_table' is being created で拒否されます。)これは、CREATE TABLE ... SELECT ステートメントが作成されているテーブル以外のテーブルで変更を行うことを許可していた、MySQL 5.5 およびそれ以前からの動作の変更です。

SELECT キーワードのあとに、このステートメントの操作に影響を与えるいくつかのオプションを使用できます。HIGH_PRIORITY、STRAIGHT_JOIN、および SQL_ で始まるオプションは、標準 SQL への MySQL 拡張です。

- ALL および DISTINCT オプションは、重複した行を返すかどうかを指定します。ALL (デフォルト) は、重複を含め、一致するすべての行を返すように指定します。DISTINCT は、重複した行の結果セットからの削除を指定します。両方のオプションを指定するとエラーになります。DISTINCTROW は DISTINCT のシノニムです。

- HIGH_PRIORITY は SELECT に、テーブルを更新するステートメントより高い優先度を与えます。これは、非常に高速であり、かつただちに実行する必要のあるクエリーにのみ使用するようにしてください。テーブルが読み取りに対してロックされている間に発行された SELECT HIGH_PRIORITY クエリーは、そのテーブルが未使用になるのを待機している更新ステートメントが存在する場合でも実行されます。これは、テーブルレベルロックのみを使用するストレージエンジン (MyISAM、MEMORY、および MERGE) にのみ影響を与えます。

  HIGH_PRIORITY を、UNION の一部である SELECT ステートメントで使用することはできません。

- STRAIGHT_JOIN は、オプティマイザに、テーブルを FROM 句にリストされている順序で強制的に結合させます。オプティマイザがテーブルを最適でない順序で結合する場合は、これを使用してクエリーを高速化できます。STRAIGHT_JOIN はまた、table_references リストでも使用できます。セクション13.2.9.2「JOIN 構文」を参照してください。

  STRAIGHT_JOIN は、オプティマイザが const テーブルまたは system テーブルとして処理するテーブルには適用されません。このようなテーブルは単一行を生成し、クエリー実行の最適化フェーズ中に読み取られます。また、そのカラムへの参照は、クエリー実行が続行される前に適切なカラム値で置き換えられます。これらのテーブルは、EXPLAIN によって表示されるクエリー計画に最初に表示されます。「セクション8.8.1「EXPLAIN によるクエリーの最適化」」を参照してください。この例外は、外部結合の NULL で補完された側で使用されている const テーブルまたは system テーブル (つまり、LEFT JOIN の右側のテーブルまたは RIGHT JOIN の左側のテーブル) には適用されない可能性があります。

- SQL_BIG_RESULT または SQL_SMALL_RESULT は、結果セットの行数がそれぞれ多いこと、または少ないことをオプティマイザに通知するために GROUP BY または DISTINCT とともに使用できま

す。SQL_BIG_RESULT の場合、MySQL は、必要に応じてディスクベースの一時テーブルを直接使用し、ソートでは GROUP BY 要素に関するキーで一時テーブルを使用することを優先します。SQL_SMALL_RESULT の場合、MySQL はソートを使用する代わりに、高速な一時テーブルを使用して結果のテーブルを格納します。これは、通常は必要ないはずです。

- SQL_BUFFER_RESULT は、結果を強制的に一時テーブルに配置します。これは、MySQL がテーブルロックを早期に解放する場合や、結果セットをクライアントに送信するのに長い時間がかかる場合に役立ちます。このオプションは、サブクエリーまたは後続の UNION ではなく、トップレベルの SELECT ステートメントでのみ使用できます。
- SQL_CALC_FOUND_ROWS は MySQL に、LIMIT 句をすべて無視して、結果セットに含まれる行数を計算するよう指示します。そのあと、行数は SELECT FOUND_ROWS() を使用して取得できます。セクション 12.14「情報関数」を参照してください。
- SQL_CACHE および SQL_NO_CACHE オプションは、クエリーキャッシュ内のクエリー結果のキャッシュに影響を与えます (セクション8.9.3「MySQL クエリーキャッシュ」を参照してください)。SQL_CACHE は MySQL に、結果がキャッシュ可能であり、query_cache_type システム変数の値が 2 または DEMAND である場合は、結果をクエリーキャッシュに格納するよう指示します。SQL_NO_CACHE を指定すると、サーバーはクエリーキャッシュを使用しません。それは、結果がすでにキャッシュされているかどうかを確認するためにクエリーキャッシュをチェックせず、クエリー結果もキャッシュしません。(パーサーの制限のため、スペース文字の前後に SQL_NO_CACHE キーワードを付ける必要があります。改行などのスペース以外では、結果がすでにキャッシュされているかどうかを確認するために、サーバーにクエリーキャッシュをチェックさせます。)

  ビューの場合、SQL_NO_CACHE は、クエリー内のいずれかの SELECT に現れた場合に適用されます。キャッシュ可能なクエリーでは、SQL_CACHE は、そのクエリーによって参照されるビューの最初の SELECT に現れた場合に適用されます。

  MySQL 5.6 では、これらの 2 つのオプションは相互に排他的であり、両方が指定された場合はエラーが発生します。また、これらのオプションは、サブクエリー (FROM 句内のサブクエリーを含む) や、最初の SELECT を除く和集合内の SELECT ステートメント内では許可されません。

MySQL 5.6.6 より前は、テーブルレベルのロックを採用した MyISAM などのストレージエンジンを使用しているパーティション化されたテーブルからの SELECT によって、そのテーブルのすべてのパーティションがロックされました。これは、SELECT ... PARTITION クエリーにも当てはまりました。(これは、行レベルロックを採用した InnoDB などのストレージエンジンでは発生しておらず、現在も発生しません。)MySQL 5.6.6 以降では、MySQL はパーティションロックプルーニングを使用します。これにより、SELECT ステートメントの WHERE 句に一致する行を含むパーティションだけが実際にロックされるようになります。詳細は、セクション 19.6.4「パーティショニングとロック」を参照してください。

13.2.9.1 SELECT ... INTO 構文

SELECT の SELECT ... INTO 形式を使用すると、クエリー結果を変数に格納したり、ファイルに書き込んだりできます。

- SELECT ... INTO var_list はカラム値を選択し、それらを変数に格納します。
- SELECT ... INTO OUTFILE は、選択された行をファイルに書き込みます。カラムおよび行ターミネータを指定すると、特定の出力形式を生成できます。
- SELECT ... INTO DUMPFILE は、単一行をファイルに形式設定なしで書き込みます。

SELECT の構文の説明 (セクション13.2.9「SELECT 構文」を参照してください) では、INTO 句がステートメントの最後の近くに示されています。INTO はまた、select_expr リストの直後に使用することもできます。

ネストされた SELECT はその結果を外側のコンテキストに返す必要があるため、このような SELECT では INTO 句を使用してはいけません。

INTO 句は、1 つ以上の変数のリストを指定できます。この変数には、ユーザー定義変数、ストアドプロシージャーやストアドファンクションのパラメータ、またはストアドプログラムのローカル変数を指定できます。(準備済み SELECT ... INTO OUTFILE ステートメント内では、ユーザー定義変数のみが許可されます。セクション 13.6.4.2「ローカル変数のスコープと解決」を参照してください。)

選択された値は変数に割り当てられます。変数の数がカラム数に一致している必要があります。クエリーは、単一行を返すようにしてください。クエリーが行を返さない場合は、エラーコード 1329 で警告が発生し (No data)、変数値は変更されないままになります。クエリーが複数の行を返す場合は、エラー 1172 が発生します (結果が 2 行以上です)。このステートメントが複数の行を取得する可能性がある場合は、LIMIT 1 を使用して結果セットを単一行に制限できます。

```
SELECT id, data INTO @x, @y FROM test.t1 LIMIT 1;
```

ユーザー変数名では大文字と小文字を区別しません。セクション9.4「ユーザー定義変数」を参照してください。

SELECT の SELECT ... INTO OUTFILE 'file_name' 形式は、選択された行をファイルに書き込みます。このファイルはサーバーホスト上で作成されるため、この構文を使用するには FILE 権限が必要です。file_name を既存のファイルにすることはできません。これにより、特に /etc/passwd などのファイルやデータベーステーブルが破棄されることが回避されます。character_set_filesystem システム変数は、ファイル名の解釈を制御します。

SELECT ... INTO OUTFILE ステートメントは、主に、テーブルをサーバーマシン上のテキストファイルに非常にすばやくダンプできるようにすることを目的にしています。結果として得られるファイルをサーバーホスト以外のホスト上で作成する場合は通常、そのサーバーホストのファイルシステムを基準にしたファイルへのパスを記述する方法が存在しないため、SELECT ... INTO OUTFILE は使用できません。

ただし、MySQL クライアントソフトウェアがリモートマシンにインストールされている場合は、代わりに mysql -e "SELECT ..." > file_name などのクライアントコマンドを使用してクライアントホスト上にファイルを生成できます。

また、サーバーのファイルシステム上でネットワークにマップされたパスを使用してリモートホスト上のファイルの場所にアクセスできる場合も、結果として得られるファイルをサーバーホストとは異なるホスト上に作成できます。この場合は、ターゲットホスト上に mysql (または、その他の何らかの MySQL クライアントプログラム) が存在する必要はありません。

SELECT ... INTO OUTFILE は、LOAD DATA INFILE を補完するものです。カラム値は、CHARACTER SET 句で指定されている文字セットに変換されて書き込まれます。このような句が存在しない場合、値は binary 文字セットを使用してダンプされます。事実上、文字セットの変換は実行されません。結果セットにカラムが複数の文字セットで含まれている場合は、出力データファイルにもそのまま含まれるため、そのファイルを正しくリロードできない可能性があります。

このステートメントの export_options 部分の構文は、LOAD DATA INFILE ステートメントで使用されるのと同じ FIELDS および LINES 句で構成されています。FIELDS および LINES 句 (それぞれのデフォルト値と許可される値を含む) については、セクション13.2.6「LOAD DATA INFILE 構文」を参照してください。

FIELDS ESCAPED BY は、特殊文字を書き込む方法を制御します。FIELDS ESCAPED BY 文字が空でない場合、その文字は、出力上で次の文字の前に付けられるプリフィクスとして、あいまいさを避けるために必要な場合に使用されます。

- FIELDS ESCAPED BY 文字
- FIELDS [OPTIONALLY] ENCLOSED BY 文字
- FIELDS TERMINATED BY および LINES TERMINATED BY 値の最初の文字
- ASCII NUL (0 の値のバイト。エスケープ文字のあとに実際に書き込まれる文字は 0 の値のバイトではなく、ASCII の「0」です)

FIELDS TERMINATED BY、ENCLOSED BY、ESCAPED BY、または LINES TERMINATED BY 文字は、そのファイルを確実に読み戻すことができるように、エスケープする必要があります。ASCII NUL は、一部のページャーで見やすくなるようにエスケープされます。

結果として得られるファイルは SQL 構文に準拠する必要はないため、ほかは何もエスケープする必要がありません。

FIELDS ESCAPED BY 文字が空である場合は、どの文字もエスケープされず、NULL は \N ではなく、NULL として出力されます。特に、データ内のフィールド値に今指定したリスト内のいずれかの文字が含まれている場合、空のエスケープ文字を指定することはおそらく適切な方法ではありません。

多くのプログラムで使用されているカンマ区切り値 (CSV) 形式のファイルを生成する例を次に示します。

```
SELECT a,b,a+b INTO OUTFILE '/tmp/result.txt'
  FIELDS TERMINATED BY ',' OPTIONALLY ENCLOSED BY ""
  LINES TERMINATED BY '\n'
  FROM test_table;
```

INTO OUTFILE の代わりに INTO DUMPFILE を使用した場合、MySQL はエスケープ処理を実行することなく、カラムや行の終了のない 1 行のみをファイルに書き込みます。これは、ファイルに BLOB 値を格納する場合に役立ちます。

> **注記**
>
> INTO OUTFILE または INTO DUMPFILE によって作成されたファイルはすべて、サーバーホスト上のすべてのユーザーから書き込み可能です。これは、MySQL サーバー

> が、それを実行しているアカウントを持つユーザー以外のどのユーザーによって所有されるファイルも作成できないためです。(これらの理由のために、mysqld を root としては決して実行しないでください。) したがって、このファイルは、その内容を操作できるようにすべてのユーザーから書き込み可能である必要があります。
>
> secure_file_priv システム変数が空以外のディレクトリ名に設定されている場合、書き込まれるファイルはそのディレクトリ内に存在する必要があります。

イベントスケジューラによって実行されるイベントの一部として実行された SELECT ... INTO ステートメントのコンテキストでは、診断メッセージ (エラーだけでなく、警告も含みます) がエラーログに (Windows ではアプリケーションイベントログにも) 書き込まれます。詳細は、セクション20.4.5「イベントスケジューラのステータス」を参照してください。

13.2.9.2 JOIN 構文

MySQL は、SELECT ステートメントと複数テーブルの DELETE および UPDATE ステートメントの table_references 部分に対して次の JOIN 構文をサポートします。

```
table_references:
    escaped_table_reference [, escaped_table_reference] ...

escaped_table_reference:
    table_reference
  | { OJ table_reference }

table_reference:
    table_factor
  | join_table

table_factor:
    tbl_name [PARTITION (partition_names)]
        [[AS] alias] [index_hint_list]
  | table_subquery [AS] alias
  | ( table_references )

join_table:
    table_reference [INNER | CROSS] JOIN table_factor [join_condition]
  | table_reference STRAIGHT_JOIN table_factor
  | table_reference STRAIGHT_JOIN table_factor ON conditional_expr
  | table_reference {LEFT|RIGHT} [OUTER] JOIN table_reference join_condition
  | table_reference NATURAL [{LEFT|RIGHT} [OUTER]] JOIN table_factor

join_condition:
    ON conditional_expr
  | USING (column_list)

index_hint_list:
    index_hint [, index_hint] ...

index_hint:
    USE {INDEX|KEY}
      [FOR {JOIN|ORDER BY|GROUP BY}] ([index_list])
  | IGNORE {INDEX|KEY}
      [FOR {JOIN|ORDER BY|GROUP BY}] (index_list)
  | FORCE {INDEX|KEY}
      [FOR {JOIN|ORDER BY|GROUP BY}] (index_list)

index_list:
    index_name [, index_name] ...
```

テーブル参照は、結合式とも呼ばれます。

MySQL 5.6.2 以降では、テーブル参照 (パーティション化されたテーブルを参照する場合) には、パーティション、サブパーティション、またはその両方のカンマ区切りリストを含む PARTITION オプションを含めることができます。このオプションはテーブルの名前のあとで、かつエイリアス宣言 (存在する場合) の前に指定されます。このオプションの効果は、リストされているパーティションまたはサブパーティションからのみ行が選択されることです。つまり、そのリストで指定されていないパーティションまたはサブパーティションはすべて無視されます。詳細は、セクション19.5「パーティション選択」を参照してください。

table_factor の構文は SQL 標準と比較して拡張されています。後者は table_reference のみを受け付け、かっこ内のそれらのリストは受け付けません。

これは、table_reference 項目のリストの各カンマを内部結合と同等とみなす場合、保守的な拡張です。例:

```
SELECT * FROM t1 LEFT JOIN (t2, t3, t4)
```

```
            ON (t2.a=t1.a AND t3.b=t1.b AND t4.c=t1.c)
```

次と同等です。

```
SELECT * FROM t1 LEFT JOIN (t2 CROSS JOIN t3 CROSS JOIN t4)
            ON (t2.a=t1.a AND t3.b=t1.b AND t4.c=t1.c)
```

MySQL では、JOIN、CROSS JOIN、および INNER JOIN は構文上同等です (互いに置き換えることができます)。標準 SQL では、それらは同等ではありません。INNER JOIN は ON 句とともに使用され、CROSS JOIN はそれ以外のときに使用されます。

一般に、内部結合操作のみを含む結合式内のかっこは無視できます。MySQL はまた、ネストされた結合もサポートしています (セクション8.2.1.11「ネストした結合の最適化」を参照してください)。

インデックスヒントを指定すると、MySQL オプティマイザによるインデックスの使用方法に影響を与えることができます。詳細は、セクション13.2.9.3「インデックスヒントの構文」を参照してください。

次のリストは、結合を記述するときに考慮に入れるべき一般的な要因について説明しています。

- テーブル参照には、tbl_name AS alias_name または tbl_name alias_name を使用してエイリアスを指定できます。

  ```
  SELECT t1.name, t2.salary
    FROM employee AS t1 INNER JOIN info AS t2 ON t1.name = t2.name;

  SELECT t1.name, t2.salary
    FROM employee t1 INNER JOIN info t2 ON t1.name = t2.name;
  ```

- table_subquery は、FROM 句内のサブクエリーとも呼ばれます。サブクエリー結果にテーブル名を付けるには、このようなサブクエリーにエイリアスを含める必要があります。簡単な例を次に示します。セクション13.2.10.8「FROM 句内のサブクエリー」も参照してください。

  ```
  SELECT * FROM (SELECT 1, 2, 3) AS t1;
  ```

- 結合条件が存在しない場合、INNER JOIN と , (カンマ) は意味的に同等です。どちらも、指定されたテーブル間のデカルト積を生成します (つまり、最初のテーブル内のすべての各行が 2 番目のテーブル内のすべての各行に結合されます)。

  ただし、カンマ演算子の優先順位は、INNER JOIN、CROSS JOIN、LEFT JOIN などの優先順位より低くなります。結合条件が存在するときにカンマ結合をほかの結合型と混在させた場合は、「カラム 'col_name' は 'on clause' にはありません」という形式のエラーが発生する可能性があります。この問題への対処に関する情報は、このセクションのあとの方で提供します。

- ON とともに使用される conditional_expr は、WHERE 句で使用できる形式の任意の条件式です。一般に、テーブルの結合方法を指定する条件には ON 句を、また結果セット内に必要な行を制限するには WHERE 句を使用してください。

- LEFT JOIN 内の ON または USING 部分にある右側のテーブルに一致する行が存在しない場合は、すべてのカラムが NULL に設定された行が右側のテーブルに使用されます。このことを使用して、別のテーブルに対応する行が存在しないテーブル内の行を検索できます。

  ```
  SELECT left_tbl.*
    FROM left_tbl LEFT JOIN right_tbl ON left_tbl.id = right_tbl.id
    WHERE right_tbl.id IS NULL;
  ```

  この例では、right_tbl に存在しない id 値を持つ left_tbl 内のすべての行 (つまり、right_tbl 内に対応する行のない left_tbl 内のすべての行) を検索します。これは、right_tbl.id が NOT NULL として宣言されていることを前提にしています。セクション8.2.1.9「LEFT JOIN および RIGHT JOIN の最適化」を参照してください。

- USING(column_list) 句は、両方のテーブル内に存在する必要のあるカラムのリストを指定します。テーブル a と b の両方にカラム c1、c2、および c3 が含まれている場合、次の結合は、この 2 つのテーブルの対応するカラムを比較します。

  ```
  a LEFT JOIN b USING (c1,c2,c3)
  ```

- 2 つのテーブルの NATURAL [LEFT] JOIN は、両方のテーブル内に存在するすべてのカラムを指定する USING 句を含む INNER JOIN または LEFT JOIN と意味的に同等であるとして定義されます。

- RIGHT JOIN は、LEFT JOIN と同じように機能します。コードをデータベース間で移植可能な状態に維持するため、RIGHT JOIN の代わりに LEFT JOIN を使用することをお勧めします。

- 結合構文の説明に示されている { OJ ... } 構文は、ODBC との互換性のためにのみ存在します。構文内のカールした中括弧は文字どおりに書き込まれる必要があります。それらは構文説明の別の部分で利用されているようなメタ構文ではありません。

```
SELECT left_tbl.*
  FROM { OJ left_tbl LEFT OUTER JOIN right_tbl ON left_tbl.id = right_tbl.id }
  WHERE right_tbl.id IS NULL;
```

  { OJ ... } 内では、INNER JOIN や RIGHT OUTER JOIN などのほかの型の結合を使用できます。これは、一部のサードパーティー製アプリケーションとの互換性に役立ちますが、正式な ODBC 構文ではありません。

- STRAIGHT_JOIN は、左側のテーブルが常に右側のテーブルの前に読み取られる点を除き、JOIN と同じです。これは、結合オプティマイザがテーブルを間違った順序で配置する (数少ない) 場合に使用できます。

結合のいくつかの例:

```
SELECT * FROM table1, table2;

SELECT * FROM table1 INNER JOIN table2 ON table1.id=table2.id;

SELECT * FROM table1 LEFT JOIN table2 ON table1.id=table2.id;

SELECT * FROM table1 LEFT JOIN table2 USING (id);

SELECT * FROM table1 LEFT JOIN table2 ON table1.id=table2.id
 LEFT JOIN table3 ON table2.id=table3.id;
```

MySQL 5.0.12 での結合処理の変更

> **注記**
>
> 自然結合や USING を使用した結合 (外部結合のバリアントを含む) は、SQL:2003 標準に従って処理されます。その目標は、NATURAL JOIN と JOIN ... USING に関連した MySQL の構文とセマンティクスを SQL:2003 に合わせることでした。ただし、結合処理でのこれらの変更によって、一部の結合で異なる出力カラムが生成される可能性があります。また、古いバージョン (5.0.12 より前) では正しく機能するように見えた一部のクエリーを、この標準に準拠するように書き換える必要があります。

これらの変更の主要な側面として、次の 5 つがあります。

- MySQL が NATURAL または USING 結合操作の結果カラム (したがって、FROM 句全体の結果) を決定する方法。
- SELECT * および SELECT tbl_name.* の選択されたカラムのリストへの展開。
- NATURAL または USING 結合でのカラム名の解決。
- NATURAL または USING 結合の JOIN ... ON への変換。
- JOIN ... ON の ON 条件でのカラム名の解決。

次のリストは、現在の結合処理のいくつかの効果を古いバージョンでの結合処理と比較した場合のさらに詳細な情報を示しています。「以前」という用語は、「MySQL 5.0.12 より前」を示しています。

- NATURAL 結合または USING 結合のカラムが以前とは異なる可能性があります。具体的には、冗長な出力カラムが表示されなくなっており、また SELECT * の展開でのカラムの順序が以前とは異なる可能性があります。

  次の一連のステートメントを考えてみます。

```
CREATE TABLE t1 (i INT, j INT);
CREATE TABLE t2 (k INT, j INT);
INSERT INTO t1 VALUES(1,1);
INSERT INTO t2 VALUES(1,1);
SELECT * FROM t1 NATURAL JOIN t2;
SELECT * FROM t1 JOIN t2 USING (j);
```

  以前は、これらのステートメントによって次の出力が生成されました。

```
+------+------+------+------+
| i    | j    | k    | j    |
+------+------+------+------+
|    1 |    1 |    1 |    1 |
+------+------+------+------+
+------+------+------+------+
| i    | j    | k    | j    |
```

```
+------+------+------+------+
|   1 |   1 |   1 |   1 |
+------+------+------+------+
```

最初の SELECT ステートメントでは、カラム j は両方のテーブルに現れるため、結合カラムになります。そのため、標準 SQL に従って、出力には 2 回ではなく 1 回だけ表示されるべきです。同様に、2 番目の SELECT ステートメントでは、カラム j は USING 句で指定されているため、出力には 2 回ではなく 1 回だけ表示されるべきです。ただし、どちらの場合も、冗長なカラムは削除されていません。また、標準 SQL に従うとカラムの順序も正しくありません。

現在は、このステートメントによって次の出力が生成されます。

```
+------+------+------+
| j   | i   | k   |
+------+------+------+
|   1 |   1 |   1 |
+------+------+------+
+------+------+------+
| j   | i   | k   |
+------+------+------+
|   1 |   1 |   1 |
+------+------+------+
```

冗長なカラムは削除され、カラムの順序も標準 SQL に従って正しくなっています。

- 最初に、結合された 2 つのテーブルの合体した共通カラムが、最初のテーブルに現れた順序で
- 2 番目に、最初のテーブルに一意のカラムが、そのテーブルに現れた順序で
- 3 番目に、2 番目のテーブルに一意のカラムが、そのテーブルに現れた順序で

2 つの共通カラムを置き換える 1 つの結果カラムは、合体操作を使用して定義されます。つまり、t1.a と t2.a の 2 つに対して、結果として得られる 1 つの結合カラム a は a = COALESCE(t1.a, t2.a) として定義されます。ここでは:

```
COALESCE(x, y) = (CASE WHEN V1 IS NOT NULL THEN V1 ELSE V2 END)
```

この結合操作がほかのいずれかの結合である場合、その結合の結果カラムは、結合されたテーブルのすべてのカラムの連結で構成されます。これは以前と同じです。

合体したカラムの定義の結果として、外部結合では、2 つのカラムのいずれかが常に NULL である場合、合体したカラムには NULL 以外のカラムの値が含まれます。どちらのカラムも NULL でないか、または両方のカラムがこの値である場合、両方の共通カラムに同じ値が含まれているため、合体したカラムの値としてどちらが選択されるかは問題にはなりません。これを解釈するための簡単な方法として、外部結合の合体したカラムが JOIN の内部テーブルの共通カラムによって表されると考えてみます。テーブル t1(a,b) と t2(a,c) に次の内容が含まれているとします。

```
t1   t2
---- ----
1 x  2 z
2 y  3 w
```

このとき、次のようになります。

```
mysql> SELECT * FROM t1 NATURAL LEFT JOIN t2;
+------+------+------+
| a   | b   | c   |
+------+------+------+
|   1 | x   | NULL |
|   2 | y   | z   |
+------+------+------+
```

ここでは、カラム a に t1.a の値が含まれています。

```
mysql> SELECT * FROM t1 NATURAL RIGHT JOIN t2;
+------+------+------+
| a   | c   | b   |
+------+------+------+
|   2 | z   | y   |
|   3 | w   | NULL |
```

```
+------+------+------+
```

ここでは、カラム a に t2.a の値が含まれています。

これらの結果を、JOIN ... ON を使用した、それ以外では同等のクエリーと比較してください。

```
mysql> SELECT * FROM t1 LEFT JOIN t2 ON (t1.a = t2.a);
+------+------+------+------+
| a    | b    | a    | c    |
+------+------+------+------+
|    1 | x    | NULL | NULL |
|    2 | y    |    2 | z    |
+------+------+------+------+
```

```
mysql> SELECT * FROM t1 RIGHT JOIN t2 ON (t1.a = t2.a);
+------+------+------+------+
| a    | b    | a    | c    |
+------+------+------+------+
|    2 | y    |    2 | z    |
| NULL | NULL |    3 | w    |
+------+------+------+------+
```

- 以前は、USING 句を、対応するカラムを比較する ON 句として書き換えることができました。たとえば、次の 2 つの句は意味的に同一でした。

```
a LEFT JOIN b USING (c1,c2,c3)
a LEFT JOIN b ON a.c1=b.c1 AND a.c2=b.c2 AND a.c3=b.c3
```

現在、この 2 つの句はまったく同じではなくなっています。

- どの行が結合条件を満たすかの判定に関しては、どちらの結合も意味的に同一のままです。
- SELECT * の展開に対してどのカラムを表示するかの判定に関しては、この 2 つの結合は意味的に同一ではありません。USING 結合が対応するカラムの合体した値を選択するのに対して、ON 結合は、すべてのテーブルのすべてのカラムを選択します。前の USING 結合の場合、SELECT * は次の値を選択します。

  ```
  COALESCE(a.c1,b.c1), COALESCE(a.c2,b.c2), COALESCE(a.c3,b.c3)
  ```

  ON 結合の場合、SELECT * は次の値を選択します。

  ```
  a.c1, a.c2, a.c3, b.c1, b.c2, b.c3
  ```

  内部結合では、a.c1 と b.c1 の両方のカラムに同じ値が含まれるため、COALESCE(a.c1,b.c1) はどちらのカラムとも同じです。外部結合 (LEFT JOIN など) では、2 つのカラムのどちらかが NULL になる場合があります。そのカラムは結果から省略されます。

- 多方向自然結合の評価は、NATURAL または USING 結合の結果に影響を与え、さらにはクエリーの書き換えが必要になる場合もある非常に重要な点で異なります。3 つのテーブル t1(a,b)、t2(c,b)、および t3(a,c) があり、各テーブルに t1(1,2)、t2(10,2)、および t3(7,10) の 1 行が含まれているとします。また、これらの 3 つのテーブルに対して次の NATURAL JOIN を実行するとします。

```
SELECT ... FROM t1 NATURAL JOIN t2 NATURAL JOIN t3;
```

以前は、2 番目の結合の左のオペランドが t2 であると見なされたのに対して、現在はネストされた結合 (t1 NATURAL JOIN t2) であると見なされます。その結果、t3 のカラムは t2 でのみ共通カラムに対してチェックされ、さらに t3 に t1 との共通カラムが含まれている場合、これらのカラムは等価結合カラムとして使用されません。そのため、以前は、前のクエリーは次の等価結合に変換されました。

```
SELECT ... FROM t1, t2, t3
  WHERE t1.b = t2.b AND t2.c = t3.c;
```

その結合には、もう 1 つの等価結合述語 (t1.a = t3.a) がありません。その結果、本来生成すべき空の結果ではなく、1 行が生成されます。正しい同等のクエリーは次のとおりです。

```
SELECT ... FROM t1, t2, t3
  WHERE t1.b = t2.b AND t2.c = t3.c AND t1.a = t3.a;
```

現在のバージョンの MySQL で古いバージョンと同じクエリー結果が必要な場合は、自然結合を最初の等価結合として書き換えてください。

- 以前は、カンマ演算子 (,) と JOIN はどちらも同じ優先順位を持っていたため、結合式 t1, t2 JOIN t3 は ((t1, t2) JOIN t3) として解釈されました。現在は、JOIN の優先順位の方が高いため、この式は (t1, (t2 JOIN t3)) として解釈されます。ON 句は結合のオペランド内のカラムしか参照できず、また優先順位の変更によってそれらの

オペランドが示す内容の解釈が変更されるため、この変更はその句を使用するステートメントに影響を与えます。

例:

```
CREATE TABLE t1 (i1 INT, j1 INT);
CREATE TABLE t2 (i2 INT, j2 INT);
CREATE TABLE t3 (i3 INT, j3 INT);
INSERT INTO t1 VALUES(1,1);
INSERT INTO t2 VALUES(1,1);
INSERT INTO t3 VALUES(1,1);
SELECT * FROM t1, t2 JOIN t3 ON (t1.i1 = t3.i3);
```

以前は、この SELECT は、(t1,t2) として t1,t2 の暗黙的なグループ化のために正当でした。現在は、JOIN が優先されるため、ON 句のオペランドは t2 と t3 になります。t1.i1 はどのオペランドのカラムでもないため、その結果は「カラム 't1.i1' は 'on clause' にはありません」というエラーになります。この結合の処理を可能にするには、ON 句のオペランドが (t1,t2) と t3 になるように、最初の 2 つのテーブルを明示的に括弧でグループ化します。

```
SELECT * FROM (t1, t2) JOIN t3 ON (t1.i1 = t3.i3);
```

あるいは、カンマ演算子の使用を避け、代わりに JOIN を使用します。

```
SELECT * FROM t1 JOIN t2 JOIN t3 ON (t1.i1 = t3.i3);
```

この変更はまた、カンマ演算子を INNER JOIN、CROSS JOIN、LEFT JOIN、および RIGHT JOIN (これらはすべて現在、カンマ演算子より高い優先順位を持っています) と混在させているステートメントにも適用されます。

- 以前は、ON 句は、その右側で指定されているテーブル内のカラムを参照することができました。現在は、ON 句は自身のオペランドしか参照できません。

  例:

  ```
  CREATE TABLE t1 (i1 INT);
  CREATE TABLE t2 (i2 INT);
  CREATE TABLE t3 (i3 INT);
  SELECT * FROM t1 JOIN t2 ON (i1 = i3) JOIN t3;
  ```

  以前は、この SELECT ステートメントは正当でした。現在は、i3 が ON 句のオペランドではない t3 内のカラムであるため、このステートメントは「カラム 'i3' は 'on clause' にはありません」というエラーで失敗します。このステートメントを次のように書き換えるようにしてください。

  ```
  SELECT * FROM t1 JOIN t2 JOIN t3 ON (i1 = i3);
  ```

- NATURAL または USING 結合でのカラム名の解決が以前とは異なります。FROM 句の外部にあるカラム名の場合、MySQL は現在、以前と比較してクエリーのスーパーセットを処理します。つまり、MySQL が以前、一部のカラムがあいまいであるというエラーを発行したケースでも、そのクエリーは現在、正しく処理されます。これは、MySQL が現在、NATURAL または USING 結合の共通カラムを単一カラムとして処理するため、クエリーがこのようなカラムを参照しても、クエリーコンパイラがそのカラムをあいまいであるとは見なさないことによります。

  例:

  ```
  SELECT * FROM t1 NATURAL JOIN t2 WHERE b > 1;
  ```

  以前は、このクエリーによってエラー ERROR 1052 (23000): Column 'b' in where clause is ambiguous が生成されました。現在は、このクエリーによって正しい結果が生成されます。

  ```
  +------+------+------+
  | b    | c    | y    |
  +------+------+------+
  |    4 |    2 |    3 |
  +------+------+------+
  ```

  SQL:2003 標準と比べた場合の MySQL の 1 つの拡張として、MySQL では、NATURAL または USING 結合の共通 (合体した) カラムを (以前と同様に) 修飾できるのに対して、標準ではそれが禁止される点があります。

13.2.9.3 インデックスヒントの構文

クエリー処理中にインデックスを選択する方法に関する情報をオプティマイザに提供するためのヒントを指定できます。セクション13.2.9.2「JOIN 構文」では、SELECT ステートメントでテーブルを指定するための一般的

な構文について説明しています。個々のテーブルの構文 (インデックスヒントの構文を含む) は次のようになります。

```
tbl_name [[AS] alias] [index_hint_list]

index_hint_list:
    index_hint [, index_hint] ...

index_hint:
    USE {INDEX|KEY}
      [FOR {JOIN|ORDER BY|GROUP BY}] ([index_list])
  | IGNORE {INDEX|KEY}
      [FOR {JOIN|ORDER BY|GROUP BY}] (index_list)
  | FORCE {INDEX|KEY}
      [FOR {JOIN|ORDER BY|GROUP BY}] (index_list)

index_list:
    index_name [, index_name] ...
```

USE INDEX (index_list) を指定することによって、テーブル内の行を検索するために、指定されたインデックスの 1 つのみを使用するよう MySQL に指示できます。代わりの構文 IGNORE INDEX (index_list) を使用すると、いくつかの特定の (1 つまたは複数の) インデックスを使用しないよう MySQL に指示できます。これらのヒントは、EXPLAIN によって、MySQL が可能性のあるインデックスのリストから間違ったインデックスを使用していることが示された場合に役立ちます。

また、USE INDEX (index_list) と同様の機能を持つが、テーブルスキャンが非常に負荷が大きいと見なされる点が追加された FORCE INDEX を使用することもできます。つまり、テーブルスキャンは、指定されたインデックスのいずれかを使用してテーブル内の行を検索する方法がない場合にのみ使用されます。

各ヒントには、カラムの名前ではなく、インデックスの名前が必要です。PRIMARY KEY の名前は PRIMARY です。テーブルのインデックス名を表示するには、SHOW INDEX を使用します。

index_name 値は、完全なインデックス名である必要はありません。インデックス名のあいまいでないプリフィクスにすることができます。プリフィクスがあいまいな場合は、エラーが発生します。

例:

```
SELECT * FROM table1 USE INDEX (col1_index,col2_index)
  WHERE col1=1 AND col2=2 AND col3=3;

SELECT * FROM table1 IGNORE INDEX (col3_index)
  WHERE col1=1 AND col2=2 AND col3=3;
```

インデックスヒントの構文には、次の特性があります。

- USE INDEX に空の index_list を指定する (つまり、「インデックスを使用しない」) ことは構文として有効です。FORCE INDEX または IGNORE INDEX に空の index_list を指定することは構文エラーです。
- ヒントに FOR 句を追加することによって、インデックスヒントのスコープを指定できます。これにより、クエリー処理のさまざまなフェーズに対するオプティマイザの実行計画の選択をよりきめ細かく制御できるようになります。MySQL がテーブル内の行の検索方法および結合の処理方法を決定するときに使用されるインデックスにのみ影響を与えるには、FOR JOIN を使用します。行をソートまたはグループ化するためのインデックス使用に影響を与えるには、FOR ORDER BY または FOR GROUP BY を使用します。(ただし、テーブルを範囲に含むインデックスが存在し、それがテーブルへのアクセスに使用されている場合、オプティマイザは、そのインデックスを無効にする IGNORE INDEX FOR {ORDER BY|GROUP BY} ヒントを無視します。)
- 複数のインデックスヒントを指定できます。

  ```
  SELECT * FROM t1 USE INDEX (i1) IGNORE INDEX FOR ORDER BY (i2) ORDER BY a;
  ```

  複数のヒントで同じインデックスを指定することは (同じヒント内であっても) エラーではありません。

  ```
  SELECT * FROM t1 USE INDEX (i1) USE INDEX (i1,i1);
  ```

  ただし、同じテーブルに対して USE INDEX と FORCE INDEX を混在させると、エラーが発生します。

  ```
  SELECT * FROM t1 USE INDEX FOR JOIN (i1) FORCE INDEX FOR JOIN (i2);
  ```

インデックスヒントで FOR 句を指定しない場合、デフォルトでは、そのヒントはステートメントのすべての部分に適用されます。たとえば、次のヒント

```
IGNORE INDEX (i1)
```

は次のヒントの組み合わせと同等です。

```
IGNORE INDEX FOR JOIN (i1)
IGNORE INDEX FOR ORDER BY (i1)
IGNORE INDEX FOR GROUP BY (i1)
```

サーバーで、FOR 句が存在しないときに (ヒントが行の取得にのみ適用されるように) ヒントスコープに対する古い動作が使用されるようにするには、サーバーの起動時に古いシステム変数を有効にします。レプリケーションセットアップでこの変数を有効にする場合は注意してください。ステートメントベースのバイナリロギングで、マスターとスレーブに異なるモードを指定するとレプリケーションエラーが発生する場合があります。

インデックスヒントが処理されるとき、これらのインデックスヒントは、型 (USE、FORCE、IGNORE) およびスコープ (FOR JOIN、FOR ORDER BY、FOR GROUP BY) ごとに 1 つのリストに収集されます。例:

```
SELECT * FROM t1
  USE INDEX () IGNORE INDEX (i2) USE INDEX (i1) USE INDEX (i2);
```

次と同等です。

```
SELECT * FROM t1
  USE INDEX (i1,i2) IGNORE INDEX (i2);
```

そのあと、インデックスヒントは、スコープごとに次の順序で適用されます。

1. {USE|FORCE} INDEX が存在する場合は、これが適用されます。(存在しない場合は、オプティマイザによって決定されたインデックスのセットが使用されます。)
2. 前の手順の結果に対して、IGNORE INDEX が適用されます。たとえば、次の 2 つのクエリーは同等です。

   ```
   SELECT * FROM t1 USE INDEX (i1) IGNORE INDEX (i2) USE INDEX (i2);

   SELECT * FROM t1 USE INDEX (i1);
   ```

FULLTEXT の検索の場合、インデックスヒントは次のように機能します。

- 自然言語モードの検索の場合、インデックスヒントは暗黙のうちに無視されます。たとえば、IGNORE INDEX(i) は警告なしで無視され、インデックスが引き続き使用されます。

  ブールモードの検索の場合、FOR ORDER BY または FOR GROUP BY を含むインデックスヒントは暗黙のうちに無視されます。FOR JOIN を含むインデックスヒント、または FOR 修飾子を含まないインデックスヒントは受け付けられます。ヒントが FULLTEXT 以外の検索に適用される場合とは異なり、このヒントは、クエリー実行のすべてのフェーズ (行の検索と取得、グループ化、および順序付け) に使用されます。これは、ヒントが FULLTEXT 以外のインデックスに対して指定されている場合でも当てはまります。

たとえば、次の 2 つのクエリーは同等です。

```
SELECT * FROM t
  USE INDEX (index1)
  IGNORE INDEX (index1) FOR ORDER BY
  IGNORE INDEX (index1) FOR GROUP BY
  WHERE ... IN BOOLEAN MODE ... ;

SELECT * FROM t
  USE INDEX (index1)
  WHERE ... IN BOOLEAN MODE ... ;
```

13.2.9.4 UNION 構文

```
SELECT ...
UNION [ALL | DISTINCT] SELECT ...
[UNION [ALL | DISTINCT] SELECT ...]
```

UNION は、複数の SELECT ステートメントからの結果を 1 つの結果セットに結合するために使用されます。

最初の SELECT ステートメントからのカラム名が、返される結果のカラム名として使用されます。各 SELECT ステートメントの対応する位置にリストされている選択されるカラムは、データ型が同じになるようにしてください。(たとえば、最初のステートメントによって選択される最初のカラムが、ほかのステートメントによって選択される最初のカラムと型が同じになるようにしてください。)

対応する SELECT カラムのデータ型が一致しない場合、UNION の結果内のカラムの型と長さは、すべての SELECT ステートメントによって取得された値を考慮に入れて決定されます。たとえば、次の例を考えてみます。

```
mysql> SELECT REPEAT('a',1) UNION SELECT REPEAT('b',10);
+---------------+
| REPEAT('a',1) |
+---------------+
| a             |
| bbbbbbbbbb    |
+---------------+
```

これらの SELECT ステートメントは通常の選択ステートメントですが、次の制限があります。

- INTO OUTFILE を使用できるのは、最後の SELECT ステートメントだけです。(ただし、UNION の結果全体がファイルに書き込まれます。)
- HIGH_PRIORITY を、UNION の一部である SELECT ステートメントで使用することはできません。それを最初の SELECT に対して指定しても、何の効果もありません。それを以降のいずれかの SELECT ステートメントに対して指定すると、構文エラーが発生します。

UNION のデフォルトの動作では、重複した行が結果から削除されます。オプションの DISTINCT キーワードは、これも重複した行の削除を指定するため、デフォルト以外の効果は何もありません。オプションの ALL キーワードを指定すると、重複した行の削除は実行されず、その結果には、すべての SELECT ステートメントからの一致するすべての行が含まれます。

UNION ALL と UNION DISTINCT を同じクエリー内で混在させることができます。混在した UNION 型は、DISTINCT 和集合がその左側にある ALL 和集合をすべてオーバーライドするように処理されます。DISTINCT 和集合は、UNION DISTINCT を使用して明示的に、あるいはそのあとに DISTINCT または ALL キーワードのない UNION を使用して暗黙的に生成できます。

個々の SELECT に ORDER BY または LIMIT を適用するには、この句を SELECT を囲む括弧内に配置します。

```
(SELECT a FROM t1 WHERE a=10 AND B=1 ORDER BY a LIMIT 10)
UNION
(SELECT a FROM t2 WHERE a=11 AND B=2 ORDER BY a LIMIT 10);
```

ただし、個々の SELECT ステートメントに対して ORDER BY を使用しても、UNION がデフォルトでは、順序付けされていない行のセットを生成するため、最終的な結果に行が現れる順序には何も影響を与えません。そのため、このコンテキストでは通常、ORDER BY は LIMIT と組み合わせて使用されます。それにより、選択された行の UNION の最終結果での順序に必ずしも影響を与えるわけではないにもかかわらず、SELECT で取得するためのこれらの行のサブセットを決定するために使用されるようになります。ORDER BY が SELECT 内に LIMIT なしで現れた場合、この句はいずれにしても何も効果がないため、最適化によって削除されます。

ORDER BY または LIMIT 句を使用して UNION の結果全体をソートまたは制限するには、個々の SELECT ステートメントを括弧で囲み、最後のステートメントのあとに ORDER BY または LIMIT を配置します。次の例では、この両方の句を使用しています。

```
(SELECT a FROM t1 WHERE a=10 AND B=1)
UNION
(SELECT a FROM t2 WHERE a=11 AND B=2)
ORDER BY a LIMIT 10;
```

括弧のないステートメントは、今示した括弧で囲まれたステートメントと同等です。

この種の ORDER BY は、テーブル名 (つまり、tbl_name.col_name という形式の名前) を含むカラム参照を使用できません。代わりに、最初の SELECT ステートメント内にカラムのエイリアスを指定し、そのエイリアスを ORDER BY 内で参照します。(あるいは、ORDER BY 内でカラムを、そのカラム位置を使用して参照します。ただし、カラム位置の使用は非推奨です。)

また、ソートされるカラムにエイリアスが指定されている場合、ORDER BY 句はそのカラム名ではなく、エイリアスを参照する必要があります。次のうちの最初のステートメントは機能しますが、2 番目は「カラム 'a' は 'order clause' にはありません」という エラーで失敗します。

```
(SELECT a AS b FROM t) UNION (SELECT ...) ORDER BY b;
(SELECT a AS b FROM t) UNION (SELECT ...) ORDER BY a;
```

UNION の結果内の行が、各 SELECT によって 1 つずつ取得された行のセットで構成されるようにするには、ソートカラムとして使用する各 SELECT 内の追加のカラムを選択し、最後の SELECT のあとに ORDER BY を追加します。

```
(SELECT 1 AS sort_col, col1a, col1b, ... FROM t1)
UNION
```

```
(SELECT 2, col2a, col2b, ... FROM t2) ORDER BY sort_col;
```

さらに個々の SELECT の結果内のソート順序を維持するには、ORDER BY 句にセカンダリカラムを追加します。

```
(SELECT 1 AS sort_col, col1a, col1b, ... FROM t1)
UNION
(SELECT 2, col2a, col2b, ... FROM t2) ORDER BY sort_col, col1a;
```

また、追加のカラムを使用すると、各行がどの SELECT から取得されるかを決定することもできます。追加のカラムでは、テーブル名を示す文字列などのほかの識別情報も指定できます。

13.2.10 サブクエリー構文

サブクエリーは、別のステートメント内の SELECT ステートメントです。

MySQL 4.1 から、SQL 標準に必要なサブクエリーのすべての形式および操作だけでなく、MySQL 固有のいくつかの機能がサポートされています。

サブクエリーの例を次に示します。

```
SELECT * FROM t1 WHERE column1 = (SELECT column1 FROM t2);
```

この例では、SELECT * FROM t1 ... が外部クエリー (または 外部ステートメント) であり、(SELECT column1 FROM t2) がサブクエリーです。これを、このサブクエリーは外部クエリー内でネストされていると表現し、また実際、サブクエリーをほかのサブクエリー内で (かなりの深さまで) ネストできます。サブクエリーは常に、括弧内に指定する必要があります。

サブクエリーの主な利点は次のとおりです。

- ステートメントの各部分を分離できるように、構造化されたクエリーを可能にします。
- 通常であれば複雑な結合や和集合を必要とする操作を実行するための代替手段を提供します。
- 多くの人びとが、サブクエリーを複雑な結合や和集合より読みやすいと感じています。実際、早期の SQL である「構造化クエリー言語」を呼び出すという元の考え方を人びとに提供したのは、サブクエリーの技術革新でした。

SQL 標準で指定され、MySQL でサポートされているサブクエリー構文に関する主なポイントを示すステートメントの例を次に示します。

```
DELETE FROM t1
WHERE s11 > ANY
 (SELECT COUNT(*) /* no hint */ FROM t2
  WHERE NOT EXISTS
   (SELECT * FROM t3
    WHERE ROW(5*t2.s1,77)=
     (SELECT 50,11*s1 FROM t4 UNION SELECT 50,77 FROM
      (SELECT * FROM t5) AS t5)));
```

サブクエリーは、スカラー (単一値)、単一行、単一カラム、またはテーブル (1 つ以上のカラムの 1 つ以上の行) を返すことができます。これらは、スカラー、カラム、行、およびテーブルサブクエリーと呼ばれます。特定の種類の結果を返すサブクエリーは多くの場合、次の各セクションで説明されているように、特定のコンテキストでのみ使用できます。

サブクエリーを使用できるステートメントのタイプに関する制限はほとんどありません。サブクエリーには、DISTINCT、GROUP BY、ORDER BY、LIMIT、結合、インデックスヒント、UNION 構造構文、コメント、関数などの、通常の SELECT に含めることのできる多くのキーワードや句を含めることができます。

サブクエリーの外部ステートメントは、SELECT、INSERT、UPDATE、DELETE、SET、DO のいずれでもかまいません。

MySQL では、テーブルを変更し、さらにサブクエリーで同じテーブルから選択することはできません。これは、DELETE、INSERT、REPLACE、UPDATE、LOAD DATA INFILE (サブクエリーは SET 句で使用できるため) などのステートメントに適用されます。

オプティマイザによるサブクエリーの処理方法については、セクション8.2.1.18「サブクエリーの最適化」を参照してください。サブクエリーの使用に関する制限の説明 (特定の形式のサブクエリー構文でのパフォーマンスの問題を含む) については、セクションD.4「サブクエリーの制約」を参照してください。

13.2.10.1 スカラーオペランドとしてのサブクエリー

もっとも単純な形式のサブクエリーは、単一値を返すスカラーサブクエリーです。スカラーサブクエリーは単純なオペランドであるため、単一カラム値またはリテラルが正当である場所であればほぼどこでも使用できるほか、データ型、長さ、NULL にできることの表示などの、すべてのオペランドが持っている特性を持つことを期待できます。例:

```
CREATE TABLE t1 (s1 INT, s2 CHAR(5) NOT NULL);
INSERT INTO t1 VALUES(100, 'abcde');
SELECT (SELECT s2 FROM t1);
```

この SELECT 内のサブクエリーは、CHAR のデータ型、5 の長さ、CREATE TABLE の時点で有効なデフォルトに等しい文字セットと照合順序、およびこのカラム内の値を NULL にできることの表示を持つ単一値 ('abcde') を返します。サブクエリー結果が空であればその結果は NULL になるため、スカラーサブクエリーによって選択された値の NULL 可能性はコピーされません。今示したサブクエリーで t1 が空であった場合は、s2 が NOT NULL であるにもかかわらず、その結果は NULL になります。

スカラーサブクエリーを使用できないコンテキストがいくつか存在します。ステートメントでリテラル値のみが許可されている場合は、サブクエリーを使用できません。たとえば、LIMIT にはリテラル整数の引数が必要であり、LOAD DATA INFILE にはリテラル文字列のファイル名が必要です。サブクエリーを使用してこれらの値を指定することはできません。

次の各セクションにある、やや簡素な構造構文 (SELECT column1 FROM t1) が含まれた例を参照するときは、はるかに多様で、かつ複雑な構造構文を含む独自のコードがあるものと考えてください。

次の 2 つのテーブルを作成するとします。

```
CREATE TABLE t1 (s1 INT);
INSERT INTO t1 VALUES (1);
CREATE TABLE t2 (s1 INT);
INSERT INTO t2 VALUES (2);
```

次に、SELECT を実行します。

```
SELECT (SELECT s1 FROM t2) FROM t1;
```

t2 には、2 の値を持つカラム s1 が含まれている行が存在するため、その結果は 2 になります。

スカラーサブクエリーを式の一部にすることはできますが、そのサブクエリーが関数への引数を提供するオペランドである場合でも、括弧を忘れないでください。例:

```
SELECT UPPER((SELECT s1 FROM t1)) FROM t2;
```

13.2.10.2 サブクエリーを使用した比較

サブクエリーのもっとも一般的な使用の形式は次のとおりです。

```
non_subquery_operand comparison_operator (subquery)
```

ここで、comparison_operator は次の演算子のいずれかです。

```
= > < >= <= <> != <=>
```

例:

```
... WHERE 'a' = (SELECT column1 FROM t1)
```

MySQL では、次の構造構文も許可されます。

```
non_subquery_operand LIKE (subquery)
```

以前は、サブクエリーの唯一の正当な場所は比較の右側であり、この方法にこだわったいくつかの古い DBMS がまだ見つかることもあります。

結合では実行できない一般的な形式のサブクエリー比較の例を次に示します。これは、column1 値がテーブル t2 内の最大値に等しいテーブル t1 内のすべての行を検索します。

```
SELECT * FROM t1
  WHERE column1 = (SELECT MAX(column2) FROM t2);
```

次に別の例を示します。これもまた、いずれかのテーブルに対する集約が含まれているため、結合では実行できません。これは、特定のカラムに 2 回現れる値を含むテーブル t1 内のすべての行を検索します。

```
SELECT * FROM t1 AS t
 WHERE 2 = (SELECT COUNT(*) FROM t1 WHERE t1.id = t.id);
```

スカラーに対するサブクエリーの比較の場合、サブクエリーはスカラーを返す必要があります。行コンストラクタに対するサブクエリーの比較の場合、サブクエリーは、その行コンストラクタと同じ数の値を含む行を返す行サブクエリーである必要があります。セクション13.2.10.5「行サブクエリー」を参照してください。

13.2.10.3 ANY、IN、または SOME を使用したサブクエリー

構文:

```
operand comparison_operator ANY (subquery)
operand IN (subquery)
operand comparison_operator SOME (subquery)
```

ここで、comparison_operator は次の演算子のいずれかです。

```
= > < >= <= <> !=
```

ANY キーワード (これは比較演算子のあとに指定する必要があります) は、「このサブクエリーが返すカラム内の値の ANY (いずれか) に対して比較が TRUE である場合は TRUE を返す」ことを示します。例:

```
SELECT s1 FROM t1 WHERE s1 > ANY (SELECT s1 FROM t2);
```

テーブル t1 内に (10) を含む行が存在するとします。テーブル t2 に (21,14,7) が含まれている場合、t2 には 10 より小さい値 7 が存在するため、この式は TRUE です。テーブル t2 に (20,10) が含まれている場合、またはテーブル t2 が空である場合、この式は FALSE です。テーブル t2 に (NULL,NULL,NULL) が含まれている場合、この式は不明 (つまり、NULL) です。

サブクエリーで使用されている場合、ワード IN は = ANY のエイリアスです。そのため、次の 2 つのステートメントは同じです。

```
SELECT s1 FROM t1 WHERE s1 = ANY (SELECT s1 FROM t2);
SELECT s1 FROM t1 WHERE s1 IN    (SELECT s1 FROM t2);
```

式リストで使用されている場合、IN と = ANY はシノニムではありません。IN は式リストを取得できますが、= ANY はできません。セクション12.3.2「比較関数と演算子」を参照してください。

NOT IN は <> ANY ではなく、<> ALL のエイリアスです。セクション13.2.10.4「ALL を使用したサブクエリー」を参照してください。

ワード SOME は ANY のエイリアスです。そのため、次の 2 つのステートメントは同じです。

```
SELECT s1 FROM t1 WHERE s1 <> ANY  (SELECT s1 FROM t2);
SELECT s1 FROM t1 WHERE s1 <> SOME (SELECT s1 FROM t2);
```

ワード SOME はほとんど使用されませんが、この例は、これがなぜ役立つ可能性があるかを示しています。ほとんどの人びとにとって、「a is not equal to any b」(a はどの b にも等しくない) という英語のフレーズは「there is no b which is equal to a」(a に等しい b は存在しない) を示しますが、それはこの SQL 構文が示す内容とは異なります。この構文は、「there is some b to which a is not equal」(a に等しくない b がいくつか存在する) を示します。代わりに <> SOME を使用すると、このクエリーの本当の意味がすべての人に理解されるようにするのに役立ちます。

13.2.10.4 ALL を使用したサブクエリー

構文:

```
operand comparison_operator ALL (subquery)
```

ワード ALL (これは比較演算子のあとに指定する必要があります) は、「このサブクエリーが返すカラム内の値の ALL (すべて) に対して比較が TRUE である場合は TRUE を返す」ことを示します。例:

```
SELECT s1 FROM t1 WHERE s1 > ALL (SELECT s1 FROM t2);
```

テーブル t1 内に (10) を含む行が存在するとします。テーブル t2 に (-5,0,+5) が含まれている場合、10 が t2 内の 3 つのすべての値より大きいため、この式は TRUE です。テーブル t2 に (12,6,NULL,-100) が含まれている場合、

テーブル t2 には 10 より大きい単一値 12 が存在するため、この式は FALSE です。テーブル t2 に (0,NULL,1) が含まれている場合、この式は不明 (つまり、NULL) です。

最後に、テーブル t2 が空である場合、この式は TRUE です。そのため、テーブル t2 が空であるとき、次の式は TRUE です。

```
SELECT * FROM t1 WHERE 1 > ALL (SELECT s1 FROM t2);
```

ただし、テーブル t2 が空であるとき、次の式は NULL です。

```
SELECT * FROM t1 WHERE 1 > (SELECT s1 FROM t2);
```

さらに、テーブル t2 が空であるとき、次の式は NULL です。

```
SELECT * FROM t1 WHERE 1 > ALL (SELECT MAX(s1) FROM t2);
```

一般に、NULL 値を含むテーブルと空のテーブルは「エッジケース」です。サブクエリーを記述するときは、常に、これらの 2 つの可能性を考慮に入れたかどうかを考慮してください。

NOT IN は <> ALL のエイリアスです。そのため、次の 2 つのステートメントは同じです。

```
SELECT s1 FROM t1 WHERE s1 <> ALL (SELECT s1 FROM t2);
SELECT s1 FROM t1 WHERE s1 NOT IN (SELECT s1 FROM t2);
```

13.2.10.5 行サブクエリー

ここまでの説明は、スカラーまたはカラムサブクエリー、つまり、単一値または値のカラムを返すサブクエリーについてでした。行サブクエリーは、単一行を返し、そのために複数のカラム値を返すことができるサブクエリーバリアントです。行サブクエリーの比較のための正当な演算子は次のとおりです。

```
= > < >= <= <> != <=>
```

次に、2 つの例を示します。

```
SELECT * FROM t1
  WHERE (col1,col2) = (SELECT col3, col4 FROM t2 WHERE id = 10);
SELECT * FROM t1
  WHERE ROW(col1,col2) = (SELECT col3, col4 FROM t2 WHERE id = 10);
```

どちらのクエリーでも、テーブル t2 に id = 10 を持つ単一行が含まれている場合、このサブクエリーは単一行を返します。この行に t1 内のいずれかの行の col1 および col2 値に等しい col3 および col4 値が含まれている場合、WHERE 式は TRUE であり、各クエリーはこれらの t1 行を返します。t2 行の col3 および col4 値が、いずれの t1 行の col1 および col2 値にも等しくない場合、この式は FALSE であり、このクエリーは空の結果セットを返します。サブクエリーによって行が生成されない場合、この式は不明 (つまり、NULL) です。サブクエリーによって複数の行が生成される場合は、行サブクエリーが最大で 1 行しか返すことができないため、エラーが発生します。

式 (1,2) や ROW(1,2) は、行コンストラクタとも呼ばれます。この 2 つは同等です。行コンストラクタと、サブクエリーによって返される行には、同じ数の値が含まれている必要があります。

行コンストラクタは、2 つ以上のカラムを返すサブクエリーとの比較に使用されます。サブクエリーが単一カラムを返すと、これは行ではなく、スカラー値として見なされるため、少なくとも 2 つのカラムを返さないサブクエリーで行コンストラクタを使用することはできません。そのため、次のクエリーは構文エラーで失敗します。

```
SELECT * FROM t1 WHERE ROW(1) = (SELECT column1 FROM t2)
```

行コンストラクタは、ほかのコンテキストでも正当です。たとえば、次の 2 つのステートメントは意味的に同等です (また、オプティマイザによって同じように処理されます)。

```
SELECT * FROM t1 WHERE (column1,column2) = (1,1);
SELECT * FROM t1 WHERE column1 = 1 AND column2 = 1;
```

次のクエリーは、「テーブル t2 内にも存在するテーブル t1 内のすべての行を検索する」という要求にこたえます。

```
SELECT column1,column2,column3
 FROM t1
 WHERE (column1,column2,column3) IN
       (SELECT column1,column2,column3 FROM t2);
```

13.2.10.6 EXISTS または NOT EXISTS を使用したサブクエリー

サブクエリーが少なくとも 1 行を返す場合、EXISTS subquery は TRUE であり、NOT EXISTS subquery は FALSE です。例:

```
SELECT column1 FROM t1 WHERE EXISTS (SELECT * FROM t2);
```

従来より、EXISTS サブクエリーは SELECT * で始まりますが、SELECT 5 や SELECT column1、あるいはほかの何で始まってもかまいません。MySQL はこのようなサブクエリー内の SELECT リストを無視するため、何も違いは生まれません。

前の例では、t2 に何らかの行が含まれている場合 (NULL 値以外は何も含まれていない行でも)、EXISTS 条件は TRUE です。[NOT] EXISTS サブクエリーには、ほぼ常に相互関係が含まれるため、これは実際にはありそうにもない例です。次に、より現実的な例をいくつか示します。

- 1 つ以上の市に存在するのはどのような種類のお店ですか?

```
SELECT DISTINCT store_type FROM stores
  WHERE EXISTS (SELECT * FROM cities_stores
              WHERE cities_stores.store_type = stores.store_type);
```

- どの市にも存在しないのはどのような種類のお店ですか?

```
SELECT DISTINCT store_type FROM stores
  WHERE NOT EXISTS (SELECT * FROM cities_stores
                WHERE cities_stores.store_type = stores.store_type);
```

- すべての市に存在するのはどのような種類のお店ですか?

```
SELECT DISTINCT store_type FROM stores s1
  WHERE NOT EXISTS (
    SELECT * FROM cities WHERE NOT EXISTS (
      SELECT * FROM cities_stores
       WHERE cities_stores.city = cities.city
       AND cities_stores.store_type = stores.store_type));
```

最後の例は、二重にネストされた NOT EXISTS クエリーです。つまり、NOT EXISTS 句の中に NOT EXISTS 句が存在します。これは正式には、「Stores にないお店が含まれている市は存在しますか」? という質問に答えます。ただし、ネストされた NOT EXISTS が、「x はすべての y に対して TRUE ですか?」という質問に答えるという方が簡単です。

13.2.10.7 相関サブクエリー

相関サブクエリーは、外部クエリーにも現れるテーブルへの参照を含むサブクエリーです。例:

```
SELECT * FROM t1
  WHERE column1 = ANY (SELECT column1 FROM t2
                WHERE t2.column2 = t1.column2);
```

このサブクエリーには、サブクエリーの FROM 句でテーブル t1 が指定されていない場合でも、t1 のカラムへの参照が含まれます。そのため、MySQL はこのサブクエリーの外部を探し、外部クエリー内の t1 を見つけます。

テーブル t1 に column1 = 5 かつ column2 = 6 である行が含まれている一方、テーブル t2 に column1 = 5 かつ column2 = 7 である行が含まれているとします。単純な式 ... WHERE column1 = ANY (SELECT column1 FROM t2) は TRUE になりますが、この例では、サブクエリー内の WHERE 句は ((5,6) が (5,7) に等しくないため) FALSE です。そのため、全体としての式は FALSE です。

スコープルール: MySQL は、内部から外部に評価します。例:

```
SELECT column1 FROM t1 AS x
  WHERE x.column1 = (SELECT column1 FROM t2 AS x
    WHERE x.column1 = (SELECT column1 FROM t3
      WHERE x.column2 = t3.column1));
```

このステートメントでは、SELECT column1 FROM t2 AS x ... が t2 の名前を変更するため、x.column2 はテーブル t2 内のカラムである必要があります。SELECT column1 FROM t1 ... がさらに外部にある外部クエリーであるため、これはテーブル t1 内のカラムではありません。

HAVING または ORDER BY 句内のサブクエリーの場合、MySQL は外部選択リスト内でもカラム名を探します。

特定のケースでは、相関サブクエリーが最適化されます。例:

```
val IN (SELECT key_val FROM tbl_name WHERE correlated_condition)
```

そうしないと、これらは非効率的であり、遅くなる可能性があります。クエリーを結合として書き換えると、パフォーマンスが向上することがあります。

相関サブクエリー内の集約関数には、その関数に外部参照以外は何も含まれておらず、かつその関数が別の関数または式に含まれていない場合、外部参照を含めることができます。

13.2.10.8 FROM 句内のサブクエリー

サブクエリーは、SELECT ステートメントの FROM 句内で正当です。その実際の構文は次のとおりです。

```
SELECT ... FROM (subquery) [AS] name ...
```

FROM 句内のどのテーブルも名前を持っている必要があるため、[AS] name 句は必須です。subquery 選択リスト内にカラムが存在する場合は、それが一意の名前を持っている必要があります。

説明のために、次のテーブルがあるとします。

```
CREATE TABLE t1 (s1 INT, s2 CHAR(5), s3 FLOAT);
```

このテーブルの例を使用して、FROM 句内のサブクエリーを使用する方法を次に示します。

```
INSERT INTO t1 VALUES (1,'1',1.0);
INSERT INTO t1 VALUES (2,'2',2.0);
SELECT sb1,sb2,sb3
  FROM (SELECT s1 AS sb1, s2 AS sb2, s3*2 AS sb3 FROM t1) AS sb
  WHERE sb1 > 1;
```

結果: 2, '2', 4.0。

次に別の例を示します。グループ化されたテーブルに関する一連の合計の平均を知りたいとします。次のクエリーは機能しません。

```
SELECT AVG(SUM(column1)) FROM t1 GROUP BY column1;
```

ただし、次のクエリーは目的の情報を提供します。

```
SELECT AVG(sum_column1)
  FROM (SELECT SUM(column1) AS sum_column1
        FROM t1 GROUP BY column1) AS t1;
```

サブクエリー内で使用されているカラム名 (sum_column1) は外部クエリーで認識されます。

FROM 句内のサブクエリーは、スカラー、カラム、行、またはテーブルを返すことができます。FROM 句内のサブクエリーは、JOIN 操作の ON 句内で使用されないかぎり、相関サブクエリーにすることはできません。

MySQL 5.6.3 より前は、FROM 句内のサブクエリーは EXPLAIN ステートメントに対しても実行されます (つまり、派生した一時テーブルが実体化されます)。これは、最適化フェーズ中に上位レベルのクエリーにすべてのテーブルに関する情報が必要であり、かつサブクエリーが実行されないかぎり FROM 句内のサブクエリーによって表されているテーブルを使用できないために発生します。MySQL 5.6.3 の時点では、オプティマイザは派生テーブルに関する情報を別の方法で特定するため、それらの実体化が EXPLAIN に対して発生しません。FROM 句内のサブクエリー (派生テーブル) の最適化を参照してください。

特定の状況では、EXPLAIN SELECT を使用してテーブルデータを変更できます。これは、外部クエリーがいずれかのテーブルにアクセスし、内部クエリーが、テーブルの 1 つ以上の行を変更するストアドファンクションを呼び出す場合に発生する可能性があります。データベース d1 内に、次に示すように作成された 2 つのテーブル t1 と t2 があるとします。

```
mysql> CREATE DATABASE d1;
Query OK, 1 row affected (0.00 sec)

mysql> USE d1;
Database changed

mysql> CREATE TABLE t1 (c1 INT);
Query OK, 0 rows affected (0.15 sec)

mysql> CREATE TABLE t2 (c1 INT);
Query OK, 0 rows affected (0.08 sec)
```

ここで、t2 を変更するストアドファンクション f1 を作成します。

```
mysql> DELIMITER //
mysql> CREATE FUNCTION f1(p1 INT) RETURNS INT
mysql>   BEGIN
```

```
mysql>    INSERT INTO t2 VALUES (p1);
mysql>    RETURN p1;
mysql>  END //
Query OK, 0 rows affected (0.01 sec)

mysql> DELIMITER ;
```

次に示すように、EXPLAIN SELECT でこの関数を直接参照しても、t2 には何も影響を与えません。

```
mysql> SELECT * FROM t2;
Empty set (0.00 sec)

mysql> EXPLAIN SELECT f1(5);
+----+-------------+-------+------+---------------+------+---------+------+------+----------------+
| id | select_type | table | type | possible_keys | key  | key_len | ref  | rows | Extra          |
+----+-------------+-------+------+---------------+------+---------+------+------+----------------+
|  1 | SIMPLE      | NULL  | NULL | NULL          | NULL | NULL    | NULL | NULL | No tables used |
+----+-------------+-------+------+---------------+------+---------+------+------+----------------+
1 row in set (0.00 sec)

mysql> SELECT * FROM t2;
Empty set (0.00 sec)
```

これは、出力の table および Extra カラムでわかるように、SELECT ステートメントがどのテーブルも参照しなかったためです。これはまた、次のネストされた SELECT にも当てはまります。

```
mysql> EXPLAIN SELECT NOW() AS a1, (SELECT f1(5)) AS a2;
+----+-------------+-------+------+---------------+------+---------+------+------+----------------+
| id | select_type | table | type | possible_keys | key  | key_len | ref  | rows | Extra          |
+----+-------------+-------+------+---------------+------+---------+------+------+----------------+
|  1 | PRIMARY     | NULL  | NULL | NULL          | NULL | NULL    | NULL | NULL | No tables used |
+----+-------------+-------+------+---------------+------+---------+------+------+----------------+
1 row in set, 1 warning (0.00 sec)

mysql> SHOW WARNINGS;
+-------+------+-----------------------------------------+
| Level | Code | Message                                 |
+-------+------+-----------------------------------------+
| Note  | 1249 | Select 2 was reduced during optimization |
+-------+------+-----------------------------------------+
1 row in set (0.00 sec)

mysql> SELECT * FROM t2;
Empty set (0.00 sec)
```

ただし、外部の SELECT がいずれかのテーブルを参照している場合、オプティマイザはそのサブクエリー内のステートメントも実行します。

```
mysql> EXPLAIN SELECT * FROM t1 AS a1, (SELECT f1(5)) AS a2;
+----+-------------+------------+--------+---------------+------+---------+------+------+----------------------+
| id | select_type | table      | type   | possible_keys | key  | key_len | ref  | rows | Extra                |
+----+-------------+------------+--------+---------------+------+---------+------+------+----------------------+
|  1 | PRIMARY     | a1         | system | NULL          | NULL | NULL    | NULL |    0 | const row not found  |
|  1 | PRIMARY     | <derived2> | system | NULL          | NULL | NULL    | NULL |    1 |                      |
|  2 | DERIVED     | NULL       | NULL   | NULL          | NULL | NULL    | NULL | NULL | No tables used       |
+----+-------------+------------+--------+---------------+------+---------+------+------+----------------------+
3 rows in set (0.00 sec)

mysql> SELECT * FROM t2;
+------+
| c1   |
+------+
|    5 |
+------+
1 row in set (0.00 sec)
```

これはまた、次に示すような EXPLAIN SELECT ステートメントは、t1 内の行ごとに 1 回 BENCHMARK() 関数が実行されるため、実行に長い時間がかかる可能性があることも示しています。

```
EXPLAIN SELECT * FROM t1 AS a1, (SELECT BENCHMARK(1000000, MD5(NOW())));
```

13.2.10.9 サブクエリーのエラー

サブクエリーにのみ適用されるエラーがいくつか存在します。このセクションでは、これらについて説明します。

- サポートされていないサブクエリー構文:

```
ERROR 1235 (ER_NOT_SUPPORTED_YET)
SQLSTATE = 42000
Message = "This version of MySQL doesn't yet support
'LIMIT & IN/ALL/ANY/SOME subquery'"
```

これは、MySQL が次の形式のステートメントをサポートしていないことを示しています。

```
SELECT * FROM t1 WHERE s1 IN (SELECT s2 FROM t2 ORDER BY s1 LIMIT 1)
```

- サブクエリーからの正しくないカラム数:

```
ERROR 1241 (ER_OPERAND_COL)
SQLSTATE = 21000
Message = "Operand should contain 1 column(s)"
```

このエラーは、次のような場合に発生します。

```
SELECT (SELECT column1, column2 FROM t2) FROM t1;
```

目的が行の比較である場合は、複数のカラムを返すサブクエリーを使用できます。ほかのコンテキストでは、サブクエリーはスカラーオペランドである必要があります。セクション13.2.10.5「行サブクエリー」を参照してください。

- サブクエリーからの正しくない行数:

```
ERROR 1242 (ER_SUBSELECT_NO_1_ROW)
SQLSTATE = 21000
Message = "Subquery returns more than 1 row"
```

このエラーは、サブクエリーが最大で 1 行しか返す必要がないにもかかわらず、複数の行を返すステートメントで発生します。次の例を考えてみます。

```
SELECT * FROM t1 WHERE column1 = (SELECT column1 FROM t2);
```

SELECT column1 FROM t2 が 1 行だけを返す場合、前のクエリーは機能します。このサブクエリーが複数の行を返す場合は、エラー 1242 が発生します。その場合は、このクエリーを次のように書き換えてください。

```
SELECT * FROM t1 WHERE column1 = ANY (SELECT column1 FROM t2);
```

- サブクエリー内の誤って使用されているテーブル:

```
Error 1093 (ER_UPDATE_TABLE_USED)
SQLSTATE = HY000
Message = "You can't specify target table 'x'
for update in FROM clause"
```

このエラーは、テーブルを変更し、さらにサブクエリーで同じテーブルから選択しようとする次のような場合に発生します。

```
UPDATE t1 SET column2 = (SELECT MAX(column1) FROM t1);
```

サブクエリーは SELECT ステートメントだけでなく、UPDATE および DELETE ステートメント内でも正当であるため、UPDATE ステートメント内の割り当てのためにサブクエリーを使用できます。ただし、サブクエリーの FROM 句と更新のターゲットの両方に同じテーブル (この場合は、テーブル t1) を使用することはできません。

トランザクションストレージエンジンの場合は、サブクエリーが失敗するとステートメント全体が失敗します。非トランザクションストレージエンジンの場合は、エラーが検出される前に行われたデータ変更が保持されます。

13.2.10.10 サブクエリーの最適化

開発が進行中であるため、長期にわたって信頼できる最適化のヒントはありません。次のリストは、試してみる価値のある、興味深いいくつかのコツを示しています。

- サブクエリーで、行の数や順序に影響を与えるサブクエリー句を使用します。例:

```
SELECT * FROM t1 WHERE t1.column1 IN
  (SELECT column1 FROM t2 ORDER BY column1);
SELECT * FROM t1 WHERE t1.column1 IN
  (SELECT DISTINCT column1 FROM t2);
SELECT * FROM t1 WHERE EXISTS
```

```
 (SELECT * FROM t2 LIMIT 1);
```

- 結合をサブクエリーに置き換えます。たとえば、次を試してみてください:

```
SELECT DISTINCT column1 FROM t1 WHERE t1.column1 IN (
 SELECT column1 FROM t2);
```

次の代替として:

```
SELECT DISTINCT t1.column1 FROM t1, t2
 WHERE t1.column1 = t2.column1;
```

- サブクエリーをサポートしていない古いバージョンの MySQL との互換性のために、一部のサブクエリーを結合に変換できます。ただし、場合によっては、サブクエリーを結合に変換するとパフォーマンスが向上することがあります。セクション13.2.10.11「サブクエリーの結合としての書き換え」を参照してください。

- 句をサブクエリーの外部から内部に移動します。たとえば、次のクエリーを使用してください:

```
SELECT * FROM t1
 WHERE s1 IN (SELECT s1 FROM t1 UNION ALL SELECT s1 FROM t2);
```

次のクエリーの代替として:

```
SELECT * FROM t1
 WHERE s1 IN (SELECT s1 FROM t1) OR s1 IN (SELECT s1 FROM t2);
```

別の例として、このクエリーを使用してください:

```
SELECT (SELECT column1 + 5 FROM t1) FROM t2;
```

次のクエリーの代替として:

```
SELECT (SELECT column1 FROM t1) + 5 FROM t2;
```

- 相関サブクエリーの代わりに行サブクエリーを使用します。たとえば、次のクエリーを使用してください:

```
SELECT * FROM t1
 WHERE (column1,column2) IN (SELECT column1,column2 FROM t2);
```

次のクエリーの代替として:

```
SELECT * FROM t1
 WHERE EXISTS (SELECT * FROM t2 WHERE t2.column1=t1.column1
               AND t2.column2=t1.column2);
```

- a <> ALL (...) ではなく、NOT (a = ANY (...)) を使用します。

- x=1 OR x=2 ではなく、x = ANY (table containing (1,2)) を使用します。

- EXISTS ではなく、= ANY を使用します。

- 常に 1 行を返す非相関サブクエリーの場合、IN は常に = より低速です。たとえば、次のクエリーを使用してください:

```
SELECT * FROM t1
 WHERE t1.col_name = (SELECT a FROM t2 WHERE b = some_const);
```

次のクエリーの代替として:

```
SELECT * FROM t1
 WHERE t1.col_name IN (SELECT a FROM t2 WHERE b = some_const);
```

これらのコツにより、プログラムは速くなる場合も遅くなる場合もあります。BENCHMARK() 関数などの MySQL 機能を使用すると、現在の状況に何が役立つかについてのアイデアを得ることができます。セクション 12.14「情報関数」を参照してください。

MySQL 自体が行ういくつかの最適化を次に示します。

- MySQL は、非相関サブクエリーを 1 回だけ実行します。特定のサブクエリーが実際に非相関になるようにするには、EXPLAIN を使用します。

- MySQL は、サブクエリー内の選択リストカラムにインデックスが設定される可能性を利用しようとして、IN、ALL、ANY、および SOME サブクエリーを書き換えます。

- MySQL は、次の形式のサブクエリーをインデックス検索関数に置き換えます。この関数は、EXPLAIN によって特殊な結合型 (unique_subquery または index_subquery) として記述されます。

```
... IN (SELECT indexed_column FROM single_table ...)
```

- MySQL は次の形式の式を、NULL 値または空のセットが含まれていないかぎり、MIN() または MAX() を含む式に拡張します。

```
value {ALL|ANY|SOME} {> | < | >= | <=} (uncorrelated subquery)
```

たとえば、次の WHERE 句

```
WHERE 5 > ALL (SELECT x FROM t)
```

は、オプティマイザによって次のように処理される可能性があります。

```
WHERE 5 > (SELECT MAX(x) FROM t)
```

「MySQL Internals: How MySQL Transforms Subqueries」も参照してください。

13.2.10.11 サブクエリーの結合としての書き換え

場合によっては、一連の値におけるメンバーシップをテストするために、サブクエリーを使用する以外の方法が存在することがあります。また、クエリーをサブクエリーなしで書き換えることが可能なだけでなく、サブクエリーを使用する代わりにこれらの手法のいくつかを使用する方が効率的になる場合もあります。これらのうちの 1 つが IN() 構造構文です。

たとえば、次のクエリー

```
SELECT * FROM t1 WHERE id IN (SELECT id FROM t2);
```

は次のように書き換えることができます。

```
SELECT DISTINCT t1.* FROM t1, t2 WHERE t1.id=t2.id;
```

次のクエリー

```
SELECT * FROM t1 WHERE id NOT IN (SELECT id FROM t2);
SELECT * FROM t1 WHERE NOT EXISTS (SELECT id FROM t2 WHERE t1.id=t2.id);
```

は次のように書き換えることができます。

```
SELECT table1.*
 FROM table1 LEFT JOIN table2 ON table1.id=table2.id
 WHERE table2.id IS NULL;
```

LEFT [OUTER] JOIN は、サーバーがそれをより適切に最適化できる可能性がある (MySQL Server だけに特有の事実ではありません) ため、同等のサブクエリーより高速になる場合があります。SQL-92 より前は、外部結合が存在しなかったため、サブクエリーが特定の処理を実行するための唯一の方法でした。今日では、MySQL Server やその他の多くの最新データベースシステムがさまざまなタイプの外部結合を提供しています。

MySQL Server は、1 つのテーブルからの情報や、場合によっては一度に多数のテーブルからの情報に基づいて行を効率的に削除するために使用できる複数テーブルの DELETE ステートメントをサポートしています。また、複数テーブルの UPDATE ステートメントもサポートされています。セクション13.2.2「DELETE 構文」およびセクション13.2.11「UPDATE 構文」を参照してください。

13.2.11 UPDATE 構文

単一テーブル構文:

```
UPDATE [LOW_PRIORITY] [IGNORE] table_reference
   SET col_name1={expr1|DEFAULT} [, col_name2={expr2|DEFAULT}] ...
   [WHERE where_condition]
   [ORDER BY ...]
   [LIMIT row_count]
```

複数テーブル構文:

```
UPDATE [LOW_PRIORITY] [IGNORE] table_references
   SET col_name1={expr1|DEFAULT} [, col_name2={expr2|DEFAULT}] ...
```

```
[WHERE where_condition]
```

単一テーブル構文の場合、UPDATE ステートメントは、指定されたテーブル内の既存の行のカラムを新しい値に更新します。SET 句は、変更するカラムと、それらのカラムに指定される値を示します。各値は式か、またはカラムを明示的にそのデフォルト値に設定するキーワード DEFAULT として指定できます。WHERE 句 (指定されている場合) は、どの行を更新するかを識別する条件を指定します。WHERE 句がない場合は、すべての行が更新されます。ORDER BY 句が指定されている場合は、指定されている順序で行が更新されます。LIMIT 句は、更新できる行数に制限を設定します。

複数テーブル構文の場合、UPDATE は、条件を満たす table_references で指定されている各テーブル内の行を更新します。一致した各行は、条件に複数回一致した場合でも、1 回更新されます。複数テーブル構文の場合は、ORDER BY および LIMIT を使用できません。

パーティション化されたテーブルの場合は、このステートメントの単一テーブルと複数テーブルの両方の形式で、テーブル参照の一部としての PARTITION オプションの使用がサポートされます。このオプションは、1 つ以上のパーティションまたはサブパーティション (またはその両方) のリストを受け取ります。リストされているパーティション (またはサブパーティション) だけが一致をチェックされ、これらのパーティションまたはサブパーティションのいずれにも存在しない行は、where_condition を満たすかどうかにかかわらず更新されません。

> **注記**
>
> INSERT または REPLACE ステートメントで PARTITION を使用している場合とは異なり、それ以外は有効な UPDATE ... PARTITION ステートメントは、リストされているパーティション (またはサブパーティション) 内のどの行も where_condition に一致しない場合でも成功したと見なされます。

詳細および例については、セクション19.5「パーティション選択」を参照してください。

where_condition は、更新される各行に対して true に評価される式です。式の構文については、セクション9.5「式の構文」を参照してください。

table_references と where_condition は、セクション13.2.9「SELECT 構文」で説明されているように指定されます。

実際に更新された、UPDATE 内で参照されているカラムに対してのみ UPDATE 権限が必要です。読み取られるが、変更されないカラムに対しては、SELECT 権限のみが必要です。

UPDATE ステートメントは、次の修飾子をサポートします。

- LOW_PRIORITY キーワードを使用すると、UPDATE の実行は、ほかのどのクライアントもそのテーブルから読み取らなくなるまで遅延されます。これは、テーブルレベルロックのみを使用するストレージエンジン (MyISAM、MEMORY、および MERGE) にのみ影響を与えます。
- IGNORE キーワードを指定すると、更新中にエラーが発生した場合でも、更新ステートメントは中止されません。一意のキー値に関して重複キーの競合が発生した行は更新されません。データ変換エラーの原因になる値に更新された行は、代わりに、もっとも近い有効な値に更新されます。

MySQL 5.6.4 以降では、UPDATE IGNORE ステートメント (ORDER BY 句が存在するものを含む)、には、ステートメントベースのレプリケーションには安全でないというフラグが付けられます。(これは、どの行が無視されるかが、行が更新される順序によって決定されるためです。)この変更により、このようなステートメントは、ステートメントベースモードを使用しているときは口グ内に警告を生成し、MIXED モードを使用しているときは行ベース形式を使用して口グに記録されます。(Bug #11758262、Bug #50439) 詳細は、セクション17.1.2.3「バイナリロギングでの安全および安全でないステートメントの判断」を参照してください。

式で更新されるテーブルのカラムにアクセスする場合、UPDATE はそのカラムの現在の値を使用します。たとえば、次のステートメントは、col1 をその現在の値より 1 大きい値に設定します。

```
UPDATE t1 SET col1 = col1 + 1;
```

次のステートメントの 2 番目の割り当ては、col2 を元の col1 値ではなく、現在の (更新された) col1 値に設定します。この結果、col1 と col2 の値が同じになります。この動作は標準 SQL とは異なります。

```
UPDATE t1 SET col1 = col1 + 1, col2 = col1;
```

単一テーブルの UPDATE の割り当ては一般に、左から右に評価されます。複数テーブルの更新では、割り当てが特定の順序で実行される保証はありません。

カラムをその現在の値に設定した場合は、MySQL がこれに気付き、その更新を行いません。

NOT NULL として宣言されているカラムを NULL に設定することによって更新すると、厳密な SQL モードが有効になっている場合は、エラーが発生します。そうでない場合、カラムはそのカラムデータ型の暗黙のデフォルト値に設定され、警告数が 1 増やされます。暗黙のデフォルト値は、数値型では 0、文字列型では空の文字列 (''')、および日付と時間型では「0」の値です。セクション11.6「データ型デフォルト値」を参照してください。

UPDATE は、実際に変更された行数を返します。mysql_info() C API 関数は、一致して更新された行数と、UPDATE 中に発生した警告の数を返します。

LIMIT row_count を使用すると、UPDATE のスコープを制限できます。LIMIT 句は、一致した行の制限です。このステートメントは、実際に変更されたかどうかにかかわらず、WHERE 句を満たす row_count 行を見つけるとすぐに停止します。

UPDATE ステートメントに ORDER BY 句が含まれている場合は、この句で指定されている順序で行が更新されます。これは、通常であればエラーが発生する可能性のある特定の状況で役立つ場合があります。テーブル t に、一意のインデックスを持つカラム id が含まれているとします。次のステートメントは、行が更新される順序によっては、重複キーエラーで失敗する可能性があります。

```
UPDATE t SET id = id + 1;
```

たとえば、このテーブルの id カラムに 1 と 2 が含まれており、2 が 3 に更新される前に 1 が 2 に更新された場合は、エラーが発生します。この問題を回避するには、大きな id 値を持つ行が小さな値を持つ行の前に更新されるように、ORDER BY 句を追加します。

```
UPDATE t SET id = id + 1 ORDER BY id DESC;
```

また、複数のテーブルを範囲に含む UPDATE 操作を実行することもできます。ただし、複数テーブルの UPDATE では ORDER BY または LIMIT を使用できません。table_references 句は、結合に含まれるテーブルをリストします。その構文については、セクション13.2.9.2「JOIN 構文」で説明されています。次に例を示します。

```
UPDATE items,month SET items.price=month.price
WHERE items.id=month.id;
```

前の例は、カンマ演算子を使用する内部結合を示していますが、複数テーブルの UPDATE ステートメントは、SELECT ステートメント内で許可されている任意の型の結合 (LEFT JOIN など) を使用できます。

外部キー制約が存在する InnoDB テーブルを含む、複数テーブルの UPDATE ステートメントを使用した場合は、MySQL オプティマイザが、それらの親子関係の順序とは異なる順序でテーブルを処理する可能性があります。この場合、このステートメントは失敗し、ロールバックされます。代わりに、1 つのテーブルを更新したあと、InnoDB が提供する ON UPDATE 機能を使用して、ほかのテーブルがそれに応じて変更されるようにします。セクション14.6.6「InnoDB と FOREIGN KEY 制約」を参照してください。

現在、テーブルを更新し、さらにサブクエリーで同じテーブルから選択することはできません。

MySQL 5.6.6 より前は、テーブルレベルのロックを採用した MyISAM などのストレージエンジンを使用しているパーティション化されたテーブルに対する UPDATE によって、そのテーブルのすべてのパーティションがロックされました。これは、UPDATE ... PARTITION クエリーにも当てはまりました。(これは、行レベルロックを採用した InnoDB などのストレージエンジンでは発生しておらず、現在も発生しません。)MySQL 5.6.6 以降では、MySQL はパーティションロックプルーニングを使用します。これにより、そのテーブルのいずれかのパーティション化カラムが更新されないかぎり、UPDATE ステートメントの WHERE 句に一致する行を含むパーティションだけが実際にロックされるようになります。詳細は、セクション19.6.4「パーティショニングとロック」を参照してください。

13.3 MySQL トランザクションおよびロックステートメント

MySQL は、SET autocommit、START TRANSACTION、COMMIT、ROLLBACK などのステートメントを介して (特定のクライアントセッション内の) ローカルトランザクションをサポートしています。セクション13.3.1「START TRANSACTION、COMMIT、および ROLLBACK 構文」を参照してください。XA トランザクションサポートにより、MySQL は分散トランザクションにも参加できます。セクション13.3.7「XA トランザクション」を参照してください。

13.3.1 START TRANSACTION、COMMIT、および ROLLBACK 構文

```
START TRANSACTION
    [transaction_characteristic [, transaction_characteristic] ...]
```

```
transaction_characteristic:
    WITH CONSISTENT SNAPSHOT
  | READ WRITE
  | READ ONLY

BEGIN [WORK]
COMMIT [WORK] [AND [NO] CHAIN] [[NO] RELEASE]
ROLLBACK [WORK] [AND [NO] CHAIN] [[NO] RELEASE]
SET autocommit = {0 | 1}
```

次のステートメントにより、トランザクションの使用を制御できます。

- START TRANSACTION または BEGIN は、新しいトランザクションを開始します。
- COMMIT は、現在のトランザクションをコミットして、その変更を永続的なものにします。
- ROLLBACK は、現在のトランザクションをロールバックして、その変更を取り消します。
- SET autocommit は、現在のセッションのデフォルトの自動コミットモードを無効または有効にします。

デフォルトでは、MySQL は自動コミットモードが有効になった状態で動作します。つまり、テーブルを更新 (変更) するステートメントを実行するとすぐに、MySQL によってその更新がディスクに格納されて永続的になります。この変更はロールバックできません。

一連のステートメントに対して自動コミットモードを暗黙的に無効にするには、START TRANSACTION ステートメントを使用します。

```
START TRANSACTION;
SELECT @A:=SUM(salary) FROM table1 WHERE type=1;
UPDATE table2 SET summary=@A WHERE type=1;
COMMIT;
```

START TRANSACTION を使用すると、そのトランザクションを COMMIT または ROLLBACK で終了するまで、自動コミットは無効のままになります。そのあと、自動コミットモードはその以前の状態に戻ります。

START TRANSACTION では、トランザクションの特性を制御するいくつかの修飾子が許可されます。複数の修飾子を指定するには、それらをカンマで区切ります。

- WITH CONSISTENT SNAPSHOT 修飾子は、この機能に対応しているストレージエンジンでの一貫性読み取りを開始します。これは、InnoDB にのみ適用されます。その効果は、任意の InnoDB テーブルから START TRANSACTION に続けて SELECT を発行することと同じです。セクション14.2.4「一貫性非ロック読み取り」を参照してください。WITH CONSISTENT SNAPSHOT 修飾子は、現在のトランザクション分離レベルを変更しないため、現在の分離レベルが一貫性読み取りを許可するものである場合にのみ、整合性のあるスナップショットを提供します。一貫性読み取りを許可する分離レベルは、REPEATABLE READ だけです。その他のすべての分離レベルの場合、WITH CONSISTENT SNAPSHOT 句は無視されます。
- READ WRITE および READ ONLY 修飾子は、トランザクションアクセスモードを設定します。これらは、そのトランザクションで使用されるテーブルへの変更を許可または禁止します。READ ONLY の制限は、そのトランザクションが、ほかのトランザクションに表示されるトランザクションテーブルと非トランザクションテーブルの両方を変更または口ックしないようにします。このトランザクションは引き続き、一時テーブルを変更または口ックできます。これらの修飾子は、MySQL 5.6.5 の時点で使用できます。

  MySQL では、トランザクションが読み取り専用であることがわかっている場合、InnoDB テーブルに対するクエリーの追加の最適化が可能です。READ ONLY を指定すると、読み取り専用ステータスを自動的に特定できない場合に、これらの最適化が適用されることが保証されます。詳細は、セクション14.13.14「InnoDB の読み取り専用トランザクションの最適化」を参照してください。

  アクセスモードが指定されていない場合は、デフォルトモードが適用されます。デフォルトが変更されていないかぎり、それは読み取り/書き込みです。同じステートメント内で READ WRITE と READ ONLY の両方を指定することは許可されません。

  読み取り専用モードでは、DML ステートメントを使用して TEMPORARY キーワードで作成されたテーブルは引き続き変更できます。永続的なテーブルと同様に、DDL ステートメントによって行われる変更は許可されません。

  トランザクションアクセスモードの詳細 (デフォルトモードを変更する方法を含む) は、セクション13.3.6「SET TRANSACTION 構文」を参照してください。

  read_only システム変数が有効になっている場合、トランザクションを START TRANSACTION READ WRITE で明示的に開始するには、SUPER 権限が必要です。

> **重要**
>
> MySQL クライアントアプリケーションを記述するために使用される多くの API (JDBC など) は、クライアントから START TRANSACTION ステートメントを送信する代わりに使用できる (また、場合によっては使用すべき)、トランザクションを開始するための独自のメソッドを提供しています。詳細は、第23章「Connector および API」または API のドキュメントを参照してください。

自動コミットモードを明示的に無効にするには、次のステートメントを使用します。

```
SET autocommit=0;
```

autocommit 変数を 0 に設定することによって自動コミットモードを無効にしたあと、トランザクションセーフテーブル (InnoDB または NDB のテーブルなど) への変更がただちに永続的になることはありません。COMMIT を使用して変更をディスクに格納するか、または ROLLBACK を使用して変更を無視する必要があります。

autocommit はセッション変数であるため、セッションごとに設定する必要があります。新しい接続ごとに自動コミットモードを無効にするには、セクション5.1.4「サーバーシステム変数」にある autocommit システム変数の説明を参照してください。

BEGIN と BEGIN WORK は、トランザクションを開始するための START TRANSACTION のエイリアスとしてサポートされています。標準の SQL 構文である START TRANSACTION は、アドホックトランザクションを開始するための推奨される方法であり、BEGIN では許可されない修飾子が許可されます。

BEGIN ステートメントは、BEGIN ... END 複合ステートメントを開始する BEGIN キーワードの使用とは異なります。後者はトランザクションを開始しません。セクション13.6.1「BEGIN ... END 複合ステートメント構文」を参照してください。

> **注記**
>
> すべてのストアドプログラム (ストアドプロシージャーとストアドファンクション、トリガー、およびイベント) 内で、パーサーは、BEGIN [WORK] を BEGIN ... END ブロックの開始として扱います。このコンテキストでは、代わりに START TRANSACTION を使用してトランザクションを開始します。

オプションの WORK キーワードは、CHAIN および RELEASE 句と同様に、COMMIT と ROLLBACK に対してサポートされています。CHAIN と RELEASE は、トランザクションの完了に対する追加の制御に使用できます。completion_type システム変数の値によって、デフォルトの完了動作が決定されます。セクション5.1.4「サーバーシステム変数」を参照してください。

AND CHAIN 句を指定すると、現在のトランザクションが終了するとすぐに新しいトランザクションが開始され、新しいトランザクションの分離レベルは終了したばかりのトランザクションと同じになります。RELEASE 句を指定すると、サーバーは、現在のトランザクションを終了したあと現在のクライアントセッションを切り離します。NO キーワードを含めると、CHAIN または RELEASE の完了が抑制されます。これは、completion_type システム変数がデフォルトで、チェーンまたはリリースの完了が実行されるように設定されている場合に役立つことがあります。

トランザクションを開始すると、保留中のトランザクションはすべてコミットされます。詳細は、セクション13.3.3「暗黙的なコミットを発生させるステートメント」を参照してください。

また、トランザクションを開始すると、ユーザーが UNLOCK TABLES を実行したかのように、LOCK TABLES によって取得されたテーブルロックも解放されます。トランザクションを開始しても、FLUSH TABLES WITH READ LOCK によって取得されたグローバルな読み取りロックは解放されません。

最適な結果を得るために、トランザクションは、1 つのトランザクションセーフストレージエンジンによって管理されているテーブルのみを使用して実行するようにしてください。そうしないと、次の問題が発生する場合があります。

- 複数のトランザクションセーフストレージエンジン (InnoDB など) からのテーブルを使用しており、かつトランザクション分離レベルが SERIALIZABLE でない場合は、あるトランザクションがコミットしたときに、同じテーブルを使用している別の進行中のトランザクションに最初のトランザクションによって行われた変更の一部しか表示されない可能性があります。つまり、混在したエンジンではトランザクションのアトミック性が保証されないため、不整合が発生する場合があります。(混在したエンジンでのトランザクションの頻度が低い場合は、SET TRANSACTION ISOLATION LEVEL を使用して、必要に応じてトランザクションごとに分離レベルを SERIALIZABLE に設定できます。)
- トランザクション内でトランザクションセーフでないテーブルを使用する場合は、自動コミットモードのステータスには関係なく、それらのテーブルへの変更が一度に格納されます。

- トランザクション内で非トランザクションテーブルを更新したあとに ROLLBACK ステートメントを発行すると、ER_WARNING_NOT_COMPLETE_ROLLBACK 警告が発生します。トランザクションセーフテーブルへの変更はロールバックされますが、非トランザクションセーフテーブルへの変更はロールバックされません。

各トランザクションは、COMMIT 時に、1 つのまとまりでバイナリログに格納されます。ロールバックされたトランザクションはログに記録されません。(例外: 非トランザクションテーブルへの変更はロールバックできません。ロールバックされるトランザクションに非トランザクションテーブルへの変更が含まれている場合は、非トランザクションテーブルへの変更が確実にレプリケートされるようにするために、最後に ROLLBACK ステートメントを使用してトランザクション全体がログに記録されます。) セクション5.2.4「バイナリログ」を参照してください。

トランザクションの分離レベルまたはアクセスモードは、SET TRANSACTION ステートメントを使用して変更できます。セクション13.3.6「SET TRANSACTION 構文」を参照してください。

ロールバックは、ユーザーが明示的に求めることなく (たとえば、エラーの発生時に) 暗黙的に発生する可能性のある低速な操作になる場合があります。このため、ROLLBACK ステートメントを使用して実行された明示的なロールバックに対してだけでなく、暗黙のロールバックに対しても、SHOW PROCESSLIST はセッションの State カラムに Rolling back を表示します。

> **注記**
>
> MySQL 5.6 では、BEGIN、COMMIT、および ROLLBACK は --replicate-do-db または --replicate-ignore-db ルールによって影響を受けません。

13.3.2 ロールバックできないステートメント

いくつかのステートメントはロールバックできません。これには一般に、データベースを作成または削除したり、テーブルやストアドルーチンを作成、削除、または変更したりするデータ定義言語 (DDL) ステートメントが含まれます。

このようなステートメントを含まないようにトランザクションを設計してください。ロールバックできないステートメントをトランザクション内で早期に発行し、そのあと別のステートメントが失敗したとすると、このような場合に ROLLBACK ステートメントを発行してもそのトランザクションのすべての効果をロールバックすることはできません。

13.3.3 暗黙的なコミットを発生させるステートメント

このセクションに示されているステートメント (およびそのすべてのシノニム) は、ユーザーがこのステートメントを実行する前に COMMIT を実行したかのように、現在のセッション内でアクティブなすべてのトランザクションを暗黙的に終了します。MySQL 5.5.3 の時点では、これらのステートメントのほとんどが、実行後に暗黙的なコミットも発生させます。詳細は、このセクションの最後を参照してください。

- データベースオブジェクトを定義または変更するデータ定義言語 (DDL) ステートメント。ALTER DATABASE ... UPGRADE DATA DIRECTORY NAME、ALTER EVENT、ALTER PROCEDURE、ALTER SERVER、ALTER TABLE、ALTER VIEW、CREATE DATABASE、CREATE EVENT、CREATE INDEX、CREATE PROCEDURE、CREATE SERVER、CREATE TABLE、CREATE TRIGGER、CREATE VIEW、DROP DATABASE、DROP EVENT、DROP INDEX、DROP PROCEDURE、DROP SERVER、DROP TABLE、DROP TRIGGER、DROP VIEW、RENAME TABLE、TRUNCATE TABLE。

  ALTER FUNCTION、CREATE FUNCTION、および DROP FUNCTION もまた、ストアドファンクション (ただし、UDF を除く) で使用された場合は暗黙的なコミットを発生させます。(ALTER FUNCTION は、ストアドファンクションでのみ使用できます。)

  CREATE TABLE および DROP TABLE ステートメントは、TEMPORARY キーワードが使用されている場合はトランザクションをコミットしません。(これは、コミットを発生させる ALTER TABLE や CREATE INDEX などの、一時テーブルに対するその他の操作には適用されません。) ただし、暗黙的なコミットは発生しませんが、ステートメントのロールバックもできません。つまり、このようなステートメントを使用すると、トランザクションのアトミック性が侵害されます。たとえば、CREATE TEMPORARY TABLE を使用したあとにトランザクションをロールバックしても、そのテーブルは存在し続けます。

  InnoDB での CREATE TABLE ステートメントは、1 つのトランザクションとして処理されます。つまり、ユーザーが ROLLBACK を発行しても、ユーザーがそのトランザクション中に実行した CREATE TABLE ステートメントは元に戻されません。

  CREATE TABLE ... SELECT は、一時テーブル以外のテーブルを作成している場合、そのステートメントが実行される前後に暗黙的なコミットを発生させます。(CREATE TEMPORARY TABLE ... SELECT に対してコ

ミットは発生しません。)これは、ロールバック後にマスター上でテーブルを作成できたが、バイナリログへの記録に失敗したため、スレーブにはレプリケートされないというレプリケーション中の問題を回避するために行われます。詳細は、Bug #22865 を参照してください。

- mysql データベース内のテーブルを暗黙的に使用または変更するステートメント。CREATE USER、DROP USER、GRANT、RENAME USER、REVOKE、SET PASSWORD。
- トランザクション制御およびロックステートメント。BEGIN、LOCK TABLES、SET autocommit = 1 (この値がまだ 1 でない場合)、START TRANSACTION、UNLOCK TABLES。

  UNLOCK TABLES は、非トランザクションテーブルロックを取得するために現在 LOCK TABLES でロックされているテーブルがある場合にのみ、トランザクションをコミットします。FLUSH TABLES WITH READ LOCK はテーブルレベルのロックを取得しないため、このステートメントに続く UNLOCK TABLES に対してコミットは発生しません。

  トランザクションをネストすることはできません。これは、START TRANSACTION ステートメントまたはそのシノニムのいずれかを発行するときに、現在のすべてのトランザクションに対して実行される暗黙的なコミットの結果です。

  XA トランザクションが ACTIVE 状態にある間に、暗黙的なコミットを発生させるステートメントをそのトランザクションで使用することはできません。

  BEGIN ステートメントは、BEGIN ... END 複合ステートメントを開始する BEGIN キーワードの使用とは異なります。後者は暗黙的なコミットを発生させません。セクション13.6.1「BEGIN ... END 複合ステートメント構文」を参照してください。
- データロードステートメント。LOAD DATA INFILE。LOAD DATA INFILE は、NDB ストレージエンジンを使用しているテーブルに対してのみ暗黙的なコミットを発生させます。詳細は、Bug #11151 を参照してください。
- 管理ステートメント。ANALYZE TABLE、CACHE INDEX、CHECK TABLE、LOAD INDEX INTO CACHE、OPTIMIZE TABLE、REPAIR TABLE。
- レプリケーション制御ステートメント。MySQL 5.6.7 から: START SLAVE、STOP SLAVE、RESET SLAVE、CHANGE MASTER TO。(Bug #13858841)

MySQL 5.5.3 の時点では、以前は実行前に暗黙的なコミットを発生させたステートメントのほとんどが、実行後にも発生させます。その目的は、このような各ステートメントはいずれにしてもロールバックできないため、それを独自の特殊なトランザクションで処理することにあります。次のリストは、この変更に関連する追加の詳細を示しています。

- 以前は特殊なケースであった CREATE TABLE バリアント (InnoDB テーブルに対する CREATE TABLE や CREATE TABLE ... SELECT) は、CREATE TABLE が一様に実行の前後に暗黙的なコミットを発生させるため、現在では特殊ではなくなっています。
- FLUSH および RESET ステートメントは暗黙的なコミットを発生させます。
- トランザクション制御およびロックステートメントは、以前と同様に動作します。

13.3.4 SAVEPOINT、ROLLBACK TO SAVEPOINT、および RELEASE SAVEPOINT 構文

```
SAVEPOINT identifier
ROLLBACK [WORK] TO [SAVEPOINT] identifier
RELEASE SAVEPOINT identifier
```

InnoDB は、SQL ステートメント SAVEPOINT、ROLLBACK TO SAVEPOINT、RELEASE SAVEPOINT のほか、ROLLBACK のオプションの WORK キーワードをサポートしています。

SAVEPOINT ステートメントは、identifier の名前を持つ名前付きのトランザクションセーブポイントを設定します。現在のトランザクションに同じ名前を持つセーブポイントが含まれている場合、古いセーブポイントは削除され、新しいセーブポイントが設定されます。

ROLLBACK TO SAVEPOINT ステートメントは、トランザクションを終了することなく、そのトランザクションを指定されたセーブポイントにロールバックします。セーブポイントが設定されたあとに現在のトランザクションが行に対して行なった変更はロールバックで元に戻されますが、InnoDB は、セーブポイントのあとにメモリーに格納された行ロックを解放しません。(新しく挿入された行の場合、ロック情報は、その行に格納されているトランザクション ID によって伝達されます。ロックが個別にメモリーに格納されるわけではありません。この場

合、行ロックは Undo で解放されます。)指定されたセーブポイントよりあとで設定されたセーブポイントは削除されます。

ROLLBACK TO SAVEPOINT ステートメントが次のエラーを返した場合は、指定された名前を持つセーブポイントが存在しないことを示しています。

```
ERROR 1305 (42000): SAVEPOINT identifier does not exist
```

RELEASE SAVEPOINT ステートメントは、指定されたセーブポイントを現在のトランザクションの一連のセーブポイントから削除します。コミットまたはロールバックは発生しません。そのセーブポイントが存在しない場合はエラーになります。

COMMIT、またはセーブポイントを指定しない ROLLBACK を実行した場合は、現在のトランザクションのすべてのセーブポイントが削除されます。

ストアドファンクションが呼び出されるか、またはトリガーがアクティブ化されると、新しいセーブポイントレベルが作成されます。以前のレベルにあるセーブポイントは使用できなくなるため、新しいレベルのセーブポイントとは競合しません。関数またはトリガーが終了すると、その関数またはトリガーによって作成されたセーブポイントはすべて解放され、以前のセーブポイントレベルがリストアされます。

13.3.5 LOCK TABLES および UNLOCK TABLES 構文

```
LOCK TABLES
    tbl_name [[AS] alias] lock_type
    [, tbl_name [[AS] alias] lock_type] ...

lock_type:
    READ [LOCAL]
  | [LOW_PRIORITY] WRITE

UNLOCK TABLES
```

MySQL では、クライアントセッションは、ほかのセッションと連携してテーブルにアクセスするために、またはそのセッションにテーブルへの排他的アクセスが必要な期間中はほかのセッションによってそのテーブルが変更されないようにするために、明示的にテーブルロックを取得できます。セッションがロックを取得または解放できるのは、それ自体のためだけです。あるセッションが別のセッションのためにロックを取得したり、別のセッションによって保持されているロックを解放したりすることはできません。

ロックを使用すると、トランザクションをエミュレートするか、またはテーブル更新時の速度を向上させることができます。これについては、このセクションのあとの方でさらに詳細に説明されています。

LOCK TABLES は、現在のクライアントセッションのテーブルロックを明示的に取得します。テーブルロックは、ベーステーブルまたはビューに対して取得できます。ロックされる各オブジェクトに対する LOCK TABLES 権限と SELECT 権限が必要です。

ビューのロックの場合、LOCK TABLES は、そのビューで使用されているすべてのベーステーブルをロックされるテーブルのセットに追加し、それらのテーブルを自動的にロックします。セクション13.3.5.2「LOCK TABLES とトリガー」で説明されているように、LOCK TABLES によって明示的にテーブルをロックした場合は、トリガーで使用されているテーブルもすべて暗黙的にロックされます。

UNLOCK TABLES は、現在のセッションによって保持されているテーブルロックをすべて明示的に解放します。LOCK TABLES は、新しいロックを取得する前に、現在のセッションによって保持されているテーブルロックをすべて暗黙的に解放します。

UNLOCK TABLES の別の使用法として、すべてのデータベース内のすべてのテーブルをロックできる FLUSH TABLES WITH READ LOCK ステートメントによって取得されたグローバルな読み取りロックの解放があります。セクション13.7.6.3「FLUSH 構文」を参照してください。(これは、特定時点のスナップショットを取得できる、Veritas などのファイルシステムがある場合にバックアップを取得するための非常に便利な方法です。)

テーブルロックでは、ほかのセッションによる不適切な読み取りまたは書き込みからのみ保護されます。WRITE ロックを保持しているセッションは、DROP TABLE や TRUNCATE TABLE などのテーブルレベルの操作を実行できます。READ ロックを保持しているセッションの場合、DROP TABLE および TRUNCATE TABLE 操作は許可されません。TRUNCATE TABLE 操作はトランザクションセーフではないため、セッションがアクティブなトランザクション中または READ ロックを保持している間にこの操作を行おうとすると、エラーが発生します。

次の説明は、TEMPORARY 以外のテーブルにのみ適用されます。LOCK TABLES は TEMPORARY テーブルに対して許可されます (ただし、無視されます)。テーブルは、ほかのどのようなロックが有効になっているかには関係なく、そのテーブルが作成されたセッションから自由にアクセスできます。ほかのどのセッションもそのテーブルを参照できないため、ロックは必要ありません。

LOCK TABLES の使用に関するその他の条件や、LOCK TABLES が有効になっている間は使用できないステートメントについては、セクション13.3.5.3「テーブルロックの制限と条件」を参照してください。

ロック取得のルール

現在のセッション内でテーブルロックを取得するには、LOCK TABLES ステートメントを使用します。次のロックタイプを使用できます。

READ [LOCAL] ロック:

- このロックを保持しているセッションは、テーブルを読み取ることができます (ただし、書き込みはできません)。
- 複数のセッションが同時にテーブルに対する READ ロックを取得できます。
- ほかのセッションは、READ ロックを明示的に取得することなく、テーブルを読み取ることができます。
- LOCAL 修飾子を使用すると、ロックが保持されている間、ほかのセッションによる競合しない INSERT ステートメント (並列挿入) を実行できます。(セクション8.10.3「同時挿入」を参照してください。)ただし、ロックを保持している間、サーバーの外部にあるプロセスを使用してデータベースを操作しようとしている場合は、READ LOCAL を使用できません。InnoDB テーブルの場合、READ LOCAL は READ と同じです。

[LOW_PRIORITY] WRITE ロック:

- このロックを保持しているセッションは、テーブルの読み取りおよび書き込みが可能です。
- このロックを保持しているセッションだけがテーブルにアクセスできます。ロックが解放されるまで、ほかのどのセッションもアクセスできません。
- WRITE ロックが保持されている間、テーブルに対するほかのセッションからのロック要求はブロックされます。
- LOW_PRIORITY 修飾子は何の効果もありません。以前のバージョンの MySQL では、ロックの動作に影響を与えましたが、これは当てはまらなくなっています。MySQL 5.6.5 の時点では、これは非推奨であり、使用すると警告が生成されます。代わりに、LOW_PRIORITY のない WRITE を使用してください。

LOCK TABLES ステートメントが、いずれかのテーブルに対するほかのセッションによって保持されているロックのために待機する必要がある場合、このステートメントはすべてのロックを取得できるまでブロックされます。

ロックが必要なセッションは、必要なすべてのロックを 1 つの LOCK TABLES ステートメントで取得する必要があります。このように取得されたロックが保持されている間、このセッションは、ロックされたテーブルにのみアクセスできます。たとえば、次のステートメントシーケンスでは、t2 が LOCK TABLES ステートメントでロックされていないため、このテーブルにアクセスしようとするとエラーが発生します。

```
mysql> LOCK TABLES t1 READ;
mysql> SELECT COUNT(*) FROM t1;
+----------+
| COUNT(*) |
+----------+
|        3 |
+----------+
mysql> SELECT COUNT(*) FROM t2;
ERROR 1100 (HY000): Table 't2' was not locked with LOCK TABLES
```

INFORMATION_SCHEMA データベース内のテーブルは例外です。これらのテーブルは、セッションが LOCK TABLES によって取得されたテーブルロックを保持している間であっても、明示的にロックされることなくアクセスできます。

ロックされたテーブルを、同じ名前を使用して 1 つのクエリーで複数回参照することはできません。代わりにエイリアスを使用し、そのテーブルと各エイリアスのための個別のロックを取得します。

```
mysql> LOCK TABLE t WRITE, t AS t1 READ;
mysql> INSERT INTO t SELECT * FROM t;
ERROR 1100: Table 't' was not locked with LOCK TABLES
mysql> INSERT INTO t SELECT * FROM t AS t1;
```

最初の INSERT では、ロックされたテーブルに対する同じ名前への参照が 2 つ存在するため、エラーが発生します。2 番目の INSERT は、テーブルへの参照で異なる名前が使用されるため、成功します。

ステートメントがエイリアスを使用してテーブルを参照する場合は、その同じエイリアスを使用してテーブルをロックする必要があります。エイリアスを指定しないでテーブルをロックすることはできません。

```
mysql> LOCK TABLE t READ;
mysql> SELECT * FROM t AS myalias;
ERROR 1100: Table 'myalias' was not locked with LOCK TABLES
```

逆に、エイリアスを使用してテーブルをロックする場合は、ステートメント内でそのエイリアスを使用してテーブルを参照する必要があります。

```
mysql> LOCK TABLE t AS myalias READ;
mysql> SELECT * FROM t;
ERROR 1100: Table 't' was not locked with LOCK TABLES
mysql> SELECT * FROM t AS myalias;
```

WRITE ロックは通常、更新ができるだけ早く処理されるように、READ ロックより高い優先度を持っています。つまり、あるセッションが READ ロックを取得したあと、別のセッションが WRITE ロックを要求した場合は、WRITE ロックを要求したセッションがロックを取得して解放するまで、以降の READ ロック要求が待たされます。

LOCK TABLES は、次のようにロックを取得します。

1. ロックされるすべてのテーブルを内部で定義された順序でソートします。ユーザーから見て、この順序は定義されていません。
2. テーブルを読み取りおよび書き込みロックでロックする場合は、書き込みロック要求を読み取りロック要求の前に配置します。
3. セッションがすべてのロックを取得するまで、1 回につき 1 つのテーブルをロックします。

このポリシーによって、テーブルロックでデッドロックが発生しないことが保証されます。

> 注記
>
> LOCK TABLES または UNLOCK TABLES は、パーティション化されたテーブルに適用された場合、常にテーブル全体をロックまたはロック解除します。これらのステートメントは、パーティションロックプルーニングをサポートしていません。セクション19.6.4「パーティショニングとロック」を参照してください。

ロック解放のルール

セッションによって保持されているテーブルロックが解放される場合は、すべてのテーブルロックが一度に解放されます。セッションは明示的にロックを解放できます。また、特定の状況で、ロックが暗黙的に解放される場合もあります。

- セッションは、UNLOCK TABLES によって明示的にロックを解放できます。
- セッションがすでにロックを保持している間にロックを取得するために LOCK TABLES ステートメントを発行した場合は、新しいロックが付与される前に、その既存のロックが暗黙的に解放されます。
- セッションが (たとえば、START TRANSACTION で) トランザクションを開始した場合は、暗黙的な UNLOCK TABLES が実行され、既存のロックが解放されます。(テーブルロックとトランザクションの間の通信の詳細は、セクション13.3.5.1「テーブルロックとトランザクションの通信」を参照してください。)

クライアントセッションの接続が (正常または異常にかかわらず) 終了した場合、サーバーは、そのセッションによって保持されているすべてのテーブルロック (トランザクションおよび非トランザクション) を暗黙的に解放します。そのクライアントが再接続した場合、ロックは有効でなくなります。さらに、クライアントにアクティブなトランザクションがある場合、サーバーは切断時にそのトランザクションをロールバックし、再接続が発生した場合は、自動コミットが有効になった状態で新しいセッションが開始されます。このため、クライアントは自動再接続を無効にすることが必要になる場合があります。自動再接続が有効な場合、再接続が発生してもクライアントには通知されませんが、すべてのテーブルロックまたは現在のトランザクションが失われます。自動再接続が無効になっている場合は、接続が削除されると、発行された次のステートメントに対してエラーが発生します。クライアントはそのエラーを検出し、ロックの再取得やトランザクションの再実行などの適切なアクションを実行できます。セクション23.7.16「自動再接続動作の制御」を参照してください。

> 注記
>
> ロックされたテーブル上で ALTER TABLE を使用すると、そのテーブルがロック解除される場合があります。たとえば、2 番目の ALTER TABLE 操作を試みると、エラー「テーブル 'tbl_name' は LOCK TABLES でロックされていません」が発生する場合があります。これに対処するには、2 番目の変更の前にテーブルを再度ロックします。セクションB.5.7.1「ALTER TABLE での問題」も参照してください。

13.3.5.1 テーブルロックとトランザクションの通信

LOCK TABLES および UNLOCK TABLES は、トランザクションの使用との間で次のように通信します。

- LOCK TABLES はトランザクションセーフではないため、テーブルをロックしようとする前に、アクティブなトランザクションをすべて暗黙的にコミットします。
- UNLOCK TABLES は、アクティブなトランザクションをすべて暗黙的にコミットしますが、これが行われるのは、テーブルロックを取得するために LOCK TABLES が使用された場合のみです。たとえば、次の一連のステートメントでは、UNLOCK TABLES がグローバルな読み取りロックを解放しますが、有効なテーブルロックがないためにトランザクションはコミットされません。

```
FLUSH TABLES WITH READ LOCK;
START TRANSACTION;
SELECT ... ;
UNLOCK TABLES;
```

- トランザクションを (たとえば、START TRANSACTION で) 開始すると、現在のトランザクションはすべて暗黙的にコミットされ、既存のテーブルロックが解放されます。
- FLUSH TABLES WITH READ LOCK は、グローバルな読み取りロックを取得しますが、テーブルロックは取得しないため、テーブルロックと暗黙的なコミットに関して LOCK TABLES および UNLOCK TABLES と同じ動作には従いません。たとえば、START TRANSACTION は、グローバルな読み取りロックを解放しません。セクション13.7.6.3「FLUSH 構文」を参照してください。
- 暗黙的にトランザクションのコミットを発生させるその他のステートメントは、既存のテーブルロックを解放しません。このようなステートメントのリストについては、セクション13.3.3「暗黙的なコミットを発生させるステートメント」を参照してください。
- トランザクションテーブル (InnoDB テーブルなど) で LOCK TABLES および UNLOCK TABLES を使用するための正しい方法は、SET autocommit = 0 (START TRANSACTION ではなく) に続けて LOCK TABLES を指定することによってトランザクションを開始し、そのトランザクションを明示的にコミットするまで UNLOCK TABLES を呼び出さないことです。たとえば、テーブル t1 に書き込み、テーブル t2 から読み取る必要がある場合は、次のように実行できます。

```
SET autocommit=0;
LOCK TABLES t1 WRITE, t2 READ, ...;
... do something with tables t1 and t2 here ...
COMMIT;
UNLOCK TABLES;
```

  LOCK TABLES を呼び出すと、InnoDB は内部的に独自のテーブルロックを取得し、MySQL は独自のテーブルロックを取得します。InnoDB は次のコミット時に内部のテーブルロックを解放しますが、MySQL でテーブルロックが解放されるようにするには、UNLOCK TABLES を呼び出す必要があります。autocommit = 1 を指定すると、LOCK TABLES の呼び出しの直後に InnoDB によって内部のテーブルロックが解放され、デッドロックが非常に発生しやすくなる場合があるため、この指定は行わないようにしてください。autocommit = 1 が指定された場合、古いアプリケーションが不必要なデッドロックを回避するのに役立つように、InnoDB は内部のテーブルロックをまったく取得しません。
- ROLLBACK は、テーブルロックを解放しません。

13.3.5.2 LOCK TABLES とトリガー

LOCK TABLES によって明示的にテーブルをロックした場合は、トリガーで使用されているテーブルもすべて暗黙的にロックされます。

- これらのロックは、LOCK TABLES ステートメントによって明示的に取得されるロックと同時に取得されます。
- トリガーで使用されているテーブルに対するロックは、そのテーブルが読み取りのみに使用されているかどうかによって異なります。読み取りのみに使用されている場合は、読み取りロックで十分です。そうでない場合は、書き込みロックが使用されます。
- テーブルが LOCK TABLES によって読み取りに対して明示的にロックされているが、トリガー内で変更される可能性があるために書き込みに対してロックする必要がある場合は、読み取りロックではなく書き込みロックが取得されます。(つまり、トリガー内でのテーブルの表示のために必要な暗黙の書き込みロックによって、テーブルに対する明示的な読み取りロック要求が書き込みロック要求に変換されます。)

次のステートメントを使用して、2 つのテーブル t1 と t2 をロックするとします。

```
LOCK TABLES t1 WRITE, t2 READ;
```

t1 または t2 にトリガーが含まれている場合は、そのトリガー内で使用されているテーブルもロックされます。t1 に、次のように定義されたトリガーが含まれているとします。

```
CREATE TRIGGER t1_a_ins AFTER INSERT ON t1 FOR EACH ROW
BEGIN
  UPDATE t4 SET count = count+1
     WHERE id = NEW.id AND EXISTS (SELECT a FROM t3);
  INSERT INTO t2 VALUES(1, 2);
END;
```

LOCK TABLES ステートメントの結果として、t1 と t2 は、このステートメントに現れるために ロックされます。また、t3 と t4 は、トリガー内で使用されているために ロックされます。

- t1 は、WRITE ロック要求ごとに、書き込みに対して ロックされます。
- t2 は、要求が READ ロックに対するものであったとしても、書き込みに対して ロックされます。これは、トリガー内で t2 に挿入されるために発生します。したがって、READ 要求は WRITE 要求に変換されます。
- t3 は、トリガー内から読み取られるだけであるため、読み取りに対して ロックされます。
- t4 は、トリガー内で更新される可能性があるため、書き込みに対して ロックされます。

13.3.5.3 テーブルロックの制限と条件

テーブルロックを待機しているセッションを終了するために、KILL を安全に使用できます。セクション 13.7.6.4「KILL 構文」を参照してください。

INSERT DELAYED で使用しているテーブルは ロックしないでください。挿入は、ロックを保持するセッションではなく、別のスレッドによって処理される必要があるため、この場合の INSERT DELAYED はエラーになります。

LOCK TABLES および UNLOCK TABLES は、ストアドプログラム内では使用できません。

performance_schema データベース内のテーブルは、setup_xxx テーブルを除き、LOCK TABLES では ロックできません。

LOCK TABLES ステートメントが有効になっている間、次のステートメントは禁止されます。CREATE TABLE、CREATE TABLE ... LIKE、CREATE VIEW、DROP VIEW、およびストアドファンクション、ストアドプロシージャー、イベントでの DDL ステートメント。

一部の操作では、mysql データベース内のシステムテーブルにアクセスする必要があります。たとえば、HELP ステートメントにはサーバー側のヘルプテーブルの内容が必要であり、また CONVERT_TZ() はタイムゾーンテーブルの読み取りが必要になる可能性があります。サーバーは、ユーザーが明示的に ロックしなくても済むように、必要に応じてシステムテーブルを読み取りに対して暗黙的に ロックします。次のテーブルは、今説明したように処理されます。

```
mysql.help_category
mysql.help_keyword
mysql.help_relation
mysql.help_topic
mysql.proc
mysql.time_zone
mysql.time_zone_leap_second
mysql.time_zone_name
mysql.time_zone_transition
mysql.time_zone_transition_type
```

これらのテーブルのいずれかに対する WRITE ロックを LOCK TABLES ステートメントで明示的に設定する場合は、そのテーブルが ロックされる唯一のテーブルである必要があります。ほかのどのテーブルも、同じステートメントでは ロックできません。

1 つの UPDATE ステートメントはすべてアトミックであるため、通常、テーブルを ロックする必要はありません。現在実行中の SQL ステートメントを、ほかのどのセッションも妨げることはできません。ただし、テーブルの ロックによって利点が得られる可能性のある場合がいくつかあります。

- 一連の MyISAM テーブルに対して多くの操作を実行しようとしている場合は、使用しようとしているテーブルを ロックする方がはるかに高速です。MyISAM テーブルを ロックすると、MySQL は ロックされたテーブルのキーキャッシュを UNLOCK TABLES が呼び出されるまでフラッシュしないため、そのテーブルに対する挿

入、更新、または削除が高速化されます。通常、キーキャッシュは各 SQL ステートメントのあとでフラッシュされます。

テーブルロックのマイナス面は、READ によってロックされたテーブルをどのセッションも更新できず (ロックを保持しているセッションを含む)、ロックを保持しているセッションを除き、WRITE によってロックされたテーブルにどのセッションもアクセスできない点です。

- 非トランザクションストレージエンジンに対してテーブルを使用している場合、SELECT と UPDATE の間にテーブルがほかのセッションによって変更されないようにするには、LOCK TABLES を使用する必要があります。次に示す例では、安全に実行するために LOCK TABLES が必要です。

```
LOCK TABLES trans READ, customer WRITE;
SELECT SUM(value) FROM trans WHERE customer_id=some_id;
UPDATE customer
  SET total_value=sum_from_previous_statement
  WHERE customer_id=some_id;
UNLOCK TABLES;
```

LOCK TABLES を使用しない場合は、SELECT ステートメントと UPDATE ステートメントの実行の間に、別のセッションによって trans テーブルに新しい行が挿入される可能性があります。

多くの場合は、相対的な更新 (UPDATE customer SET value=value+new_value) または LAST_INSERT_ID() 関数を使用することによって LOCK TABLES の使用を回避できます。セクション1.7.2.3「トランザクションおよびアトミック操作の違い」を参照してください。

場合によっては、ユーザーレベルのアドバイザリロック関数 GET_LOCK() および RELEASE_LOCK() を使用してテーブルのロックを回避することもできます。高速化のために、これらのロックはサーバーのハッシュテーブル内に保存され、pthread_mutex_lock() と pthread_mutex_unlock() で実装されます。セクション12.18「その他の関数」を参照してください。

ロックポリシーの詳細は、セクション8.10.1「内部ロック方法」を参照してください。

13.3.6 SET TRANSACTION 構文

```
SET [GLOBAL | SESSION] TRANSACTION
    transaction_characteristic [, transaction_characteristic] ...

transaction_characteristic:
    ISOLATION LEVEL level
  | READ WRITE
  | READ ONLY

level:
     REPEATABLE READ
   | READ COMMITTED
   | READ UNCOMMITTED
   | SERIALIZABLE
```

このステートメントは、トランザクションの特性を指定します。これは、カンマで区切られた 1 つ以上の特性値のリストを受け取ります。これらの特性は、トランザクションの分離レベルまたはアクセスモードを設定します。分離レベルは、InnoDB テーブルに対する操作に使用されます。アクセスモードは MySQL 5.6.5 の時点で指定することができ、トランザクションが読み取り/書き込みまたは読み取り専用のどちらのモードで動作するかを示します。

さらに、SET TRANSACTION には、ステートメントのスコープを示すオプションの GLOBAL または SESSION キーワードを含めることができます。

トランザクションの特性のスコープ

トランザクションの特性はグローバルに、現在のセッションに対して、または次のトランザクションに対して設定できます。

- GLOBAL キーワードを指定すると、このステートメントは、以降のすべてのセッションに対してグローバルに適用されます。既存のセッションは影響を受けません。
- SESSION キーワードを指定すると、このステートメントは、現在のセッション内で実行される以降のすべてのトランザクションに適用されます。
- SESSION または GLOBAL キーワードのどちらも指定しない場合、このステートメントは、現在のセッション内で実行される次の (開始されていない) トランザクションに適用されます。

トランザクションの特性をグローバルに変更するには、SUPER 権限が必要です。どのセッションも、そのセッションの特性 (トランザクションの途中であっても)、または次のトランザクションの特性を自由に変更できます。

アクティブなトランザクションが存在する間は、GLOBAL または SESSION を指定しない SET TRANSACTION は許可されません。

```
mysql> START TRANSACTION;
Query OK, 0 rows affected (0.02 sec)

mysql> SET TRANSACTION ISOLATION LEVEL SERIALIZABLE;
ERROR 1568 (25001): Transaction characteristics can't be changed
while a transaction is in progress
```

サーバーの起動時にグローバルなデフォルトの分離レベルを設定するには、コマンド行またはオプションファイルで mysqld に対して --transaction-isolation=level オプションを使用します。このオプションの level の値では、スペースではなくダッシュが使用されるため、許可される値は READ-UNCOMMITTED、READ-COMMITTED、REPEATABLE-READ、または SERIALIZABLE です。たとえば、デフォルトの分離レベルを REPEATABLE READ に設定するには、オプションファイルの [mysqld] セクションで次の行を使用します。

```
[mysqld]
transaction-isolation = REPEATABLE-READ
```

グローバルなトランザクション分離レベルやセッションのトランザクション分離レベルは、tx_isolation システム変数を使用して、実行時にチェックまたは設定できます。

```
SELECT @@GLOBAL.tx_isolation, @@tx_isolation;
SET GLOBAL tx_isolation='REPEATABLE-READ';
SET SESSION tx_isolation='SERIALIZABLE';
```

同様に、サーバーの起動時または実行時にトランザクションアクセスモードを設定するには、--transaction-read-only オプションまたは tx_read_only システム変数を使用します。デフォルトでは、これらは OFF (モードは読み取り/書き込み) ですが、ON に設定して読み取り専用のデフォルトモードにすることができます。

tx_isolation または tx_read_only のグローバルまたはセッション値を設定することは、SET GLOBAL TRANSACTION または SET SESSION TRANSACTION で分離レベルまたはアクセスモードを設定することと同等です。

分離レベルの詳細および使用方法

InnoDB は、ここで説明されている各トランザクション分離レベルを、異なるロックの方法を使用してサポートしています。ACID 準拠が重要な要件である重要なデータに対する操作の場合は、デフォルトの REPEATABLE READ レベルを使用して高度な一貫性を適用できます。あるいは、正確な一貫性や繰り返し可能な結果がロックのためのオーバーヘッドの量の最少化ほど重要でない一括レポートなどの状況では、READ COMMITTED や場合によっては READ UNCOMMITTED を使用して一貫性のルールを緩和できます。SERIALIZABLE は REPEATABLE READ よりさらに厳密なルールを適用し、主に XA トランザクションのほか、並列性やデッドロックに関する問題のトラブルシューティングなどの特殊な状況で使用されます。

これらの分離レベルが InnoDB トランザクションと連携する方法に関する完全な情報については、セクション 14.2.2「InnoDB のトランザクションモデルおよびロック」を参照してください。特に、InnoDB のレコードレベルのロック、およびそれを使用してさまざまな種類のステートメントが実行される方法の詳細は、セクション 14.2.6「InnoDB のレコード、ギャップ、およびネクストキーロック」およびセクション14.2.8「InnoDB のさまざまな SQL ステートメントで設定されたロック」を参照してください。

次のリストは、MySQL が各種のトランザクションレベルをどのようにサポートするかについて説明しています。このリストは、もっとも一般的に使用されるレベルから使用頻度の低い順に並べられています。

- REPEATABLE READ

  これが InnoDB のデフォルトの分離レベルです。一貫性読み取りでは、READ COMMITTED 分離レベルとの重要な違いがあります。同じトランザクション内の一貫性読み取りはすべて、最初の読み取りによって確立されたスナップショットを読み取ります。この規則は、同じトランザクション内で複数のプレーン (非ロック) SELECT ステートメントを発行した場合、これらの SELECT ステートメントは互いに関しても一貫性があることを示します。セクション14.2.4「一貫性非ロック読み取り」を参照してください。

  ロック読み取り (FOR UPDATE または LOCK IN SHARE MODE を含む SELECT)、UPDATE、および DELETE ステートメントの場合、ロックは、そのステートメントが一意のインデックスを一意の検索条件または範囲タイプの検索条件のどちらで使用しているかによって異なります。一意の検索条件を使用した一意のインデックスの場合、InnoDB は見つかったインデックスレコードのみをロックし、その前にあるギャップはロックしませ

ん。その他の検索条件の場合、InnoDB は、ギャップロックまたはネクストキーロックを使用して、範囲に含まれるギャップへのほかのセッションによる挿入をブロックすることによって、スキャンされたインデックス範囲をロックします。

- READ COMMITTED

  一貫性 (非ロック) 読み取りに関して、いくぶん Oracle に似た分離レベルです。各一貫性読み取りが (同じトランザクション内であっても)、独自の新しいスナップショットを設定して読み取ります。セクション14.2.4「一貫性非ロック読み取り」を参照してください。

  ロック読み取り (FOR UPDATE または LOCK IN SHARE MODE を含む SELECT)、UPDATE ステートメント、および DELETE ステートメントの場合、InnoDB はインデックスレコードのみをロックし、その前にあるギャップはロックしないため、ロックされたレコードの横への新しいレコードの自由な挿入が許可されます。

  > 注記
  >
  > MySQL 5.6 では、READ COMMITTED 分離レベルが使用されている場合、または非推奨の innodb_locks_unsafe_for_binlog システム変数が有効になっている場合、外部キー制約チェックと重複キーチェックを除き、InnoDB のギャップロックは存在しません。また、一致しない行に対するレコードロックも、MySQL が WHERE 条件を評価したあとに解放されます。
  >
  > READ COMMITTED を使用するか、または innodb_locks_unsafe_for_binlog を有効にする場合は、行ベースのバイナリロギングを使用する必要があります。

- READ UNCOMMITTED

  SELECT ステートメントは非ロックの方法で実行されますが、以前のバージョンの行が使用される可能性もあります。そのため、この分離レベルを使用すると、このような読み取りには一貫性がありません。これは、ダーティー読み取りとも呼ばれます。そうでなければ、この分離レベルは READ COMMITTED のように機能します。

- SERIALIZABLE

  このレベルは REPEATABLE READ に似ていますが、自動コミットが無効になっている場合、InnoDB はすべてのプレーン SELECT ステートメントを SELECT ... LOCK IN SHARE MODE に暗黙的に変換します。自動コミットが有効になっている場合、SELECT は独自のトランザクションです。したがって、読み取り専用であることがわかっているため、一貫性のある (非ロック) 読み取りとして実行された場合は直列化することができ、ほかのトランザクションのためのブロックは必要ありません。(選択された行がほかのトランザクションによって変更された場合、プレーン SELECT で強制的にブロックするには、自動コミットを無効にします。)

トランザクションアクセスモード

MySQL 5.6.5 の時点では、トランザクションアクセスモードは SET TRANSACTION で指定できます。デフォルトでは、トランザクションは読み取り/書き込みモードで実行され、そのトランザクションで使用されるテーブルに対して読み取りと書き込みの両方が許可されます。このモードは、READ WRITE のアクセスモードを使用して明示的に指定できます。

トランザクションアクセスモードが READ ONLY に設定されている場合は、テーブルへの変更が禁止されます。これにより、書き込みが許可されていない場合に可能になる、ストレージエンジンのパフォーマンス向上が実現される可能性があります。

同じステートメント内で READ WRITE と READ ONLY の両方を指定することは許可されません。

読み取り専用モードでは、DML ステートメントを使用して TEMPORARY キーワードで作成されたテーブルは引き続き変更できます。永続的なテーブルと同様に、DDL ステートメントによって行われる変更は許可されません。

READ WRITE および READ ONLY アクセスモードは、START TRANSACTION ステートメントを使用して個々のトランザクションに対しても指定できます。

13.3.7 XA トランザクション

XA トランザクションのサポートは、InnoDB ストレージエンジンに対して使用できます。MySQL XA 実装は、X/Open CAE ドキュメント分散トランザクション処理: XA 仕様に基づいています。このドキュメントは The Open

Group によって発行されており、http://www.opengroup.org/public/pubs/catalog/c193.htm で入手できます。現在の XA 実装の制限については、セクションD.6「XA トランザクションの制約」で説明されています。

クライアント側には、特殊な要件は何もありません。MySQL サーバーへの XA インタフェースは、XA キーワードで始まる SQL ステートメントで構成されています。MySQL クライアントプログラムは、SQL ステートメントを送信したり、XA ステートメントインタフェースのセマンティクスを理解したりできる必要があります。これらが、最新のクライアントライブラリに対してリンクされている必要はありません。古いクライアントライブラリも機能します。

現在、MySQL Connector の中で、MySQL Connector/J 5.0.0 以降は、XA SQL ステートメントインタフェースを自動的に処理するクラスインタフェースを使用して XA を直接サポートします。

XA は分散トランザクション、つまり、複数の個別のトランザクションリソースがグローバルトランザクションに参加することを許可する機能をサポートしています。トランザクションリソースは多くの場合 RDBMS ですが、ほかの種類のリソースであってもかまいません。

グローバルトランザクションには、それ自体でトランザクションである複数のアクションが含まれますが、そのすべてがグループとして正常に完了するか、またはすべてがグループとしてロールバックされるかのどちらかである必要があります。基本的に、これは ACID プロパティーを「1 レベル上に」拡張することにより、複数の ACID トランザクションを、同じく ACID プロパティーを持つグローバル操作のコンポーネントとして連携して実行できるようにします。(ただし、分散トランザクションに対しては、SERIALIZABLE 分離レベルを使用して ACID プロパティーを実現する必要があります。非分散トランザクションに対しては REPEATABLE READ を使用すれば十分ですが、分散トランザクションに対しては不十分です。)

分散トランザクションのいくつかの例:

- あるアプリケーションが、メッセージングサービスを RDBMS と組み合わせる統合ツールとして機能する場合があります。このアプリケーションは、同じくトランザクションデータベースを含む、メッセージの送信、取得、および処理を行うトランザクションがすべて、確実にグローバルトランザクション内で実行されるようにします。これは、「トランザクション電子メール」と考えることができます。
- アプリケーションが、MySQL サーバーや Oracle サーバー (または複数の MySQL サーバー) などの異なるデータベースサーバーに関連するアクションを実行します。ここで、複数のサーバーに関連するアクションは、各サーバーに対してローカルな個別のトランザクションとしてではなく、グローバルトランザクションの一部として実行する必要があります。
- 銀行が口座情報を RDBMS 内に保持し、現金自動預け払い機 (ATM) を通して現金を出し入れしています。ATM のアクションが口座に正しく反映されるように保証することが必要ですが、これは RDBMS だけでは実行できません。グローバルなトランザクションマネージャーが ATM とデータベースリソースを統合して、財務トランザクションの全体的な一貫性を確保します。

グローバルトランザクションを使用するアプリケーションには、1 つまたは複数のリソースマネージャーと 1 つのトランザクションマネージャーが含まれています。

- リソースマネージャー (RM) は、トランザクションリソースへのアクセスを提供します。データベースサーバーは、1 つの種類のリソースマネージャーです。これは、RM によって管理されているトランザクションをコミットまたはロールバックできる必要があります。
- トランザクションマネージャー (TM) は、グローバルトランザクションの一部であるトランザクションを調整します。これは、これらの各トランザクションを処理する RM と通信します。グローバルトランザクション内の個々のトランザクションは、グローバルトランザクションの「ブランチ」です。グローバルトランザクションとそのブランチは、あとで説明されている名付けスキームによって識別されます。

XA MySQL の MySQL 実装では、MySQL サーバーは、グローバルトランザクション内の XA トランザクションを処理するリソースマネージャーとして機能できます。MySQL サーバーに接続するクライアントプログラムは、トランザクションマネージャーとして機能します。

グローバルトランザクションを実行するには、どのコンポーネントが関連しているかを知り、各コンポーネントをそのコミットまたはロールバックが可能なポイントに持っていくことが必要です。各コンポーネントが自身の成功する能力に関してレポートする内容に応じて、それらのすべてが、アトミックグループとしてコミットまたはロールバックする必要があります。つまり、すべてのコンポーネントがコミットするか、またはすべてのコンポーネントがロールバックする必要があります。グローバルトランザクションを管理するには、いずれかのコンポーネントまたは接続しているネットワークが失敗する可能性があることを考慮に入れる必要があります。

グローバルトランザクションを実行するためのプロセスでは、2 フェーズコミット (2PC) が使用されます。これは、グローバルトランザクションのブランチによって実行されるアクションが実行されたあとに行われます。

1. 最初のフェーズでは、すべてのブランチが準備されます。つまり、これらは TM からコミットの準備を行うよう指示されます。これは通常、ブランチを管理する各 RM が、そのブランチのアクションを安定したストレージ内に記録することを示します。これらのブランチはこれを実行できるかどうかを示し、これらの結果が 2 番目のフェーズに使用されます。

2. 2 番目のフェーズでは、TM が RM にコミットまたはロールバックのどちらを行うかを指示します。すべてのブランチが、準備できたときにコミットできることを示した場合は、すべてのブランチがコミットするよう指示されます。いずれかのブランチが、準備できたときにコミットできないことを示した場合は、すべてのブランチがロールバックするよう指示されます。

場合によっては、グローバルトランザクションで 1 フェーズコミット (1PC) が使用されることがあります。たとえば、グローバルトランザクションが 1 つのトランザクションリソース (つまり、1 つのブランチ) だけで構成されていることがトランザクションマネージャーによって検出された場合は、そのリソースに準備とコミットを一度に行うよう指示できます。

13.3.7.1 XA トランザクションの SQL 構文

MySQL で XA トランザクションを実行するには、次のステートメントを使用します。

```
XA {START|BEGIN} xid [JOIN|RESUME]

XA END xid [SUSPEND [FOR MIGRATE]]

XA PREPARE xid

XA COMMIT xid [ONE PHASE]

XA ROLLBACK xid

XA RECOVER
```

XA START では、JOIN および RESUME 句はサポートされていません。

XA END では、SUSPEND [FOR MIGRATE] 句はサポートされていません。

各 XA ステートメントは XA キーワードで始まり、そのほとんどに xid 値が必要です。xid は XA トランザクション識別子です。これは、このステートメントがどのトランザクションに適用されるかを示します。xid 値はクライアントによって指定されるか、または MySQL サーバーによって生成されます。xid 値には、1 つから 3 つの部分が含まれています。

```
xid: gtrid [, bqual [, formatID ]]
```

gtrid はグローバルトランザクション識別子であり、bqual はブランチ修飾子であり、formatID は、gtrid および bqual 値で使用される形式を識別する数値です。構文で示されているように、bqual と formatID はオプションです。bqual が指定されていない場合、そのデフォルト値は '' です。formatID が指定されていない場合、そのデフォルト値は 1 です。

gtrid と bqual はそれぞれ、最大 64 バイト長 (64 文字ではありません) の文字列リテラルである必要があります。gtrid と bqual は、いくつかの方法で指定できます。引用符で囲まれた文字列 ('ab')、16 進文字列 (0x6162、X'ab')、またはビット値 (b'nnnn') を使用できます。

formatID は符号なし整数です。

gtrid および bqual 値は、MySQL サーバーのベースとなる XA サポートルーチンによってバイト単位で解釈されます。ただし、XA ステートメントを含む SQL ステートメントが解析されている間、サーバーは何からの特定の文字セットで動作します。安全のために、gtrid と bqual は 16 進文字列として記述してください。

xid 値は通常、トランザクションマネージャーによって生成されます。ある TM によって生成される値は、ほかの TM によって生成される値とは異なっている必要があります。特定の TM は、XA RECOVER ステートメントによって返された値のリスト内の自身の xid 値を認識できる必要があります。

XA START xid は、指定された xid 値を使用して XA トランザクションを開始します。各 XA トランザクションが一意の xid 値を持っている必要があるため、その値が現在、別の XA トランザクションによって使用されていてはいけません。一意性は、gtrid および bqual 値を使用して評価されます。XA トランザクションに対する以降のすべての XA ステートメントを、XA START ステートメントで指定されたものと同じ xid 値を使用して指定する必要があります。これらのステートメントのいずれかを使用しているが、既存の XA トランザクションに対応していない xid 値を指定した場合は、エラーが発生します。

1 つ以上の XA トランザクションを同じグローバルトランザクションの一部にすることができます。特定のグローバルトランザクション内のすべての XA トランザクションが xid 値内の同じ gtrid 値を使用する必要があります。

このため、特定の XA トランザクションがどのグローバルトランザクションの一部であるかについてのあいまいさがないように、gtrid 値はグローバルに一意である必要があります。xid 値の bqual 部分は、グローバルトランザクション内の XA トランザクションごとに異なっている必要があります。(bqual 値が異なっているという要件は、現在の MySQL XA 実装の制限です。これは XA 仕様の一部ではありません。)

XA RECOVER ステートメントは、PREPARED 状態にある MySQL サーバー上の XA トランザクションに関する情報を返します。(セクション13.3.7.2「XA トランザクションの状態」を参照してください。) この出力には、どのクライアントによって開始されたかには関係なく、サーバー上のこのような XA トランザクションごとの行が含まれています。

XA RECOVER の出力行は次のようになります ('abc'、'def'、7 の各部分から成る xid 値の例の場合)。

```
mysql> XA RECOVER;
+----------+--------------+--------------+--------+
| formatID | gtrid_length | bqual_length | data   |
+----------+--------------+--------------+--------+
|        7 |            3 |            3 | abcdef |
+----------+--------------+--------------+--------+
```

出力カラムには次の意味があります。

- formatID は、トランザクション xid の formatID 部分です。
- gtrid_length は、xid の gtrid 部分の長さ (バイト単位) です。
- bqual_length は、xid の bqual 部分の長さ (バイト単位) です。
- data は、xid の gtrid および bqual 部分の連結です。

13.3.7.2 XA トランザクションの状態

XA トランザクションは、次の各状態を経由して処理されます。

1. XA START を使用して、XA トランザクションを開始し、それを ACTIVE 状態にします。
2. ACTIVE XA トランザクションに対しては、トランザクションを構成する SQL ステートメントを発行したあと、XA END ステートメントを発行します。XA END は、トランザクションを IDLE 状態にします。
3. IDLE XA トランザクションに対しては、XA PREPARE ステートメントまたは XA COMMIT ... ONE PHASE ステートメントのどちらかを発行できます。
   - XA PREPARE は、トランザクションを PREPARED 状態にします。この時点での XA RECOVER ステートメントは、XA RECOVER が PREPARED 状態にあるすべての XA トランザクションをリストするため、その出力にトランザクションの xid 値が含まれます。
   - XA COMMIT ... ONE PHASE は、トランザクションの準備とコミットを行います。トランザクションが終了するため、xid 値は XA RECOVER によってリストされません。
4. PREPARED XA トランザクションに対しては、XA COMMIT ステートメントを発行してトランザクションをコミットおよび終了するか、または XA ROLLBACK を発行してトランザクションをロールバックおよび終了することができます。

グローバルトランザクションの一部としてテーブルに行を挿入する単純な XA トランザクションを次に示します。

```
mysql> XA START 'xatest';
Query OK, 0 rows affected (0.00 sec)

mysql> INSERT INTO mytable (i) VALUES(10);
Query OK, 1 row affected (0.04 sec)

mysql> XA END 'xatest';
Query OK, 0 rows affected (0.00 sec)

mysql> XA PREPARE 'xatest';
Query OK, 0 rows affected (0.00 sec)

mysql> XA COMMIT 'xatest';
Query OK, 0 rows affected (0.00 sec)
```

特定のクライアント接続のコンテキスト内では、XA トランザクションとローカル (非 XA) トランザクションは相互に排他的です。たとえば、XA トランザクションを開始するために XA START が発行された場合は、その

XA トランザクションがコミットまたはロールバックされるまでローカルトランザクションを開始できません。逆に、START TRANSACTION を使用してローカルトランザクションが開始された場合は、そのトランザクションがコミットまたはロールバックされるまで XA ステートメントを使用できません。

XA トランザクションが ACTIVE 状態にある場合は、暗黙的なコミットを発生させるどのステートメントも発行できません。その XA トランザクションをロールバックできないため、それを行うことは XA 規約に違反します。このようなステートメントを実行しようとすると、次のエラーが表示されます。

```
ERROR 1399 (XAE07): XAER_RMFAIL: The command cannot be executed
when global transaction is in the ACTIVE state
```

前の注意事項が適用されるステートメントは、セクション13.3.3「暗黙的なコミットを発生させるステートメント」に示されています。

13.4 レプリケーションステートメント

レプリケーションは、このセクションで説明されているステートメントを使用した SQL インタフェースを通して制御できます。ステートメントの 1 つのグループがマスターサーバーを制御し、もう 1 つのグループがスレーブサーバーを制御します。

13.4.1 マスターサーバーを制御するための SQL ステートメント

このセクションでは、マスターレプリケーションサーバーを管理するためのステートメントについて説明します。セクション13.4.2「スレーブサーバーを制御するための SQL ステートメント」では、スレーブサーバーを管理するためのステートメントについて説明しています。

ここで説明されているステートメントに加えて、レプリケーションのマスターサーバーでは次の SHOW ステートメントが使用されます。これらのステートメントについては、セクション13.7.5「SHOW 構文」を参照してください。

- SHOW BINARY LOGS
- SHOW BINLOG EVENTS
- SHOW MASTER STATUS
- SHOW SLAVE HOSTS

13.4.1.1 PURGE BINARY LOGS 構文

```
PURGE { BINARY | MASTER } LOGS
   { TO 'log_name' | BEFORE datetime_expr }
```

バイナリログは、MySQL サーバーによって行われたデータ変更に関する情報を含む一連のファイルです。このログは一連のバイナリログファイルのほか、インデックスファイルで構成されています (セクション5.2.4「バイナリログ」を参照してください)。

PURGE BINARY LOGS ステートメントは、指定されたログファイル名または日付の前にあるログインデックスファイルにリストされているすべてのバイナリログファイルを削除します。BINARY と MASTER はシノニムです。削除されたログファイルはインデックスファイル内に記録されているリストからも削除されるため、特定のログファイルがそのリスト内の先頭になります。

このステートメントは、サーバーがバイナリロギングを有効にする --log-bin オプションで起動されていない場合は何の効果もありません。

例:

```
PURGE BINARY LOGS TO 'mysql-bin.010';
PURGE BINARY LOGS BEFORE '2008-04-02 22:46:26';
```

BEFORE バリアントの datetime_expr 引数は、DATETIME 値 ('YYYY-MM-DD hh:mm:ss' 形式の値) に評価されます。

このステートメントは、スレーブがレプリケートしている間も安全に実行できます。それらを停止する必要はありません。削除しようとしているログファイルのいずれかを現在読み取っているアクティブなスレーブが存在する場合、このステートメントは何もしません。MySQL 5.6.12 以降では、このような場合はエラーで失敗します。

(Bug #13727933) ただし、スレーブが接続されていないときに、そのスレーブがまだ読み取っていないログファイルのいずれかを偶然にパージした場合、そのスレーブは再接続したあとにレプリケートできなくなります。

バイナリログファイルを安全にパージするには、次の手順に従います。

1. 各スレーブサーバー上で、SHOW SLAVE STATUS を使用して、そのスレーブサーバーがどのログファイルを読み取っているかをチェックします。
2. SHOW BINARY LOGS を使用して、マスターサーバー上のバイナリログファイルのリストを取得します。
3. すべてのスレーブ間でもっとも早いログファイルを特定します。これがターゲットファイルです。すべてのスレーブが最新である場合は、そのリスト上の最後のログファイルになります。
4. 削除しようとしているすべてのログファイルのバックアップを作成します。(この手順はオプションですが、常に実行することをお勧めします。)
5. ターゲットファイルの直前までのすべてのログファイルをパージします。

また、特定の日数が経過したらバイナリログファイルが自動的に期限切れになるように、expire_logs_days システム変数を設定することもできます (セクション5.1.4「サーバーシステム変数」を参照してください)。レプリケーションを使用している場合、スレーブがマスターよりも遅延できる最大日数を下回らないような変数を設定するようにしてください。

PURGE BINARY LOGS TO と PURGE BINARY LOGS BEFORE はどちらも、.index ファイルにリストされているバイナリログファイルが、ほかの何らかの手段 (Linux 上での rm の使用など) によってシステムから削除されている場合はエラーで失敗します。(Bug #18199、Bug #18453) このようなエラーに対処するには、.index ファイル (これは単純なテキストファイルです) を手動で編集して、実際に存在するバイナリログファイルのみがリストされていることを確認したあと、失敗した PURGE BINARY LOGS ステートメントを再度実行します。

13.4.1.2 RESET MASTER 構文

```
RESET MASTER
```

インデックスファイルにリストされているすべてのバイナリログファイルを削除し、バイナリログインデックスファイルを空にリセットして、新しいバイナリログファイルを作成します。

MySQL 5.6.5 以降では、RESET MASTER は gtid_purged システム変数 (MySQL 5.6.8 以前では gtid_lost と呼ばれます) の値や、gtid_executed (MySQL 5.6.9 より前では gtid_done) システム変数のグローバル値 (ただし、そのセッション値を除きます) もクリアします。つまり、このステートメントを実行すると、これらの各値が空の文字列 ('') に設定されます。

このステートメントは、マスターがはじめて起動された場合にのみ使用されるように考慮されています。

> **重要**
>
> RESET MASTER の効果は、PURGE BINARY LOGS の効果とは次の 2 つの重要な点で異なります。
>
> 1. RESET MASTER が、インデックスファイルにリストされているすべてのバイナリログファイルを削除し、.000001 の数字のサフィクスを持つ 1 つの空のバイナリログファイルだけを残すのに対して、PURGE BINARY LOGS では番号はリセットされません。
> 2. RESET MASTER は、いずれかのレプリケーションスレーブの実行中に使用されるようには考慮されていません。スレーブの実行中に使用された場合の RESET MASTER の動作は不定です (そのため、サポートされていません)。これに対して、PURGE BINARY LOGS はレプリケーションスレーブの実行中にも安全に使用できます。
>
> セクション13.4.1.1「PURGE BINARY LOGS 構文」も参照してください。

RESET MASTER は、最初にマスターとスレーブを設定する場合に役立つことがあります。その場合は、設定を次のように確認できます。

1. マスターとスレーブを起動し、レプリケーションを開始します (セクション17.1.1「レプリケーションのセットアップ方法」を参照してください)。
2. マスター上でいくつかのテストクエリーを実行します。

3. それらのクエリーがスレーブにレプリケートされたことを確認します。
4. レプリケーションが正しく実行されている場合は、スレーブ上で STOP SLAVE に続けて RESET SLAVE を発行したあと、不要なすべてのデータがスレーブ上に存在しなくなっていることを確認します。
5. マスター上で RESET MASTER を発行して、テストクエリーをクリーンアップします。

設定を確認し、テストによって生成された不要なファイルやログファイルをすべて削除したら、スレーブを起動してレプリケーションを開始できます。

13.4.1.3 SET sql_log_bin 構文

```
SET sql_log_bin = {0|1}
```

sql_log_bin 変数は、バイナリログへのロギングを実行するかどうかを制御します。デフォルト値は 1 (ロギングを実行する) です。現在のセッションのロギングを変更するには、この変数のセッション値を変更します。この変数を設定するには、セッションユーザーが SUPER 権限を持つ必要があります。スレーブにはレプリケートしたくない変更をマスターに対して行なっている間、セッションでバイナリロギングを一時的に無効にするには、この変数を 0 に設定します。

MySQL 5.5 の時点では、sql_log_bin はグローバルまたはセッション変数として設定できます。sql_log_bin のグローバルな設定は、新しいセッションが開始された時点でのみ検出されます。sql_log_bin をグローバルに設定した場合、以前から実行されているセッションは影響を受けません。

> **警告**
>
> グローバルスコープで sql_log_bin を誤って使用すると、すでに実行されているセッションで行われたすべての変更が引き続きバイナリログに記録され、そのためにレプリケートされてしまいます。この状況によって、レプリケーションの失敗を含む予期しない結果が発生する可能性があるため、グローバルスコープで sql_log_bin を使用する場合は特に注意してください。

MySQL 5.6 では、トランザクションまたはサブクエリー内で @@SESSION.sql_log_bin を設定することはできません。(バグ #53437)

13.4.2 スレーブサーバーを制御するための SQL ステートメント

このセクションでは、スレーブレプリケーションサーバーを管理するためのステートメントについて説明します。セクション13.4.1「マスターサーバーを制御するための SQL ステートメント」では、マスターサーバーを管理するためのステートメントについて説明しています。

ここで説明されているステートメントに加えて、レプリケーションスレーブでは SHOW SLAVE STATUS および SHOW RELAYLOG EVENTS も使用されます。これらのステートメントについては、セクション13.7.5.35「SHOW SLAVE STATUS 構文」およびセクション13.7.5.33「SHOW RELAYLOG EVENTS 構文」を参照してください。

13.4.2.1 CHANGE MASTER TO 構文

```
CHANGE MASTER TO option [, option] ...

option:
    MASTER_BIND = 'interface_name'
  | MASTER_HOST = 'host_name'
  | MASTER_USER = 'user_name'
  | MASTER_PASSWORD = 'password'
  | MASTER_PORT = port_num
  | MASTER_CONNECT_RETRY = interval
  | MASTER_RETRY_COUNT = count
  | MASTER_DELAY = interval
  | MASTER_HEARTBEAT_PERIOD = interval
  | MASTER_LOG_FILE = 'master_log_name'
  | MASTER_LOG_POS = master_log_pos
  | MASTER_AUTO_POSITION = {0|1}
  | RELAY_LOG_FILE = 'relay_log_name'
  | RELAY_LOG_POS = relay_log_pos
  | MASTER_SSL = {0|1}
  | MASTER_SSL_CA = 'ca_file_name'
  | MASTER_SSL_CAPATH = 'ca_directory_name'
  | MASTER_SSL_CERT = 'cert_file_name'
  | MASTER_SSL_CRL = 'crl_file_name'
```

```
  | MASTER_SSL_CRLPATH = 'crl_directory_name'
  | MASTER_SSL_KEY = 'key_file_name'
  | MASTER_SSL_CIPHER = 'cipher_list'
  | MASTER_SSL_VERIFY_SERVER_CERT = {0|1}
  | IGNORE_SERVER_IDS = (server_id_list)

server_id_list:
    [server_id [, server_id] ... ]
```

CHANGE MASTER TO は、スレーブサーバーがマスターサーバーへの接続、マスターバイナリログの読み取り、およびスレーブリレーログの読み取りに使用するパラメータを変更します。また、マスター情報およびリレーログ情報リポジトリの内容も更新します (セクション17.2.2「レプリケーションリレーおよびステータスログ」を参照してください)。CHANGE MASTER TO を使用するには、スレーブレプリケーションスレッドを停止する必要があります (必要に応じて STOP SLAVE を使用します)。MySQL 5.6.11 以降では、gtid_next も AUTOMATIC に設定する必要があります (Bug #16062608)。

次の説明に示されているものを除き、指定されていないオプションはその値を保持します。そのため、ほとんどの場合、変更されないオプションを指定する必要はありません。たとえば、MySQL マスターに接続するためのパスワードが変更された場合は、新しいパスワードについてスレーブに通知するために次のステートメントを発行するだけで済みます。

```
STOP SLAVE; -- if replication was running
CHANGE MASTER TO MASTER_PASSWORD='new3cret';
START SLAVE; -- if you want to restart replication
```

MASTER_HOST、MASTER_USER、MASTER_PASSWORD、および MASTER_PORT はスレーブに、そのマスターに接続する方法に関する情報を提供します。

- MASTER_HOST と MASTER_PORT は、マスターホストのホスト名 (または IP アドレス) とその TCP/IP ポートです。

  > **注記**
  >
  > レプリケーションでは、Unix ソケットファイルを使用できません。また、TCP/IP を使用してマスター MySQL サーバーに接続できる必要があります。

  MASTER_HOST または MASTER_PORT オプションを指定すると、スレーブは (そのオプション値が現在の値と同じ場合でも) マスターサーバーが以前とは異なっていると見なします。この場合、マスターバイナリログファイルの名前と位置の古い値は適用されなくなったと見なされるため、このステートメントで MASTER_LOG_FILE と MASTER_LOG_POS を指定しないと、そのあとに MASTER_LOG_FILE='' と MASTER_LOG_POS=4 が暗黙のうちに付加されます。

  MASTER_HOST='' を設定する (つまり、その値を明示的に空の文字列に設定する) ことは、MASTER_HOST をまったく設定しないことと同じではありません。MySQL 5.5 からは、MASTER_HOST を空の文字列に設定しようとするとエラーで失敗します。以前は、MASTER_HOST を空の文字列に設定すると、そのあとの START SLAVE が失敗しました。(Bug #28796)

  MySQL 5.6.5 以降では、MASTER_HOST やその他の CHANGE MASTER TO オプションに使用されている値にラインフィード (\n または 0x0A) 文字が含まれていないかがチェックされます。このような文字がこれらの値に含まれていると、ステートメントが ER_MASTER_INFO で失敗します。(Bug #11758581、Bug #50801)

- MASTER_USER と MASTER_PASSWORD は、マスターへの接続に使用するアカウントのユーザー名とパスワードです。

  MySQL 5.6.4 以降では、MASTER_USER を空にすることはできません。MASTER_PASSWORD の値を設定するときに MASTER_USER = '' を設定するか、または未設定のままにすると エラーが発生します (Bug #13427949)。

  CHANGE MASTER TO ステートメントで MySQL レプリケーションスレーブアカウントに使用されるパスワードは、長さが 32 文字に制限されます。パスワードがこれより長い場合、このステートメントは成功しますが、超過した文字はすべて暗黙のうちに切り捨てられます。これは MySQL レプリケーションに固有の問題であり、MySQL 5.7 で修正されています。(Bug #11752299、Bug #43439)

  実行中の CHANGE MASTER TO ステートメントのテキスト (MASTER_USER と MASTER_PASSWORD の値を含む) は、並列 SHOW PROCESSLIST ステートメントの出力で確認できます。(SHOW PROCESSLIST には START SLAVE ステートメントの完全なテキストも表示されます。)

MASTER_SSL_xxx オプションは、接続での SSL の使用に関する情報を提供します。これらは、セクション6.3.10.4「SSL コマンドのオプション」およびセクション17.3.7「SSL を使用してレプリケーションをセットアッ

プする」で説明されている --ssl-xxx オプションに対応します。これらのオプションは、SSL サポートなしでコンパイルされたスレーブ上でも変更できます。これらはマスター情報リポジトリに保存されますが、スレーブで SSL サポートが有効になっていない場合は無視されます。MASTER_SSL_CRL と MASTER_SSL_CRLPATH は MySQL 5.6.3 で追加されました。

MASTER_CONNECT_RETRY は、接続再試行の間で待機する秒数を指定します。デフォルトは 60 です。

MASTER_RETRY_COUNT (MySQL 5.6.1 で追加されました) は、再接続の試行の回数を制限し、SHOW SLAVE STATUS (これも MySQL 5.6.1 で追加されました) の出力内の Master_Retry_Count カラムの値を更新します。デフォルト値は 24 * 3600 = 86400 です。MASTER_RETRY_COUNT は古い --master-retry-count サーバーオプションの置き換えを目的にしており、現在ではこの制限を設定するための推奨される方法です。新しいアプリケーションでは --master-retry-count に依存しないようにするとともに、以前のバージョンの MySQL から MySQL 5.6.1 以降にアップグレードする場合は、これに依存する既存のすべてのアプリケーションを、代わりに CHANGE MASTER TO ... MASTER_RETRY_COUNT を使用するように更新することをお勧めします。

MASTER_DELAY は、スレーブがマスターから遅延する必要がある秒数を指定します。マスターから受信されたイベントは、そのマスター上での実行より少なくとも interval 秒あとになるまで実行されません。デフォルトは 0 です。interval が 0 から 2^{31}-1 までの範囲の負ではない整数でない場合は、エラーが発生します。詳細は、セクション17.3.9「遅延レプリケーション」を参照してください。このオプションは MySQL 5.6.0 で追加されました。

MASTER_BIND は、複数のネットワークインタフェースを備えたレプリケーションスレーブ上で使用されることを目的にしており、マスターへの接続にそのスレーブのどのネットワークインタフェースが選択されるかを決定します。

このオプションを使用して構成されたアドレス (存在する場合) は、SHOW SLAVE STATUS からの出力の Master_Bind カラムで確認できます。スレーブステータスログテーブルを使用している (サーバーを --master-info-repository=TABLE で起動している) 場合、この値はまた、mysql.slave_master_info テーブルの Master_bind カラムとしても確認できます。

レプリケーションスレーブを特定のネットワークインタフェースにバインドする機能は、MySQL 5.6.2 で追加されました。これはまた、MySQL Cluster NDB 7.3.1 以降でもサポートされています。

MASTER_HEARTBEAT_PERIOD は、レプリケーションハートビートの間の間隔 (秒単位) を設定します。イベントによってマスターのバイナリログが更新されると常に、次のハートビートの待機中の期間がリセットされます。interval は、0 から 4294967 秒までの範囲とミリ秒単位の分解能を持つ 10 進数値です。0 以外の最小の値は 0.001 です。ハートビートがマスターによって送信されるのは、バイナリログファイル内に未送信のイベントが interval より長い期間にわたって存在しない場合だけです。

マスター接続情報をテーブルにロギングしている場合、MASTER_HEARTBEAT_PERIOD は、mysql.slave_master_info テーブルの Heartbeat カラムの値として確認できます。

interval を 0 に設定すると、ハートビートが完全に無効になります。interval のデフォルト値は、slave_net_timeout を 2 で割った値に等しくなります。

@@GLOBAL.slave_net_timeout を現在のハートビート間隔より小さい値に設定すると、警告が発行されます。ハートビート間隔に対して RESET SLAVE を発行する効果は、その間隔のデフォルト値へのリセットです。

MASTER_LOG_FILE と MASTER_LOG_POS は、スレーブ I/O スレッドが、次回そのスレッドが起動されたときにマスターからの読み取りを開始すべき座標です。RELAY_LOG_FILE と RELAY_LOG_POS は、スレーブ SQL スレッドが、次回そのスレッドが起動されたときにリレーログからの読み取りを開始すべき座標です。MASTER_LOG_FILE または MASTER_LOG_POS のどちらかを指定する場合は、RELAY_LOG_FILE や RELAY_LOG_POS を指定できません。MySQL 5.6.5 以降では、MASTER_LOG_FILE または MASTER_LOG_POS のどちらかを指定する場合は、MASTER_AUTO_POSITION = 1 (このセクションのあとの方で説明されています) も指定できません。MASTER_LOG_FILE と MASTER_LOG_POS のどちらも指定されていない場合、スレーブは、CHANGE MASTER TO が発行される前のスレーブ SQL スレッドの最後の座標を使用します。これにより、たとえば、単に使用するパスワードだけを変更する場合、スレーブ SQL スレッドがスレーブ I/O スレッドに比べて遅くなったとしても、レプリケーションに不連続性が発生しないことが保証されます。

MASTER_AUTO_POSITION は、MySQL 5.6.5 で追加されました。MASTER_AUTO_POSITION = 1 が CHANGE MASTER TO とともに使用された場合、スレーブは、GTID ベースのレプリケーションプロトコルを使用してマスターに接続しようとします。

GTID を使用している場合、スレーブは、どのトランザクションの受信または実行、あるいはその両方をすでに行なっているかをマスターに通知します。このセットを計算するために、スレーブは、gtid_executed のグローバル

値と SHOW SLAVE STATUS からの Retrieved_gtid_set カラムの値を読み取ります。Retrieved_gtid_set には最後に転送されたトランザクションの GTID が、そのトランザクションが部分的にしか転送されなかった場合でも含まれているため、最後に受信された GTID はこのセットから除かれます。そのため、スレーブは次のセットを計算します。

```
UNION(@@GLOBAL.gtid_executed, Retrieved_gtid_set - last_received_GTID)
```

このセットは初期ハンドシェークの一部としてマスターに送信され、マスターは実行したうちの、そのセットに含まれていないすべてのトランザクションを戻します。これらのトランザクションのいずれかがすでにマスターのバイナリログからパージされている場合は、マスターからスレーブにエラー ER_MASTER_HAS_PURGED_REQUIRED_GTIDS が送信され、レプリケーションは開始されません。

GTID ベースのレプリケーションが使用されている場合、MASTER_LOG_FILE と MASTER_LOG_POS によって表された座標は使用されず、代わりにグローバルトランザクション識別子が使用されます。そのため、これらのオプションのどちらかまたは両方を MASTER_AUTO_POSITION とともに使用するとエラーが発生します。

MySQL 5.6.10 からは、SHOW SLAVE STATUS の出力をチェックすることによって、自動ポジショニングが有効な状態でレプリケーションが実行されているかどうかを確認できます。(Bug #15992220)

CHANGE MASTER TO ... MASTER_AUTO_POSITION = 1 を発行する前に、gtid_mode も有効にする必要があります。そうしないと、このステートメントはエラーで失敗します。

GTID を使用したあとに古いファイルベースのレプリケーションプロトコルに戻すには、MASTER_LOG_FILE または MASTER_LOG_POSITION のうちの少なくとも 1 つだけでなく、MASTER_AUTO_POSITION = 0 を指定する新しい CHANGE MASTER TO ステートメントを発行できます。

CHANGE MASTER TO は、RELAY_LOG_FILE または RELAY_LOG_POS が指定されていないかぎり、すべてのリレーログファイルを削除し、新しいリレーログファイルを開始します。指定されている場合、リレーログファイルは保持され、relay_log_purge グローバル変数は暗黙のうちに 0 に設定されます。

MySQL 5.6.2 より前は、RELAY_LOG_FILE には絶対パスが必要でした。MySQL 5.6.2 からは、パスを相対パスにすることができます。その場合は、スレーブのデータディレクトリを基準にしていると見なされます。(Bug #12190)

IGNORE_SERVER_IDS は、0 個以上のサーバー ID のカンマ区切りリストを受け取ります。対応するサーバーから発信されているイベントは、引き続リレーログ内に記録されるログのローテーションおよび削除イベントを除いて無視されます。

循環レプリケーションでは、発信元のサーバーは通常、独自のイベントのターミネータとして機能するため、これらのイベントが複数回適用されることはありません。そのため、このオプションは、循環内のいずれかのサーバーが削除されたときの循環レプリケーションで役立ちます。1、2、3、および 4 のサーバー ID を持つ 4 台のサーバーを含む循環レプリケーションセットアップが存在するとき、サーバー 3 に障害が発生したとします。サーバー 2 からサーバー 4 へのレプリケーションを開始してこのギャップをブリッジする場合、サーバー 4 上で発行する CHANGE MASTER TO ステートメント内に IGNORE_SERVER_IDS = (3) を含めることにより、サーバー 4 にそのマスターとしてサーバー 3 の代わりにサーバー 2 を使用するよう指示できます。それにより、サーバー 4 は、使用されなくなっているサーバーで発信されたすべてのステートメントを無視し、伝播しなくなります。

CHANGE MASTER TO ステートメントが IGNORE_SERVER_IDS オプションなしで発行された場合は、既存のリストがすべて保持されます。無視されるサーバーのリストをクリアするには、このオプションを空のリストとともに使用する必要があります。

```
CHANGE MASTER TO IGNORE_SERVER_IDS = ();
```

RESET SLAVE ALL は、サーバー ID リストには影響を与えません。この問題は MySQL 5.7 で修正されています。(Bug #18816897)

IGNORE_SERVER_IDS にサーバーの独自の ID が含まれているときに、--replicate-same-server-id オプションが有効な状態でそのサーバーが起動された場合は、エラーが発生します。

MySQL 5.6 では、マスター情報リポジトリおよび SHOW SLAVE STATUS の出力によって、現在無視されているサーバーのリストが提供されます。詳細は、セクション17.2.2.2「スレーブステータスログ」およびセクション13.7.5.35「SHOW SLAVE STATUS 構文」を参照してください。

MySQL 5.6 では、CHANGE MASTER TO を呼び出すと、MASTER_HOST、MASTER_PORT、MASTER_LOG_FILE、および MASTER_LOG_POS の以前の値が、実行の前のスレーブの状態に関するその他の情報とともにエラーログに書き込まれます。

MySQL 5.6.7 以降では、CHANGE MASTER TO によって進行中のトランザクションの暗黙的なコミットが発生します。セクション13.3.3「暗黙的なコミットを発生させるステートメント」を参照してください。

CHANGE MASTER TO は、マスターのスナップショットが存在し、かつそのスナップショットの時間に対応するマスターバイナリログの座標が記録されている場合のスレーブの設定に役立ちます。そのスナップショットをスレーブにロードしてスレーブをマスターと同期させたあと、スレーブ上で CHANGE MASTER TO MASTER_LOG_FILE='log_name'、MASTER_LOG_POS=log_pos を実行して、スレーブがマスターバイナリログの読み取りを開始すべき座標を指定できます。

次の例では、スレーブが使用するマスターサーバーを変更し、スレーブが読み取りを開始するマスターバイナリログの座標を確立します。これは、マスターをレプリケートするスレーブを設定する場合に使用されます。

```
CHANGE MASTER TO
  MASTER_HOST='master2.mycompany.com',
  MASTER_USER='replication',
  MASTER_PASSWORD='bigs3cret',
  MASTER_PORT=3306,
  MASTER_LOG_FILE='master2-bin.001',
  MASTER_LOG_POS=4,
  MASTER_CONNECT_RETRY=10;
```

次の例は、使用される頻度の低い操作を示しています。これは、何らかの理由で再度実行したいリレーログファイルがスレーブに存在する場合に使用されます。これを行うために、マスターに到達できる必要はありません。CHANGE MASTER TO を使用し、SQL スレッドを開始する (START SLAVE SQL_THREAD) だけで済みます。

```
CHANGE MASTER TO
  RELAY_LOG_FILE='slave-relay-bin.006',
  RELAY_LOG_POS=4025;
```

この 2 番目の操作は、クラッシュのあとのリカバリのために、スレーブ以外のスタンドアロンサーバーを含む非レプリケーションセットアップで使用することもできます。サーバーがクラッシュし、それをバックアップからリストアしたとします。そのサーバーの独自の (たとえば、myhost-bin.* という名前の) バイナリログファイル (リレーログファイルではなく、通常のバイナリログファイル) を再実行しようとしています。まず、下の手順のとおりに実行せず、バイナリログがサーバーで誤ってパージされた場合に備えて、これらのバイナリログファイルのバックアップコピーをどこか安全な場所に作成します。さらに安全にするために、SET GLOBAL relay_log_purge=0 を使用します。次に、--log-bin オプションなしでサーバーを起動します。代わりに、--replicate-same-server-id、--relay-log=myhost-bin (サーバーに、これらの通常のバイナリログファイルがリレーログファイルであると信じ込ませるため)、および --skip-slave-start オプションを使用します。サーバーが起動したら、次のステートメントを発行します。

```
CHANGE MASTER TO
  RELAY_LOG_FILE='myhost-bin.153',
  RELAY_LOG_POS=410,
  MASTER_HOST='some_dummy_string';
START SLAVE SQL_THREAD;
```

サーバーは独自のバイナリログファイルを読み取って実行し、それによりクラッシュリカバリを実現します。リカバリが完了したら、STOP SLAVE を実行し、サーバーをシャットダウンし、マスター情報およびリレーログ情報リポジトリをクリアして、サーバーをその元のオプションで再起動します。

MASTER_HOST オプションの指定 (ダミー値であっても) は、サーバーにそれがスレーブであると思わせるために必要です。

次の表は、文字列値のオプションに許可される最大長を示しています。

オプション	最大長
MASTER_HOST	60
MASTER_USER	16
MASTER_PASSWORD	32
MASTER_LOG_FILE	255
RELAY_LOG_FILE	255
MASTER_SSL_CA	255
MASTER_SSL_CAPATH	255
MASTER_SSL_CERT	255

オプション	最大長
MASTER_SSL_CRL	255
MASTER_SSL_CRLPATH	255
MASTER_SSL_KEY	255
MASTER_SSL_CIPHER	511

13.4.2.2 MASTER_POS_WAIT() 構文

```
SELECT MASTER_POS_WAIT('master_log_file', master_log_pos [, timeout])
```

これはステートメントではなく、実際には関数です。これは、スレーブがマスターのバイナリログ内の特定の位置までのイベントを読み取って実行したことを確認するために使用されます。完全な説明については、セクション12.18「その他の関数」を参照してください。

13.4.2.3 RESET SLAVE 構文

```
RESET SLAVE [ALL]
```

RESET SLAVE は、スレーブに、マスターのバイナリログ内のそのレプリケーション位置を忘れさせます。このステートメントは、クリーンな開始に使用されるように考慮されています。つまり、マスター情報およびリレーログ情報リポジトリをクリアし、すべてのリレーログファイルを削除して、新しいリレーログファイルを開始します。また、CHANGE MASTER TO の MASTER_DELAY オプションで指定されたレプリケーション遅延も 0 にリセットします。RESET SLAVE を使用するには、スレーブレプリケーションスレッドを停止する必要があります (必要に応じて STOP SLAVE を使用します)。

> **注記**
>
> スレーブ SQL スレッドによってリレーログファイルが完全には実行されていない場合でも、すべてのリレーログファイルが削除されます。(これは、STOP SLAVE ステートメントを発行した場合や、スレーブの負荷が高い場合に、レプリケーションスレーブ上に存在する可能性がある条件です。)

MySQL 5.6 では (MySQL 5.1 以前の場合とは異なり)、RESET SLAVE は、メモリー内に保持されているマスターホスト、マスターポート、マスターユーザー、マスターパスワードなどのレプリケーション接続パラメータを変更しません。つまり、RESET SLAVE のあとに CHANGE MASTER TO ステートメントを必要とすることなく START SLAVE を発行できます。

RESET SLAVE に続いてスレーブの mysqld がシャットダウンされた場合は、接続パラメータがリセットされます。MySQL 5.6.3 以降では、代わりに RESET SLAVE ALL を使用して、これらの接続パラメータをリセットできます (Bug #11809016)。

RESET SLAVE ALL は、CHANGE MASTER TO によって設定された IGNORE_SERVER_IDS リストをクリアしません。この問題は MySQL 5.7 で修正されています。(Bug #18816897)

MySQL 5.6.7 以降では、RESET SLAVE によって進行中のトランザクションの暗黙的なコミットが発生します。セクション13.3.3「暗黙的なコミットを発生させるステートメント」を参照してください。

スレーブ SQL スレッドが停止され、RESET SLAVE が発行されたときにそのスレッドが一時テーブルをレプリケートしている最中であった場合、これらのレプリケートされた一時テーブルはスレーブ上で削除されます。

13.4.2.4 SET GLOBAL sql_slave_skip_counter 構文

```
SET GLOBAL sql_slave_skip_counter = N
```

このステートメントは、マスターからの次の N 個のイベントをスキップします。これは、あるステートメントによって発生したレプリケーション停止からのリカバリに役立ちます。

このステートメントは、スレーブスレッドが実行されていない場合にのみ有効です。そうでないと、エラーが生成されます。

このステートメントを使用する場合は、バイナリログが実際には、イベントグループと呼ばれるグループのシーケンスとして構成される点を理解することが重要です。各イベントグループは、イベントのシーケンスで構成されます。

- トランザクションテーブルの場合、イベントグループはトランザクションに対応しています。

- 非トランザクションテーブルの場合、イベントグループは 1 つの SQL ステートメントに対応しています。

> **注記**
>
> 1 つのトランザクションに、トランザクションテーブルと非トランザクションテーブルの両方の変更を含めることができます。

SET GLOBAL sql_slave_skip_counter を使用してイベントをスキップした結果がグループの途中である場合、スレーブは、そのグループの最後に達するまで引き続きイベントをスキップします。そのあと、次のイベントグループから実行が開始されます。

MySQL 5.6 では、このステートメントを発行すると、RELAY_LOG_FILE、RELAY_LOG_POS、および sql_slave_skip_counter の以前の値がエラーログに書き込まれます。

13.4.2.5 START SLAVE 構文

```
START SLAVE [thread_types] [until_option] [connection_options]

thread_types:
    [thread_type [, thread_type] ... ]

thread_type:
    IO_THREAD | SQL_THREAD

until_option:
    UNTIL {   {SQL_BEFORE_GTIDS | SQL_AFTER_GTIDS} = gtid_set
          |   MASTER_LOG_FILE = 'log_name', MASTER_LOG_POS = log_pos
          |   RELAY_LOG_FILE = 'log_name', RELAY_LOG_POS = log_pos
          |   SQL_AFTER_MTS_GAPS  }

connection_options:
    [USER='user_name'] [PASSWORD='user_pass'] [DEFAULT_AUTH='plugin_name'] [PLUGIN_DIR='plugin_dir']


gtid_set:
    uuid_set [, uuid_set] ...
    | ''

uuid_set:
    uuid:interval[:interval]...

uuid:
    hhhhhhhh-hhhh-hhhh-hhhh-hhhhhhhhhhhh

h:
    [0-9,A-F]

interval:
    n[-n]

    (n >= 1)
```

thread_type オプションのない START SLAVE は、両方のスレーブスレッドを開始します。I/O スレッドはマスターサーバーからのイベントを読み取り、それをリレーログ内に格納します。SQL スレッドはリレーログからのイベントを読み取り、それを実行します。START SLAVE には、SUPER 権限が必要です。

START SLAVE は、スレーブスレッドの開始に成功すると、エラーなしで復帰します。ただし、その場合でも、それらのスレーブスレッドは開始したあとに (たとえば、マスターに接続できない、そのバイナリログを読み取れない、またはその他の何らかの問題のために) 停止する可能性があります。START SLAVE では、これについての警告が発生されません。スレーブのエラーログでスレーブスレッドによって生成されたエラーメッセージをチェックするか、または SHOW SLAVE STATUS を使用してスレーブスレッドが正常に実行されていることをチェックする必要があります。

MySQL 5.6.7 以降では、START SLAVE によって進行中のトランザクションの暗黙的なコミットが発生します。セクション13.3.3「暗黙的なコミットを発生させるステートメント」を参照してください。

MySQL 5.6.11 からは、このステートメントを発行する前に、gtid_next を AUTOMATIC に設定する必要があります (Bug #16062608)。

MySQL 5.6.4 以降では、次のリストで説明されているように、START SLAVE を USER、PASSWORD、DEFAULT_AUTH および PLUGIN_DIR オプションとともに使用したプラガブルなユーザーパスワード認証がサポートされます。

- USER: ユーザー名。PASSWORD が使用されている場合は、空または NULL 文字列に設定したり、未設定のままにしたりすることはできません。
- PASSWORD: パスワード。
- DEFAULT_AUTH: プラグインの名前。デフォルトは MySQL ネイティブ認証です。
- PLUGIN_DIR: プラグインの場所。

MySQL 5.6.4 からは、USER、PASSWORD、DEFAULT_AUTH、PLUGIN_DIR のいずれかを指定するときは、IO_THREAD オプションも同時に指定されていないかぎり、SQL_THREAD オプションを使用できません (Bug #13083642)。

詳細は、セクション6.3.7「プラガブル認証」を参照してください。

これらのいずれかのオプションとともにセキュアでない接続が使用されている場合、サーバーは次の警告を発行します: Sending passwords in plain text without SSL/TLS is extremely insecure

MySQL 5.6.6 から、START SLAVE ... UNTIL は、グローバルトランザクション識別子 (GTID) で使用するための 2 つの追加オプションをサポートしています (セクション17.1.3「グローバルトランザクション識別子を使用したレプリケーション」を参照してください)。これらの各オプションは、引数として 1 つ以上のグローバルトランザクション識別子のセット gtid_set を受け取ります (詳細は、GTID セットを参照してください)。

thread_type が指定されていない場合は、START SLAVE UNTIL SQL_BEFORE_GTIDS を指定すると、スレーブ SQL スレッドは、gtid_set に GTID がリストされている最初のトランザクションに達するまでトランザクションを処理します。START SLAVE UNTIL SQL_AFTER_GTIDS を指定すると、スレーブスレッドは、gtid_set 内の最後のトランザクションが両方のスレッドによって処理されるまですべてのトランザクションを処理します。つまり、START SLAVE UNTIL SQL_BEFORE_GTIDS を指定すると、スレーブ SQL スレッドは gtid_set 内の最初の GTID に達する前に現れたすべてのトランザクションを処理し、START SLAVE UNTIL SQL_AFTER_GTIDS を指定すると、スレーブスレッドはそれぞれがこのセットに GTID が含まれていないトランザクションを見つけるまで、gtid_set 内に GTID が見つかったトランザクションを含むすべてのトランザクションを処理します。SQL_BEFORE_GTIDS と SQL_AFTER_GTIDS はそれぞれ、SQL_THREAD および IO_THREAD オプションをサポートしますが、IO_THREAD を一緒に使用しても現在は何の効果もありません。

たとえば、START SLAVE SQL_THREAD UNTIL SQL_BEFORE_GTIDS = 3E11FA47-71CA-11E1-9E33-C80AA9429562:11-56 を指定すると、スレーブ SQL スレッドはシーケンス番号 11 を持つトランザクションを見つけるまで、server_uuid が 3E11FA47-71CA-11E1-9E33-C80AA9429562 であるマスターから発信されているすべてのトランザクションを処理したあと、このトランザクションを処理することなく停止します。つまり、シーケンス番号 10 を持つトランザクションまでのすべてのトランザクションが処理されます。これに対して、START SLAVE SQL_THREAD UNTIL SQL_AFTER_GTIDS = 3E11FA47-71CA-11E1-9E33-C80AA9429562:11-56 を実行すると、スレーブ SQL スレッドは 11 から 56 までのシーケンス番号を持つすべてのトランザクションを含む、マスターから今指定されたすべてのトランザクションを取得したあと、追加のどのトランザクションも処理することなく停止します。つまり、シーケンス番号 56 を持つトランザクションは、スレーブ SQL スレッドによってフェッチされた最後のトランザクションになります。

MySQL 5.6.14 より前は、示されたトランザクションが完了しても SQL_AFTER_GTIDS はスレーブを停止せず、別の GTID イベントが受信されるまで待機しました (Bug #14767986)。

> **注記**
>
> SQL_BEFORE_GTIDS および SQL_AFTER_GTIDS キーワードは MySQL 5.6.5 サーバー内に存在します。ただし、そのどちらのキーワードも、そのバージョンでは START SLAVE [SQL_THREAD | IO_THREAD] UNTIL のオプションとして正しく機能せず、サポートされたのは MySQL 5.6.6 からのみでした。(Bug#13810456)

START SLAVE UNTIL SQL_AFTER_MTS_GAPS は、MySQL 5.6.6 以降で使用できます。このステートメントを発行すると、マルチスレッドスレーブの SQL スレッドはリレーログ内にそれ以上ギャップが見つからなくなるまで実行してから、停止します。このステートメントは SQL_THREAD オプションを受け取ることができますが、ステートメントの効果は変更されないままです。スレーブ I/O スレッドには影響を与えません (また、IO_THREAD オプションとともに使用することはできません)。START SLAVE UNTIL SQL_AFTER_MTS_GAPS は、スレーブがマルチスレッドモードで エラーで失敗したあと、そのスレーブをマルチスレッドモードからシングルスレッドモードに切り替える前に (つまり、slave_parallel_workers を 0 以外の正の値から元の 0 にリセットするときに) 使用するようにしてください。

失敗したマルチスレッドスレーブをシングルスレッドモードに変更するには、次の一連のステートメントを示されている順序で発行できます。

```
START SLAVE UNTIL SQL_AFTER_MTS_GAPS;

SET @@GLOBAL.slave_parallel_workers = 0;

START SLAVE SQL_THREAD;
```

失敗したマルチスレッドスレーブを relay_log_recovery が有効になった状態で実行していた場合は、START SLAVE UNTIL SQL_AFTER_MTS_GAPS を発行してから CHANGE MASTER TO を実行する必要があります。そうしないと、後者のステートメントが失敗します。

> **注記**
>
> 実行中の START SLAVE ... ステートメントのテキスト全体 (使用された USER または PASSWORD 値を含む) を SHOW PROCESSLIST の出力で表示できます。これはまた、実行中の CHANGE MASTER TO ステートメントのテキスト (MASTER_USER または MASTER_PASSWORD に使用されたすべての値を含む) にも当てはまります。

START SLAVE は、I/O スレッドと SQL スレッドの両方が開始されたあと、ユーザーに肯定応答を送信します。ただし、I/O スレッドはまだ接続していない可能性があります。このため、正常な START SLAVE により SHOW SLAVE STATUS は Slave_SQL_Running=Yes を示しますが、これによって Slave_IO_Running=Yes が保証されるわけではありません (Slave_IO_Running=Yes は I/O スレッドが実行中でかつ接続されている場合だけであるため)。詳細は、セクション13.7.5.35「SHOW SLAVE STATUS 構文」およびセクション17.1.5.1「レプリケーションステータスの確認」を参照してください。

どちらのスレッドを開始するかを指定するために、このステートメントに IO_THREAD および SQL_THREAD オプションを追加できます。MySQL 5.6.4 以降では、USER、PASSWORD、DEFAULT_AUTH、PLUGIN_DIR のいずれかを指定するときは、IO_THREAD オプションも同時に指定されていないかぎり、SQL_THREAD オプションが許可されません (Bug #13083642)。

UNTIL 句 (前の文法では until_option) を追加することにより、スレーブを起動し、SQL スレッドが MASTER_LOG_POS および MASTER_LOG_FILE オプションで指定されているマスターバイナリログ内の特定のポイント、または RELAY_LOG_POS および RELAY_LOG_FILE オプションで示されているスレーブリレーログ内の特定のポイントに達するまで実行するように指定できます。SQL スレッドは、指定されたポイントに達すると停止します。このステートメントで SQL_THREAD オプションが指定されている場合、このオプションは SQL スレッドのみを開始します。それ以外の場合は、両方のスレーブスレッドを開始します。SQL スレッドが実行中である場合、UNTIL 句は無視され、警告が発行されます。UNTIL 句を IO_THREAD オプションとともに使用することはできません。

MySQL 5.6.6 以降では、このセクションの前の方で説明されているように、オプション SQL_BEFORE_GTIDS または SQL_AFTER_GTIDS のいずれかを使用して、START SLAVE UNTIL で特定の GTID または GTID のセットを基準にした停止ポイントを指定することもできます。これらのオプションのいずれかを使用している場合は、SQL_THREAD または IO_THREAD、あるいはこれらの両方を指定できます。また、どちらも指定しないことも可能です。SQL_THREAD のみを指定した場合は、スレーブ SQL スレッドのみがこのステートメントの影響を受けます。IO_THREAD のみが使用された場合は、スレーブ I/O のみが影響を受けます。SQL_THREAD と IO_THREAD の両方が使用されている場合、またはそのどちらも使用されていない場合は、SQL スレッドと I/O スレッドの両方がこのステートメントの影響を受けます。

UNTIL 句は、SQL_AFTER_MTS_GAPS も同時に使用している場合を除き、マルチスレッドスレーブではサポートされません。MySQL 5.6.6 より前は、UNTIL はマルチスレッドスレーブではまったくサポートされていませんでした。

UNTIL 句では、次のいずれかを指定する必要があります。

- ログファイル名とそのファイル内の位置の両方
- (MySQL 5.6.6 以降:) SQL_BEFORE_GTIDS または SQL_AFTER_GTIDS のいずれか
- (MySQL 5.6.6 以降:) SQL_AFTER_MTS_GAPS

マスターとリレーログのオプションを混在させないでください。MySQL 5.6.6 以降では、ログファイルのオプションを GTID オプションと混在させないでください。

UNTIL 条件は、以降の STOP SLAVE ステートメント、UNTIL 句を含まない START SLAVE ステートメント、またはサーバーの再起動によってすべてリセットされます。

ログファイルと位置を指定する場合は、SQL スレッドのみがこのステートメントの影響を受けるにもかかわらず、START SLAVE ... UNTIL で IO_THREAD オプションを使用できます。このような場合、IO_THREAD オプ

ションは無視されます。前の制限は、MySQL 5.6.6 で導入された GTID オプション (SQL_BEFORE_GTIDS および SQL_AFTER_GTIDS) のいずれかを使用している場合は適用されません。このセクションの前の方で説明されているように、GTID オプションは SQL_THREAD と IO_THREAD の両方をサポートします。

UNTIL 句は、レプリケーションのデバッグや、あるイベントがスレーブによってレプリケートされないようにしたいポイントの直前までレプリケーションを続行させるために役立つ場合があります。たとえば、適切でない DROP TABLE ステートメントがマスター上で実行された場合は、UNTIL を使用して、スレーブにそのポイントの直前までしか実行しないよう指示できます。そのイベントがどのようなものかを見つけるには、マスターバイナリログまたはスレーブリレーログに対して mysqlbinlog を使用するか、または SHOW BINLOG EVENTS ステートメントを使用します。

セクション内でレプリケートされたクエリーをスレーブに処理させるために UNTIL を使用している場合は、スレーブサーバーが起動したときに SQL スレッドが実行されるのを回避するために、スレーブを --skip-slave-start オプションで起動することをお勧めします。このオプションはおそらく、予期しないサーバーの再起動によって忘れてしまうことがないように、コマンド行ではなく、オプションファイルで使用することが最善です。

SHOW SLAVE STATUS ステートメントには、UNTIL 条件の現在の値を表示する出力フィールドが含まれています。

非常に古いバージョンの MySQL (4.0.5 より前) では、このステートメントは SLAVE START と呼ばれていました。その構文は、MySQL 5.6.1 の時点で受け入れられなくなりました。

13.4.2.6 STOP SLAVE 構文

```
STOP SLAVE [thread_types]

thread_types:
    [thread_type [, thread_type] ... ]

thread_type: IO_THREAD | SQL_THREAD
```

スレーブスレッドを停止します。STOP SLAVE には、SUPER 権限が必要です。推奨されるベストプラクティスとして、スレーブサーバーを停止する前に、スレーブ上で STOP SLAVE を実行してください (詳細は、セクション5.1.12「シャットダウンプロセス」を参照してください)。

行ベースのロギング形式を使用している場合: 非トランザクションストレージエンジンを使用するいずれかのテーブルをレプリケートしている場合は、スレーブサーバーをシャットダウンする前にスレーブ上で STOP SLAVE または STOP SLAVE SQL_THREAD を実行するようにしてください (このセクションのあとの方にある「注」を参照してください)。

START SLAVE と同様に、このステートメントを IO_THREAD および SQL_THREAD オプションとともに使用すると、停止される 1 つまたは複数のスレッドを指定できます。

MySQL 5.6.7 以降では、STOP SLAVE によって進行中のトランザクションの暗黙的なコミットが発生します。セクション13.3.3「暗黙的なコミットを発生させるステートメント」を参照してください。

MySQL 5.6.11 からは、このステートメントを発行する前に、gtid_next を AUTOMATIC に設定する必要があります (Bug #16062608)。

MySQL 5.6.13 以降では、rpl_stop_slave_timeout システム変数を設定することによって、タイムアウトまでに STOP SLAVE が待機する時間を制御できます。これは、STOP SLAVE ステートメントと、スレーブへのさまざまなクライアント接続を使用するほかのスレーブ SQL ステートメントとの間のデッドロックを回避するために使用できます。(Bug #16856735)

> **注記**
>
> MySQL 5.6 では、STOP SLAVE は、1 つ以上の非トランザクションテーブルに影響を与えている現在のレプリケーションイベントグループが実行を完了する (このようなレプリケーショングループが存在する場合) か、あるいはユーザーが KILL QUERY または KILL CONNECTION ステートメントを発行するまで待機します。(Bug #319、Bug #38205)

古いバージョンの MySQL (4.0.5 より前) では、このステートメントは SLAVE STOP と呼ばれていました。その構文は、MySQL 5.6.1 の時点で受け入れられなくなりました。

13.5 準備済みステートメントのための SQL 構文

MySQL 5.6 は、サーバー側の準備済みステートメントをサポートしています。このサポートでは、MySQL 4.1 から使用可能な、効率的なクライアント/サーバーのバイナリプロトコルを利用しています。パラメータ値のためのプレースホルダを含む準備済みステートメントの使用には、次の利点があります。

- ステートメントを実行のたびに解析するためのオーバーヘッドが少なくなります。通常、データベースアプリケーションは、クエリーや削除の場合の WHERE、更新の場合の SET、挿入の場合の VALUES などの句でリテラルまたは変数値しか変更されていない、ほぼ同一の大量のステートメントを処理します。
- SQL インジェクション攻撃からの保護。パラメータ値には、エスケープされていない SQL 引用符および区切り文字を含めることができます。

アプリケーションプログラムでの準備済みステートメント

サーバー側の準備済みステートメントは、C プログラムのための MySQL C API クライアントライブラリまたは、Java プログラムのための MySQL Connector/J、.NET テクノロジを使用したプログラムのための MySQL Connector/Net などのクライアントプログラミングインタフェース経由で使用できます。たとえば、C API は、その準備済みステートメント API を構成する一連の関数呼び出しを提供しています。セクション23.7.8「C API プリペアドステートメント」を参照してください。ほかの言語インタフェースは、C クライアントライブラリ内でリンクすることによって、バイナリプロトコルを使用する準備済みステートメントに対するサポートを提供できます。その 1 つの例が、PHP 5.0 以降で使用可能な mysqli 拡張機能です。

SQL スクリプトでの準備済みステートメント

準備済みステートメントへの代替 SQL インタフェースを使用できます。このインタフェースは、準備済みステートメント API 経由でのバイナリプロトコルの使用ほど効率的ではありませんが、SQL レベルで直接使用できるためプログラミングが必要ありません。

- 使用できるプログラミングインタフェースが存在しない場合でも使用できます。
- mysql クライアントプログラムなどの、サーバーに SQL ステートメントを送信して実行させることのできる任意のプログラムから使用できます。
- MySQL 4.1 以降を実行しているサーバーに接続しているかぎり、クライアントが古いバージョンのクライアントライブラリを使用している場合でも使用できます。

準備済みステートメントのための SQL 構文は、次のような状況で使用されるように考慮されています。

- 準備済みステートメントのコーディングの前に、それがアプリケーションでどのように動作するかをテストする場合。
- サポートしているプログラミング API にアクセスできないときに準備済みステートメントを使用する場合。
- 準備済みステートメントに関するアプリケーションの問題を対話的にトラブルシューティングする場合。
- バグレポートを提出できるように、準備済みステートメントに関する問題を再現するテストケースを作成する場合。

PREPARE、EXECUTE、および DEALLOCATE PREPARE ステートメント

準備済みステートメントのための SQL 構文は、次の 3 つの SQL ステートメントに基づいています。

- PREPARE は、ステートメントを実行のために準備します (セクション13.5.1「PREPARE 構文」を参照してください)。
- EXECUTE は、準備済みステートメントを実行します (セクション13.5.2「EXECUTE 構文」を参照してください)。
- DEALLOCATE PREPARE は、準備済みステートメントを解放します (セクション13.5.3「DEALLOCATE PREPARE 構文」を参照してください)。

次の例は、2 辺の長さが与えられた三角形の斜辺を計算するステートメントを準備するための 2 つの同等の方法を示しています。

最初の例は、文字列リテラルを使用してステートメントのテキストを指定することによって準備済みステートメントを作成する方法を示しています。

```
mysql> PREPARE stmt1 FROM 'SELECT SQRT(POW(?,2) + POW(?,2)) AS hypotenuse';
mysql> SET @a = 3;
mysql> SET @b = 4;
mysql> EXECUTE stmt1 USING @a, @b;
+------------+
| hypotenuse |
+------------+
|          5 |
+------------+
mysql> DEALLOCATE PREPARE stmt1;
```

2 番目の例も同様ですが、ステートメントのテキストをユーザー変数として指定します。

```
mysql> SET @s = 'SELECT SQRT(POW(?,2) + POW(?,2)) AS hypotenuse';
mysql> PREPARE stmt2 FROM @s;
mysql> SET @a = 6;
mysql> SET @b = 8;
mysql> EXECUTE stmt2 USING @a, @b;
+------------+
| hypotenuse |
+------------+
|         10 |
+------------+
mysql> DEALLOCATE PREPARE stmt2;
```

次の追加の例は、クエリーを実行する対象となるテーブルの名前をユーザー変数として格納することによって、実行時にそのテーブルを選択する方法を示しています。

```
mysql> USE test;
mysql> CREATE TABLE t1 (a INT NOT NULL);
mysql> INSERT INTO t1 VALUES (4), (8), (11), (32), (80);

mysql> SET @table = 't1';
mysql> SET @s = CONCAT('SELECT * FROM ', @table);

mysql> PREPARE stmt3 FROM @s;
mysql> EXECUTE stmt3;
+----+
| a  |
+----+
|  4 |
|  8 |
| 11 |
| 32 |
| 80 |
+----+

mysql> DEALLOCATE PREPARE stmt3;
```

準備済みステートメントは、そのステートメントが作成されたセッションに固有です。以前に作成された準備済みステートメントを解放せずにセッションを終了した場合、そのステートメントはサーバーによって自動的に解放されます。

準備済みステートメントはまた、セッションに対してグローバルでもあります。ストアドルーチン内で準備済みステートメントを作成した場合、そのステートメントはストアドルーチンが終了しても解放されません。

同時に作成される準備済みステートメントが多くなりすぎないようにするには、max_prepared_stmt_count システム変数を設定します。準備済みステートメントの使用を回避するには、この値を 0 に設定します。

準備済みステートメント内で許可される SQL 構文

次の SQL ステートメントは、準備済みステートメントとして使用できます。

```
ALTER TABLE
ALTER USER (as of MySQL 5.6.8)
ANALYZE TABLE
CACHE INDEX
CALL
CHANGE MASTER
CHECKSUM {TABLE | TABLES}
COMMIT
{CREATE | RENAME | DROP} DATABASE
{CREATE | DROP} INDEX
{CREATE | RENAME | DROP} TABLE
{CREATE | RENAME | DROP} USER
{CREATE | DROP} VIEW
```

```
DELETE
DO
FLUSH {TABLE | TABLES | TABLES WITH READ LOCK | HOSTS | PRIVILEGES
  | LOGS | STATUS | MASTER | SLAVE | DES_KEY_FILE | USER_RESOURCES}
GRANT
INSERT
INSTALL PLUGIN
KILL
LOAD INDEX INTO CACHE
OPTIMIZE TABLE
REPAIR TABLE
REPLACE
RESET {MASTER | SLAVE | QUERY CACHE}
REVOKE
SELECT
SET
SHOW {AUTHORS | CONTRIBUTORS | WARNINGS | ERRORS}
SHOW BINLOG EVENTS
SHOW CREATE {PROCEDURE | FUNCTION | EVENT | TABLE | VIEW}
SHOW {MASTER | BINARY} LOGS
SHOW {MASTER | SLAVE} STATUS
SLAVE {START | STOP}
TRUNCATE TABLE
UNINSTALL PLUGIN
UPDATE
```

その他のステートメントは、MySQL 5.6 ではサポートされていません。

通常、SQL 準備済みステートメントで許可されていないステートメントは、ストアドプログラムでも許可されません。例外については、セクションD.1「ストアドプログラムの制約」に示されています。

プリペアドステートメントによって参照されているテーブルやビューのメタデータの変更が検出され、それが次に実行されるときに、ステートメントが自動再準備されます。詳細については、セクション8.9.4「プリペアドステートメントおよびストアドプログラムのキャッシュ」を参照してください。

準備済みステートメントを使用する場合は、LIMIT 句の引数にプレースホルダを使用できます。セクション13.2.9「SELECT 構文」を参照してください。

PREPARE および EXECUTE とともに使用される準備済み CALL ステートメントでは、OUT および INOUT パラメータに対するプレースホルダのサポートが MySQL 5.6 から使用できます。例および以前のバージョンでの回避方法については、セクション13.2.1「CALL 構文」を参照してください。IN パラメータには、バージョンには関係なくプレースホルダを使用できます。

準備済みステートメントのための SQL 構文は、ネストされた方法では使用できません。つまり、PREPARE に渡されるステートメント自体を、PREPARE、EXECUTE、または DEALLOCATE PREPARE ステートメントにすることはできません。

準備済みステートメントのための SQL 構文は、準備済みステートメント API 呼び出しの使用とは異なります。たとえば、mysql_stmt_prepare() C API 関数を使用して、PREPARE、EXECUTE、または DEALLOCATE PREPARE ステートメントを準備することはできません。

準備済みステートメントのための SQL 構文はストアドプロシージャー内で使用できますが、ストアドファンクションまたはトリガー内では使用できません。ただし、PREPARE と EXECUTE で準備および実行される動的なステートメントにはカーソルを使用できません。カーソルのステートメントはカーソル作成時にチェックされるため、そのステートメントを動的にすることはできません。

準備済みステートメントのための SQL 構文は、マルチステートメント (つまり、「;」文字で区切られた 1 つの文字列内の複数のステートメント) をサポートしていません。

プリペアドステートメントは、セクション8.9.3.1「クエリーキャッシュの動作」に説明する状況でクエリーキャッシュを使用します。

CALL SQL ステートメントを使用して、準備済みステートメントを含むストアドプロシージャーを実行する C プログラムを記述するには、CLIENT_MULTI_RESULTS フラグが有効になっている必要があります。これは、各 CALL によって、プロシージャー内で実行されるステートメントによって返される可能性のある結果セットに加えて、呼び出しステータスを示すための結果が返されるためです。

CLIENT_MULTI_RESULTS は、mysql_real_connect() を呼び出すときに、CLIENT_MULTI_RESULTS フラグ自体を渡すことによって明示的に、または CLIENT_MULTI_STATEMENTS を渡すことによって暗黙的に有効にする (これによって CLIENT_MULTI_RESULTS も有効になります) ことができます。詳細は、セクション13.2.1「CALL 構文」を参照してください。

13.5.1 PREPARE 構文

```
PREPARE stmt_name FROM preparable_stmt
```

PREPARE ステートメントは SQL ステートメントを準備し、それに名前 stmt_name を割り当てます。この名前は、あとでそのステートメントを参照するために使用されます。この準備済みステートメントは EXECUTE で実行され、DEALLOCATE PREPARE で解放されます。例については、セクション13.5「準備済みステートメントのための SQL 構文」を参照してください。

ステートメント名は大文字と小文字が区別されません。preparable_stmt は、SQL ステートメントのテキストを含む文字列リテラルまたはユーザー変数のどちらかです。このテキストは複数のステートメントではなく、1 つのステートメントを表している必要があります。このステートメント内では、? 文字を、あとでクエリーを実行するときに、そのクエリーのどこにデータ値をバインドするかを示すパラメータマーカーとして使用できます。文字列値にバインドしようとしている場合でも、? 文字を引用符で囲んではいけません。パラメータマーカーは、SQL キーワードや識別子などではなく、データ値を指定するべき場所にしか使用できません。

指定された名前を持つ準備済みステートメントがすでに存在する場合、そのステートメントは、新しいステートメントが準備される前に暗黙的に解放されます。つまり、新しいステートメントにエラーが含まれていて準備できない場合は、エラーが返され、指定された名前を持つステートメントは存在しなくなります。

準備済みステートメントのスコープは、そのステートメントが作成されたセッションです。これには、次のいくつかの注意点があります。

- あるセッションで作成された準備済みステートメントを別のセッションで使用することはできません。
- セッションが (正常または異常にかかわらず) 終了すると、その準備済みステートメントは存在しなくなります。自動再接続が有効になっていると、クライアントには接続が失われたことが通知されません。このため、クライアントは自動再接続を無効にすることが必要になる場合があります。セクション23.7.16「自動再接続動作の制御」を参照してください。
- ストアドプログラム内で作成された準備済みステートメントは、そのプログラムが実行を完了したあとも引き続き存在し、あとでそのプログラムの外部で実行できます。
- ストアドプログラムのコンテキストで準備されたステートメントは、ストアドプロシージャーやストアドファンクションのパラメータまたはローカル変数を参照できません。これらは、そのプログラムが終了するとスコープから外れ、このステートメントがあとでプログラムの外部で実行されたときに使用できなくなるためです。回避方法として、代わりに、同様にセッションスコープを持つユーザー定義変数を参照します。セクション9.4「ユーザー定義変数」を参照してください。

13.5.2 EXECUTE 構文

```
EXECUTE stmt_name
    [USING @var_name [, @var_name] ...]
```

PREPARE でステートメントを準備したあと、準備済みステートメント名を参照する EXECUTE ステートメントでそのステートメントを実行します。準備済みステートメントにパラメータマーカーが含まれている場合は、そのパラメータにバインドされる値を含むユーザー変数をリストした USING 句を指定する必要があります。パラメータ値はユーザー変数でのみ提供することができ、USING 句では、このステートメント内のパラメータマーカーの数とまったく同じ数の変数を指定する必要があります。

特定の準備済みステートメントを複数回実行できます。それには、各ステートメントに異なる変数を渡すか、または各実行の前にその変数を異なる値に設定します。

例については、セクション13.5「準備済みステートメントのための SQL 構文」を参照してください。

13.5.3 DEALLOCATE PREPARE 構文

```
{DEALLOCATE | DROP} PREPARE stmt_name
```

PREPARE で生成された準備済みステートメントを解放するには、その準備済みステートメント名を参照する DEALLOCATE PREPARE ステートメントを使用します。準備済みステートメントを解放したあとにそのステートメントを実行しようとすると、エラーが発生します。多すぎる準備済みステートメントが作成され、DEALLOCATE PREPARE ステートメントまたはセッションの終了のどちらによっても解放されない場合は、max_prepared_stmt_count システム変数によって上限が適用されることがあります。

例については、セクション13.5「準備済みステートメントのための SQL 構文」を参照してください。

13.6 MySQL 複合ステートメント構文

このセクションでは、BEGIN ... END 複合ステートメントや、ストアドプログラム (ストアドプロシージャーとストアドファンクション、トリガー、およびイベント) の本体で使用できるその他のステートメントの構文について説明します。これらのオブジェクトは、あとで呼び出すためにサーバー上に格納されている SQL コードに対して定義されます (第20章「ストアドプログラムおよびビュー」を参照してください)。

複合ステートメントとは、ほかのブロック、つまり、変数、条件ハンドラ、およびカーソルの宣言、ループや条件付きテストなどのフロー制御構造構文を含めることのできるブロックのことです。

13.6.1 BEGIN ... END 複合ステートメント構文

```
[begin_label:] BEGIN
  [statement_list]
END [end_label]
```

BEGIN ... END 構文は、ストアドプログラム (ストアドプロシージャーとストアドファンクション、トリガー、およびイベント) 内に指定できる複合ステートメントを記述するために使用されます。複合ステートメントには、BEGIN および END キーワードで囲まれた複数のステートメントを含めることができます。statement_list は、それぞれがセミコロン (;) ステートメント区切り文字で終了する 1 つ以上のステートメントのリストを表します。statement_list 自体がオプションであるため、空の複合ステートメント (BEGIN END) は正当です。

BEGIN ... END ブロックはネストできます。

複数のステートメントを使用するには、クライアントが ; ステートメント区切り文字を含むステートメント文字列を送信できる必要があります。mysql コマンド行クライアントでは、これは delimiter コマンドで処理されます。ステートメント終了の区切り文字 ; を (たとえば、// に) 変更すると、プログラム本体での ; の使用が許可されます。例については、セクション20.1「ストアドプログラムの定義」を参照してください。

BEGIN ... END ブロックにはラベルを付けることができます。セクション13.6.2「ステートメントラベルの構文」を参照してください。

オプションの [NOT] ATOMIC 句はサポートされていません。つまり、この命令ブロックの先頭でトランザクションセーブポイントは設定されず、このコンテキストで使用されている BEGIN 句は現在のトランザクションに影響を与えません。

> **注記**
>
> すべてのストアドプログラム内で、パーサーは、BEGIN [WORK] を BEGIN ... END ブロックの開始として扱います。このコンテキストでトランザクションを開始するには、代わりに START TRANSACTION を使用します。

13.6.2 ステートメントラベルの構文

```
[begin_label:] BEGIN
  [statement_list]
END [end_label]

[begin_label:] LOOP
  statement_list
END LOOP [end_label]

[begin_label:] REPEAT
  statement_list
UNTIL search_condition
END REPEAT [end_label]

[begin_label:] WHILE search_condition DO
  statement_list
END WHILE [end_label]
```

BEGIN ... END ブロックや、LOOP、REPEAT、および WHILE ステートメントに対してラベルが許可されます。これらのステートメントに使用されるラベルは、次のルールに従います。

- begin_label のあとにコロンを付ける必要があります。
- begin_label は、end_label なしでも指定できます。end_label が存在する場合、それは begin_label と同じである必要があります。
- end_label は、begin_label なしでは指定できません。

- 同じネストレベルにあるラベルは異なっている必要があります。
- ラベルは最大 16 文字の長さで指定できます。

ラベルが付けられた構造構文内でラベルを参照するには、ITERATE または LEAVE ステートメントを使用します。次の例では、これらのステートメントを使用して繰り返しを続行するか、またはループを終了します。

```
CREATE PROCEDURE doiterate(p1 INT)
BEGIN
  label1: LOOP
    SET p1 = p1 + 1;
    IF p1 < 10 THEN ITERATE label1; END IF;
    LEAVE label1;
  END LOOP label1;
END;
```

ブロックラベルのスコープには、そのブロック内で宣言されているハンドラのコードは含まれません。詳細は、セクション13.6.7.2「DECLARE ... HANDLER 構文」を参照してください。

13.6.3 DECLARE 構文

DECLARE ステートメントは、プログラムにローカルな、次のさまざまな項目を定義するために使用されます。

- ローカル変数。セクション13.6.4「ストアドプログラム内の変数」を参照してください。
- 条件とハンドラ。セクション13.6.7「条件の処理」を参照してください。
- カーソル。セクション13.6.6「カーソル」を参照してください。

DECLARE は、BEGIN ... END 複合ステートメントの内部でのみ許可され、ほかのどのステートメントよりも前の、その複合ステートメントの先頭に存在する必要があります。

宣言は、特定の順序に従う必要があります。カーソル宣言は、ハンドラ宣言の前に指定する必要があります。変数および条件宣言は、カーソルまたはハンドラ宣言の前に指定する必要があります。

13.6.4 ストアドプログラム内の変数

システム変数とユーザー定義変数は、ストアドプログラムのコンテキストの外部で使用できるのと同様に、ストアドプログラム内で使用できます。さらに、ストアドプログラムは DECLARE を使用してローカル変数を定義でき、またストアドルーチン (プロシージャーおよびファンクション) は、そのルーチンとその呼び出し元の間で値を通信するパラメータを受け取るように宣言できます。

- セクション13.6.4.1「ローカル変数の DECLARE 構文」で説明されているように、ローカル変数を宣言するには、DECLARE ステートメントを使用します。
- 変数は、SET ステートメントを使用して直接設定できます。セクション13.7.4「SET 構文」を参照してください。
- クエリーからの結果は、SELECT ... INTO var_list を使用するか、またはカーソルを開き、FETCH ... INTO var_list を使用することによってローカル変数に取得できます。セクション13.2.9.1「SELECT ... INTO 構文」およびセクション13.6.6「カーソル」を参照してください。

ローカル変数のスコープ、および MySQL があいまいな名前を解決する方法については、セクション13.6.4.2「ローカル変数のスコープと解決」を参照してください。

ストアドプロシージャーやストアドファンクションのパラメータまたはストアドプログラムのローカル変数に (たとえば、SET var_name = DEFAULT ステートメントを使用して) 値 DEFAULT を割り当てることは許可されません。MySQL 5.6.6 の時点では、これは構文エラーになります。

13.6.4.1 ローカル変数の DECLARE 構文

```
DECLARE var_name [, var_name] ... type [DEFAULT value]
```

このステートメントは、ストアドプログラム内のローカル変数を宣言します。変数のデフォルト値を指定するには、DEFAULT 句を含めます。この値は式として指定できます。定数である必要はありません。DEFAULT 句がない場合、初期値は NULL になります。

ローカル変数は、データ型やオーバーフローチェックに関して、ストアドルーチンパラメータと同様に処理されます。セクション13.1.15「CREATE PROCEDURE および CREATE FUNCTION 構文」を参照してください。

変数宣言は、カーソルまたはハンドラ宣言の前に指定する必要があります。

ローカル変数名は大文字と小文字が区別されません。セクション9.2「スキーマオブジェクト名」で説明されているように、許可される文字や引用符のルールはほかの識別子の場合と同じです。

ローカル変数のスコープは、それが宣言されている BEGIN ... END ブロックです。この変数は、同じ名前を持つ変数を宣言しているブロックを除き、宣言しているブロック内でネストされたブロック内で参照できます。

13.6.4.2 ローカル変数のスコープと解決

ローカル変数のスコープは、それが宣言されている BEGIN ... END ブロックです。この変数は、同じ名前を持つ変数を宣言しているブロックを除き、宣言しているブロック内でネストされたブロック内で参照できます。

ローカル変数はストアドプログラムの実行中にのみスコープ内にあるので、これらの参照は、ストアドプログラム内で作成された準備済みステートメントでは許可されていません。準備済みステートメントのスコープは現在のセッションであり、ストアドプログラムではないので、ステートメントはプログラムの終了後に実行でき、この時点で変数はスコープ内に存在しなくなります。たとえば、SELECT ... INTO local_var は準備済みステートメントとして使用できません。この制約は、ストアドプロシージャーおよびストアドファンクションのパラメータにも適用されます。セクション13.5.1「PREPARE 構文」を参照してください。

ローカル変数にテーブルカラムと同じ名前を付けてはいけません。SELECT ... INTO ステートメントなどの SQL ステートメントに、カラムおよび同じ名前を持つ宣言されたローカル変数への参照が含まれている場合、MySQL は現在、その参照を変数の名前として解釈します。次のプロシージャー定義を考えてみます。

```
CREATE PROCEDURE sp1 (x VARCHAR(5))
BEGIN
  DECLARE xname VARCHAR(5) DEFAULT 'bob';
  DECLARE newname VARCHAR(5);
  DECLARE xid INT;

  SELECT xname, id INTO newname, xid
    FROM table1 WHERE xname = xname;
  SELECT newname;
END;
```

MySQL は、SELECT ステートメント内の xname を、xname カラムではなく xname 変数への参照として解釈します。その結果、プロシージャー sp1() が呼び出されると、table1.xname カラムの値には関係なく、newname 変数は値 'bob' を返します。

同様に、次のプロシージャー内のカーソル定義には、xname を参照する SELECT ステートメントが含まれています。MySQL はこれをカラム参照ではなく、その名前の変数への参照として解釈します。

```
CREATE PROCEDURE sp2 (x VARCHAR(5))
BEGIN
  DECLARE xname VARCHAR(5) DEFAULT 'bob';
  DECLARE newname VARCHAR(5);
  DECLARE xid INT;
  DECLARE done TINYINT DEFAULT 0;
  DECLARE cur1 CURSOR FOR SELECT xname, id FROM table1;
  DECLARE CONTINUE HANDLER FOR NOT FOUND SET done = 1;

  OPEN cur1;
  read_loop: LOOP
    FETCH FROM cur1 INTO newname, xid;
    IF done THEN LEAVE read_loop; END IF;
    SELECT newname;
  END LOOP;
  CLOSE cur1;
END;
```

セクションD.1「ストアドプログラムの制約」も参照してください。

13.6.5 フロー制御ステートメント

MySQL は、ストアドプログラム内のフロー制御のために、IF、CASE、ITERATE、LEAVE LOOP、WHILE、および REPEAT 構造構文をサポートしています。また、ストアドファンクション内の RETURN もサポートしています。

これらの構造構文の多くには、次の各セクションの文法仕様に示されているその他のステートメントが含まれています。このような構造構文はネストできます。たとえば、IF ステートメントには、それ自体に CASE ステートメントを含む WHILE ループが含まれている可能性があります。

MySQL は、FOR ループをサポートしていません。

13.6.5.1 CASE 構文

```
CASE case_value
    WHEN when_value THEN statement_list
    [WHEN when_value THEN statement_list] ...
    [ELSE statement_list]
END CASE
```

または:

```
CASE
    WHEN search_condition THEN statement_list
    [WHEN search_condition THEN statement_list] ...
    [ELSE statement_list]
END CASE
```

ストアドプログラムの CASE ステートメントは、複雑な条件構造構文を実装します。

> **注記**
>
> ここで説明されている CASE ステートメントとは異なる CASE 式も存在します。セクション12.4「制御フロー関数」を参照してください。CASE ステートメントは ELSE NULL 句を持つことができず、END でなく、END CASE で終了します。

最初の構文の場合、case_value は式です。この値は、各 WHEN 句内の when_value 式のいずれかに等しくなるまで、それらの式と比較されます。等しい when_value が見つかると、対応する THEN 句の statement_list が実行されます。どの when_value も等しくない場合は、ELSE 句の statement_list が実行されます (この句が存在する場合)。

NULL = NULL は false であるため、この構文を NULL と等しいかどうかのテストに使用することはできません。セクション3.3.4.6「NULL 値の操作」を参照してください。

2 番目の構文の場合、各 WHEN 句の search_condition 式のいずれかが true になるまでそれらの式が評価され、いずれかが true になった時点で、それに対応する THEN 句の statement_list が実行されます。どの search_condition も等しくない場合は、ELSE 句の statement_list が実行されます (この句が存在する場合)。

どの when_value も search_condition もテストされた値に一致せず、かつ CASE ステートメントに ELSE 句が含まれていない場合は、Case not found for CASE statement エラーになります。

各 statement_list は、1 つ以上の SQL ステートメントで構成されます。空の statement_list は許可されません。

どの WHEN 句でも値が一致しない状況を処理するには、次の例に示すように、空の BEGIN ... END ブロックを含む ELSE を使用します。(ここの ELSE 句で使用されているインデントは透明性のみを目的にしており、それ以外の意味はありません。)

```
DELIMITER |

CREATE PROCEDURE p()
  BEGIN
    DECLARE v INT DEFAULT 1;

    CASE v
      WHEN 2 THEN SELECT v;
      WHEN 3 THEN SELECT 0;
      ELSE
        BEGIN
        END;
    END CASE;
  END;
  |
```

13.6.5.2 IF 構文

```
IF search_condition THEN statement_list
    [ELSEIF search_condition THEN statement_list] ...
    [ELSE statement_list]
END IF
```

ストアドプログラムの IF ステートメントは、基本的な条件構造構文を実装します。

> **注記**
>
> ここで説明されている IF ステートメントとは異なる IF() 関数も存在します。セクション12.4「制御フロー関数」を参照してください。IF ステートメントは THEN、ELSE、および ELSEIF 句を含むことができ、END IF で終了します。

search_condition が true に評価された場合は、対応する THEN または ELSEIF 句の statement_list が実行されます。どの search_condition も一致しない場合は、ELSE 句の statement_list が実行されます。

各 statement_list は、1 つ以上の SQL ステートメントで構成されます。空の statement_list は許可されません。

IF ... END IF ブロックは、次の例に示すように、ストアドプログラム内で使用されるその他のすべてのフロー制御ブロックと同様にセミコロンで終了する必要があります。

```
DELIMITER //

CREATE FUNCTION SimpleCompare(n INT, m INT)
 RETURNS VARCHAR(20)

 BEGIN
  DECLARE s VARCHAR(20);

  IF n > m THEN SET s = '>';
  ELSEIF n = m THEN SET s = '=';
  ELSE SET s = '<';
  END IF;

  SET s = CONCAT(n, ' ', s, ' ', m);

  RETURN s;
 END //

DELIMITER ;
```

ほかのフロー制御構造構文と同様に、IF ... END IF ブロックは、ほかのフロー制御構造構文 (ほかの IF ステートメントを含む) 内にネストできます。各 IF は、独自の END IF とそれに続くセミコロンで終了する必要があります。次に示すように、インデントを使用して、ネストされたフロー制御ブロックを人間が読みやすくすることができます (ただし、これが MySQL に必要なわけではありません)。

```
DELIMITER //

CREATE FUNCTION VerboseCompare (n INT, m INT)
 RETURNS VARCHAR(50)

 BEGIN
  DECLARE s VARCHAR(50);

  IF n = m THEN SET s = 'equals';
  ELSE
    IF n > m THEN SET s = 'greater';
    ELSE SET s = 'less';
    END IF;

    SET s = CONCAT('is ', s, ' than');
  END IF;

  SET s = CONCAT(n, ' ', s, ' ', m, '.');

  RETURN s;
 END //

DELIMITER ;
```

この例では、内側の IF は n が m に等しくない場合にのみ評価されます。

13.6.5.3 ITERATE 構文

```
ITERATE label
```

ITERATE は、LOOP、REPEAT、および WHILE ステートメント内にのみ指定できます。ITERATE は、「ループをふたたび開始する」ことを示します。

例については、セクション13.6.5.5「LOOP 構文」を参照してください。

13.6.5.4 LEAVE 構文

```
LEAVE label
```

このステートメントは、特定のラベルを持つフロー制御構造構文を終了するために使用されます。そのラベルがもっとも外側のストアドプログラムブロックのものである場合、LEAVE はプログラムを終了します。

LEAVE は、BEGIN ... END またはループ構造構文 (LOOP、REPEAT、WHILE) 内で使用できます。

例については、セクション13.6.5.5「LOOP 構文」を参照してください。

13.6.5.5 LOOP 構文

```
[begin_label:] LOOP
    statement_list
END LOOP [end_label]
```

LOOP は単純なループ構造構文を実装し、それぞれがセミコロン (;) ステートメント区切り文字で終了する 1 つ以上のステートメントで構成されたステートメントリストの繰り返し実行を可能にします。ループ内の各ステートメントは、そのループが終了するまで繰り返されます。通常、これは LEAVE ステートメントで実行されます。ストアドファンクション内では、RETURN も使用できます。これにより、そのストアドファンクションが完全に終了します。

ループ終了ステートメントが含まれていない場合は、無限ループが発生します。

LOOP ステートメントにはラベルを付けることができます。ラベルの使用に関連したルールについては、セクション13.6.2「ステートメントラベルの構文」を参照してください。

例:

```
CREATE PROCEDURE doiterate(p1 INT)
BEGIN
  label1: LOOP
    SET p1 = p1 + 1;
    IF p1 < 10 THEN
      ITERATE label1;
    END IF;
    LEAVE label1;
  END LOOP label1;
  SET @x = p1;
END;
```

13.6.5.6 REPEAT 構文

```
[begin_label:] REPEAT
    statement_list
UNTIL search_condition
END REPEAT [end_label]
```

REPEAT ステートメント内のステートメントリストは、search_condition 式が true になるまで繰り返されます。そのため、REPEAT は常に、少なくとも 1 回はループに入ります。statement_list は、それぞれがセミコロン (;) ステートメント区切り文字で終了する 1 つ以上のステートメントで構成されます。

REPEAT ステートメントにはラベルを付けることができます。ラベルの使用に関連したルールについては、セクション13.6.2「ステートメントラベルの構文」を参照してください。

例:

```
mysql> delimiter //

mysql> CREATE PROCEDURE dorepeat(p1 INT)
    -> BEGIN
    ->   SET @x = 0;
    ->   REPEAT
    ->     SET @x = @x + 1;
    ->   UNTIL @x > p1 END REPEAT;
    -> END
    -> //
Query OK, 0 rows affected (0.00 sec)

mysql> CALL dorepeat(1000)//
Query OK, 0 rows affected (0.00 sec)

mysql> SELECT @x//
+------+
```

```
| @x   |
+------+
| 1001 |
+------+
1 row in set (0.00 sec)
```

13.6.5.7 RETURN 構文

```
RETURN expr
```

RETURN ステートメントはストアドファンクションの実行を終了し、そのストアドファンクションの呼び出し元に値 expr を返します。ストアドファンクション内には、少なくとも 1 つの RETURN ステートメントが存在する必要があります。そのストアドファンクションに複数の終了ポイントがある場合は、複数存在してもかまいません。

このステートメントは、ストアドプロシージャー、トリガー、またはイベントでは使用されません。LEAVE ステートメントを使用すると、これらのタイプのストアドプログラムを終了できます。

13.6.5.8 WHILE 構文

```
[begin_label:] WHILE search_condition DO
    statement_list
END WHILE [end_label]
```

WHILE ステートメント内のステートメントリストは、search_condition 式が true であるかぎり繰り返されます。statement_list は、それぞれがセミコロン (;) ステートメント区切り文字で終了する 1 つ以上の SQL ステートメントで構成されます。

WHILE ステートメントにはラベルを付けることができます。ラベルの使用に関連したルールについては、セクション13.6.2「ステートメントラベルの構文」を参照してください。

例:

```
CREATE PROCEDURE dowhile()
BEGIN
  DECLARE v1 INT DEFAULT 5;

  WHILE v1 > 0 DO
    ...
    SET v1 = v1 - 1;
  END WHILE;
END;
```

13.6.6 カーソル

MySQL は、ストアドプログラム内部のカーソルをサポートします。その構文は、組み込み SQL の場合と同様です。カーソルには次のプロパティーがあります。

- Asensitive: サーバーは、結果テーブルのコピーを作成する場合としない場合があります
- 読み取り専用: 更新できません
- スクロール不可: 1 方向にしかトラバースできず、行をスキップできません

カーソル宣言は、ハンドラ宣言の前で、かつ変数および条件宣言のあとに指定する必要があります。

例:

```
CREATE PROCEDURE curdemo()
BEGIN
  DECLARE done INT DEFAULT FALSE;
  DECLARE a CHAR(16);
  DECLARE b, c INT;
  DECLARE cur1 CURSOR FOR SELECT id,data FROM test.t1;
  DECLARE cur2 CURSOR FOR SELECT i FROM test.t2;
  DECLARE CONTINUE HANDLER FOR NOT FOUND SET done = TRUE;

  OPEN cur1;
  OPEN cur2;

  read_loop: LOOP
    FETCH cur1 INTO a, b;
    FETCH cur2 INTO c;
```

```
  IF done THEN
    LEAVE read_loop;
  END IF;
  IF b < c THEN
    INSERT INTO test.t3 VALUES (a,b);
  ELSE
    INSERT INTO test.t3 VALUES (a,c);
  END IF;
 END LOOP;

 CLOSE cur1;
 CLOSE cur2;
END;
```

13.6.6.1 カーソルの CLOSE 構文

```
CLOSE cursor_name
```

このステートメントは、以前に開かれたカーソルを閉じます。例については、セクション13.6.6「カーソル」を参照してください。

カーソルが開いていない場合は、エラーが発生します。

明示的に閉じられない場合は、そのカーソルが宣言された BEGIN ... END ブロックの最後に閉じられます。

13.6.6.2 カーソルの DECLARE 構文

```
DECLARE cursor_name CURSOR FOR select_statement
```

このステートメントはカーソルを宣言し、そのカーソルによってトラバースされる行を取得する SELECT ステートメントに関連付けます。行をあとでフェッチするには、FETCH ステートメントを使用します。SELECT ステートメントによって取得されるカラムの数が、FETCH ステートメントで指定された出力変数の数に一致している必要があります。

SELECT ステートメントに INTO 句を含めることはできません。

カーソル宣言は、ハンドラ宣言の前で、かつ変数および条件宣言のあとに指定する必要があります。

ストアドプログラムには複数のカーソル宣言を含めることができますが、特定のブロック内で宣言された各カーソルが一意の名前を持っている必要があります。例については、セクション13.6.6「カーソル」を参照してください。

SHOW ステートメントで入手できる情報については、多くの場合、INFORMATION_SCHEMA テーブルでカーソルを使用することによって同等の情報を取得できます。

13.6.6.3 カーソルの FETCH 構文

```
FETCH [[NEXT] FROM] cursor_name INTO var_name [, var_name] ...
```

このステートメントは、指定されたカーソル (これは開いている必要があります) に関連付けられた SELECT ステートメントの次の行をフェッチし、そのカーソルのポインタを進めます。行が存在する場合は、フェッチされたカラムが指定された変数に格納されます。SELECT ステートメントによって取得されるカラムの数が、FETCH ステートメントで指定された出力変数の数に一致している必要があります。

それ以上の行を取得できない場合は、SQLSTATE 値 '02000' で「データなし」状況が発生します。この状況を検出するには、その状況 (または、NOT FOUND 状況) のハンドラを設定できます。例については、セクション13.6.6「カーソル」を参照してください。

13.6.6.4 カーソルの OPEN 構文

```
OPEN cursor_name
```

このステートメントは、以前に宣言されたカーソルを開きます。例については、セクション13.6.6「カーソル」を参照してください。

13.6.7 条件の処理

条件は、現在のプログラムブロックの終了や実行の続行などの、特殊な処理が必要なストアドプログラムの実行中に発生する可能性があります。警告や例外などの一般的な条件、または特定のエラーコードなどの具体的な条

件に対してハンドラを定義できます。具体的な条件には名前を割り当てることができるため、ハンドラではその名前で参照できます。

条件に名前を付けるには、DECLARE ... CONDITION ステートメントを使用します。ハンドラを宣言するには、DECLARE ... HANDLER ステートメントを使用します。セクション13.6.7.1「DECLARE ... CONDITION 構文」およびセクション13.6.7.2「DECLARE ... HANDLER 構文」を参照してください。条件が発生したときにサーバーがハンドラを選択する方法については、セクション13.6.7.6「ハンドラのスコープに関するルール」を参照してください。

条件を発生させるには、SIGNAL ステートメントを使用します。条件ハンドラ内で条件情報を変更するには、RESIGNAL を使用します。セクション13.6.7.1「DECLARE ... CONDITION 構文」およびセクション13.6.7.2「DECLARE ... HANDLER 構文」を参照してください。

診断領域から情報を取得するには、GET DIAGNOSTICS ステートメントを使用します (セクション13.6.7.3「GET DIAGNOSTICS 構文」を参照してください)。診断領域については、セクション13.6.7.7「MySQL の診断領域」を参照してください。

13.6.7.1 DECLARE ... CONDITION 構文

```
DECLARE condition_name CONDITION FOR condition_value

condition_value:
    mysql_error_code
  | SQLSTATE [VALUE] sqlstate_value
```

DECLARE ... CONDITION ステートメントは名前付きエラー条件を宣言し、特定の処理が必要な条件に名前を関連付けます。この名前は、以降の DECLARE ... HANDLER ステートメントで参照できます (セクション13.6.7.2「DECLARE ... HANDLER 構文」を参照してください)。

条件宣言は、カーソルまたはハンドラ宣言の前に指定する必要があります。

DECLARE ... CONDITION の condition_value は、MySQL エラーコード (番号) または SQLSTATE 値 (5 文字の文字列リテラル) にすることができます。MySQL エラーコード 0 または '00' で始まる SQLSTATE 値は、エラー条件ではなく成功を示すため、使用すべきではありません。MySQL エラーコードおよび SQLSTATE 値のリストについては、セクションB.3「サーバーのエラーコードおよびメッセージ」を参照してください。

条件に名前を使用すると、ストアドプログラムのコードの明確化に役立つ場合があります。たとえば、次のハンドラは存在しないテーブルを削除しようとする試みに適用されますが、それが明らかなのは MySQL エラーコード 1051 の意味がわかっている場合だけです。

```
DECLARE CONTINUE HANDLER FOR 1051
 BEGIN
   -- body of handler
 END;
```

条件の名前を宣言することによって、このハンドラの目的がより簡単にわかるようになります。

```
DECLARE no_such_table CONDITION FOR 1051;
DECLARE CONTINUE HANDLER FOR no_such_table
 BEGIN
   -- body of handler
 END;
```

これは、同じ条件の、MySQL エラーコードではなく、対応する SQLSTATE 値に基づく名前付き条件です。

```
DECLARE no_such_table CONDITION FOR SQLSTATE '42S02';
DECLARE CONTINUE HANDLER FOR no_such_table
 BEGIN
   -- body of handler
 END;
```

SIGNAL で参照されるか、または RESIGNAL ステートメントで使用される条件名は MySQL エラーコードではなく、SQLSTATE 値に関連付けられている必要があります。

13.6.7.2 DECLARE ... HANDLER 構文

```
DECLARE handler_action HANDLER
    FOR condition_value [, condition_value] ...
    statement

handler_action:
```

```
  CONTINUE
| EXIT
| UNDO

condition_value:
  mysql_error_code
| SQLSTATE [VALUE] sqlstate_value
| condition_name
| SQLWARNING
| NOT FOUND
| SQLEXCEPTION
```

DECLARE ... HANDLER ステートメントは、1 つ以上の条件を処理するハンドラを指定します。これらの条件のいずれかが発生した場合は、指定された statement が実行されます。statement は SET var_name = value などの単純なステートメントでも、BEGIN と END を使用して記述された複合ステートメントでもかまいません (セクション13.6.1「BEGIN ... END 複合ステートメント構文」を参照してください)。

ハンドラ宣言は、変数または条件宣言のあとに指定する必要があります。

handler_action 値は、ハンドラステートメントの実行後にハンドラがどのようなアクションを実行するかを示します。

- CONTINUE: 現在のプログラムの実行が続行されます。
- EXIT: このハンドラが宣言されている BEGIN ... END 複合ステートメントの実行が終了します。これは、この条件が内側のブロックで発生した場合にも当てはまります。
- UNDO: サポートされていません。

DECLARE ... HANDLER の condition_value は、このハンドラをアクティブ化する具体的な条件または条件のクラスを示します。

- MySQL エラーコード (番号) または SQLSTATE 値 (5 文字の文字列リテラル)。MySQL エラーコード 0 または '00' で始まる SQLSTATE 値は、エラー条件ではなく成功を示すため、使用すべきではありません。MySQL エラーコードおよび SQLSTATE 値のリストについては、セクションB.3「サーバーのエラーコードおよびメッセージ」を参照してください。
- 以前に DECLARE ... CONDITION で指定された条件名。条件名は MySQL エラーコードまたは SQLSTATE 値に関連付けることができます。セクション13.6.7.1「DECLARE ... CONDITION 構文」を参照してください。
- SQLWARNING は、'01' で始まる SQLSTATE 値のクラスの短縮形です。
- NOT FOUND は、'02' で始まる SQLSTATE 値のクラスの短縮形です。これは、カーソルのコンテキストに関係しており、カーソルがデータセットの最後に達したときの動作を制御するために使用します。それ以上の行を取得できない場合は、SQLSTATE 値 '02000' で「データなし」状況が発生します。この状況を検出するには、その状況 (または、NOT FOUND 状況) のハンドラを設定できます。例については、セクション13.6.6「カーソル」を参照してください。この状況は、行が取得されない SELECT ... INTO var_list ステートメントでも発生します。
- SQLEXCEPTION は、'00'、'01'、または '02' で始まらない SQLSTATE 値のクラスの短縮形です。

条件が発生したときにサーバーがハンドラを選択する方法については、セクション13.6.7.6「ハンドラのスコープに関するルール」を参照してください。

対応するハンドラが宣言されていない条件が発生した場合、実行されるアクションはその条件のクラスによって異なります。

- SQLEXCEPTION 条件の場合は、EXIT ハンドラが存在するかのように、ストアドプログラムはその条件を発生させたステートメントで終了します。そのプログラムが別のストアドプログラムから呼び出されていた場合は、呼び出し元プログラムが、独自のハンドラに適用されるハンドラ選択ルールを使用してその条件を処理します。
- SQLWARNING 条件の場合は、CONTINUE ハンドラが存在するかのように、プログラムは実行を続行します。
- NOT FOUND 条件では、その条件が正常に発生した場合、アクションは CONTINUE です。SIGNAL または RESIGNAL によって発生した場合、アクションは EXIT です。

次の例では、重複キーエラーに対して発生する SQLSTATE '23000' のハンドラを使用します。

```
mysql> CREATE TABLE test.t (s1 INT, PRIMARY KEY (s1));
```

```
Query OK, 0 rows affected (0.00 sec)

mysql> delimiter //

mysql> CREATE PROCEDURE handlerdemo ()
    -> BEGIN
    ->   DECLARE CONTINUE HANDLER FOR SQLSTATE '23000' SET @x2 = 1;
    ->   SET @x = 1;
    ->   INSERT INTO test.t VALUES (1);
    ->   SET @x = 2;
    ->   INSERT INTO test.t VALUES (1);
    ->   SET @x = 3;
    -> END;
    -> //
Query OK, 0 rows affected (0.00 sec)

mysql> CALL handlerdemo()//
Query OK, 0 rows affected (0.00 sec)

mysql> SELECT @x//
   +------+
   | @x   |
   +------+
   | 3    |
   +------+
   1 row in set (0.00 sec)
```

このプロシージャーの実行後、@x が 3 になっていることを確認してください。これは、エラーが発生したあと、プロシージャーの最後まで実行が続行されたことを示しています。DECLARE ... HANDLER ステートメントが存在しなかったとすると、PRIMARY KEY 制約のために 2 番目の INSERT が失敗したあとに MySQL はデフォルトのアクション (EXIT) を実行するため、SELECT @x は 2 を返していました。

条件を無視するには、その条件の CONTINUE ハンドラを宣言し、それを空のブロックに関連付けます。例:

```
DECLARE CONTINUE HANDLER FOR SQLWARNING BEGIN END;
```

ブロックラベルのスコープには、そのブロック内で宣言されているハンドラのコードは含まれません。そのため、ハンドラに関連付けられたステートメントは、ITERATE または LEAVE を使用して、そのハンドラ宣言を囲むブロックのラベルを参照することができません。REPEAT ブロックに retry のラベルが含まれている次の例を考えてみます。

```
CREATE PROCEDURE p ()
BEGIN
 DECLARE i INT DEFAULT 3;
 retry:
   REPEAT
     BEGIN
       DECLARE CONTINUE HANDLER FOR SQLWARNING
         BEGIN
           ITERATE retry;    # illegal
         END;
       IF i < 0 THEN
         LEAVE retry;        # legal
       END IF;
       SET i = i - 1;
     END;
   UNTIL FALSE END REPEAT;
END;
```

retry ラベルは、そのブロック内の IF ステートメントのスコープ内にあります。CONTINUE ハンドラのスコープ内にはないため、そこでの参照は無効であり、エラーが発生します。

```
ERROR 1308 (42000): LEAVE with no matching label: retry
```

ハンドラ内の外側のラベルへの参照を回避するには、次の方法のいずれかを使用します。

- このブロックを離れるには、EXIT ハンドラを使用します。ブロックのクリーンアップが必要ない場合は、BEGIN ... END ハンドラ本体を空にすることができます。

  ```
  DECLARE EXIT HANDLER FOR SQLWARNING BEGIN END;
  ```

  そうでない場合は、ハンドラ本体内にクリーンアップステートメントを配置します。

  ```
  DECLARE EXIT HANDLER FOR SQLWARNING
    BEGIN
      block cleanup statements
  ```

```
 END;
```

- 実行を続行するには、CONTINUE ハンドラ内に、囲んでいるブロック内でチェックすることによってそのハンドラが呼び出されたかどうかを判定できるステータス変数を設定します。次の例では、この目的のために変数 done を使用します。

```
CREATE PROCEDURE p ()
BEGIN
  DECLARE i INT DEFAULT 3;
  DECLARE done INT DEFAULT FALSE;
  retry:
    REPEAT
      BEGIN
        DECLARE CONTINUE HANDLER FOR SQLWARNING
          BEGIN
            SET done = TRUE;
          END;
        IF done OR i < 0 THEN
          LEAVE retry;
        END IF;
        SET i = i - 1;
      END;
    UNTIL FALSE END REPEAT;
END;
```

13.6.7.3 GET DIAGNOSTICS 構文

```
GET [CURRENT] DIAGNOSTICS
{
    statement_information_item
    [, statement_information_item] ...
  | CONDITION condition_number
    condition_information_item
    [, condition_information_item] ...
}

statement_information_item:
    target = statement_information_item_name

condition_information_item:
    target = condition_information_item_name

statement_information_item_name:
    NUMBER
  | ROW_COUNT

condition_information_item_name:
    CLASS_ORIGIN
  | SUBCLASS_ORIGIN
  | RETURNED_SQLSTATE
  | MESSAGE_TEXT
  | MYSQL_ERRNO
  | CONSTRAINT_CATALOG
  | CONSTRAINT_SCHEMA
  | CONSTRAINT_NAME
  | CATALOG_NAME
  | SCHEMA_NAME
  | TABLE_NAME
  | COLUMN_NAME
  | CURSOR_NAME

condition_number, target:
    (see following discussion)
```

SQL ステートメントは、診断領域を移入する診断情報を生成します。GET DIAGNOSTICS ステートメントを使用すると、アプリケーションでこの情報を検査できます。これは、MySQL 5.6.4 の時点で使用できます。(SHOW WARNINGS または SHOW ERRORS を使用して、条件またはエラーを確認することもできます。)

GET DIAGNOSTICS を実行するために特殊な権限は必要ありません。

キーワード CURRENT は、現在の診断領域から情報を取得することを示します。MySQL では、それがデフォルトの動作であるため、これは何の効果もありません。

GET DIAGNOSTICS は通常、ストアドプログラム内のハンドラで使用されますが、これは、任意の SQL ステートメントの実行をチェックするためにハンドラのコンテキストの外部で許可される MySQL 拡張です。たとえば、mysql クライアントプログラムを呼び出す場合は、プロンプトで次のステートメントを入力できます。

```
mysql> DROP TABLE test.no_such_table;
ERROR 1051 (42S02): Unknown table 'test.no_such_table'
mysql> GET DIAGNOSTICS CONDITION 1
    ->   @p1 = RETURNED_SQLSTATE, @p2 = MESSAGE_TEXT;
mysql> SELECT @p1, @p2;
+-------+--------------------------------------+
| @p1   | @p2                                  |
+-------+--------------------------------------+
| 42S02 | Unknown table 'test.no_such_table' |
+-------+--------------------------------------+
```

診断領域については、セクション13.6.7.7「MySQL の診断領域」を参照してください。簡単に言うと、ここには次の 2 種類の情報が含まれています。

- 発生した条件の数や、影響を受けた行数などのステートメント情報。
- エラーコードやメッセージなどの条件情報。ステートメントが複数の条件を発生させた場合、診断領域のこの部分には条件ごとの条件領域が含まれています。ステートメントがどの条件も発生させない場合、診断領域のこの部分は空です。

3 つの条件を生成するステートメントの場合、診断領域には、次のようなステートメント情報と条件情報が含まれています。

```
Statement information:
  row count
  ... other statement information items ...
Condition area list:
  Condition area 1:
    error code for condition 1
    error message for condition 1
    ... other condition information items ...
  Condition area 2:
    error code for condition 2:
    error message for condition 2
    ... other condition information items ...
  Condition area 3:
    error code for condition 3
    error message for condition 3
    ... other condition information items ...
```

GET DIAGNOSTICS はステートメントまたは条件情報のどちらかを取得できますが、同じステートメントで両方を取得することはできません。

- ステートメント情報を取得するには、目的のステートメント項目をターゲット変数に取得します。GET DIAGNOSTICS の次の例では、使用可能な条件の数と影響を受けた行数をユーザー変数 @p1 と @p2 に割り当てます。

  ```
  GET DIAGNOSTICS @p1 = NUMBER, @p2 = ROW_COUNT;
  ```

- 条件情報を取得するには、条件番号を指定し、目的の条件項目をターゲット変数に取得します。GET DIAGNOSTICS の次の例では、SQLSTATE 値とエラーメッセージをユーザー変数 @p3 と @p4 に割り当てます。

  ```
  GET DIAGNOSTICS CONDITION 1
    @p3 = RETURNED_SQLSTATE, @p4 = MESSAGE_TEXT;
  ```

取得リストには、カンマで区切られた 1 つ以上の target = item_name 代入を指定します。各代入では、ターゲット変数と、このステートメントがステートメントまたは条件情報のどちらを取得するかに応じて statement_information_item_name または condition_information_item_name 指示子のどちらかを指定します。

項目情報を格納するための有効な target 指示子は、ストアドプロシージャーやストアドファンクションのパラメータ、DECLARE で宣言されたストアドプログラムのローカル変数、ユーザー定義変数のいずれかです。

有効な condition_number 指示子は、ストアドプロシージャーやストアドファンクションのパラメータ、DECLARE で宣言されたストアドプログラムのローカル変数、ユーザー定義変数、システム変数、リテラルのいずれかです。文字リテラルには、_charset イントロデューサを含めることができます。条件番号が 1 から、情報が含まれている条件領域の数までの範囲にない場合は、警告が発生します。この場合、この警告は、診断領域にその領域をクリアすることなく追加されます。

現在は、条件が発生したときに、MySQL が GET DIAGNOSTICS によって認識されるすべての条件項目を移入するわけではありません。例:

```
mysql> GET DIAGNOSTICS CONDITION 1
```

```
   ->   @p5 = SCHEMA_NAME, @p6 = TABLE_NAME;
mysql> SELECT @p5, @p6;
+------+------+
| @p5  | @p6  |
+------+------+
|      |      |
+------+------+
```

標準 SQL では、複数の条件が存在する場合、最初の条件は前の SQL ステートメントに対して返された SQLSTATE 値に関連しています。MySQL では、これが保証されません。メインのエラーを取得するために、次のように行うことはできません。

```
GET DIAGNOSTICS CONDITION 1 @errno = MYSQL_ERRNO;
```

代わりに、まず条件数を取得し、次にそれを使用してどの条件番号を検査するかを指定します。

```
GET DIAGNOSTICS @cno = NUMBER;
GET DIAGNOSTICS CONDITION @cno @errno = MYSQL_ERRNO;
```

許可されるステートメント情報と条件情報の項目、および条件が発生したときにどの項目が移入されるかについては、診断領域の情報項目を参照してください。

ストアドプロシージャーのコンテキストで GET DIAGNOSTICS と例外ハンドラを使用して、挿入操作の結果を評価する例を次に示します。挿入が成功した場合、このプロシージャーは GET DIAGNOSTICS を使用して、影響を受けた行数を取得します。これは、診断領域がクリアされていないかぎり、GET DIAGNOSTICS を複数回使用してステートメントに関する情報を取得できることを示しています。

```
CREATE PROCEDURE do_insert(value INT)
BEGIN
  -- Declare variables to hold diagnostics area information
  DECLARE code CHAR(5) DEFAULT '00000';
  DECLARE msg TEXT;
  DECLARE rows INT;
  DECLARE result TEXT;
  -- Declare exception handler for failed insert
  DECLARE CONTINUE HANDLER FOR SQLEXCEPTION
    BEGIN
      GET DIAGNOSTICS CONDITION 1
        code = RETURNED_SQLSTATE, msg = MESSAGE_TEXT;
    END;

  -- Perform the insert
  INSERT INTO t1 (int_col) VALUES(value);
  -- Check whether the insert was successful
  IF code = '00000' THEN
    GET DIAGNOSTICS rows = ROW_COUNT;
    SET result = CONCAT('insert succeeded, row count = ',rows);
  ELSE
    SET result = CONCAT('insert failed, error = ',code,', message = ',msg);
  END IF;
  -- Say what happened
  SELECT result;
END;
```

t1.int_col が、NOT NULL として宣言された整数カラムであるとします。このプロシージャーは、NULL 以外の値と NULL 値を挿入するために呼び出されると、それぞれ次の結果を生成します。

```
mysql> CALL do_insert(1);
+---------------------------------+
| result                          |
+---------------------------------+
| insert succeeded, row count = 1 |
+---------------------------------+

mysql> CALL do_insert(NULL);
+-------------------------------------------------------------------------+
| result                                                                  |
+-------------------------------------------------------------------------+
| insert failed, error = 23000, message = Column 'int_col' cannot be null |
+-------------------------------------------------------------------------+
```

GET DIAGNOSTICS は、条件ハンドラ内で、診断領域をクリアしたり、そのハンドラをアクティブ化した条件に関する情報が失われたりする可能性のあるほかのステートメントの前に使用するようにしてください。診断領域がいつ設定およびクリアされるかについては、セクション13.6.7.7「MySQL の診断領域」を参照してください。

13.6.7.4 RESIGNAL 構文

```
RESIGNAL [condition_value]
    [SET signal_information_item
    [, signal_information_item] ...]

condition_value:
    SQLSTATE [VALUE] sqlstate_value
  | condition_name

signal_information_item:
    condition_information_item_name = simple_value_specification

condition_information_item_name:
    CLASS_ORIGIN
  | SUBCLASS_ORIGIN
  | MESSAGE_TEXT
  | MYSQL_ERRNO
  | CONSTRAINT_CATALOG
  | CONSTRAINT_SCHEMA
  | CONSTRAINT_NAME
  | CATALOG_NAME
  | SCHEMA_NAME
  | TABLE_NAME
  | COLUMN_NAME
  | CURSOR_NAME

condition_name, simple_value_specification:
    (see following discussion)
```

RESIGNAL は、ストアドプロシージャーやストアドファンクションの内部にある複合ステートメント内の条件ハンドラ、トリガー、またはイベントの実行中に使用可能なエラー条件情報を渡します。RESIGNAL は、その情報の一部またはすべてを、渡す前に変更する可能性があります。RESIGNAL は SIGNAL に関連していますが、SIGNAL のように条件を発信する代わりに、RESIGNAL は既存の条件情報を (おそらく、変更してから) 中継します。

RESIGNAL は、エラーを処理することと、エラー情報を返すことの両方を可能にします。それ以外の場合は、ハンドラ内の SQL ステートメントを実行することによって、そのハンドラのアクティブ化を発生させた情報が破棄されます。RESIGNAL は、指定されたハンドラが状況の一部を処理できる場合はプロシージャーの一部を短くしてから、条件を「遡って」別のハンドラに渡すことができます。

RESIGNAL ステートメントを実行するために特殊な権限は必要ありません。

RESIGNAL のすべての形式で、現在のコンテキストが条件ハンドラである必要があります。そうでない場合、RESIGNAL は不正であり、RESIGNAL when handler not active エラーが発生します。

診断領域から情報を取得するには、GET DIAGNOSTICS ステートメントを使用します (セクション13.6.7.3「GET DIAGNOSTICS 構文」を参照してください)。診断領域については、セクション13.6.7.7「MySQL の診断領域」を参照してください。

RESIGNAL に対する condition_value と signal_information_item の定義やルールは、SIGNAL に対するものと同じです。たとえば、condition_value を SQLSTATE 値にすることができ、この値は、エラー、警告、または「見つからない」を示す場合があります。詳細は、セクション13.6.7.5「SIGNAL 構文」を参照してください。

RESIGNAL ステートメントは、どちらもオプションである condition_value と SET 句を受け取ります。このため、次のいくつかの使用法が考えられます。

- RESIGNAL のみ:

```
RESIGNAL;
```

- 新しいシグナル情報を含む RESIGNAL:

```
RESIGNAL SET signal_information_item [, signal_information_item] ...;
```

- 条件値と場合によっては新しいシグナル情報を含む RESIGNAL:

```
RESIGNAL condition_value
    [SET signal_information_item [, signal_information_item] ...];
```

これらのユースケースはすべて、診断および条件領域の変更を発生させます。

- 診断領域には 1 つ以上の条件領域が含まれています。
- 条件領域には、SQLSTATE 値、MYSQL_ERRNO、MESSAGE_TEXT などの条件情報項目が含まれています。

診断領域内の条件領域の最大数は、max_error_count システム変数の値によって決定されます。診断領域関連のシステム変数を参照してください。

RESIGNAL のみ

単純な RESIGNAL のみとは、「エラーを変更せずに渡す」ことを示します。これは最後の診断領域をリストアし、それを現在の診断領域にします。つまり、診断領域スタックを「ポップします」。

条件をキャッチする条件ハンドラ内での RESIGNAL のみの 1 つの使用法として、ほかのいくつかのアクションを実行したあと、元の条件情報 (ハンドラに入る前に存在していた情報) を変更せずに渡す方法があります。

例:

```
DROP TABLE IF EXISTS xx;
delimiter //
CREATE PROCEDURE p ()
BEGIN
  DECLARE EXIT HANDLER FOR SQLEXCEPTION
  BEGIN
    SET @error_count = @error_count + 1;
    IF @a = 0 THEN RESIGNAL; END IF;
  END;
  DROP TABLE xx;
END//
delimiter ;
SET @error_count = 0;
SET @a = 0;
CALL p();
```

DROP TABLE xx ステートメントが失敗したとします。診断領域スタックは次のようになります。

```
DA 1. ERROR 1051 (42S02): Unknown table 'xx'
```

次に、実行が EXIT ハンドラに入ります。このハンドラは最初に、診断領域をスタックの先頭にプッシュします。これで、診断領域スタックは次のようになります。

```
DA 1. ERROR 1051 (42S02): Unknown table 'xx'
DA 2. ERROR 1051 (42S02): Unknown table 'xx'
```

この時点で、最初の (現在の) 診断領域と 2 番目の (スタックされた) 診断領域の内容は同じです。最初の診断領域は、そのあとにハンドラ内で実行されるステートメントによって変更される可能性があります。

通常、プロシージャーステートメントは最初の診断領域をクリアします。BEGIN は例外です。これはクリアせず、何も行いません。SET は例外ではありません。これはクリアし、操作を実行して、「成功」の結果を生成します。これで、診断領域スタックは次のようになります。

```
DA 1. ERROR 0000 (00000): Successful operation
DA 2. ERROR 1051 (42S02): Unknown table 'xx'
```

この時点で、@a = 0 の場合、RESIGNAL は診断領域スタックをポップします。これで、診断領域スタックは次のようになります。

```
DA 1. ERROR 1051 (42S02): Unknown table 'xx'
```

そして、これが呼び出し元に表示される内容です。

@a が 0 でない場合、このハンドラは単純に終了します。つまり、現在の診断領域はそれ以上使用されなくなる (「処理済み」になった) ため、破棄することが可能になり、スタックされた診断領域がふたたび現在の診断領域になります。診断領域スタックは次のようになります。

```
DA 1. ERROR 0000 (00000): Successful operation
```

詳細を調べると複雑に見えますが、最終結果はきわめて有効です。ハンドラは、そのハンドラのアクティブ化を発生させた条件に関する情報を破棄することなく実行できます。

新しいシグナル情報を含む RESIGNAL

SET 句を含む RESIGNAL は新しいシグナル情報を指定するため、このステートメントは、「エラーを変更してから渡す」ことを示します。

```
RESIGNAL SET signal_information_item [, signal_information_item] ...;
```

RESIGNAL のみと同様に、考え方としては、元の情報が消えるように診断領域スタックをポップします。RESIGNAL のみとは異なり、SET 句で指定されたものはすべて変更されます。

例:

```
DROP TABLE IF EXISTS xx;
delimiter //
CREATE PROCEDURE p ()
BEGIN
 DECLARE EXIT HANDLER FOR SQLEXCEPTION
 BEGIN
   SET @error_count = @error_count + 1;
   IF @a = 0 THEN RESIGNAL SET MYSQL_ERRNO = 5; END IF;
 END;
 DROP TABLE xx;
END//
delimiter ;
SET @error_count = 0;
SET @a = 0;
CALL p();
```

前の説明から、RESIGNAL のみによって、診断領域スタックが次のようになることを思い出してください。

```
DA 1. ERROR 1051 (42S02): Unknown table 'xx'
```

RESIGNAL SET MYSQL_ERRNO = 5 ステートメントでは、代わりに、スタックが次のようになります。これが呼び出し元に表示される内容です。

```
DA 1. ERROR 5 (42S02): Unknown table 'xx'
```

つまり、エラー番号だけが変更され、ほかは何も変更されません。

RESIGNAL ステートメントはシグナル情報項目のいずれかまたはすべてを変更できるため、診断領域の最初の条件領域がまったく異なっているように見えます。

条件値とオプションの新しいシグナル情報を含む RESIGNAL

条件値を含む RESIGNAL は、「条件を現在の診断領域にプッシュする」ことを示します。SET 句が存在する場合は、エラー情報も変更されます。

```
RESIGNAL condition_value
   [SET signal_information_item [, signal_information_item] ...];
```

この形式の RESIGNAL は最後の診断領域をリストアし、それを現在の診断領域にします。つまり、診断領域スタックを「ポップします」。これは、単純な RESIGNAL のみが行う動作と同じです。ただし、条件値またはシグナル情報に応じて、診断領域も変更されます。

例:

```
DROP TABLE IF EXISTS xx;
delimiter //
CREATE PROCEDURE p ()
BEGIN
 DECLARE EXIT HANDLER FOR SQLEXCEPTION
 BEGIN
   SET @error_count = @error_count + 1;
   IF @a = 0 THEN RESIGNAL SQLSTATE '45000' SET MYSQL_ERRNO=5; END IF;
 END;
 DROP TABLE xx;
END//
delimiter ;
SET @error_count = 0;
SET @a = 0;
SET @@max_error_count = 2;
CALL p();
SHOW ERRORS;
```

これは前の例に似ていて、その効果も同じですが、RESIGNAL が発生した場合は、現在の条件領域が最後には異なっているように見える点が異なります。(この条件が既存の条件を置き換えるのではなく、それに追加される理由は、条件値が使用されているためです。)

この RESIGNAL ステートメントには条件値 (SQLSTATE '45000') が含まれているため、新しい条件領域が追加され、診断領域スタックは次のようになります。

```
DA 1. (condition 2) ERROR 1051 (42S02): Unknown table 'xx'
     (condition 1) ERROR 5 (45000) Unknown table 'xx'
```

この例での CALL p() と SHOW ERRORS の結果は次のとおりです。

```
mysql> CALL p();
ERROR 5 (45000): Unknown table 'xx'
mysql> SHOW ERRORS;
+-------+------+---------------------------------+
| Level | Code | Message                         |
+-------+------+---------------------------------+
| Error | 1051 | Unknown table 'xx'              |
| Error |    5 | Unknown table 'xx'              |
+-------+------+---------------------------------+
```

RESIGNAL には条件ハンドラのコンテキストが必要

RESIGNAL のすべての形式で、現在のコンテキストが条件ハンドラである必要があります。そうでない場合、RESIGNAL は不正であり、RESIGNAL when handler not active エラーが発生します。例:

```
mysql> CREATE PROCEDURE p () RESIGNAL;
Query OK, 0 rows affected (0.00 sec)

mysql> CALL p();
ERROR 1645 (0K000): RESIGNAL when handler not active
```

さらに難しい例を次に示します。

```
delimiter //
CREATE FUNCTION f () RETURNS INT
BEGIN
  RESIGNAL;
  RETURN 5;
END//
CREATE PROCEDURE p ()
BEGIN
  DECLARE EXIT HANDLER FOR SQLEXCEPTION SET @a=f();
  SIGNAL SQLSTATE '55555';
END//
delimiter ;
CALL p();
```

RESIGNAL がストアドファンクション f() 内に現れています。f() 自体は EXIT ハンドラのコンテキスト内で呼び出されますが、f() 内での実行は独自のコンテキストを持ち、それはハンドラのコンテキストではありません。そのため、f() 内での RESIGNAL によって、「handler not active」 エラーが発生します。

MySQL 5.5 では、ハンドラのスコープに関するルールが十分に発達していません。f() はハンドラのコンテキスト内で実行されると見なされ、f() 内での RESIGNAL は正当です。

13.6.7.5 SIGNAL 構文

```
SIGNAL condition_value
    [SET signal_information_item
    [, signal_information_item] ...]

condition_value:
    SQLSTATE [VALUE] sqlstate_value
  | condition_name

signal_information_item:
    condition_information_item_name = simple_value_specification

condition_information_item_name:
    CLASS_ORIGIN
  | SUBCLASS_ORIGIN
  | MESSAGE_TEXT
  | MYSQL_ERRNO
  | CONSTRAINT_CATALOG
  | CONSTRAINT_SCHEMA
  | CONSTRAINT_NAME
  | CATALOG_NAME
  | SCHEMA_NAME
  | TABLE_NAME
  | COLUMN_NAME
  | CURSOR_NAME
```

```
condition_name, simple_value_specification:
    (see following discussion)
```

SIGNAL は、エラー「を返す」ための方法です。SIGNAL は、ハンドラ、アプリケーションの外側の部分、またはクライアントにエラー情報を提供します。また、エラーの特性 (エラー番号、SQLSTATE 値、メッセージ) に対する制御も提供します。SIGNAL を使用しない場合は、存在しないテーブルを意図的に参照してルーチンがエラーを返すようにする、などの回避方法に頼る必要があります。

SIGNAL ステートメントを実行するために特殊な権限は必要ありません。

診断領域から情報を取得するには、GET DIAGNOSTICS ステートメントを使用します (セクション13.6.7.3「GET DIAGNOSTICS 構文」を参照してください)。診断領域については、セクション13.6.7.7「MySQL の診断領域」を参照してください。

SIGNAL ステートメント内の condition_value は、返されるエラー値を示します。これは、SQLSTATE 値 (5 文字の文字列リテラル) か、または以前に DECLARE ... CONDITION で定義された名前付き条件を参照する condition_name にすることができます (セクション13.6.7.1「DECLARE ... CONDITION 構文」を参照してください)。

SQLSTATE 値は、エラー、警告、または「見つからない」を示す場合があります。シグナルの条件情報項目で説明されているように、この値の最初の 2 文字はそのエラークラスを示します。一部のシグナル値はステートメントを終了させます。ハンドラ、カーソル、およびステートメントに対するシグナルの影響を参照してください。

'00' は成功を示し、エラーの通知には有効でないため、SIGNAL ステートメントの SQLSTATE 値をこのような値で始めるべきではありません。これは、SQLSTATE 値が SIGNAL ステートメントで直接、またはこのステートメントで参照されている名前付き条件のどちらで指定されている場合にも当てはまります。この値が無効である場合は、Bad SQLSTATE エラーが発生します。

一般的な SQLSTATE 値を通知するには、'45000' を使用します。これは、「未処理のユーザー定義の例外」を示します。

SIGNAL ステートメントには、オプションで、複数のシグナル項目が condition_information_item_name = simple_value_specification 代入のカンマ区切りリストで含まれている SET 句が含まれます。

各 condition_information_item_name は、SET 句内で 1 回だけ指定できます。そうでない場合は、Duplicate condition information item エラーが発生します。

有効な simple_value_specification 指示子は、ストアドプロシージャーやストアドファンクションのパラメータ、DECLARE で宣言されたストアドプログラムのローカル変数、ユーザー定義変数、システム変数、またはリテラルを使用して指定できます。文字リテラルには、_charset イントロデューサを含めることができます。

許可される condition_information_item_name 値については、シグナルの条件情報項目を参照してください。

次のプロシージャーは、その入力パラメータである pval の値に応じて、エラーまたは警告を通知します。

```
CREATE PROCEDURE p (pval INT)
BEGIN
 DECLARE specialty CONDITION FOR SQLSTATE '45000';
 IF pval = 0 THEN
   SIGNAL SQLSTATE '01000';
 ELSEIF pval = 1 THEN
   SIGNAL SQLSTATE '45000'
     SET MESSAGE_TEXT = 'An error occurred';
 ELSEIF pval = 2 THEN
   SIGNAL specialty
     SET MESSAGE_TEXT = 'An error occurred';
 ELSE
   SIGNAL SQLSTATE '01000'
     SET MESSAGE_TEXT = 'A warning occurred', MYSQL_ERRNO = 1000;
   SIGNAL SQLSTATE '45000'
     SET MESSAGE_TEXT = 'An error occurred', MYSQL_ERRNO = 1001;
 END IF;
END;
```

pval が 0 である場合、'01' で始まる SQLSTATE 値は警告クラス内のシグナルであるため、p() は警告を通知します。この警告はプロシージャーを終了させず、そのプロシージャーから戻ったあとに SHOW WARNINGS で確認できます。

pval が 1 である場合、p() はエラーを通知し、MESSAGE_TEXT 条件情報項目を設定します。このエラーはプロシージャーを終了させ、そのテキストがエラー情報とともに返されます。

pval が 2 である場合、この場合は名前付き条件を使用して SQLSTATE 値が指定されているにもかかわらず、同じエラーが通知されます。

pval がその他の任意の値である場合、p() は最初に警告を通知し、メッセージテキストとエラー番号の条件情報項目を設定します。この警告はプロシージャーを終了させないため、実行が続行され、そのあと p() はエラーを通知します。このエラーはプロシージャーを終了させます。警告によって設定されたメッセージテキストとエラー番号は、エラーによって設定された値によって置き換えられ、それがエラー情報とともに返されます。

SIGNAL は通常、ストアドプログラム内で使用されますが、これはハンドラのコンテキストの外部で許可される MySQL 拡張です。たとえば、mysql クライアントプログラムを呼び出す場合は、プロンプトで次のステートメントのいずれかを入力できます。

```
mysql> SIGNAL SQLSTATE '77777';
mysql> CREATE TRIGGER t_bi BEFORE INSERT ON t
    -> FOR EACH ROW SIGNAL SQLSTATE '77777';
mysql> CREATE EVENT e ON SCHEDULE EVERY 1 SECOND
    -> DO SIGNAL SQLSTATE '77777';
```

SIGNAL は、次のルールに従って実行されます。

SIGNAL ステートメントが特定の SQLSTATE 値を示している場合、その値は、指定された条件を通知するために使用されます。例:

```
CREATE PROCEDURE p (divisor INT)
BEGIN
  IF divisor = 0 THEN
    SIGNAL SQLSTATE '22012';
  END IF;
END;
```

SIGNAL ステートメントが名前付き条件を使用している場合、その条件は SIGNAL ステートメントに適用される何らかのスコープ内で宣言される必要があり、また MySQL エラー番号ではなく SQLSTATE 値を使用して定義される必要があります。例:

```
CREATE PROCEDURE p (divisor INT)
BEGIN
  DECLARE divide_by_zero CONDITION FOR SQLSTATE '22012';
  IF divisor = 0 THEN
    SIGNAL divide_by_zero;
  END IF;
END;
```

その名前付き条件が SIGNAL ステートメントのスコープ内に存在しない場合は、Undefined CONDITION エラーが発生します。

SIGNAL が、SQLSTATE 値ではなく MySQL エラー番号で定義された名前付き条件を参照している場合は、SIGNAL/RESIGNAL can only use a CONDITION defined with SQLSTATE エラーが発生します。次のステートメントを発行すると、名前付き条件が MySQL エラー番号に関連付けられているため、そのエラーが発生します。

```
DECLARE no_such_table CONDITION FOR 1051;
SIGNAL no_such_table;
```

特定の名前を持つ条件が異なるスコープで複数回宣言されている場合は、もっともローカルなスコープを持つ宣言が適用されます。次のプロシージャーを考えてみます。

```
CREATE PROCEDURE p (divisor INT)
BEGIN
  DECLARE my_error CONDITION FOR SQLSTATE '45000';
  IF divisor = 0 THEN
    BEGIN
      DECLARE my_error CONDITION FOR SQLSTATE '22012';
      SIGNAL my_error;
    END;
  END IF;
  SIGNAL my_error;
END;
```

divisor が 0 である場合は、最初の SIGNAL ステートメントが実行されます。もっとも内側の my_error 条件宣言が適用され、SQLSTATE '22012' が発生します。

divisor が 0 でない場合は、2 番目の SIGNAL ステートメントが実行されます。もっとも外側の my_error 条件宣言が適用され、SQLSTATE '45000' が発生します。

条件が発生したときにサーバーがハンドラを選択する方法については、セクション13.6.7.6「ハンドラのスコープに関するルール」を参照してください。

シグナルが例外ハンドラ内で発生する場合があります。

```
CREATE PROCEDURE p ()
BEGIN
 DECLARE EXIT HANDLER FOR SQLEXCEPTION
 BEGIN
   SIGNAL SQLSTATE VALUE '99999'
     SET MESSAGE_TEXT = 'An error occurred';
 END;
 DROP TABLE no_such_table;
END;
```

CALL p() が DROP TABLE ステートメントに達します。no_such_table という名前のテーブルが存在しないため、エラーハンドラがアクティブ化されます。エラーハンドラは元のエラー (「このようなテーブルがない」) を破棄し、SQLSTATE '99999' の新しいエラーとメッセージ An error occurred を作成します。

シグナルの条件情報項目

次の表は、SIGNAL (または RESIGNAL) ステートメントで設定できる診断領域条件情報項目の名前を一覧表示しています。MySQL 拡張である MYSQL_ERRNO を除き、すべての項目が標準 SQL です。これらの項目の詳細は、セクション13.6.7.7「MySQL の診断領域」を参照してください。

```
Item Name            Definition
---------            ----------
CLASS_ORIGIN         VARCHAR(64)
SUBCLASS_ORIGIN      VARCHAR(64)
CONSTRAINT_CATALOG   VARCHAR(64)
CONSTRAINT_SCHEMA    VARCHAR(64)
CONSTRAINT_NAME      VARCHAR(64)
CATALOG_NAME         VARCHAR(64)
SCHEMA_NAME          VARCHAR(64)
TABLE_NAME           VARCHAR(64)
COLUMN_NAME          VARCHAR(64)
CURSOR_NAME          VARCHAR(64)
MESSAGE_TEXT         VARCHAR(128)
MYSQL_ERRNO          SMALLINT UNSIGNED
```

文字項目の文字セットは UTF-8 です。

SIGNAL ステートメント内で条件情報項目に NULL を割り当てることはできません。

SIGNAL ステートメントは直接にか、または SQLSTATE 値で定義された名前付き条件を参照することによって間接的にかにかかわらず、常に SQLSTATE 値を指定します。SQLSTATE 値の最初の 2 文字はそのクラスであり、このクラスによって、その条件情報項目のデフォルト値が決定されます。

- クラス = '00' (成功)

  不正です。'00' で始まる SQLSTATE 値は成功を示すため、SIGNAL には有効でありません。

- クラス = '01' (警告)

  ```
  MESSAGE_TEXT = 'Unhandled user-defined warning condition';
  MYSQL_ERRNO = ER_SIGNAL_WARN
  ```

- クラス = '02' (見つからない)

  ```
  MESSAGE_TEXT = 'Unhandled user-defined not found condition';
  MYSQL_ERRNO = ER_SIGNAL_NOT_FOUND
  ```

- クラス > '02' (例外)

  ```
  MESSAGE_TEXT = 'Unhandled user-defined exception condition';
  MYSQL_ERRNO = ER_SIGNAL_EXCEPTION
  ```

正当なクラスの場合は、その他の条件情報項目が次のように設定されます。

```
CLASS_ORIGIN = SUBCLASS_ORIGIN = '';
CONSTRAINT_CATALOG = CONSTRAINT_SCHEMA = CONSTRAINT_NAME = '';
CATALOG_NAME = SCHEMA_NAME = TABLE_NAME = COLUMN_NAME = '';
CURSOR_NAME = '';
```

SIGNAL が実行されたあとにアクセスできるエラー値は、SIGNAL ステートメントによって発生した SQLSTATE 値と、MESSAGE_TEXT および MYSQL_ERRNO 項目です。これらの値は、次の C API から取得できます。

- SQLSTATE 値: mysql_sqlstate() の呼び出し
- MYSQL_ERRNO 値: mysql_errno() の呼び出し
- MESSAGE_TEXT 値: mysql_error() の呼び出し

SQL からは、SHOW WARNINGS と SHOW ERRORS からの出力の Code および Message カラムに MYSQL_ERRNO および MESSAGE_TEXT 値が示されます。

診断領域から情報を取得するには、GET DIAGNOSTICS ステートメントを使用します (セクション13.6.7.3「GET DIAGNOSTICS 構文」を参照してください)。診断領域については、セクション13.6.7.7「MySQL の診断領域」を参照してください。

ハンドラ、カーソル、およびステートメントに対するシグナルの影響

ステートメントの実行に対するシグナルの影響は、そのシグナルのクラスによって異なります。このクラスによって、エラーの重大性が決定されます。MySQL は、sql_mode システム変数の値を無視します。特に、厳密な SQL モードは問題になりません。MySQL は IGNORE も無視します。SIGNAL の目的はユーザーが生成した エラーを明示的に発生させることであるため、シグナルが無視されることはありません。

次の説明で、「未処理」は、通知された SQLSTATE 値に対するハンドラが DECLARE ... HANDLER で定義されていないことを示します。

- クラス = '00' (成功)

  不正です。'00' で始まる SQLSTATE 値は成功を示すため、SIGNAL には有効でありません。

- クラス = '01' (警告)

  warning_count システム変数の値が増やされます。SHOW WARNINGS がシグナルを示します。SQLWARNING ハンドラがシグナルをキャッチします。このシグナルが関数内で未処理である場合、ステートメントは終了しません。

- クラス = '02' (見つからない)

  NOT FOUND ハンドラがシグナルをキャッチします。カーソルには影響しません。このシグナルが関数内で未処理である場合、ステートメントは終了します。

- クラス > '02' (例外)

  SQLEXCEPTION ハンドラがシグナルをキャッチします。このシグナルが関数内で未処理である場合、ステートメントは終了します。

- クラス = '40'

  通常の例外として処理されます。

例:

```
mysql> delimiter //
mysql> CREATE FUNCTION f () RETURNS INT
    -> BEGIN
    ->   SIGNAL SQLSTATE '01234';  -- signal a warning
    ->   RETURN 5;
    -> END//
mysql> delimiter ;
mysql> CREATE TABLE t (s1 INT);
mysql> INSERT INTO t VALUES (f());
```

結果として、5 を含む行がテーブル t に挿入されます。通知された警告は、SHOW WARNINGS で表示できます。

13.6.7.6 ハンドラのスコープに関するルール

ストアドプログラムには、そのプログラム内で特定の条件が発生したときに呼び出されるハンドラを含めることができます。各ハンドラの適用性は、プログラム定義の中でのそのハンドラの場所や、そのハンドラが処理する 1 つまたは複数の条件によって異なります。

- BEGIN ... END ブロック内で宣言されたハンドラは、そのブロック内でハンドラ宣言のあとにある SQL ステートメントのスコープ内にしかありません。そのハンドラ自体が条件を発生させた場合は、そのハンドラも、そのブロック内で宣言されているほかのどのハンドラもその条件を処理できません。次の例では、ハンドラ H1 と H2 は、ステートメント stmt1 および stmt2 によって発生した条件のスコープ内にあります。ただし、H1 も H2 も、H1 または H2 の本体で発生した条件のスコープ内にはありません。

```
BEGIN -- outer block
  DECLARE EXIT HANDLER FOR ...;  -- handler H1
  DECLARE EXIT HANDLER FOR ...;  -- handler H2
  stmt1;
  stmt2;
END;
```

- ハンドラは、それが宣言されているブロックのスコープ内にしかなく、そのブロックの外部で発生した条件に対してアクティブ化することはできません。次の例では、ハンドラ H1 は内側のブロックにある stmt1 のスコープ内にありますが、外側のブロックにある stmt2 のスコープ内にはありません。

```
BEGIN -- outer block
  BEGIN -- inner block
    DECLARE EXIT HANDLER FOR ...;  -- handler H1
    stmt1;
  END;
  stmt2;
END;
```

- ハンドラは、特定のハンドラまたは一般的なハンドラのどちらかです。特定のハンドラとは、MySQL エラーコード、SQLSTATE 値、または条件名を処理するためのものです。一般的なハンドラとは、SQLWARNING、SQLEXCEPTION、または NOT FOUND クラス内の条件を処理するためのものです。あとで説明されているように、条件の特定性は条件の優先順位に関連しています。

複数のハンドラを異なるスコープ内で、かつ異なる特定性で宣言できます。たとえば、外側のブロックには特定の MySQL エラーコードハンドラが、また内側のブロックには一般的な SQLWARNING ハンドラが存在する可能性があります。あるいは、特定の MySQL エラーコードのハンドラと、一般的な SQLWARNING クラスのハンドラが同じブロック内に存在することもあります。

あるハンドラがアクティブ化されるかどうかは、それ自体のスコープや条件値だけでなく、ほかにどのようなハンドラが存在するかによっても異なります。ストアドプログラム内で条件が発生すると、サーバーは、適用可能なハンドラを現在のスコープ (現在の BEGIN ... END ブロック) 内で検索します。適用可能なハンドラが存在しない場合は、連続した包含する各スコープ (ブロック) 内のハンドラに関して外側に検索を続行します。特定のスコープで適用可能なハンドラを 1 つ以上見つけると、サーバーは、次の条件の優先順位に基づいてそれらのハンドラから選択します。

- MySQL エラーコードハンドラは、SQLSTATE 値ハンドラより優先されます。
- SQLSTATE 値ハンドラは、一般的な SQLWARNING、SQLEXCEPTION、または NOT FOUND ハンドラより優先されます。
- SQLEXCEPTION ハンドラは、SQLWARNING ハンドラより優先されます。
- NOT FOUND の優先順位は、その条件がどのように発生したかによって異なります。
  - 通常、NOT FOUND クラス内の条件は SQLWARNING または NOT FOUND ハンドラで処理することができ、その両方が存在する場合は SQLWARNING ハンドラが優先されます。NOT FOUND は通常、一連の行をフェッチするために使用されるカーソルがデータセットの最後に達した場合か、または WHERE 句で行が見つからない SELECT ... INTO var_list のインスタンスに対して発生します。
  - NOT FOUND 条件が SIGNAL (または RESIGNAL) ステートメントによって発生した場合、その条件は NOT FOUND ハンドラで処理できますが、SQLWARNING ハンドラでは処理できません。
- 同じ優先順位を持つ適用可能なハンドラが複数存在する可能性があります。たとえば、ステートメントが、それぞれに対してエラー固有のハンドラが存在する、異なるエラーコードを持つ複数の警告を生成する可能性があります。この場合は、サーバーがどのハンドラをアクティブ化するかの選択は不確定であり、その条件が発生した状況に応じて変わることがあります。

ハンドラ選択ルールの 1 つの側面として、複数の適用可能なハンドラが異なるスコープ内に存在する場合は、もっともローカルなスコープを持つハンドラが外側のスコープにあるハンドラより (それが、より具体的な条件のハンドラであっても) 優先される点があります。

ある条件が発生したときに適切なハンドラが存在しない場合、実行されるアクションはその条件のクラスによって異なります。

- SQLEXCEPTION 条件の場合は、EXIT ハンドラが存在するかのように、ストアドプログラムはその条件を発生させたステートメントで終了します。そのプログラムが別のストアドプログラムから呼び出されていた場合は、呼び出し元プログラムが、独自のハンドラに適用されるハンドラ選択ルールを使用してその条件を処理します。
- SQLWARNING 条件の場合は、CONTINUE ハンドラが存在するかのように、プログラムは実行を続行します。
- NOT FOUND 条件では、その条件が正常に発生した場合、アクションは CONTINUE です。SIGNAL または RESIGNAL によって発生した場合、アクションは EXIT です。

次の例は、MySQL によってハンドラ選択ルールがどのように適用されるかを示しています。

次のプロシージャーには 2 つのハンドラが含まれています。つまり、存在しないテーブルを削除しようとする試みに対して発生する特定の SQLSTATE 値 ('42S02') 用に 1 つと、一般的な SQLEXCEPTION クラス用に 1 つです。

```
CREATE PROCEDURE p1()
BEGIN
  DECLARE CONTINUE HANDLER FOR SQLSTATE '42S02'
    SELECT 'SQLSTATE handler was activated' AS msg;
  DECLARE CONTINUE HANDLER FOR SQLEXCEPTION
    SELECT 'SQLEXCEPTION handler was activated' AS msg;

  DROP TABLE test.t;
END;
```

両方のハンドラが同じブロック内で宣言され、同じスコープを持っています。ただし、SQLSTATE ハンドラは SQLEXCEPTION ハンドラより優先されるため、テーブル t が存在しない場合、DROP TABLE ステートメントは SQLSTATE ハンドラをアクティブ化する条件を発生させます。

```
mysql> CALL p1();
+-------------------------------+
| msg                           |
+-------------------------------+
| SQLSTATE handler was activated |
+-------------------------------+
```

次のプロシージャーにも、同じ 2 つのハンドラが含まれています。ただし、今回は、DROP TABLE ステートメントと SQLEXCEPTION ハンドラが SQLSTATE ハンドラに対して内側のブロック内にあります。

```
CREATE PROCEDURE p2()
BEGIN -- outer block
    DECLARE CONTINUE HANDLER FOR SQLSTATE '42S02'
      SELECT 'SQLSTATE handler was activated' AS msg;
  BEGIN -- inner block
    DECLARE CONTINUE HANDLER FOR SQLEXCEPTION
      SELECT 'SQLEXCEPTION handler was activated' AS msg;

    DROP TABLE test.t; -- occurs within inner block
  END;
END;
```

この場合は、条件が発生した場所に対してよりローカルなハンドラが優先されます。SQLSTATE ハンドラより一般的であるにもかかわらず、SQLEXCEPTION ハンドラがアクティブ化されます。

```
mysql> CALL p2();
+-----------------------------------+
| msg                               |
+-----------------------------------+
| SQLEXCEPTION handler was activated |
+-----------------------------------+
```

次のプロシージャーでは、ハンドラの 1 つが、DROP TABLE ステートメントのスコープに対して内側のブロック内で宣言されています。

```
CREATE PROCEDURE p3()
BEGIN -- outer block
  DECLARE CONTINUE HANDLER FOR SQLEXCEPTION
    SELECT 'SQLEXCEPTION handler was activated' AS msg;
  BEGIN -- inner block
    DECLARE CONTINUE HANDLER FOR SQLSTATE '42S02'
      SELECT 'SQLSTATE handler was activated' AS msg;
  END;

  DROP TABLE test.t; -- occurs within outer block
```

```
END;
```

もう一方のハンドラが DROP TABLE によって発生した条件のスコープ内にないため、SQLEXCEPTION ハンドラのみが適用されます。

```
mysql> CALL p3();
+-----------------------------------+
| msg                               |
+-----------------------------------+
| SQLEXCEPTION handler was activated |
+-----------------------------------+
```

次のプロシージャーでは、両方のハンドラが、DROP TABLE ステートメントのスコープに対して内側のブロック内で宣言されています。

```
CREATE PROCEDURE p4()
BEGIN -- outer block
  BEGIN -- inner block
    DECLARE CONTINUE HANDLER FOR SQLEXCEPTION
      SELECT 'SQLEXCEPTION handler was activated' AS msg;
    DECLARE CONTINUE HANDLER FOR SQLSTATE '42S02'
      SELECT 'SQLSTATE handler was activated' AS msg;
  END;

  DROP TABLE test.t; -- occurs within outer block
END;
```

DROP TABLE のスコープ内にないため、どちらのハンドラも適用されません。このステートメントによって発生した条件は未処理になり、プロシージャーをエラーで終了させます。

```
mysql> CALL p4();
ERROR 1051 (42S02): Unknown table 'test.t'
```

13.6.7.7 MySQL の診断領域

SQL ステートメントは、診断領域を移入する診断情報を生成します。標準 SQL には、ネスト化された実行のコンテキストごとの診断領域を含んだ、診断領域スタックがあります。標準 SQL はまた、条件ハンドラの実行中に 2 番目の診断領域を参照するための GET STACKED DIAGNOSTICS 構文もサポートしています。MySQL は、MySQL 5.7 まで STACKED キーワードをサポートしていません。MySQL 5.6 では、診断領域への書き込みを行なった最新のステートメントからの情報を含む 1 つの診断領域が存在します。

このセクションでは、MySQL の診断領域の構造、MySQL によって認識される情報項目、およびステートメントで診断領域をクリアおよび設定する方法について説明します。

診断領域の構造

診断領域には、次の 2 種類の情報が含まれています。

- 発生した条件の数や、影響を受けた行数などのステートメント情報。
- エラーコードやメッセージなどの条件情報。ステートメントが複数の条件を発生させた場合、診断領域のこの部分には条件ごとの条件領域が含まれています。ステートメントがどの条件も発生させない場合、診断領域のこの部分は空です。

3 つの条件を生成するステートメントの場合、診断領域には、次のようなステートメント情報と条件情報が含まれています。

```
Statement information:
  row count
  ... other statement information items ...
Condition area list:
  Condition area 1:
    error code for condition 1
    error message for condition 1
    ... other condition information items ...
  Condition area 2:
    error code for condition 2:
    error message for condition 2
    ... other condition information items ...
  Condition area 3:
    error code for condition 3
    error message for condition 3
    ... other condition information items ...
```

診断領域の情報項目

診断領域には、ステートメント情報と条件情報項目が含まれています。数値項目は整数です。文字項目の文字セットは UTF-8 です。どの項目も NULL にはできません。診断領域を移入するステートメントによってステートメントまたは条件項目が設定されていない場合、その値は、その項目のデータ型に応じて 0 または空の文字列になります。

診断領域のステートメント情報の部分には、次の項目が含まれています。

- NUMBER: 情報が含まれている条件領域の数を示す整数。
- ROW_COUNT: このステートメントによって影響を受けた行数を示す整数。ROW_COUNT には、ROW_COUNT() 関数と同じ値が含まれています (セクション12.14「情報関数」を参照してください)。

診断領域の条件情報の部分には、条件ごとの条件領域が含まれています。条件領域には、1 から NUMBER ステートメント条件項目の値の番号が付けられています。NUMBER が 0 である場合、条件領域は存在しません。

各条件領域には、次のリスト内の項目が含まれています。MySQL 拡張である MYSQL_ERRNO を除き、すべての項目が標準 SQL です。これらの定義は、シグナル以外によって (つまり、SIGNAL または RESIGNAL ステートメントによって) 生成された条件に適用されます。シグナル以外の条件の場合、MySQL は、常に空であるとは示されていない条件項目のみを移入します。条件領域に対するシグナルの影響については、あとで説明されています。

- CLASS_ORIGIN RETURNED_SQLSTATE 値のクラスを含む文字列。RETURNED_SQLSTATE 値が SQL 標準のドキュメント ISO 9075-2 (セクション 24.1、SQLSTATE) で定義されているクラス値で始まる場合、CLASS_ORIGIN は 'ISO 9075' です。それ以外の場合、CLASS_ORIGIN は 'MySQL' です。
- SUBCLASS_ORIGIN: RETURNED_SQLSTATE 値のサブクラスを含む文字列。CLASS_ORIGIN が 'ISO 9075' であるか、または RETURNED_SQLSTATE が '000' で終わる場合、SUBCLASS_ORIGIN は 'ISO 9075' です。それ以外の場合、SUBCLASS_ORIGIN は 'MySQL' です。
- RETURNED_SQLSTATE: この条件の SQLSTATE 値を示す文字列。
- MESSAGE_TEXT: この条件のエラーメッセージを示す文字列。
- MYSQL_ERRNO: この条件の MySQL エラーコードを示す整数。
- CONSTRAINT_CATALOG、CONSTRAINT_SCHEMA、CONSTRAINT_NAME: 違反した制約のカタログ、スキーマ、および名前を示す文字列。これらは常に空です。
- CATALOG_NAME、SCHEMA_NAME、TABLE_NAME、COLUMN_NAME: この条件に関連したカタログ、スキーマ、テーブル、およびカラムを示す文字列。これらは常に空です。
- CURSOR_NAME: カーソル名を示す文字列。これは常に空です。

RETURNED_SQLSTATE、MESSAGE_TEXT、および MYSQL_ERRNO 値の特定のエラーについては、セクションB.3「サーバーのエラーコードおよびメッセージ」を参照してください。

SIGNAL (または RESIGNAL) ステートメントが診断領域を移入する場合、その SET 句は、RETURNED_SQLSTATE を除く任意の条件情報項目に、その項目のデータ型に対して正当な任意の値を割り当てることができます。SIGNAL はまた、RETURNED_SQLSTATE 値も設定しますが、その SET 句で直接設定するわけではありません。その値は、SIGNAL ステートメントの SQLSTATE 引数から取得されます。

SIGNAL は、ステートメント情報項目も設定します。NUMBER を 1 に設定し、ROW_COUNT については、エラーの場合は -1、それ以外の場合は 0 に設定します。

診断領域が移入される方法

診断以外のほとんどの SQL ステートメントは診断領域を自動的に移入しますが、その内容は SIGNAL および RESIGNAL ステートメントを使用して明示的に設定できます。診断領域は、特定の項目を抽出するために GET DIAGNOSTICS を使用して、あるいは条件またはエラーを確認するために SHOW WARNINGS または SHOW ERRORS を使用して検査できます。

SQL ステートメントは、次のように診断領域をクリアおよび設定します。

- サーバーは、ステートメントを解析してからその実行を開始するとき、テーブルを使用する診断以外のステートメントについては診断領域をクリアします。診断ステートメントは、診断領域をクリアしません (SHOW WARNINGS、SHOW ERRORS、GET DIAGNOSTICS)。

- ステートメントが条件を発生させた場合は、以前のステートメントに属する条件の診断領域がクリアされます。例外として、GET DIAGNOSTICS および RESIGNAL によって発生した条件は、診断領域にその領域をクリアすることなく追加されます。

そのため、通常は実行開始時に診断領域をクリアしないステートメントであっても、そのステートメントが条件を発生させた場合は診断領域をクリアします。

次の例は、診断領域に対するさまざまなステートメントの影響を、SHOW WARNINGS を使用して、そこに格納されている条件に関する情報を表示することによって示しています。

次の DROP TABLE ステートメントはテーブルを使用するため、診断領域をクリアし、条件が発生した場合はそれを移入します。

```
mysql> DROP TABLE IF EXISTS test.no_such_table;
Query OK, 0 rows affected, 1 warning (0.01 sec)

mysql> SHOW WARNINGS;
+-------+------+----------------------------------------+
| Level | Code | Message                                |
+-------+------+----------------------------------------+
| Note  | 1051 | Unknown table 'test.no_such_table' |
+-------+------+----------------------------------------+
1 row in set (0.00 sec)
```

次の SET ステートメントはテーブルを使用せず、警告も生成しないため、診断領域を変更されないままにします。

```
mysql> SET @x = 1;
Query OK, 0 rows affected (0.00 sec)

mysql> SHOW WARNINGS;
+-------+------+----------------------------------------+
| Level | Code | Message                                |
+-------+------+----------------------------------------+
| Note  | 1051 | Unknown table 'test.no_such_table' |
+-------+------+----------------------------------------+
1 row in set (0.00 sec)
```

次の SET ステートメントはエラーを生成するため、診断領域をクリアして移入します。

```
mysql> SET @x = @@x;
ERROR 1193 (HY000): Unknown system variable 'x'

mysql> SHOW WARNINGS;
+-------+------+-----------------------------+
| Level | Code | Message                     |
+-------+------+-----------------------------+
| Error | 1193 | Unknown system variable 'x' |
+-------+------+-----------------------------+
1 row in set (0.00 sec)
```

前の SET ステートメントは 1 つの条件を生成したため、1 がこの時点での GET DIAGNOSTICS の唯一の有効な条件番号です。次のステートメントは 2 の条件番号を使用しています。これにより、診断領域にその領域をクリアすることなく追加される警告が生成されます。

```
mysql> GET DIAGNOSTICS CONDITION 2 @p = MESSAGE_TEXT;
Query OK, 0 rows affected, 1 warning (0.00 sec)

mysql> SHOW WARNINGS;
+-------+------+-----------------------------+
| Level | Code | Message                     |
+-------+------+-----------------------------+
| Error | 1193 | Unknown system variable 'xx' |
| Error | 1753 | Invalid condition number    |
+-------+------+-----------------------------+
2 rows in set (0.00 sec)
```

これで、診断領域には 2 つの条件が存在するようになったため、同じ GET DIAGNOSTICS ステートメントが成功します。

```
mysql> GET DIAGNOSTICS CONDITION 2 @p = MESSAGE_TEXT;
Query OK, 0 rows affected (0.00 sec)

mysql> SELECT @p;
+--------------------------+
| @p                       |
```

```
+-------------------------+
| Invalid condition number |
+-------------------------+
1 row in set (0.01 sec)
```

診断領域関連のシステム変数

特定のシステム変数が診断領域のいくつかの側面を制御するか、またはそれに関連しています。

- max_error_count は、診断領域内の条件領域の数を制御します。これより多い条件が発生した場合、MySQL は、超過した条件に関する情報を暗黙のうちに破棄します。(RESIGNAL によって追加された条件は、常に追加されます。空きを作るために、古い条件が必要に応じて破棄されます。)
- warning_count は、発生した条件の数を示します。これには、エラー、警告、および注意が含まれます。通常、NUMBER と warning_count は同じです。ただし、生成された条件の数が max_error_count を超えると、診断領域にはそれ以上の条件が格納されないため、warning_count の値が引き続き増えるのに対して、NUMBER は max_error_count に上限が設定されたままになります。
- error_count は、発生したエラーの数を示します。この値には「見つからない」と例外条件が含まれますが、警告と注意は除外されます。warning_count と同様に、その値は max_error_count を超えることができます。
- sql_notes システム変数が 0 に設定されている場合、注意は格納されず、warning_count も増分しません。

例: max_error_count が 10 である場合、診断領域には最大 10 個の条件領域を含めることができます。ステートメントが 20 個の条件を発生させ、そのうちの 12 個がエラーであるとします。その場合、診断領域には最初の 10 個の条件が含まれ、NUMBER は 10、warning_count は 20、error_count は 12 です。

max_error_count の値を変更しても、次に診断領域を変更しようとするまでは何の効果もありません。診断領域に 10 個の条件領域が含まれているときに max_error_count が 5 に設定された場合、その診断領域のサイズまたは内容への即座の影響は何もありません。

MySQL 5.6 より前は、ステートメント情報項目を直接使用できません。ROW_COUNT は、ROW_COUNT() 関数を呼び出すことによって取得できます。NUMBER は、warning_count システム変数の値によって近似されます。ただし、NUMBER が max_error_count の値に上限が設定されるのに対して、warning_count には上限がありません。

13.7 データベース管理ステートメント

13.7.1 アカウント管理ステートメント

MySQL アカウント情報は、mysql データベースのテーブルに格納されています。このデータベースとアクセス制御システムについては、第5章「MySQL サーバーの管理」で広範囲にわたって説明されています。詳細は、この章を参照するようにしてください。

> **重要**
>
> MySQL の一部のリリースでは、新たな権限や機能を追加するために付与テーブルの構造に変更を加えているものもあります。すべての新しい機能を確実に活用できるようにするには、新しいバージョンの MySQL に更新するときに常に付与テーブルを更新して、最新の構造を持つようにします。セクション4.4.7「mysql_upgrade — MySQL テーブルのチェックとアップグレード」を参照してください。

13.7.1.1 ALTER USER 構文

```
ALTER USER user_specification [, user_specification] ...

user_specification:
    user PASSWORD EXPIRE
```

ALTER USER ステートメントは、MySQL アカウントを変更します。これを使用するには、mysql データベースに対するグローバルな CREATE USER 権限または INSERT 権限が必要です。

> **警告**
>
> ALTER USER は、MySQL 5.6.6 で追加されました。ただし、5.6.6 で ALTER USER は Password カラムも空の文字列に設定するため、このステートメントは 5.6.7 まで使用しないでください。

各アカウントについて、ALTER USER はそのパスワードを期限切れにします。例:

```
ALTER USER 'jeffrey'@'localhost' PASSWORD EXPIRE;
```

各アカウント名には、セクション6.2.3「アカウント名の指定」で説明されている形式が使用されます。アカウント名のユーザー名の部分のみを指定した場合は、'%' のホスト名の部分が使用されます。

アカウントのパスワードを期限切れにすると、mysql.user テーブルの対応する行に影響を与えます。サーバーは、password_expired カラムを 'Y' に設定します。

アカウントのパスワードが期限切れになった場合は、クライアントセッションが制限モードで動作します。制限モードでは、ユーザーが SET PASSWORD ステートメントを発行してアカウントの新しいパスワードを確立するまで、そのセッションで実行された操作はエラーになります。

```
mysql> SELECT 1;
ERROR 1820 (HY000): You must SET PASSWORD before executing this statement

mysql> SET PASSWORD = PASSWORD('new_password');
Query OK, 0 rows affected (0.01 sec)

mysql> SELECT 1;
+---+
| 1 |
+---+
| 1 |
+---+
1 row in set (0.00 sec)
```

MySQL 5.6.8 の時点では、この制限された動作モードでは SET ステートメントが許可されます。これは、そのアカウントのパスワードが、old_passwords をそのデフォルトとは異なる値に設定する必要のあるハッシュ形式を使用している場合に役立ちます。

管理ユーザーがアカウントのパスワードをリセットすることも可能ですが、そのアカウントの既存のセッションはすべて制限されたままになります。ステートメントを正常に実行するには、そのアカウントを使用しているクライアントは切断し、再接続する必要があります。

> 注記
>
> SET PASSWORD を使用してパスワードをその現在の値に設定することによって、パスワードを「リセット」できます。適切なポリシーとして、別のパスワードを選択することをお勧めします。

13.7.1.2 CREATE USER 構文

```
CREATE USER user_specification [, user_specification] ...

user_specification:
    user
    [
      | IDENTIFIED WITH auth_plugin [AS 'auth_string']
        IDENTIFIED BY [PASSWORD] 'password'
    ]
```

CREATE USER ステートメントは、新しい MySQL アカウントを作成します。すでに存在するアカウントに対しては、エラーが発生します。このステートメントを使用するには、mysql データベースに対するグローバルな CREATE USER 権限または INSERT 権限が必要です。各アカウントについて、CREATE USER は、権限のない新しい行を mysql.user テーブル内に作成し、そのアカウントに認証プラグインを割り当てます。使用されている構文に応じて、CREATE USER はそのアカウントにパスワードも割り当てる可能性があります。

各 user_specification 句は、アカウント名と、そのアカウントを使用するクライアントの認証方法に関する情報で構成されます。CREATE USER 構文のこの部分は GRANT と共有されるため、ここでの説明は GRANT にも適用されます。

各アカウント名には、セクション6.2.3「アカウント名の指定」で説明されている形式が使用されます。例:

```
CREATE USER 'jeffrey'@'localhost' IDENTIFIED BY 'mypass';
```

アカウント名のユーザー名の部分のみを指定した場合は、'%' のホスト名の部分が使用されます。

サーバーは、ユーザー指定句にプラグインを指定するための IDENTIFIED WITH またはパスワードを指定するための IDENTIFIED BY が含まれているかどうかに応じて、各アカウントに認証プラグインとパスワードを次のように割り当てます。

- IDENTIFIED WITH を指定すると、サーバーは指定されたプラグインを割り当てますが、そのアカウントにパスワードはありません。
- IDENTIFIED BY を指定すると、サーバーはプラグインを暗黙的に割り当て、さらに指定されたパスワードを割り当てます。
- IDENTIFIED WITH と IDENTIFIED BY のどちらも指定しない場合、サーバーはプラグインを暗黙的に割り当てますが、そのアカウントにパスワードはありません。

アカウントにパスワードがない場合は、そのアカウントの mysql.user テーブル行内の Password カラムが空のままになります。これはセキュアではありません。パスワードを設定するには、SET PASSWORD を使用します。セクション13.7.1.7「SET PASSWORD 構文」を参照してください。

認証プラグインの暗黙的な割り当てでは、サーバーは次のルールを使用します。

- MySQL 5.6.6 の時点では、サーバーはそのアカウントにデフォルトのプラグインを割り当てます。このプラグインが、そのアカウントの mysql.user テーブル行内の plugin カラムの値になります。デフォルトのプラグインは、サーバーの起動時に --default-authentication-plugin オプションがほかの値に設定されないかぎり、mysql_native_password です。
- MySQL 5.6.6 より前は、サーバーはそのアカウントにプラグインを割り当てません。そのアカウントの mysql.user テーブル行内の plugin カラムが空のままになります。

特定のアカウントを使用するクライアント接続の場合は、サーバーがそのアカウントに割り当てられた認証プラグインを呼び出すと、クライアントは、そのプラグインが実装している認証方法によって要求される資格証明を指定する必要があります。サーバーが (アカウントの作成時または接続時に) そのプラグインを見つけることができない場合は、エラーが発生します。

アカウントの mysql.user テーブル行に空以外の plugin カラムが含まれている場合:

- サーバーは、指定されたプラグインを使用してクライアントの接続試行を認証します。
- SET PASSWORD を PASSWORD() とともに使用したアカウントのパスワードへの変更は、PASSWORD() で適切なパスワードハッシュ方式が使用されるように、old_passwords システム変数が認証プラグインに必要な値に設定された状態で実行する必要があります。プラグインが mysql_old_password である場合は、SET PASSWORD を、old_passwords の値には関係なく 4.1 より前のパスワードハッシュを使用する OLD_PASSWORD() とともに使用してパスワードを変更することもできます。

アカウントの mysql.user テーブル行に空の plugin カラムが含まれている場合:

- サーバーは、Password カラムに格納されているパスワードのハッシュ形式に応じて、mysql_native_password または mysql_old_password 認証プラグインを使用してクライアントの接続試行を認証します。
- SET PASSWORD を使用したアカウントのパスワードへの変更は、old_passwords が 4.1 または 4.1 より前のパスワードハッシュを示す、それぞれ 0 または 1 に設定された状態で PASSWORD() とともに、あるいは old_passwords の値には関係なく 4.1 より前のパスワードハッシュを使用する OLD_PASSWORD() とともに実行できます。

CREATE USER の例:

- アカウントの認証プラグインを指定するには、IDENTIFIED WITH auth_plugin を使用します。プラグイン名は、引用符で囲まれた文字列リテラルまたは引用符で囲まれていない名前にすることができます。'auth_string' は、プラグインに渡すオプションの引用符で囲まれた文字列リテラルです。その文字列の意味はプラグインによって解釈されるため、その形式はプラグイン固有です。特定のプラグインが受け入れる認証文字列の値 (存在する場合) については、そのプラグインのドキュメントを参照してください。

  ```
  CREATE USER 'jeffrey'@'localhost' IDENTIFIED WITH mysql_native_password;
  ```

  サーバーは、そのアカウントに指定された認証プラグインを割り当てますが、パスワードは割り当てません。クライアントは、接続時にパスワードを指定する必要がありません。ただし、パスワードのないアカウントはセキュアではありません。アカウントが確実に特定の認証プラグインを使用し、かつ対応するハッシュ形式を持つパスワードが設定されるようにするには、IDENTIFIED WITH でプラグインを明示的に指定したあと、SET PASSWORD を使用してパスワードを設定します。

  ```
  CREATE USER 'jeffrey'@'localhost' IDENTIFIED WITH mysql_native_password;
  SET old_passwords = 0;
  SET PASSWORD FOR 'jeffrey'@'localhost' = PASSWORD('mypass');
  ```

  SET PASSWORD を PASSWORD() とともに使用したアカウントのパスワードへの変更は、PASSWORD() で適切なパスワードハッシュ方式が使用されるように、old_passwords システム変数がアカウントの認証プ

ラグインに必要な値に設定された状態で実行する必要があります。そのため、代わりに sha256_password または mysql_old_password プラグインを使用するには、CREATE USER ステートメントでそのプラグインを指定し、SET PASSWORD を使用する前に old_passwords をそれぞれ 2 または 1 に設定します。(mysql_old_password の使用はお勧めできません。これは非推奨であり、そのサポートは将来の MySQL リリースで削除される予定です。)

- アカウントの作成時にアカウントのパスワードを指定するには、IDENTIFIED BY をプレーンテキストのパスワードのリテラル値とともに使用します。

```
CREATE USER 'jeffrey'@'localhost' IDENTIFIED BY 'mypass';
```

サーバーは、前に説明したようにそのアカウントに認証プラグインを暗黙的に割り当て、さらに特定のパスワードを割り当てます。クライアントは、接続時に特定のパスワードを指定する必要があります。

暗黙的に割り当てられたプラグインが mysql_native_password である場合は、old_passwords システム変数が 0 に設定されている必要があります。そうでない場合、CREATE USER はそのプラグインに必要な形式でパスワードをハッシュしないため、エラーが発生します。

```
mysql> SET old_passwords = 1;
mysql> CREATE USER 'jeffrey'@'localhost' IDENTIFIED BY 'mypass';
ERROR 1827 (HY000): The password hash doesn't have the expected
format. Check if the correct password algorithm is being used with
the PASSWORD() function.

mysql> SET old_passwords = 0;
mysql> CREATE USER 'jeffrey'@'localhost' IDENTIFIED BY 'mypass';
Query OK, 0 rows affected (0.00 sec)
```

- パスワードのハッシュ値 (パスワードに対して PASSWORD() が返す値) がわかっている場合にプレーンテキストのパスワードを指定しないようにするには、そのハッシュ値の前にキーワード PASSWORD を指定します。

```
CREATE USER 'jeffrey'@'localhost'
IDENTIFIED BY PASSWORD '*90E462C37378CED12064BB3388827D2BA3A9B689';
```

サーバーは、前に説明したようにそのアカウントに認証プラグインを暗黙的に割り当て、さらに特定のパスワードを割り当てます。このパスワードハッシュは、割り当てられたプラグインに必要な形式である必要があります。クライアントは、接続時にそのパスワードを指定する必要があります。

- ユーザーがパスワードなしで接続できるようにするには、IDENTIFIED BY 句を含めないようにします。

```
CREATE USER 'jeffrey'@'localhost';
```

サーバーは、前に説明したようにそのアカウントに認証プラグインを暗黙的に割り当てますが、パスワードは割り当てません。クライアントは、接続時にパスワードを指定する必要がありません。ただし、パスワードのないアカウントはセキュアではありません。これを回避するには、SET PASSWORD を使用してアカウントのパスワードを設定します。

先に説明したように、プラグインの暗黙的な割り当ては、デフォルトの認証プラグインによって異なります。--default-authentication-plugin の許可される値は mysql_native_plugin と sha256_password であり、mysql_old_password は許可されません。つまり、CREATE USER ... IDENTIFIED BY 構文で mysql_old_password を使用するアカウントを作成できるように、デフォルトのプラグインを設定することはできません。mysql_old_password を使用するアカウントを作成するには、CREATE USER ... IDENTIFIED WITH を使用してプラグインを明示的に指定したあと、そのパスワードを設定します。

```
CREATE USER 'jeffrey'@'localhost' IDENTIFIED WITH mysql_old_password;
SET old_passwords = 1;
SET PASSWORD FOR 'jeffrey'@'localhost' = PASSWORD('mypass');
```

ただし、mysql_old_password は非推奨であるため、前の手順はお勧めできません。

パスワードと認証プラグインの設定の詳細は、セクション6.3.5「アカウントパスワードの割り当て」およびセクション6.3.7「プラガブル認証」を参照してください。

> **重要**
>
> 状況によっては、CREATE USER がサーバーログ、またはクライアント側にある ~/.mysql_history などの履歴ファイル内に記録されることがあります。つまり、平文のパスワードが、その情報に対する読み取りアクセス権を持つ任意のユーザーによって読み取られる可能性があります。これがサーバーログで発生する条件およびこれを制御する方法については、セクション6.1.2.3「パスワードおよびロギング」を参照し

てください。クライアント側のロギングに関する同様の情報については、セクション4.5.1.3「mysql のロギング」を参照してください。

重要

MySQL の一部のリリースでは、新たな権限や機能を追加するために付与テーブルの構造に変更を加えているものもあります。すべての新しい機能を確実に活用できるようにするには、新しいバージョンの MySQL に更新するときに常に付与テーブルを更新して、最新の構造を持つようにします。セクション4.4.7「mysql_upgrade — MySQL テーブルのチェックとアップグレード」を参照してください。

13.7.1.3 DROP USER 構文

```
DROP USER user [, user] ...
```

DROP USER ステートメントは、1 つ以上の MySQL アカウントとその権限を削除します。これにより、そのアカウントの権限行がすべての付与テーブルから削除されます。存在しないアカウントに対しては、エラーが発生します。このステートメントを使用するには、mysql データベースに対するグローバルな CREATE USER 権限または DELETE 権限が必要です。

各アカウント名には、セクション6.2.3「アカウント名の指定」で説明されている形式が使用されます。例:

```
DROP USER 'jeffrey'@'localhost';
```

アカウント名のユーザー名の部分のみを指定した場合は、'%' のホスト名の部分が使用されます。

重要

DROP USER は、開かれたどのユーザーセッションも自動的には閉じません。さらに、開かれたセッションを持つユーザーが削除されても、このステートメントはそのユーザーのセッションが閉じられるまで有効になりません。セッションが閉じられると、そのユーザーは削除され、そのユーザーが次にログインしようとすると失敗します。これは意図的なものです。

DROP USER は、古いユーザーが作成したどのデータベースまたはそれらのデータベース内のどのオブジェクトも自動的には削除したり、無効にしたりしません。これには、DEFINER 属性に削除されたユーザーが指定されているストアドプログラムまたはビューが含まれます。このようなオブジェクトにアクセスしようとすると、それが定義者のセキュリティーコンテキストで実行された場合は、エラーが生成される可能性があります。(セキュリティーコンテキストについては、セクション20.6「ストアドプログラムおよびビューのアクセスコントロール」を参照してください。)

13.7.1.4 GRANT 構文

```
GRANT
    priv_type [(column_list)]
      [, priv_type [(column_list)]] ...
    ON [object_type] priv_level
    TO user_specification [, user_specification] ...
    [REQUIRE {NONE | ssl_option [[AND] ssl_option] ...}]
    [WITH with_option ...]

GRANT PROXY ON user_specification
    TO user_specification [, user_specification] ...
    [WITH GRANT OPTION]

object_type:
    TABLE
  | FUNCTION
  | PROCEDURE

priv_level:
    *
  | *.*
  | db_name.*
  | db_name.tbl_name
  | tbl_name
  | db_name.routine_name

user_specification:
    user
    [
```

```
    | IDENTIFIED WITH auth_plugin [AS 'auth_string']
      IDENTIFIED BY [PASSWORD] 'password'
  ]

ssl_option:
   SSL
 | X509
 | CIPHER 'cipher'
 | ISSUER 'issuer'
 | SUBJECT 'subject'

with_option:
   GRANT OPTION
 | MAX_QUERIES_PER_HOUR count
 | MAX_UPDATES_PER_HOUR count
 | MAX_CONNECTIONS_PER_HOUR count
 | MAX_USER_CONNECTIONS count
```

GRANT ステートメントは、MySQL ユーザーアカウントに権限を付与します。GRANT にはまた、セキュア接続の使用やサーバーリソースへのアクセスに関する制限などの、その他のアカウント特性を指定する機能もあります。GRANT を使用するには、GRANT OPTION 権限が必要であり、かつ付与しようとしている権限を持っている必要があります。

通常、データベース管理者は最初に CREATE USER を使用してアカウントを作成し、次に GRANT を使用してその権限や特性を定義します。例:

```
CREATE USER 'jeffrey'@'localhost' IDENTIFIED BY 'mypass';
GRANT ALL ON db1.* TO 'jeffrey'@'localhost';
GRANT SELECT ON db2.invoice TO 'jeffrey'@'localhost';
GRANT USAGE ON *.* TO 'jeffrey'@'localhost' WITH MAX_QUERIES_PER_HOUR 90;
```

ただし、GRANT ステートメントで指定されたアカウントがまだ存在しない場合、GRANT は、あとで NO_AUTO_CREATE_USER SQL モードの説明に示されている条件の下でそのアカウントを作成する可能性があります。

REVOKE ステートメントは GRANT に関連しており、管理者がアカウントの権限を削除できるようにします。セクション13.7.1.6「REVOKE 構文」を参照してください。

mysql プログラムから正常に実行された場合、GRANT は Query OK, 0 rows affected で応答します。この操作によってどのような権限が付与されたかを判定するには、SHOW GRANTS を使用します。セクション13.7.5.22「SHOW GRANTS 構文」を参照してください。

GRANT ステートメントには、このセクションの次のトピックで説明されているいくつかの側面があります。

- MySQL によってサポートされる権限
- グローバル権限
- データベース権限
- テーブル権限
- カラム権限
- ストアドルーチン権限
- プロキシユーザー権限
- アカウント名とパスワード
- その他のアカウント特性
- MySQL バージョンと標準 SQL バージョンの GRANT

> **重要**
>
> MySQL の一部のリリースでは、新たな権限や機能を追加するために付与テーブルの構造に変更を加えているものもあります。すべての新しい機能を確実に活用できるようにするには、新しいバージョンの MySQL に更新するときに常に付与テーブルを更新して、最新の構造を持つようにします。セクション4.4.7「mysql_upgrade — MySQL テーブルのチェックとアップグレード」を参照してください。

MySQL によってサポートされる権限

次の表は、GRANT および REVOKE ステートメントに対して指定できる許可される priv_type 権限タイプと、各権限を付与できるレベルについて要約したものです。これらの権限の詳細は、セクション6.2.1「MySQL で提供される権限」を参照してください。

表 13.1 GRANT および REVOKE に対して許容可能な権限

権限	意味と付与可能なレベル
ALL [PRIVILEGES]	GRANT OPTION を除き、指定されたアクセスレベルにあるすべての権限を付与します
ALTER	ALTER TABLE の使用を有効にします。レベル: グローバル、データベース、テーブル。
ALTER ROUTINE	ストアドルーチンの変更または削除を有効にします。レベル: グローバル、データベース、プロシージャー。
CREATE	データベースおよびテーブルの作成を有効にします。レベル: グローバル、データベース、テーブル。
CREATE ROUTINE	ストアドルーチンの作成を有効にします。レベル: グローバル、データベース。
CREATE TABLESPACE	テーブルスペースおよびログファイルグループの作成、変更、または削除を有効にします。レベル: グローバル。
CREATE TEMPORARY TABLES	CREATE TEMPORARY TABLE の使用を有効にします。レベル: グローバル、データベース。
CREATE USER	CREATE USER、DROP USER、RENAME USER、および REVOKE ALL PRIVILEGES の使用を有効にします。レベル: グローバル。
CREATE VIEW	ビューの作成または変更を有効にします。レベル: グローバル、データベース、テーブル。
DELETE	DELETE の使用を有効にします。レベル: グローバル、データベース、テーブル。
DROP	データベース、テーブル、およびビューの削除を有効にします。レベル: グローバル、データベース、テーブル。
EVENT	イベントスケジューラでのイベントの使用を有効にします。レベル: グローバル、データベース。
EXECUTE	ユーザーがストアドルーチンを実行できるようにします。レベル: グローバル、データベース、テーブル。
FILE	ユーザーがサーバーにファイルを読み取らせたり書き込ませたりできるようにします。レベル: グローバル。
GRANT OPTION	権限のほかのアカウントへの付与、またはほかのアカウントからの削除を有効にします。レベル: グローバル、データベース、テーブル、プロシージャー、プロキシ。
INDEX	インデックスの作成または削除を有効にします。レベル: グローバル、データベース、テーブル。
INSERT	INSERT の使用を有効にします。レベル: グローバル、データベース、テーブル、カラム。
LOCK TABLES	ユーザーが SELECT 権限を持っているテーブルに対する LOCK TABLES の使用を有効にします。レベル: グローバル、データベース。
PROCESS	ユーザーが SHOW PROCESSLIST を使用してすべてのプロセスを表示できるようにします。レベル: グローバル。
PROXY	ユーザーのプロキシ設定を有効にします。レベル: ユーザーからユーザーへ。
REFERENCES	実装されていません
RELOAD	FLUSH 操作の使用を有効にします。レベル: グローバル。
REPLICATION CLIENT	ユーザーがマスターまたはスレーブサーバーの場所を問い合わせできるようにします。レベル: グローバル。
REPLICATION SLAVE	レプリケーションスレーブがマスターからバイナリログイベントを読み取れるようにします。レベル: グローバル。

権限	意味と付与可能なレベル
SELECT	SELECT の使用を有効にします。レベル: グローバル、データベース、テーブル、カラム。
SHOW DATABASES	SHOW DATABASES がすべてのデータベースを表示できるようにします。レベル: グローバル。
SHOW VIEW	SHOW CREATE VIEW の使用を有効にします。レベル: グローバル、データベース、テーブル。
SHUTDOWN	mysqladmin shutdown の使用を有効にします。レベル: グローバル。
SUPER	CHANGE MASTER TO、KILL、PURGE BINARY LOGS、SET GLOBAL、mysqladmin debug コマンドなどのその他の管理操作の使用を有効にします。レベル: グローバル。
TRIGGER	トリガー操作を有効にします。レベル: グローバル、データベース、テーブル。
UPDATE	UPDATE の使用を有効にします。レベル: グローバル、データベース、テーブル、カラム。
USAGE	「権限なし」のシノニムです

トリガーはテーブルに関連付けられているため、トリガーを作成または削除するには、そのトリガーではなく、このテーブルに対する TRIGGER 権限が必要です。

GRANT ステートメントでは、ALL [PRIVILEGES] または PROXY 権限は単独で指定する必要があり、ほかの権限とともに指定することはできません。ALL [PRIVILEGES] は、GRANT OPTION および PROXY 権限を除き、権限が付与されるレベルで使用可能なすべての権限を表します。

USAGE を指定すると、権限を持っていないユーザーを作成したり、あるアカウントの REQUIRE または WITH 句を、その既存の権限を変更することなく指定したりできます。

MySQL アカウント情報は、mysql データベースのテーブルに格納されています。このデータベースとアクセス制御システムについては、セクション6.2「MySQL アクセス権限システム」で広範囲にわたって説明されています。詳細は、このセクションを参照するようにしてください。

付与テーブルに、大文字と小文字が混在したデータベースまたはテーブル名を含む権限行が保持されており、かつ lower_case_table_names システム変数が 0 以外の値に設定されている場合は、REVOKE を使用してこれらの権限を取り消すことはできません。付与テーブルを直接操作することが必要になります。(lower_case_table_names が設定されているとき、GRANT はこのような行を作成しませんが、その変数が設定される前にこのような行が作成されていた可能性があります。)

権限は、ON 句に使用される構文に応じて、いくつかのレベルで付与できます。REVOKE の場合は、その同じ ON 構文で、どの権限を取り消すかを指定します。次に示す例には、簡略化のために IDENTIFIED BY 'password' 句が含まれていませんが、そのアカウントがまだ存在しない場合は、パスワードのないセキュアでないアカウントが作成されないようにこの句を含めるようにしてください。

グローバル権限

グローバル権限は管理権限です。つまり、特定のサーバー上のすべてのデータベースに適用されます。グローバル権限を割り当てるには、ON *.* 構文を使用します。

```
GRANT ALL ON *.* TO 'someuser'@'somehost';
GRANT SELECT, INSERT ON *.* TO 'someuser'@'somehost';
```

CREATE TABLESPACE、CREATE USER、FILE、PROCESS、RELOAD、REPLICATION CLIENT、REPLICATION SLAVE、SHOW DATABASES、SHUTDOWN、および SUPER 権限は管理権限であり、グローバルにのみ付与できます。

その他の権限はグローバルに、またはより具体的なレベルで付与できます。

MySQL は、グローバル権限を mysql.user テーブル内に格納します。

データベース権限

データベース権限は、特定のデータベース内のすべてのオブジェクトに適用されます。データベースレベルの権限を割り当てるには、ON db_name.* 構文を使用します。

```
GRANT ALL ON mydb.* TO 'someuser'@'somehost';
```

```
GRANT SELECT, INSERT ON mydb.* TO 'someuser'@'somehost';
```

ON *.* ではなく ON * 構文を使用し、かつデフォルトデータベースを選択した場合、権限はデフォルトデータベースのデータベースレベルで割り当てられます。デフォルトデータベースが存在しない場合は、エラーが発生します。

CREATE、DROP、EVENT、GRANT OPTION、および LOCK TABLES 権限は、データベースレベルで指定できます。また、テーブルまたはルーチン権限もデータベースレベルで指定できます。この場合、これらの権限はデータベース内のすべてのテーブルまたはルーチンに適用されます。

MySQL は、データベース権限を mysql.db テーブル内に格納します。

テーブル権限

テーブル権限は、特定のテーブル内のすべてのカラムに適用されます。テーブルレベルの権限を割り当てるには、ON db_name.tbl_name 構文を使用します。

```
GRANT ALL ON mydb.mytbl TO 'someuser'@'somehost';
GRANT SELECT, INSERT ON mydb.mytbl TO 'someuser'@'somehost';
```

db_name.tbl_name ではなく tbl_name を指定した場合、このステートメントは、デフォルトデータベース内の tbl_name に適用されます。デフォルトデータベースが存在しない場合は、エラーが発生します。

テーブルレベルで許可される priv_type 値は、ALTER、CREATE VIEW、CREATE、DELETE、DROP、GRANT OPTION、INDEX、INSERT、SELECT、SHOW VIEW、TRIGGER、および UPDATE です。

MySQL は、テーブル権限を mysql.tables_priv テーブル内に格納します。

カラム権限

カラム権限は、特定のテーブル内の単一カラムに適用されます。カラムレベルで付与される各権限のあとに、括弧で囲まれた 1 つまたは複数のカラムを指定する必要があります。

```
GRANT SELECT (col1), INSERT (col1,col2) ON mydb.mytbl TO 'someuser'@'somehost';
```

カラムに対して (つまり、column_list 句を使用するときに) 許可される priv_type 値は、INSERT、SELECT、および UPDATE です。

MySQL は、カラム権限を mysql.columns_priv テーブル内に格納します。

ストアドルーチン権限

ALTER ROUTINE、CREATE ROUTINE、EXECUTE、および GRANT OPTION 権限は、ストアドルーチン (プロシージャーおよびファンクション) に適用されます。これらの権限は、グローバルおよびデータベースレベルで付与できます。CREATE ROUTINE を除き、これらの権限は、個々のルーチンに対してルーチンレベルで付与できます。

```
GRANT CREATE ROUTINE ON mydb.* TO 'someuser'@'somehost';
GRANT EXECUTE ON PROCEDURE mydb.myproc TO 'someuser'@'somehost';
```

ルーチンレベルで許可される priv_type 値は、ALTER ROUTINE、EXECUTE、および GRANT OPTION です。CREATE ROUTINE は、最初にルーチンを作成するためにこの権限が必要であるため、ルーチンレベルの権限ではありません。

MySQL は、ルーチンレベルの権限を mysql.procs_priv テーブル内に格納します。

プロキシユーザー権限

PROXY 権限は、あるユーザーを別のユーザーのプロキシにできるようにします。プロキシユーザーは、プロキシ設定されたユーザーになりすますか、またはそのユーザーの識別情報を取得します。

```
GRANT PROXY ON 'localuser'@'localhost' TO 'externaluser'@'somehost';
```

PROXY が付与される場合、その権限は GRANT ステートメントで指定されている唯一の権限である必要があり、REQUIRE 句は指定できず、また許可される唯一の WITH オプションは WITH GRANT OPTION です。

プロキシ設定では、プロキシユーザーが、接続時にプロキシ設定されたユーザーの名前をサーバーに返すプラグイン経由で認証すること、およびプロキシユーザーがプロキシ設定されたユーザーに対する PROXY 権限を持っていることが必要です。詳細および例については、セクション6.3.9「プロキシユーザー」を参照してください。

MySQL は、プロキシ権限を mysql.proxies_priv テーブル内に格納します。

グローバル、データベース、テーブル、およびルーチンレベルの場合、GRANT ALL は、付与しようとしているレベルに存在する権限のみを割り当てます。たとえば、GRANT ALL ON db_name.* はデータベースレベルのステートメントであるため、FILE などのグローバルのみの権限を付与しません。ALL を付与しても、PROXY 権限が割り当てられるわけではありません。

object_type 句 (存在する場合) は、それに続くオブジェクトがテーブル、ストアドファンクション、またはストアドプロシージャーであるときは TABLE、FUNCTION、または PROCEDURE として指定するようにしてください。

データベース、テーブル、カラム、またはルーチンに対する権限は、各権限レベルにある権限の論理 OR として付加的に形成されます。たとえば、ユーザーがグローバルな SELECT 権限を持っている場合は、データベース、テーブル、またはカラムレベルの権限がないことによってその権限を拒否することはできません。権限確認手順の詳細については、セクション6.2.5「アクセス制御、ステージ 2: リクエストの確認」で説明されています。

MySQL では、存在しないデータベースまたはテーブルに対する権限を付与できます。テーブルの場合は、付与される権限に CREATE 権限が含まれている必要があります。この動作は設計によるものであり、データベース管理者がユーザーアカウントと、あとで作成されるデータベースまたはテーブルに対する権限を準備できるようにすることを目的にしています。

> **重要**
>
> MySQL では、データベースまたはテーブルを削除しても、どの権限も自動的には取り消されません。ただし、ルーチンを削除した場合は、そのルーチンに付与されたルーチンレベルの権限がすべて取り消されます。

アカウント名とパスワード

user 値は、GRANT ステートメントが適用される MySQL アカウントを示します。任意のホストのユーザーへの権利の付与に対応するために、MySQL は、user_name@host_name という形式での user 値の指定をサポートしています。user_name または host_name 値が引用符で囲まれていない識別子として正当である場合、それを引用符で囲む必要はありません。ただし、特殊文字 (「-」など) を含む user_name 文字列、または特殊文字やワイルドカード文字 (「%」など) を含む host_name 文字列 (たとえば、'test-user'@'%.com') を指定するには引用符が必要です。ユーザー名とホスト名は個別に引用符で囲みます。

ホスト名には、ワイルドカードを指定できます。たとえば、user_name@'%.example.com' は example.com ドメイン内の任意のホストの user_name に適用され、user_name@'192.168.1.%' は 192.168.1 クラス C サブネット内の任意のホストの user_name に適用されます。

単純な形式 user_name は、user_name@'%' のシノニムです。

MySQL は、ユーザー名でのワイルドカードをサポートしていません。匿名ユーザーを参照するには、GRANT ステートメントで空のユーザー名を含むアカウントを指定します。

```
GRANT ALL ON test.* TO ''@'localhost' ...
```

この場合は、匿名ユーザーの正しいパスワードを使用してローカルホストから接続するすべてのユーザーに、匿名ユーザーアカウントに関連付けられた権限でのアクセスが許可されます。

アカウント名内のユーザー名とホスト名の値の詳細は、セクション6.2.3「アカウント名の指定」を参照してください。

引用符で囲まれた値を指定するには、データベース、テーブル、カラム、およびルーチン名は識別子として引用符で囲みます。ユーザー名とホスト名は識別子または文字列として引用符で囲みます。パスワードは文字列として引用符で囲みます。文字列および識別子として引用符で囲む方法のガイドラインについては、セクション9.1.1「文字列リテラル」およびセクション9.2「スキーマオブジェクト名」を参照してください。

グローバルまたはデータベースレベルの権限を付与する GRANT ステートメントでデータベース名を指定する場合は、「_」や「%」のワイルドカードが許可されます。つまり、たとえば、データベース名の一部として「_」文字を使用する場合は、そのユーザーがワイルドカードパターンに一致する追加のデータベースにアクセスできないようにするために、その文字を GRANT ステートメントで「_」として指定するようにしてください (たとえば、GRANT ... ON `foo\_bar`.* TO ...)。

> **警告**
>
> 匿名ユーザーから MySQL サーバーへの接続を許可する場合は、user_name@localhost として、すべてのローカルユーザーにも権限を付与するようにしてください。それ以

> 外の場合は、指定されたユーザーがローカルマシンから MySQL サーバーにログインしようとすると、(MySQL のインストール中に作成された) mysql.user テーブル内の localhost の匿名ユーザーアカウントが使用されます。詳細は、セクション6.2.4「アクセス制御、ステージ 1: 接続の検証」を参照してください。

前の警告が適用されるかどうかを判定するには、すべての匿名ユーザーを一覧表示する次のクエリーを実行します。

```
SELECT Host, User FROM mysql.user WHERE User='';
```

今説明した問題を回避するには、次のステートメントを使用して、ローカルの匿名ユーザーアカウントを削除します。

```
DROP USER ''@'localhost';
```

GRANT は、最大 60 文字の長さのホスト名をサポートしています。データベース、テーブル、カラム、およびルーチン名には、最大 64 文字を指定できます。ユーザー名には、最大 16 文字を指定できます。

> **警告**
>
> mysql.user テーブルを変更しても、ユーザー名に許可される長さは変更できません。それを行おうとすると、予測できない動作が発生し、ユーザーが MySQL サーバーにログインできなくなる可能性さえあります。mysql データベース内のどのテーブルも、セクション4.4.7「mysql_upgrade — MySQL テーブルのチェックとアップグレード」で説明されている手順による場合を除き、どのような方法でも決して変更しないようにしてください。

ユーザーのサーバー接続時の認証方法を示すために、user_specification 句には、認証プラグインを指定するための IDENTIFIED WITH またはパスワードを指定するための IDENTIFIED BY を含めることができます。ユーザー指定の構文は、CREATE USER ステートメントの場合と同じです。詳細は、セクション13.7.1.2「CREATE USER 構文」を参照してください。

IDENTIFIED BY が存在し、かつグローバルな付与権限 (GRANT OPTION) を指定しているときは、アカウントが存在し、すでにパスワードが設定されている場合でも、そのパスワードがアカウントの新しいパスワードになります。IDENTIFIED BY を指定しない場合、アカウントのパスワードは変更されないままになります。

GRANT ステートメントで指定されたアカウントが存在しない場合、実行されるアクションは NO_AUTO_CREATE_USER SQL モードによって異なります。

- NO_AUTO_CREATE_USER が有効になっていない場合、GRANT はこのアカウントを作成します。IDENTIFIED BY を使用して空以外のパスワードを指定していないかぎり、これはまったくセキュアではありません。
- NO_AUTO_CREATE_USER が有効になっている場合は、IDENTIFIED BY を使用して空以外のパスワードを指定するか、または IDENTIFIED WITH を使用して認証プラグインを指定していないかぎり、GRANT は失敗し、このアカウントを作成しません。

MySQL 5.6.12 の時点では、アカウントがすでに存在する場合、IDENTIFIED WITH は、新しいアカウントの作成時に使用されることのみを目的にしているため禁止されます。

> **重要**
>
> 状況によっては、GRANT がサーバーログ、またはクライアント側にある ~/.mysql_history などの履歴ファイル内に記録されることがあります。つまり、平文のパスワードが、その情報に対する読み取りアクセス権を持つ任意のユーザーによって読み取られる可能性があります。これがサーバーログで発生する条件およびこれを制御する方法については、セクション6.1.2.3「パスワードおよびロギング」を参照してください。クライアント側のロギングに関する同様の情報については、セクション4.5.1.3「mysql のロギング」を参照してください。

その他のアカウント特性

WITH 句は、次のいくつかの目的に使用されます。

- ユーザーがほかのユーザーに権限を付与できるようにするため
- ユーザーに対するリソース制限を指定するため

- ユーザーがサーバーへのセキュア接続を使用する必要があるかどうか、およびその方法を指定するため

WITH GRANT OPTION 句は、ユーザーが、そのユーザーの持つ指定された権限レベルにある任意の権限をほかのユーザーに与えることができるようにします。異なる権限を持つ 2 人のユーザーが権限を組み合わせることができる可能性があるため、GRANT OPTION 権限を与える相手には十分に注意してください。

自分が保持していない権限を別のユーザーに付与することはできません。GRANT OPTION 権限を使用して割り当てることができるのは、自分が保持している権限だけです。

あるユーザーに特定の権限レベルにある GRANT OPTION 権限を付与すると、そのユーザーがそのレベルに保持している (または、将来与えられる可能性のある) すべての権限も、そのユーザーからほかのユーザーに付与される場合があることに注意してください。あるユーザーに、データベースに対する INSERT 権限を付与するとします。次に、そのデータベースに対する SELECT 権限を付与し、WITH GRANT OPTION を指定した場合、そのユーザーはほかのユーザーに SELECT 権限だけでなく、INSERT 権限も与えることができます。そのあと、そのユーザーにデータベースに対する UPDATE 権限を付与すると、そのユーザーは INSERT、SELECT、および UPDATE を付与できます。

管理者以外のユーザーには、グローバルな、または mysql データベースに対する ALTER 権限を付与してはいけません。それを行うと、そのユーザーはテーブルの名前を変更することによって権限システムの破壊を試みることができます。

特定の権限に関連付けられたセキュリティーリスクの詳細は、セクション6.2.1「MySQL で提供される権限」を参照してください。

いくつかの WITH 句オプションは、アカウントによるサーバーリソースの使用に関する制限を指定します。

- MAX_QUERIES_PER_HOUR count、MAX_UPDATES_PER_HOUR count、および MAX_CONNECTIONS_PER_HOUR count 制限は、いずれかの特定の 1 時間の間にこのアカウントに対して許可されるサーバーへのクエリー、更新、および接続の数を制限します。(結果がクエリーキャッシュから得られたクエリーは、MAX_QUERIES_PER_HOUR 制限に対してカウントされません。) count が 0 (デフォルト) である場合、これは、このアカウントに対する制限が存在しないことを示します。

- MAX_USER_CONNECTIONS count 制限は、このアカウントによるサーバーへの同時接続の最大数を制限します。0 以外の count は、このアカウントに対する制限を明示的に指定します。count が 0 (デフォルト) である場合、サーバーは、max_user_connections システム変数のグローバル値からこのアカウントの同時接続の数を決定します。max_user_connections もゼロである場合は、アカウントに制限がありません。

既存の権限に影響を与えることなく既存のユーザーに対するリソース制限を指定するには、GRANT USAGE をグローバルレベルで使用し (ON *.*)、変更される制限を指定します。例:

```
GRANT USAGE ON *.* TO ...
  WITH MAX_QUERIES_PER_HOUR 500 MAX_UPDATES_PER_HOUR 100;
```

指定されない制限は、その現在の値を保持します。

サーバーリソースへのアクセス制限の詳細は、セクション6.3.4「アカウントリソース制限の設定」を参照してください。

MySQL は、ユーザー名とパスワードに基づいた通常の認証に加えて、X509 証明書の属性を確認できます。MySQL アカウントの SSL 関連オプションを指定するには、GRANT ステートメントの REQUIRE 句を使用します。(MySQL での SSL の使用に関する背景情報については、セクション6.3.10「セキュアな接続のための SSL の使用」を参照してください。)

特定のアカウントの接続タイプを制限するには、次のいくつかの可能性があります。

- REQUIRE NONE は、このアカウントに SSL または X509 の要件がないことを示します。これは、SSL 関連の REQUIRE オプションが指定されていない場合のデフォルトです。ユーザー名とパスワードが有効であれば、暗号化されていない接続が許可されます。ただし、クライアントに適切な証明書と鍵ファイルが存在する場合は、そのクライアントのオプションで暗号化された接続も使用できます。つまり、クライアントはどの SSL コマンドオプションも指定する必要がなく、その場合は接続が暗号化されません。暗号化された接続を使用するには、クライアントは --ssl-ca オプションか、または --ssl-ca、--ssl-key、--ssl-cert の 3 つのすべてのオプションを指定する必要があります。

- REQUIRE SSL オプションは、このアカウントの SSL 暗号化接続のみを許可するようサーバーに指示します。

```
GRANT ALL PRIVILEGES ON test.* TO 'root'@'localhost'
  IDENTIFIED BY 'goodsecret' REQUIRE SSL;
```

接続するには、クライアントはサーバー証明書を認証するための --ssl-ca オプションを指定する必要があり、さらに --ssl-key および --ssl-cert オプションを指定する可能性があります。--ssl-ca オプションと --ssl-capath オプションのどちらも指定されていない場合、クライアントはサーバー証明書を認証しません。

- REQUIRE X509 は、クライアントに有効な証明書が存在する必要があるが、正確な証明書、発行者、およびサブジェクトは問題にならないことを示します。唯一の要件は、いずれかの CA 証明書でその署名を検証できるべきであるということです。

```
GRANT ALL PRIVILEGES ON test.* TO 'root'@'localhost'
  IDENTIFIED BY 'goodsecret' REQUIRE X509;
```

接続するには、クライアントは --ssl-ca、--ssl-key、および --ssl-cert オプションを指定する必要があります。REQUIRE のオプション ISSUER や SUBJECT は暗黙的に X509 を示しているため、これはまた、これらのオプションにも当てはまります。

- REQUIRE ISSUER 'issuer' は、接続試行に対して、クライアントが CA 'issuer' によって発行された有効な X509 証明書を提供する必要があるという制限を設定します。クライアントが、有効ではあるが、別の発行者を含む証明書を提供した場合、サーバーはその接続を拒否します。X509 証明書の使用には常に暗号化が含まれるため、この場合、SSL オプションは必要ありません。

```
GRANT ALL PRIVILEGES ON test.* TO 'root'@'localhost'
  IDENTIFIED BY 'goodsecret'
  REQUIRE ISSUER '/C=FI/ST=Some-State/L=Helsinki/
    O=MySQL Finland AB/CN=Tonu Samuel/emailAddress=tonu@example.com';
```

'issuer' 値は、1 つの文字列として入力するようにしてください。

> **注記**
>
> MySQL が 0.9.6h より古いバージョンの OpenSSL に対してリンクされている場合、'issuer' 値には emailAddress ではなく Email を使用します。

- REQUIRE SUBJECT 'subject' は、接続試行に対して、クライアントがサブジェクト subject を含む有効な X509 証明書を提供する必要があるという制限を設定します。クライアントが、有効ではあるが、別のサブジェクトを含む証明書を提供した場合、サーバーはその接続を拒否します。

```
GRANT ALL PRIVILEGES ON test.* TO 'root'@'localhost'
  IDENTIFIED BY 'goodsecret'
  REQUIRE SUBJECT '/C=EE/ST=Some-State/L=Tallinn/
    O=MySQL demo client certificate/
    CN=Tonu Samuel/emailAddress=tonu@example.com';
```

'subject' 値は、1 つの文字列として入力するようにしてください。MySQL は、この値と証明書内の値の単純な文字列比較を実行するため、大文字と小文字の区別やコンポーネントの順序は、証明書内に存在するものと正確に同じように指定する必要があります。

> **注記**
>
> emailAddress については、REQUIRE ISSUER の説明にある注を参照してください。

- 十分な強度の暗号化と鍵の長さが確実に使用されるようにするには、REQUIRE CIPHER 'cipher' が必要です。短い暗号化鍵を使用する古いアルゴリズムが使用されると、SSL 自体が弱くなる場合があります。このオプションを使用すると、接続に特定の暗号化方式を使用するように要求できます。

```
GRANT ALL PRIVILEGES ON test.* TO 'root'@'localhost'
  IDENTIFIED BY 'goodsecret'
  REQUIRE CIPHER 'EDH-RSA-DES-CBC3-SHA';
```

SUBJECT、ISSUER、および CIPHER オプションを REQUIRE 句内で次のように組み合わせることができます。

```
GRANT ALL PRIVILEGES ON test.* TO 'root'@'localhost'
  IDENTIFIED BY 'goodsecret'
  REQUIRE SUBJECT '/C=EE/ST=Some-State/L=Tallinn/
    O=MySQL demo client certificate/
    CN=Tonu Samuel/emailAddress=tonu@example.com'
  AND ISSUER '/C=FI/ST=Some-State/L=Helsinki/
    O=MySQL Finland AB/CN=Tonu Samuel/emailAddress=tonu@example.com'
  AND CIPHER 'EDH-RSA-DES-CBC3-SHA';
```

これらのオプションの順序は問題になりませんが、どのオプションも 2 回指定することはできません。AND キーワードは、REQUIRE オプション間のオプションです。

1 人のユーザーのテーブル、カラム、またはルーチン権限を使用している場合でも、サーバーはすべてのユーザーのテーブル、カラム、およびルーチン権限を検査するため、これにより MySQL が少し遅くなります。同様に、いずれかのユーザーのクエリー、更新、または接続の数を制限した場合、サーバーはこれらの値をモニターする必要があります。

MySQL バージョンと標準 SQL バージョンの GRANT

MySQL バージョンと標準 SQL バージョンの GRANT の最大の違いは次のとおりです。

- MySQL は、権限をユーザー名だけではなく、ホスト名とユーザー名の組み合わせに関連付けます。
- 標準 SQL はグローバルまたはデータベースレベルの権限を持たず、また MySQL がサポートするすべての権限タイプをサポートしているわけでもありません。
- MySQL は、標準 SQL の UNDER 権限をサポートしていません。
- 標準 SQL の権限は、階層的な方法で構造化されています。ユーザーを削除した場合は、そのユーザーに付与されていたすべての権限が取り消されます。これはまた、DROP USER を使用した場合の MySQL にも当てはまります。セクション13.7.1.3「DROP USER 構文」を参照してください。
- 標準 SQL では、テーブルを削除すると、そのテーブルに対するすべての権限が取り消されます。標準 SQL では、権限を取り消すと、その権限に基づいて付与されていたすべての権限も取り消されます。MySQL では、権限は明示的な DROP USER または REVOKE ステートメントを使用するか、あるいは MySQL 付与テーブルを直接操作することによってのみ削除できます。
- MySQL では、テーブル内の一部のカラムに対してのみ INSERT 権限を持つことができます。この場合、INSERT 権限を持っているカラムの値のみを挿入するのであれば、そのテーブルに対して引き続き INSERT ステートメントを実行できます。厳密な SQL モードが有効になっていない場合、省略されたカラムはその暗黙のデフォルト値に設定されます。厳密モードでは、省略されたカラムのいずれかにデフォルト値がない場合、このステートメントは拒否されます。(標準 SQL では、すべてのカラムに対して INSERT 権限を持つ必要があります。) セクション5.1.7「サーバー SQL モード」では、厳密モードについて説明しています。セクション11.6「データ型デフォルト値」では、暗黙のデフォルト値について説明しています。

13.7.1.5 RENAME USER 構文

```
RENAME USER old_user TO new_user
  [, old_user TO new_user] ...
```

RENAME USER ステートメントは、既存の MySQL アカウントの名前を変更します。存在しない古いアカウント、またはすでに存在する新しいアカウントに対しては、エラーが発生します。このステートメントを使用するには、mysql データベースに対するグローバルな CREATE USER 権限または UPDATE 権限が必要です。

各アカウント名には、セクション6.2.3「アカウント名の指定」で説明されている形式が使用されます。例:

```
RENAME USER 'jeffrey'@'localhost' TO 'jeff'@'127.0.0.1';
```

アカウント名のユーザー名の部分のみを指定した場合は、'%' のホスト名の部分が使用されます。

RENAME USER により、古いユーザーによって保持されていた権限は新しいユーザーによって保持される権限になります。ただし、RENAME USER は、古いユーザーが作成したどのデータベースまたはそれらのデータベース内のどのオブジェクトも自動的には削除したり、無効にしたりしません。これには、DEFINER 属性に古いユーザーが指定されているストアドプログラムまたはビューが含まれます。このようなオブジェクトにアクセスしようとすると、それが定義者のセキュリティーコンテキストで実行された場合は、エラーが生成される可能性があります。(セキュリティーコンテキストについては、セクション20.6「ストアドプログラムおよびビューのアクセスコントロール」を参照してください。)

権限の変更は、セクション6.2.6「権限変更が有効化される時期」に示されているように有効になります。

13.7.1.6 REVOKE 構文

```
REVOKE
    priv_type [(column_list)]
      [, priv_type [(column_list)]] ...
    ON [object_type] priv_level
    FROM user [, user] ...

REVOKE ALL PRIVILEGES, GRANT OPTION
    FROM user [, user] ...
```

```
REVOKE PROXY ON user
    FROM user [, user] ...
```

REVOKE ステートメントを使用すると、システム管理者は MySQL アカウントから権限を取り消すことができます。各アカウント名には、セクション6.2.3「アカウント名の指定」で説明されている形式が使用されます。例:

```
REVOKE INSERT ON *.* FROM 'jeffrey'@'localhost';
```

アカウント名のユーザー名の部分のみを指定した場合は、'%' のホスト名の部分が使用されます。

各権限が存在するレベル、priv_type と priv_level の許可される値、およびユーザーとパスワードを指定するための構文の詳細は、セクション13.7.1.4「GRANT 構文」を参照してください。

最初の REVOKE 構文を使用するには、GRANT OPTION 権限が必要であり、かつ取り消そうとしている権限を持っている必要があります。

すべての権限を取り消すには、2 番目の構文を使用します。これにより、指定された 1 人または複数のユーザーのすべてのグローバル、データベース、テーブル、カラム、およびルーチン権限が削除されます。

```
REVOKE ALL PRIVILEGES, GRANT OPTION FROM user [, user] ...
```

この REVOKE 構文を使用するには、mysql データベースに対するグローバルな CREATE USER 権限または UPDATE 権限が必要です。

REVOKE は権限を削除しますが、mysql.user テーブルエントリは削除しません。ユーザーアカウントを完全に削除するには、DROP USER (セクション13.7.1.3「DROP USER 構文」を参照してください) または DELETE を使用します。

付与テーブルに、大文字と小文字が混在したデータベースまたはテーブル名を含む権限行が保持されており、かつ lower_case_table_names システム変数が 0 以外の値に設定されている場合は、REVOKE を使用してこれらの権限を取り消すことはできません。付与テーブルを直接操作することが必要になります。(lower_case_table_names が設定されているとき、GRANT はこのような行を作成しませんが、その変数が設定される前にこのような行が作成されていた可能性があります。)

mysql プログラムから正常に実行された場合、REVOKE は Query OK, 0 rows affected で応答します。この操作によってどのような権限が付与されたかを判定するには、SHOW GRANTS を使用します。セクション13.7.5.22「SHOW GRANTS 構文」を参照してください。

13.7.1.7 SET PASSWORD 構文

```
SET PASSWORD [FOR user] =
    {
        PASSWORD('cleartext password')
      | OLD_PASSWORD('cleartext password')
      | 'encrypted password'
    }
```

SET PASSWORD ステートメントは、MySQL ユーザーアカウントにパスワードを割り当てます。

- FOR user 句を指定しない場合、このステートメントは、現在のユーザーのパスワードを設定します。

  ```
  SET PASSWORD = PASSWORD('cleartext password');
  ```

  匿名以外のアカウントを使用してサーバーに接続したクライアントはすべて、そのアカウントのパスワードを変更できます。サーバーがどのアカウントを自動式に認証したかを表示するには、CURRENT_USER() 関数を呼び出します。

  ```
  SELECT CURRENT_USER();
  ```

- FOR user 句を指定した場合、このステートメントは、指定されたアカウントのパスワードを設定します。ただし、そのアカウントが存在する必要があります。

  ```
  SET PASSWORD FOR 'jeffrey'@'localhost' = PASSWORD('cleartext password');
  ```

  この場合は、mysql データベースに対する UPDATE 権限が必要です。

read_only システム変数が有効になっている場合、SET PASSWORD には、ほかのすべての必要な権限に加えて SUPER 権限が必要です。

FOR user 句が指定されている場合、このアカウント名には、セクション6.2.3「アカウント名の指定」で説明されている形式が使用されます。user 値は、'user_name'@'host_name' として指定するようにしてください。ここ

で、'user_name' と 'host_name' は、そのアカウントの mysql.user テーブル行の User および Host カラムにリストされている内容とまったく同じです。(ユーザー名のみを指定した場合は、'%' のホスト名が使用されます。) たとえば、'bob' と '%.example.org' の User および Host カラム値を使用してアカウントのパスワードを設定するには、このステートメントを次のように記述します。

```
SET PASSWORD FOR 'bob'@'%.example.org' = PASSWORD('cleartext password');
```

パスワードは、次の方法で指定できます。

- PASSWORD() 関数を使用して

  この関数の引数は、平文 (暗号化されていない) パスワードです。PASSWORD() はパスワードをハッシュし、暗号化されたパスワード文字列を返します。

  old_passwords システム変数値により、PASSWORD() によって使用されるハッシュ方式が決定されます。SET PASSWORD がパスワードを正しい形式でないとして拒否した場合は、ハッシュ方式を変更するために old_passwords を変更することが必要になる可能性があります。たとえば、そのアカウントが mysql_native_password プラグインを使用している場合、old_passwords 値は 0 である必要があります。

  ```
  SET old_passwords = 0;
  SET PASSWORD FOR 'jeffrey'@'localhost' = PASSWORD('mypass');
  ```

  old_passwords 値が認証プラグインに必要な値と異なる場合は、PASSWORD() によって返されたハッシュされたパスワード値がそのプラグインに許容されず、パスワードを設定しようとするとエラーが生成されます。例:

  ```
  mysql> SET old_passwords = 1;
  mysql> SET PASSWORD FOR 'jeffrey'@'localhost' = PASSWORD('mypass');
  ERROR 1372 (HY000): Password hash should be a 41-digit hexadecimal number
  ```

- OLD_PASSWORD() 関数を使用して:

  この関数の引数は、平文 (暗号化されていない) パスワードです。OLD_PASSWORD() は 4.1 より前のハッシュを使用してパスワードをハッシュし、暗号化されたパスワード文字列を返します。このハッシュ方式は、mysql_old_password 認証プラグインを使用するアカウントにのみ適しています。

- すでに暗号化されたパスワード文字列を使用して

  このパスワードは、文字列リテラルとして指定されます。これは、このアカウントに使用されている認証方法に必要なハッシュ形式の、すでに暗号化されたパスワード値を表している必要があります。

次の表は、old_passwords の許可される値、それぞれの値に対するパスワードハッシュ方式、およびそれぞれの方式でハッシュされたパスワードを使用する認証プラグインを示します。これらの値は MySQL 5.6.6 以降で許可されます。5.6.6 より前では、許可される値は 0 (または OFF) および 1 (または ON) です。

値	パスワードハッシュ方式	関連付けられた認証プラグイン
0	MySQL 4.1 ネイティブハッシュ	mysql_native_password
1	4.1 以前の (「古い」) ハッシュ	mysql_old_password
2	SHA-256 ハッシュ	sha256_password

パスワードの設定の詳細は、セクション6.3.5「アカウントパスワードの割り当て」を参照してください。

> **重要**
>
> 状況によっては、SET PASSWORD がサーバーログ、またはクライアント側にある ~/.mysql_history などの履歴ファイル内に記録されることがあります。つまり、平文のパスワードが、その情報に対する読み取りアクセス権を持つ任意のユーザーによって読み取られる可能性があります。これがサーバーログで発生する条件およびこれを制御する方法については、セクション6.1.2.3「パスワードおよびロギング」を参照してください。クライアント側のロギングに関する同様の情報については、セクション4.5.1.3「mysql のロギング」を参照してください。

> **注意**
>
> 4.1 より前のクライアントプログラムを使用して MySQL 4.1 以降のサーバーに接続している場合は、パスワードを変更する前に、必ずセクション6.1.2.4「MySQL でのパスワードハッシュ」を読んでください。デフォルトのパスワードハッシュ形式が MySQL

> 4.1 で変更されたため、パスワードを変更すると、そのパスワードが 4.1 より前のクライアントが生成できないハッシュ形式を使用して格納され、それにより、あとでサーバーに接続できなくなる可能性があります。

MySQL レプリケーションを使用している場合は、現在、CHANGE MASTER TO ステートメントの一部としてレプリケーションスレーブで使用されるパスワードは、実質的に長さが 32 文字に制限されます。パスワードがこれより長い場合、超過した文字はすべて切り捨てられます。これは、一般に MySQL Server によって適用される制限のためではなく、どちらかというと、MySQL レプリケーションに固有の問題です。(詳細は、Bug #43439 を参照してください。)

13.7.2 テーブル保守ステートメント

13.7.2.1 ANALYZE TABLE 構文

```
ANALYZE [NO_WRITE_TO_BINLOG | LOCAL] TABLE
    tbl_name [, tbl_name] ...
```

ANALYZE TABLE は、テーブルのキー分布を分析して格納します。分析中、そのテーブルは InnoDB および MyISAM に対する読み取りロックでロックされます。このステートメントは、InnoDB、NDB、および MyISAM テーブルで機能します。MyISAM テーブルの場合、このステートメントは myisamchk --analyze を使用することと同等です。

InnoDB 内で分析がどのように機能するかの詳細は、セクション14.13.16.1「永続的オプティマイザ統計のパラメータの構成」およびセクション14.13.17「InnoDB テーブルに対する ANALYZE TABLE の複雑さの推定」を参照してください。セクション14.6.7「InnoDB テーブル上の制限」も参照してください。特に、innodb_stats_persistent オプションを有効にした場合は、InnoDB テーブルに大量のデータをロードするか、またはそのテーブルの新しいインデックスを作成したあとに ANALYZE TABLE を実行する必要があります。

MySQL は、格納されたキー分布を使用して、定数以外の何かに対して結合が実行されたときにテーブルを結合する順序を決定します。さらに、クエリー内の特定のテーブルにどのインデックスを使用するかを決定する場合は、キー分布を使用できます。

このステートメントには、このテーブルに対する SELECT および INSERT 権限が必要です。

ANALYZE TABLE はパーティション化されたテーブルに対してサポートされているため、ALTER TABLE ... ANALYZE PARTITION を使用して 1 つ以上のパーティションを分析できます。詳細は、セクション 13.1.7「ALTER TABLE 構文」およびセクション19.3.4「パーティションの保守」を参照してください。

MySQL 5.6.11 でのみ、このステートメントを発行する前に、gtid_next を AUTOMATIC に設定する必要があります。(Bug #16062608、Bug #16715809、Bug #69045)

ANALYZE TABLE は、次のカラムを含む結果セットを返します。

カラム	値
Table	テーブル名
Op	常に analyze
Msg_type	status、error、info、note、または warning
Msg_text	情報メッセージ

SHOW INDEX ステートメントを使用して、格納されたキー分布を確認できます。セクション13.7.5.23「SHOW INDEX 構文」を参照してください。

テーブルが最後の ANALYZE TABLE ステートメントのあとに変更されていない場合、そのテーブルが再度分析されることはありません。

デフォルトでは、サーバーは ANALYZE TABLE ステートメントをバイナリログに書き込み、それらがレプリケーションスレーブにレプリケートされるようにします。ロギングを抑制するには、オプションの NO_WRITE_TO_BINLOG キーワード、またはその エイリアス LOCAL を指定します。

13.7.2.2 CHECK TABLE 構文

```
CHECK TABLE tbl_name [, tbl_name] ... [option] ...
```

```
option = {FOR UPGRADE | QUICK | FAST | MEDIUM | EXTENDED | CHANGED}
```

CHECK TABLE は、1 つまたは複数のテーブルをエラーがないかどうかチェックします。CHECK TABLE は、InnoDB、MyISAM、ARCHIVE、および CSV テーブルに対して機能します。MyISAM テーブルの場合は、キー統計の更新も実行されます。

テーブルをチェックするには、それに対する何らかの権限が必要です。

CHECK TABLE はまた、ビューをチェックして、そのビュー定義で参照されているテーブルが存在しなくなっているなどの問題がないかどうかを調べることもできます。

CHECK TABLE はパーティション化されたテーブルに対してサポートされているため、ALTER TABLE ... CHECK PARTITION を使用して 1 つ以上のパーティションをチェックできます。詳細は、セクション13.1.7「ALTER TABLE 構文」およびセクション19.3.4「パーティションの保守」を参照してください。

MySQL 5.6.11 でのみ、このステートメントを発行する前に、gtid_next を AUTOMATIC に設定する必要があります。(Bug #16062608、Bug #16715809、Bug #69045)

出力

CHECK TABLE は、次のカラムを含む結果セットを返します。

カラム	値
Table	テーブル名
Op	常に check
Msg_type	status、error、info、note、または warning
Msg_text	情報メッセージ

このステートメントによって、チェックされたテーブルごとに多数の情報行が生成される可能性があります。最後の行には status の Msg_type 値が含まれ、Msg_text は通常 OK になります。OK または Table is already up to date が得られない場合は通常、そのテーブルの修復を実行するべきです。セクション7.6「MyISAM テーブルの保守とクラッシュリカバリ」を参照してください。Table is already up to date は、そのテーブルのストレージエンジンがテーブルのチェックは必要ないと判断したことを示します。

バージョン互換性のチェック

FOR UPGRADE オプションは、指定されたテーブルが現在のバージョンの MySQL と互換性があるかどうかをチェックします。FOR UPGRADE を指定すると、サーバーは各テーブルをチェックして、テーブルの作成後にそのテーブルのいずれかのデータ型またはインデックスで互換性のない変更が発生しているかどうかを判定します。発生していない場合は、チェックが成功します。それ以外で、非互換性の可能性がある場合、サーバーはそのテーブルに対して完全なチェックを実行します (これには、ある程度時間がかかることがあります)。完全なチェックが成功した場合、サーバーは、そのテーブルの .frm ファイルを現在の MySQL バージョン番号でマークします。.frm ファイルをマークすると、このテーブルに対する同じバージョンのサーバーによるそれ以降のチェックが確実に速くなります。

データ型のストレージフォーマットが変更されたか、またはそのソート順序が変更されたために非互換性が発生する可能性があります。弊社の目的はそれらの変更を避けることですが、各リリースの間の非適合性よりもさらに深刻な問題を修正するために必要である場合もあります。

現在、FOR UPGRADE では、次の非互換性が検出されています。

- InnoDB および MyISAM テーブルの TEXT カラム内の最後の領域のインデックス順序が MySQL 4.1 と 5.0 の間で変更されました。
- 新しい DECIMAL データ型のストレージ方法が MySQL 5.0.3 と 5.0.5 の間で変更されました。
- テーブルが、現在実行しているものとは異なるバージョンの MySQL サーバーによって作成された場合、FOR UPGRADE は、そのテーブルに互換性のないバージョンの .frm ファイルが含まれていることを示します。この場合、CHECK TABLE によって返される結果セットには、Msg_type 値が error で、Msg_text 値が Table upgrade required. Please do "REPAIR TABLE `tbl_name`" to fix it! である行が含まれています。
- 文字セットまたは照合順序に対して、テーブルインデックスの再構築が必要な変更が加えられる場合があります。これらの変更や、それが FOR UPGRADE によっていつ検出されるかの詳細は、セクション2.11.3「テーブルまたはインデックスの再構築が必要かどうかのチェック」を参照してください。

- YEAR(2) データ型は、MySQL 5.6.6 の時点では非推奨です。YEAR(2) カラムを含むテーブルの場合、CHECK TABLE では、YEAR(2) を YEAR(4) に変換する REPAIR TABLE が推奨されます。

データー貫性のチェック

指定できるその他のチェックオプションを次の表に示します。これらのオプションはストレージエンジンに渡されますが、そこで使用される場合とされない場合があります。

型	意味
QUICK	正しくないリンクをチェックするための行のスキャンを行いません。InnoDB および MyISAM テーブルとビューに適用されます。
FAST	正しく閉じられていないテーブルのみを検査します。MyISAM テーブルとビューにのみ適用されます。InnoDB では無視されます。
CHANGED	最後のチェック以降に変更されたか、または正しく閉じられていないテーブルのみをチェックします。MyISAM テーブルとビューにのみ適用されます。InnoDB では無視されます。
MEDIUM	削除されたリンクが有効であることを検証するために行をスキャンします。また、行のキーチェックサムも計算し、キーの計算されたチェックサムを使用してこれを検証します。MyISAM テーブルとビューにのみ適用されます。InnoDB では無視されます。
EXTENDED	行ごとにすべてのキーの完全なキールックアップを実行します。これにより、テーブルの 100% の整合性が保証されますが、長い時間がかかります。MyISAM テーブルとビューにのみ適用されます。InnoDB では無視されます。

QUICK、MEDIUM、または EXTENDED オプションのいずれも指定されていない場合、動的フォーマットの MyISAM テーブルに対するデフォルトのチェックタイプは MEDIUM です。これにより、そのテーブルに対して myisamchk --medium-check tbl_name を実行したのと同じ結果が得られます。また、CHANGED または FAST が指定されていないかぎり、静的フォーマットの MyISAM テーブルに対するデフォルトのチェックタイプも MEDIUM です。指定されている場合、デフォルトは QUICK です。CHANGED および FAST の場合、行はめったに破損しないため、行スキャンはスキップされます。

チェックオプションは、次の例のように組み合わせることができます。この例では、テーブルが正しく閉じられたかどうかを判定するために、そのテーブルに対してすばやいチェックを実行します。

```
CHECK TABLE test_table FAST QUICK;
```

> **注記**
>
> 場合によっては、CHECK TABLE によってテーブルが変更されます。これは、テーブルが「破損している」または「正しく閉じられていない」としてマークされているが、CHECK TABLE でそのテーブル内に何も問題が見つからなかった場合に発生します。この場合、CHECK TABLE はそのテーブルを正常としてマークします。

テーブルが破損している場合、もっとも可能性が高いのはデータ部分ではなく、インデックス内の問題です。前のチェックタイプはすべて、インデックスを徹底的にチェックするため、ほとんどのエラーが見つかるはずです。

正常と見なしているテーブルをチェックするだけの場合は、チェックオプションを使用しないか、または QUICK オプションを使用するようにしてください。後者は、急いでおり、かつ QUICK でデータファイル内のエラーが見つからないという非常に小さなリスクを負える場合に使用するようにしてください。(ほとんどの場合、通常の使用状況では、MySQL でデータファイル内のどのようなエラーも見つかります。見つかった場合、そのテーブルは「破損している」としてマークされ、修復されるまで使用できなくなります。)

FAST および CHANGED は主に、テーブルをときどきチェックする場合にスクリプトから使用される (たとえば、cron から実行される) ことを目的にしています。ほとんどの場合、FAST は CHANGED より優先されます。(優先されない唯一の場合は、MyISAM コード内にバグが見つかったのではないかと疑われるときです。)

EXTENDED は、通常のチェックを実行したが、MySQL が行を更新するか、またはキーで行を検索しようとすると引き続きテーブルで奇妙なエラーが発生する場合にのみ使用されます。通常のチェックが成功した場合、これはめったに発生しません。

CHECK TABLE ... EXTENDED を使用すると、クエリーオプティマイザによって生成される実行計画に影響を与える可能性があります。

CHECK TABLE によってレポートされる次のいくつかの問題は、自動的には修正できません。

- Found row where the auto_increment column has the value 0。

  これは、AUTO_INCREMENT インデックスカラムに値 0 が含まれている行がテーブル内に存在することを示します。(AUTO_INCREMENT カラムが 0 である行は、UPDATE ステートメントを使用してそのカラムを明示的に 0 に設定することによって作成できます。)

  これは、それ自体エラーではありませんが、そのテーブルをダンプしてリストアするか、またはそのテーブルに対して ALTER TABLE を実行しようとした場合に問題が発生する可能性があります。この場合、AUTO_INCREMENT カラムはその AUTO_INCREMENT カラムのルールに従って値を変更するため、重複キーエラーなどの問題が発生する可能性があります。

  この警告を解消するには、単純に、そのカラムを 0 以外の何からの値に設定するために UPDATE ステートメントを実行します。

InnoDB テーブル

次の注意事項は、InnoDB テーブルに適用されます。

- CHECK TABLE で InnoDB テーブルの問題が見つかった場合、サーバーはエラーの伝播を回避するためにシャットダウンする可能性があります。このエラーの詳細はエラーログに書き込まれます。
- CHECK TABLE は、InnoDB テーブルまたはインデックスに破損またはエラーを検出すると、エラーをレポートします。サーバーをシャットダウンすることはしません。MySQL 5.5 からは、そのインデックスまたはテーブルのそれ以上の使用を防ぐために、CHECK TABLE は通常、そのインデックスを破損しているとしてマークし、またそのテーブルも同様にマークする場合があります。
- CHECK TABLE は、セカンダリインデックス内のエントリ数の間違いを見つけた場合、エラーをレポートしますが、サーバーをシャットダウンしたり、ファイルへのアクセスを防いだりはしません。
- CHECK TABLE はインデックスページの構造を調査してから、各キーエントリを調査します。キーポインタをクラスタ化されたレコードに対して検証したり、BLOB ポインタのパスに従ったりはしません。
- InnoDB テーブルが file-per-table モードで独自の .ibd ファイル に格納されると、.ibd の最初の 3 つのページには、テーブルまたはインデックスデータではなくヘッダー情報が含まれます。CHECK TABLE ステートメントは、ヘッダーデータにのみ影響を与える不整合を検出しません。InnoDB .ibd ファイルの内容全体を検証するには、innochecksum コマンドを使用します。
- 大きな InnoDB テーブルに対して CHECK TABLE を実行すると、CHECK TABLE の実行中にほかのスレッドがブロックされる可能性があります。タイムアウトを回避するために、CHECK TABLE 操作の場合は、セマフォー待機のしきい値 (600 秒) が 2 時間 (7200 秒) 延長されます。InnoDB は、240 秒以上のセマフォー待機を検出すると、InnoDB モニターの出力をエラーログに出力し始めます。ロック要求がセマフォー待機のしきい値を超えて延長された場合、InnoDB はそのプロセスを中止します。セマフォー待機が完全にタイムアウトする可能性を回避するには、CHECK TABLE の代わりに CHECK TABLE QUICK を実行できます。

13.7.2.3 CHECKSUM TABLE 構文

```
CHECKSUM TABLE tbl_name [, tbl_name] ... [ QUICK | EXTENDED ]
```

CHECKSUM TABLE は、テーブルの内容に対するチェックサムをレポートします。チェックサム操作中、そのテーブルは InnoDB および MyISAM に対する読み取りロックでロックされます。このステートメントを使用すると、その内容が、バックアップ、ロールバック、またはデータを元の既知の状態に戻すことを目的としたその他の操作の前後で同じであることを検証できます。このステートメントには、このテーブルに対する SELECT 権限が必要です。

パフォーマンスに関する考慮事項

デフォルトでは、テーブル全体が 1 行ごとに読み取られ、チェックサムが計算されます。大きなテーブルでは長い時間がかかる可能性があるため、この操作は、状況に応じてのみ実行されます。この 1 行ごとの計算は、InnoDB や MyISAM 以外のその他のすべてのストレージエンジン、および CHECKSUM=1 句で作成されていない MyISAM テーブルの場合に EXTENDED 句で得られるものと同じです。

CHECKSUM=1 句で作成された MyISAM テーブルの場合、CHECKSUM TABLE または CHECKSUM TABLE ... QUICK は、非常に速く返すことができる「ライブ」テーブルチェックサムを返します。テーブルがこれらのすべての条件を満たさない場合、QUICK による方法は NULL を返します。CHECKSUM 句の構文については、セクション13.1.17「CREATE TABLE 構文」を参照してください。

存在しないテーブルに対しては、CHECKSUM TABLE は NULL を返し、警告を生成します。

MySQL 5.6.4 より前は、EXTENDED オプションが使用されていないかぎり、CHECKSUM TABLE はパーティション化されたテーブルに対して 0 を返しました。(Bug #11933226、Bug #60681)

チェックサム値は、テーブル行フォーマットによって異なります。行フォーマットが変更された場合は、チェックサムも変更されます。たとえば、VARCHAR のストレージフォーマットは MySQL 4.1 と 5.0 の間で変更されたため、4.1 テーブルが MySQL 5.0 にアップグレードされた場合、チェックサム値は変更される可能性があります。

> **重要**
>
> 2 つのテーブルのチェックサムが異なる場合は、それらのテーブルが何らかの点で異なることがほぼ確実です。ただし、CHECKSUM TABLE によって使用されるハッシュ関数は衝突がないことは保証されないため、同一でない 2 つのテーブルが同じチェックサムを生成する可能性が若干あります。

13.7.2.4 OPTIMIZE TABLE 構文

```
OPTIMIZE [NO_WRITE_TO_BINLOG | LOCAL] TABLE
    tbl_name [, tbl_name] ...
```

ストレージ領域を削減し、テーブルアクセス時の I/O 効率を向上させるために、テーブルデータとそれに関連付けられたインデックスデータの物理ストレージを再編成します。各テーブルに加えられる正確な変更は、そのテーブルによって使用されているストレージエンジンによって異なります。

OPTIMIZE TABLE は、テーブルのタイプに応じて次の場合に使用します。

- innodb_file_per_table オプションが有効な状態で作成されたために独自の .ibd ファイル を含む InnoDB テーブルに対して大量の挿入、更新、または削除操作を行なったあと。テーブルとインデックスが再編成されるため、ディスク領域をオペレーティングシステムによる使用のために再利用できます。
- InnoDB テーブル内の FULLTEXT インデックスの一部であるカラムに対して大量の挿入、更新、または削除操作を行なったあと。最初に、構成オプション innodb_optimize_fulltext_only=1 を設定します。インデックスの保守期間を妥当な時間に維持するために、検索インデックスで更新するワード数を指定する innodb_ft_num_word_optimize オプションを設定し、検索インデックスが完全に更新されるまで OPTIMIZE TABLE ステートメントのシーケンスを実行します。
- MyISAM または ARCHIVE テーブルの大きな部分を削除するか、あるいは可変長行を含む MyISAM または ARCHIVE テーブル (VARCHAR、VARBINARY、BLOB、または TEXT カラムを含むテーブル) に多くの変更を行なったあと。削除された行はリンクリスト内に保持され、以降の INSERT 操作は古い行の位置を再利用します。OPTIMIZE TABLE を使用すると、未使用領域を再利用したり、データファイルをデフラグしたりできます。テーブルを大幅に変更したあとは、このステートメントにより、そのテーブルを使用するステートメントのパフォーマンスを (場合によっては大幅に) 向上させることができます。

このステートメントには、このテーブルに対する SELECT および INSERT 権限が必要です。

OPTIMIZE TABLE は、パーティション化されたテーブルでもサポートされます。このステートメントのパーティション化されたテーブルでの使用やテーブルパーティションについては、セクション19.3.4「パーティションの保守」を参照してください。

MySQL 5.6.11 でのみ、このステートメントを発行する前に、gtid_next を AUTOMATIC に設定する必要があります。(Bug #16062608、Bug #16715809、Bug #69045)

OPTIMIZE TABLE は、InnoDB、MyISAM、および ARCHIVE テーブルに対して機能します。OPTIMIZE TABLE は、インメモリー NDB テーブルの動的なカラムに対してもサポートされます。ディスクデータテーブルに対しては機能しません。クラスタテーブルに対する OPTIMIZE のパフォーマンスは、OPTIMIZE TABLE による行のバッチの処理間で待機するミリ秒数を制御する ndb_optimization_delay システム変数の値を調整することによってチューニングできます。詳細は、セクション18.1.6.11「MySQL Cluster NDB 7.3 で解決された以前の MySQL Cluster の問題」を参照してください。

MySQL Cluster テーブルの場合、OPTIMIZE TABLE は、OPTIMIZE 操作を実行している SQL スレッドを (たとえば) 強制終了することによって中断できます。

デフォルトでは、OPTIMIZE TABLE はその他のストレージエンジンを使用して作成されたテーブルに対しては機能せず、このサポートがないことを示す結果を返します。--skip-new オプションを使用して mysqld を起動することによって、その他のストレージエンジンに対して OPTIMIZE TABLE を機能させることができます。この場合、OPTIMIZE TABLE は単に ALTER TABLE にマップされます。

InnoDB の詳細

InnoDB テーブルの場合、OPTIMIZE TABLE は ALTER TABLE ... FORCE にマップされます。これは、インデックス統計を更新し、クラスタ化されたインデックス内の未使用領域を解放するためにテーブルを再構築します。これは、次に示すように、InnoDB テーブルに対して実行したときに OPTIMIZE TABLE の出力に表示されます。

```
mysql> OPTIMIZE TABLE foo;
+----------+----------+----------+-------------------------------------------------------------------+
| Table    | Op       | Msg_type | Msg_text                                                          |
+----------+----------+----------+-------------------------------------------------------------------+
| test.foo | optimize | note     | Table does not support optimize, doing recreate + analyze instead |
| test.foo | optimize | status   | OK                                                                |
+----------+----------+----------+-------------------------------------------------------------------+
```

Mysql 5.6.17 より前は、OPTIMIZE TABLE はオンライン DDL (ALGORITHM=INPLACE) を使用しません。その結果、OPTIMIZE TABLE が実行中のテーブルに対しては (つまり、そのテーブルのロック中は) 並列 DML (INSERT、UPDATE、DELETE) が許可されません。また、主キーに現れる順序でキーが挿入されるため、セカンダリインデックスはそれほど効率的に作成されません。

5.6.17 の時点では、OPTIMIZE TABLE は、InnoDB の通常のテーブルとパーティション化されたテーブルの両方に対してオンライン DDL (ALGORITHM=INPLACE) を使用します。OPTIMIZE TABLE によってトリガーされ、ALTER TABLE ... FORCE の下で実行されるテーブル再構築は現在、オンライン DDL (ALGORITHM=INPLACE) を使用して実行され、短期間しかテーブルをロックしないため、並列 DML 操作のためのダウンタイムが短縮されます。

OPTIMIZE TABLE は引き続き、次の条件の下で ALGORITHM=COPY を使用します。

- old_alter_table システム変数が ON になっている場合。
- mysqld --skip-new オプションが有効になっている場合。

オンライン DDL (ALGORITHM=INPLACE) を使用した OPTIMIZE TABLE は、FULLTEXT インデックスを含む InnoDB テーブルではサポートされません。代わりに ALGORITHM=COPY を使用する必要があります。

InnoDB はページ割り当ての方法を使用してデータを格納するため、従来のストレージエンジン (MyISAM など) の場合のような断片化は発生しません。最適化を実行するかどうかを検討する場合は、サーバーが処理するトランザクションのワークロードを考慮してください。

- ある程度の断片化は予測されます。InnoDB は、ページを分割しなくても更新できる余地を残すために、ページを 93% までしかいっぱいにしません。
- 削除操作によってギャップが残され、ページの空きが目的より多くなることがあります。これにより、テーブルを最適化する価値が生まれる可能性があります。
- 行を更新すると通常、十分な領域が使用可能であれば、データ型と行フォーマットに応じて同じページ内のデータが書き換えられます。セクション14.7.5「InnoDB テーブルでの圧縮の動作」およびセクション14.9.1「InnoDB 行ストレージの概要」を参照してください。
- InnoDB はその MVCC メカニズムのために同じデータの複数のバージョンを保持するため、並列性の高いワークロードでは、時間の経過とともにインデックス内にギャップが残される可能性があります。セクション14.2.12「InnoDB マルチバージョン」を参照してください。

MyISAM の詳細

MyISAM テーブルの場合、OPTIMIZE TABLE は次のように機能します。

1. テーブルが行を削除または分割した場合は、そのテーブルを修復します。
2. インデックスページがソートされていない場合は、それをソートします。
3. テーブルの統計が最新でない (そのため、インデックスのソートによって修復を実行できない) 場合は、それを更新します。

その他の考慮事項

OPTIMIZE TABLE は、次のカラムを含む結果セットを返します。

カラム	値
Table	テーブル名

カラム	値
Op	常に optimize
Msg_type	status、error、info、note、または warning
Msg_text	情報メッセージ

5.6.17 より前の InnoDB テーブルやその他のテーブルタイプの場合、MySQL は、OPTIMIZE TABLE の実行中はそのテーブルをロックします。MySQL 5.6.17 の時点では、OPTIMIZE TABLE は、InnoDB の通常のテーブルとパーティション化されたテーブルに対してオンラインで実行されます。

デフォルトでは、サーバーは OPTIMIZE TABLE ステートメントをバイナリログに書き込み、それらがレプリケーションスレーブにレプリケートされるようにします。ロギングを抑制するには、オプションの NO_WRITE_TO_BINLOG キーワード、またはその エイリアス LOCAL を指定します。

OPTIMIZE TABLE は、POINT カラム上の空間インデックスなどの R ツリーインデックスをソートしません。(Bug #23578)

OPTIMIZE TABLE テーブルは、古いファイルから新しく作成されたファイルへのテーブル統計のコピー中に発生したすべてのエラーをキャッチしてスローします。たとえば、.frm、.MYD、または .MYI ファイルの所有者のユーザー ID が mysqld プロセスのユーザー ID と異なる場合は、mysqld が root ユーザーによって起動されていないかぎり、OPTIMIZE TABLE は "cannot change ownership of the file" エラーを生成します。

13.7.2.5 REPAIR TABLE 構文

```
REPAIR [NO_WRITE_TO_BINLOG | LOCAL] TABLE
    tbl_name [, tbl_name] ...
    [QUICK] [EXTENDED] [USE_FRM]
```

REPAIR TABLE は、特定のストレージエンジンに対してのみ、破損している可能性のあるテーブルを修復します。デフォルトでは、これには myisamchk --recover tbl_name と同じ効果があります。

> **注記**
>
> REPAIR TABLE は MyISAM、ARCHIVE、および CSV の各テーブルのみに適用されます。セクション15.2「MyISAM ストレージエンジン」、セクション15.5「ARCHIVE ストレージエンジン」、およびセクション15.4「CSV ストレージエンジン」を参照してください。

このステートメントには、このテーブルに対する SELECT および INSERT 権限が必要です。

REPAIR TABLE は、パーティション化されたテーブルに対してサポートされています。ただし、パーティション化されたテーブルに対して、このステートメントで USE_FRM オプションを使用することはできません。

MySQL 5.6.11 でのみ、このステートメントを発行する前に、gtid_next を AUTOMATIC に設定する必要があります。(Bug #16062608、Bug #16715809、Bug #69045)

ALTER TABLE ... REPAIR PARTITION を使用すると、1 つ以上のパーティションを修復できます。詳細は、セクション13.1.7「ALTER TABLE 構文」およびセクション19.3.4「パーティションの保守」を参照してください。

通常、REPAIR TABLE を実行する必要はありませんが、災害が発生した場合は、このステートメントを使用すると MyISAM テーブルからすべてのデータをリストアできる可能性があります。テーブルが頻繁に破損する場合は、その原因を見つけることにより、REPAIR TABLE を使用する必要がなくなるようにしてください。セクションB.5.4.2「MySQL が繰り返しクラッシュする場合の対処方法」およびセクション15.2.4「MyISAM テーブルの問題点」を参照してください。

> **注意**
>
> テーブルの修復操作を実行する前に、そのテーブルのバックアップを作成してください。状況によっては、この操作のためにデータ損失が発生することがあります。考えられる原因としては、ファイルシステムのエラーなどがありますがこれに限りません。第7章「バックアップとリカバリ」を参照してください。

> **警告**
>
> REPAIR TABLE 操作中にサーバーがクラッシュした場合は、サーバーを再起動したあと、そのテーブルに対してほかの操作を実行する前に、再度 REPAIR TABLE ステートメントをただちに実行することが重要です。最悪の場合は、データファイルに関する情

> 報のない新しいクリーンなインデックスファイルが生成されており、実行する次の操作によってデータファイルが上書きされる可能性があります。この状況はめったに発生しませんが、最初にバックアップを作成することの価値を強調している、可能性のあるシナリオです。

REPAIR TABLE は、次のカラムを含む結果セットを返します。

カラム	値
Table	テーブル名
Op	常に repair
Msg_type	status、error、info、note、または warning
Msg_text	情報メッセージ

REPAIR TABLE ステートメントによって、修復されたテーブルごとに多数の情報行が生成される可能性があります。最後の行には status の Msg_type 値が含まれ、Msg_test は通常 OK になります。MyISAM テーブルに対して OK が表示されない場合は、myisamchk --safe-recover を使用して修復してみてください。(REPAIR TABLE によって、myisamchk のすべてのオプションが実装されるわけではありません。) myisamchk --safe-recover では、REPAIR TABLE がサポートしていないオプション (--max-record-length など) も使用できます。

QUICK オプションを使用した場合、REPAIR TABLE はデータファイルではなく、インデックスファイルのみを修復しようとします。このタイプの修復は、myisamchk --recover --quick によって実行される修復と同様です。

EXTENDED オプションを使用した場合、MySQL はソートしながら 1 回につき 1 つのインデックスを作成する代わりに、1 行ごとにインデックスを作成します。このタイプの修復は、myisamchk --safe-recover によって実行される修復と同様です。

USE_FRM オプションは、.MYI インデックスファイルがない場合や、そのヘッダーが破損している場合に使用できます。このオプションは MySQL に、.MYI ファイルヘッダー内の情報を信頼せずに、.frm ファイルからの情報を使用してファイルヘッダーを再作成するよう指示します。この種類の修復は、myisamchk では実行できません。

> 注記
>
> 通常の REPAIR モードを使用できない場合は、USE_FRM オプションのみを使用します。サーバーに .MYI ファイルを無視するよう指示すると、.MYI に格納されている重要なテーブルメタデータが修復プロセスから使用できなくなるため、有害な結果を招く場合があります。
>
> - 現在の AUTO_INCREMENT 値は失われます。
> - テーブル内の削除されたレコードへのリンクは失われます。つまり、削除されたレコードの空き領域は、それ以降も占有されずに残ります。
> - .MYI ヘッダーは、テーブルが圧縮されているかどうかを示します。サーバーがこの情報を無視すると、テーブルが圧縮されていることがわからないため、修復によってテーブルの内容の変更または損失が発生する場合があります。つまり、圧縮テーブルでは USE_FRM を使用しないようにしてください。いずれにしても、これは必須ではありません。圧縮テーブルは読み取り専用であるため、破損することはありません。

> 注意
>
> 現在実行しているものとは異なるバージョンの MySQL サーバーによって作成されたテーブルに対して USE_FRM を使用した場合、REPAIR TABLE はそのテーブルを修復しようとしません。この場合、REPAIR TABLE によって返される結果セットには、Msg_type 値が error で、Msg_text 値が Failed repairing incompatible .FRM file である行が含まれています。

USE_FRM が使用されていない場合、REPAIR TABLE はテーブルをチェックして、アップグレードが必要かどうかを確認します。アップグレードが必要な場合は、CHECK TABLE ... FOR UPGRADE と同じルールに従ってアップグレードを実行します。詳細は、セクション13.7.2.2「CHECK TABLE 構文」を参照してください。USE_FRM のない REPAIR TABLE は、.frm ファイルを現在のバージョンにアップグレードします。

デフォルトでは、サーバーは REPAIR TABLE ステートメントをバイナリログに書き込み、それらがレプリケーションスレーブにレプリケートされるようにします。ロギングを抑制するには、オプションの NO_WRITE_TO_BINLOG キーワード、またはその エイリアス LOCAL を指定します。

> **重要**
>
> マスター上のテーブルが破損し、そのテーブルに対して REPAIR TABLE を実行した場合、その結果としての元のテーブルへの変更はスレーブに伝播されません。

特定のシステム変数を設定することによって、REPAIR TABLE のパフォーマンスを向上させることができる可能性があります。セクション8.6.3「REPAIR TABLE ステートメントの速度」を参照してください。

REPAIR TABLE テーブルは、古い破損したファイルから新しく作成されたファイルへのテーブル統計のコピー中に発生したすべてのエラーをキャッチしてスローします。たとえば、.frm、.MYD、または .MYI ファイルの所有者のユーザー ID が mysqld プロセスのユーザー ID と異なる場合は、mysqld が root ユーザーによって起動されていないかぎり、REPAIR TABLE は "cannot change ownership of the file" エラーを生成します。

13.7.3 プラグインおよびユーザー定義関数ステートメント

13.7.3.1 ユーザー定義関数のための CREATE FUNCTION 構文

```
CREATE [AGGREGATE] FUNCTION function_name RETURNS {STRING|INTEGER|REAL|DECIMAL}
    SONAME shared_library_name
```

ユーザー定義関数 (UDF) は、ABS() や CONCAT() などのネイティブな (組み込みの) MySQL 関数のように機能する新しい関数によって MySQL を拡張するための方法です。

function_name は、この関数を呼び出すために SQL ステートメントで使用される名前です。RETURNS 句は、この関数の戻り値の型を示します。DECIMAL は RETURNS のあとの正当な値ですが、現在 DECIMAL 関数は文字列値を返すため、STRING 関数のように記述してください。

shared_library_name は、この関数を実装するコードを含む共有オブジェクトファイルのベース名です。このファイルは、プラグインディレクトリに存在する必要があります。このディレクトリは、plugin_dir システム変数の値から取得できます。詳細は、セクション24.3.2.5「ユーザー定義関数のコンパイルおよびインストール」を参照してください。

関数を作成するには、mysql データベースに対する INSERT 権限が必要です。これが必要なのは、CREATE FUNCTION によって、関数の名前、型、および共有ライブラリ名を記録する mysql.func システムテーブルに行が追加されるためです。このテーブルが存在しない場合は、mysql_upgrade コマンドを実行して作成するようにしてください。セクション4.4.7「mysql_upgrade — MySQL テーブルのチェックとアップグレード」を参照してください。

アクティブ関数とは、CREATE FUNCTION を使用してロードされていて、DROP FUNCTION を使用して削除されていない関数です。すべてのアクティブ関数は、サーバーが起動するたびにリロードされますが、--skip-grant-tables オプションを指定して mysqld を起動した場合は異なります。この場合は、UDF の初期化がスキップされ、UDF は使用できません。

ユーザー定義関数を記述するための手順については、セクション24.3.2「新しいユーザー定義関数の追加」を参照してください。UDF のメカニズムが機能するには、関数を C か C++ (または、C の呼び出し規則を使用できる別の言語) で記述する必要があり、オペレーティングシステムが動的ロードをサポートしている必要があり、さらに mysqld を (静的にではなく) 動的にコンパイルしている必要があります。

AGGREGATE 関数は、SUM や COUNT() などのネイティブな MySQL 集約 (サマリー) 関数とまったく同じように機能します。AGGREGATE が機能するには、mysql.func テーブルに type カラムが含まれている必要があります。mysql.func テーブルにこのカラムが含まれていない場合は、mysql_upgrade プログラムを実行して作成するようにしてください (セクション4.4.7「mysql_upgrade — MySQL テーブルのチェックとアップグレード」を参照してください)。

> **注記**
>
> UDF に関連付けられた共有ライブラリをアップグレードするには、DROP FUNCTION ステートメントを発行し、共有ライブラリをアップグレードしたあと、CREATE FUNCTION ステートメントを発行します。最初に共有ライブラリをアップグレードしてから DROP FUNCTION を使用すると、サーバーがクラッシュする可能性があります。

13.7.3.2 DROP FUNCTION 構文

```
DROP FUNCTION function_name
```

このステートメントは、function_name という名前のユーザー定義関数 (UDF) を削除します。

関数を削除するには、mysql データベースに対する DELETE 権限が必要です。これは、DROP FUNCTION によって、関数の名前、型、および共有ライブラリ名を記録する mysql.func システムテーブルから行が削除されるためです。

> **注記**
>
> UDF に関連付けられた共有ライブラリをアップグレードするには、DROP FUNCTION ステートメントを発行し、共有ライブラリをアップグレードしたあと、CREATE FUNCTION ステートメントを発行します。最初に共有ライブラリをアップグレードしてから DROP FUNCTION を使用すると、サーバーがクラッシュする可能性があります。

DROP FUNCTION はまた、ストアドファンクションを削除するためにも使用されます (セクション 13.1.26「DROP PROCEDURE および DROP FUNCTION 構文」を参照してください)。

13.7.3.3 INSTALL PLUGIN 構文

```
INSTALL PLUGIN plugin_name SONAME 'shared_library_name'
```

このステートメントは、サーバープラグインをインストールします。これには、mysql.plugin テーブルに対する INSERT 権限が必要です。

plugin_name は、ライブラリファイルに含まれているプラグインディスクリプタ構造で定義されているプラグインの名前です (セクション24.2.4.2「プラグインのデータ構造体」を参照してください)。プラグイン名は大文字と小文字が区別されません。プラグイン名は C ソースファイル、シェルコマンド行、M4 および Bourne シェルスクリプト、SQL 環境などで使用されるため、最大化の互換性のために、プラグイン名は ASCII 文字、数字、およびアンダースコアに制限するようにしてください。

shared_library_name は、プラグインコードを含む共有ライブラリの名前です。この名前には、ファイル名拡張子が含まれています (libmyplugin.so、libmyplugin.dll、libmyplugin.dylib など)。

共有ライブラリは、プラグインディレクトリ (plugin_dir システム変数で指定されているディレクトリ) 内に存在する必要があります。このライブラリは、サブディレクトリ内ではなく、プラグインディレクトリ自体に存在する必要があります。デフォルトでは、plugin_dir は pkglibdir 構成変数で指定されているディレクトリの下にある plugin ディレクトリですが、サーバーの起動時に plugin_dir の値を設定することによって変更できます。たとえば、my.cnf ファイル内でその値を設定します。

```
[mysqld]
plugin_dir=/path/to/plugin/directory
```

plugin_dir の値が相対パス名である場合は、MySQL ベースディレクトリ (basedir システム変数の値) を基準にしていると見なされます。

INSTALL PLUGIN は、プラグインを使用可能にするために、そのプラグインコードをロードして初期化します。プラグインは、使用可能になる前にそのプラグインが実行する必要のあるすべての設定を処理するその初期化関数を実行することによって初期化されます。サーバーは、シャットダウン時に、ロードされている各プラグインの初期化解除関数を実行することにより、そのプラグインに最終クリーンアップを実行するための変更が発生するようにします。

INSTALL PLUGIN はまた、そのプラグイン名とライブラリファイル名を示す行を mysql.plugin テーブルに追加することによって、そのプラグインの登録も行います。サーバーの起動時に、サーバーは、mysql.plugin テーブルにリストされているすべてのプラグインをロードして初期化します。つまり、プラグインはサーバーが起動するたびにではなく、1 回だけ INSTALL PLUGIN によってインストールされます。起動時のプラグインのロードは、サーバーが --skip-grant-tables オプションで起動された場合は実行されません。

プラグインライブラリには、複数のプラグインを含めることができます。各プラグインをインストールするには、個別の INSTALL PLUGIN ステートメントを使用します。各ステートメントは異なるプラグインを指定しますが、そのすべてが同じライブラリ名を指定します。

INSTALL PLUGIN を指定すると、サーバーは、サーバーの起動中と同様にオプション (my.cnf) ファイルを読み取ります。これにより、プラグインは、これらのファイルからすべての関連オプションを取得できるようになります。プラグインをロードする前でも、オプションファイルにプラグインオプションを追加できます (loose プリフィクスが使用されている場合)。また、プラグインをアンインストールしたり、my.cnf を編集したり、プラグインを再度インストールしたりすることもできます。プラグインをこの方法で再起動すると、サーバーを再起動することなく新しいオプション値を指定できます。

サーバーの起動時に個々のプラグインロードを制御するオプションについては、セクション5.1.8.1「プラグインのインストールおよびアンインストール」を参照してください。サーバーにシステムテーブルを読み取らないよ

う指示する --skip-grant-tables オプションが指定されたとき、1 回のサーバー起動時にプラグインをロードする必要がある場合は、--plugin-load オプションを使用します。セクション5.1.3「サーバーコマンドオプション」を参照してください。

プラグインを削除するには、UNINSTALL PLUGIN ステートメントを使用します。

プラグインのロードについての追加情報は、セクション5.1.8.1「プラグインのインストールおよびアンインストール」を参照してください。

インストールされているプラグインを確認するには、SHOW PLUGINS ステートメントを使用するか、または INFORMATION_SCHEMA.PLUGINS テーブルにクエリーします。

プラグインライブラリを再コンパイルするとき、それを再インストールする必要がある場合は、次の方法のいずれかを使用できます。

- UNINSTALL PLUGIN を使用してライブラリ内のすべてのプラグインをアンインストールし、新しいプラグインライブラリファイルをプラグインディレクトリにインストールしてから、INSTALL PLUGIN を使用してすべてのプラグインをライブラリにインストールします。この手順には、サーバーを停止することなく使用できるという利点があります。ただし、プラグインライブラリに多数のプラグインが含まれている場合は、多数の INSTALL PLUGIN および UNINSTALL PLUGIN ステートメントを発行する必要があります。
- サーバーを停止し、新しいプラグインライブラリファイルをプラグインディレクトリにインストールしてから、サーバーを再起動します。

13.7.3.4 UNINSTALL PLUGIN 構文

```
UNINSTALL PLUGIN plugin_name
```

このステートメントは、インストールされているサーバープラグインを削除します。これには、mysql.plugin テーブルに対する DELETE 権限が必要です。

plugin_name は、mysql.plugin テーブルにリストされている何らかのプラグインの名前である必要があります。サーバーはプラグインの初期化解除関数を実行し、以降のサーバーの再起動でそのプラグインがロードおよび初期化されないように mysql.plugin テーブルからそのプラグインの行を削除します。UNINSTALL PLUGIN では、そのプラグインの共有ライブラリファイルは削除されません。

プラグインを使用しているテーブルが開いている場合は、そのプラグインをアンインストールできません。

プラグインの削除は、関連付けられたテーブルの使用に影響を与えます。たとえば、全文パーサープラグインがテーブル上の FULLTEXT インデックスに関連付けられている場合は、そのプラグインをアンインストールするとそのテーブルが使用できなくなります。そのテーブルにアクセスしようとすると、エラーが発生します。そのテーブルを開くこともできないため、そのプラグインが使用されているインデックスを削除できません。つまり、テーブルの内容が必要であるかぎり、プラグインのアンインストールは慎重に行う必要があります。あとで再インストールする予定のないプラグインをアンインストールしており、テーブルの内容が必要である場合は、あとでそのテーブルをリロードできるように、そのテーブルを mysqldump でダンプし、ダンプされた CREATE TABLE ステートメントから WITH PARSER 句を削除するようにしてください。テーブルの内容が必要でない場合は、そのテーブルに関連付けられたいずれかのプラグインがない場合でも DROP TABLE を使用できます。

プラグインのロードについての追加情報は、セクション5.1.8.1「プラグインのインストールおよびアンインストール」を参照してください。

13.7.4 SET 構文

```
SET variable_assignment [, variable_assignment] ...

variable_assignment:
      user_var_name = expr
    | [GLOBAL | SESSION] system_var_name = expr
    | [@@GLOBAL. | @@SESSION. | @@]system_var_name = expr
```

SET ステートメントは、サーバーまたはクライアントの操作に影響を与える各種の変数に値を割り当てます。

このセクションでは、変数に値を割り当てるための SET の使用について説明します。SET ステートメントを使用すると、次の種類の変数に値を割り当てることができます。

- システム変数。セクション5.1.4「サーバーシステム変数」を参照してください。システム変数はまた、セクション5.1.5「システム変数の使用」で説明されているように、サーバーの起動時にも設定できます。

ユーザー定義変数。セクション9.4「ユーザー定義変数」を参照してください。

- ストアドプロシージャーやストアドファンクションのパラメータ、およびストアドプログラムのローカル変数。セクション13.6.4「ストアドプログラム内の変数」を参照してください。

ほかのコンテキストでは、SET 構文のいくつかのバリアントが使用されます。

- SET CHARACTER SET と SET NAMES は、サーバーへの接続に関連付けられた文字セットおよび照合順序変数に値を割り当てます。SET ONE_SHOT は、レプリケーションに使用されます。これらのバリアントについては、このセクションのあとの方で説明されています。
- SET PASSWORD は、アカウントのパスワードを割り当てます。セクション13.7.1.7「SET PASSWORD 構文」を参照してください。
- SET TRANSACTION ISOLATION LEVEL は、トランザクション処理の分離レベルを設定します。セクション13.3.6「SET TRANSACTION 構文」を参照してください。

次の説明は、変数を設定するために使用できる各種の SET 構文を示しています。これらの例では = 割り当て演算子を使用していますが、この目的には := 割り当て演算子も使用できます。

ユーザー変数は @var_name として記述され、次のように設定できます。

```
SET @var_name = expr;
```

多くのシステム変数は動的であり、SET ステートメントを利用してサーバーが実行している間に変更できます。リストついては、セクション5.1.5.2「動的システム変数」を参照してください。SET を利用してシステム変数を変更するには、任意で修飾子が先行する var_name としてシステム変数を参照してください。

- 変数がグローバル変数であることを明示的に指示するためには、その名前の前に GLOBAL または @@GLOBAL. を付けます。グローバル変数を設定するには SUPER 権限が必要です。
- 変数がセッション変数であることを明示的に指示するには、その名前の前に SESSION、@@SESSION.、または @@ を付けます。セッション変数を正常に設定するために特殊な権限は必要ありません。ただし、例外があります (sql_log_bin など)。クライアントは自分のセッション変数を変更できますが、ほかのどのクライアントのセッション変数も変更できません。
- LOCAL と @@LOCAL. は SESSION と @@SESSION. のシノニムです。
- 修飾子が何もなければ、SET はセッション変数を変更します。

SET ステートメントは、カンマで区切られた複数の変数割り当てを含むことができます。たとえば、ステートメントは、ユーザー定義変数やシステム変数に値を割り当てることができます。複数のシステム変数を設定した場合、ステートメント内のいちばん最近の GLOBAL または SESSION 修飾子が、指定された修飾子を持たない後続の変数に利用されます。

例:

```
SET sort_buffer_size=10000;
SET @@LOCAL.sort_buffer_size=10000;
SET GLOBAL sort_buffer_size=1000000, SESSION sort_buffer_size=1000000;
SET @@sort_buffer_size=1000000;
SET @@GLOBAL.sort_buffer_size=1000000, @@LOCAL.sort_buffer_size=1000000;
```

システム変数の @@var_name 構文では、ほかの一部のデータベースシステムとの互換性をサポートしています。

セッションシステム変数を変更すると、セッションが終了するまで、または変数を異なる値に変更するまではその値は有効になります。別のクライアントは変更を見ることができません。

グローバルシステム変数を変更すると、その値はサーバーが再起動するまでの間記憶され、新しい接続に利用されます。(グローバルシステム変数を永続的に設定するには、オプションファイルに設定する必要があります。) そのグローバル変数にアクセスするすべてのクライアントが変更を確認できます。ただし変更は、変更後に接続するクライアントの対応するセッション変数にのみ影響を与えます。グローバル変数の変更は、現在接続中のクライアントのセッション変数に影響を与えません (SET GLOBAL ステートメントを発行するクライアントのセッション変数にも影響を与えません)。

誤用を防ぐために、SET SESSION とのみ利用できる変数とともに SET GLOBAL を使用したり、グローバル変数の設定時に GLOBAL (または @@GLOBAL.) を指定しなかったりした場合に、MySQL でエラーが生じます。

SESSION 変数を GLOBAL 値に設定したり、GLOBAL 値をコンパイル時の MySQL のデフォルト値に設定したりするには、DEFAULT キーワードを使用します。たとえば、次の 2 つのステートメントは、max_join_size のセッション値をグローバル値に設定する上で同一です。

```
SET max_join_size=DEFAULT;
SET @@SESSION.max_join_size=@@GLOBAL.max_join_size;
```

すべてのシステム変数を DEFAULT に設定できるわけではありません。そのような場合、DEFAULT を使用すると エラーが発生します。

ユーザー定義変数、ストアドプロシージャーやストアドファンクションのパラメータ、またはストアドプログラムのローカル変数に値 DEFAULT を割り当てることは許可されません。ユーザー定義変数の場合、これは構文エラーになります。また、MySQL 5.6.6 の時点では、パラメータまたはローカル変数の場合も同様です。

いずれかの @@ 修飾子を使用することによって、特定のグローバルシステム変数またはセッションシステム変数の値を式で参照できます。たとえば、次のようにして SELECT ステートメントで値を取得できます。

```
SELECT @@GLOBAL.sql_mode, @@SESSION.sql_mode, @@sql_mode;
```

@@var_name のような式でシステム変数を参照するとき (つまり、@@GLOBAL. または @@SESSION. を指定しない場合)、MySQL はセッション値が存在すればそれを返し、それ以外の場合はグローバル値を返します。(これは、常にセッション値を参照する SET @@var_name = value とは異なります。)

> **注記**
>
> SHOW VARIABLES によって表示される一部の変数は、SELECT @@var_name 構文で使用できない場合があり、「不明なシステム変数です」と表示されます。その場合の回避方法として、SHOW VARIABLES LIKE 'var_name' を使用できます。

値乗数を指定するサフィクスは、サーバーの起動時に変数を設定するときに使用できますが、実行時に SET で値を設定するためには使用できません。一方、SET を使用すると、式を使用して変数の値を割り当てることができますが、サーバーの起動時に変数を設定するときには使用できません。たとえば、サーバーの起動時に次の 1 行目は有効ですが 2 行目は無効です。

```
shell> mysql --max_allowed_packet=16M
shell> mysql --max_allowed_packet=16*1024*1024
```

逆に、実行時に次の 2 行目は有効ですが 1 行目は無効です。

```
mysql> SET GLOBAL max_allowed_packet=16M;
mysql> SET GLOBAL max_allowed_packet=16*1024*1024;
```

システム変数の名前と値を表示するには、SHOW VARIABLES ステートメントを使用します。(セクション 13.7.5.40「SHOW VARIABLES 構文」を参照してください。)

次のリストは、標準以外の構文を持つ SET オプション (つまり、name = value 構文では設定されないオプション) を示しています。

- CHARACTER SET {charset_name | DEFAULT}

  これは、マッピングが指定されているすべての文字列をクライアントとの間でマップします。MySQL ソース配布内の sql/convert.cc を編集することによって、新しいマッピングを追加できます。SET CHARACTER SET は、3 つのセッションシステム変数を設定します。character_set_client と character_set_results は指定された文字セットに設定され、character_set_connection は character_set_database の値に設定されます。セクション 10.1.4「接続文字セットおよび照合順序」を参照してください。

  デフォルトのマッピングは、値 DEFAULT を使用してリストアできます。このデフォルトは、サーバー構成によって異なります。

  ucs2、utf16、および utf32 は、クライアント文字セットとして使用できません。つまり、これらは SET CHARACTER SET では機能しません。

- NAMES {'charset_name' [COLLATE 'collation_name'] | DEFAULT}

  SET NAMES は、3 つのセッションシステム変数 character_set_client、character_set_connection、および character_set_results を指定された文字セットに設定します。character_set_connection を charset_name に設定すると、collation_connection も charset_name のデフォルトの照合順序に設定されます。オプションの COLLATE 句を使用すると、照合順序を明示的に指定できます。セクション10.1.4「接続文字セットおよび照合順序」を参照してください。

デフォルトのマッピングは、DEFAULT の値を使用してリストアできます。このデフォルトは、サーバー構成によって異なります。

ucs2、utf16、および utf32 は、クライアント文字セットとして使用できません。つまり、これらは SET NAMES では機能しません。

- ONE_SHOT

  ONE_SHOT は、内部でのみ使用されます。これは、MySQL 5.0 から非推奨であり、MySQL 5.6.1 で削除されました。

13.7.5 SHOW 構文

SHOW には、データベース、テーブル、カラムに関する情報、またはサーバーに関するステータス情報を提供するための多くの形式があります。このセクションでは、次のものについて説明します。

```
SHOW AUTHORS
SHOW {BINARY | MASTER} LOGS
SHOW BINLOG EVENTS [IN 'log_name'] [FROM pos] [LIMIT [offset,] row_count]
SHOW CHARACTER SET [like_or_where]
SHOW COLLATION [like_or_where]
SHOW [FULL] COLUMNS FROM tbl_name [FROM db_name] [like_or_where]
SHOW CONTRIBUTORS
SHOW CREATE DATABASE db_name
SHOW CREATE EVENT event_name
SHOW CREATE FUNCTION func_name
SHOW CREATE PROCEDURE proc_name
SHOW CREATE TABLE tbl_name
SHOW CREATE TRIGGER trigger_name
SHOW CREATE VIEW view_name
SHOW DATABASES [like_or_where]
SHOW ENGINE engine_name {STATUS | MUTEX}
SHOW [STORAGE] ENGINES
SHOW ERRORS [LIMIT [offset,] row_count]
SHOW EVENTS
SHOW FUNCTION CODE func_name
SHOW FUNCTION STATUS [like_or_where]
SHOW GRANTS FOR user
SHOW INDEX FROM tbl_name [FROM db_name]
SHOW MASTER STATUS
SHOW OPEN TABLES [FROM db_name] [like_or_where]
SHOW PLUGINS
SHOW PROCEDURE CODE proc_name
SHOW PROCEDURE STATUS [like_or_where]
SHOW PRIVILEGES
SHOW [FULL] PROCESSLIST
SHOW PROFILE [types] [FOR QUERY n] [OFFSET n] [LIMIT n]
SHOW PROFILES
SHOW SLAVE HOSTS
SHOW SLAVE STATUS
SHOW [GLOBAL | SESSION] STATUS [like_or_where]
SHOW TABLE STATUS [FROM db_name] [like_or_where]
SHOW [FULL] TABLES [FROM db_name] [like_or_where]
SHOW TRIGGERS [FROM db_name] [like_or_where]
SHOW [GLOBAL | SESSION] VARIABLES [like_or_where]
SHOW WARNINGS [LIMIT [offset,] row_count]

like_or_where:
    LIKE 'pattern'
  | WHERE expr
```

特定の SHOW ステートメントの構文に LIKE 'pattern' 部分が含まれている場合、'pattern' は SQL の「%」と「_」のワイルドカード文字を含むことのできる文字列です。このパターンは、ステートメント出力を一致する値に制限するために役立ちます。

いくつかの SHOW ステートメントは、どの行を表示するかをより柔軟に指定できる WHERE 句も受け入れます。セクション21.32「SHOW ステートメントの拡張」を参照してください。

多くの MySQL API (PHP など) では、SHOW ステートメントから返された結果を SELECT からの結果セットのように処理できます。詳細は、第23章「Connector および API」または API のドキュメントを参照してください。さらに、SQL では、INFORMATION_SCHEMA データベース内のテーブルに対するクエリーからの結果を操作できます。これは、SHOW ステートメントからの結果では簡単にはできません。第21章「INFORMATION_SCHEMA テーブル」を参照してください。

13.7.5.1 SHOW AUTHORS 構文

```
SHOW AUTHORS
```

SHOW AUTHORS ステートメントは、MySQL に関して作業している人びとに関する情報を表示します。作成者ごとに、Name、Location、および Comment 値を表示します。

このステートメントは、MySQL 5.6.8 の時点で削除されています。

13.7.5.2 SHOW BINARY LOGS 構文

```
SHOW BINARY LOGS
SHOW MASTER LOGS
```

サーバー上のバイナリログファイルを一覧表示します。このステートメントは、どのログをパージできるかを決定する方法を示す、セクション13.4.1.1「PURGE BINARY LOGS 構文」で説明されている手順の一部として使用されます。

```
mysql> SHOW BINARY LOGS;
+---------------+-----------+
| Log_name      | File_size |
+---------------+-----------+
| binlog.000015 |    724935 |
| binlog.000016 |    733481 |
+---------------+-----------+
```

SHOW MASTER LOGS は SHOW BINARY LOGS と同等です。

MySQL 5.6.5 以前では、このステートメントを使用するには SUPER 権限が必要でした。MySQL 5.6.6 からは、REPLICATION CLIENT 権限を持つユーザーもこのステートメントを実行できます。

13.7.5.3 SHOW BINLOG EVENTS 構文

```
SHOW BINLOG EVENTS
   [IN 'log_name'] [FROM pos] [LIMIT [offset,] row_count]
```

バイナリログ内のイベントを表示します。'log_name' を指定しない場合は、最初のバイナリログが表示されます。

LIMIT 句の構文は、SELECT ステートメントの場合と同じです。セクション13.2.9「SELECT 構文」を参照してください。

> 注記
>
> SHOW BINLOG EVENTS を LIMIT 句なしで発行すると、サーバーはクライアントに (サーバーによって実行された、データを変更するすべてのステートメントを含む) バイナリログの完全な内容を返すため、時間とリソースを大量に消費するプロセスが開始される可能性があります。あとの調査や分析のためにバイナリログをテキストファイルに保存するには、SHOW BINLOG EVENTS の代わりに mysqlbinlog ユーティリティーを使用してください。セクション4.6.8「mysqlbinlog — バイナリログファイルを処理するためのユーティリティー」を参照してください。

> 注記
>
> SHOW BINLOG EVENTS からの出力には、ユーザーおよびシステム変数の設定に関連した一部のイベントが含まれていません。バイナリログ内のイベントを完全に取得するには、mysqlbinlog を使用します。

> 注記
>
> SHOW BINLOG EVENTS は、リレーログファイルを操作しません。この目的には、SHOW RELAYLOG EVENTS を使用できます。

13.7.5.4 SHOW CHARACTER SET 構文

```
SHOW CHARACTER SET
   [LIKE 'pattern' | WHERE expr]
```

SHOW CHARACTER SET ステートメントは使用可能な文字セットをすべて表示します。LIKE 句 (存在する場合) は、どの文字セット名と照合するかを示します。セクション21.32「SHOW ステートメントの拡張」で説明されているように、WHERE 句を指定すると、より一般的な条件を使用して行を選択できます。例:

```
mysql> SHOW CHARACTER SET LIKE 'latin%';
+---------+-----------------------------+-------------------+--------+
| Charset | Description                 | Default collation | Maxlen |
+---------+-----------------------------+-------------------+--------+
| latin1  | cp1252 West European        | latin1_swedish_ci |      1 |
| latin2  | ISO 8859-2 Central European | latin2_general_ci |      1 |
| latin5  | ISO 8859-9 Turkish          | latin5_turkish_ci |      1 |
| latin7  | ISO 8859-13 Baltic          | latin7_general_ci |      1 |
+---------+-----------------------------+-------------------+--------+
```

Maxlen カラムは、1 文字を格納するために必要な最大バイト数を表示します。

filename 文字セットは、内部でのみ使用されます。そのため、SHOW CHARACTER SET では表示されません。

13.7.5.5 SHOW COLLATION 構文

```
SHOW COLLATION
    [LIKE 'pattern' | WHERE expr]
```

このステートメントは、サーバーによってサポートされる照合順序を一覧表示します。デフォルトでは、SHOW COLLATION からの出力には、使用可能なすべての照合順序が含まれます。LIKE 句 (存在する場合) は、どの照合順序名と照合するかを示します。セクション21.32「SHOW ステートメントの拡張」で説明されているように、WHERE 句を指定すると、より一般的な条件を使用して行を選択できます。例:

```
mysql> SHOW COLLATION LIKE 'latin1%';
+-------------------+---------+----+---------+----------+---------+
| Collation         | Charset | Id | Default | Compiled | Sortlen |
+-------------------+---------+----+---------+----------+---------+
| latin1_german1_ci | latin1  |  5 |         |          |       0 |
| latin1_swedish_ci | latin1  |  8 | Yes     | Yes      |       0 |
| latin1_danish_ci  | latin1  | 15 |         |          |       0 |
| latin1_german2_ci | latin1  | 31 |         | Yes      |       2 |
| latin1_bin        | latin1  | 47 |         | Yes      |       0 |
| latin1_general_ci | latin1  | 48 |         |          |       0 |
| latin1_general_cs | latin1  | 49 |         |          |       0 |
| latin1_spanish_ci | latin1  | 94 |         |          |       0 |
+-------------------+---------+----+---------+----------+---------+
```

Collation および Charset カラムは、照合順序の名前と、その照合順序が関連付けられている文字セットを示します。Id は照合順序 ID です。Default は、この照合順序がその文字セットのデフォルトであるかどうかを示します。Compiled は、この文字セットがサーバーにコンパイルされているかどうかを示します。Sortlen は、この文字セットで表される文字列をソートするために必要なメモリーの量に関連しています。

各文字セットのデフォルトの照合順序を表示するには、次のステートメントを使用します。Default は予約語であるため、それを識別子として使用するには、次のように引用符で囲む必要があります。

```
mysql> SHOW COLLATION WHERE `Default` = 'Yes';
+---------------------+----------+----+---------+----------+---------+
| Collation           | Charset  | Id | Default | Compiled | Sortlen |
+---------------------+----------+----+---------+----------+---------+
| big5_chinese_ci     | big5     |  1 | Yes     | Yes      |       1 |
| dec8_swedish_ci     | dec8     |  3 | Yes     | Yes      |       1 |
| cp850_general_ci    | cp850    |  4 | Yes     | Yes      |       1 |
| hp8_english_ci      | hp8      |  6 | Yes     | Yes      |       1 |
| koi8r_general_ci    | koi8r    |  7 | Yes     | Yes      |       1 |
| latin1_swedish_ci   | latin1   |  8 | Yes     | Yes      |       1 |
...
```

13.7.5.6 SHOW COLUMNS 構文

```
SHOW [FULL] COLUMNS {FROM | IN} tbl_name [{FROM | IN} db_name]
    [LIKE 'pattern' | WHERE expr]
```

SHOW COLUMNS は、特定のテーブル内のカラムに関する情報を表示します。これはビューに対しても機能します。LIKE 句 (存在する場合) は、どのカラム名と照合するかを示します。セクション21.32「SHOW ステートメントの拡張」で説明されているように、WHERE 句を指定すると、より一般的な条件を使用して行を選択できます。

SHOW COLUMNS は、ユーザーが何らかの権限を持っているカラムの情報のみを表示します。

```
mysql> SHOW COLUMNS FROM City;
+------------+----------+------+-----+---------+----------------+
| Field      | Type     | Null | Key | Default | Extra          |
```

```
+------------+----------+------+-----+---------+----------------+
| Id         | int(11)  | NO   | PRI | NULL    | auto_increment |
| Name       | char(35) | NO   |     |         |                |
| Country    | char(3)  | NO   | UNI |         |                |
| District   | char(20) | YES  | MUL |         |                |
| Population | int(11)  | NO   |     | 0       |                |
+------------+----------+------+-----+---------+----------------+
5 rows in set (0.00 sec)
```

データ型が、CREATE TABLE ステートメントに基づいて予測されるものと異なる場合は、テーブルの作成または変更時に MySQL によってデータ型が変更される場合があります。この状態が発生する条件は、セクション 13.1.17.3「暗黙のカラム指定の変更」で説明されています。

FULL キーワードを指定すると、カラムの照合順序とコメント、およびユーザーが各カラムに対して持っている権限が出力に含まれます。

tbl_name FROM db_name 構文の代わりに db_name.tbl_name を使用できます。つまり、次の 2 つのステートメントは同等です。

```
mysql> SHOW COLUMNS FROM mytable FROM mydb;
mysql> SHOW COLUMNS FROM mydb.mytable;
```

SHOW COLUMNS は、テーブルカラムごとに次の値を表示します。

Field は、カラム名を示します。

Type は、カラムデータ型を示します。

Collation は、バイナリ以外の文字列カラムの場合は照合順序、その他のカラムの場合は NULL を示します。この値は、FULL キーワードを使用した場合にのみ表示されます。

Null フィールドには、このカラムに NULL 値を格納できる場合は YES、できない場合は NO が含まれます。

Key フィールドは、このカラムがインデックス設定されているかどうかを示します。

- Key が空の場合、このカラムはインデックス設定されていないか、またはマルチカラム内のセカンダリカラム (一意でないインデックス) としてのみインデックス設定されているかのどちらかです。
- Key が PRI の場合、このカラムは PRIMARY KEY であるか、またはマルチカラム PRIMARY KEY 内のいずれかのカラムです。
- Key が UNI の場合、このカラムは UNIQUE インデックスの最初のカラムです。(UNIQUE インデックスは複数の NULL 値を許可しますが、そのカラムが NULL を許可するかどうかは Null フィールドをチェックすることによってわかります。)
- Key が MUL の場合、このカラムは、特定の値がカラム内に複数回現れることが許可されている一意でないインデックスの最初のカラムです。

テーブルの特定のカラムに複数の Key 値が適用される場合、Key には、もっとも優先度の高い値が PRI、UNI、MUL の順序で表示されます。

UNIQUE インデックスは、NULL 値を含むことができず、かつテーブル内に PRIMARY KEY が存在しない場合は PRI として表示される可能性があります。UNIQUE インデックスは、複数のカラムが複合 UNIQUE インデックスを形成している場合は MUL として表示される可能性があります。このカラムの組み合わせは一意であるにもかかわらず、各カラムには引き続き、特定の値が複数回現れることがあります。

Default フィールドは、このカラムに割り当てられているデフォルト値を示します。このカラムの明示的なデフォルト値が NULL である場合や、カラム定義に DEFAULT 句が含まれていない場合、これは NULL です。

Extra フィールドには、特定のカラムに関して使用可能な任意の追加情報が含まれます。この値が空以外になるのは、AUTO_INCREMENT 属性を持つカラムに対する auto_increment と、ON UPDATE CURRENT_TIMESTAMP 属性を持つ TIMESTAMP または DATETIME カラムに対する on update CURRENT_TIMESTAMP です。

Privileges は、ユーザーがこのカラムに対して持っている権限を示します。この値は、FULL キーワードを使用した場合にのみ表示されます。

Comment は、このカラムに含まれている任意のコメントを示します。この値は、FULL キーワードを使用した場合にのみ表示されます。

SHOW FIELDS は SHOW COLUMNS のシノニムです。また、mysqlshow db_name tbl_name コマンドを使用してテーブルのカラムを一覧表示することもできます。

DESCRIBE ステートメントは、SHOW COLUMNS と同様の情報を提供します。セクション13.8.1「DESCRIBE 構文」を参照してください。

また、SHOW CREATE TABLE、SHOW TABLE STATUS、および SHOW INDEX ステートメントでは、テーブルに関する情報も提供されます。セクション13.7.5「SHOW 構文」を参照してください。

13.7.5.7 SHOW CONTRIBUTORS 構文

```
SHOW CONTRIBUTORS
```

SHOW CONTRIBUTORS ステートメントは、MySQL ソースまたはサポートされている原因に貢献している人びとに関する情報を表示します。貢献者ごとに、Name、Location、および Comment 値を表示します。

このステートメントは、MySQL 5.6.8 の時点で削除されています。

13.7.5.8 SHOW CREATE DATABASE 構文

```
SHOW CREATE {DATABASE | SCHEMA} [IF NOT EXISTS] db_name
```

指定されたデータベースを作成する CREATE DATABASE ステートメントを表示します。SHOW ステートメントに IF NOT EXISTS 句が含まれている場合は、このような句が出力にも含まれます。SHOW CREATE SCHEMA は SHOW CREATE DATABASE のシノニムです。

```
mysql> SHOW CREATE DATABASE test\G
*************************** 1. row ***************************
       Database: test
Create Database: CREATE DATABASE `test`
                 /*!40100 DEFAULT CHARACTER SET latin1 */

mysql> SHOW CREATE SCHEMA test\G
*************************** 1. row ***************************
       Database: test
Create Database: CREATE DATABASE `test`
                 /*!40100 DEFAULT CHARACTER SET latin1 */
```

SHOW CREATE DATABASE は、sql_quote_show_create オプションの値に従って、テーブル名とカラム名を引用符で囲みます。セクション5.1.4「サーバーシステム変数」を参照してください。

13.7.5.9 SHOW CREATE EVENT 構文

```
SHOW CREATE EVENT event_name
```

このステートメントは、特定のイベントを再作成するために必要な CREATE EVENT ステートメントを表示します。これには、このイベントが示される元のデータベースに対する EVENT 権限が必要です。例 (セクション13.7.5.19「SHOW EVENTS 構文」で定義され、あとで変更された同じイベント e_daily を使用しています):

```
mysql> SHOW CREATE EVENT test.e_daily\G
*************************** 1. row ***************************
               Event: e_daily
            sql_mode:
           time_zone: SYSTEM
      Create Event: CREATE EVENT `e_daily`
                    ON SCHEDULE EVERY 1 DAY
                    STARTS CURRENT_TIMESTAMP + INTERVAL 6 HOUR
                    ON COMPLETION NOT PRESERVE
                    ENABLE
                    COMMENT 'Saves total number of sessions then
                         clears the table each day'
                    DO BEGIN
                      INSERT INTO site_activity.totals (time, total)
                        SELECT CURRENT_TIMESTAMP, COUNT(*)
                        FROM site_activity.sessions;
                      DELETE FROM site_activity.sessions;
                    END
character_set_client: latin1
collation_connection: latin1_swedish_ci
  Database Collation: latin1_swedish_ci
```

character_set_client は、このイベントが作成されたときの character_set_client システム変数のセッション値です。collation_connection は、このイベントが作成されたときの collation_connection システム変数のセッション値です。Database Collation は、このイベントが関連付けられているデータベースの照合順序です。

この出力には、このイベントが作成されたときのステータスではなく、その現在のステータス (ENABLE) が反映されます。

13.7.5.10 SHOW CREATE FUNCTION 構文

```
SHOW CREATE FUNCTION func_name
```

このステートメントは、ストアドファンクションである点を除き、SHOW CREATE PROCEDURE と同じです。セクション13.7.5.11「SHOW CREATE PROCEDURE 構文」を参照してください。

13.7.5.11 SHOW CREATE PROCEDURE 構文

```
SHOW CREATE PROCEDURE proc_name
```

このステートメントは、MySQL 拡張です。これは、指定されたストアドプロシージャーを再作成するために使用できる正確な文字列を返します。同様のステートメントである SHOW CREATE FUNCTION は、ストアドファンクションに関する情報を表示します (セクション13.7.5.10「SHOW CREATE FUNCTION 構文」を参照してください)。

どちらのステートメントを使用するにも、このルーチンの DEFINER 句で指定されたユーザーであるか、または mysql.proc テーブルへの SELECT アクセス権を持っている必要があります。このルーチン自体に対する権限を持っていない場合、Create Procedure または Create Function フィールドに表示される値は NULL になります。

```
mysql> SHOW CREATE PROCEDURE test.simpleproc\G
*************************** 1. row ***************************
           Procedure: simpleproc
            sql_mode:
    Create Procedure: CREATE PROCEDURE `simpleproc`(OUT param1 INT)
                      BEGIN
                      SELECT COUNT(*) INTO param1 FROM t;
                      END
character_set_client: latin1
collation_connection: latin1_swedish_ci
  Database Collation: latin1_swedish_ci

mysql> SHOW CREATE FUNCTION test.hello\G
*************************** 1. row ***************************
            Function: hello
            sql_mode:
     Create Function: CREATE FUNCTION `hello`(s CHAR(20))
                      RETURNS CHAR(50)
                      RETURN CONCAT('Hello, ',s,'!')
character_set_client: latin1
collation_connection: latin1_swedish_ci
  Database Collation: latin1_swedish_ci
```

character_set_client は、このルーチンが作成されたときの character_set_client システム変数のセッション値です。collation_connection は、このルーチンが作成されたときの collation_connection システム変数のセッション値です。Database Collation は、このルーチンが関連付けられているデータベースの照合順序です。

13.7.5.12 SHOW CREATE TABLE 構文

```
SHOW CREATE TABLE tbl_name
```

指定されたテーブルを作成する CREATE TABLE ステートメントを表示します。このステートメントを使用するには、そのテーブルに対する何らかの権限が必要です。また、このステートメントはビューでも機能します。

```
mysql> SHOW CREATE TABLE t\G
*************************** 1. row ***************************
       Table: t
Create Table: CREATE TABLE t (
  id INT(11) default NULL auto_increment,
  s char(60) default NULL,
  PRIMARY KEY (id)
) ENGINE=MyISAM
```

SHOW CREATE TABLE は、sql_quote_show_create オプションの値に従って、テーブル名とカラム名を引用符で囲みます。セクション5.1.4「サーバーシステム変数」を参照してください。

13.7.5.13 SHOW CREATE TRIGGER 構文

```
SHOW CREATE TRIGGER trigger_name
```

このステートメントは、指定されたトリガーを作成する CREATE TRIGGER ステートメントを表示します。

```
mysql> SHOW CREATE TRIGGER ins_sum\G
*************************** 1. row ***************************
             Trigger: ins_sum
            sql_mode: NO_ENGINE_SUBSTITUTION
SQL Original Statement: CREATE DEFINER=`me`@`localhost` TRIGGER ins_sum
                        BEFORE INSERT ON account
                        FOR EACH ROW SET @sum = @sum + NEW.amount
  character_set_client: utf8
  collation_connection: utf8_general_ci
    Database Collation: latin1_swedish_ci
```

SHOW CREATE TRIGGER の出力には、次のカラムがあります。

- Trigger: トリガー名。
- sql_mode: このトリガーが実行されるときに有効な SQL モード。
- SQL Original Statement: このトリガーを定義する CREATE TRIGGER ステートメント。
- character_set_client: このトリガーが作成されたときの character_set_client システム変数のセッション値。
- collation_connection: このトリガーが作成されたときの collation_connection システム変数のセッション値。
- Database Collation: このトリガーが関連付けられているデータベースの照合順序。

また、TRIGGERS テーブルを含む INFORMATION_SCHEMA からトリガーオブジェクトに関する情報を取得することもできます。セクション21.26「INFORMATION_SCHEMA TRIGGERS テーブル」を参照してください。

13.7.5.14 SHOW CREATE VIEW 構文

```
SHOW CREATE VIEW view_name
```

このステートメントは、指定されたビューを作成する CREATE VIEW ステートメントを表示します。

```
mysql> SHOW CREATE VIEW v\G
*************************** 1. row ***************************
                View: v
         Create View: CREATE ALGORITHM=UNDEFINED
                      DEFINER=`bob`@`localhost`
                      SQL SECURITY DEFINER VIEW
                      `v` AS select 1 AS `a`,2 AS `b`
character_set_client: latin1
collation_connection: latin1_swedish_ci
```

character_set_client は、このビューが作成されたときの character_set_client システム変数のセッション値です。collation_connection は、このビューが作成されたときの collation_connection システム変数のセッション値です。

SHOW CREATE VIEW を使用するには、SHOW VIEW 権限、および対象のビューに対する SELECT 権限が必要です。

また、VIEWS テーブルを含む INFORMATION_SCHEMA からビューオブジェクトに関する情報を取得することもできます。セクション21.28「INFORMATION_SCHEMA VIEWS テーブル」を参照してください。

MySQL では、異なる sql_mode 設定を使用すると、サポートする SQL 構文のタイプをサーバーに指示できます。たとえば、ANSI SQL モードを使用すると、クエリーで、MySQL で標準 SQL 連結演算子の二重バー (||) が正しく解釈されます。その後、項目を連結するビューを作成した場合、sql_mode 設定を ANSI とは別の値に変更すると、そのビューが無効になるという懸念がある場合があります。ただし、そのようなことはありません。MySQL は、記述方法には関係なく、常にビュー定義を正規の形式で同じ方法で格納します。サーバーが二重バーの連結演算子を CONCAT() 関数にどのように変更するかを示す例を次に示します。

```
mysql> SET sql_mode = 'ANSI';
Query OK, 0 rows affected (0.00 sec)

mysql> CREATE VIEW test.v AS SELECT 'a' || 'b' as col1;
Query OK, 0 rows affected (0.01 sec)

mysql> SHOW CREATE VIEW test.v\G
*************************** 1. row ***************************
                View: v
```

```
       Create View: CREATE VIEW "v" AS select concat('a','b') AS "col1"
...
1 row in set (0.00 sec)
```

ビュー定義を正規の形式で格納する利点は、あとで sql_mode の値を変更しても、ビューの結果に影響を与えないことにあります。ただし、SELECT の前にあるコメントが、サーバーによって定義から取り除かれるというその他の影響があります。

13.7.5.15 SHOW DATABASES 構文

```
SHOW {DATABASES | SCHEMAS}
    [LIKE 'pattern' | WHERE expr]
```

SHOW DATABASES は、MySQL サーバーホスト上のデータベースを一覧表示します。SHOW SCHEMAS は SHOW DATABASES のシノニムです。LIKE 句 (存在する場合) は、どのデータベース名と照合するかを示します。セクション21.32「SHOW ステートメントの拡張」で説明されているように、WHERE 句を指定すると、より一般的な条件を使用して行を選択できます。

グローバルな SHOW DATABASES 権限を持っていないかぎり、何らかの種類の権限を持っているデータベースしか表示できません。このリストはまた、mysqlshow コマンドを使用して取得することもできます。

サーバーが --skip-show-database オプションで起動された場合は、SHOW DATABASES 権限を持っていないかぎり、このステートメントをまったく使用できません。

MySQL はデータベースをデータディレクトリ内のディレクトリとして実装するため、このステートメントは単純に、その場所にあるディレクトリを一覧表示します。ただし、実際のデータベースには対応しないディレクトリの名前が出力に含まれる可能性があります。

13.7.5.16 SHOW ENGINE 構文

```
SHOW ENGINE engine_name {STATUS | MUTEX}
```

SHOW ENGINE は、ストレージエンジンに関する動作情報を表示します。これには PROCESS 権限が必要です。このステートメントは、次のバリアントがあります。

```
SHOW ENGINE INNODB STATUS
SHOW ENGINE INNODB MUTEX
SHOW ENGINE {NDB | NDBCLUSTER} STATUS
SHOW ENGINE PERFORMANCE_SCHEMA STATUS
```

SHOW ENGINE INNODB STATUS は、InnoDB ストレージエンジンの状態に関する InnoDB 標準モニターからの広範囲にわたる情報を表示します。InnoDB の処理に関する情報を提供する標準モニターやその他の InnoDB モニターについては、セクション14.15「InnoDB モニター」を参照してください。

SHOW ENGINE INNODB MUTEX は、InnoDB 相互排他ロックおよび読み書きロックの統計を表示します。ステートメント出力には、次のカラムがあります。

> **注記**
>
> ほとんどの SHOW ENGINE INNODB MUTEX 出力は 5.6.14 で削除されます。SHOW ENGINE INNODB MUTEX 出力は、MySQL 5.7.2 で完全に削除されます。パフォーマンススキーマテーブル上にビューを作成することによって、比較可能な情報を生成できます。

- Type

  常に InnoDB です。

- Name

  相互排他ロックが実装されているソースファイルと、そのファイル内の相互排他ロックが作成されている行番号。この行番号は、使用している MySQL のバージョンに固有です。

- Status

  相互排他ロックのステータス。MySQL のコンパイル時に WITH_DEBUG が定義された場合、このフィールドにはいくつかの値が表示されます。WITH_DEBUG が定義されなかった場合、このステートメントは os_waits 値のみを表示します。後者の (WITH_DEBUG なしの) 場合は、この出力の基になっている情報が通常の相互排他

ロックと、読み書きロックを保護する (複数読み取りまたは単一書き込みを許可する) 相互排他ロックを区別するには不足しています。その結果、出力が、同じ相互排他ロックに対して複数の行を含むように見える可能性があります。

- count は、相互排他ロックが要求された回数を示します。
- spin_waits は、スピンロックの実行が必要になった回数を示します。
- spin_rounds は、スピンロックラウンドの数を示します。(spin_rounds を spin_waits で割ると、平均のラウンド数が得られます。)
- os_waits は、オペレーティングシステムの待機の数を示します。これは、スピンロックが機能しなかった (スピンロック中に相互排他ロックがロックされず、オペレーティングシステムに譲って待機する必要があった) 場合に発生します。
- os_yields は、相互排他ロックをロックしようとしているスレッドがそのタイムスライスをあきらめ、オペレーティングシステムに譲った回数を示します (ほかのスレッドの実行を許可すると、相互排他ロックが解放され、ロックできるようになることを前提にしています)。
- os_wait_times は、オペレーティングシステムの待機に費やされた時間 (ミリ秒単位) を示します。MySQL 5.6 ではタイミングが無効になるため、この値は常に 0 です。

大きなバッファープールでは出力の量がシステム上で膨大になる場合があるため、SHOW ENGINE INNODB MUTEX は、バッファープールブロックの相互排他ロックと読み書きロックをスキップします。(各 16K バッファープールブロック内に 1 つの相互排他ロックと 1 つの読み書きロックが存在し、1G バイトあたり 65,536 個のブロックが存在します。) SHOW ENGINE INNODB MUTEX はまた、待機されなかった (os_waits=0) 相互排他ロックまたは読み書きロックも一覧表示しません。そのため、SHOW ENGINE INNODB MUTEX は、OS レベルの待機を少なくとも 1 回は発生させた、バッファープールの外部の相互排他ロックと読み書きロックに関する情報のみを表示します。

SHOW ENGINE INNODB MUTEX の情報を使用すると、システムの問題を診断できます。たとえば、spin_waits や spin_rounds の値が大きい場合は、スケーラビリティーの問題を示している可能性があります。

SHOW ENGINE PERFORMANCE_SCHEMA STATUS を使用して、パフォーマンススキーマコードの内部操作を検査します。

```
mysql> SHOW ENGINE PERFORMANCE_SCHEMA STATUS\G
...
*************************** 3. row ***************************
  Type: performance_schema
  Name: events_waits_history.row_size
Status: 76
*************************** 4. row ***************************
  Type: performance_schema
  Name: events_waits_history.row_count
Status: 10000
*************************** 5. row ***************************
  Type: performance_schema
  Name: events_waits_history.memory
Status: 760000
...
*************************** 57. row ***************************
  Type: performance_schema
  Name: performance_schema.memory
Status: 26459600
...
```

このステートメントは、さまざまなパフォーマンススキーマオプションがメモリー要件に与える効果について、DBA が理解できるようにすることを目的としています。

Name 値は、それぞれ、内部バッファーとバッファー属性を指定する 2 つの部分で構成されます。バッファー名は、次のように解釈します。

- テーブルとして公開されていない内部バッファーは括弧内に指定されます。例: (pfs_cond_class).row_size、(pfs_mutex_class).memory。
- performance_schema データベース内のテーブルとして公開されている内部バッファーは、そのテーブル名で (括弧なしで) 指定されます。例: events_waits_history.row_size、mutex_instances.row_count。
- 全体としてのパフォーマンススキーマに適用される値は、performance_schema で始まります。例: performance_schema.memory。

バッファー属性には、次の意味があります。

- row_size は、実装によって使用される内部レコードのサイズ (テーブル内の行のサイズなど) です。row_size 値は変更できません。
- row_count は、内部レコードの数 (テーブル内の行数など) です。row_count 値は、パフォーマンススキーマの構成オプションを使用して変更できます。
- テーブルの場合、tbl_name.memory は row_size と row_count の積です。全体としてのパフォーマンススキーマの場合、performance_schema.memory は、使用されているすべてのメモリーの合計 (ほかのすべての memory 値の合計) です。

場合によっては、パフォーマンススキーマの構成パラメータと SHOW ENGINE 値の間に直接の関係が存在します。たとえば、events_waits_history_long.row_count は performance_schema_events_waits_history_long_size に対応します。その他の場合、この関係はより複雑です。たとえば、events_waits_history.row_count は、performance_schema_events_waits_history_size (スレッドあたりの行数) に performance_schema_max_thread_instances (スレッドの数) を掛けた値に対応します。

SHOW ENGINE NDB STATUS　サーバーで NDB ストレージエンジンが有効になっている場合、SHOW ENGINE NDB STATUS は、接続されているデータノードの数、クラスタの接続文字列、クラスタバイナリログのエポックや、クラスタに接続したときに MySQL Server によって作成されたさまざまなクラスタ API オブジェクトの数などのクラスタステータス情報を表示します。このステートメントからのサンプル出力を次に示します。

```
mysql> SHOW ENGINE NDB STATUS;
+------------+-----------------------+--------------------------------------------------+
| Type       | Name                  | Status                                           |
+------------+-----------------------+--------------------------------------------------+
| ndbcluster | connection            | cluster_node_id=7,
  connected_host=192.168.0.103, connected_port=1186, number_of_data_nodes=4,
  number_of_ready_data_nodes=3, connect_count=0                                        |
| ndbcluster | NdbTransaction        | created=6, free=0, sizeof=212                    |
| ndbcluster | NdbOperation          | created=8, free=8, sizeof=660                    |
| ndbcluster | NdbIndexScanOperation | created=1, free=1, sizeof=744                    |
| ndbcluster | NdbIndexOperation     | created=0, free=0, sizeof=664                    |
| ndbcluster | NdbRecAttr            | created=1285, free=1285, sizeof=60               |
| ndbcluster | NdbApiSignal          | created=16, free=16, sizeof=136                  |
| ndbcluster | NdbLabel              | created=0, free=0, sizeof=196                    |
| ndbcluster | NdbBranch             | created=0, free=0, sizeof=24                     |
| ndbcluster | NdbSubroutine         | created=0, free=0, sizeof=68                     |
| ndbcluster | NdbCall               | created=0, free=0, sizeof=16                     |
| ndbcluster | NdbBlob               | created=1, free=1, sizeof=264                    |
| ndbcluster | NdbReceiver           | created=4, free=0, sizeof=68                     |
| ndbcluster | binlog                | latest_epoch=155467, latest_trans_epoch=148126,
  latest_received_binlog_epoch=0, latest_handled_binlog_epoch=0,
  latest_applied_binlog_epoch=0                                                        |
+------------+-----------------------+--------------------------------------------------+
```

Name カラム内に connection と binlog を含む行は、MySQL 5.1 でこのステートメントの出力に追加されました。これらの各行内の Status カラムは、それぞれ、MySQL サーバーのクラスタへの接続に関する情報と、クラスタバイナリログのステータスに関する情報を提供します。Status 情報は、カンマで区切られた一連の名前と値のペアの形式をしています。

connection 行の Status カラムには、次の表で説明されている名前と値のペアが含まれています。

名前	値
cluster_node_id	クラスタ内の MySQL サーバーのノード ID
connected_host	MySQL サーバーが接続されているクラスタ管理サーバーのホスト名または IP アドレス
connected_port	MySQL サーバーが管理サーバー (connected_host) に接続するために使用するポート
number_of_data_nodes	クラスタのために構成されているデータノードの数 (つまり、そのクラスタの config.ini ファイル内の [ndbd] セクションの数)
number_of_ready_data_nodes	実際に実行されているクラスタ内のデータノードの数
connect_count	この mysqld がクラスタデータノードに接続または再接続した回数

binlog 行の Status カラムには、MySQL Cluster レプリケーションに関連した情報が含まれています。そこに含まれている名前と値のペアについて、次の表で説明します。

名前	値
latest_epoch	この MySQL サーバー上で直近で実行された最新のエポック (つまり、このサーバー上で実行された最新のトランザクションのシーケンス番号)
latest_trans_epoch	クラスタのデータノードによって処理された最新のエポック
latest_received_binlog_epoch	バイナリログスレッドによって受信された最新のエポック
latest_handled_binlog_epoch	(バイナリログへの書き込みのために) バイナリログスレッドによって処理された最新のエポック
latest_applied_binlog_epoch	実際にバイナリログに書き込まれた最新のエポック

詳細は、セクション18.6「MySQL Cluster レプリケーション」を参照してください。

クラスタのモニタリングにもっとも役立つ可能性のある SHOW ENGINE NDB STATUS の出力の残りの行を、次に Name で一覧表示します。

- NdbTransaction: 作成された NdbTransaction オブジェクトの数とサイズ。NdbTransaction は、NDB テーブル上で (CREATE TABLE や ALTER TABLE などの) テーブルスキーマ操作が実行されるたびに作成されます。
- NdbOperation: 作成された NdbOperation オブジェクトの数とサイズ。
- NdbIndexScanOperation: 作成された NdbIndexScanOperation オブジェクトの数とサイズ。
- NdbIndexOperation: 作成された NdbIndexOperation オブジェクトの数とサイズ。
- NdbRecAttr: 作成された NdbRecAttr オブジェクトの数とサイズ。一般に、これらのいずれかは、SQL ノードによってデータ操作ステートメントが実行されるたびに作成されます。
- NdbBlob: 作成された NdbBlob オブジェクトの数とサイズ。NdbBlob は、NDB テーブル内の BLOB カラムに関連する新しい操作が実行されるたびに作成されます。
- NdbReceiver: 作成されたすべての NdbReceiver オブジェクトの数とサイズ。created カラム内の数は、MySQL サーバーが接続されているクラスタ内のデータノードの数と同じです。

> **注記**
>
> 現在のセッション中に、このステートメントが実行されている SQL ノードにアクセスしている MySQL クライアントによって NDB テーブルに関連する操作が実行されていない場合、SHOW ENGINE NDB STATUS は空の結果を返します。

13.7.5.17 SHOW ENGINES 構文

```
SHOW [STORAGE] ENGINES
```

SHOW ENGINES は、サーバーのストレージエンジンに関するステータス情報を表示します。これは、ストレージエンジンがサポートされているかどうかをチェックしたり、デフォルトのエンジンが何であるかを確認したりするために特に役立ちます。この情報はまた、INFORMATION_SCHEMA ENGINES テーブルからも取得できます。セクション21.6「INFORMATION_SCHEMA ENGINES テーブル」を参照してください。

```
mysql> SHOW ENGINES\G
*************************** 1. row ***************************
      Engine: MEMORY
     Support: YES
     Comment: Hash based, stored in memory, useful for temporary tables
Transactions: NO
          XA: NO
  Savepoints: NO
*************************** 2. row ***************************
      Engine: MyISAM
     Support: YES
     Comment: MyISAM storage engine
Transactions: NO
          XA: NO
  Savepoints: NO
*************************** 3. row ***************************
      Engine: InnoDB
     Support: DEFAULT
     Comment: Supports transactions, row-level locking, and foreign keys
Transactions: YES
```

```
        XA: YES
 Savepoints: YES
*************************** 4. row ***************************
     Engine: EXAMPLE
    Support: YES
    Comment: Example storage engine
Transactions: NO
        XA: NO
 Savepoints: NO
*************************** 5. row ***************************
     Engine: ARCHIVE
    Support: YES
    Comment: Archive storage engine
Transactions: NO
        XA: NO
 Savepoints: NO
*************************** 6. row ***************************
     Engine: CSV
    Support: YES
    Comment: CSV storage engine
Transactions: NO
        XA: NO
 Savepoints: NO
*************************** 7. row ***************************
     Engine: BLACKHOLE
    Support: YES
    Comment: /dev/null storage engine (anything you write »
            to it disappears)
Transactions: NO
        XA: NO
 Savepoints: NO
*************************** 8. row ***************************
     Engine: FEDERATED
    Support: YES
    Comment: Federated MySQL storage engine
Transactions: NO
        XA: NO
 Savepoints: NO
*************************** 9. row ***************************
     Engine: MRG_MYISAM
    Support: YES
    Comment: Collection of identical MyISAM tables
Transactions: NO
        XA: NO
 Savepoints: NO
```

SHOW ENGINES からの出力は、使用されている MySQL バージョンやその他の要因によって異なる可能性があります。Support カラムに表示されている値は、次の表に示す、ストレージエンジンに対するサーバーのサポートのレベルを示します。

値	意味
YES	このエンジンはサポートされており、アクティブです
DEFAULT	YES と同様であることに加え、これがデフォルトのエンジンです
NO	このエンジンはサポートされていません
DISABLED	このエンジンはサポートされていますが、無効になっています

NO の値は、サーバーがこのエンジンに対するサポートなしでコンパイルされたことを示すため、実行時にこのエンジンを有効にすることはできません。

DISABLED の値は、サーバーがこのエンジンを無効にするオプションを使用して起動されたか、またはこのエンジンを有効にするために必要な一部のオプションが指定されなかったために発生します。後者の場合は、このオプションがなぜ無効になっているかを示す理由がエラーログファイルに含まれているはずです。セクション 5.2.2「エラーログ」 を参照してください。

ストレージエンジンに対する DISABLED はまた、サーバーがそれをサポートするようにコンパイルされたが、--skip-engine_name オプションで起動された場合にも表示される可能性があります。NDB ストレージエンジンの場合、DISABLED は、サーバーが MySQL Cluster に対するサポート付きでコンパイルされたが、--ndbcluster オプションで起動されなかったことを示します。

MyISAM はデフォルトのストレージエンジンであるため、すべての MySQL サーバーが MyISAM テーブルをサポートします。MyISAM を無効にすることはできません。

Transactions、XA、および Savepoints カラムは、それぞれ、このストレージエンジンがトランザクション、XA トランザクション、およびセーブポイントをサポートするかどうかを示します。

13.7.5.18 SHOW ERRORS 構文

```
SHOW ERRORS [LIMIT [offset,] row_count]
SHOW COUNT(*) ERRORS
```

SHOW ERRORS は SHOW WARNINGS に似た診断ステートメントですが、エラー、警告、および注意ではなく、エラーに関する情報のみを表示する点が異なります。

LIMIT 句の構文は、SELECT ステートメントの場合と同じです。セクション13.2.9「SELECT 構文」を参照してください。

SHOW COUNT(*) ERRORS ステートメントは、エラーの数を表示します。この数はまた、error_count 変数からも取得できます。

```
SHOW COUNT(*) ERRORS;
SELECT @@error_count;
```

SHOW ERRORS および error_count は、警告や注意ではなく、エラーにのみ適用されます。その他の点では、SHOW WARNINGS および warning_count と同様です。特に、SHOW ERRORS が max_error_count 個を超えるメッセージに関する情報を表示できないのに対して、error_count は、エラーの数が max_error_count を超えた場合は max_error_count の値を超えることができます。

詳細は、セクション13.7.5.41「SHOW WARNINGS 構文」を参照してください。

13.7.5.19 SHOW EVENTS 構文

```
SHOW EVENTS [{FROM | IN} schema_name]
   [LIKE 'pattern' | WHERE expr]
```

このステートメントは、イベントマネージャーのイベントに関する情報を表示します。これには、これらのイベントが示される元のデータベースに対する EVENT 権限が必要です。

SHOW EVENTS は、そのもっとも単純な形式では、現在のスキーマ内のすべてのイベントを一覧表示します。

```
mysql> SELECT CURRENT_USER(), SCHEMA();
+----------------+----------+
| CURRENT_USER() | SCHEMA() |
+----------------+----------+
| jon@ghidora    | myschema |
+----------------+----------+
1 row in set (0.00 sec)

mysql> SHOW EVENTS\G
*************************** 1. row ***************************
                  Db: myschema
                Name: e_daily
             Definer: jon@ghidora
           Time zone: SYSTEM
                Type: RECURRING
          Execute at: NULL
      Interval value: 10
      Interval field: SECOND
              Starts: 2006-02-09 10:41:23
                Ends: NULL
              Status: ENABLED
          Originator: 0
character_set_client: latin1
collation_connection: latin1_swedish_ci
  Database Collation: latin1_swedish_ci
```

特定のスキーマのイベントを表示するには、FROM 句を使用します。たとえば、test スキーマのイベントを表示するには、次のステートメントを使用します。

```
SHOW EVENTS FROM test;
```

LIKE 句 (存在する場合) は、どのイベント名と照合するかを示します。セクション21.32「SHOW ステートメントの拡張」で説明されているように、WHERE 句を指定すると、より一般的な条件を使用して行を選択できます。

SHOW EVENTS の出力には、次のカラムがあります。

- Db: このイベントが定義されているスキーマ (データベース)。
- Name: このイベントの名前。
- Time zone: このイベントのタイムゾーン。これは、イベントのスケジューリングに使用され、イベントが実行されるとそのイベント内で有効になるタイムゾーンです。デフォルト値は SYSTEM です。
- Definer: このイベントを作成したユーザーのアカウント。形式は 'user_name'@'host_name' です。
- Type: このイベントの繰り返しのタイプ。ONE TIME (一時的) または RECURRING (繰り返し) のどちらかです。
- Execute At: 一時的なイベントの実行が設定されている日付と時間。DATETIME 値として示されます。

  繰り返しのイベントの場合、このカラムの値は常に NULL です。
- Interval Value: 繰り返しのイベントの場合は、イベント実行の間で待機する間隔の数。

  一時的なイベントの場合、このカラムの値は常に NULL です。
- Interval Field: 繰り返しのイベントが繰り返しの前に待機する間隔に使用される時間単位。

  一時的なイベントの場合、このカラムの値は常に NULL です。
- Starts: 繰り返しのイベントの開始日付と開始時間。これは DATETIME 値として表示され、このイベントの開始日付と開始時間が定義されていない場合は NULL です。

  一時的なイベントの場合、このカラムは常に NULL です。
- Ends: 繰り返しのイベントの終了日付と終了時間。これは DATETIME 値として表示され、このイベントの終了日付と終了時間が定義されていない場合は、デフォルトで NULL になります。

  一時的なイベントの場合、このカラムは常に NULL です。
- Status: このイベントのステータス。ENABLED、DISABLED、SLAVESIDE_DISABLED のいずれか。

  SLAVESIDE_DISABLED は、イベントの作成が、レプリケーションマスターとして機能している別の MySQL サーバーで発生し、スレーブとして機能している現在の MySQL サーバーにレプリケートされたが、スレーブでは現在そのイベントが実行されていないことを示します。
- Originator: このイベントが作成された MySQL サーバーのサーバー ID。デフォルトで 0 になります。
- character_set_client は、このルーチンが作成されたときの character_set_client システム変数のセッション値です。collation_connection は、このルーチンが作成されたときの collation_connection システム変数のセッション値です。Database Collation は、このルーチンが関連付けられているデータベースの照合順序です。

SLAVE_DISABLED および Originator カラムの詳細は、セクション17.4.1.11「呼び出される機能のレプリケーション」を参照してください。

イベントのアクションステートメントは、SHOW EVENTS の出力には表示されません。SHOW CREATE EVENT または INFORMATION_SCHEMA.EVENTS テーブルを使用してください。

SHOW EVENTS によって表示される時間は、セクション20.4.4「イベントメタデータ」で説明されているように、このイベントのタイムゾーンで示されます。

SHOW EVENTS の出力内のカラムは、INFORMATION_SCHEMA.EVENTS テーブル内のカラムに似ていますが、同じではありません。セクション21.7「INFORMATION_SCHEMA EVENTS テーブル」を参照してください。

13.7.5.20 SHOW FUNCTION CODE 構文

```
SHOW FUNCTION CODE func_name
```

このステートメントは、ストアドファンクションである点を除き、SHOW PROCEDURE CODE と同じです。セクション13.7.5.28「SHOW PROCEDURE CODE 構文」を参照してください。

13.7.5.21 SHOW FUNCTION STATUS 構文

```
SHOW FUNCTION STATUS
```

```
  [LIKE 'pattern' | WHERE expr]
```

このステートメントは、ストアドファンクションである点を除き、SHOW PROCEDURE STATUS と同じです。セクション13.7.5.29「SHOW PROCEDURE STATUS 構文」を参照してください。

13.7.5.22 SHOW GRANTS 構文

```
SHOW GRANTS [FOR user]
```

このステートメントは、MySQL ユーザーアカウントに付与される権限を複製するために発行する必要のある 1 つまたは複数の GRANT ステートメントを一覧表示します。このアカウントは、GRANT ステートメントの場合と同じ形式 ('jeffrey'@'localhost' など) を使用して指定されます。アカウント名のユーザー名の部分のみを指定した場合は、'%' のホスト名の部分が使用されます。アカウント名の指定の詳細は、セクション13.7.1.4「GRANT 構文」を参照してください。

```
mysql> SHOW GRANTS FOR 'root'@'localhost';
+---------------------------------------------------------------------+
| Grants for root@localhost                                           |
+---------------------------------------------------------------------+
| GRANT ALL PRIVILEGES ON *.* TO 'root'@'localhost' WITH GRANT OPTION |
+---------------------------------------------------------------------+
```

サーバーに接続するために使用しているアカウントに付与される権限を一覧表示するには、次のいずれのステートメントでも使用できます。

```
SHOW GRANTS;
SHOW GRANTS FOR CURRENT_USER;
SHOW GRANTS FOR CURRENT_USER();
```

SHOW GRANTS FOR CURRENT_USER (または、いずれかの同等の構文) が DEFINER コンテキスト (SQL SECURITY DEFINER で定義されているストアドプロシージャー内など) で使用されている場合、表示される付与権限は呼び出し元ではなく、定義者のものです。

SHOW GRANTS は、指定されたアカウントに明示的に付与される権限のみを表示します。そのアカウントでほかの権限を使用できる可能性もありますが、それらは表示されません。たとえば、匿名アカウントが存在する場合、指定されたアカウントはその権限を使用できる可能性がありますが、SHOW GRANTS はそれらを表示しません。

SHOW GRANTS には、mysql データベースに対する SELECT 権限 (現在のユーザーの権限の表示を除く) が必要です。

13.7.5.23 SHOW INDEX 構文

```
SHOW {INDEX | INDEXES | KEYS}
  {FROM | IN} tbl_name
  [{FROM | IN} db_name]
  [WHERE expr]
```

SHOW INDEX は、テーブルインデックス情報を返します。この形式は、ODBC での SQLStatistics 呼び出しの形式に似ています。このステートメントには、このテーブル内のいずれかのカラムに対する何らかの権限が必要です。

SHOW INDEX は、次のフィールドを返します。

- Table

  テーブルの名前。

- Non_unique

  このインデックスが重複を含むことができない場合は 0、できる場合は 1。

- Key_name

  インデックスの名前。このインデックスが主キーである場合、その名前は常に PRIMARY です。

- Seq_in_index

  インデックス内のカラムシーケンス番号であり、1 から始まります。

- Column_name

カラム名。

- Collation

  インデックス内でのカラムのソート方法。MySQL では、これは「A」(昇順) またはNULL(ソートされない) のどちらかの値です。

- Cardinality

  このインデックス内の一意の値の数の推定値。これは、ANALYZE TABLE または myisamchk -a を実行することによって更新されます。Cardinality は整数として格納された統計に基づいてカウントされるため、この値は、小さなテーブルの場合でも必ずしも正確であるとはかぎりません。カーディナリティーが高いほど、MySQL が結合を実行するときにこのインデックスを使用する可能性は高くなります。

- Sub_part

  カラムが部分的にしかインデックス設定されていない場合は、インデックス設定された文字の数。カラム全体がインデックス設定されている場合は NULL。

- Packed

  キーがパックされる方法を示します。パックされない場合は NULL。

- Null

  このカラムに NULL 値を含めることができる場合は YES が、できない場合は '' が含まれます。

- Index_type

  使用されるインデックス方法 (BTREE、FULLTEXT、HASH、RTREE)。

- Comment

  各カラムで説明されていないこのインデックスに関する情報 (このインデックスが無効になっている場合の disabled など)。

- Index_comment

  このインデックスが作成されたときに COMMENT 属性でインデックスに対して提供された任意のコメント。

tbl_name FROM db_name 構文の代わりに db_name.tbl_name を使用できます。次の 2 つのステートメントは同等です。

```
SHOW INDEX FROM mytable FROM mydb;
SHOW INDEX FROM mydb.mytable;
```

セクション21.32「SHOW ステートメントの拡張」で説明されているように、WHERE 句を指定すると、より一般的な条件を使用して行を選択できます。

また、mysqlshow -k db_name tbl_name コマンドを使用してテーブルのインデックスを一覧表示することもできます。

13.7.5.24 SHOW MASTER STATUS 構文

```
SHOW MASTER STATUS
```

このステートメントは、マスターのバイナリログファイルに関するステータス情報を提供します。これには、SUPER または REPLICATION CLIENT 権限のどちらかが必要です。

例:

```
mysql> SHOW MASTER STATUS\G
*************************** 1. row ***************************
             File: master-bin.000002
         Position: 1307
     Binlog_Do_DB: test
 Binlog_Ignore_DB: manual, mysql
Executed_Gtid_Set: 3E11FA47-71CA-11E1-9E33-C80AA9429562:1-5
1 row in set (0.00 sec)
```

Executed_Gtid_Set カラムは、MySQL 5.6.5 で追加されました。グローバルトランザクション ID が使用されている場合、このカラムは、マスター上で実行されたトランザクションの GTID のセットを表示します。これは、このサーバー上の gtid_executed システム変数 (MySQL 5.6.9 より前の名前は gtid_done) の値や、このサーバー上での SHOW SLAVE STATUS の出力内の Executed_Gtid_Set の値と同じです。

13.7.5.25 SHOW OPEN TABLES 構文

```
SHOW OPEN TABLES [{FROM | IN} db_name]
    [LIKE 'pattern' | WHERE expr]
```

SHOW OPEN TABLES は、現在テーブルキャッシュ内で開いている TEMPORARY 以外のテーブルを一覧表示します。セクション8.4.3.1「MySQL でのテーブルのオープンとクローズの方法」を参照してください。FROM 句 (存在する場合) は、表示されるテーブルを db_name データベース内に存在するテーブルに制限します。LIKE 句 (存在する場合) は、どのテーブル名と照合するかを示します。セクション21.32「SHOW ステートメントの拡張」で説明されているように、WHERE 句を指定すると、より一般的な条件を使用して行を選択できます。

SHOW OPEN TABLES の出力には、次のカラムがあります。

- Database

  このテーブルを含むデータベース。

- Table

  テーブル名。

- In_use

  このテーブルのために存在するテーブルロックまたはロック要求の数。たとえば、あるクライアントが LOCK TABLE t1 WRITE を使用してテーブルに対するロックを取得した場合、In_use は 1 になります。このテーブルがロックされたままになっている間に別のクライアントが LOCK TABLE t1 WRITE を発行した場合、このクライアントはロックを待機してブロックされますが、この ロック要求が In_use を 2 にします。このカウントが 0 の場合、このテーブルは開いていますが、現在使用されていません。In_use はまた、HANDLER ... OPEN ステートメントによって増加し、HANDLER ... CLOSE ステートメントによって減少します。

- Name_locked

  テーブル名がロックされているかどうか。名前のロックは、テーブルの削除や名前の変更などの操作に使用されます。

テーブルに対する権限を持っていない場合、そのテーブルは SHOW OPEN TABLES の出力に表示されません。

13.7.5.26 SHOW PLUGINS 構文

```
SHOW PLUGINS
```

SHOW PLUGINS は、サーバープラグインについての情報を表示します。プラグイン情報はまた、INFORMATION_SCHEMA.PLUGINS テーブルでも入手できます。セクション21.14「INFORMATION_SCHEMA PLUGINS テーブル」を参照してください。

SHOW PLUGINS の出力の例:

```
mysql> SHOW PLUGINS\G
*************************** 1. row ***************************
   Name: binlog
 Status: ACTIVE
   Type: STORAGE ENGINE
Library: NULL
License: GPL
*************************** 2. row ***************************
   Name: CSV
 Status: ACTIVE
   Type: STORAGE ENGINE
Library: NULL
License: GPL
*************************** 3. row ***************************
   Name: MEMORY
 Status: ACTIVE
   Type: STORAGE ENGINE
Library: NULL
License: GPL
```

```
*************************** 4. row ***************************
  Name: MyISAM
Status: ACTIVE
  Type: STORAGE ENGINE
Library: NULL
License: GPL
...
```

SHOW PLUGINS の出力には、次のカラムがあります。

- Name: INSTALL PLUGIN や UNINSTALL PLUGIN などのステートメントでこのプラグインを参照するために使用される名前。
- Status: このプラグインのステータスであり、ACTIVE、INACTIVE、DISABLED、DELETED のいずれかです。
- Type: このプラグインのタイプであり、STORAGE ENGINE、INFORMATION_SCHEMA、AUTHENTICATION などがあります。
- Library: このプラグインの共有オブジェクトファイルの名前。これは、INSTALL PLUGIN や UNINSTALL PLUGIN などのステートメントでプラグインファイルを参照するために使用される名前です。このファイルは、plugin_dir システム変数によって指名されたディレクトリに置かれます。ライブラリ名が NULL である場合、プラグインはコンパイルされますが、UNINSTALL PLUGIN でアンインストールできません。
- License: このプラグインのライセンス方法 (GPL など)。

INSTALL PLUGIN でインストールされたプラグインの場合、Name および Library 値は mysql.plugin テーブルにも登録されます。

SHOW PLUGINS によって表示される情報の基礎を形成するプラグインのデータ構造については、セクション 24.2「MySQL プラグイン API」を参照してください。

13.7.5.27 SHOW PRIVILEGES 構文

```
SHOW PRIVILEGES
```

SHOW PRIVILEGES は、MySQL サーバーがサポートするシステム権限のリストを表示します。権限の正確なリストは、使用しているサーバーのバージョンによって異なります。

```
mysql> SHOW PRIVILEGES\G
*************************** 1. row ***************************
Privilege: Alter
  Context: Tables
  Comment: To alter the table
*************************** 2. row ***************************
Privilege: Alter routine
  Context: Functions,Procedures
  Comment: To alter or drop stored functions/procedures
*************************** 3. row ***************************
Privilege: Create
  Context: Databases,Tables,Indexes
  Comment: To create new databases and tables
*************************** 4. row ***************************
Privilege: Create routine
  Context: Databases
  Comment: To use CREATE FUNCTION/PROCEDURE
*************************** 5. row ***************************
Privilege: Create temporary tables
  Context: Databases
  Comment: To use CREATE TEMPORARY TABLE
...
```

特定のユーザーに属する権限は、SHOW GRANTS ステートメントによって表示されます。詳細は、セクション 13.7.5.22「SHOW GRANTS 構文」を参照してください。

13.7.5.28 SHOW PROCEDURE CODE 構文

```
SHOW PROCEDURE CODE proc_name
```

このステートメントは、デバッグサポート付きで構築されたサーバーでのみ使用可能な MySQL 拡張です。これは、指定されたストアドプロシージャーの内部実装の表現を表示します。同様のステートメントである SHOW FUNCTION CODE は、ストアドファンクションに関する情報を表示します (セクション13.7.5.20「SHOW FUNCTION CODE 構文」を参照してください)。

どちらのステートメントを使用するにも、このルーチンの所有者であるか、または mysql.proc テーブルへの SELECT アクセス権を持っている必要があります。

指定されたルーチンが使用可能な場合、各ステートメントは結果セットを生成します。結果セット内の各行は、このルーチン内の 1 つの「命令」に対応します。最初のカラムは、0 で始まる順序番号である Pos です。2 番目のカラムは Instruction であり、SQL ステートメント (通常は、元のソースから変更されています)、またはストアドルーチンのハンドラに対してのみ意味を持つディレクティブが含まれています。

```
mysql> DELIMITER //
mysql> CREATE PROCEDURE p1 ()
    -> BEGIN
    ->   DECLARE fanta INT DEFAULT 55;
    ->   DROP TABLE t2;
    ->   LOOP
    ->     INSERT INTO t3 VALUES (fanta);
    ->     END LOOP;
    ->   END//
Query OK, 0 rows affected (0.00 sec)

mysql> SHOW PROCEDURE CODE p1//
+-----+----------------------------------------+
| Pos | Instruction                            |
+-----+----------------------------------------+
|   0 | set fanta@0 55                         |
|   1 | stmt 9 "DROP TABLE t2"                 |
|   2 | stmt 5 "INSERT INTO t3 VALUES (fanta)" |
|   3 | jump 2                                 |
+-----+----------------------------------------+
4 rows in set (0.00 sec)
```

この例では、実行不可能な BEGIN および END ステートメントが消えており、DECLARE variable_name ステートメントでは、実行可能ファイルの部分 (デフォルトが割り当てられている部分) のみが表示されています。ソースから取得されたステートメントごとに、コードワード stmt とそれに続くタイプ (DROP を示す 9、INSERT を示す 5 など) が存在します。最終行には、GOTO instruction #2 を示す命令 jump 2 が含まれています。

13.7.5.29 SHOW PROCEDURE STATUS 構文

```
SHOW PROCEDURE STATUS
  [LIKE 'pattern' | WHERE expr]
```

このステートメントは、MySQL 拡張です。これは、ストアドプロシージャーの特性 (データベース、名前、型、作成者、作成日と変更日、文字セット情報など) を返します。同様のステートメントである SHOW FUNCTION STATUS は、ストアドファンクションに関する情報を表示します (セクション13.7.5.21「SHOW FUNCTION STATUS 構文」を参照してください)。

LIKE 句 (存在する場合) は、どのプロシージャーまたは関数名と照合するかを示します。セクション 21.32「SHOW ステートメントの拡張」で説明されているように、WHERE 句を指定すると、より一般的な条件を使用して行を選択できます。

```
mysql> SHOW PROCEDURE STATUS LIKE 'sp1'\G
*************************** 1. row ***************************
                  Db: test
                Name: sp1
                Type: PROCEDURE
             Definer: testuser@localhost
            Modified: 2004-08-03 15:29:37
             Created: 2004-08-03 15:29:37
       Security_type: DEFINER
             Comment:
character_set_client: latin1
collation_connection: latin1_swedish_ci
  Database Collation: latin1_swedish_ci
```

character_set_client は、このルーチンが作成されたときの character_set_client システム変数のセッション値です。collation_connection は、このルーチンが作成されたときの collation_connection システム変数のセッション値です。Database Collation は、このルーチンが関連付けられているデータベースの照合順序です。

ストアドルーチンに関する情報はまた、INFORMATION_SCHEMA 内の ROUTINES テーブルからも取得できます。セクション21.18「INFORMATION_SCHEMA ROUTINES テーブル」を参照してください。

13.7.5.30 SHOW PROCESSLIST 構文

```
SHOW [FULL] PROCESSLIST
```

SHOW PROCESSLIST は、どのスレッドが実行されているかを表示します。この情報はまた、INFORMATION_SCHEMA PROCESSLIST テーブルまたは mysqladmin processlist コマンドからも取得できます。PROCESS 権限がある場合は、すべてのスレッドを表示できます。そうでない場合は、ユーザー独自のスレッド (つまり、使用している MySQL アカウントに関連付けられたスレッド) のみを表示できます。FULL キーワードを使用しない場合は、各ステートメントの最初の 100 文字のみが Info フィールドに表示されます。

処理情報は、performance_schema.threads テーブルから利用することもできます。ただし、threads へのアクセスには相互排他ロックは必要なく、サーバーパフォーマンスへの影響は最小です。INFORMATION_SCHEMA.PROCESSLIST および SHOW PROCESSLIST は、相互排他ロックを必要とするので、負のパフォーマンスの結果になります。threads はまた、バックグラウンドスレッドに関する情報も表示しますが、INFORMATION_SCHEMA.PROCESSLIST および SHOW PROCESSLIST は表示しません。これは、threads は、ほかのスレッド情報源では行えないアクティビティーのモニターに使用できることを意味します。

SHOW PROCESSLIST ステートメントは、「接続が多すぎます」というエラーメッセージが表示されるために、何が発生しているかを突き止めたい場合に非常に役立ちます。MySQL は、管理者がいつでもシステムに接続し、システムを確実にチェックできるようにするために、SUPER 権限を持つアカウントによって使用される追加の接続を 1 つ予約しています (この権限をすべてのユーザーには与えていないと仮定します)。

スレッドは、KILL ステートメントを使用して強制終了できます。セクション13.7.6.4「KILL 構文」を参照してください。

SHOW PROCESSLIST の出力の例を次に示します。

```
mysql> SHOW FULL PROCESSLIST\G
*************************** 1. row ***************************
Id: 1
User: system user
Host:
db: NULL
Command: Connect
Time: 1030455
State: Waiting for master to send event
Info: NULL
*************************** 2. row ***************************
Id: 2
User: system user
Host:
db: NULL
Command: Connect
Time: 1004
State: Has read all relay log; waiting for the slave
       I/O thread to update it
Info: NULL
*************************** 3. row ***************************
Id: 3112
User: replikator
Host: artemis:2204
db: NULL
Command: Binlog Dump
Time: 2144
State: Has sent all binlog to slave; waiting for binlog to be updated
Info: NULL
*************************** 4. row ***************************
Id: 3113
User: replikator
Host: iconnect2:45781
db: NULL
Command: Binlog Dump
Time: 2086
State: Has sent all binlog to slave; waiting for binlog to be updated
Info: NULL
*************************** 5. row ***************************
Id: 3123
User: stefan
Host: localhost
db: apollon
Command: Query
Time: 0
State: NULL
Info: SHOW FULL PROCESSLIST
5 rows in set (0.00 sec)
```

SHOW PROCESSLIST によって生成されるカラムには、次の意味があります。

- Id

接続識別子。これは、INFORMATION_SCHEMA.PROCESSLIST テーブルの ID カラム、パフォーマンススキーマの threads テーブルの PROCESSLIST_ID カラムに表示される値、および CONNECTION_ID() 関数によって返される値と同じ型の値です。

- User

  このステートメントを発行した MySQL ユーザー。これが system user である場合は、タスクを内部的に処理するためにサーバーによって生成された非クライアントスレッドを示します。これは、レプリケーションスレーブ上で使用されている I/O または SQL スレッドか、あるいは遅延行ハンドラである可能性があります。unauthenticated user は、クライアント接続に関連付けられたが、クライアントユーザーの認証がまだ実行されていないスレッドを示します。event_scheduler は、スケジュールされたイベントをモニターするスレッドを示します。system user の場合、Host カラムで指定されるホストは存在しません。

- Host

  このステートメントを発行しているクライアントのホスト名 (ホストが存在しない system user を除きます)。SHOW PROCESSLIST は、どのクライアントが何を実行しているかの判定を容易にするために、TCP/IP 接続のホスト名を host_name:client_port の形式でレポートします。

- db

  デフォルトデータベース (選択されている場合)。そうでない場合は NULL。

- Command

  スレッドが実行しているコマンドの種類。スレッドのコマンドの説明については、セクション8.12.5「スレッド情報の検査」を参照してください。このカラムの値は、クライアント/サーバープロトコルの COM_xxx コマンドと Com_xxx ステータス変数に対応します。セクション5.1.6「サーバーステータス変数」を参照してください

- Time

  スレッドが現在の状態になってからの秒数。スレーブ SQL スレッドの場合、この値は、最後にレプリケートされたイベントのタイムスタンプと、スレーブマシンの実際の時間の間の秒数です。セクション17.2.1「レプリケーション実装の詳細」を参照してください。

- State

  スレッドが行なっていることを示すアクション、イベント、または状態。State 値の説明については、セクション8.12.5「スレッド情報の検査」を参照してください。

  ほとんどの状態がきわめてすばやい操作に対応します。スレッドの状態が何秒間も特定の状態にとどまっている場合は、調査が必要な問題が発生している可能性があります。

  SHOW PROCESSLIST ステートメントの場合、State の値は NULL です。

- Info

  スレッドが実行しているステートメント、またはそれがどのステートメントも実行していない場合は NULL。このステートメントは、サーバーに送信されるステートメント、またはこのステートメントがほかのステートメントを実行する場合は、もっとも内側のステートメントである可能性があります。たとえば、CALL ステートメントが、SELECT ステートメントを実行しているストアドプロシージャーを実行する場合、Info 値はその SELECT ステートメントを示します。

13.7.5.31 SHOW PROFILE 構文

```
SHOW PROFILE [type [, type] ... ]
    [FOR QUERY n]
    [LIMIT row_count [OFFSET offset]]

type:
    ALL
  | BLOCK IO
  | CONTEXT SWITCHES
  | CPU
  | IPC
  | MEMORY
  | PAGE FAULTS
  | SOURCE
```

```
| SWAPS
```

SHOW PROFILE および SHOW PROFILES ステートメントは、現在のセッションの過程で実行されたステートメントのリソース使用状況を示すプロファイリング情報を表示します。

> **注記**
>
> これらのステートメントは、MySQL 5.6.7 の時点では非推奨であり、将来の MySQL リリースで削除される予定です。代わりに、パフォーマンススキーマを使用してください。第22章「MySQL パフォーマンススキーマ」を参照してください。

プロファイリングは、profiling セッション変数によって制御されます。このデフォルト値は 0 (OFF) です。プロファイリングは、profiling を 1 または ON に設定することによって有効になります。

```
mysql> SET profiling = 1;
```

SHOW PROFILES は、サーバーに送信された最新のステートメントのリストを表示します。このリストのサイズは、profiling_history_size セッション変数によって制御されます。このデフォルト値は 15 です。最大値は 100 です。この値を 0 に設定すると、実質的にプロファイリングが無効になります。

SHOW PROFILE と SHOW PROFILES を除くすべてのステートメントがプロファイルされるため、このどちらのステートメントもプロファイルリスト内に見つかりません。不正な形式のステートメントはプロファイルされます。たとえば、SHOW PROFILING は不正なステートメントであるため、それを実行しようとすると構文エラーが発生しますが、プロファイリングリストには表示されます。

SHOW PROFILE は、1 つのステートメントに関する詳細情報を表示します。FOR QUERY n 句を指定しない場合、出力は、直近で実行されたステートメントに関連したものになります。FOR QUERY n が含まれている場合、SHOW PROFILE は、ステートメント n に関する情報を表示します。n の値は、SHOW PROFILES によって表示される Query_ID 値に対応します。

LIMIT row_count 句を指定すると、出力を row_count 行に制限できます。LIMIT が指定されている場合は、OFFSET offset を追加することで、行セット全体が offset 行分オフセットされた状態で出力を開始できます。

デフォルトでは、SHOW PROFILE は Status および Duration カラムを表示します。この Status 値は SHOW PROCESSLIST によって表示される State 値に似ていますが、一部のステータス値では、この 2 つのステートメントの解釈にわずかな違いがいくつか存在する可能性があります (セクション8.12.5「スレッド情報の検査」を参照してください)。

オプションの type 値を指定すると、次のその他の特定のタイプの情報を表示できます。

- ALL は、すべての情報を表示します
- BLOCK IO は、ブロック入力および出力操作の数を表示します
- CONTEXT SWITCHES は、自発的および非自発的コンテキストスイッチの数を表示します
- CPU は、ユーザーとシステムの CPU 使用時間を表示します
- IPC は、送受信されたメッセージの数を表示します
- MEMORY は現在、実装されていません
- PAGE FAULTS は、メジャーおよびマイナーページフォルトの数を表示します
- SOURCE は、ソースコードの関数の名前を、その関数が含まれているファイルの名前および行番号とともに表示します
- SWAPS は、スワップ数を表示します

プロファイリングは、セッション単位で有効になります。セッションが終了すると、そのプロファイリング情報は失われます。

```
mysql> SELECT @@profiling;
+-------------+
| @@profiling |
+-------------+
|           0 |
+-------------+
1 row in set (0.00 sec)

mysql> SET profiling = 1;
```

```
Query OK, 0 rows affected (0.00 sec)

mysql> DROP TABLE IF EXISTS t1;
Query OK, 0 rows affected, 1 warning (0.00 sec)

mysql> CREATE TABLE T1 (id INT);
Query OK, 0 rows affected (0.01 sec)

mysql> SHOW PROFILES;
+----------+----------+--------------------------+
| Query_ID | Duration | Query                    |
+----------+----------+--------------------------+
|        0 | 0.000088 | SET PROFILING = 1        |
|        1 | 0.000136 | DROP TABLE IF EXISTS t1  |
|        2 | 0.011947 | CREATE TABLE t1 (id INT) |
+----------+----------+--------------------------+
3 rows in set (0.00 sec)

mysql> SHOW PROFILE;
+----------------------+----------+
| Status               | Duration |
+----------------------+----------+
| checking permissions | 0.000040 |
| creating table       | 0.000056 |
| After create         | 0.011363 |
| query end            | 0.000375 |
| freeing items        | 0.000089 |
| logging slow query   | 0.000019 |
| cleaning up          | 0.000005 |
+----------------------+----------+
7 rows in set (0.00 sec)

mysql> SHOW PROFILE FOR QUERY 1;
+--------------------+----------+
| Status             | Duration |
+--------------------+----------+
| query end          | 0.000107 |
| freeing items      | 0.000008 |
| logging slow query | 0.000015 |
| cleaning up        | 0.000006 |
+--------------------+----------+
4 rows in set (0.00 sec)

mysql> SHOW PROFILE CPU FOR QUERY 2;
+----------------------+----------+----------+------------+
| Status               | Duration | CPU_user | CPU_system |
+----------------------+----------+----------+------------+
| checking permissions | 0.000040 | 0.000038 |   0.000002 |
| creating table       | 0.000056 | 0.000028 |   0.000028 |
| After create         | 0.011363 | 0.000217 |   0.001571 |
| query end            | 0.000375 | 0.000013 |   0.000028 |
| freeing items        | 0.000089 | 0.000010 |   0.000014 |
| logging slow query   | 0.000019 | 0.000009 |   0.000010 |
| cleaning up          | 0.000005 | 0.000003 |   0.000002 |
+----------------------+----------+----------+------------+
7 rows in set (0.00 sec)
```

> **注記**
>
> 一部のアーキテクチャーでは、プロファイリングが部分的にしか機能しません。getrusage() システムコールに依存する値の場合、このシステムコールをサポートしていない Windows などのシステムでは NULL が返されます。さらに、プロファイリングはスレッド単位ではなく、プロセス単位です。つまり、サーバー内の、ユーザー独自のスレッド以外のスレッド上のアクティビティーが、ユーザーに表示されるタイミング情報に影響を与える可能性があります。

プロファイリング情報はまた、INFORMATION_SCHEMA 内の PROFILING テーブルからも取得できます。セクション21.16「INFORMATION_SCHEMA PROFILING テーブル」を参照してください。たとえば、次のクエリーは同じ結果を生成します。

```
SHOW PROFILE FOR QUERY 2;

SELECT STATE, FORMAT(DURATION, 6) AS DURATION
FROM INFORMATION_SCHEMA.PROFILING
WHERE QUERY_ID = 2 ORDER BY SEQ;
```

13.7.5.32 SHOW PROFILES 構文

```
SHOW PROFILES
```

SHOW PROFILES ステートメントは、SHOW PROFILE とともに、現在のセッションの過程で実行されたステートメントのリソース使用状況を示すプロファイリング情報を表示します。詳細は、セクション13.7.5.31「SHOW PROFILE 構文」を参照してください。

> **注記**
>
> これらのステートメントは、MySQL 5.6.7 の時点では非推奨であり、将来の MySQL リリースで削除される予定です。代わりに、パフォーマンススキーマを使用してください。第22章「MySQL パフォーマンススキーマ」を参照してください。

13.7.5.33 SHOW RELAYLOG EVENTS 構文

```
SHOW RELAYLOG EVENTS
   [IN 'log_name'] [FROM pos] [LIMIT [offset,] row_count]
```

レプリケーションスレーブのリレーログ内のイベントを表示します。'log_name' を指定しない場合は、最初のリレーログが表示されます。このステートメントは、マスターには影響を与えません。

LIMIT 句の構文は、SELECT ステートメントの場合と同じです。セクション13.2.9「SELECT 構文」を参照してください。

> **注記**
>
> SHOW RELAYLOG EVENTS を LIMIT 句なしで発行すると、サーバーはクライアントに (スレーブによって受信されたデータを変更するすべてのステートメントを含む) リレーログの完全な内容を返すため、時間とリソースを大量に消費するプロセスが開始される可能性があります。

> **注記**
>
> SHOW RELAYLOG EVENTS からの出力には、ユーザーおよびシステム変数の設定に関連した一部のイベントが含まれていません。リレーログ内のイベントを完全に取得するには、mysqlbinlog を使用します。

13.7.5.34 SHOW SLAVE HOSTS 構文

```
SHOW SLAVE HOSTS
```

現在マスターに登録されているレプリケーションスレーブのリストを表示します。

SHOW SLAVE HOSTS は、レプリケーションマスターとして機能するサーバー上で実行するようにしてください。このステートメントは、レプリケーションスレーブとして接続されているか、またはこれまでに接続されたサーバーに関する情報を、次に示すように結果の各行が 1 つのスレーブサーバーに対応するように表示します。

```
mysql> SHOW SLAVE HOSTS;
+-----------+-----------+-------+-----------+--------------------------------------+
| Server_id | Host      | Port  | Master_id | Slave_UUID                           |
+-----------+-----------+-------+-----------+--------------------------------------+
|  192168010 | iconnect2 | 3306 | 192168011 | 14cb6624-7f93-11e0-b2c0-c80aa9429562 |
| 1921680101 | athena    | 3306 | 192168011 | 07af4990-f41f-11df-a566-7ac56fdaf645 |
+------------+-----------+------+-----------+--------------------------------------+
```

- Server_id: このスレーブサーバーの一意のサーバー ID。これは、このスレーブサーバーのオプションファイルで、または --server-id=value を使用してコマンド行で構成されます。
- Host: このスレーブサーバーのホスト名。これは、このスレーブサーバーのオプションファイルで、または --report-host=host_name を使用してコマンド行で構成されます。これは、オペレーティングシステムで構成されているマシン名とは異なる場合があります。
- Port: このスレーブサーバーが待機しているマスター上のポート。

  MySQL 5.6.5 以降では、このカラムが 0 である場合、スレーブポート (--report-port) が設定されなかったことを示します。MySQL 5.6.5 より前は、このような場合は 3306 がデフォルトとして使用されました (Bug #13333431)。
- Master_id: このスレーブサーバーのレプリケーション元であるマスターサーバーの一意のサーバー ID。これは、SHOW SLAVE HOSTS が実行されているサーバーのサーバー ID であるため、この同じ値が結果の各行にリストされます。

- Slave_UUID: このスレーブのグローバルに一意の ID。これはスレーブ上で生成され、スレーブの auto.cnf ファイル内に格納されています。

  このカラムは、MySQL 5.6.0 で追加されました。

13.7.5.35 SHOW SLAVE STATUS 構文

```
SHOW SLAVE STATUS
```

このステートメントは、スレーブスレッドの基本的なパラメータに関するステータス情報を提供します。これには、SUPER または REPLICATION CLIENT 権限のどちらかが必要です。

mysql クライアントを使用してこのステートメントを発行する場合は、セミコロンの代わりに \G ステートメントターミネータを使用すると、より読みやすい縦のレイアウトが得られます。

```
mysql> SHOW SLAVE STATUS\G
*************************** 1. row ***************************
               Slave_IO_State: Waiting for master to send event
                  Master_Host: localhost
                  Master_User: root
                  Master_Port: 13000
                Connect_Retry: 60
              Master_Log_File: master-bin.000002
          Read_Master_Log_Pos: 1307
               Relay_Log_File: slave-relay-bin.000003
                Relay_Log_Pos: 1508
        Relay_Master_Log_File: master-bin.000002
             Slave_IO_Running: Yes
            Slave_SQL_Running: Yes
              Replicate_Do_DB:
          Replicate_Ignore_DB:
           Replicate_Do_Table:
       Replicate_Ignore_Table:
      Replicate_Wild_Do_Table:
  Replicate_Wild_Ignore_Table:
                   Last_Errno: 0
                   Last_Error:
                 Skip_Counter: 0
          Exec_Master_Log_Pos: 1307
              Relay_Log_Space: 1858
              Until_Condition: None
               Until_Log_File:
                Until_Log_Pos: 0
           Master_SSL_Allowed: No
           Master_SSL_CA_File:
           Master_SSL_CA_Path:
              Master_SSL_Cert:
            Master_SSL_Cipher:
               Master_SSL_Key:
        Seconds_Behind_Master: 0
Master_SSL_Verify_Server_Cert: No
                Last_IO_Errno: 0
                Last_IO_Error:
               Last_SQL_Errno: 0
               Last_SQL_Error:
  Replicate_Ignore_Server_Ids:
             Master_Server_Id: 1
                  Master_UUID: 3e11fa47-71ca-11e1-9e33-c80aa9429562
             Master_Info_File: /var/mysqld.2/data/master.info
                    SQL_Delay: 0
          SQL_Remaining_Delay: NULL
      Slave_SQL_Running_State: Slave has read all relay log; waiting for the slave I/O thread to update it
           Master_Retry_Count: 10
                  Master_Bind:
      Last_IO_Error_Timestamp:
     Last_SQL_Error_Timestamp:
               Master_SSL_Crl:
           Master_SSL_Crlpath:
           Retrieved_Gid_Set: 3e11fa47-71ca-11e1-9e33-c80aa9429562:1-5
            Executed_Gid_Set: 3e11fa47-71ca-11e1-9e33-c80aa9429562:1-5
                Auto_Position: 1
1 row in set (0.00 sec)
```

次のリストは、SHOW SLAVE STATUS によって返されるフィールドについて説明しています。これらの意味の解釈の詳細は、セクション8.12.5.6「レプリケーションスレーブの I/O スレッド状態」を参照してください。

- Slave_IO_State

スレーブ I/O スレッドに対する SHOW PROCESSLIST の出力の State フィールドのコピー。これにより、このスレッドが何を実行しているか (マスターに接続しようとしている、マスターからのイベントを待機している、マスターに再接続しているなど) がわかります。可能性のある状態のリストについては、セクション 8.12.5.6「レプリケーションスレーブの I/O スレッド状態」を参照してください。

- Master_Host

  このスレーブが接続されているマスターホスト。

- Master_User

  マスターに接続するために使用されるアカウントのユーザー名。

- Master_Port

  マスターに接続するために使用されるポート。

- Connect_Retry

  接続再試行の間の秒数 (デフォルトは 60)。これは、CHANGE MASTER TO ステートメントで設定できます。

- Master_Log_File

  I/O スレッドが現在読み取っている元のマスターバイナリログファイルの名前。

- Read_Master_Log_Pos

  現在のマスターバイナリログファイル内の I/O スレッドが最後に読み取った位置。

- Relay_Log_File

  SQL スレッドが現在読み取って実行している元のリレーログファイルの名前。

- Relay_Log_Pos

  現在のリレーログファイル内の SQL スレッドが最後に読み取って実行した位置。

- Relay_Master_Log_File

  SQL スレッドによって実行された最新のイベントを含むマスターバイナリログファイルの名前。

- Slave_IO_Running

  I/O スレッドが起動され、マスターに正常に接続したかどうか。内部的には、このスレッドの状態は次の 3 つの値のいずれかによって表されます。

  - MYSQL_SLAVE_NOT_RUN　スレーブ I/O スレッドは実行されていません。この状態の場合、Slave_IO_Running は No です。
  - MYSQL_SLAVE_RUN_NOT_CONNECT　スレーブ I/O スレッドは実行されていますが、レプリケーションマスターに接続されていません。この状態の場合、Slave_IO_Running は、次の表に示すようにサーバーバージョンによって異なります。

MySQL バージョン	Slave_IO_Running
4.1 (4.1.13 以前)、5.0 (5.0.11 以前)	Yes
4.1 (4.1.14 以降)、5.0 (5.0.12 以降)	No
5.1 (5.1.45 以前)	No
5.1 (5.1.46 以降)、5.5、5.6	Connecting

  - MYSQL_SLAVE_RUN_CONNECT　スレーブ I/O スレッドは実行されており、レプリケーションマスターに接続されています。この状態の場合、Slave_IO_Running は Yes です。

  Slave_running システムステータス変数の値は、この値に対応します。

- Slave_SQL_Running

  SQL スレッドが起動されたかどうか。

- Replicate_Do_DB、Replicate_Ignore_DB

  --replicate-do-db および --replicate-ignore-db オプションで指定されたデータベースのリスト (存在する場合)。

- Replicate_Do_Table、Replicate_Ignore_Table、Replicate_Wild_Do_Table、Replicate_Wild_Ignore_Table

  --replicate-do-table、--replicate-ignore-table、--replicate-wild-do-table、および --replicate-wild-ignore-table オプションで指定されたテーブルのリスト (存在する場合)。

- Last_Errno、Last_Error

  これらのカラムは、Last_SQL_Errno および Last_SQL_Error のエイリアスです。

  RESET MASTER または RESET SLAVE を発行すると、これらのカラムに表示されている値がリセットされます。

  > **注記**
  >
  > スレーブ SQL スレッドは、エラーを受信すると、まずそのエラーをレポートしてから SQL スレッドを停止します。つまり、Slave_SQL_Running にまだ Yes が表示されているにもかかわらず、SHOW SLAVE STATUS が Last_SQL_Errno に対して 0 以外の値を示す短い時間帯が存在します。

- Skip_Counter

  sql_slave_skip_counter システム変数の現在の値。セクション13.4.2.4「SET GLOBAL sql_slave_skip_counter 構文」を参照してください。

- Exec_Master_Log_Pos

  現在のマスターバイナリログファイル内の、SQL スレッドが処理対象の次のトランザクションまたはイベントの先頭をマークしながら読み取って実行した最後の位置。既存のスレーブから新しいスレーブを起動するときに、その新しいスレーブがこの位置から読み取りを開始するように、CHANGE MASTER TO ステートメントの MASTER_LOG_POS オプションでこの値を使用できます。マスターのバイナリログ内の (Relay_Master_Log_File、Exec_Master_Log_Pos) で指定される座標は、リレーログ内の (Relay_Log_File、Relay_Log_Pos) で指定される座標に対応します。

  (MySQL 5.6.3 以降で slave_parallel_workers を 0 以外の値に設定することによって) マルチスレッドスレーブを使用している場合は、このカラム内の値が実際に、コミットされていないトランザクションがその前には残っていない「低位境界」値を表します。現在の実装では、別のデータベース上のトランザクションをスレーブ上でマスター上とは異なる順序で実行することが許可されるため、これは必ずしも、直近で実行されたトランザクションの位置ではありません。

- Relay_Log_Space

  既存のすべてのリレーログファイルの合計サイズ。

- Until_Condition、Until_Log_File、Until_Log_Pos

  START SLAVE ステートメントの UNTIL 句で指定された値。

  Until_Condition の値は次のとおりです。

  - UNTIL 句が指定されなかった場合は None
  - このスレーブがマスターのバイナリログ内の特定の位置まで読み取っている場合は Master
  - このスレーブがそのリレーログ内の特定の位置まで読み取っている場合は Relay

  MySQL 5.6.6 からは、次の値を持つ、GTID に関連した UNTIL 句が追加されました。

  - スレーブ SQL スレッドが、gtid_set に GTID がリストされている最初のトランザクションに達するまでトランザクションを処理している場合は、SQL_BEFORE_GTIDS。
  - スレーブスレッドが、gtid_set 内の最後のトランザクションが両方のスレッドによって処理されるまですべてのトランザクションを処理している場合は、SQL_AFTER_GTIDS。
  - マルチスレッドスレーブの SQL スレッドが、リレーログ内にそれ以上ギャップが見つからなくなるまで実行する場合は SQL_AFTER_MTS_GAPS。

Until_Log_File と Until_Log_Pos は、ログファイルの名前と、SQL スレッドが実行を停止する座標を定義している位置を示します。

UNTIL 句の詳細は、セクション13.4.2.5「START SLAVE 構文」を参照してください。

- Master_SSL_Allowed、Master_SSL_CA_File、Master_SSL_CA_Path、Master_SSL_Cert、Master_SSL_Cipher、Master_SSL

  これらのフィールドは、このスレーブがマスターに接続するために使用する SSL パラメータを表示します (存在する場合)。

  Master_SSL_Allowed の値は次のとおりです。

  - マスターへの SSL 接続が許可されている場合は Yes
  - マスターへの SSL 接続が許可されていない場合は No
  - SSL 接続は許可されているが、スレーブサーバーで SSL サポートが有効になっていない場合は Ignored

  その他の SSL 関連フィールドの値は、CHANGE MASTER TO ステートメントに対する MASTER_SSL_CA、MASTER_SSL_CAPATH、MASTER_SSL_CERT、MASTER_SSL_CIPHER、MASTER_SSL_CRL、MAS および MASTER_SSL_VERIFY_SERVER_CERT オプションの値に対応します。セクション 13.4.2.1「CHANGE MASTER TO 構文」を参照してください。

  Master_SSL_CRL_File と Master_SSL_CRL_Path は MySQL 5.6.3 で追加されました。

- Seconds_Behind_Master

  このフィールドは、このスレーブがどれだけ「遅延している」かを示します。

  - このスレーブが更新をアクティブに処理している場合、このフィールドは、現在このスレーブ上で処理されている最新のイベントに対するスレーブ上の現在のタイムスタンプと、マスター上でログに記録されている元のタイムスタンプの違いを示します。
  - 現在このスレーブ上でイベントが処理されていない場合、この値は 0 です。

  基本的に、このフィールドは、スレーブ SQL スレッドとスレーブ I/O スレッドの間の時間差 (秒単位) を測定します。マスターとスレーブの間のネットワーク接続が高速である場合は、スレーブ I/O スレッドがマスターに非常に近いため、このフィールドは、スレーブ SQL スレッドがマスターに比べてどれだけ遅延しているかを示す良い近似値になります。このネットワークが低速である場合、これは良い近似値ではありません。スレーブ SQL スレッドが、読み取りの遅いスレーブ I/O スレッドに非常に頻繁に追い付かれる可能性があるため、I/O スレッドがマスターに比べて遅延している場合でも、Seconds_Behind_Master は多くの場合 0 の値を示します。つまり、このカラムは高速ネットワークの場合にのみ有効です。

  この時間差の計算は、スレーブ I/O スレッドの起動時に計算された時間差がそれ以降も一定のままであれば、マスターとスレーブのクロック時間が同じでない場合でも機能します。何らかの変更 (NTP の更新を含む) があると、Seconds_Behind_Master の計算の信頼性を低下させるクロックスキューが発生する場合があります。

  MySQL 5.6.9 以降では、このフィールドは、スレーブ SQL スレッドが実行されていない場合、または SQL スレッドがすべてのリレーログを消費し、かつスレーブ I/O スレッドが実行されていない場合に NULL (未定義または不明) になります。以前は、このフィールドは、スレーブ SQL スレッドまたはスレーブ I/O スレッドが実行されていないか、あるいはマスターに接続されていない場合に NULL になりました。(Bug #12946333) たとえば、(MySQL 5.6.9 より前) スレーブ I/O スレッドが実行されているが、マスターに接続されておらず、再接続の前に CHANGE MASTER TO ステートメントまたは --master-connect-retry オプションで指定された秒数 (デフォルトは 60) だけスリープしている場合、この値は NULL でした。現在、このようなケースでは、マスターへの接続はテストされません。代わりに、I/O スレッドが実行されているが、リレーログが使い果たされた場合は、Seconds_Behind_Master が 0 に設定されます。

  Seconds_Behind_Master の値は、イベント内に格納されているタイムスタンプに基づいており、このタイムスタンプはレプリケーションを通して保持されます。つまり、マスター M1 がそれ自体 M0 のスレーブである場合、M0 のバイナリログに起因する M1 のバイナリログからのイベントはすべて、そのイベントに M0 のタイムスタンプを含んでいます。これにより、MySQL は TIMESTAMP を正常にレプリケートできます。ただし、Seconds_Behind_Master に関する問題は、M1 がクライアントからの直接の更新も受信した場合、M1 からの最後のイベントがときには M0 に起因し、またときには M1 上での直接の更新の結果であるため、Seconds_Behind_Master 値がランダムに変動することにあります。

  マルチスレッドスレーブ (MySQL 5.6.3 以降) を使用している場合は、この値が Exec_Master_Log_Pos に基づいているため、直近でコミットされたトランザクションの位置を反映していない可能性があります。

- Last_IO_Errno、Last_IO_Error

  I/O スレッドを停止させた最新のエラーのエラー番号とエラーメッセージ。0 のエラー番号および空の文字列のメッセージは、「エラーなし」を示します。Last_IO_Error 値が空でない場合、このエラー値はスレーブのエラーログにも現れます。

  MySQL 5.6.1 から、I/O エラーの情報には、最新の I/O スレッドエラーがいつ発生したかを示すタイムスタンプが含まれています。このタイムスタンプには、YYMMDD HH:MM:SS という形式が使用されます。

  MySQL 5.6.3 より前は、このタイムスタンプは、Last_IO_Error カラムに表示されるエラーメッセージテキストの前に付加されました。MySQL 5.6.3 以降では、このタイムスタンプは、代わりに Last_SQL_Error_Timestamp カラムに表示されます。

  RESET MASTER または RESET SLAVE を発行すると、これらのカラムに表示されている値がリセットされます。

- Last_SQL_Errno、Last_SQL_Error

  SQL スレッドを停止させた最新のエラーのエラー番号とエラーメッセージ。0 のエラー番号および空の文字列のメッセージは、「エラーなし」を示します。Last_SQL_Error 値が空でない場合、このエラー値はスレーブのエラーログにも現れます。

  MySQL 5.6.1 から、SQL エラーの情報には、最新の SQL スレッドエラーがいつ発生したかを示すタイムスタンプが含まれています。このタイムスタンプには、YYMMDD HH:MM:SS という形式が使用されます。

  MySQL 5.6.3 より前は、このタイムスタンプは、Last_SQL_Error カラムに表示されるエラーメッセージテキストの前に付加されました。MySQL 5.6.3 以降では、このタイムスタンプは、代わりに Last_SQL_Error_Timestamp カラムに表示されます。

  RESET MASTER または RESET SLAVE を発行すると、これらのカラムに表示されている値がリセットされます。

- Replicate_Ignore_Server_Ids

  MySQL 5.6 では、CHANGE MASTER TO ステートメントの IGNORE_SERVER_IDS オプションを使用して、0 以上のマスターからのイベントを無視するようにスレーブを設定します。これはデフォルトではブランクであり、通常は、循環またはその他のマルチマスターレプリケーションセットアップを使用している場合にのみ変更されます。ブランクでないときに Replicate_Ignore_Server_Ids に表示されるメッセージは、無視されるサーバー ID を示す 1 つ以上の番号のカンマ区切りリストで構成されます。例:

  ```
  Replicate_Ignore_Server_Ids: 2, 6, 9
  ```

  > **注記**
  >
  > Ignored_server_ids も無視されるサーバー ID を示しますが、これは、無視されるサーバー ID の総数が前にあるスペースで区切られたリストです。たとえば、サーバー ID 2、6、または 9 を持つマスターを無視するようスレーブに指示するために IGNORE_SERVER_IDS = (2,6,9) オプションを含む CHANGE MASTER TO ステートメントが発行された場合、その情報は次のように表示されます。
  >
  > ```
  > Ignored_server_ids: 3 2 6 9
  > ```
  >
  > ここで、3 は、無視されるサーバー ID の総数です。

  Replicate_Ignore_Server_Ids のフィルタリングは SQL スレッドではなく、I/O スレッドによって実行されます。つまり、フィルタで除外されるイベントはリレーログに書き込まれません。これは、SQL スレッドに適用される --replicate-do-table などのサーバーオプションによって実行されるフィルタリングアクションとは異なります。

- Master_Server_Id

  マスターからの server_id 値。

- Master_UUID

  マスターからの server_uuid 値。このフィールドは、MySQL 5.6.0 で追加されました。

- Master_Info_File

  master.info ファイルの場所。このフィールドは、MySQL 5.6.0 で追加されました。

- SQL_Delay

  このスレーブがマスターから遅延する必要がある秒数。このフィールドは、MySQL 5.6.0 で追加されました。

- SQL_Remaining_Delay

  Slave_SQL_Running_State が Waiting until MASTER_DELAY seconds after master executed event である場合、このフィールドには残りの遅延秒数が含まれます。ほかのときは、このフィールドは NULL です。このフィールドは、MySQL 5.6.0 で追加されました。

- Slave_SQL_Running_State

  SQL スレッドの状態 (Slave_IO_State に類似しています)。この値は、SHOW PROCESSLIST によって表示される SQL スレッドの State 値と同じです。セクション8.12.5.7「レプリケーションスレーブ SQL スレッドの状態」には、可能性のある状態のリストが示されています。このフィールドは、MySQL 5.6.0 で追加されました。

- Master_Retry_Count

  接続が失われた場合に、このスレーブがマスターへの再接続を試みることができる回数。この値は、CHANGE MASTER TO ステートメントの MASTER_RETRY_COUNT オプション (推奨)、または古い --master-retry-count サーバーオプション (下位互換性のために引き続きサポートされています) を使用して設定できます。このフィールドは、MySQL 5.6.1 で追加されました。

- Master_Bind

  このスレーブのバインド先のネットワークインタフェース (存在する場合)。これは、CHANGE MASTER TO ステートメントの MASTER_BIND オプションを使用して設定されます。

  このカラムは、MySQL 5.6.2 で追加されました。

- Last_IO_Error_Timestamp

  最新の I/O エラーがいつ発生したかを示す YYMMDD HH:MM:SS 形式のタイムスタンプ。

  このカラムは、MySQL 5.6.3 で追加されました。以前 MySQL 5.6 では、このタイムスタンプは、Last_IO_Error に表示されるエラーテキストの前に付加されました。

- Last_SQL_Error_Timestamp

  最新の SQL エラーがいつ発生したかを示す YYMMDD HH:MM:SS 形式のタイムスタンプ。

  このカラムは、MySQL 5.6.3 で追加されました。以前 MySQL 5.6 では、このタイムスタンプは、Last_SQL_Error に表示されるエラーテキストの前に付加されました。

- Retrieved_Gtid_Set

  このスレーブによって受信されたすべてのトランザクションに対応するグローバルトランザクション ID のセット。GTID が使用されていない場合は空です。

  これは、リレーログ内に存在するか、またはこれまでに存在したすべての GTID のセットです。各 GTID は、Gtid_log_event が受信されるとすぐに追加されます。そのため、このセットには、部分的に転送されたトランザクションの GTID も含まれる場合があります。

  RESET SLAVE または CHANGE MASTER TO を実行したため、あるいは --relay-log-recovery オプションの効果のためにすべてのリレーログが失われた場合は、このセットがクリアされます。relay_log_purge = 1 のときは、最新のリレーログが常に保持されるため、このセットはクリアされません。

  このカラムは、MySQL 5.6.5 で追加されました。

  MySQL 5.6.10 より前は、この値は大文字で出力されました。MySQL 5.6.10 以降では、常に小文字で出力されます。(Bug #15869441)

- Executed_Gtid_Set

  バイナリログに書き込まれたグローバルトランザクション ID のセット。これは、このサーバー上のグローバルな gtid_executed システム変数 (MySQL 5.6.9 より前の名前は gtid_done) の値や、このサーバー上での SHOW MASTER STATUS の出力内の Executed_Gtid_Set の値と同じです。GTID が使用されていない場合は空です。

  このカラムは、MySQL 5.6.5 で追加されました。

  MySQL 5.6.10 より前は、この値は大文字で出力されました。MySQL 5.6.10 以降では、常に小文字で出力されます。(Bug #15869441)

- Auto_Position

  自動ポジショニングが使用されている場合は 1、それ以外の場合は 0。

  このカラムは、MySQL 5.6.10 で追加されました。(Bug #15992220)

13.7.5.36 SHOW STATUS 構文

```
SHOW [GLOBAL | SESSION] STATUS
    [LIKE 'pattern' | WHERE expr]
```

SHOW STATUS は、サーバーステータス情報を提供します。この情報はまた、mysqladmin extended-status コマンドを使用して取得することもできます。LIKE 句 (存在する場合) は、どの変数名と照合するかを示します。セクション21.32「SHOW ステートメントの拡張」で説明されているように、WHERE 句を指定すると、より一般的な条件を使用して行を選択できます。このステートメントにはどの権限も必要ありません。これには、サーバーに接続できることのみが必要です。

部分的な出力を次に示します。名前と値のリストが、実際のサーバーでは異なることがあります。各変数の意味は、セクション5.1.6「サーバーステータス変数」に示されています。

```
mysql> SHOW STATUS;
+---------------------------+------------+
| Variable_name             | Value      |
+---------------------------+------------+
| Aborted_clients           | 0          |
| Aborted_connects          | 0          |
| Bytes_received            | 155372598  |
| Bytes_sent                | 1176560426 |
| Connections               | 30023      |
| Created_tmp_disk_tables   | 0          |
| Created_tmp_tables        | 8340       |
| Created_tmp_files         | 60         |
...
| Open_tables               | 1          |
| Open_files                | 2          |
| Open_streams              | 0          |
| Opened_tables             | 44600      |
| Questions                 | 2026873    |
...
| Table_locks_immediate     | 1920382    |
| Table_locks_waited        | 0          |
| Threads_cached            | 0          |
| Threads_created           | 30022      |
| Threads_connected         | 1          |
| Threads_running           | 1          |
| Uptime                    | 80380      |
+---------------------------+------------+
```

LIKE 句を指定すると、このステートメントは、そのパターンに一致する名前を持つ変数の行のみを表示します。

```
mysql> SHOW STATUS LIKE 'Key%';
+--------------------+----------+
| Variable_name      | Value    |
+--------------------+----------+
| Key_blocks_used    | 14955    |
| Key_read_requests  | 96854827 |
| Key_reads          | 162040   |
| Key_write_requests | 7589728  |
| Key_writes         | 3813196  |
+--------------------+----------+
```

GLOBAL 修飾子を指定すると、SHOW STATUS は、MySQL へのすべての接続のステータス値を表示します。SESSION を指定すると、現在の接続のステータス値を表示します。修飾子が存在しない場合、デフォルトは SESSION です。LOCAL は SESSION のシノニムです。

一部のステータス変数にはグローバル値しかありません。これらの変数では、GLOBAL と SESSION の両方に同じ値が表示されます。各ステータス変数のスコープは、セクション5.1.6「サーバーステータス変数」に示されています。

SHOW STATUS ステートメントを呼び出すたびに内部一時テーブルが使用され、グローバルの Created_tmp_tables 値が増加します。

13.7.5.37 SHOW TABLE STATUS 構文

```
SHOW TABLE STATUS [{FROM | IN} db_name]
    [LIKE 'pattern' | WHERE expr]
```

SHOW TABLE STATUS は SHOW TABLES のように機能しますが、TEMPORARY 以外の各テーブルに関する多くの情報を提供します。このリストはまた、mysqlshow --status db_name コマンドを使用して取得することもできます。LIKE 句 (存在する場合) は、どのテーブル名と照合するかを示します。セクション21.32「SHOW ステートメントの拡張」で説明されているように、WHERE 句を指定すると、より一般的な条件を使用して行を選択できます。

このステートメントはまた、ビューに関する情報も表示します。

SHOW TABLE STATUS の出力には、次のカラムがあります。

- Name

  テーブルの名前。

- Engine

  このテーブルのストレージエンジン。第15章「代替ストレージエンジン」を参照してください。

- Version

  このテーブルの .frm ファイルのバージョン番号。

- Row_format

  行ストレージフォーマット (Fixed、Dynamic、Compressed、Redundant、Compact)。MyISAM テーブルの場合、Dynamic は、myisamchk -dvv が Packed としてレポートするものに対応します。InnoDB テーブルのフォーマットは、Redundant または Compact としてレポートされます。InnoDB Plugin の Barracuda ファイル形式の場合、このフォーマットは Compressed または Dynamic である可能性があります。

- Rows

  行数。MyISAM などの一部のストレージエンジンは、正確な数を格納します。InnoDB などのその他のストレージエンジンの場合、この値は近似値であり、実際の値から 40 から 50% 変動する可能性があります。このような場合、正確な数を取得するには SELECT COUNT(*) を使用します。

  INFORMATION_SCHEMA データベース内のテーブルの場合、Rows 値は NULL です。

- Avg_row_length

  平均行長。

- Data_length

  データファイルの長さ。

- Max_data_length

  データファイルの最大長。これは、このテーブル内に格納できるデータの合計バイト数です (使用されるデータポインタサイズが指定された場合)。

- Index_length

  インデックスファイルの長さ。

- Data_free

  割り当てられているが、使用されていないバイト数。

この情報はまた、InnoDB テーブルに対しても示されます (以前は、Comment 値に含まれていました)。InnoDB テーブルは、このテーブルが属するテーブルスペースの空き領域をレポートします。共有テーブルスペース内に存在するテーブルの場合、これはその共有テーブルスペースの空き領域です。複数のテーブルスペースを使用していて、このテーブルに独自のテーブルスペースがある場合は、そのテーブルのみの空き領域になります。空き領域とは、完全な空きエクステントから安全上のマージンを引いたバイト数を示します。空き領域が 0 として表示されている場合でも、新しいエクステントを割り当てる必要がないかぎり、行を挿入できる可能性があります。

パーティション化されたテーブルの場合、この値は推定値にすぎず、絶対的に正しいとはかぎりません。このような場合、この情報を取得するためのより正確な方法は、次の例に示すように INFORMATION_SCHEMA.PARTITIONS テーブルへのクエリーです。

```
SELECT    SUM(DATA_FREE)
  FROM  INFORMATION_SCHEMA.PARTITIONS
  WHERE TABLE_SCHEMA = 'mydb'
  AND   TABLE_NAME   = 'mytable';
```

詳細は、セクション21.13「INFORMATION_SCHEMA PARTITIONS テーブル」を参照してください。

- Auto_increment

  次の AUTO_INCREMENT 値。

- Create_time

  いつテーブルが作成されたか。

- Update_time

  いつデータファイルが最後に更新されたか。一部のストレージエンジンでは、この値は NULL です。たとえば、InnoDB はそのシステムテーブルスペース内に複数のテーブルを格納するため、データファイルのタイムスタンプは適用されません。各 InnoDB テーブルが個別の .ibd ファイル内に存在する file-per-table モードの場合でも、変更バッファリングによってデータファイルへの書き込みが遅延される可能性があるため、ファイルの変更時間は最後の挿入、更新、または削除の時間とは異なります。MyISAM の場合は、データファイルのタイムスタンプが使用されます。ただし、Windows ではタイムスタンプが更新によって更新されないため、この値は不正確です。

- Check_time

  いつテーブルが最後にチェックされたか。すべてのストレージエンジンがこの時間を更新するわけではありません。その場合、この値は常に NULL です。

- Collation

  このテーブルの文字セットと照合順序。

- Checksum

  ライブチェックサム値 (存在する場合)。

- Create_options

  CREATE TABLE で使用される追加のオプション。CREATE TABLE が呼び出されるときに指定される元のオプションは保持されており、ここでレポートされるオプションは、アクティブなテーブル設定やオプションとは異なる可能性があります。

- Comment

  このテーブルを作成するときに使用されたコメント (または、MySQL がテーブル情報にアクセスできなかった理由に関する情報)。

MEMORY テーブルの場合、Data_length、Max_data_length、および Index_length 値はほぼ、割り当てられているメモリーの実際の量を表します。割り当てアルゴリズムは、割り当て操作の数を減らすために、大量のメモリーを確保します。

NDB テーブルの場合、このステートメントの出力は Avg_row_length および Data_length カラムの適切な値を示しますが、例外として BLOB カラムは考慮に入れられません。

ビューの場合は、Name がビュー名を示し、Comment が view になる点を除き、SHOW TABLE STATUS によって表示されるすべてのフィールドが NULL です。

13.7.5.38 SHOW TABLES 構文

```
SHOW [FULL] TABLES [{FROM | IN} db_name]
    [LIKE 'pattern' | WHERE expr]
```

SHOW TABLES は、特定のデータベース内の TEMPORARY 以外のテーブルを一覧表示します。このリストはまた、mysqlshow db_name コマンドを使用して取得することもできます。LIKE 句 (存在する場合) は、どのテーブル名と照合するかを示します。セクション21.32「SHOW ステートメントの拡張」で説明されているように、WHERE 句を指定すると、より一般的な条件を使用して行を選択できます。

LIKE 句によって実行される照合は、lower_case_table_names システム変数の設定に依存します。

このステートメントはまた、このデータベース内のビューもすべて一覧表示します。FULL 修飾子は、SHOW FULL TABLES が 2 番目の出力カラムを表示するようにサポートされます。2 番目のカラムの値は、テーブルの場合は BASE TABLE、ビューの場合は VIEW です。

ベーステーブルまたはビューに対する権限を持っていない場合、そのテーブルまたはビューは SHOW TABLES または mysqlshow db_name の出力に表示されません。

13.7.5.39 SHOW TRIGGERS 構文

```
SHOW TRIGGERS [{FROM | IN} db_name]
    [LIKE 'pattern' | WHERE expr]
```

SHOW TRIGGERS は、データベース (FROM 句が指定されていないかぎり、デフォルトデータベース) 内のテーブルに対して現在定義されているトリガーを一覧表示します。このステートメントは、ユーザーが TRIGGER 権限を持っているデータベースとテーブルに対してのみ結果を返します。LIKE 句 (存在する場合) は、(トリガー名ではなく) どのテーブル名と照合するかを示し、このステートメントでそのテーブルのトリガーを表示するようにします。セクション21.32「SHOW ステートメントの拡張」で説明されているように、WHERE 句を指定すると、より一般的な条件を使用して行を選択できます。

セクション20.3「トリガーの使用」で定義されているトリガー ins_sum の場合、このステートメントの出力は次に示すようになります。

```
mysql> SHOW TRIGGERS LIKE 'acc%'\G
*************************** 1. row ***************************
             Trigger: ins_sum
               Event: INSERT
               Table: account
           Statement: SET @sum = @sum + NEW.amount
              Timing: BEFORE
             Created: NULL
            sql_mode: NO_ENGINE_SUBSTITUTION
             Definer: me@localhost
character_set_client: utf8
collation_connection: utf8_general_ci
  Database Collation: latin1_swedish_ci
```

SHOW TRIGGERS の出力には、次のカラムがあります。

- Trigger: トリガー名。
- Event: このトリガーがアクティブ化される操作の種類。この値は、'INSERT'、'UPDATE'、または 'DELETE' です。
- Table: このトリガーが定義されているテーブル。
- Statement: トリガー本体。つまり、このトリガーがアクティブ化されるときに実行されるステートメント。
- Timing: このトリガーが、トリガーイベントの前またはあとのどちらにアクティブ化されるか。値は 'BEFORE' または 'AFTER' です。
- Created: 現在、このカラムの値は常に NULL です。
- sql_mode: このトリガーが実行されるときに有効な SQL モード。
- Definer: このトリガーを作成したユーザーのアカウント。形式は 'user_name'@'host_name' です。
- character_set_client: このトリガーが作成されたときの character_set_client システム変数のセッション値。
- collation_connection: このトリガーが作成されたときの collation_connection システム変数のセッション値。
- Database Collation: このトリガーが関連付けられているデータベースの照合順序。

また、TRIGGERS テーブルを含む INFORMATION_SCHEMA からトリガーオブジェクトに関する情報を取得することもできます。セクション21.26「INFORMATION_SCHEMA TRIGGERS テーブル」を参照してください。

13.7.5.40 SHOW VARIABLES 構文

```
SHOW [GLOBAL | SESSION] VARIABLES
    [LIKE 'pattern' | WHERE expr]
```

SHOW VARIABLES は、MySQL システム変数の値を表示します。この情報はまた、mysqladmin variables コマンドを使用して取得することもできます。LIKE 句 (存在する場合) は、どの変数名と照合するかを示します。セクション21.32「SHOW ステートメントの拡張」で説明されているように、WHERE 句を指定すると、より一般的な条件を使用して行を選択できます。このステートメントにはどの権限も必要ありません。これには、サーバーに接続できることのみが必要です。

GLOBAL 修飾子を指定すると、SHOW VARIABLES は、MySQL への新しい接続に使用される値を表示します。MySQL 5.6 では、変数にグローバル値がない場合、値は表示されません。SESSION を指定すると、SHOW VARIABLES は、現在の接続に有効な値を表示します。修飾子が存在しない場合、デフォルトは SESSION です。LOCAL は SESSION のシノニムです。

SHOW VARIABLES は、バージョンに依存する表示幅の制限に従います。完全には表示されない非常に長い値を持つ変数の場合、回避策として SELECT を使用します。例:

```
SELECT @@GLOBAL.innodb_data_file_path;
```

デフォルトのシステム変数値が適切でない場合は、mysqld の起動時にコマンドオプションを使用して設定できるほか、そのほとんどは SET ステートメントで実行時に変更できます。セクション5.1.5「システム変数の使用」およびセクション13.7.4「SET 構文」を参照してください。

部分的な出力を次に示します。名前と値のリストが、実際のサーバーでは異なることがあります。セクション5.1.4「サーバーシステム変数」では、各変数の意味について説明しています。また、セクション8.11.2「サーバーパラメータのチューニング」には、そのチューニングに関する情報が示されています。

```
mysql> SHOW VARIABLES;
+-----------------------------------------+----------------------------+
| Variable_name                           | Value                      |
+-----------------------------------------+----------------------------+
| auto_increment_increment                | 1                          |
| auto_increment_offset                   | 1                          |
| autocommit                              | ON                         |
| automatic_sp_privileges                 | ON                         |
| back_log                                | 50                         |
| basedir                                 | /home/jon/bin/mysql-5.5    |
| big_tables                              | OFF                        |
| binlog_cache_size                       | 32768                      |
| binlog_direct_non_transactional_updates | OFF                        |
| binlog_format                           | STATEMENT                  |
| binlog_stmt_cache_size                  | 32768                      |
| bulk_insert_buffer_size                 | 8388608                    |
...
| max_allowed_packet                      | 1048576                    |
| max_binlog_cache_size                   | 18446744073709547520       |
| max_binlog_size                         | 1073741824                 |
| max_binlog_stmt_cache_size              | 18446744073709547520       |
| max_connect_errors                      | 10                         |
| max_connections                         | 151                        |
| max_delayed_threads                     | 20                         |
| max_error_count                         | 64                         |
| max_heap_table_size                     | 16777216                   |
| max_insert_delayed_threads              | 20                         |
| max_join_size                           | 18446744073709551615       |
...

| thread_handling                         | one-thread-per-connection  |
| thread_stack                            | 262144                     |
| time_format                             | %H:%i:%s                   |
| time_zone                               | SYSTEM                     |
| timed_mutexes                           | OFF                        |
| timestamp                               | 1316689732                 |
| tmp_table_size                          | 16777216                   |
| tmpdir                                  | /tmp                       |
| transaction_alloc_block_size            | 8192                       |
| transaction_prealloc_size               | 4096                       |
| tx_isolation                            | REPEATABLE-READ            |
| unique_checks                           | ON                         |
| updatable_views_with_limit              | YES                        |
```

```
| version                          | 5.5.17-log             |
| version_comment                  | Source distribution    |
| version_compile_machine          | x86_64                 |
| version_compile_os               | Linux                  |
| wait_timeout                     | 28800                  |
| warning_count                    | 0                      |
+----------------------------------+------------------------+
```

LIKE 句を指定すると、このステートメントは、そのパターンに一致する名前を持つ変数の行のみを表示します。特定の変数の行を取得するには、LIKE 句を次に示すように使用します。

```
SHOW VARIABLES LIKE 'max_join_size';
SHOW SESSION VARIABLES LIKE 'max_join_size';
```

名前がパターンと一致する変数のリストを取得するには、LIKE 句の中で 「%」 のワイルドカード文字を使用します。

```
SHOW VARIABLES LIKE '%size%';
SHOW GLOBAL VARIABLES LIKE '%size%';
```

ワイルドカード文字は、照合されるパターン内のどの場所でも利用できます。厳密には、「_」 は任意の 1 文字と一致するワイルドカードであるため、文字どおりに照合するには 「_」 としてエスケープしてください。実際には、これはほとんど必要ありません。

13.7.5.41 SHOW WARNINGS 構文

```
SHOW WARNINGS [LIMIT [offset,] row_count]
SHOW COUNT(*) WARNINGS
```

SHOW WARNINGS は、現在のセッションでのステートメントの実行の結果として得られた条件 (エラー、警告、および注意) に関する情報を表示する診断ステートメントです。警告は、INSERT、UPDATE、LOAD DATA INFILE などの DML ステートメントのほか、CREATE TABLE や ALTER TABLE などの DDL ステートメントに対して生成されます。

LIMIT 句の構文は、SELECT ステートメントの場合と同じです。セクション13.2.9「SELECT 構文」を参照してください。

SHOW WARNINGS はまた、EXTENDED キーワードが使用されている EXPLAIN によって生成された追加の情報を表示するために、EXPLAIN EXTENDED のあとにも使用されます。セクション8.8.3「EXPLAIN EXTENDED 出力フォーマット」を参照してください。

SHOW WARNINGS は、メッセージを生成した、現在のセッションでの最新のステートメントの結果として得られた条件に関する情報を表示します。最新のステートメントがテーブルを使用し、メッセージを生成しなかった場合は、何も表示しません。(つまり、テーブルを使用するが、メッセージを生成しないステートメントはメッセージリストをクリアします。) テーブルを使用せず、メッセージを生成しないステートメントは、メッセージリストに影響を与えません。

SHOW COUNT(*) WARNINGS 診断ステートメントは、エラー、警告、および注意の総数を表示します。この数はまた、warning_count システム変数からも取得できます。

```
SHOW COUNT(*) WARNINGS;
SELECT @@warning_count;
```

関連する診断ステートメント SHOW ERRORS はエラー状態のみ (警告と注意は除外されます) を表示し、SHOW COUNT(*) ERRORS ステートメントはエラーの総数を表示します。セクション13.7.5.18「SHOW ERRORS 構文」を参照してください。GET DIAGNOSTICS を使用すると、個々の条件に関する情報を検査できます。セクション13.6.7.3「GET DIAGNOSTICS 構文」を参照してください。

INSERT でのデータ変換の警告を表示する単純な例を次に示します。

```
mysql> CREATE TABLE t1 (a TINYINT NOT NULL, b CHAR(4));
Query OK, 0 rows affected (0.05 sec)

mysql> INSERT INTO t1 VALUES(10,'mysql'), (NULL,'test'), (300,'xyz');
Query OK, 3 rows affected, 3 warnings (0.00 sec)
Records: 3  Duplicates: 0  Warnings: 3

mysql> SHOW WARNINGS\G
*************************** 1. row ***************************
  Level: Warning
   Code: 1265
Message: Data truncated for column 'b' at row 1
*************************** 2. row ***************************
  Level: Warning
   Code: 1048
```

```
Message: Column 'a' cannot be null
*************************** 3. row ***************************
  Level: Warning
   Code: 1264
Message: Out of range value for column 'a' at row 3
3 rows in set (0.00 sec)
```

max_error_count システム変数は、サーバーが情報を格納する対象となるエラー、警告、および注意メッセージの最大数、したがって SHOW WARNINGS が表示するメッセージの数を制御します。サーバーが格納できるメッセージの数を変更するには、max_error_count の値を変更します。デフォルトは 64 です。

max_error_count は、カウントされるメッセージの数ではなく、格納されるメッセージの数のみを制御します。生成されたメッセージの数が max_error_count を超えた場合でも、warning_count の値は max_error_count によって制限されません。この点について次の例で説明します。この ALTER TABLE ステートメントは、3 つの警告メッセージを生成します (この例では、変換の問題が 1 つ発生したあとにエラーが発生しないように、厳密な SQL モードが無効になっています)。max_error_count が 1 に設定されたため、格納されて表示されたメッセージは 1 つだけですが、warning_count の値で示されているように 3 つすべてがカウントされています。

```
mysql> SHOW VARIABLES LIKE 'max_error_count';
+-----------------+-------+
| Variable_name   | Value |
+-----------------+-------+
| max_error_count | 64    |
+-----------------+-------+
1 row in set (0.00 sec)

mysql> SET max_error_count=1, sql_mode = '';
Query OK, 0 rows affected (0.00 sec)

mysql> ALTER TABLE t1 MODIFY b CHAR;
Query OK, 3 rows affected, 3 warnings (0.00 sec)
Records: 3  Duplicates: 0  Warnings: 3

mysql> SHOW WARNINGS;
+---------+------+-----------------------------------------+
| Level   | Code | Message                                 |
+---------+------+-----------------------------------------+
| Warning | 1263 | Data truncated for column 'b' at row 1 |
+---------+------+-----------------------------------------+
1 row in set (0.00 sec)

mysql> SELECT @@warning_count;
+-----------------+
| @@warning_count |
+-----------------+
|               3 |
+-----------------+
1 row in set (0.01 sec)
```

メッセージの格納を無効にするには、max_error_count を 0 に設定します。この場合、warning_count は引き続き、発生した警告の数を示しますが、メッセージは格納されないため表示できません。

sql_notes システム変数は、注意メッセージで warning_count が増分されるかどうか、またサーバーがそれらを格納するかどうかを制御します。デフォルトでは、sql_notes は 1 ですが、0 に設定されている場合は、注意で warning_count が増分されず、またサーバーはそれらを格納しません。

```
mysql> SET sql_notes = 1;
mysql> DROP TABLE IF EXISTS test.no_such_table;
Query OK, 0 rows affected, 1 warning (0.00 sec)
mysql> SHOW WARNINGS;
+-------+------+----------------------------------------+
| Level | Code | Message                                |
+-------+------+----------------------------------------+
| Note  | 1051 | Unknown table 'test.no_such_table' |
+-------+------+----------------------------------------+
1 row in set (0.00 sec)

mysql> SET sql_notes = 0;
mysql> DROP TABLE IF EXISTS test.no_such_table;
Query OK, 0 rows affected (0.00 sec)
mysql> SHOW WARNINGS;
Empty set (0.00 sec)
```

MySQL サーバーは、各クライアントに、そのクライアントによって実行された最新のステートメントの結果として得られたエラー、警告、および注意の総数を示す数を送信します。C API からは、この値は mysql_warning_count() を呼び出すことによって取得できます。セクション 23.7.7.73「mysql_warning_count()」を参照してください。

13.7.6 その他の管理ステートメント

13.7.6.1 BINLOG 構文

```
BINLOG 'str'
```

BINLOG は、内部で使用されるステートメントです。これは、バイナリログファイル内の特定のイベントの印刷可能な表現として mysqlbinlog プログラムによって生成されます。(セクション4.6.8「mysqlbinlog — バイナリログファイルを処理するためのユーティリティー」を参照してください。)'str' 値は、サーバーが、対応するイベントによって示されているデータ変更を判定するためにデコードする、base 64 でエンコードされた文字列です。このステートメントには、SUPER 権限が必要です。

MySQL 5.6 の時点では、このステートメントは形式記述イベントと行イベントのみを実行できます。以前は、すべてのタイプのイベントを実行できました。

13.7.6.2 CACHE INDEX 構文

```
CACHE INDEX
  tbl_index_list [, tbl_index_list] ...
  [PARTITION (partition_list | ALL)]
  IN key_cache_name

tbl_index_list:
  tbl_name [[INDEX|KEY] (index_name[, index_name] ...)]

partition_list:
  partition_name[, partition_name][, ...]
```

CACHE INDEX ステートメントは、テーブルインデックスを特定のキーキャッシュに割り当てます。これは、MyISAM テーブルにのみ使用されます。インデックスが割り当てられたら、これらのインデックスを、必要に応じて LOAD INDEX INTO CACHE でキャッシュにプリロードできます。

次のステートメントは、テーブル t1、t2、および t3 のインデックスを hot_cache という名前のキーキャッシュに割り当てます。

```
mysql> CACHE INDEX t1, t2, t3 IN hot_cache;
+---------+--------------------+----------+----------+
| Table   | Op                 | Msg_type | Msg_text |
+---------+--------------------+----------+----------+
| test.t1 | assign_to_keycache | status   | OK       |
| test.t2 | assign_to_keycache | status   | OK       |
| test.t3 | assign_to_keycache | status   | OK       |
+---------+--------------------+----------+----------+
```

CACHE INDEX の構文では、テーブルの特定のインデックスのみをキャッシュに割り当てるように指定できます。現在の実装では、テーブルのすべてのインデックスをキャッシュに割り当てるため、テーブル名以外のものを指定する理由は何もありません。

CACHE INDEX ステートメントで参照されるキーキャッシュは、パラメータ設定ステートメントを使用して、またはサーバーのパラメータ設定でそのサイズを設定することによって作成できます。例:

```
mysql> SET GLOBAL keycache1.key_buffer_size=128*1024;
```

キーキャッシュのパラメータには、構造化されたシステム変数のメンバーとしてアクセスできます。セクション5.1.5.1「構造化システム変数」を参照してください。

インデックスをキーキャッシュに割り当てるには、そのキーキャッシュが存在している必要があります。

```
mysql> CACHE INDEX t1 IN non_existent_cache;
ERROR 1284 (HY000): Unknown key cache 'non_existent_cache'
```

デフォルトで、テーブルインデックスは、サーバー起動時に作成されるメイン (デフォルト) キーキャッシュに割り当てられます。キーキャッシュが破棄されると、それに割り当てられていたインデックスはすべて、デフォルトのキーキャッシュに再度割り当てられます。

インデックスの割り当ては、サーバーにグローバルに影響を与えます。あるクライアントがインデックスを特定のキャッシュに割り当てると、どのクライアントがクエリーを発行したかには関係なく、このキャッシュはそのインデックスに関連するすべてのクエリーに使用されます。

MySQL 5.6 では、このステートメントは、パーティション化された MyISAM テーブルに対してもサポートされます。1 つ、複数、またはすべてのパーティションの 1 つ以上のインデックスを特定のキーキャッシュに割り当てることができます。たとえば、次のステートメントを実行できます。

```
CREATE TABLE pt (c1 INT, c2 VARCHAR(50), INDEX i(c1))
```

```
  PARTITION BY HASH(c1)
  PARTITIONS 4;

SET GLOBAL kc_fast.key_buffer_size = 128 * 1024;
SET GLOBAL kc_slow.key_buffer_size = 128 * 1024;

CACHE INDEX pt PARTITION (p0) IN kc_fast;
CACHE INDEX pt PARTITION (p1, p3) IN kc_slow;
```

前の一連のステートメントは、次のアクションを実行します。

- 4 つのパーティションを含むパーティション化されたテーブルを作成します。これらのパーティションには、自動的に p0、...、p3 という名前が付けられます。このテーブルには、カラム c1 上に i という名前のインデックスが含まれています。
- kc_fast と kc_slow という名前の 2 つのキーキャッシュを作成します。
- パーティション p0 のインデックスを kc_fast キーキャッシュに、パーティション p1 と p3 のインデックスを kc_slow キーキャッシュに割り当てます。残りのパーティション (p2) のインデックスは、サーバーのデフォルトのキーキャッシュを使用します。

代わりに、テーブル pt 内のすべてのパーティションのインデックスを kc_all という名前の 1 つのキーキャッシュに割り当てる場合は、次の 2 つのステートメントのどちらでも使用できます。

```
CACHE INDEX pt PARTITION (ALL) IN kc_all;

CACHE INDEX pt IN kc_all;
```

今示した 2 つのステートメントは同等であり、このうちのどちらを発行しても効果はまったく同じです。つまり、パーティション化されたテーブルのすべてのパーティションのインデックスを同じキーキャッシュに割り当てる場合、PARTITION (ALL) 句はオプションです。

複数のパーティションのインデックスをキーキャッシュに割り当てる場合、それらのパーティションが連続している必要はなく、それらの名前を特定の順序でリストする必要もありません。キーキャッシュに明示的に割り当てられていないパーティションのインデックスはすべて、自動的にサーバーのデフォルトのキーキャッシュを使用します。

MySQL 5.6 では、インデックスのプリロードも、パーティション化された MyISAM テーブルに対してサポートされます。詳細は、セクション13.7.6.5「LOAD INDEX INTO CACHE 構文」を参照してください。

MySQL 5.6.11 でのみ、このステートメントを発行する前に、gtid_next を AUTOMATIC に設定する必要があります。(Bug #16062608、Bug #16715809、Bug #69045)

13.7.6.3 FLUSH 構文

```
FLUSH [NO_WRITE_TO_BINLOG | LOCAL]
  flush_option [, flush_option] ...
```

FLUSH ステートメントには、さまざまな内部キャッシュをクリアまたはリロードしたり、テーブルをフラッシュしたり、ロックを取得したりするいくつかのバリアント形式があります。FLUSH を実行するには、RELOAD 権限が必要です。あとで説明されているように、特定のフラッシュオプションには追加の権限が必要になることがあります。

デフォルトでは、サーバーは FLUSH ステートメントをバイナリログに書き込み、それらがレプリケーションスレーブにレプリケートされるようにします。ロギングを抑制するには、オプションの NO_WRITE_TO_BINLOG キーワード、またはそのエイリアス LOCAL を指定します。

> **注記**
>
> FLUSH LOGS、FLUSH TABLES WITH READ LOCK (テーブルリスト付き、またはなし)、および FLUSH TABLES tbl_name ... FOR EXPORT は、スレーブにレプリケートされると問題が発生するため、どのような場合でもバイナリログに書き込まれません。

SIGHUP シグナルをサーバーに送信すると、さまざまな形式の FLUSH ステートメントに似たいくつかのフラッシュ操作が発生します。セクション5.1.11「シグナルへのサーバー応答」を参照してください。

FLUSH ステートメントは暗黙的なコミットを発生させます。セクション13.3.3「暗黙的なコミットを発生させるステートメント」を参照してください。

RESET ステートメントは、FLUSH に似ています。レプリケーションでの RESET ステートメントの使用については、セクション13.7.6.6「RESET 構文」を参照してください。

flush_option には、次のいずれかの項目を指定できます。

- DES_KEY_FILE

  サーバーの起動時に --des-key-file オプションで指定されたファイルから DES キーをリロードします。

- HOSTS

  ホストキャッシュを空にします。ホストキャッシュは、一部のホストが IP アドレスを変更したか、または Host 'host_name' is blocked というエラーメッセージが表示された場合にフラッシュするようにしてください。(セクションB.5.2.6「ホスト 'host_name' は拒否されました」を参照してください。) MySQL サーバーへの接続中に、特定のホストに関して max_connect_errors 個を超えるエラーが連続して発生すると、MySQL は何か問題があると見なし、そのホストをそれ以上の接続要求からブロックします。ホストキャッシュをフラッシュすると、ホストからのそれ以上の接続試行が可能になります。max_connect_errors のデフォルト値は 10 です。このエラーメッセージを回避するには、max_connect_errors が大きな値に設定された状態でサーバーを起動します。

- [log_type] LOGS

  log_type オプションを指定しない場合、FLUSH LOGS はすべてのログファイルを閉じて、再度開きます。バイナリロギングが有効になっている場合は、バイナリログファイルのシーケンス番号が、前のファイルを基準にして 1 増分されます。

  log_type オプションを指定すると、指定されたログタイプのみがフラッシュされます。次の log_type オプションが許可されます。

  - BINARY はバイナリログファイルを閉じて、再度開きます。
  - ENGINE は、インストールされているストレージエンジンのフラッシュ可能なログをすべて閉じて、再度開きます。現在、これにより InnoDB はそのログをディスクにフラッシュします。
  - ERROR はエラーログファイルを閉じて、再度開きます。
  - GENERAL は一般的なクエリーログファイルを閉じて、再度開きます。
  - RELAY はリレーログファイルを閉じて、再度開きます。
  - SLOW はスロークエリーログファイルを閉じて、再度開きます。

- PRIVILEGES

  mysql データベース内の付与テーブルから権限をリロードします。

  GRANT、CREATE USER、CREATE SERVER、および INSTALL PLUGIN ステートメントの結果として、サーバーは情報をメモリーにキャッシュします。このメモリーは、対応する REVOKE、DROP USER、DROP SERVER、および UNINSTALL PLUGIN ステートメントによって解放されないため、キャッシュを発生させるステートメントの多数のインスタンスを実行するサーバーでは、メモリー使用量が増加します。このキャッシュされたメモリーは FLUSH PRIVILEGES で解放できます。

- QUERY CACHE

  クエリーキャッシュをデフラグして、そのメモリーをより有効に活用します。FLUSH QUERY CACHE は、FLUSH TABLES や RESET QUERY CACHE とは異なり、キャッシュからクエリーを削除しません。

- STATUS

  このオプションは、現在のスレッドのセッションステータス変数値をグローバル値に追加し、セッション値を 0 にリセットします。一部のグローバル変数も 0 にリセットされる可能性があります。また、(デフォルトおよび指定された) キーキャッシュのカウンタも 0 にリセットし、Max_used_connections を開いている接続の現在の数に設定します。これは、クエリーをデバッグしている場合にのみ使用するようにしてください。セクション1.6「質問またはバグをレポートする方法」を参照してください。

- TABLES

  FLUSH TABLES はテーブルをフラッシュし、使用されているバリアントに応じてロックを取得します。許可される構文については、このセクションのあとの方で説明されています。

- USER_RESOURCES

  時間あたりのすべてのユーザーリソースを 0 にリセットします。これにより、時間あたりの接続、クエリー、または更新の制限に達したクライアントが、ただちにアクティビティーを再開できるようになります。FLUSH

USER_RESOURCES は、最大同時接続に対する制限には適用されません。セクション6.3.4「アカウントリソース制限の設定」を参照してください。

mysqladmin ユーティリティーは、flush-hosts、flush-logs、flush-privileges、flush-status、flush-tables などのコマンドを使用して、いくつかのフラッシュ操作へのコマンド行インタフェースを提供します。セクション4.5.2「mysqladmin — MySQL サーバーの管理を行うクライアント」を参照してください。

> **注記**
>
> ストアドファンクションまたはトリガー内で FLUSH ステートメントを発行することはできません。ただし、ストアドプロシージャーでは、それがストアドファンクションまたはトリガーから呼び出されないかぎり、FLUSH を使用できます。セクションD.1「ストアドプログラムの制約」を参照してください。

MySQL 5.6.11 でのみ、このステートメントを発行する前に、gtid_next を AUTOMATIC に設定する必要があります。(Bug #16062608、Bug #16715809、Bug #69045)

FLUSH TABLES 構文

FLUSH TABLES には、次に説明されているいくつかの形式があります。TABLES オプションのいずれかのバリアントが FLUSH ステートメントで使用されている場合は、それが、使用されている唯一のオプションである必要があります。FLUSH TABLES は FLUSH TABLES のシノニムです。

- FLUSH TABLES

  開かれているすべてのテーブルを閉じ、使用されているすべてのテーブルを強制的に閉じて、クエリーキャッシュをフラッシュします。FLUSH TABLES はまた、RESET QUERY CACHE ステートメントと同様に、クエリーキャッシュからすべてのクエリー結果を削除します。

  MySQL 5.6 では、アクティブな LOCK TABLES ... READ が存在する場合、FLUSH TABLES は許可されません。テーブルをフラッシュしてロックするには、代わりに FLUSH TABLES tbl_name ... WITH READ LOCK を使用します。

- FLUSH TABLES tbl_name [, tbl_name] ...

  カンマで区切られた 1 つ以上のテーブル名のリストを指定した場合、このステートメントは、サーバーが指定されたテーブルのみをフラッシュする点を除き、名前のない FLUSH TABLES と同様です。指定されたテーブルが存在しない場合、エラーは発生しません。

- FLUSH TABLES WITH READ LOCK

  開かれているすべてのテーブルを閉じ、グローバルな読み取りロックを保持しているすべてのデータベースのすべてのテーブルをロックします。これは、特定時点のスナップショットを取得できる、Veritas または ZFS などのファイルシステムがある場合にバックアップを取得するための非常に便利な方法です。このロックを解放するには、UNLOCK TABLES を使用します。

  FLUSH TABLES WITH READ LOCK は、グローバルな読み取りロックを取得しますが、テーブルロックは取得しないため、テーブルロックと暗黙的なコミットに関して LOCK TABLES および UNLOCK TABLES と同じ動作には従いません。

  - UNLOCK TABLES は、現在 LOCK TABLES でロックされているテーブルがある場合にのみ、アクティブなトランザクションをすべて暗黙的にコミットします。FLUSH TABLES WITH READ LOCK はテーブルロックを取得しないため、このステートメントに続く UNLOCK TABLES に対してコミットは発生しません。
  - トランザクションを開始すると、ユーザーが UNLOCK TABLES を実行したかのように、LOCK TABLES によって取得されたテーブルロックが解放されます。トランザクションを開始しても、FLUSH TABLES WITH READ LOCK によって取得されたグローバルな読み取りロックは解放されません。

  FLUSH TABLES WITH READ LOCK では、サーバーがログテーブルに行を挿入することは妨げられません (セクション5.2.1「一般クエリーログおよびスロークエリーログの出力先の選択」を参照してください)。

- FLUSH TABLES tbl_name [, tbl_name] ... WITH READ LOCK

  このステートメントは、指定されたテーブルをフラッシュし、それに対する読み取りロックを取得します。このステートメントは、まずテーブルに対する排他的なメタデータロックを取得するため、これらのテーブルを開いているトランザクションの完了を待機します。次に、このステートメントはテーブルキャッシュからテーブルをフラッシュし、テーブルを再度開き、(LOCK TABLES ... READ と同様に) テーブルロックを取得したあと、そのメタデータロックを排他的から共有にダウングレードします。このステートメントがロックを取得

し、そのメタデータロックをダウングレードしたら、ほかのセッションはそれらのテーブルを読み取ることができますが、変更することはできません。

このステートメントはテーブルロックを取得するため、いずれかの FLUSH ステートメントを使用するために必要な RELOAD 権限に加えて、各テーブルに対する LOCK TABLES 権限が必要です。

このステートメントは、既存のベーステーブルにのみ適用されます。名前がベーステーブルを参照している場合は、そのテーブルが使用されます。TEMPORARY テーブルを参照している場合、その名前は無視されます。名前がビューに適用される場合は、ER_WRONG_OBJECT エラーが発生します。それ以外の場合は、ER_NO_SUCH_TABLE エラーが発生します。

ロックを解放するには UNLOCK TABLES を、ロックを解放し、ほかのロックを取得するには LOCK TABLES を、またはロックを解放し、新しいトランザクションを開始するには START TRANSACTION を使用します。

FLUSH のこのバリアントを使用すると、テーブルのフラッシュとロックを 1 つの操作で実行できます。これにより、アクティブな LOCK TABLES ... READ が存在する場合は FLUSH TABLES が許可されないという MySQL 5.6 での制限の回避方法が提供されます。

このステートメントは暗黙的な UNLOCK TABLES を実行しないため、このステートメントをアクティブな LOCK TABLES が存在する間に使用したり、取得されたロックをまず解放することなくふたたび使用したりすると、エラーが発生します。

フラッシュされたテーブルが HANDLER で開かれた場合、そのハンドラは暗黙的にフラッシュされ、その位置を失います。

- FLUSH TABLES tbl_name [, tbl_name] ... FOR EXPORT

  この FLUSH TABLES バリアントは、InnoDB テーブルに適用されます。これは、MySQL 5.6.6 の時点で使用できます。このステートメントは、サーバー稼働中にバイナリテーブルのコピーができるように、名前付きテーブルへの変更をディスクにフラッシュします。

  このステートメントは、次のように機能します。

  1. 指定されたテーブルに対する共有メタデータロックを取得します。これらのテーブルを変更したか、またはそれに対するテーブルロックを保持するアクティブなトランザクションがほかのセッションに存在するかぎり、このステートメントはブロックされます。ロックが取得されると、このステートメントは、これらのテーブルを更新しようとするトランザクションをブロックしながら、読み取り専用操作は続行できるようにします。

  2. これらのテーブルのすべてのストレージエンジンが FOR EXPORT をサポートしているかどうかをチェックします。サポートしていないものがある場合は、ER_ILLEGAL_HA エラーが発生し、このステートメントは失敗します。

  3. このステートメントは、各テーブルのストレージエンジンに、そのテーブルをエクスポートのために準備するよう通知します。そのストレージエンジンは、保留中の変更がすべてディスクに書き込まれるようにする必要があります。

  4. このステートメントは、FOR EXPORT ステートメントの完了時に以前に取得されたメタデータロックが解放されないように、そのセッションをテーブルロックモードにします。

  FLUSH TABLES ... FOR EXPORT ステートメントには、各テーブルに対する SELECT 権限が必要です。このステートメントはテーブルロックを取得するため、いずれかの FLUSH ステートメントを使用するために必要な RELOAD 権限に加えて、各テーブルに対する LOCK TABLES 権限も必要です。

  このステートメントは、既存のベーステーブルにのみ適用されます。名前がベーステーブルを参照している場合は、そのテーブルが使用されます。TEMPORARY テーブルを参照している場合、その名前は無視されます。名前がビューに適用される場合は、ER_WRONG_OBJECT エラーが発生します。それ以外の場合は、ER_NO_SUCH_TABLE エラーが発生します。

  InnoDB は、独自の .ibd ファイルを持つテーブル (つまり、innodb_file_per_table 設定が有効になった状態で作成されたテーブル) に対する FOR EXPORT をサポートしています。InnoDB は、FOR EXPORT ステートメントから通知されると、すべての変更を確実にディスクにフラッシュします。.ibd ファイルはトランザクション一貫性があり、サーバーの実行中にコピーできるため、これにより、FOR EXPORT ステートメントが有効に

なっている間にテーブルの内容のバイナリコピーを作成できます。FOR EXPORT は、InnoDB システムテーブルスペースファイルや、いずれかの FULLTEXT インデックスを含む InnoDB テーブルには適用されません。

FLUSH TABLES ...FOR EXPORT は、MySQL 5.6.17 より前のパーティション化された InnoDB テーブルでは機能しませんが、MySQL 5.6.17 以降ではこのようなテーブルに対してもサポートされます。(Bug #16943907)。

FOR EXPORT から通知されると、InnoDB は、通常はメモリー内か、またはテーブルスペースファイルの外部にある個別のディスクバッファーに保持される特定の種類のデータをディスクに書き込みます。InnoDB はまた、テーブルごとに、そのテーブルと同じデータベースディレクトリ内に table_name.cfg という名前のファイルも生成します。.cfg ファイルには、あとでテーブルスペースファイルを同じサーバーまたは別のサーバーに再インポートするために必要なメタデータが含まれています。

FOR EXPORT ステートメントが完了すると、InnoDB によって、すべてのダーティーページがテーブルデータファイルにフラッシュされています。変更バッファーのエントリはすべて、フラッシュの前にマージされます。この時点で、テーブルはロックされ、静止します。これらのテーブルはディスク上でトランザクション的に一貫性のある状態にあるため、.ibd テーブルスペースファイルを対応する .cfg ファイルとともにコピーすることによって、これらのテーブルの整合性のあるスナップショットを取得できます。

コピーされたテーブルデータを MySQL インスタンスに再インポートする手順については、セクション 14.5.5「テーブルスペースの別のサーバーへのコピー (トランスポータブルテーブルスペース)」を参照してください。

テーブルの処理を完了したあと、ロックを解放するには UNLOCK TABLES を、ロックを解放し、ほかのロックを取得するには LOCK TABLES を、またはロックを解放し、新しいトランザクションを開始するには START TRANSACTION を使用します。

セッション内で次のいずれかのステートメントが有効になっている間は、FLUSH TABLES ... FOR EXPORT を使用しようとするとエラーが生成されます。

```
FLUSH TABLES ... WITH READ LOCK
FLUSH TABLES ... FOR EXPORT
LOCK TABLES ... READ
LOCK TABLES ... WRITE
```

セッション内で FLUSH TABLES ... FOR EXPORT が有効になっている間は、次のいずれかのステートメントを使用しようとするとエラーが生成されます。

```
FLUSH TABLES WITH READ LOCK
FLUSH TABLES ... WITH READ LOCK
FLUSH TABLES ... FOR EXPORT
```

13.7.6.4 KILL 構文

```
KILL [CONNECTION | QUERY] processlist_id
```

mysqld への各接続は、個別のスレッドで実行されます。スレッドは、KILL processlist_id ステートメントで強制終了できます。

スレッドのプロセスリスト識別子は、INFORMATION_SCHEMA.PROCESSLIST テーブルの ID カラム、SHOW PROCESSLIST の出力の Id カラム、およびパフォーマンススキーマの threads テーブルの PROCESSLIST_ID カラムから特定できます。現在のスレッドの値は、CONNECTION_ID() 関数によって返されます。

KILL では、オプションの CONNECTION または QUERY 修飾子が許可されます。

- KILL CONNECTION は、修飾子のない KILL と同じです。これは、指定された processlist_id に関連付けられた接続を終了します。
- KILL QUERY は、接続が現在実行しているステートメントを終了しますが、その接続自体はそのままの状態で残します。

PROCESS 権限がある場合は、すべてのスレッドを表示できます。SUPER 権限がある場合は、すべてのスレッドとステートメントを強制終了できます。そうでない場合は、ユーザー独自のスレッドとステートメントのみを表示および強制終了できます。

mysqladmin processlist および mysqladmin kill コマンドを使用して、スレッドを検査および強制終了することもできます。

> **注記**
>
> 組み込みサーバーはホストアプリケーションのスレッドの内部でしか実行されないため、組み込み MySQL Server ライブラリでは KILL を使用できません。独自の接続スレッドは作成されません。

KILL を使用すると、そのスレッドのスレッド固有の強制終了フラグが設定されます。強制終了フラグは次の一定の間隔でしかチェックされないため、ほとんどの場合、スレッドが終了するまでにある程度時間がかかることがあります。

- SELECT 操作中、ORDER BY および GROUP BY ループでは、このフラグは行ブロックの読み取りのあとにチェックされます。強制終了フラグが設定されている場合、このステートメントは中止されます。
- ALTER TABLE 操作中、強制終了フラグは、元のテーブルから各行ブロックが読み取られる前にチェックされます。強制終了フラグが設定されていた場合、このステートメントは中止され、一時テーブルが削除されます。
- UPDATE または DELETE 操作中、強制終了フラグは、ブロックが読み取られるたび、および行が更新または削除されるたびにチェックされます。強制終了フラグが設定されている場合、このステートメントは中止されます。トランザクションを使用していない場合は、変更がロールバックされません。
- GET_LOCK() は中止され、NULL を返します。
- INSERT DELAYED スレッドは、メモリー内にあるすべての行をすばやくフラッシュ (挿入) してから終了します。
- このスレッドがテーブルロックハンドラ内にある場合 (状態: Locked)、そのテーブルロックはすばやく中止されます。
- このスレッドが書き込みコールでディスクの空き容量を待機している場合、その書き込みは「ディスク領域不足」というエラーメッセージで中止されます。

> **警告**
>
> MyISAM テーブルに対する REPAIR TABLE または OPTIMIZE TABLE 操作を強制終了すると、テーブルが破損して使用できなくなります。このようなテーブルに対する読み取りまたは書き込みはすべて、そのテーブルをふたたび最適化または修復するまで失敗します (割り込みはなし)。

13.7.6.5 LOAD INDEX INTO CACHE 構文

```
LOAD INDEX INTO CACHE
  tbl_index_list [, tbl_index_list] ...

tbl_index_list:
  tbl_name
    [PARTITION (partition_list | ALL)]
    [[INDEX|KEY] (index_name[, index_name] ...)]
    [IGNORE LEAVES]

partition_list:
    partition_name[, partition_name][, ...]
```

LOAD INDEX INTO CACHE ステートメントは、明示的な CACHE INDEX ステートメントによって割り当てられたキーキャッシュ、またはそれ以外の場合はデフォルトのキーキャッシュにテーブルインデックスをプリロードします。

LOAD INDEX INTO CACHE は、MyISAM テーブルにのみ使用されます。MySQL 5.6 では、パーティション化された MyISAM テーブルに対してもサポートされます。さらに、パーティション化されたテーブル上のインデックスを 1 つ、複数、またはすべてのパーティションに対してプリロードできます。

IGNORE LEAVES 修飾子によって、インデックスの非リーフノードのブロックのみがプリロードされます。

IGNORE LEAVES は、パーティション化された MyISAM テーブルに対してもサポートされます。

次のステートメントは、テーブル t1 と t2 のインデックスのノード (インデックスブロック) をプリロードします。

```
mysql> LOAD INDEX INTO CACHE t1, t2 IGNORE LEAVES;
+---------+--------------+----------+----------+
```

```
| Table   | Op           | Msg_type | Msg_text |
+---------+--------------+----------+----------+
| test.t1 | preload_keys | status   | OK       |
| test.t2 | preload_keys | status   | OK       |
+---------+--------------+----------+----------+
```

このステートメントは、t1 からすべてのインデックスブロックをプリロードします。t2 からは、非リーフノードのブロックのみをプリロードします。

LOAD INDEX INTO CACHE の構文では、テーブルの特定のインデックスのみをプリロードするように指定できます。現在の実装では、テーブルのすべてのインデックスをキャッシュにプリロードするため、テーブル名以外のものを指定する理由は何もありません。

MySQL 5.6.11 でのみ、このステートメントを発行する前に、gtid_next を AUTOMATIC に設定する必要があります。(Bug #16062608、Bug #16715809、Bug #69045)

MySQL 5.6 では、パーティション化された MyISAM テーブルの特定のパーティション上のインデックスをプリロードできます。たとえば、次の 2 つのステートメントでは、最初が、パーティション化されたテーブル pt のパーティション p0 のインデックスをプリロードするのに対して、2 番目は同じテーブルのパーティション p1 と p3 のインデックスをプリロードします。

```
LOAD INDEX INTO CACHE pt PARTITION (p0);
LOAD INDEX INTO CACHE pt PARTITION (p1, p3);
```

テーブル pt 内のすべてのパーティションのインデックスをプリロードするには、次の 2 つのステートメントのどちらでも使用できます。

```
LOAD INDEX INTO CACHE pt PARTITION (ALL);

LOAD INDEX INTO CACHE pt;
```

今示した 2 つのステートメントは同等であり、このうちのどちらを発行しても効果はまったく同じです。つまり、パーティション化されたテーブルのすべてのパーティションのインデックスをプリロードする場合、PARTITION (ALL) 句はオプションです。

複数のパーティションのインデックスをプリロードする場合、それらのパーティションが連続している必要はなく、それらの名前を特定の順序でリストする必要もありません。

LOAD INDEX INTO CACHE ... IGNORE LEAVES は、テーブル内のすべてのインデックスのブロックサイズが同じでないかぎり失敗します。テーブルのインデックスブロックサイズは、myisamchk -dv を使用し、Blocksize カラムをチェックすることによって特定できます。

13.7.6.6 RESET 構文

```
RESET reset_option [, reset_option] ...
```

RESET ステートメントは、さまざまなサーバー操作の状態をクリアするために使用されます。RESET を実行するには、RELOAD 権限が必要です。

RESET は、FLUSH ステートメントのより強力なバージョンとして機能します。セクション13.7.6.3「FLUSH 構文」を参照してください。

RESET ステートメントは暗黙的なコミットを発生させます。セクション13.3.3「暗黙的なコミットを発生させるステートメント」を参照してください。

MySQL 5.6.11 でのみ、このステートメントを発行する前に、gtid_next を AUTOMATIC に設定する必要があります。(Bug #16062608、Bug #16715809、Bug #69045)

reset_option には、次のいずれかを指定できます。

- MASTER

  インデックスファイルにリストされているすべてのバイナリログを削除し、バイナリログインデックスファイルを空にリセットして、新しいバイナリログファイルを作成します。

- QUERY CACHE

  クエリーキャッシュからすべてのクエリー結果を削除します。

- SLAVE

スレーブに、マスターバイナリログ内のそのレプリケーション位置を忘れさせます。また、既存のリレーログファイルをすべて削除し、新しいリレーログファイルを開始することによってリレーログもリセットします。

13.8 MySQL ユーティリティーステートメント

13.8.1 DESCRIBE 構文

DESCRIBE ステートメントと EXPLAIN ステートメントはシノニムであり、テーブル構造またはクエリー実行計画に関する情報を取得するために使用されます。詳細は、セクション13.7.5.6「SHOW COLUMNS 構文」およびセクション13.8.2「EXPLAIN 構文」を参照してください。

13.8.2 EXPLAIN 構文

```
{EXPLAIN | DESCRIBE | DESC}
    tbl_name [col_name | wild]

{EXPLAIN | DESCRIBE | DESC}
    [explain_type]
    explainable_stmt

explain_type: {
    EXTENDED
  | PARTITIONS
  | FORMAT = format_name
}

format_name: {
    TRADITIONAL
  | JSON
}

explainable_stmt: {
    SELECT statement
  | DELETE statement
  | INSERT statement
  | REPLACE statement
  | UPDATE statement
}
```

DESCRIBE ステートメントと EXPLAIN ステートメントはシノニムです。実際には、DESCRIBE キーワードがテーブル構造に関する情報を取得するためにより頻繁に使用されるのに対して、EXPLAIN は、クエリー実行計画 (つまり、MySQL がクエリーをどのように実行するかの説明) を取得するために使用されます。次の説明では、DESCRIBE および EXPLAIN キーワードをそのような用途に従って使用しますが、MySQL パーサーはこれらを完全にシノニムとして処理します。

テーブル構造に関する情報の取得

DESCRIBE は、テーブル内のカラムに関する情報を提供します。

```
mysql> DESCRIBE City;
+------------+----------+------+-----+---------+----------------+
| Field      | Type     | Null | Key | Default | Extra          |
+------------+----------+------+-----+---------+----------------+
| Id         | int(11)  | NO   | PRI | NULL    | auto_increment |
| Name       | char(35) | NO   |     |         |                |
| Country    | char(3)  | NO   | UNI |         |                |
| District   | char(20) | YES  | MUL |         |                |
| Population | int(11)  | NO   |     | 0       |                |
+------------+----------+------+-----+---------+----------------+
```

DESCRIBE は SHOW COLUMNS のショートカットです。これらのステートメントはまた、ビューに関する情報も表示します。SHOW COLUMNS の説明では、出力カラムに関してより多くの情報が提供されます。セクション13.7.5.6「SHOW COLUMNS 構文」を参照してください。

デフォルトでは、DESCRIBE は、そのテーブル内のすべてのカラムに関する情報を表示します。col_name (指定されている場合) は、そのテーブル内のカラムの名前です。この場合、このステートメントは、指定されたカラムの情報のみを表示します。wild (指定されている場合) は、パターン文字列です。これには、SQL の「%」と「_」のワイルドカード文字を含めることができます。この場合、このステートメントは、その文字列に一致する名前を持つカラムの出力のみを表示します。スペースやその他の特殊文字が含まれていないかぎり、この文字列を引用符で囲む必要はありません。

DESCRIBE ステートメントは、Oracle との互換性のために提供されています。

また、SHOW CREATE TABLE、SHOW TABLE STATUS、および SHOW INDEX ステートメントでは、テーブルに関する情報も提供されます。セクション13.7.5「SHOW 構文」を参照してください。

実行計画に関する情報の取得

EXPLAIN ステートメントは、MySQL がステートメントをどのように実行するかに関する情報を提供します。

- MySQL 5.6.3 現在、EXPLAIN に使用できる説明可能なステートメントは、SELECT、DELETE、INSERT、REPLACE、および UPDATE です。MySQL 5.6.3 より前では、SELECT が唯一の説明可能なステートメントです。
- 説明可能なステートメントで EXPLAIN を使用すると、MySQL は、オプティマイザからのステートメント実行プランに関する情報を表示します。つまり、MySQL はテーブルがどのように、どんな順番で結合されているかに関する情報を含む、ステートメントを処理する方法を説明します。EXPLAIN を使用して、実行プラン情報を取得することについては、セクション8.8.2「EXPLAIN 出力フォーマット」を参照してください。
- EXPLAIN EXTENDED を使用して、追加の実行プラン情報を取得できます。セクション8.8.3「EXPLAIN EXTENDED 出力フォーマット」を参照してください。
- EXPLAIN PARTITIONS は、パーティション化されたテーブルを含むクエリーの調査に役立ちます。セクション19.3.5「パーティションに関する情報を取得する」を参照してください。
- MySQL 5.6.5 の時点では、FORMAT オプションを使用して出力形式を選択できます。TRADITIONAL は表形式で出力を表示します。FORMAT オプションが存在しない場合、これはデフォルトです。JSON フォーマットは JSON フォーマットで情報を表示します。FORMAT = JSON を使用すると、出力には拡張されたパーティション情報が含まれます。

EXPLAIN によって、インデックスを使用して行を見つけることで、ステートメントが高速に実行されるように、テーブルにインデックスを追加するべき場所がわかります。また、EXPLAIN を使用して、オプティマイザがテーブルを最適な順序で結合しているかどうかを確認することもできます。SELECT ステートメントでテーブルが指定されている順序に対応する結合順序を使用するように、オプティマイザにヒントを提供するには、ステートメントを SELECT だけでなく、SELECT STRAIGHT_JOIN で始めます。(セクション13.2.9「SELECT 構文」を参照してください。)

インデックスが使われるはずであると思うタイミングでそれらが使われていない問題がある場合、ANALYZE TABLE を実行して、オプティマイザが行う選択に影響する可能性があるキーのカーディナリティーなどのテーブル統計を更新します。セクション13.7.2.1「ANALYZE TABLE 構文」を参照してください。

13.8.3 HELP 構文

```
HELP 'search_string'
```

HELP は、MySQL リファレンスマニュアルのオンライン情報を返します。これが正しく動作するには、mysql データベース内のヘルプテーブルがヘルプトピック情報で初期化されている必要があります (セクション5.1.10「サーバー側のヘルプ」を参照してください)。

HELP ステートメントは、ヘルプテーブル内の指定された検索文字列を検索し、その検索の結果を表示します。検索文字列は大文字と小文字が区別されません。

検索文字列にはワイルドカード文字 「%」 および 「_」 を含めることができます。これらは LIKE 演算子で実行されるパターンマッチング演算と同じ意味を持ちます。たとえば、HELP 'rep%' は rep で始まるトピックのリストを返します。

HELP ステートメントは、次のいくつかの種類の検索文字列を理解します。

- もっとも一般的なレベルでは、トップレベルのヘルプカテゴリのリストを取得するには contents を使用します。

  ```
  HELP 'contents'
  ```

- Data Types などの、特定のヘルプカテゴリ内のトピックのリストを取得するには、そのカテゴリ名を使用します。

  ```
  HELP 'data types'
  ```

- ASCII() 関数や CREATE TABLE ステートメントなどの、特定のヘルプトピックに関するヘルプを表示するには、関連する 1 つまたは複数のキーワードを使用します。

```
HELP 'ascii'
HELP 'create table'
```

つまり、検索文字列はカテゴリ、多数のトピック、または 1 つのトピックに一致します。特定の検索文字列が項目のリストか、または 1 つのヘルプトピックのヘルプ情報のどちらを返すかが前もってわかるとはかぎりません。ただし、結果セット内の行数やカラム数を検査することによって、HELP がどのような種類の応答を返したかがわかります。

次の説明は、結果セットの可能性のある形式を示しています。ステートメントの例の出力は、mysql クライアントの使用時に表示されるなじみのある「表」または「垂直」形式を使用して示されています。

> **注記**
>
> mysql 自体は、HELP の結果セットを別の方法で再フォーマットします。

- 空の結果セット

  検索文字列に一致するものが見つかりませんでした。

- 3 つのカラムを含む単一行が含まれた結果セット

  これは、検索文字列が 1 つのヘルプトピックに一致したことを示します。この結果には 3 つのカラムが含まれています。

  - name: トピック名。
  - description: トピックの説明的なヘルプテキスト。
  - example: 使用例または例。このカラムはブランクである可能性があります。

  例: HELP 'replace'

  生成される結果:

```
name: REPLACE
description: Syntax:
REPLACE(str,from_str,to_str)

Returns the string str with all occurrences of the string from_str
replaced by the string to_str. REPLACE() performs a case-sensitive
match when searching for from_str.
example: mysql> SELECT REPLACE('www.mysql.com', 'w', 'Ww');
       -> 'WwWwWw.mysql.com'
```

- 2 つのカラムを含む複数の行が含まれた結果セット

  これは、検索文字列が多数のヘルプトピックに一致したことを示します。この結果セットは、ヘルプトピック名を示します。

  - name: ヘルプトピック名。
  - is_it_category: この名前がヘルプカテゴリを表す場合は Y、それ以外の場合は N。それ以外の場合、name 値は、HELP ステートメントへの引数として指定されると、指定された項目の説明が含まれた単一行の結果セットを生成するはずです。

  例: HELP 'status'

  生成される結果:

```
+----------------------+---------------+
| name                 | is_it_category |
+----------------------+---------------+
| SHOW                 | N             |
| SHOW ENGINE          | N             |
| SHOW MASTER STATUS   | N             |
| SHOW PROCEDURE STATUS | N            |
| SHOW SLAVE STATUS    | N             |
| SHOW STATUS          | N             |
| SHOW TABLE STATUS    | N             |
+----------------------+---------------+
```

- 3 つのカラムを含む複数の行が含まれた結果セット

これは、検索文字列がカテゴリに一致したことを示します。この結果セットには、カテゴリエントリが含まれています。

- source_category_name: ヘルプカテゴリ名。
- name: カテゴリまたはトピック名
- is_it_category: この名前がヘルプカテゴリを表す場合は Y、それ以外の場合は N。それ以外の場合、name 値は、HELP ステートメントへの引数として指定されると、指定された項目の説明が含まれた単一行の結果セットを生成するはずです。

例: HELP 'functions'

生成される結果:

```
+--------------------+-------------------------+----------------+
| source_category_name | name                  | is_it_category |
+--------------------+-------------------------+----------------+
| Functions          | CREATE FUNCTION         | N              |
| Functions          | DROP FUNCTION           | N              |
| Functions          | Bit Functions           | Y              |
| Functions          | Comparison operators    | Y              |
| Functions          | Control flow functions  | Y              |
| Functions          | Date and Time Functions | Y              |
| Functions          | Encryption Functions    | Y              |
| Functions          | Information Functions   | Y              |
| Functions          | Logical operators       | Y              |
| Functions          | Miscellaneous Functions | Y              |
| Functions          | Numeric Functions       | Y              |
| Functions          | String Functions        | Y              |
+--------------------+-------------------------+----------------+
```

13.8.4 USE 構文

```
USE db_name
```

USE db_name ステートメントは、以降のステートメントのデフォルトの (現在の) データベースとして db_name データベースを使用するよう MySQL に指示します。このデータベースは、セッションが終了するか、または別の USE ステートメントが発行されるまでデフォルトのままになります。

```
USE db1;
SELECT COUNT(*) FROM mytable;   # selects from db1.mytable
USE db2;
SELECT COUNT(*) FROM mytable;   # selects from db2.mytable
```

USE ステートメントを使用して特定のデータベースをデフォルトにしても、ユーザーがほかのデータベース内のテーブルにアクセスすることは除外されません。次の例では、db1 データベースの author テーブルと、db2 データベースの editor テーブルにアクセスします。

```
USE db1;
SELECT author_name,editor_name FROM author,db2.editor
  WHERE author.editor_id = db2.editor.editor_id;
```

第 14 章 InnoDB ストレージエンジン

目次

14.1 InnoDB 入門 1495
14.1.1 デフォルトの MySQL ストレージエンジンとしての InnoDB 1496
14.1.2 InnoDB の可用性チェック 1499
14.1.3 InnoDB の無効化 1499
14.2 InnoDB の概念とアーキテクチャー 1500
14.2.1 MySQL および ACID モデル 1500
14.2.2 InnoDB のトランザクションモデルおよびロック 1502
14.2.3 InnoDB のロックモード 1502
14.2.4 一貫性非ロック読み取り 1504
14.2.5 ロック読み取り (SELECT ... FOR UPDATE および SELECT ... LOCK IN SHARE MODE) 1506
14.2.6 InnoDB のレコード、ギャップ、およびネクストキーロック 1507
14.2.7 ネクストキーロックによるファントム問題の回避 1508
14.2.8 InnoDB のさまざまな SQL ステートメントで設定されたロック 1509
14.2.9 暗黙的なトランザクションコミットとロールバック 1512
14.2.10 デッドロックの検出とロールバック 1512
14.2.11 デッドロックの対処方法 1512
14.2.12 InnoDB マルチバージョン 1513
14.2.13 InnoDB テーブルおよびインデックスの構造 1514
14.3 InnoDB の構成 1522
14.3.1 読み取り専用操作用の InnoDB の構成 1526
14.4 InnoDB の管理 1527
14.5 InnoDB テーブルスペース管理 1527
14.5.1 InnoDB テーブルスペースの作成 1527
14.5.2 InnoDB File-Per-Table モード 1528
14.5.3 File-Per-Table モードの有効化および無効化 1530
14.5.4 テーブルスペースの位置の指定 1531
14.5.5 テーブルスペースの別のサーバーへのコピー (トランスポータブルテーブルスペース) 1532
14.5.6 個別のテーブルスペースへの InnoDB Undo ログの格納 1535
14.5.7 InnoDB ログファイルの数またはサイズの変更、および InnoDB テーブルスペースのサイズの変更 1536
14.5.8 共有テーブルスペースでの RAW ディスクパーティションの使用 1538
14.6 InnoDB テーブルの管理 1539
14.6.1 InnoDB テーブルの作成 1539
14.6.2 別のマシンへの InnoDB テーブルの移動またはコピー 1540
14.6.3 トランザクションを使用した DML 操作のグループ化 1541
14.6.4 MyISAM から InnoDB へのテーブルの変換 1542
14.6.5 InnoDB での AUTO_INCREMENT 処理 1546
14.6.6 InnoDB と FOREIGN KEY 制約 1551
14.6.7 InnoDB テーブル上の制限 1552
14.7 InnoDB 圧縮テーブル 1555
14.7.1 テーブル圧縮の概要 1555
14.7.2 テーブル圧縮の有効化 1555
14.7.3 InnoDB テーブルの圧縮の調整 1556
14.7.4 実行時の圧縮のモニタリング 1560
14.7.5 InnoDB テーブルでの圧縮の動作 1560
14.7.6 OLTP ワークロードの圧縮 1563
14.7.7 SQL 圧縮構文の警告とエラー 1564
14.8 InnoDB のファイル形式管理 1566
14.8.1 ファイル形式の有効化 1566
14.8.2 ファイル形式の互換性の確認 1566
14.8.3 使用されているファイル形式の識別 1570
14.8.4 ファイル形式のダウングレード 1571
14.8.5 将来の InnoDB ファイル形式 1571
14.9 InnoDB の行ストレージと行フォーマット 1571
14.9.1 InnoDB 行ストレージの概要 1571
14.9.2 テーブルの行フォーマットの指定 1571
14.9.3 DYNAMIC および COMPRESSED 行フォーマット 1572
14.9.4 COMPACT および REDUNDANT 行フォーマット 1572

14.10 InnoDB のディスク I/O とファイル領域管理 ... 1572
14.10.1 InnoDB ディスク I/O ... 1573
14.10.2 ファイル領域管理 ... 1573
14.10.3 InnoDB チェックポイント ... 1574
14.10.4 テーブルのデフラグ ... 1575
14.10.5 TRUNCATE TABLE によるディスク領域の再利用 ... 1575
14.11 InnoDB とオンライン DDL ... 1575
14.11.1 オンライン DDL の概要 ... 1576
14.11.2 オンライン DDL でのパフォーマンスと並列性に関する考慮事項 ... 1581
14.11.3 オンライン DDL の SQL 構文 ... 1583
14.11.4 DDL ステートメントの結合または分離 ... 1583
14.11.5 オンライン DDL の例 ... 1584
14.11.6 オンライン DDL の実装の詳細 ... 1602
14.11.7 オンライン DDL でのクラッシュリカバリの動作のしくみ ... 1604
14.11.8 パーティション化された InnoDB テーブルに対するオンライン DDL ... 1604
14.11.9 オンライン DDL の制限 ... 1605
14.12 InnoDB の起動オプションおよびシステム変数 ... 1605
14.13 InnoDB のパフォーマンス ... 1663
14.13.1 InnoDB バッファープールの構成 ... 1663
14.13.2 InnoDB 相互排他ロックおよび読み取り/書き込みロックの実装 ... 1669
14.13.3 InnoDB のためのメモリーアロケータの構成 ... 1669
14.13.4 InnoDB 変更バッファリングの構成 ... 1670
14.13.5 InnoDB のスレッド並列性の構成 ... 1671
14.13.6 InnoDB バックグラウンド I/O スレッドの数の構成 ... 1672
14.13.7 グループコミット ... 1672
14.13.8 InnoDB マススレッドの I/O レートの構成 ... 1672
14.13.9 InnoDB スピンループでの PAUSE 命令の使用 ... 1673
14.13.10 スピンロックのポーリングの構成 ... 1673
14.13.11 InnoDB の MySQL パフォーマンススキーマとの統合 ... 1674
14.13.12 複数のロールバックセグメントによるスケーラビリティーの向上 ... 1674
14.13.13 InnoDB のパージスケジューリングの構成 ... 1675
14.13.14 InnoDB の読み取り専用トランザクションの最適化 ... 1675
14.13.15 チェックサムの高速化のための CRC32 チェックサムアルゴリズムの使用 ... 1676
14.13.16 オプティマイザ統計 ... 1676
14.13.17 InnoDB テーブルに対する ANALYZE TABLE の複雑さの推定 ... 1683
14.14 InnoDB INFORMATION_SCHEMA テーブル ... 1684
14.14.1 圧縮に関する InnoDB INFORMATION_SCHEMA テーブル ... 1685
14.14.2 InnoDB INFORMATION_SCHEMA トランザクションおよびロックテーブル ... 1686
14.14.3 InnoDB INFORMATION_SCHEMA システムテーブル ... 1691
14.14.4 InnoDB INFORMATION_SCHEMA FULLTEXT インデックステーブル ... 1696
14.14.5 InnoDB INFORMATION_SCHEMA バッファープールテーブル ... 1698
14.14.6 InnoDB INFORMATION_SCHEMA メトリックテーブル ... 1702
14.15 InnoDB モニター ... 1709
14.15.1 InnoDB モニターのタイプ ... 1710
14.15.2 InnoDB モニターの有効化 ... 1710
14.15.3 InnoDB 標準モニターおよびロックモニターの出力 ... 1712
14.15.4 InnoDB テーブルスペースモニターの出力 ... 1716
14.15.5 InnoDB テーブルモニターの出力 ... 1718
14.16 InnoDB のバックアップとリカバリ ... 1720
14.16.1 InnoDB のリカバリプロセス ... 1722
14.17 InnoDB と MySQL レプリケーション ... 1723
14.18 InnoDB と memcached の統合 ... 1724
14.18.1 InnoDB と memcached の組み合わせの利点 ... 1725
14.18.2 InnoDB および memcached の統合のアーキテクチャー ... 1726
14.18.3 InnoDB Memcached プラグインの概要 ... 1729
14.18.4 InnoDB memcached プラグインのセキュリティーに関する考慮事項 ... 1731
14.18.5 InnoDB memcached インタフェース用のアプリケーションの作成 ... 1733
14.18.6 レプリケーションでの InnoDB memcached プラグインの使用 ... 1742
14.18.7 InnoDB memcached プラグインの内部構造 ... 1745
14.18.8 InnoDB memcached プラグインのトラブルシューティング ... 1749
14.19 InnoDB のトラブルシューティング ... 1751
14.19.1 InnoDB の I/O に関する問題のトラブルシューティング ... 1751
14.19.2 InnoDB のリカバリの強制的な実行 ... 1752
14.19.3 InnoDB データディクショナリの操作のトラブルシューティング ... 1753

14.19.4 InnoDB のエラー処理 1755
14.19.5 InnoDB のエラーコード 1755
14.19.6 オペレーティングシステムのエラーコード 1756

14.1 InnoDB 入門

InnoDB は、高い信頼性と高いパフォーマンスとのバランスをとる汎用のストレージエンジンです。MySQL 5.5 の時点では、これがデフォルトの MySQL ストレージエンジンです。MySQL 5.6 では、ENGINE= 句を指定せずに CREATE TABLE ステートメントを発行すると、InnoDB テーブルが作成されます。

InnoDB の主要な利点

InnoDB テーブルの主要な利点は、次のとおりです。

- その DML 操作は、トランザクションにユーザーデータを保護するためのコミット、ロールバック、およびクラッシュリカバリ機能が備わっている ACID モデルに従っています。
- 行レベルのロックと Oracle スタイルの一貫性読み取りを使用すると、複数ユーザーの並列性およびパフォーマンスが向上します。
- InnoDB テーブルでは、主キーに基づいてクエリーが最適化されるように、ディスク上のデータが整列されます。
- データの整合性を保つために、InnoDB では FOREIGN KEY 制約もサポートされています。挿入、更新、および削除によってさまざまなテーブル間で不整合が発生しないかを確認するために、これらの操作がすべてチェックされます。
- 同じステートメント内でも、InnoDB のテーブルと別の MySQL ストレージエンジンのテーブルを混在させることができます。たとえば、結合操作を使用すると、単一のクエリーで InnoDB テーブルと MEMORY テーブルのデータを結合できます。
- InnoDB は、大きなデータボリュームを処理する際に、高い CPU の効率性と最大のパフォーマンスが実現されるように設計されています。

表 14.1 InnoDB ストレージエンジンの機能

機能	Support
B ツリーインデックス	はい
MVCC	はい
T ツリーインデックス	いいえ
インデックスキャッシュ	はい
クエリーキャッシュのサポート	はい
クラスタデータベースのサポート	いいえ
クラスタ化されたインデックス	はい
ストレージの制限	64T バイト
データキャッシュ	はい
データディクショナリ向け更新統計	はい
トランザクション	はい
ハッシュインデックス	いいえ (InnoDB は、アダプティブハッシュインデックス機能に対して、内部的にハッシュインデックスを利用します。)
バックアップ/ポイントインタイムリカバリ (ストレージエンジン内ではなくサーバー内で実装されています。)	はい
レプリケーションのサポート (ストレージエンジン内ではなくサーバー内で実装されています。)	はい
ロック粒度	行
全文検索インデックス	はい (InnoDB の FULLTEXT インデックスサポートは MySQL 5.6.4 以降で使用できます。)

機能	Support
圧縮データ	はい (圧縮された InnoDB テーブルは InnoDB Barracuda ファイルフォーマットを必要とします。)
地理空間インデックスのサポート	はい (InnoDB の地理空間インデックスサポートは MySQL 5.7.5 以降で使用できます。)
地理空間データ型のサポート	はい
外部キーのサポート	はい
暗号化データ (ストレージエンジン内ではなくサーバー内で (暗号化関数を使って) 実装されています。)	はい

InnoDB ストレージエンジンには、データとインデックスをメインメモリーにキャッシュするための独自のバッファープールが保持されています。デフォルトでは、innodb_file_per_table 設定が有効になっているため、新しい各 InnoDB テーブルとそれに関連付けられたインデックスが個別のファイルに格納されます。innodb_file_per_table オプションを無効にすると、InnoDB ではそのテーブルとインデックスがすべて単一のシステムテーブルスペースに格納されます。システムテーブルスペースは、複数のファイル (または生のディスクパーティション) で構成されている場合があります。InnoDB テーブルは、ファイルサイズが 2G バイトに制限されているオペレーティングシステム上でも、大量のデータを処理できます。

InnoDB の機能と MySQL で提供されているその他のストレージエンジンを比較する方法については、第15章「代替ストレージエンジン」の「ストレージエンジンの機能」表を参照してください。

InnoDB の拡張機能と新機能

MySQL 5.6 での InnoDB の拡張機能と新機能については、次を参照してください。

- InnoDB の拡張機能については、セクション1.4「MySQL 5.6 の新機能」に一覧表示されています。ここでは、MySQL 5.6 で追加された機能の概要が示されています。
- リリースノート。ここでは、各バージョンでの変更に関する情報が提供されています。

追加のリソース

- InnoDB 関連の用語および定義については、MySQL 用語集を参照してください。
- InnoDB ストレージエンジン専用のフォーラムには、MySQL Forums::InnoDB からアクセスできます。
- InnoDB は、MySQL と同じ GNU GPL ライセンスバージョン 2 (1991 年 6 月) によって発行されています。MySQL ライセンスの詳細は、http://www.mysql.com/company/legal/licensing/を参照してください。

14.1.1 デフォルトの MySQL ストレージエンジンとしての InnoDB

MySQL は、使いやすく、高いパフォーマンスと拡張性を実現するという評価を勝ち得ています。MySQL 5.5 よりも前では、MyISAM がデフォルトのストレージエンジンでした。我々の経験上、ほとんどのユーザーはデフォルト設定を変更しませんでした。MySQL 5.5 以上では、InnoDB がデフォルトのストレージエンジンです。やはりほとんどのユーザーがデフォルト設定を変更しないと予想されます。ただし、InnoDB を使用すれば、デフォルト設定でもユーザーが RDBMS から期待する利点 (ACID トランザクション、参照整合性、およびクラッシュリカバリ) が得られます。InnoDB テーブルを使用して MySQL ユーザー、DBA、または開発者としての生活を改善する方法を探ってみましょう。

ストレージエンジンの使用傾向

MySQL の成長期の 1 年目には、初期の Web ベースのアプリケーションによって並列性と可用性の限界が押し広げられることはありませんでした。近年では、ハードドライブやメモリーの容量および価格性能比がすべて向上しています。MySQL のパフォーマンスの限界を押し広げているユーザーは、信頼性やクラッシュリカバリに多くの関心を持っています。MySQL データベースは、大規模で、高負荷で、強固で、分散型で、重要です。

InnoDB は、このようなユーザーの最優先事項に対処します。ストレージエンジンの使用傾向は、より拡張可能な InnoDB の方へシフトしています。したがって、MySQL 5.5 は、InnoDB をデフォルトのストレージエンジンにするための論理遷移リリースでした。

MySQL は、以前は MyISAM テーブルが必要だったユースケースへの対処に取り組み続けています。MySQL 5.6 以上には、次のような特性があります。

- InnoDB は、FULLTEXT インデックスタイプを使用した全文検索を実行できます。詳細は、セクション 14.2.13.3「FULLTEXT インデックス」を参照してください。
- InnoDB は、読み取り専用または読み取りが大半のワークロードで、より適切に機能するようになりました。自動コミットモードでは、InnoDB クエリーに自動的な最適化が適用され、START TRANSACTION READ ONLY という構文を使用すると、読み取り専用としてトランザクションに明示的にマークできます。詳細は、セクション14.13.14「InnoDB の読み取り専用トランザクションの最適化」を参照してください。
- 読み取り専用メディア上に配布されたアプリケーションでは、InnoDB テーブルが使用されるようになりました。詳細は、セクション14.3.1「読み取り専用操作用の InnoDB の構成」を参照してください。

デフォルトの MySQL ストレージエンジンとしての InnoDB の重要性

MySQL 5.5.5 以降、新しいテーブル用のデフォルトのストレージエンジンは InnoDB です。この変更は、新たに作成されたテーブルの中で、ENGINE=MyISAM などの句を使用してストレージエンジンが指定されていないものに適用されます。このようにデフォルトの動作を変更すると、MySQL 5.5 は、MyISAM が使用されているテーブルが InnoDB に切り替えることによる利点を得られるかどうかを評価するための論理ポイントになる可能性があります。

MySQL 内部仕様の一部が実装されている mysql および information_schema データベースでは、引き続き MyISAM が使用されます。特に、付与テーブルを切り替えても、InnoDB を使用できません。

InnoDB テーブルの利点

MyISAM テーブルを使用しているが、技術的な理由でそれらに関与していない場合は、InnoDB テーブルを使用すると、さらに便利な点が数多く見つかるでしょう。

- ハードウェアまたはソフトウェアの問題が原因でサーバーがクラッシュした場合でも、その時点でデータベースに何が発生していたのかには関係なく、データベースの再起動後に特別なことは何もする必要がありません。InnoDB のクラッシュリカバリを使用すると自動的に、クラッシュ時の前にコミットされた変更はすべて完了し、処理中だったがコミットされなかった変更はすべて取り消されます。単に再起動し、終了した場所から続行するだけです。このプロセスは、MySQL 5.1 以前よりも大幅に高速になりました。
- テーブルおよびインデックスのデータにアクセスすると、そのデータは InnoDB のバッファープールにキャッシュされます。頻繁に使用されるデータは、直接メモリーから処理されます。このキャッシュは非常に数多くのタイプの情報に適用され、これにより処理速度が大幅に上がります。その結果、専用のデータベースサーバーでは、最大で物理メモリーの 80% が InnoDB のバッファープールに割り当てられます。
- 関連データをさまざまなテーブルに分割すると、強制的に参照整合性が適用される外部キーを設定できます。データを更新または削除すると、ほかのテーブル内の関連データも自動的に更新または削除されます。プライマリテーブル内に対応するデータが存在しないセカンダリテーブルにデータを挿入しようとすると、自動的に不正なデータが除外されます。
- ディスク上またはメモリー内のデータが破損した場合は、偽のデータを使用する前に、チェックサムメカニズムによって警告が発行されます。
- テーブルごとに適切な主キーカラムを持つデータベースを設計すると、これらのカラムが関与する操作が自動的に最適化されます。WHERE 句、ORDER BY 句、GROUP BY 句、および結合操作では、主キーカラムへの参照が非常に高速です。
- 挿入、更新、および削除は、変更バッファリングと呼ばれる自動化メカニズムによって最適化されます。InnoDB では、同じテーブルへの並列読み取りおよび書き込みアクセスが許可されているだけでなく、ディスク I/O が効率化されるように変更されたデータがキャッシュに入れられます。
- パフォーマンスの利点は、長時間実行されるクエリーを含む巨大なテーブルだけに限定されません。同じ行が 1 つのテーブルから何度もアクセスされると、適応型ハッシュインデックスと呼ばれる機能に引き継がれ、ハッシュテーブルから読み取られたかのように、これらの検索がさらに高速になります。

InnoDB テーブルのベストプラクティス

長期間 InnoDB を使用していれば、すでにトランザクションや外部キーなどの機能について理解できています。そうでない場合は、この章全体でこれらについて参照してください。手短に言えば、次のとおりです。

- もっとも頻繁にクエリーが実行されるカラム (複数の場合あり) を使用しているすべてのテーブルに、主キーを指定します。明示的な主キーが存在しない場合は、自動インクリメント値を指定します。
- 複数のテーブルにある同じ ID 値に基づいて、それらのテーブルからデータを抽出する場合は、結合の概念を取り入れます。結合のパフォーマンスを高速にするには、結合カラム上に外部キーを定義し、各テーブル内でそ

れらのカラムを同じデータ型で宣言します。また、外部キーを使用すると、影響を受けるすべてのテーブルに削除または更新が反映され、親テーブルに対応する ID が存在しない場合は、子テーブル内のデータの挿入が回避されます。

- 自動コミットをオフにします。1 秒間に何百回もコミットすると、パフォーマンスに上限が設定されます (これは、ストレージデバイスの書き込み速度で制限されます)。
- 関連する DML 操作のセットを START TRANSACTION と COMMIT ステートメントで囲むことで、トランザクションにグループ化します。頻繁にはコミットしたくない一方で、コミットなしで何時間も実行される INSERT、UPDATE、または DELETE ステートメントの巨大なバッチも発生させたくありません。
- LOCK TABLE ステートメントの使用を停止します。InnoDB は、一度に同じテーブルへのすべての読み取りおよび書き込みを行うことで、信頼性や高パフォーマンスを犠牲にせずに、複数のセッションを処理できます。行のセットへの排他的な書き込みアクセス権を取得するには、SELECT ... FOR UPDATE という構文を使用して、更新対象の行のみをロックします。
- innodb_file_per_table オプションを有効にして、単一の巨大なシステムテーブルスペース内の代わりに、個別のファイルに各テーブル用のデータおよびインデックスを配置します。この設定は、テーブルの圧縮および高速の切り捨てなどのその他の機能の一部を使用する際に必要となります。
- 使用中のデータおよびアクセスパターンによって、CREATE TABLE ステートメントで新しい InnoDB テーブルの圧縮機能 (ROW_FORMAT=COMPRESSED) からの利点が得られるかどうかを評価します。読み取りおよび書き込みの機能を犠牲にせずに、InnoDB テーブルを圧縮できます。
- オプション --sql_mode=NO_ENGINE_SUBSTITUTION を付けてサーバーを実行して、CREATE TABLE の ENGINE= 句で指定されたストレージエンジンで問題が発生した場合に、別のストレージエンジンを使用してテーブルが作成されないようにします。

InnoDB テーブルに対する最近の改善点

- テーブルおよび関連付けられたインデックスを圧縮できます。
- 以前よりも大幅に小さいパフォーマンスや可用性への影響度で、インデックスを作成および削除できます。
- テーブルの切り捨てが大幅に高速になり、InnoDB でのみ再使用される可能性のあるシステムテーブルスペース内の領域を解放するのではなく、オペレーティングシステムで再使用されるディスク領域を解放できます。
- DYNAMIC 行フォーマットを使用することで、テーブルデータのストレージレイアウトが BLOB および長いテキストフィールドでより効率的になりました。
- INFORMATION_SCHEMA テーブルでクエリーを実行することで、ストレージエンジンの内部動作をモニターできます。
- performance_schema テーブルでクエリーを実行することで、ストレージエンジンのパフォーマンスを詳細にモニターできます。
- パフォーマンスに関して多くの改善点があります。特に、クラッシュリカバリ、つまりデータベースが再起動するときにすべてのデータを自動的に整合させる処理の速度および信頼性が向上しました (InnoDB ユーザーが従来経験してきた速度よりずっと高速です)。データベースが大きいほど、大幅に速度が向上します。

  ほとんどの新しいパフォーマンス機能は自動的です。そうでない場合でも、必要なことは、多くても構成オプションの値を設定するだけです。詳細は、セクション14.13「InnoDB のパフォーマンス」を参照してください。アプリケーションコードで適用できる InnoDB 固有のチューニング技術については、セクション 8.5「InnoDB テーブルの最適化」を参照してください。上級ユーザーは、セクション14.12「InnoDB の起動オプションおよびシステム変数」を再確認してください。

デフォルトのストレージエンジンとして InnoDB を使用したテストおよびベンチマーク

MySQL 5.1 以前から MySQL 5.5 以降へのアップグレードが完了する前でも、データベースサーバーまたはアプリケーションでデフォルトのストレージエンジンとして、InnoDB が正常に動作するかどうかをプレビューできます。以前の MySQL リリースでデフォルトのストレージエンジンとして InnoDB を設定するには、コマンド行で --default-storage-engine=InnoDB を指定するか、または my.cnf ファイルの [mysqld] セクションに default-storage-engine=innodb を追加してから、サーバーを再起動します。

デフォルトのストレージエンジンを変更しても、新たに作成されたテーブルしか影響を受けないため、アプリケーションのインストールおよび設定ステップをすべて実行して、すべてが正しくインストールされたことを確認します。次に、すべてのアプリケーション機能を実行して、データのロード、編集、およびクエリー機能がすべて動作することを確認します。テーブルが一部の MyISAM 固有の機能に依存している場合は、エラーが受信さ

れます。エラーを回避するには、ENGINE=MyISAM 句を CREATE TABLE ステートメントに追加します (たとえば、全文検索に依存するテーブルは InnoDB テーブルではなく、MyISAM テーブルにする必要があります)。

ストレージエンジンについて慎重な決定を行わなかった場合に、特定のテーブルが InnoDB で作成されたときにどのように動作するのかをプレビューするには、テーブルごとに ALTER TABLE table_name ENGINE=InnoDB; コマンドを発行します。また、元のテーブルを配布せずに、テストクエリーおよびその他のステートメントを実行するには、次のようなコピーを作成します。

```
CREATE TABLE InnoDB_Table (...) ENGINE=InnoDB AS SELECT * FROM MyISAM_Table;
```

MySQL 5.5 以上には、非常に多くの InnoDB のパフォーマンス拡張機能があるため、現実的なワークロードで完全なアプリケーションを使用したときのパフォーマンスについて正確な考察を得るには、最新の MySQL サーバーをインストールして、ベンチマークを実行してください。

完全なアプリケーションのライフサイクル (インストールから頻繁な使用まで)、およびサーバーの再起動をテストします。電源障害のシミュレーションを行うために、データベースの負荷が高いときにサーバープロセスを強制終了し、サーバーの再起動時にデータが正常にリカバリされるかどうかを確認します。

特に、マスターおよびスレーブ上でさまざまな MySQL バージョンやオプションを使用している場合は、レプリケーション構成をテストします。

InnoDB がデフォルトのストレージエンジンであるかどうかの確認

古い MySQL を使用して what-if テストを行うのか、最新の MySQL を使用して包括的なテストを行うのかに関係なく、InnoDB のステータスを確認する方法は、次のとおりです。

- SHOW ENGINES; コマンドを発行して、さまざまな MySQL ストレージエンジンをすべて表示します。InnoDB 行で DEFAULT を探します。
- InnoDB がまったく存在しない場合は、InnoDB のサポートなしでコンパイルされた mysqld バイナリがあるため、別のバイナリを入手する必要があります。
- InnoDB は存在するが、無効になっている場合は、起動オプションおよび構成ファイルまで戻って、すべての skip-innodb オプションを削除します。

14.1.2 InnoDB の可用性チェック

使用中のサーバーで InnoDB がサポートされているかどうかを確認するには、SHOW ENGINES ステートメントを使用します。(InnoDB はデフォルトの MySQL ストレージエンジンになったため、サポートされない可能性があるのは、非常に特殊な環境の場合だけです。)

14.1.3 InnoDB の無効化

オラクルでは、ローカルシステム上で運用されている単一ユーザーの Wiki やブログから、パフォーマンスの限界を押し広げているハイエンドのアプリケーションまでの一般的なデータベースアプリケーションで優先されるストレージエンジンとして、InnoDB が推奨されています。MySQL 5.6 では、InnoDB が新しいテーブル用のデフォルトストレージエンジンです。

InnoDB テーブルを使用しない場合:

- InnoDB ストレージエンジンを無効にするには、--innodb=OFF または --skip-innodb オプションを付けてサーバーを起動します。

  > **注記**
  >
  > MySQL 5.6.21 の時点では、--skip-innodb オプションは引き続き機能しますが、非推奨となったため、使用されると警告が返されます。これは今後の MySQL リリースで削除されます。これは、そのシノニム (--innodb=OFF や --disable-innodb など) にも適用されます。

- デフォルトのストレージエンジンは InnoDB であるため、--default-storage-engine および --default-tmp-storage-engine を使用して、永続的なテーブルと TEMPORARY テーブルの両方についてデフォルトを別のエンジンに設定しないかぎり、サーバーは起動しません。
- InnoDB 関連の information_schema テーブルでクエリーが実行されるときに、サーバーがクラッシュすることを回避するには、それらのテーブルに関連付けられたプラグインも無効にします。MySQL 構成ファイルの [mysqld] セクションで、次のように指定します。

```
loose-innodb-trx=0
loose-innodb-locks=0
loose-innodb-lock-waits=0
loose-innodb-cmp=0
loose-innodb-cmp-per-index=0
loose-innodb-cmp-per-index-reset=0
loose-innodb-cmp-reset=0
loose-innodb-cmpmem=0
loose-innodb-cmpmem-reset=0
loose-innodb-buffer-page=0
loose-innodb-buffer-page-lru=0
loose-innodb-buffer-pool-stats=0
loose-innodb-metrics=0
loose-innodb-ft-default-stopword=0
loose-innodb-ft-inserted=0
loose-innodb-ft-deleted=0
loose-innodb-ft-being-deleted=0
loose-innodb-ft-config=0
loose-innodb-ft-index-cache=0
loose-innodb-ft-index-table=0
loose-innodb-sys-tables=0
loose-innodb-sys-tablestats=0
loose-innodb-sys-indexes=0
loose-innodb-sys-columns=0
loose-innodb-sys-fields=0
loose-innodb-sys-foreign=0
loose-innodb-sys-foreign-cols=0
```

14.2 InnoDB の概念とアーキテクチャー

このセクションの情報では、InnoDB テーブルを使用することで大部分のパフォーマンスおよび機能を取得する際に役立つバックグラウンドが提供されます。対象者は次のとおりです。

- 使い慣れていると思われる機能と、まったく新しい機能を説明するために、別のデータベースシステムから MySQL に切り替えている任意のユーザー。
- InnoDB がデフォルトの MySQL ストレージエンジンになったために、MyISAM テーブルから InnoDB に移行している任意のユーザー。
- InnoDB テーブルの設計上の考慮事項、パフォーマンスの特性、および拡張性を詳細なレベルで理解するために、アプリケーションアーキテクチャーまたはソフトウェアスタックを検討している任意のユーザー。

このセクションでは、次のことを学習します。

- どのくらい厳密に InnoDB が ACID の原則に準拠しているのか。
- どのように InnoDB がトランザクションを実装するのか、およびどのようにトランザクションの内部動作と、使い慣れているその他のデータベースシステムを比較するのか。
- どのように InnoDB が、クエリーおよび DML ステートメントが同じテーブルの読み取りと書き込みを同時に実行できる行レベルロックを実装するのか。
- マルチバージョン並列処理制御 (MVCC) によって、適切な時間になる前にトランザクションがその他の各データを表示または変更することがどのように回避されるのか。
- ディスク上の InnoDB 関連のオブジェクト (テーブル、インデックス、テーブルスペース、Undo ログ、および Redo ログ) の物理レイアウト。

14.2.1 MySQL および ACID モデル

ACID モデルは、ビジネスデータおよびミッションクリティカルなアプリケーションで重要となる信頼性の側面が強調されたデータベース設計原則のセットです。ソフトウェアのクラッシュやハードウェアの誤動作などの例外的な状況でも、データが破損せず、結果が歪曲されないように、MySQL には、ACID モデルに厳密に準拠した InnoDB ストレージエンジンなどのコンポーネントが含まれています。ACID に準拠した機能に依存していれば、一貫性チェックおよびクラッシュリカバリのメカニズムを再開発する必要がありません。追加のソフトウェアの保護手段、信頼性が最高のハードウェア、または少量のデータ損失や不整合に耐えることができるアプリケーションが備わっている場合は、ACID の信頼性の一部と引き換えに、パフォーマンスやスループットが向上するように MySQL の設定を調整できます。

次のセクションでは、どのように MySQL の機能 (特に InnoDB ストレージエンジン) が ACID モデルのカテゴリとやりとりするのかについて説明します。

- A: 原子性。
- C: 一貫性。
- I:: 分離性。
- D: 持続性。

原子性

ACID モデルの原子性の側面には、主に InnoDB のトランザクションが関与しています。関連する MySQL の機能は次のとおりです。

- 自動コミット設定。
- COMMIT ステートメント。
- ROLLBACK ステートメント。
- INFORMATION_SCHEMA テーブルの運用データ。

一貫性

ACID モデルの一貫性の側面には、主にクラッシュからデータを保護するための内部的な InnoDB 処理が関与しています。関連する MySQL の機能は次のとおりです。

- InnoDB 二重書き込みバッファー。
- InnoDB クラッシュリカバリ。

分離性

ACID モデルの分離性の側面には、主に InnoDB のトランザクション (特に、各トランザクションに適用される分離レベル) が関与しています。関連する MySQL の機能は次のとおりです。

- 自動コミット設定。
- SET ISOLATION LEVEL ステートメント。
- InnoDB ロックの低レベルの詳細。これらの詳細は、パフォーマンスチューニング時に INFORMATION_SCHEMA テーブルから参照します。

持続性

ACID モデルの持続性の側面には、特定のハードウェア構成とやりとりする MySQL ソフトウェアの機能が関与しています。CPU、ネットワーク、およびストレージデバイスの性能に応じて多くの可能性が考えられるため、具体的なガイドラインを示す際は、この側面がもっとも複雑になります。(これらのガイドラインに従うことは、「新しいハードウェア」を購入するという形になる場合があります。)関連する MySQL の機能は次のとおりです。

- innodb_doublewrite 構成オプションでオンとオフが切り替えられる InnoDB の二重書き込みバッファー。
- innodb_flush_log_at_trx_commit 構成オプション。
- sync_binlog 構成オプション。
- innodb_file_per_table 構成オプション。
- ストレージデバイス内の書き込みバッファー (ディスクドライブ、SSD、RAID アレイなど)。
- ストレージデバイス内のバッテリーでバックアップされるキャッシュ。
- MySQL を実行する際に使用されるオペレーティングシステム (特に、fsync() システムコールでのサポート)。
- MySQL サーバーを実行し、MySQL データを格納するすべてのコンピュータサーバーおよびストレージデバイスへの電力を保護する無停電電源装置 (UPS)。
- バックアップ方針 (頻度、バックアップのタイプ、バックアップの保存期間など)。

- 分散型またはホスト型のデータアプリケーションの場合、MySQL サーバー用のハードウェアが配置されているデータセンター、およびデータセンター間のネットワーク接続の特定の特性。

14.2.2 InnoDB のトランザクションモデルおよびロック

トランザクションおよびロックは InnoDB ストレージエンジンに関連するため、大規模、高負荷、または高信頼性のデータベースアプリケーションを実装したり、別のデータベースシステムから大量のコードを移植したり、MySQL のパフォーマンスを調整したりするには、これらの概念を理解する必要があります。

InnoDB トランザクションモデルの目標は、マルチバージョンデータベースの最高の特性を従来の二相ロックと組み合わせることです。InnoDB は、行レベルでロックを行い、デフォルトではクエリーを Oracle 式の非ロックの一貫性読み取りとして実行します。InnoDB のロック情報は非常に高い空間効率で格納されるため、ロックのエスカレーションは必要ありません。一般に、何人かのユーザーは、InnoDB テーブル内のすべての行、または行のランダムなサブセットをロックすることが許可されています。これにより、InnoDB のメモリーが使い果たされることはありません。

InnoDB では、すべてのユーザーアクティビティーがトランザクション内部で発生します。自動コミットモードが有効になっている場合は、各 SQL ステートメント自体に単一のトランザクションが生成されます。MySQL は、デフォルトで新しい接続のセッション開始時に自動コミットを有効にするため、各 SQL ステートメントからエラーが返されなかった場合に、そのステートメントのあとでコミットを実行します。ステートメントからエラーが返された場合、コミットまたはロールバックの動作はそのエラーによって異なります。セクション 14.19.4「InnoDB のエラー処理」を参照してください。

自動コミットが有効になっているセッションでは、明示的な START TRANSACTION または BEGIN ステートメントで起動し、COMMIT または ROLLBACK ステートメントで終了することで、複数ステートメントのトランザクションを実行できます。セクション13.3.1「START TRANSACTION、COMMIT、および ROLLBACK 構文」を参照してください。

SET autocommit = 0 でセッション内の自動コミットモードを無効にすると、そのセッションでは常にトランザクションが開かれた状態になります。COMMIT または ROLLBACK ステートメントは現在のトランザクションを終了し、新しいセッションを開始します。

COMMIT は、現在のトランザクション内で行われた変更は永続的であり、その他のセッションから表示できることを意味します。反対に、ROLLBACK ステートメントは、現在のトランザクションによって行われたすべての変更を取り消します。COMMIT と ROLLBACK は両方とも、現在のトランザクション中に設定されたすべての InnoDB ロックを解除します。

SQL:1992 のトランザクション分離レベルに関しては、デフォルトの InnoDB レベルは REPEATABLE READ です。InnoDB では、SQL 標準に記載された 4 つのトランザクション分離レベル (READ UNCOMMITTED、READ COMMITTED、REPEATABLE READ、SERIALIZABLE) がすべて提供されます。

ユーザーは SET TRANSACTION ステートメントを使用して単一のセッションまたは後続のすべての接続の分離レベルを変更できます。すべての接続に対するサーバーのデフォルトの分離レベルを設定するには、コマンド行上、またはオプションファイル内で --transaction-isolation オプションを使用します。分離レベルおよびレベル設定構文についての詳細は、セクション13.3.6「SET TRANSACTION 構文」を参照してください。

InnoDB では、通常、行レベルロックでネクストキーロックが使用されます。つまり、InnoDB はインデックスレコードのほかに、インデックスレコードの前のギャップもロックすることで、インデックス付きの値がツリーデータ構造内のそのギャップに挿入されるその他のセッションによって挿入されることをブロックできます。ネクストキーロックは、インデックスレコードとその前のギャップをロックするロックを参照します。ギャップロックは、いくつかのインデックスレコードの前のギャップのみをロックするロックを参照します。

行レベルロック、およびギャップロックが無効になる状況についての詳細は、セクション14.2.6「InnoDB のレコード、ギャップ、およびネクストキーロック」を参照してください。

14.2.3 InnoDB のロックモード

InnoDB では、2 つのロックタイプ (共有 (S) ロックと排他 (X) ロック) がある標準の行レベルロックが実装されます。レコード、ギャップ、およびネクストキーの各ロックタイプについては、セクション14.2.6「InnoDB のレコード、ギャップ、およびネクストキーロック」を参照してください。

- 共有 (S) ロックでは、ロックを保持するトランザクションによる行の読み取りが許可されます。
- 排他 (X) ロックでは、ロックを保持するトランザクションによる行の更新または削除が許可されます。

トランザクション T1 が行 r に対する共有 (S) ロックを保持している場合、別のトランザクション T2 からの行 r に対するロック要求は次のように処理されます。

- T2 による S ロックに対するリクエストは、すぐに付与できます。結果として、T1 と T2 の両方が r 上で S ロックを保持します。
- T2 による X ロックに対するリクエストは、すぐに付与できません。

トランザクション T1 が行 r 上で排他 (X) ロックを保持している場合は、r 上のいずれかのタイプのロックに対する一部の個別のトランザクション T2 からのリクエストは、すぐに付与できません。代わりに、トランザクション T2 は、行 r 上でトランザクション T1 のロックが解放されるまで待機する必要があります。

インテンションロック

さらに、InnoDB では、レコードロックとテーブル全体のロックが共存することを許可する複数粒度ロックもサポートされています。複数粒度レベルでのロックを実用的にするために、インテンションロックと呼ばれる追加のロックタイプが使用されます。インテンションロックとは、あとでトランザクションがそのテーブル内の行で必要となるロックのタイプ (共有または排他) を示す InnoDB のテーブルロックです。トランザクション T がテーブル t 上に指定されたタイプのロックをリクエストしたと仮定すると、InnoDB で使用されるインテンションロックには、次の 2 つタイプがあります。

- インテンション共有 (IS): トランザクション T は意図的に、テーブル t 内の各行に S ロックを設定します。
- インテンション排他 (IX): トランザクション T は意図的に、これらの行に X ロックを設定します。

たとえば、SELECT ... LOCK IN SHARE MODE は IS ロックを設定し、SELECT ... FOR UPDATE は IX ロックを設定します。

インテンションロックの手順は次のとおりです。

- トランザクションがテーブル t のある行の S ロックを取得するには、まず t の IS またはそれより強いロックを取得する必要があります。
- トランザクションがある行の X ロックを取得するには、まず t の IX ロックを取得する必要があります。

これらのルールをまとめる際は、次に示すロックタイプ互換性マトリクスを使用すると便利です。

	X	IX	S	IS
X	競合	競合	競合	競合
IX	競合	互換	競合	互換
S	競合	競合	互換	互換
IS	競合	互換	互換	互換

ロックに既存のロックとの互換性がある場合は、リクエスト元のトランザクションにロックが付与されますが、既存のロックと競合している場合は、ロックが付与されません。トランザクションは、競合している既存のロックが解放されるまで待機します。ロックリクエストが既存のロックと競合し、デッドロックが発生するために付与できない場合は、エラーが発生します。

したがって、インテンションロックでは、完全なテーブルリクエスト (LOCK TABLES ... WRITE など) 以外は何もブロックされません。IX および IS ロックの主な目的は、だれかが行をロックしていることや、テーブル内の行をロックしようとしていることを示すことです。

デッドロックの例

次の例は、ロックリクエストによってデッドロックが発生したときに、どのようにエラーが発生するのかを示しています。この例には、A と B の 2 つのクライアントが登場します。

最初に、クライアント A が行を 1 つ含むテーブルを作成し、トランザクションを開始します。トランザクション内で、A は共有モードで選択した行で S ロックを取得します。

```
mysql> CREATE TABLE t (i INT) ENGINE = InnoDB;
Query OK, 0 rows affected (1.07 sec)

mysql> INSERT INTO t (i) VALUES(1);
Query OK, 1 row affected (0.09 sec)

mysql> START TRANSACTION;
Query OK, 0 rows affected (0.00 sec)

mysql> SELECT * FROM t WHERE i = 1 LOCK IN SHARE MODE;
+------+
```

```
| i    |
+------+
|    1 |
+------+
1 row in set (0.10 sec)
```

次に、クライアント B がトランザクションを開始し、テーブルから行を削除しようとします。

```
mysql> START TRANSACTION;
Query OK, 0 rows affected (0.00 sec)

mysql> DELETE FROM t WHERE i = 1;
```

削除操作を行うには、X ロックが必要です。クライアント A が保持している S ロックとの互換性がないために、ロックを付与できません。そのため、リクエストはその行のロックリクエストのキューに入れられ、クライアント B はブロックされます。

最後に、クライアント A もテーブルから行を削除しようとします。

```
mysql> DELETE FROM t WHERE i = 1;
ERROR 1213 (40001): Deadlock found when trying to get lock;
try restarting transaction
```

クライアント A は行を削除するために X ロックが必要であるため、ここでデッドロックが発生します。ただし、クライアント B はすでに X ロックに対するリクエストを持っていて、クライアント A がその S ロックを解放するまで待機しているため、そのロックリクエストを付与することはできません。B による X ロックに対する以前のリクエストが原因で、A が保持している S ロックを X ロックにアップグレードすることもできません。その結果、InnoDB はクライアントのいずれかに対してエラーを生成し、そのロックを解放します。クライアントは、次のエラーを返します。

```
ERROR 1213 (40001): Deadlock found when trying to get lock;
try restarting transaction
```

この時点で、ほかのクライアントに対するロックリクエストを付与できるようになり、テーブルから行が削除されます。

> **注記**
>
> InnoDB Monitor の出力の LATEST DETECTED DEADLOCK セクションには、「TOO DEEP OR LONG SEARCH IN THE LOCK TABLE WAITS-FOR GRAPH, WE WILL ROLL BACK FOLLOWING TRANSACTION」というメッセージが含まれます。これは、待機リスト上のトランザクション数が 200 の制限に達したことを示します。この制限は、LOCK_MAX_DEPTH_IN_DEADLOCK_CHECK で定義されます。200 個のトランザクションを超える待機リストはデッドロックとして処理され、待機リストをチェックしようとするトランザクションはロールバックされます。
>
> ロックスレッドが待機リスト上のトランザクションが所有する 1,000,000 個を超えるロックを参照する必要がある場合も、同じエラーが発生する可能性があります。1,000,000 個のロック制限は、LOCK_MAX_N_STEPS_IN_DEADLOCK_CHECK で定義されます。

14.2.4 一貫性非ロック読み取り

一貫性読み取りとは、InnoDB がマルチバージョンを使用して、ある時点でのデータベースのスナップショットをクエリーに提供することを意味します。クエリーには、その時点よりも前にコミットされたトランザクションによる変更のみが表示され、その時点よりもあとのトランザクションまたはコミットされていないトランザクションによる変更は表示されません。このルールの例外として、同じトランザクション内の以前のステートメントによる変更はクエリーに表示されます。この例外によって、次のような異常が発生します。テーブル内の一部の行を更新すると、更新された行の最新バージョンが SELECT に表示されますが、いずれかの行の旧バージョンも表示される可能性があります。その他のセッションで同じテーブルが同時に更新される場合、その異常は、データベースに存在しない状態でテーブルが表示される可能性があることを意味します。

トランザクション分離レベルが REPEATABLE READ (デフォルトのレベル) である場合は、同じトランザクション内のすべての一貫性読み取りで、そのトランザクション内の最初のこのような読み取りで確立されたスナップショットが読み取られます。現在のトランザクションをコミットしたあとに、新しいクエリーを発行すると、クエリーの新しいスナップショットを取得できます。

分離レベルが READ COMMITTED の場合は、トランザクション内の各一貫性読み取りで、独自の新しいスナップショットが設定され、読み取られます。

一貫性読み取りは、InnoDB が READ COMMITTED および REPEATABLE READ 分離レベルで SELECT ステートメントを処理する際のデフォルトモードです。一貫性読み取りではアクセスされるテーブル上にロックが設定されないため、その他のセッションも、そのテーブル上で一貫性読み取りが実行されるときと同時に、それらのテーブルを自由に変更できます。

デフォルトの REPEATABLE READ 分離レベルで実行していると仮定します。一貫性読み取り (つまり、通常の SELECT ステートメント) を発行すると、InnoDB は、クエリーがデータベースを参照するときの基準となるタイムポイントをトランザクションに付与します。タイムポイントが割り当てられたあとに、別のトランザクションが行を削除してコミットした場合は、その行が削除済みとして表示されません。挿入および更新も同様に処理されます。

> **注記**
>
> データベースの状態のスナップショットは、トランザクション内の SELECT ステートメントに適用されますが、DML ステートメントには必ずしも適用されるとは限りません。一部の行を挿入または変更してから、そのトランザクションをコミットする場合は、そのセッションでクエリーが実行される可能性がない場合でも、別の並列実行 REPEATABLE READ トランザクションから発行された DELETE または UPDATE ステートメントによって、コミットされたばかりの行が影響を受ける可能性があります。トランザクションによって別のトランザクションでコミットされた行が更新または削除されると、これらの変更を現在のトランザクションに表示できるようになります。たとえば、次のような状況が発生する可能性があります。
>
> ```
> SELECT COUNT(c1) FROM t1 WHERE c1 = 'xyz'; -- Returns 0: no rows match.
> DELETE FROM t1 WHERE c1 = 'xyz'; -- Deletes several rows recently committed by other transaction.
>
> SELECT COUNT(c2) FROM t1 WHERE c2 = 'abc'; -- Returns 0: no rows match.
> UPDATE t1 SET c2 = 'cba' WHERE c2 = 'abc'; -- Affects 10 rows: another txn just committed 10 rows with 'abc' values.
> SELECT COUNT(c2) FROM t1 WHERE c2 = 'cba'; -- Returns 10: this txn can now see the rows it just updated.
> ```

トランザクションをコミットしてから、別の SELECT または START TRANSACTION WITH CONSISTENT SNAPSHOT を実行すると、タイムポイントを進めることができます。

これは、マルチバージョン並列処理制御と呼ばれます。

次の例では、セッション B が挿入をコミットし、セッション A も同様にコミットした場合にのみ、B によって挿入された行が A に表示されます。これにより、タイムポイントが B のコミットよりも先に進みます。

```
         Session A              Session B

       SET autocommit=0;      SET autocommit=0;
time
|      SELECT * FROM t;
|      empty set
|                             INSERT INTO t VALUES (1, 2);
|
v      SELECT * FROM t;
       empty set
                              COMMIT;

       SELECT * FROM t;
       empty set

       COMMIT;

       SELECT * FROM t;
       ---------------------
       |  1   |  2   |
       ---------------------
       1 row in set
```

データベースの「最新」状態を確認する場合は、READ COMMITTED 分離レベルとロック読み取りのいずれかを使用します。

```
SELECT * FROM t LOCK IN SHARE MODE;
```

分離レベルが READ COMMITTED の場合は、トランザクション内の各一貫性読み取りで、独自の新しいスナップショットが設定され、読み取られます。LOCK IN SHARE MODE の場合は、代わりにロック読み取りが発生します。SELECT は、最新の行を含むトランザクションが終了するまでブロックされます (セクション14.2.5「ロック読み取り (SELECT ... FOR UPDATE および SELECT ... LOCK IN SHARE MODE)」を参照)。

特定の DDL ステートメントでは、一貫性読み取りが機能しません。

- DROP TABLE では、MySQL が削除されたテーブルを使用できず、そのテーブルは InnoDB によって破棄されるため、一貫性読み取りが機能しません。
- ALTER TABLE では、そのステートメントで元のテーブルの一時コピーが作成され、元のテーブルは一時コピーが構築されるときに削除されるため、一貫性読み取りが機能しません。トランザクション内で一貫性読み取りを再発行しても、新しいテーブル内の行はトランザクションのスナップショット取得されたときには存在していなかったため、表示できません。MySQL 5.6.6 の時点では、この場合に、トランザクションから「Table definition has changed, please retry transaction」という ER_TABLE_DEF_CHANGED エラーが返されます。

FOR UPDATE または LOCK IN SHARE MODE を指定しない INSERT INTO ... SELECT、UPDATE ... (SELECT)、CREATE TABLE ... SELECT などの句での選択では、読み取りのタイプが異なります。

- デフォルトでは、InnoDB はより強固なロックを使用し、SELECT 部分は READ COMMITTED と同様に機能します。この場合、同じトランザクション内でも、各一貫性読み取りで独自の新しいスナップショットが設定され、読み取られます。
- このような場合に一貫性読み取りを使用するには、innodb_locks_unsafe_for_binlog オプションを有効にし、トランザクションの分離レベルを READ UNCOMMITTED、READ COMMITTED、または REPEATABLE READ (つまり、SERIALIZABLE 以外のすべて) に設定します。この場合、選択したテーブルから読み取られた行には、ロックが設定されません。

14.2.5 ロック読み取り (SELECT ... FOR UPDATE および SELECT ... LOCK IN SHARE MODE)

データのクエリーを実行してから、同じトランザクション内で関連データを挿入または更新する場合は、通常の SELECT ステートメントで十分な保護が提供されません。ほかのトランザクションは、クエリーが実行されたばかりの同じ行を更新または削除できます。InnoDB では、追加の安全性が提供される 2 つのタイプのロック読み取りがサポートされています。

- SELECT ... LOCK IN SHARE MODE は、読み取られるすべての行に共有モードのロックを設定します。ほかのセッションもその行を読み取ることができますが、トランザクションがコミットするまで変更することはできません。これらの行のいずれかがコミットされていない別のトランザクションによって変更された場合、クエリーはそのトランザクションが終了するまで待機してから、最新の値を使用します。
- 検索でインデックスレコードが見つかった場合、SELECT ... FOR UPDATE は、行および関連付けられたすべてのエントリをロックします。この動作は、これらの行に UPDATE ステートメントを発行した場合と同じです。ほかのトランザクションは、これらの行の更新、SELECT ... LOCK IN SHARE MODE の実行、または特定のトランザクション分離レベルでのデータの読み取りからブロックされます。一貫性読み取りでは、読み取られたビュー内に存在するレコードに設定されたロックはすべて無視されます。(古いバージョンのレコードはロックできません。レコードのインメモリーコピー上の Undo ログに適用することで、再構築されます。)

これらの句は、主に、単一のテーブル内または複数のテーブルに分割された状態で、ツリー構造またはグラフ構造のデータを処理する際に役立ちます。エッジまたはツリー分岐をある場所から別の場所にトラバースしても、これらの「ポインタ」に戻ってその値を変更する権利を保有しています。

トランザクションがコミットまたはロールバックされると、LOCK IN SHARE MODE および FOR UPDATE クエリーで設定されたすべてのロックが解放されます。

> 注記
>
> SELECT FOR UPDATE を使用した更新対象の行のロックは、START TRANSACTION でトランザクションを開始するか、autocommit を 0 に設定することで、自動コミットが無効になっている場合にのみ適用されます。自動コミットが有効になっている場合は、指定に一致する行がロックされません。

使用例

child テーブルに新しい行を挿入し、子の行が parent テーブル内に親の行を持っていることを確認すると仮定します。アプリケーションコードを使用して、この操作シーケンス全体の参照整合性を確保できます。

まず、一貫性読み取りを使用して、PARENT テーブルでクエリーを実行し、親の行が存在することを確認します。CHILD テーブルに子の行を安全に挿入できますか。気付かないうちに、その他の一部のセッションで、SELECT と INSERT との間に親の行が削除された可能性もあるため、できません。

このような問題の可能性を回避するには、LOCK IN SHARE MODE を使用して SELECT を実行します。

```
SELECT * FROM parent WHERE NAME = 'Jones' LOCK IN SHARE MODE;
```

LOCK IN SHARE MODE クエリーから「Jones」という親が返されたら、CHILD テーブルに子のレコードを安全に追加し、トランザクションをコミットできます。PARENT テーブル内のアプリケーション行で読み取りまたは書き込みを行おうとするトランザクションは、ユーザーが完了するまで (つまり、すべてのテーブル内のデータが一貫性のある状態になるまで) 待機します。

もう 1 つの例では、CHILD テーブルに追加された各子に一意の識別子を割り当てる際に使用される CHILD_CODES テーブル内の整数カウンタフィールドを検討します。一貫性読み取りまたは共有モード読み取りを使用すると、データベースの 2 人のユーザーが同じカウンタ値を参照する可能性があり、2 つのトランザクションが同じ識別子を持つ行を CHILD テーブルに追加しようとすると、重複キーのエラーが発生するため、カウンタの現在の値を読み取る際には使用しないでください。

ここで、2 人のユーザーがカウンタを同時に読み取る場合、少なくとも 1 人のユーザーがカウンタを更新しようとするとデッドロックが発生するため、LOCK IN SHARE MODE は適切な解決策ではありません。

カウンタの読み取りおよび増分を実装するには、まず FOR UPDATE を使用してカウンタのロック読み取りを実行してから、カウンタを増分します。例:

```
SELECT counter_field FROM child_codes FOR UPDATE;
UPDATE child_codes SET counter_field = counter_field + 1;
```

SELECT ... FOR UPDATE は使用可能な最新データを読み取り、読み取られる各行上に排他ロックを設定します。したがって、検索された SQL UPDATE によって行上に設定される場合と同じロックが設定されます。

前述の説明は、単に SELECT ... FOR UPDATE がどのように機能するのかを示した例です。MySQL では、テーブルへの単一アクセスを使用するだけで、一意の識別子を生成する特定のタスクを実現できます。

```
UPDATE child_codes SET counter_field = LAST_INSERT_ID(counter_field + 1);
SELECT LAST_INSERT_ID();
```

この SELECT ステートメントは、単に (現在の接続に固有の) 識別子情報を取得するだけです。どのテーブルにもアクセスしません。

14.2.6 InnoDB のレコード、ギャップ、およびネクストキーロック

InnoDB のレコードレベルのロックには、レコードロック、ギャップロック、ネクストキーロックなどの複数のタイプがあります。共有ロック、排他ロック、およびインテンションロックについては、セクション 14.2.3「InnoDB のロックモード」を参照してください。

- レコードロック: これはインデックスレコードのロックです。
- ギャップロック: これはインデックスレコード間にあるギャップのロック、または先頭のインデックスレコードの前や末尾のインデックスレコードのあとにあるギャップのロックです。
- ネクストキーロック: これはインデックスレコードに対するレコードロックと、そのインデックスレコードの前にあるギャップに対するギャップロックとを組み合わせたものです。

レコードロック

レコードロックでは、テーブルにインデックスが定義されていなくても必ず、インデックスレコードがロックされます。このような場合は、InnoDB によって非表示のクラスタ化されたインデックスが作成され、このインデックスを使用してレコードロックが行われます。セクション14.2.13.2「クラスタインデックスとセカンダリインデックス」を参照してください。

ネクストキーロック

デフォルトでは、InnoDB は REPEATABLE READ トランザクション分離レベルで動作し、innodb_locks_unsafe_for_binlog システム変数は無効になっています。この場合、InnoDB はネクストキーロックを使用して検索およびインデックススキャンを行うため、ファントム行の発生を回避できます (セクション 14.2.7「ネクストキーロックによるファントム問題の回避」を参照)。

ネクストキーロックは、インデックス行ロックとギャップロックを組み合わせたものです。InnoDB は、テーブルインデックスを検索またはスキャンするときに、生成されたインデックスレコード上に共有ロックまたは排他ロックを設定するという方法で、行レベルロックを実行します。したがって、行レベルロックは、実際にはインデックスレコードロックです。さらに、あるインデックスレコードに対するネクストキーロックによって、そのインデックスレコードの前の「ギャップ」も影響を受けます。つまり、ネクストキーロックは、インデックス

レコードロックと、そのインデックスレコードの前のギャップに対するギャップロックとを組み合わせたものです。あるセッションがインデックス内のレコード R 上に共有ロックまたは排他ロックを持っている場合は、別のセッションがインデックスの順番で R の直前にあるギャップに新しいインデックスレコードを挿入できません。

あるインデックスに値 10、11、13、20 が含まれているとします。このインデックスでは、次の間隔をカバーするネクストキーロックが使用される可能性があります。ここで、(や) は間隔の端点が含まれないことを表し、[や] は間隔の端点が含まれることを表します。

```
(negative infinity, 10]
(10, 11]
(11, 13]
(13, 20]
(20, positive infinity)
```

最後の間隔ではネクストキーロックによって、インデックス内の最大値を上回るギャップ、およびインデックス内の実際のどの値よりも大きい値を持つ「最小上限」の擬似レコードがロックされます。最小上限は実際のインデックスレコードではないため、事実上、このネクストキーロックによってロックされるのは、最大インデックス値のあとにあるギャップのみです。

ギャップロック

前のセクションで示したネクストキーロックの例は、ギャップの範囲が単一のインデックス値、複数のインデックス値、または空になる場合もあることを示しています。

一意のインデックスを使用して一意の行を検索することで行をロックするステートメントでは、ギャップロックは必要ありません。(これには、検索条件に複数カラムの一意のインデックスの一部のカラムのみが含まれるケースは含まれません。この場合は、ギャップロックが発生します。)たとえば、id カラムに一意のインデックスが設定されている場合、次のステートメントで使用されるのは id の値が 100 の行に対するインデックスレコードロックだけとなり、ほかのセッションがそのレコードの前にあるギャップに行を挿入するかどうかは問題ではなくなります。

```
SELECT * FROM child WHERE id = 100;
```

id にインデックスが設定されていなかったり、一意でないインデックスが設定されていたりすると、このステートメントで先行するギャップがロックされます。

INSERT 操作では行の挿入前に、挿入インテンションギャップロックと呼ばれる一種のギャップロックが設定されます。このロックは、同じインデックスギャップに挿入する複数のトランザクションは、そのギャップ内の同じ場所に挿入しなければ相互に待機する必要がないように、意図的に挿入することを示しています。値が 4 と 7 のインデックスレコードが存在すると仮定します。それぞれ値 5 と 6 の挿入を試みる別々のトランザクションは、挿入される行の排他ロックを取得する前に挿入インテンションロックを使用して、4 と 7 の間にあるギャップをロックしますが、行の競合が発生しないため相互にブロックされません。インテンションロックについての詳細は、セクション14.2.3「InnoDB のロックモード」を参照してください。

さまざまなトランザクションによってギャップ上に競合するロックを保持できることも、ここで注目するべき点です。たとえば、トランザクション A はギャップ上に共有ギャップロック (ギャップ S ロック) を保持できる一方で、トランザクション B は同じギャップ上に排他ギャップロック (ギャップ X ロック) を保持します。競合するギャップロックが許可される理由は、レコードがインデックスからパージされる場合に、さまざまなトランザクションによってレコード上に保持されたギャップロックをマージする必要があるためです。

InnoDB のギャップロックは、「単に抑制的」です。つまり、ほかのトランザクションによるギャップへの挿入が停止されるだけです。したがって、ギャップ X ロックの効果はギャップ S ロックと同じです。

ギャップロックの無効化

ギャップロックは明示的に無効化できます。これは、トランザクション分離レベルを READ COMMITTED に変更するか、または innodb_locks_unsafe_for_binlog システム変数 (現在は非推奨です) を有効にすると発生します。このような状況では、ギャップロックは検索およびインデックススキャン時に無効化され、外部キー制約チェックおよび重複キーチェック時にのみ使用されます。

READ COMMITTED 分離レベルを使用するか、innodb_locks_unsafe_for_binlog を有効にした場合の効果はほかにもあります。一致しない行のレコードロックは、MySQL による WHERE 条件の評価が完了すると解放されます。UPDATE ステートメントの場合、InnoDB は最後にコミットされたバージョンが MySQL に返されるように、「半一貫性」読み取りを実行します。これにより、MySQL はその行が UPDATE の WHERE 条件に一致するかどうかを判断できます。

14.2.7 ネクストキーロックによるファントム問題の回避

同じクエリーでさまざまな時間にさまざまな行のセットが生成されると、いわゆるファントムの問題がトランザクション内で発生します。たとえば、SELECT が 2 回実行されたが、1 回目には返されなかった行が 2 回目には返された場合、その行が「ファントム」行です。

child テーブルの id カラム上にインデックスがあり、識別子の値が 100 よりも大きいすべての行をテーブルから読み取り、選択された行の一部のカラムをあとで更新するという意図でロックすると仮定します。

```
SELECT * FROM child WHERE id > 100 FOR UPDATE;
```

クエリーでは、id が 100 よりも大きい最初のレコードからインデックスがスキャンされます。このテーブルには id の値が 90 と 102 の行が格納されているものとします。スキャン範囲内のインデックスレコード上に設定されたロックによって、ギャップ (この場合のギャップは 90 から 102 まで) への挿入がロックアウトされていない場合は、別のセッションが id が 101 の新しい行をそのテーブルに挿入できます。同じトランザクション内で同じ SELECT を実行すると、クエリーから返された結果セット内に、id が 101 の新しい行 (「ファントム」) が含まれています。一連の行をデータ項目とみなせば、この新しいファントムの子は、「トランザクション中は読み取られるデータが変更されないようにトランザクションを実行できるべきである」というトランザクションの分離原則に違反しています。

ファントムの発生を回避できるように、InnoDB では通常、インデックス行ロックとギャップロックを組み合わせたネクストキーロックと呼ばれるアルゴリズムが使用されます。InnoDB は、テーブルインデックスを検索またはスキャンするときに、生成されたインデックスレコード上に共有ロックまたは排他ロックを設定するという方法で、行レベルロックを実行します。したがって、行レベルロックは、実際にはインデックスレコードロックです。さらに、あるインデックスレコードに対するネクストキーロックによって、そのインデックスレコードの前の「ギャップ」も影響を受けます。つまり、ネクストキーロックは、インデックスレコードロックと、そのインデックスレコードの前のギャップに対するギャップロックとを組み合わせたものです。あるセッションがインデックス内のレコード R 上に共有ロックまたは排他ロックを持っている場合は、別のセッションがインデックスの順番で R の直前にあるギャップに新しいインデックスレコードを挿入できません。

InnoDB はインデックスをスキャンするときに、インデックス内の最後のレコードのあとのギャップをロックすることもできます。前述の例では、まさにそれが行われています。id が 100 よりも大きいテーブルへの挿入が回避されるように、InnoDB で設定されたロックには、id 値 102 のあとのギャップに対するロックが含まれています。

ネクストキーロックを使用すると、アプリケーションに一意性チェックを実装できます。共有モードでデータを読み取るときに、挿入される行の重複が見られなければ、行を安全に挿入でき、読み取り中に後続の行に設定されたネクストキーロックによって、任意のユーザーによる重複行の挿入が回避されることを確認できます。したがって、ネクストキーロックを使用すれば、テーブル内に存在しないものも「ロック」できます。

ギャップロックは、セクション14.2.6「InnoDB のレコード、ギャップ、およびネクストキーロック」で説明した方法で無効にすることができます。ギャップロックが無効になっていると、ほかのセッションが新しい行をギャップに挿入できるため、ファントムの問題が発生する可能性があります。

14.2.8 InnoDB のさまざまな SQL ステートメントで設定されたロック

一般に、ロック読み取り、UPDATE、または DELETE では、SQL ステートメントの処理時にスキャンされるすべてのインデックスレコード上に、レコードロックが設定されます。 行を除外する WHERE 条件がステートメント内に存在するかどうかは、関係ありません。InnoDB には正確な WHERE 条件が記憶されませんが、スキャンされたインデックスの範囲は認識されます。通常、ロックはレコードの直前にある「ギャップ」への挿入もブロックするネクストキーロックです。ただし、ギャップロックは明示的に無効にすることができます。これにより、ネクストキーロックが使用されなくなります。詳細は、セクション14.2.6「InnoDB のレコード、ギャップ、およびネクストキーロック」を参照してください。トランザクション分離レベルによって、どのロックが設定されるのかも影響を受けます。セクション13.3.6「SET TRANSACTION 構文」を参照してください。

検索でセカンダリインデックスが使用され、設定されるインデックスレコードのロックが排他的である場合、InnoDB は対応するクラスタ化されたインデックスレコードを取得し、それらにロックを設定することも行います。

共有ロックと排他ロックの違いについては、セクション14.2.3「InnoDB のロックモード」を参照してください。

ステートメントに適したインデックスがなく、MySQL がステートメントを処理するためにテーブル全体をスキャンする必要がある場合は、テーブルのすべての行がロックされます。その結果、そのテーブルへのほかのユーザーによるすべての挿入がブロックされます。クエリーで不必要に複数の行がスキャンされないように、適切なインデックスを作成することが重要です。

SELECT ... FOR UPDATE または SELECT ... LOCK IN SHARE MODE では、スキャンされた行についてはロックが取得され、WHERE 句に指定された条件を満たさないなどの理由で結果セットに含める対象から除外された行

については、ロックが解放されることが予想されます。ただし場合によっては、クエリーの実行中に結果行とその元のソースとの関係が失われたために、行のロックがすぐに解除されない可能性もあります。たとえば UNION では、スキャン (およびロック) されたテーブル内の行が、結果セットに含める対象となるかどうかの評価前に、一時テーブルに挿入される可能性があります。この状況では、一時テーブル内の行と元のテーブル内の行との関係は失われているため、クエリー実行が終了するまで後者の行のロックは解除されません。

InnoDB は、次のように特定のロックタイプを設定します。

- SELECT ... FROM は一貫性読み取りであり、データベースのスナップショットを読み取り、トランザクションの分離レベルが SERIALIZABLE に設定されなければロックを設定しません。SERIALIZABLE レベルの場合、検索で見つかったインデックスレコード上に共有ネクストキーロックが設定されます。
- SELECT ... FROM ... LOCK IN SHARE MODE では、検索で見つかったすべてのインデックスレコード上に共有ネクストキーロックが設定されます。
- SELECT ... FROM ... FOR UPDATE は、検索で見つかったインデックスレコードに対して、ほかのセッションが SELECT ... FROM ... LOCK IN SHARE MODE を実行したり、特定のトランザクション分離レベルで読み取ったりすることをブロックします。一貫性読み取りでは、読み取られたビュー内に存在するレコードに設定されたロックはすべて無視されます。
- UPDATE ... WHERE ... は、検索で見つかったすべてのレコード上に排他ネクストキーロックを設定します。
- DELETE FROM ... WHERE ... は、検索で見つかったすべてのレコード上に排他ネクストキーロックを設定します。
- INSERT は、挿入される行に排他ロックを設定します。このロックは、ネクストキーロックではなくインデックスレコードロックである (つまり、ギャップロックが存在しない) ため、ほかのセッションが挿入された行の前にあるギャップに挿入することは回避されません。

  行の挿入前に、挿入インテンションギャップロックと呼ばれる一種のギャップロックが設定されます。このロックは、同じインデックスギャップに挿入する複数のトランザクションは、そのギャップ内の同じ場所に挿入しなければ相互に待機する必要がないように、意図的に挿入することを示しています。値が 4 と 7 のインデックスレコードが存在すると仮定します。それぞれ値 5 と 6 の挿入を試みる別々のトランザクションは、挿入される行の排他ロックを取得する前に挿入インテンションロックを使用して、4 と 7 の間にあるギャップをロックしますが、行の競合が発生しないため相互にブロックされません。

  重複キーエラーが発生すると、重複インデックスレコード上の共有ロックが設定されます。複数のセッションが同じ行を挿入しようとしているときに、別のセッションがすでに排他ロックを取得していた場合は、このように共有ロックを使用することでデッドロックが発生する可能性があります。これは、別のセッションがその行を削除した場合に発生する可能性があります。InnoDB テーブル t1 の構造が次のようになっているとします。

```
CREATE TABLE t1 (i INT, PRIMARY KEY (i)) ENGINE = InnoDB;
```

次に、3 つのセッションが次の処理を順番に実行するものとします。

セッション 1:

```
START TRANSACTION;
INSERT INTO t1 VALUES(1);
```

セッション 2:

```
START TRANSACTION;
INSERT INTO t1 VALUES(1);
```

セッション 3:

```
START TRANSACTION;
INSERT INTO t1 VALUES(1);
```

セッション 1:

```
ROLLBACK;
```

セッション 1 による最初の処理では、行の排他ロックが取得されます。セッション 2 と 3 の処理ではどちらも重複キーエラーが発生し、どちらのセッションも行の共有ロックをリクエストします。セッション 1 はロールバック時に行の排他ロックを解放し、キュー内のセッション 2 と 3 の共有ロックリクエストが付与されます。この時点でセッション 2 と 3 でデッドロックが発生します。どちらも他方が保持している共有ロックのために、行の排他ロックを取得できません。

キー値が 1 の行がテーブルに含まれている場合も似たような状況が発生し、3 つのセッションが次の処理を順番に実行します。

セッション 1:

```
START TRANSACTION;
DELETE FROM t1 WHERE i = 1;
```

セッション 2:

```
START TRANSACTION;
INSERT INTO t1 VALUES(1);
```

セッション 3:

```
START TRANSACTION;
INSERT INTO t1 VALUES(1);
```

セッション 1:

```
COMMIT;
```

セッション 1 による最初の処理では、行の排他ロックが取得されます。セッション 2 と 3 の処理ではどちらも重複キーエラーが発生し、どちらのセッションも行の共有ロックをリクエストします。セッション 1 はコミット時に行の排他ロックを解放し、キュー内のセッション 2 と 3 の共有ロックリクエストが付与されます。この時点でセッション 2 と 3 でデッドロックが発生します。どちらも他方が保持している共有ロックのために、行の排他ロックを取得できません。

- INSERT ... ON DUPLICATE KEY UPDATE は、重複キーエラーが発生したときに、更新される行に共有ロックではなく、排他ネクストキーロックが配置されるという点で、単純な INSERT と異なります。
- REPLACE は、一意のキーが競合していなければ、INSERT と同様に動作します。それ以外の場合は、置換される行に排他ネクストキーロックが配置されます。
- INSERT INTO T SELECT ... FROM S WHERE ... は、T に挿入された各行に、ギャップロックなしの排他インデックスレコードロックを設定します。トランザクション分離レベルが READ COMMITTED である場合、または innodb_locks_unsafe_for_binlog が有効になっていて、トランザクション分離レベルが SERIALIZABLE でない場合、InnoDB は一貫性読み取り (ロックなし) として S 上で検索を実行します。それ以外の場合、InnoDB は S から取得した行に共有ネクストキーロックを設定します。InnoDB は、後者の場合にロックを設定する必要があります。バックアップからのロールフォワードリカバリ時には、すべての SQL ステートメントを元とまったく同じ方法で実行する必要があります。

  CREATE TABLE ... SELECT ... は、INSERT ... SELECT の場合と同様に、SELECT を共有ネクストキーロックを使用して実行するか、一貫性読み取りとして実行します。

  構造文 REPLACE INTO t SELECT ... FROM s WHERE ... または UPDATE t ... WHERE col IN (SELECT ... FROM s ...) で SELECT が使用されると、InnoDB はテーブル s の行に共有ネクストキーロックを設定します。
- InnoDB は、テーブル上に事前に指定された AUTO_INCREMENT カラムの初期化中に、AUTO_INCREMENT カラムに関連付けられたインデックスの最後に排他ロックを設定します。InnoDB では、自動インクリメントカウンタにアクセスするときに、ロックがトランザクション全体の最後まででではなく、現在の SQL ステートメントの最後まで続く、特別な AUTO-INC テーブルロックモードが使用されます。AUTO-INC テーブルロックが保持されている間は、ほかのセッションはそのテーブルに挿入できません。セクション14.2.2「InnoDB のトランザクションモデルおよびロック」を参照してください。

  InnoDB は、ロックを設定せずに、事前に初期化された AUTO_INCREMENT カラムの値をフェッチします。
- FOREIGN KEY 制約がテーブル上で定義されている場合は、制約条件をチェックする必要がある挿入、更新、または削除が行われると、制約をチェックするために、参照されるレコード上に共有レコードレベルロックが設定されます。InnoDB は、制約が失敗する場合に備えて、これらのロックの設定も行います。
- LOCK TABLES はテーブルロックを設定しますが、これらのロックを設定する InnoDB レイヤーよりも上位の MySQL レイヤーです。InnoDB は、innodb_table_locks = 1 (デフォルト) かつ autocommit = 0 の場合にテーブルロックを認識し、InnoDB よりも上位の MySQL レイヤーは、行レベルロックを識別します。

  それ以外の場合は、InnoDB の自動デッドロック検出では、このようなテーブルロックが関与するデッドロックを検出できません。また、この場合には上位の MySQL レイヤーは行レベルロックを識別しないため、現在別のセッションが行レベルロックを保持しているテーブル上でテーブルロックを取得できます。ただし、セクション14.2.10「デッドロックの検出とロールバック」で説明したように、これによりトランザクションの完全

性が危険にさらされることはありません。セクション14.6.7「InnoDB テーブル上の制限」も参照してください。

14.2.9 暗黙的なトランザクションコミットとロールバック

MySQL は、デフォルトで新しい接続のセッション開始時に自動コミットモードが有効になります。そのため、各 SQL ステートメントからエラーが返されなかった場合に、MySQL はそのステートメントのあとでコミットを実行します。ステートメントからエラーが返された場合、コミットまたはロールバックの動作はそのエラーによって異なります。セクション14.19.4「InnoDB のエラー処理」を参照してください。

自動コミットが無効になっているセッションが、最後のトランザクションを明示的にコミットせずに終了した場合、MySQL はそのトランザクションをロールバックします。

一部のステートメントは、ユーザーがそのステートメントの実行前に COMMIT を実行した場合と同様に、暗黙的にトランザクションを終了します。詳細は、セクション13.3.3「暗黙的なコミットを発生させるステートメント」を参照してください。

14.2.10 デッドロックの検出とロールバック

InnoDB では、自動的にトランザクションのデッドロックが検出され、デッドロックを解除するためにトランザクション (複数の場合あり) がロールバックされます。InnoDB は、小さいトランザクションを選択してロールバックしようと試みます。トランザクションのサイズは、挿入、更新、または削除された行数によって決定されます。

InnoDB は、innodb_table_locks = 1 (デフォルト) かつ autocommit = 0 の場合にテーブルロックを認識し、それよりも上位の MySQL レイヤーは、行レベルロックを識別します。それ以外の場合、InnoDB は、MySQL LOCK TABLES ステートメントで設定されたテーブルロックまたは InnoDB 以外のストレージエンジンで設定されたロックが関連しているデッドロックを検出できません。このような状況を解決するには、innodb_lock_wait_timeout システム変数の値を設定します。

InnoDB でトランザクションの完全なロールバックが実行されると、トランザクションで設定されたすべてのロックが解放されます。ただし、エラーの結果として単一の SQL ステートメントのみがロールバックされると、ステートメントで設定された一部のロックが保持される可能性があります。これが発生する原因は、InnoDB では、どの行がどのステートメントで設定されたのかをあとで確認できないような形式で、行ロックが格納されるためです。

トランザクションで SELECT がストアドファンクションを呼び出し、そのファンクション内のステートメントに失敗した場合は、そのステートメントがロールバックされます。さらに、そのあとで ROLLBACK が実行された場合、トランザクション全体がロールバックされます。

デッドロックを回避するためにデータベース操作を編成する方法については、セクション14.2.11「デッドロックの対処方法」を参照してください。

14.2.11 デッドロックの対処方法

このセクションは、セクション14.2.10「デッドロックの検出とロールバック」に示したデッドロックに関する概念情報に基づいています。ここでは、デッドロックが最小限になるようにデータベース操作を編成する方法、およびアプリケーションで必要となる後続のエラー処理について説明します。

デッドロックは、トランザクションデータベースの古典的な問題ですが、特定のトランザクションをまったく実行できないほど発生頻度が高くなければ、危険ではありません。通常は、デッドロックが発生したためにトランザクションがロールバックされた場合に、それを再発行できる準備が常にできているようにアプリケーションを作成する必要があります。

InnoDB では自動行レベルロックが使用されます。単一の行を挿入または削除するだけのトランザクションの場合でも、デッドロックが発生する可能性があります。その原因は、これらの操作が実際には「原子的」でないためです。これらの操作では自動的に、挿入または削除される行のインデックスレコード (複数の可能性あり) にロックが設定されます。

次の方法を使用すれば、デッドロックに対処し、発生の可能性を減らすことができます。

- いつでも、SHOW ENGINE INNODB STATUS コマンドを発行して、最近のデッドロックの原因を特定してください。これは、デッドロックが回避されるようにアプリケーションを調整する際に役立ちます。
- 頻繁にデッドロックの警告が発生することに懸念がある場合は、innodb_print_all_deadlocks 構成オプションを有効にして、より広範囲にわたるデバッグ情報を収集してください。MySQL のエラーログには、最近のデッドロックだけでなく、各デッドロックに関する情報が記録されます。デバッグが完了したら、このオプションを無効にします。

- デッドロックが原因でトランザクションに失敗した場合に、そのトランザクションを再発行できるように常に準備しておきます。デッドロックは危険ではありません。再度試してください。
- トランザクションが競合する可能性を低くするために、トランザクションのサイズを小さく、期間を短く保ってください。
- トランザクションが競合する可能性を低くするために、関連する一連の変更を行なった直後にトランザクションをコミットしてください。特に、コミットされていないトランザクションを含むインタラクティブな mysql セッションは、長時間開いたままにしないでください。
- ロック読み取り (SELECT ... FOR UPDATE または SELECT ... LOCK IN SHARE MODE) を使用する場合は、READ COMMITTED などの低い分離レベルを使用してみてください。
- トランザクション内の複数のテーブルを変更する場合や、同じテーブル内のさまざまな行のセットを変更する場合は、毎回、これらの操作を一貫性のある順序で実行してください。その結果、トランザクションで明示的に定義されたキューが生成され、デッドロックは発生しません。たとえば、さまざまな場所で同様の INSERT、UPDATE、および DELETE ステートメントのシーケンスを複数回コーディングするのではなく、データベース操作をアプリケーション内の関数に編成したり、ストアドルーチンを呼び出したりします。
- テーブルに適切なインデックスを追加してください。これにより、クエリーでスキャンする必要のあるインデックスレコード数が減少するため、ロックの設定も減少します。MySQL サーバーがクエリーに最適であるとみなすインデックスを特定するために、EXPLAIN SELECT を使用してください。
- ロックの使用を減らしてください。古いスナップショットからのデータを返すために、SELECT を許可する余裕がある場合は、FOR UPDATE または LOCK IN SHARE MODE 句を追加しないでください。同じトランザクション内の各一貫性読み取りでは、独自の新しいスナップショットから読み取られるため、READ COMMITTED 分離レベルを使用することが適切な方法です。
- ほかに方法がなければ、テーブルレベルロックを使用してトランザクションを直列化してください。InnoDB テーブルなどのトランザクションテーブルで LOCK TABLES を使用する正しい方法は、(START TRANSACTION ではなく) SET autocommit = 0 でトランザクションを開始し、そのあと LOCK TABLES を実行し、UNLOCK TABLES を呼び出す前にそのトランザクションを明示的にコミットすることです。たとえば、テーブル t1 に書き込み、テーブル t2 から読み取る必要がある場合は、次のように実行できます。

```
SET autocommit=0;
LOCK TABLES t1 WRITE, t2 READ, ...;
... do something with tables t1 and t2 here ...
COMMIT;
UNLOCK TABLES;
```

  テーブルレベルロックを使用すると、テーブルへの並列更新が抑制されるため、デッドロックが回避されますが、負荷の高いシステムで応答性が低くなるという犠牲が伴います。
- トランザクションを直列化する別の方法は、単一行だけを含む補助「セマフォー」テーブルを作成することです。ほかのテーブルにアクセスする前に、各トランザクションでその行を更新してください。これにより、すべてのトランザクションが直列方式で発生します。直列化ロックは行レベルロックであるため、この場合、InnoDB のインスタントデッドロック検出アルゴリズムも機能することに注意してください。MySQL のテーブルレベルロックを使用してデッドロックを解決するには、タイムアウト方式を使用する必要があります。

14.2.12 InnoDB マルチバージョン

InnoDB はマルチバージョンストレージエンジンです。並列実行やロールバックなどのトランザクション機能をサポートするために、変更された行の古いバージョンに関する情報が保持されます。この情報は、テーブルスペース内にロールバックセグメントと呼ばれるデータ構造で (Oracle では類似したデータ構造のあとに) 格納されます。InnoDB では、トランザクションのロールバックで必要となる取り消し操作を実行するために、ロールバックセグメント内の情報が使用されます。また、この情報は、一貫性読み取りのために行の初期バージョンを構築する際にも使用されます。

マルチバージョン内部の詳細

InnoDB は内部的に、データベース内に格納された各行に 3 つのフィールドを追加します。6 バイトの DB_TRX_ID フィールドは、行を挿入または更新した最後のトランザクションに対して、トランザクション識別子を指示します。また、行内の特別ビットが削除されたとマークするように設定されている場合、削除は内部的に更新として処理されます。各行には、ロールポインタと呼ばれる 7 バイトの DB_ROLL_PTR フィールドも含まれています。ロールポインタは、ロールバックセグメントに書き込まれた Undo ログレコードを示しています。行が更新された場合は、Undo ログレコードに、更新される前の行の内容を再構築するために必要な情報が含まれ

ます。6 バイトの DB_ROW_ID フィールドには、新しい行が挿入されると単調に増加する行 ID が含まれています。InnoDB によって自動生成されたクラスタ化されたインデックスには、行 ID 値が含まれます。それ以外の場合、インデックスに DB_ROW_ID カラムが含まれることはありません。

ロールバックセグメント内の Undo ログは、挿入および更新 Undo ログに分割されます。挿入 Undo ログはトランザクションロールバックでのみ必要であるため、トランザクションのコミット直後に破棄できます。更新 Undo ログも一貫性読み取りで使用されますが、InnoDB によってスナップショットが割り当てられたトランザクションが存在しなくなったあとでのみ破棄できます。更新 Undo ログ内のスナップショット情報は、データベース行の以前のバージョンを構築する際に一貫性読み取りで必要となる可能性があります。

ロールバックセグメントを管理するためのガイドライン

トランザクション (一貫性読み取りのみを発行するトランザクションを含む) を定期的にコミットしてください。それ以外の場合は、InnoDB は更新 Undo ログからデータを破棄できないため、ロールバックセグメントが大きくなり過ぎてテーブルスペースがいっぱいになる可能性があります。

一般に、ロールバックセグメント内の Undo ログレコードの物理的サイズは、それに対応する挿入された行や更新された行よりも小さいです。この情報を使用すると、ロールバックセグメントで必要となる領域を計算できます。

InnoDB マルチバージョンスキームでは、SQL ステートメントで行を削除しても、その行はすぐにデータベースから物理的に削除されません。InnoDB は、削除用に書き込まれた更新 Undo ログレコードが破棄されたときにのみ、対応する行およびそのインデックスレコードを物理的に削除します。このような削除操作はパージと呼ばれ、非常に高速です。通常は、削除が行われなかった SQL ステートメントと同じ時系列順で実行されます。

テーブル内で小さめのバッチの行をほぼ同じ速度で挿入および削除すると、すべての「デッド」行が原因で、パージスレッドが遅延し始め、増加し続ける可能性があります。これにより、すべてにおいてディスクが抑制され、非常に低速になります。このような場合は、新たな行操作を抑制し、innodb_max_purge_lag システム変数を調整することで、より多くのリソースをパージスレッドに割り当てます。詳細は、セクション14.12「InnoDB の起動オプションおよびシステム変数」を参照してください。

14.2.13 InnoDB テーブルおよびインデックスの構造

このセクションでは、InnoDB のテーブル、インデックス、およびこれらに関連付けられたメタデータを物理レベルで表示する方法について説明します。この情報は、主にパフォーマンスチューニングおよびトラブルシューティングに役立ちます。

14.2.13.1 InnoDB テーブル用の .frm ファイルの役割

MySQL では、データベースディレクトリ内の .frm ファイルに、テーブルに関するそのデータディクショナリ情報が格納されます。その他の MySQL ストレージエンジンとは異なり、InnoDB では、テーブルスペース内にある独自の内部データディクショナリのテーブルに関する情報のエンコードも行われます。MySQL でテーブルまたはデータベースが削除されると、1 つ以上の .frm ファイルおよび InnoDB データディクショナリ内の対応するエントリも削除されます。単に .frm ファイルを移動するだけでは、データベース間で InnoDB テーブルを移動できません。

14.2.13.2 クラスタインデックスとセカンダリインデックス

すべての InnoDB テーブルは、行のデータが格納されているクラスタ化されたインデックスと呼ばれる特別なインデックスを持っています。一般に、クラスタ化されたインデックスは主キーのシノニムです。クエリー、挿入、およびその他のデータベース操作で最適なパフォーマンスを実現するには、InnoDB がクラスタ化されたインデックスを使用して、テーブルごとにもっとも一般的な検索と DML 操作を最適化する方法について理解する必要があります。

- テーブル上で PRIMARY KEY を定義すると、InnoDB ではそれがクラスタ化されたインデックスとして使用されます。作成するテーブルごとに主キーを定義します。論理的に一意で、Null 以外のカラムまたはカラムのセットが存在しない場合は、自動的に値が入力される新しい自動インクリメントカラムを追加します。
- テーブルに PRIMARY KEY が定義されていない場合、MySQL はすべてのキーカラムが NOT NULL の UNIQUE インデックスを最初に検索し、InnoDB はそれをクラスタ化されたインデックスとして使用します。
- テーブルに PRIMARY KEY も適切な UNIQUE インデックスも存在しない場合には、InnoDB の内部で、行 ID 値を含む合成カラム上に非表示のクラスタ化されたインデックスが生成されます。そのようなテーブルでは、InnoDB が行に割り当てる ID に基づいて行の順序付けが行われます。行 ID は、新しい行が挿入されると単調に増加する 6 バイトのフィールドです。したがって、行 ID で順序付けられた行が物理的な挿入順になります。

クラスタ化されたインデックスでクエリーを高速にする方法

クラスタ化されたインデックスから行にアクセスすると、インデックス検索がすべての行データを持つページで直接実行されるため、高速になります。多くの場合、テーブルのサイズが大きい場合にクラスタ化されたインデックスアーキテクチャーを使用すれば、インデックスレコードとは別のページに行データを格納するストレージ編成と比べて、ディスク I/O 操作を節約できます。(たとえば MyISAM では、データ行に使用されるファイルとインデックスレコードに使用されるファイルは異なります。)

セカンダリインデックスとクラスタ化されたインデックスとの関係

クラスタ化されたインデックス以外のインデックスは、すべてセカンダリインデックスと呼ばれます。InnoDB では、セカンダリインデックス内の各レコードに、行の主キーカラム、およびセカンダリインデックスに指定されたカラムが含まれます。InnoDB では、クラスタ化されたインデックス内で行を検索する際に、この主キー値が使用されます。

主キーが長くなると、セカンダリインデックスで使用される領域も多くなるため、主キーは短い方が利点があります。

InnoDB のクラスタインデックスおよびセカンダリインデックスの利点を得るためのコーディングのガイドラインについては、セクション8.3.2「主キーの使用」、セクション8.3「最適化とインデックス」、セクション8.5「InnoDB テーブルの最適化」、セクション8.3.2「主キーの使用」を参照してください。

14.2.13.3 FULLTEXT インデックス

テキストベースのカラム (CHAR、VARCHAR、または TEXT カラム) 上に作成されたインデックスのタイプです。これを使用すると、ストップワードとして定義されている任意の単語が省略されることで、これらのカラム内に含まれるデータ上での InnoDB のクエリーおよび DML 操作の速度を上げる際に役立ちます。

FULLTEXT インデックスは、CREATE TABLE ステートメントの一部として定義することも、あとで ALTER TABLE または CREATE INDEX を使用して追加することもできます。

全文検索は、MATCH() ... AGAINST 構文を使用して実行されます。使用法については、セクション12.9「全文検索関数」を参照してください。

全文インデックスの設計

InnoDB の FULLTEXT インデックスでは、「転置インデックス」の設計が使用されています。転置インデックスには、単語のリスト、および単語ごとに、その単語が出現するドキュメントのリストが格納されます。近接検索をサポートするために、単語ごとの位置情報もバイトオフセットとして格納されます。

全文インデックステーブル

次の例に示すように、InnoDB の FULLTEXT インデックスごとに、インデックステーブルのセットが作成されます。

```
CREATE TABLE opening_lines (
id INT UNSIGNED AUTO_INCREMENT NOT NULL PRIMARY KEY,
opening_line TEXT(500),
author VARCHAR(200),
title VARCHAR(200),
FULLTEXT idx (opening_line)
) ENGINE=InnoDB;

mysql> SELECT table_id, name, space from INFORMATION_SCHEMA.INNODB_SYS_TABLES WHERE name LIKE 'test/%';
+----------+--------------------------------------------------+-------+
| table_id | name                                             | space |
+----------+--------------------------------------------------+-------+
|      333 | test/FTS_0000000000000147_00000000000001c9_INDEX_1 |   289 |
|      334 | test/FTS_0000000000000147_00000000000001c9_INDEX_2 |   290 |
|      335 | test/FTS_0000000000000147_00000000000001c9_INDEX_3 |   291 |
|      336 | test/FTS_0000000000000147_00000000000001c9_INDEX_4 |   292 |
|      337 | test/FTS_0000000000000147_00000000000001c9_INDEX_5 |   293 |
|      338 | test/FTS_0000000000000147_00000000000001c9_INDEX_6 |   294 |
|      330 | test/FTS_0000000000000147_BEING_DELETED          |   286 |
|      331 | test/FTS_0000000000000147_BEING_DELETED_CACHE    |   287 |
|      332 | test/FTS_0000000000000147_CONFIG                 |   288 |
|      328 | test/FTS_0000000000000147_DELETED                |   284 |
|      329 | test/FTS_0000000000000147_DELETED_CACHE          |   285 |
|      327 | test/opening_lines                               |   283 |
+----------+--------------------------------------------------+-------+
12 rows in set (0.02 sec)
```

最初の 6 つのテーブルは転置インデックスを表し、「近接検索インデックステーブル」と呼ばれます。受信ドキュメントがトークン化されると、各単語が位置情報およびドキュメント ID (DOC_ID) とともに、インデックステーブルに挿入されます。単語は完全にソートされてから、単語の最初の文字の文字セット重みに基づいて、6 つのインデックステーブル間でパーティション化されます。

転置インデックスは、インデックスの並列作成をサポートするために、6 つの補助インデックステーブルに「パーティション化」されます。デフォルトでは、2 つのスレッドを使用して、単語および関連するデータのトークン化、ソート、およびインデックステーブルへの挿入が実行されます。スレッドの数は、innodb_ft_sort_pll_degree オプションを使用することで構成可能です。大きなテーブル上に FULLTEXT インデックスを作成する際には、スレッドの数を多くすることを検討してください。

補助インデックステーブル名の前には FTS_、後ろには INDEX_* が付けられます。各インデックステーブルは、インデックス付きのテーブルの table_id と一致するインデックステーブル名に含まれる 16 進値によって、インデックス付きのテーブルに関連付けられます。たとえば、test/opening_lines テーブルの table_id は 327 (16 進値は 0x147) です。前述の例で示したように、16 進値の 「147」 は、test/opening_lines テーブルに関連付けられたインデックステーブルの名前に表示されます。

補助インデックス名に表示されるもう 1 つの 16 進値は、FULLTEXT インデックスの index_id です。たとえば、補助テーブル名 test/FTS_0000000000000147_00000000000001c9_INDEX_1 では、16 進値 1c9 の 10 進値は 457 です。opening_lines テーブルで定義されたインデックス (idx) は、INFORMATION_SCHEMA.INNODB_SYS_INDEXES テーブルでこの値 (457) に対してクエリーを実行することで識別できます。

```
mysql> SELECT index_id, name, table_id, space from INFORMATION_SCHEMA.INNODB_SYS_INDEXES WHERE index_id=457;
+----------+------+----------+-------+
| index_id | name | table_id | space |
+----------+------+----------+-------+
|      457 | idx  |      327 |   283 |
+----------+------+----------+-------+
1 row in set (0.00 sec)
```

innodb_file_per_table を有効にすると、インデックステーブルが独自のテーブルスペースに格納されます。innodb_file_per_table を無効にすると、インデックステーブルが InnoDB のシステムテーブルスペース (スペース 0) に格納されます。

> **注記**
>
> MySQL 5.6.5 で導入されたバグが原因で、innodb_file_per_table を有効にしても、インデックステーブルは InnoDB のシステムテーブルスペース (スペース 0) に作成されます。このバグは、MySQL 5.6.20 および MySQL 5.7.5 で修正されました (Bug#18635485)。

前述の例で示したその他のインデックステーブルは、FULLTEXT インデックスの削除処理および内部状態の格納で使用されます。

- FTS_*_DELETED および FTS_*_DELETED_CACHE: 削除されるが、データはまだ全文インデックスから削除されないドキュメントのドキュメント ID (DOC_ID) が含まれます。FTS_*_DELETED_CACHE は、FTS_*_DELETED テーブルのインメモリーバージョンです。
- FTS_*_BEING_DELETED および FTS_*_BEING_DELETED_CACHE: 削除され、現在データが全文インデックスから削除中であるドキュメントのドキュメント ID (DOC_ID) が含まれます。FTS_*_BEING_DELETED_CACHE テーブルは、FTS_*_BEING_DELETED テーブルのインメモリーバージョンです。
- FTS_*_CONFIG: FULLTEXT インデックスの内部状態に関する情報が格納されます。もっとも重要な点は、解析され、ディスクにフラッシュされたドキュメントを識別する FTS_SYNCED_DOC_ID が格納されることです。クラッシュリカバリの場合、ドキュメントを再解析し、FULLTEXT インデックスキャッシュに追加し直すことができるように、ディスクにフラッシュされていないドキュメントを識別する際に、FTS_SYNCED_DOC_ID 値が使用されます。このテーブル内のデータを表示するには、INFORMATION_SCHEMA.INNODB_FT_CONFIG テーブルでクエリーを実行します。

全文インデックスキャッシュ

ドキュメントが挿入されると、トークン化され、各単語および関連付けられたデータが FULLTEXT インデックスに挿入されます。このプロセスが実行されると、小さなドキュメントの場合でも、補助インデックステーブルへの多数の小規模な挿入が発生します。これにより、競合の発生時に、これらのテーブルへの並列アクセスが発生する可能性があります。この問題を回避するために、InnoDB では、最近挿入された行に対するインデックステーブルの挿入を一時的にキャッシュに入れるために、FULLTEXT インデックスキャッシュが使用されます。この「インメモリー」キャッシュの構造では、キャッシュがいっぱいに

なるまで挿入が保持され、そのあと、ディスク (補助インデックステーブル) にバッチフラッシュされます。INFORMATION_SCHEMA.INNODB_FT_INDEX_CACHE テーブルでクエリーを実行すると、最近挿入された行のトークン化されたデータを表示できます。

キャッシュおよびバッチフラッシュの動作によって、補助インデックステーブルへの頻繁な更新が回避されますが、負荷の高い挿入時および更新時に並列アクセスの問題が発生する可能性があります。また、バッチ技術を使用すると、同じ単語への挿入が複数回発生することも回避され、重複エントリも最小限になります。各単語を個別にフラッシュする代わりに、同じ単語の挿入がマージされ、単一のエントリとしてディスクにフラッシュされるため、補助インデックステーブルのサイズをできるかぎり小さく保ちながら、挿入の効率性が改善されます。

全文インデックスキャッシュのサイズを (テーブルごとに) 構成するには、innodb_ft_cache_size 変数が使用されます。これにより、全文インデックスキャッシュがフラッシュされる頻度が影響を受けます。特定のインスタンスで innodb_ft_total_cache_size オプションを使用すれば、すべてのテーブルに対応したグローバルな全文インデックスキャッシュのサイズ制限を定義することもできます。

全文インデックスキャッシュには、補助インデックステーブルと同じ情報が格納されます。ただし、全文インデックスキャッシュでは、最近挿入された行のトークン化されたデータのみがキャッシュに入れられます。すでにディスク (全文補助テーブル) にフラッシュされているデータは、クエリー時に全文インデックスキャッシュに戻りません。補助インデックステーブル内のデータは、直接クエリーが実行されます。補助インデックステーブルからの結果は、全文インデックスキャッシュからの結果とマージされてから返されます。

InnoDB の全文ドキュメント ID および FTS_DOC_ID カラム

InnoDB では、全文インデックス内の単語をその単語が出現するドキュメントレコードとマップする際に、ドキュメント ID (DOC_ID) と呼ばれる一意のドキュメント識別子が使用されます。このマッピングには、インデックス付きテーブル上の FTS_DOC_ID カラムが必要です。FTS_DOC_ID カラムが定義されていない場合は、全文インデックスの作成時に、InnoDB によって自動的に非表示の FTS_DOC_ID カラムが追加されます。次の例で、この動作を実演します。

次のテーブル定義には、FTS_DOC_ID カラムが含まれていません。

```
CREATE TABLE opening_lines (
id INT UNSIGNED AUTO_INCREMENT NOT NULL PRIMARY KEY,
opening_line TEXT(500),
author VARCHAR(200),
title VARCHAR(200)
) ENGINE=InnoDB;
```

CREATE FULLTEXT INDEX 構文を使用して、テーブル上に全文インデックスを作成すると、FTS_DOC_ID カラムが追加されるように InnoDB がテーブルを再構築してしていることをレポートする警告が返されます。

```
mysql> CREATE FULLTEXT INDEX idx ON opening_lines(opening_line);
Query OK, 0 rows affected, 1 warning (0.19 sec)
Records: 0  Duplicates: 0  Warnings: 1

mysql> SHOW WARNINGS;
+---------+------+----------------------------------------------------+
| Level   | Code | Message                                            |
+---------+------+----------------------------------------------------+
| Warning |  124 | InnoDB rebuilding table to add column FTS_DOC_ID |
+---------+------+----------------------------------------------------+
1 row in set (0.00 sec)
```

ALTER TABLE を使用して、FTS_DOC_ID カラムが存在しないテーブルに全文インデックスを追加するときにも、同じ警告が返されます。CREATE TABLE の実行時に全文インデックスを作成する場合に、FTS_DOC_ID カラムを定義しないと、InnoDB によって警告なしで、非表示の FTS_DOC_ID カラムが追加されます。

CREATE TABLE の実行時に FTS_DOC_ID カラムを定義するには、すでにデータがロードされているテーブル上に全文インデックスを作成する必要があります。データをロードする前に、テーブル上に FTS_DOC_ID カラムが定義されている場合は、新しいカラムが追加されるようにテーブルおよびそのインデックスを再構築する必要がありません。CREATE FULLTEXT INDEX のパフォーマンスに関心がない場合は、InnoDB で自動的に作成されるように、FTS_DOC_ID カラムを除外します。InnoDB によって、FTS_DOC_ID_INDEX という名前の FTS_DOC_ID カラム上に、一意のインデックスとともに非表示の FTS_DOC_ID が作成されます。独自の FTS_DOC_ID カラムを作成する場合は、次の例で示すように、カラムを BIGINT UNSIGNED NOT NULL として定義し、FTS_DOC_ID (すべて大文字) という名前を付けます。

> **注記**
>
> AUTO_INCREMENT として FTS_DOC_ID カラムを定義する必要がありませんが、AUTO_INCREMENT を使用した方が簡単にデータをロードできます。

```
CREATE TABLE opening_lines (
FTS_DOC_ID BIGINT UNSIGNED AUTO_INCREMENT NOT NULL,
opening_line TEXT(500),
author VARCHAR(200),
title VARCHAR(200)
) ENGINE=InnoDB;
```

FTS_DOC_ID カラムをユーザー自身で定義するように決定した場合は、空の値や重複する値が回避されるようにカラムを管理することがユーザーの責任となります。FTS_DOC_ID 値は再使用できません。つまり、FTS_DOC_ID 値は増加し続けます。

オプションで、FTS_DOC_ID カラム上に必要な一意の FTS_DOC_ID_INDEX (すべて大文字) を作成することもできます。

```
CREATE UNIQUE INDEX FTS_DOC_ID_INDEX on opening_lines(FTS_DOC_ID);
```

FTS_DOC_ID_INDEX を作成しない場合は、InnoDB によって自動的に作成されます。

InnoDB による全文インデックスの削除処理

全文インデックスカラムが含まれるレコードを削除すると、補助インデックステーブルへの多数の小規模な削除が発生します。これにより、競合の発生時に、これらのテーブルへの並列アクセスが発生する可能性があります。この問題を回避するために、インデックス付きのテーブルからレコードが削除されるたびに、削除されたドキュメントのドキュメント ID (DOC_ID) が特別な 「DELETED」 テーブルに記録され、インデックス付きのレコードが全文インデックスに残ります。クエリーの結果が返される前に、「DELETED」 テーブル内の情報を使用して、削除されたドキュメント ID が取り除かれます。この設計の利点は、削除が高速で、低負荷であることです。欠点は、レコードの削除後に、すぐにインデックスのサイズが削減されないことです。削除したエントリの全文インデックスエントリを削除するには、innodb_optimize_fulltext_only=ON を使用してインデックス付きのテーブル上で OPTIMIZE TABLE を実行して、全文インデックスを再構築します。詳細は、InnoDB 全文インデックスの最適化を参照してください。

InnoDB による全文インデックスのトランザクション処理

InnoDB の FULLTEXT インデックスには、そのキャッシュおよびバッチ処理の動作のために、特別なトランザクション処理の特性が備わっています。特に、FULLTEXT インデックス上の更新および挿入は、トランザクションのコミット時に処理されます。つまり、FULLTEXT 検索では、コミットされたデータのみを表示できます。次の例で、この動作を実演します。FULLTEXT 検索では、挿入された行がコミットされたあとにはじめて、結果が返されます。

```
mysql> CREATE TABLE opening_lines (
id INT UNSIGNED AUTO_INCREMENT NOT NULL PRIMARY KEY,
opening_line TEXT(500),
author VARCHAR(200),
title VARCHAR(200),
FULLTEXT idx (opening_line)
) ENGINE=InnoDB;

mysql> BEGIN;
Query OK, 0 rows affected (0.00 sec)

mysql> INSERT INTO opening_lines(opening_line,author,title) VALUES
('Call me Ishmael.','Herman Melville','Moby-Dick'),
('A screaming comes across the sky.','Thomas Pynchon','Gravity\'s Rainbow'),
('I am an invisible man.','Ralph Ellison','Invisible Man'),
('Where now? Who now? When now?','Samuel Beckett','The Unnamable'),
('It was love at first sight.','Joseph Heller','Catch-22'),
('All this happened, more or less.','Kurt Vonnegut','Slaughterhouse-Five'),
('Mrs. Dalloway said she would buy the flowers herself.','Virginia Woolf','Mrs. Dalloway'),
('It was a pleasure to burn.','Ray Bradbury','Fahrenheit 451');
Query OK, 8 rows affected (0.00 sec)
Records: 8  Duplicates: 0  Warnings: 0

mysql> SELECT COUNT(*) FROM opening_lines WHERE MATCH(opening_line) AGAINST('Ishmael');
+----------+
| COUNT(*) |
+----------+
|        0 |
+----------+
1 row in set (0.00 sec)

mysql> COMMIT;
Query OK, 0 rows affected (0.00 sec)

mysql> SELECT COUNT(*) FROM opening_lines WHERE MATCH(opening_line) AGAINST('Ishmael');
+----------+
```

```
| COUNT(*) |
+----------+
|        1 |
+----------+
1 row in set (0.00 sec)
```

InnoDB による全文インデックスのモニター

次の INFORMATION_SCHEMA テーブルでクエリーを実行すると、InnoDB の FULLTEXT インデックスの特別なテキスト処理の側面をモニターおよび調査できます。

- INNODB_FT_CONFIG
- INNODB_FT_INDEX_TABLE
- INNODB_FT_INDEX_CACHE
- INNODB_FT_DEFAULT_STOPWORD
- INNODB_FT_DELETED
- INNODB_FT_BEING_DELETED

INNODB_SYS_INDEXES および INNODB_SYS_TABLES でクエリーを実行すると、FULLTEXT インデックスおよびテーブルに関する基本情報を表示することもできます。

これらのテーブルについての詳細は、INFORMATION_SCHEMA のドキュメントを参照してください。

14.2.13.4 InnoDB インデックスの物理構造

すべての InnoDB インデックスは、インデックスレコードがツリーのリーフページ内に格納される B ツリーです。インデックスページのデフォルトサイズは 16K バイトです。新しいレコードが挿入されると、InnoDB はページの 1/16 を、将来のインデックスレコードの挿入や更新に備えて空けようとします。

インデックスレコードが順次 (昇順または降順) に挿入されると、インデックスページの約 15/16 までがいっぱいになります。レコードがランダムに挿入された場合は、ページの 1/2 から 15/16 までがいっぱいになります。インデックスページのフィルファクタが 1/2 を下回ると、InnoDB はページを解放するために、インデックスツリーを縮小しようとします。

> **注記**
>
> インスタンスを作成する前に、innodb_page_size 構成オプションを設定すると、MySQL インスタンス内のすべての InnoDB テーブルスペースのページサイズを指定できます。一度 MySQL インスタンスのページサイズが設定されたら、変更できません。サポートされるサイズは、オプションの値 16k、8k、および 4k に対応する 16K バイト、8K バイト、および 4K バイトです。
>
> 特定の InnoDB ページサイズを使用している MySQL インスタンスは、別のページサイズを使用するインスタンスのデータファイルやログファイルを使用できません。

14.2.13.5 挿入バッファー

多くの場合、データベースアプリケーションでは、新しい行が主キーの昇順で挿入されます。この場合、クラスタ化されたインデックスの順序が主キーと同じであるというレイアウトのために、InnoDB テーブルへの挿入時に、ディスクからのランダムな読み取りが必要ありません。

その一方で、通常、セカンダリインデックスは一意ではなく、セカンダリインデックスへの挿入は比較的ランダムな順序で発生します。同様に、削除および更新によって、セカンダリインデックス内の隣接しないデータページが影響を受ける可能性があります。このため、InnoDB で特別なメカニズムが使用されることなく、多数のランダムなディスク I/O が発生します。

インデックスレコードを挿入したり、削除対象のマークを付けたり、一意でないセカンダリインデックスから削除したりすると、InnoDB によって、セカンダリインデックスがバッファープール内に存在するかどうかがチェックされます。これに該当する場合は、InnoDB によって変更が直接インデックスページに適用されます。バッファープール内にインデックスページが見つからない場合は、InnoDB によって、挿入バッファーと呼ばれる特別な構造に変更が記録されます。挿入バッファーは、バッファープール内に完全に収容されるようにサイズが小さく保たれるため、変更を非常にすばやく適用できます。このプロセスは、変更バッファーと呼ばれます。(以前は、挿入にのみ適用されていたため、挿入バッファーと呼ばれていました。データ構造は、引き続き挿入バッファーと呼ばれています。)

挿入バッファーをフラッシュするためのディスク I/O

挿入バッファーは、データベース内のセカンダリインデックスツリーに定期的にマージされます。多くの場合は、複数の変更をインデックスツリーの同じページにマージできるため、ディスク I/O 操作を節約できます。挿入バッファーによって、テーブルへの挿入速度が最大 15 倍に上昇すると測定されています。

挿入バッファーのマージは、トランザクションがコミットされたあとに、発生し続ける可能性があります。実際、これはサーバーがシャットダウンし、再起動したあとまで発生し続ける可能性があります (セクション 14.19.2「InnoDB のリカバリの強制的な実行」を参照してください)。

多くのセカンダリインデックスが更新される必要があり、多くの行が挿入されたときは、挿入バッファーのマージに何時間もかかる可能性があります。この期間は、ディスク I/O が増加します。これにより、ディスクに負荷がかかるクエリーは大幅に低速になります。もう 1 つの重要なバックグラウンド I/O 操作は、パージスレッドです (セクション14.2.12「InnoDB マルチバージョン」を参照してください)。

14.2.13.6 適応型ハッシュインデックス

適応型ハッシュインデックス (AHI) と呼ばれる機能を使用すると、トランザクションの機能や信頼性を犠牲にせずに、バッファープールのワークロードと十分なメモリーが適切に組み合わされたシステム上でインメモリーデータベースと同様に InnoDB を実行できます。この機能はサーバーの起動時に、innodb_adaptive_hash_index オプションを使用すると有効になり、--skip-innodb_adaptive_hash_index を使用すると無効になります。

監視対象の検索パターンに基づいて、MySQL はインデックスキーのプリフィクスを使用して、ハッシュインデックスを構築します。キーのプリフィクスは任意の長さにすることができますが、ハッシュインデックスには B ツリー内の値の一部しか表示されない可能性があります。ハッシュインデックスは、頻繁にアクセスされるインデックスのページに対する要求に応じて構築されます。

テーブルがメインメモリー内にほぼ完全に収容されている場合は、任意の要素の直接検索を有効にし、インデックス値をポインタの一種に変換すると、ハッシュインデックスを使用してクエリーを高速にすることができます。InnoDB には、インデックスの検索をモニターするメカニズムが備わっています。ハッシュインデックスの構築がクエリーにとって有益であると InnoDB が判断した場合は、自動的にそのインデックスが構築されます。

一部のワークロードでは、ハッシュインデックスの検索による高速化の方が、インデックスの検索をモニターしたり、ハッシュインデックスの構造を保持したりする追加の作業よりも重要です。複数の並列結合などの負荷の高いワークロードでは、適応型ハッシュインデックスへのアクセスを保護する読み取り/書き込みロックが競合の原因となる可能性があります。LIKE 演算子と % ワイルドカードを使用したクエリーも、AHI を使用する利点が得られない傾向があります。適応型ハッシュインデックスが必要のないワークロードでは、これをオフにすれば、不要なパフォーマンスのオーバーヘッドが削減されます。この機能が特定のシステムに適しているかどうかを事前に予測することは困難であるため、現実的なワークロードを使用して、有効にした場合と無効にした場合の両方でベンチマークを実行することを検討してください。MySQL 5.6 以上ではアーキテクチャーが変更され、以前のリリースよりも、適応型ハッシュインデックスを無効にすることに適したワークロードが多くなりました。ただし、デフォルトでは引き続き有効になっています。

常に、ハッシュインデックスはテーブル上の既存の B ツリーインデックスに基づいて構築されます。InnoDB は、B ツリーインデックスに対して InnoDB が検出した検索パターンに応じて、任意の長さの B ツリーに定義されたキーのプリフィクスに、ハッシュインデックスを構築できます。ハッシュインデックスは、頻繁にアクセスされるインデックスのページのみが対象となるように、部分的に構築できます。

適応型ハッシュインデックスの使用、およびこれを SHOW ENGINE INNODB STATUS コマンドの出力の SEMAPHORES セクションで使用した場合の競合をモニターできます。btr0sea.c で作成された RW ラッチ上で待機しているスレッドが多く見られる場合は、適応型ハッシュインデックスを無効にすると役立つ可能性があります。

ハッシュインデックスのパフォーマンス特性についての詳細は、セクション8.3.8「B ツリーインデックスとハッシュインデックスの比較」を参照してください。

14.2.13.7 物理的な行構造

InnoDB テーブルの物理的な行構造は、テーブル作成時に指定された行フォーマットによって異なります。InnoDB のデフォルトでは、Antelope ファイル形式とその COMPACT 行フォーマットが使用されます。REDUNDANT 形式は、古い MySQL バージョンとの互換性を保つために使用可能です。セクション 14.9「InnoDB の行ストレージと行フォーマット」およびセクション14.7「InnoDB 圧縮テーブル」で説明するように、innodb_file_per_table 設定を有効にすると、新しい Barracuda ファイル形式を DYNAMIC および COMPRESSED 行フォーマットとともに使用することもできます。

InnoDB テーブルの行フォーマットをチェックするには、SHOW TABLE STATUS を使用します。例:

```
mysql> SHOW TABLE STATUS IN test1\G
```

```
*************************** 1. row ***************************
           Name: t1
         Engine: InnoDB
        Version: 10
     Row_format: Compact
           Rows: 0
 Avg_row_length: 0
    Data_length: 16384
Max_data_length: 0
   Index_length: 16384
      Data_free: 0
 Auto_increment: 1
    Create_time: 2014-10-31 16:02:01
    Update_time: NULL
     Check_time: NULL
      Collation: latin1_swedish_ci
       Checksum: NULL
 Create_options:
        Comment:
1 row in set (0.00 sec)
```

INFORMATION_SCHEMA.INNODB_SYS_TABLES でクエリーを実行することで、InnoDB テーブルの行フォーマットをチェックすることもできます。

```
mysql> SELECT NAME, ROW_FORMAT FROM INFORMATION_SCHEMA.INNODB_SYS_TABLES WHERE NAME='test1/t1';
+----------+------------+
| NAME     | ROW_FORMAT |
+----------+------------+
| test1/t1 | Compact    |
+----------+------------+
```

COMPACT 行フォーマットを使用すると、行のストレージ領域が約 20% 減少しますが、一部の操作では CPU 使用率が高くなります。ワークロードが、キャッシュヒット率とディスク速度によって制限される通常のワークロードであれば、COMPACT 形式が高速になる可能性があります。ワークロードが CPU 速度によって制限されるまれなケースでは、COMPACT 形式が低速になる可能性があります。

REDUNDANT 行フォーマットを使用する InnoDB テーブル内の行には、次のような特性があります。

- 各インデックスレコードには、6 バイトのヘッダーが含まれています。このヘッダーは、連続するレコードをリンクするために使用されます。また、行レベルロックでも使用されます。
- クラスタ化されたインデックス内のレコードには、すべてのユーザー定義カラムのフィールドが含まれます。さらに、6 バイトのトランザクション ID フィールドと 7 バイトのロールポインタフィールドも含まれています。
- テーブルに主キーが定義されなかった場合は、各クラスタ化されたインデックスレコードに 6 バイトの行 ID フィールドも含まれています。
- 各セカンダリインデックスレコードには、セカンダリインデックス内に存在しないクラスタ化されたインデックスキーに定義されたすべての主キーフィールドも含まれています。
- レコードには、そのレコードの各フィールドへのポインタが含まれます。レコード内のフィールド長の合計が 128 バイト未満の場合はポインタが 1 バイト、128 バイト以上の場合はポインタが 2 バイトになります。これらのポインタの配列はレコードディレクトリと呼ばれます。これらのポインタが示す領域は、レコードのデータ部分と呼ばれます。
- InnoDB には内部的に、固定長形式の CHAR(10) などの固定長文字カラムが格納されます。InnoDB は、VARCHAR カラム内の末尾の空白を切り捨てません。
- SQL の NULL 値では、レコードディレクトリに 1 バイトまたは 2 バイトが予約されます。それ以外に、SQL の NULL 値では、可変長カラム内に格納されるとレコードのデータ部分にゼロバイトが予約されます。固定長カラムでは、レコードのデータ部分内にカラムの固定長が予約されます。NULL 値用に固定領域を予約すると、インデックスページの断片化が発生せずに、カラムを NULL から非 NULL 値に更新できます。

COMPACT 行フォーマットを使用する InnoDB テーブル内の行には、次のような特性があります。

- 各インデックスレコードには、前に可変長ヘッダーが付く可能性のある 5 バイトのヘッダーが含まれています。このヘッダーは、連続するレコードをリンクするために使用されます。また、行レベルロックでも使用されます。
- レコードヘッダーの可変長部分には、NULL カラムを示すビットベクトルが含まれています。NULL にすることができるインデックス内のカラム数が N である場合は、ビットベクトルで CEILING(N/8) バイトが占有されます。(たとえば、NULL にすることができるカラムが 9 から 15 までの任意の数だけ存在する場合は、ビットベクトルで 2 バイトが使用されます。)このベクトル内のビット以外の領域は、NULL のカラムで占有されま

せん。ヘッダーの可変長部分には、可変長カラムの長さも含まれています。各長さは、カラムの最大長に応じて、1 バイトと 2 バイトのいずれかになります。インデックス内のすべてのカラムが NOT NULL でかつ固定長である場合、レコードヘッダーには可変長部分が含まれません。

- 非 NULL 可変長フィールドごとに、レコードヘッダーに 1 バイトまたは 2 バイトのカラム長が含まれます。カラムの一部が外部のオーバーフローページに格納される場合や、最大長が 255 バイトを超え、実際の長さが 127 バイトを超える場合にのみ、2 バイトが必要になります。カラムが外部に格納された場合、2 バイトの長さは、内部に格納された部分の長さに、外部に格納された部分への 20 バイトのポインタを加えた長さを示します。内部の部分は 768 バイトであるため、長さは 768+20 になります。20 バイトのポインタには、そのカラムの実際の長さが格納されます。
- レコードヘッダーのあとに、非 NULL カラムのデータ内容が続きます。
- クラスタ化されたインデックス内のレコードには、すべてのユーザー定義カラムのフィールドが含まれます。さらに、6 バイトのトランザクション ID フィールドと 7 バイトのロールポインタフィールドも含まれています。
- テーブルに主キーが定義されなかった場合は、各クラスタ化されたインデックスレコードに 6 バイトの行 ID フィールドも含まれています。
- 各セカンダリインデックスレコードには、セカンダリインデックス内に存在しないクラスタ化されたインデックスキーに定義されたすべての主キーフィールドも含まれています。これらの主キーフィールドのいずれかが可変長である場合、各セカンダリインデックスのレコードヘッダーには、そのセカンダリインデックスが固定長カラムで定義されている場合でも、それらのフィールドの長さを記録するための可変長部分が含まれます。
- InnoDB には内部的に、固定長形式の CHAR(10) などの固定長で固定幅の文字カラムが格納されます。InnoDB は、VARCHAR カラム内の末尾の空白を切り捨てません。
- InnoDB は、後続領域を切り取ることで内部的に UTF-8 CHAR(N) カラムを N バイトで格納しようとします。(REDUNDANT 行フォーマットでは、このようなカラムで 3 × N バイトが占有されます。)多くの場合は、最小領域 N を予約すると、インデックスページの断片化が発生せずに、カラムの更新を正常に実行できます。

14.3 InnoDB の構成

InnoDB 構成について最初に決定することは、InnoDB データファイルのレイアウト方法、および InnoDB ストレージエンジン用に割り当てるメモリーの量です。これらの選択を記録するには、起動時に MySQL で読み取られる構成ファイル内に記録するか、または起動スクリプトのコマンド行オプションとして指定します。オプション、説明、および許可されるパラメータ値の完全なリストについては、セクション14.12「InnoDB の起動オプションおよびシステム変数」を参照してください。

InnoDB テーブルスペースおよびログファイルの概要

InnoDB ストレージエンジンで管理されている 2 つの重要なディスクベースのリソースは、そのテーブルスペースデータファイルとログファイルです。InnoDB 構成オプションを指定しない場合は、MySQL によって MySQL データディレクトリ内に、わずかに 12M バイトよりも大きい ibdata1 という名前の自動拡張データファイルと、ib_logfile0 および ib_logfile1 という名前の 2 つのログファイルが作成されます。これらのサイズは、innodb_log_file_size システム変数のサイズで指定されます。適切なパフォーマンスを実現するには、次の例で説明するように、InnoDB パラメータを明示的に指定します。当然、ハードウェアおよび要件に合うように設定を編集します。

ここで示す例は代表的なものです。InnoDB に関連する構成パラメータに関する追加情報については、セクション14.12「InnoDB の起動オプションおよびシステム変数」を参照してください。

ストレージデバイスに関する考慮事項

場合によっては、一部のデータが同じ物理ディスク上に配置されてない場合に、データベースのパフォーマンスが改善されることがあります。非常に多くの場合、ログファイルをデータとは別のディスク上に配置すると、パフォーマンスの改善に役立ちます。次の例で、この方法を示します。2 つのデータファイルが別々のディスク上に配置され、ログファイルが 3 台目のディスク上に配置されています。InnoDB では、1 番目のデータファイルから順番にテーブルスペースに収容されます。InnoDB データファイルとして、RAW ディスクパーティション (RAW デバイス) を使用することもできます。これにより、I/O の速度が上がる可能性があります。セクション14.5.8「共有テーブルスペースでの RAW ディスクパーティションの使用」を参照してください。

> **注意**
>
> InnoDB はトランザクションセーフな (ACID に準拠した) MySQL 用のストレージエンジンであり、ユーザーデータを保護するためのコミット、ロールバック、およびクラッ

シュリカバリ機能を備えています。ただし、ベースとなるオペレーティングシステムやハードウェアが公表どおりに機能しない場合は、実行できません。多くのオペレーティングシステムやディスクサブシステムでは、パフォーマンスを改善するために書き込み操作が遅延したり、再指示されたりする可能性があります。一部のオペレーティングシステムでは、まさに fsync() システムコールは、ファイルのすべての未書き込みデータがフラッシュされるまで待機するべきですが、実際には、データが安定したストレージにフラッシュされる前に返される可能性があります。このため、オペレーティングシステムのクラッシュや停電によって最近コミットされたデータが破損したり、さらに最悪の場合、書き込み操作が再指示されたためにデータベースが破損したりすることもあります。データの完全性が重要である場合は、本番環境で何かを使用する前に、何らかの形で「電源プラグを抜く」テストを実行してください。OS X 10.3 以降の InnoDB では、特別な fcntl() ファイルフラッシュ方式が使用されます。Linux では、ライトバックキャッシュを無効にすることが推奨されています。

ATA/SATA ディスクドライブ上で hdparm -W0 /dev/hda のようなコマンドを使用すると、ライトバックキャッシュを無効にできる場合があります。一部のドライブやディスクコントローラでは、ライトバックキャッシュを無効にできない可能性があることに注意してください。

ユーザーを保護する InnoDB のリカバリ機能に関しては、InnoDB では二重書き込みバッファーと呼ばれる構造に関連したファイルフラッシュ技術が使用されています。これは、デフォルトで有効になっています (innodb_doublewrite=ON)。二重書き込みバッファーを使用すると、クラッシュや停電のあとのリカバリの安全性が高まるだけでなく、fsync() 操作の必要性が減るため、ほとんどの種類の Unix でパフォーマンスが向上します。データの完全性または エラーの可能性に関心がある場合は、innodb_doublewrite オプションを有効のままにすることが推奨されています。二重書き込みバッファーの追加情報については、セクション14.10「InnoDB のディスク I/O とファイル領域管理」を参照してください。

注意

データの信頼性が考慮事項となっている場合は、NFS ボリューム上でデータファイルやログファイルが使用されるように InnoDB を構成しないでください。発生する可能性のある問題は、OS および NFS のバージョンによって異なります。これらの問題には、競合する書き込みからの保護が不足しているなどの問題や、最大ファイルサイズ上の制限などが含まれます。

InnoDB テーブルスペースファイルの場所とサイズの指定

InnoDB テーブルスペースファイルを設定するには、my.cnf オプションファイルの [mysqld] セクションで innodb_data_file_path オプションを使用します。Windows では、代わりに my.ini を使用できます。innodb_data_file_path の値は、1 つ以上のデータファイルのリストで指定するようにしてください。複数のデータファイル名を指定する場合は、セミコロン文字 (「;」) で区切ってください。

```
innodb_data_file_path=datafile_spec1[;datafile_spec2]...
```

たとえば、次の設定では、最小サイズのシステムテーブルスペースが明示的に作成されます。

```
[mysqld]
innodb_data_file_path=ibdata1:12M:autoextend
```

この設定では、ibdata1 という名前の 12M バイトの自動拡張データファイルが 1 つ構成されます。ファイルの場所が指定されていないため、InnoDB によって、デフォルトで MySQL データディレクトリ内に作成されます。

K バイト、M バイト、または G バイトの単位を指定するために、サイズは K、M、または G のサフィクス文字を使用して指定されます。

データディレクトリ内にある ibdata1 という名前の 50M バイトの固定サイズデータファイルと、ibdata2 という名前の 50M バイトの自動拡張ファイルを含むテーブルスペースは、次のように構成できます。

```
[mysqld]
innodb_data_file_path=ibdata1:50M;ibdata2:50M:autoextend
```

データファイルを指定するための完全な構文には、ファイル名、そのサイズ、および複数のオプション属性が含まれています。

```
file_name:file_size[:autoextend[:max:max_file_size]]
```

autoextend および max 属性は、innodb_data_file_path 行内の最後のデータファイルでのみ使用できます。

最後のデータファイルに autoextend オプションを指定すると、テーブルスペースに空き領域がなくなった場合に、InnoDB はデータファイルを拡張します。デフォルトでは、一度に 8M バイトずつ増分されます。増分を変更するには、innodb_autoextend_increment システム変数を変更します。

ディスクがいっぱいになったら、別のディスク上に別のデータファイルを追加するといいでしょう。テーブルスペースを再構成する手順については、セクション14.5.7「InnoDB ログファイルの数またはサイズの変更、および InnoDB テーブルスペースのサイズの変更」を参照してください。

InnoDB ではファイルシステムの最大ファイルサイズが認識されないため、最大ファイルサイズが 2G バイトのような小さい値になっているファイルシステムでは注意してください。自動拡張データファイルの最大サイズを指定するには、autoextend 属性のあとに max 属性を使用してください。最大サイズを超えると致命的なエラーが発生し、クラッシュする可能性もあるため、ディスクの使用率を制約することが非常に重要である場合に限り、max 属性を使用してください。次の構成では、ibdata1 が最大で 500M バイトの制限まで増大することが許可されます。

```
[mysqld]
innodb_data_file_path=ibdata1:12M:autoextend:max:500M
```

InnoDB は、デフォルトで MySQL データディレクトリ内にテーブルスペースファイルを作成します。場所を明示的に指定するには、innodb_data_home_dir オプションを使用します。たとえば、ibdata1 および ibdata2 という名前の 2 つのファイルを使用するが、/ibdata ディレクトリ内に作成するには、次のように InnoDB を構成します。

```
[mysqld]
innodb_data_home_dir = /ibdata
innodb_data_file_path=ibdata1:50M;ibdata2:50M:autoextend
```

> 注記
>
> InnoDB ではディレクトリが作成されないため、サーバーを起動する前に、/ibdata ディレクトリが存在することを確認してください。このことは、ユーザーが構成するログファイルディレクトリにも当てはまります。必要なディレクトリを作成するには、Unix または DOS の mkdir コマンドを使用します。
>
> MySQL サーバーがデータディレクトリ内にファイルを作成するための適切なアクセス権を持っていることを確認してください。さらに一般的に言えば、サーバーはデータファイルやログファイルを作成する必要のあるディレクトリ内に、アクセス権を持っている必要があります。

InnoDB は innodb_data_home_dir の値をテキストとしてデータファイル名に連結させ、必要に応じてパス名の区切り文字 (スラッシュまたはバックスラッシュ) を値の間に追加することで、各データファイルのディレクトリパスを形成します。my.cnf に innodb_data_home_dir オプションがまったく指定されていない場合は、デフォルト値が「ドット」ディレクトリ ./ (つまり、MySQL データディレクトリ) になります。(MySQL サーバーは実行の開始時に、現在の作業ディレクトリをそのデータディレクトリに変更します。)

innodb_data_home_dir を空の文字列として指定すると、innodb_data_file_path 値で一覧表示されたデータファイルに絶対パスを指定できます。次の例は、前述の例と同等です。

```
[mysqld]
innodb_data_home_dir =
innodb_data_file_path=/ibdata/ibdata1:50M;/ibdata/ibdata2:50M:autoextend
```

InnoDB 構成オプションの指定

小規模なシステム向けのサンプル my.cnf ファイル。512M バイトの RAM と 1 台のハードディスクが搭載されたコンピュータを使用すると仮定します。次の例では、autoextend 属性を含む、InnoDB の my.cnf または my.ini で指定可能な構成パラメータを示します。この例は、InnoDB データファイルとログファイルをいくつかのディスクに分散することを希望しない、Unix と Windows 両方のほとんどのユーザーに適しています。ここでは、MySQL データディレクトリ内に自動拡張データファイル ibdata1 と、2 つの InnoDB ログファイル ib_logfile0 および ib_logfile1 を作成します。

```
[mysqld]
# You can write your other MySQL server options here
# ...
# Data files must be able to hold your data and indexes.
# Make sure that you have enough free disk space.
innodb_data_file_path = ibdata1:12M:autoextend
#
# Set buffer pool size to 50-80% of your computer's memory
innodb_buffer_pool_size=256M
innodb_additional_mem_pool_size=20M
#
```

```
# Set the log file size to about 25% of the buffer pool size
innodb_log_file_size=64M
innodb_log_buffer_size=8M
#
innodb_flush_log_at_trx_commit=1
```

一部のファイルシステムでは、データファイルを 2G バイト未満にする必要があることに注意してください。ログファイルを結合したサイズは、最大で 512G バイトまでにすることができます。データファイルを結合したサイズは、10M バイトをわずかに超える大きさにする必要があります。

InnoDB システムテーブルスペースの設定

はじめて InnoDB システムテーブルスペースを作成する際は、コマンドプロンプトから MySQL サーバーを起動する方法が最適です。そのあと、データベースの作成に関する情報が InnoDB の画面に出力されるため、何が発生しているのかを確認できます。たとえば、Windows の場合、mysqld が C:\Program Files\MySQL\MySQL Server 5.6\bin に配置されていれば、次のように起動できます。

```
C:\> "C:\Program Files\MySQL\MySQL Server 5.6\bin\mysqld" --console
```

サーバーの出力が画面に送信されない場合は、サーバーのエラーログをチェックして、起動プロセス中に InnoDB から出力された内容を確認してください。

InnoDB で表示される情報の出力例については、セクション14.5.1「InnoDB テーブルスペースの作成」を参照してください。

MySQL 構成ファイルの編集

サーバーの起動時に読み取られる任意のオプションファイルの [mysqld] グループ内に、InnoDB オプションを配置できます。オプションファイルの場所については、セクション4.2.6「オプションファイルの使用」で説明されています。

インストールおよび構成ウィザードを使用して、MySQL を Windows 上にインストールした場合は、オプションファイルが MySQL インストールディレクトリ内に配置された my.ini ファイルになります。セクション2.3.3「MySQL Installer を使用した MySQL の Microsoft Windows へのインストール」を参照してください。

C: ドライブがブートドライブではないブートローダーが PC で使用されている場合は、Windows ディレクトリ (通常は C:\WINDOWS) 内の my.ini ファイルを使用することが唯一のオプションとなります。コンソールウィンドウ内のコマンドプロンプトで SET コマンドを使用すると、WINDIR の値を出力できます。

```
C:\> SET WINDIR
windir=C:\WINDOWS
```

mysqld で特定のファイルからのオプションのみが読み取られることを確認するには、サーバーの起動時に --defaults-file オプションをコマンド行の最初のオプションとして使用します。

```
mysqld --defaults-file=your_path_to_my_cnf
```

大規模なシステム向けのサンプル my.cnf ファイル。ディレクトリパス /、/dr2、および /dr3 に 2G バイトの RAM と 3 台の 60G バイトハードディスクが搭載された Linux コンピュータを使用すると仮定します。次の例では、InnoDB の my.cnf で指定可能な構成パラメータを示します。

```
[mysqld]
# You can write your other MySQL server options here
# ...
innodb_data_home_dir =
#
# Data files must be able to hold your data and indexes
innodb_data_file_path = /db/ibdata1:2000M;/dr2/db/ibdata2:2000M:autoextend
#
# Set buffer pool size to 50-80% of your computer's memory,
# but make sure on Linux x86 total memory usage is < 2GB
innodb_buffer_pool_size=1G
innodb_additional_mem_pool_size=20M
innodb_log_group_home_dir = /dr3/iblogs
#
# Set the log file size to about 25% of the buffer pool size
innodb_log_file_size=250M
innodb_log_buffer_size=8M
#
innodb_flush_log_at_trx_commit=1
innodb_lock_wait_timeout=50
#
# Uncomment the next line if you want to use it
```

```
#innodb_thread_concurrency=5
```

InnoDB の最大メモリー割り当ての決定

> **警告**
>
> 32 ビット版の GNU/Linux x86 では、高すぎるメモリー使用率を設定しないように注意してください。glibc では、プロセスヒープがスレッドスタック上で増加することが許可されている可能性があるため、サーバーがクラッシュします。次の式の値が 2G バイトに近づいていたり、超えていたりする場合は危険です。
>
> ```
> innodb_buffer_pool_size
> + key_buffer_size
> + max_connections*(sort_buffer_size+read_buffer_size+binlog_cache_size)
> + max_connections*2MB
> ```
>
> 各スレッドではスタックが使用され (多くの場合は 2M バイトですが、Oracle Corporation が提供する MySQL バイナリでは 256K バイトだけです)、最悪のケースでは、sort_buffer_size + read_buffer_size の追加メモリーも使用されます。

その他の mysqld サーバーパラメータの調整。次の値は典型的であり、ほとんどのユーザーに適しています。

```
[mysqld]
skip-external-locking
max_connections=200
read_buffer_size=1M
sort_buffer_size=1M
#
# Set key_buffer to 5 - 50% of your RAM depending on how much
# you use MyISAM tables, but keep key_buffer_size + InnoDB
# buffer pool size < 80% of your RAM
key_buffer_size=value
```

Linux の場合、カーネルで大規模ページのサポートが有効になっていれば、InnoDB は、バッファープールや追加メモリープールのメモリーを割り当てる際に大規模なページを使用できます。セクション8.11.4.2「ラージページのサポートの有効化」を参照してください。

14.3.1 読み取り専用操作用の InnoDB の構成

現在は、サーバーの起動時に --innodb-read-only 構成オプションを有効にすることで、MySQL データディレクトリが読み取り専用メディア上にある InnoDB テーブルでクエリーを実行できます。

有効にする方法

読み取り操作用にインスタンスを準備するには、必要なすべての情報が読み取り専用メディア上に格納される前に、データファイルにフラッシュされることを確認します。変更バッファーが無効になっている (innodb_change_buffering=0) サーバーを実行し、低速シャットダウンを実行します。

MySQL インスタンス全体にわたって読み取り専用モードを有効にするには、サーバーの起動時に次の構成オプションを指定します。

- --innodb-read-only=1
- インスタンスが DVD や CD などの読み取り専用メディア上にある場合、または /var ディレクトリがすべてのユーザーから書き込み可能でない場合: --pid-file=path_on_writeable_media および --event-scheduler=disabled

使用シナリオ

この操作モードは、次のような状況に適しています。

- DVD や CD などの読み取り専用ストレージメディア上に、MySQL アプリケーションまたは MySQL データセットを配布する。
- (一般に、データウェアハウス構成で) 同じデータディレクトリで同時にクエリーを実行する複数の MySQL インスタンス。この方法を使用すれば、負荷の高い MySQL インスタンスで発生する可能性のあるボトルネックを回避したり、さまざまなインスタンスに対してさまざまな構成オプションを使用して、特定の種類のクエリーを個別に調整したりできる場合があります。
- 安全性またはデータの完全性の理由により読み取り専用の状態になったデータ (アーカイブされたバックアップデータなど) のクエリーを実行する。

> **注記**
>
> この機能の目的は、読み取り専用の側面に基づいた生のパフォーマンスではなく、主に配布および配備する際の柔軟性です。サーバー全体を読み取り専用にする必要なしで、読み取り専用クエリーのパフォーマンスを調整する方法については、セクション 14.13.14「InnoDB の読み取り専用トランザクションの最適化」を参照してください。

動作

--innodb-read-only オプションを使用して、サーバーが読み取り専用モードで実行されると、特定の InnoDB 機能およびコンポーネントが減少したり、完全に無効になったりします。

- 変更バッファー (特に、変更バッファーからのマージ) は実行されません。読み取り専用操作用にインスタンスを準備するときに、変更バッファーが空になっていることを確認するには、変更バッファーを無効にして (innodb_change_buffering=0)、まず低速シャットダウンを実行します。
- 起動時にはクラッシュリカバリフェーズがありません。インスタンスを読み取り専用状態にする前に、低速シャットダウンが実行されている必要があります。
- 読み取り専用操作では Redo ログが使用されないため、インスタンスを読み取り専用にする前に、innodb_log_file_size を最小限のサイズ (1M バイト) に設定できます。
- I/O 読み取りスレッド以外のバックグラウンドスレッドがすべて無効になります。その結果、読み取り専用インスタンスでデッドロックが発生する可能性がなくなります。
- デッドロックに関する情報やモニターの出力などは、一時ファイルに書き込まれません。その結果、SHOW ENGINE INNODB STATUS で出力が生成されなくなります。
- --innodb-read-only を使用して MySQL サーバーが起動されているが、まだデータディレクトリが書き込み可能メディア上にある場合は、root ユーザーが引き続き、GRANT や REVOKE などの DCL 操作を実行できます。
- 構成オプションの設定を変更すると、通常は書き込み操作の動作が変更されますが、サーバーが読み取り専用モードになっている場合は影響がありません。
- 分離レベルを強制的に適用する MVCC 処理が無効になります。更新も削除もできないため、すべてのクエリーで最新バージョンのレコードが読み取られます。
- Undo ログは使用されません。innodb_undo_tablespaces および innodb_undo_directory 構成オプションの設定を無効にします。

14.4 InnoDB の管理

InnoDB に関連する管理タスクには、主に次のような側面が伴います。

- システムテーブルスペース、InnoDB テーブル、およびそれらに関連付けられたインデックスを表すデータファイルの管理。これらのファイルをレイアウトおよび分割する方法は変更できますが、特定のテーブルで実現可能なパフォーマンスと機能の両方が影響を受けます。
- クラッシュリカバリで使用される Redo ログファイルの管理。これらのファイルのサイズを指定できます。
- InnoDB が別のストレージエンジンでなく、目的のテーブルで使用されているかどうかの確認。
- パフォーマンスに関連する一般的な管理タスク。アプリケーションの設計フェーズ時にアプリケーション開発者に相談したり、システム設定が正常に機能していることを確認するために継続的にパフォーマンスをモニターしたり、突然に発生したパフォーマンスや容量の問題を診断して修正したりする場合があります。

現在は InnoDB テーブルが MySQL のデフォルトであるため、関連する管理資料の大半は主要な「管理」の章、第 5章「MySQL サーバーの管理」に記載されています。

14.5 InnoDB テーブルスペース管理

14.5.1 InnoDB テーブルスペースの作成

MySQL をインストールし、必要な InnoDB 構成パラメータが含まれるようにオプションファイルを編集したと仮定します。MySQL を起動する前に、InnoDB のデータファイルおよびログファイルに指定したディレクトリが存在し、MySQL サーバーがそれらのディレクトリへのアクセス権を持っていることを確認します。InnoDB はファイルだけを作成し、ディレクトリは作成しません。データファイルおよびログファイル用のディスク領域が十分にあることもチェックします。

InnoDB が有効になっている状態でサーバーをはじめて起動するときの最適な方法は、MySQL サーバー mysqld を mysqld_safe からでも、Windows サービスとしてでもなく、コマンドプロンプトから実行することです。コマンドプロンプトから実行すると、mysqld で出力される内容および発生している事象が表示されます。Unix では、単に mysqld を呼び出すだけです。Windows では、出力先がコンソールウィンドウになるように、--console オプションを付けて mysqld を起動します。

オプションファイル内ではじめて InnoDB を構成したあとに、MySQL サーバーを起動すると、InnoDB によってデータファイルおよびログファイルが作成され、次のような内容が出力されます。

```
InnoDB: The first specified datafile /home/heikki/data/ibdata1
did not exist:
InnoDB: a new database to be created!
InnoDB: Setting file /home/heikki/data/ibdata1 size to 134217728
InnoDB: Database physically writes the file full: wait...
InnoDB: datafile /home/heikki/data/ibdata2 did not exist:
new to be created
InnoDB: Setting file /home/heikki/data/ibdata2 size to 262144000
InnoDB: Database physically writes the file full: wait...
InnoDB: Log file /home/heikki/data/logs/ib_logfile0 did not exist:
new to be created
InnoDB: Setting log file /home/heikki/data/logs/ib_logfile0 size
to 5242880
InnoDB: Log file /home/heikki/data/logs/ib_logfile1 did not exist:
new to be created
InnoDB: Setting log file /home/heikki/data/logs/ib_logfile1 size
to 5242880
InnoDB: Doublewrite buffer not found: creating new
InnoDB: Doublewrite buffer created
InnoDB: Creating foreign key constraint system tables
InnoDB: Foreign key constraint system tables created
InnoDB: Started
mysqld: ready for connections
```

この時点で、InnoDB によってテーブルスペースおよびログファイルが初期化されています。mysql と同様に、通常の MySQL クライアントプログラムを使用して MySQL サーバーに接続できます。mysqladmin shutdown を使用して MySQL サーバーをシャットダウンすると、次のように出力されます。

```
010321 18:33:34  mysqld: Normal shutdown
010321 18:33:34  mysqld: Shutdown Complete
InnoDB: Starting shutdown...
InnoDB: Shutdown completed
```

データファイルおよびログディレクトリを調査すれば、そこに作成されたファイルを確認できます。MySQL が再起動されるときには、データファイルおよびログファイルはすでに作成されているため、出力はさらに簡潔になります。

```
InnoDB: Started
mysqld: ready for connections
```

innodb_file_per_table オプションを my.cnf に追加すると、InnoDB では各テーブルが、.frm ファイルが作成された場所と同じ MySQL データベースディレクトリにある独自の .ibd ファイルに格納されます。セクション 14.5.2「InnoDB File-Per-Table モード」を参照してください。

14.5.2 InnoDB File-Per-Table モード

従来、InnoDB のすべてのテーブルとインデックスは、システムテーブルスペースに格納されていました。このモノリシック方式は、データの拡大を綿密に計画した、データベース処理に完全に特化したマシンに対象を絞った方式です。この方式では、MySQL に割り当てられるどのディスクストレージも、他の目的のために要求されることはありません。InnoDB の file-per-table モードは、InnoDB の各テーブルとそのインデックスを個別のファイルに格納する方式の、柔軟性を向上した代替手段です。.ibd のような各ファイルは、それぞれのテーブルスペースを表します。このモードは innodb_file_per_table 構成オプションで制御され、MySQL 5.6.6 以降ではデフォルトです。

File-Per-Table モードの利点

- テーブルを切り詰めたり、削除したりするときに、オペレーティングシステムのディスクスペースを再利用できます。file-per-table モードがオフに切り替えられたときに作成されたテーブルの場合、テーブルを切り詰めたり、削除したりすると、内部的に ibdata ファイルに空き領域が作成されますが、その空き領域は新しい InnoDB データにのみ使用できます。
- TRUNCATE TABLE 操作は、個々の .ibd ファイルで実行する場合よりも高速です。

- I/O の最適化、容量管理、またはバップアップのために、特定のテーブルを別々のストレージデバイスに格納することができます。前のリリースでは、セクション8.11.3.1「シンボリックリンクの使用」で述べられているように、ほかのドライブにデータベースディレクトリ全体を移動し、MySQL データディレクトリにシンボリックリンクを作成する必要がありました。MySQL 5.6.6 以降では、セクション14.5.4「テーブルスペースの位置の指定」で述べられているように、構文 CREATE TABLE ... DATA DIRECTORY = absolute_path_to_directory を使用して各テーブルの位置を特定できます。

- OPTIMIZE TABLE を実行して、テーブルスペースを圧縮または再作成できます。OPTIMIZE TABLE を実行すると、InnoDB は実際のデータの格納に必要なスペースだけを使用して、一時名を持つ新しい .ibd ファイルを作成します。最適化が完了すると、InnoDB は古い .ibd ファイルを削除し、新しい .ibd ファイルで置き換えます。前の .ibd ファイルは非常に大きくなったが、実際のデータはそのサイズの一部を占めるだけの場合、OPTIMIZE TABLE を実行すると、未使用領域を再利用できます。

- データベース全体ではなく、個別の InnoDB テーブルを移動できます。

- ある MySQL インスタンスから別のインスタンスに個別の InnoDB テーブルをコピーできます (トランスポータブルテーブルスペース 機能として知られています)。

- innodb_file_per_table が有効なときに作成されたテーブルは、Barracuda ファイル形式を使用できます。Barracuda ファイル形式は、圧縮 および ダイナミック の行フォーマットなどの機能に対応します。innodb_file_per_table がオフのときに作成されたテーブルは、これらの機能を使用できません。これらの機能を既存のテーブルに利用するには、file-per-table 設定をオンにして、既存のテーブルで ALTER TABLE t ENGINE=INNODB を実行します。テーブルを変換する前に、セクション14.6.4「MyISAM から InnoDB へのテーブルの変換」を参照してください。

- ダイナミック行フォーマットを使用して、大きな BLOB またはテキストカラムを持つテーブルでストレージ効率を向上させることができます。

- innodb_file_per_table を使用すると、リカバリが成功する可能性が高くなることがあります。また破損が発生した場合、サーバーが再起動できない場合、またはバックアップおよびバイナリログが使用できない場合に、時間が節約できる可能性があります。

- MySQL Enterprise Backup 製品を使用して、他の InnoDB テーブルの使用を中断せずに、単一のテーブルを迅速にバックアップしたり、リストアしたりできます。詳細は、Partial Backup and Restore Optionsを参照してください。

- File-per-table モードを使用すると、バックアップからテーブルを除外できます。これは、それほど頻繁にバックアップを取る必要のないテーブル、または別のスケジュールでバックアップが必要なテーブルに有効です。

- file-per-table モードは、テーブルをコピーまたはバックアップするときにレポートする、テーブルごとのステータスに便利です。

- file-per-table モードを使用すると、MySQL にアクセスせずに、ファイルシステムレベルでテーブルサイズをモニターできます。

- 一般的な Linux ファイルシステムでは、innodb_flush_method が O_DIRECT に設定されていると、1 つのファイルへの並列書き込みは許可されていません。その結果、innodb_flush_method と併せて innodb_file_per_table を使用すると、パフォーマンスが向上する可能性があります。

- innodb_file_per_table が無効な場合、テーブル、データディクショナリ、および Undo ログに対して 1 つの共有テーブルスペース（システムテーブルスペース）があります。この 1 つのテーブルスペースのサイズの制限値は 64TB です。innodb_file_per_table が有効である場合、各テーブルには独自のテーブルスペースがあり、各テーブルスペースのサイズの制限値は 64TB です。関連情報については、セクションD.10.3「テーブルサイズの制限」を参照してください。

File-Per-Table モードの考えられる短所

- innodb_file_per_table の場合、各テーブルには未使用のテーブルスペースがある場合があり、そのテーブルスペースを利用できるのは同じテーブルの行だけです。これは、テーブルスペースが適切に管理されていないと、無駄なテーブルスペースが減るのではなく、増える結果となる可能性があります。

- fsync 操作は、1 つのファイルではなく、各オープンテーブルで実行する必要があります。ファイルごとに別々の fsync 操作があるため、複数のテーブルへの書き込み操作を 1 つの I/O 操作に結合することはできません。このため、InnoDB は fsync 操作の総数を増やして実行する必要がある可能性があります。

- mysqld では、テーブル当たり 1 つのオープンファイルの処理を維持する必要があるため、大量のテーブルがある場合、パフォーマンスに影響が出る可能性があります。

- より多くのファイルディスクリプタが使用されます。
- innodb_file_per_table は、MySQL 5.6.6 以降ではデフォルトでオンです。MySQL 5.5 または 5.1 との下位互換性に懸念がある場合、無効化の検討をお勧めします。innodb_file_per_table を無効にすると、ALTER TABLE がテーブル (ALGORITHM=COPY) を再作成する場合、ALTER TABLE で InnoDB テーブルがシステムのテーブルスペースから個々の .ibd ファイルに移動するのが抑えられます。

  たとえば、InnoDB テーブルのクラスタインデックスを再構築する場合、テーブルはinnodb_file_per_table の現在の設定を使用して再作成されます。この動作は、InnoDB のセカンダリインデックスを追加または削除するときには適用されません。セカンダリインデックスがテーブルを再構築しないで作成された場合、現在の innodb_file_per_table 設定にかかわらず、そのインデックスはテーブルデータと同じファイルに格納されます。
- テーブルが増えていくと、DROP TABLE およびテーブルスキャンのパフォーマンスが下がることがあるフラグメンテーションが増える可能性があります。しかし、フラグメンテーションが管理されている場合、ファイル自身のテーブルスペース内にファイルがあると、パフォーマンスが向上する可能性があります。
- バッファープールはテーブルごとのテーブルスペースを削除するときにスキャンされるため、サイズが数十ギガバイトのバッファープールに数秒かかる場合があります。スキャンは広範囲に内部ロックをかけて実行されるため、他の操作を遅らせる場合があります。共有テーブルスペース内のテーブルには影響ありません。
- 自動拡張の共有テーブルスペースが満杯になったときに、そのサイズを拡張する場合の増分サイズ (MB 単位) を定義する innodb_autoextend_increment 変数は、file-per-table テーブルスペースファイルに適用されません。innodb_autoextend_increment の値にかかわらず、file-per-table テーブルスペースファイルは自動拡張です。拡張は少量で始まり、その後の拡張は増分が 4MB で発生します。

14.5.3 File-Per-Table モードの有効化および無効化

file-per-table モードを MySQL サーバーのデフォルトにするには、--innodb_file_per_table コマンド行オプションでサーバーを起動するか、my.cnf の [mysqld] セクションに次の行を追加します。

```
[mysqld]
innodb_file_per_table
```

サーバーが動作している間に、コマンドを発行することもできます。

```
SET GLOBAL innodb_file_per_table=1;
```

file-per-table モードが有効な場合、InnoDB は、適切なデータベースディレクトリ内の独自の tbl_name.ibd ファイルに、新しく作成された各テーブルを格納します。MyISAM ストレージエンジンとは異なり、インデックスとデータにそれぞれ tbl_name.MYD と tbl_name.MYI ファイルがあると、InnoDB はデータとインデックスを一緒に 1 つの .ibd ファイルに格納します。それでも、tbl_name.frm ファイルは従来どおり作成されます。

使用する起動オプションから innodb_file_per_table を削除してサーバーを再起動する場合、または SET GLOBAL コマンドでオプションをオフにした場合、InnoDB はシステムのテーブルスペース内に新しいテーブルを作成します。

file-per-table 設定にかかわらず、InnoDB テーブルの読み取りおよび書き込みは常に実行できます。

テーブルをシステムテーブルスペースから自身のテーブルスペースへ移動するには、またはその反対方向にテーブルを移動するには、innodb_file_per_table 設定を変更してテーブルを再作成します。

```
-- Move table from system tablespace to its own tablespace.
SET GLOBAL innodb_file_per_table=1;
ALTER TABLE table_name ENGINE=InnoDB;
-- Move table from its own tablespace to system tablespace.
SET GLOBAL innodb_file_per_table=0;
ALTER TABLE table_name ENGINE=InnoDB;
```

> 注記
>
> InnoDB は、内部のデータディクショナリ と Undo ログをシステムのテーブルスペースに配置するため、常にこのテーブルスペースが必要です。.ibd ファイルは InnoDB が動作するのには十分ではありません。
>
> システムテーブルスペースから独自の .ibd ファイルにテーブルが移動された場合、システムテーブルスペースを構成するデータファイルのサイズは維持されます。以前にテーブルが占有したスペースは、新しい InnoDB データ用に再利用できますが、オペレーティングシステム用には再利用されません。システムテーブルスペースから、ディスクスペースが限られているところへ大規模な InnoDB テーブルを移動する場

合、innodb_file_per_table をオンにしてから mysqldump コマンドを使用してインスタンス全体を再作成することをお勧めします。

14.5.4 テーブルスペースの位置の指定

新しい InnoDB file-per-table テーブルスペースを MySQL データディレクトリの外側の特定の位置に作成するには、CREATE TABLE ステートメントの DATA DIRECTORY = absolute_path_to_directory 句を使用します。

あらかじめ位置について計画を立てます。ALTER TABLE ステートメントで DATA DIRECTORY 句を使用できないためです。指定するディレクトリは、高速の SSD や大容量の HDD など、パフォーマンスや容量に際だった特徴のあるほかのストレージデバイス上に置くことも可能です。

MySQL は、目的のディレクトリ内にデータベース名に対応するサブディレクトリを作成し、その中に、新しいテーブル用の .ibd ファイル を作成します。MySQL は、MySQL DATADIR ディレクトリの下のデータベースディレクトリに、テーブルのパス名を含む table_name.isl ファイル作成します。.isl file は MySQL によってシンボリックリンクのように処理されます。(実際のシンボリックリンクを使用することは InnoDB テーブルではサポートされませんでした。)

次の例では、MySQL の開発またはテスト用の小さなインスタンスを、95% が使用済みのプライマリハードドライブを搭載したノート型パソコンで実行し、空きスペースが多い別のストレージデバイス上に名前が EXTERNAL の新しいテーブルを配置する方法について示します。シェルコマンドは、DATADIR ディレクトリの下のデフォルトの位置にある LOCAL テーブルおよび指定した位置にある EXTERNAL テーブルへのさまざまなパスを示します。

```
mysql> \! df -k .
Filesystem   1024-blocks      Used Available Capacity  iused   ifree %iused  Mounted on
/dev/disk0s2   244277768 231603532  12418236    95% 57964881 3104559   95%   /

mysql> use test;
Database changed
mysql> show variables like 'innodb_file_per_table';
+-----------------------+-------+
| Variable_name         | Value |
+-----------------------+-------+
| innodb_file_per_table | ON    |
+-----------------------+-------+
1 row in set (0.00 sec)

mysql> \! pwd
/usr/local/mysql
mysql> create table local (x int unsigned not null primary key);
Query OK, 0 rows affected (0.03 sec)

mysql> \! ls -l data/test/local.ibd
-rw-rw----  1 cirrus  staff  98304 Nov 13 15:24 data/test/local.ibd

mysql> create table external (x int unsigned not null primary key) data directory = '/volumes/external1/data';
Query OK, 0 rows affected (0.03 sec)

mysql> \! ls -l /volumes/external1/data/test/external.ibd
-rwxrwxrwx  1 cirrus  staff  98304 Nov 13 15:34 /volumes/external1/data/test/external.ibd

mysql> select count(*) from local;
+----------+
| count(*) |
+----------+
|        0 |
+----------+
1 row in set (0.01 sec)

mysql> select count(*) from external;
+----------+
| count(*) |
+----------+
|        0 |
+----------+
1 row in set (0.01 sec)
```

注:

- 最初、MySQL は .ibd ファイルを開いたままにして、デバイスがマウント解除されないようにしていますが、サーバーがビジーになると、結果的にテーブルを閉じることになる場合があります。MySQL の動作中に誤って外部デバイスをマウント解除した

り、デバイスが接続されていないときに MySQL を起動したりしないように注意してください。関連する .ibd ファイルが欠けているときにテーブルにアクセスしようとすると、サーバーの再起動が必要となる重大なエラーの原因となります。

それでも .ibd ファイルが期待するパス上にない場合、サーバーの再起動が失敗する可能性があります。この場合、手動でデータベースディレクトリ内の table_name.isl ファイルを削除し、再起動後に DROP TABLE を実行して .frm ファイルを削除し、テーブルに関する情報をデータディクショナリから削除します。

- NFS がマウントされたボリュームに MySQL テーブルを配置しないでください。NFS はメッセージパッシングプロトコルを使用してファイルに書き込むため、ネットワークメッセージが失われたり、順序どおりに受信しなかったりした場合に、データの一貫性が失われる原因となる可能性があります。
- LVM スナップショット、ファイルのコピー、または他のファイルベースのメカニズムを使用して、.ibd ファイルのバックアップを取る場合、必ず最初に FLUSH TABLES ... FOR EXPORT ステートメントを使用して、メモリーにバッファリングされたすべての変更が、バックアップを行う前にディスクにフラッシュされていることを確認します。
- DATA DIRECTORY 句は シンボリックリンクの使用 のサポート済み代替手段です。これはずっと問題があり、各 InnoDB テーブルには決してサポートされませんでした。

14.5.5 テーブルスペースの別のサーバーへのコピー (トランスポータブルテーブルスペース)

このセクションでは、Transportable Tablespace 機能を使用して、file-per-table テーブルスペース (.idb ファイル) をあるデータベースサーバーから別のデータベースサーバーにコピーする方法について説明します。

他の InnoDB テーブルコピー方式について詳しくは、セクション14.6.2「別のマシンへの InnoDB テーブルの移動またはコピー」を参照してください。

InnoDB file-per-table テーブルスペースを別のデータベースサーバーにコピーすることをお勧めするのには、多くの理由があります。

- 本番サーバーに余計な負荷を掛けずにレポートを実行するため。
- 新しいスレーブサーバーに、あるテーブルとまったく同じデータをセットアップするため。
- 問題や誤りが発生したあとに、テーブルのバックアップ版をリストアするため。
- mysqldump コマンドの結果をインポートするよりも、データを移動させる方が速いため。データの再挿入とインデックスの再構築を行うよりも、データがすぐに使用できるためです。
- システム要件により適したストレージ媒体を持つサーバーに file-per-table テーブルスペースを移動するため。たとえば、アクセス頻度の高いテーブルを SSD デバイスに置いたり、大規模なテーブルを大容量の HDD デバイスに置いたりする場合です。

テーブルスペースのコピーの制限と使用方法に関する注意 (トランスポータブルテーブルスペース)

- テーブルスペースのコピー手順は、innodb_file_per_table が ON (MySQL 5.6.6 以降ではデフォルトです) に設定されている場合にのみ可能です。共有のシステムテーブルスペースにあるテーブルは休止できません。
- テーブルが休止されると、影響を受けたテーブルでは読み取り専用のトランザクションのみが許可されます。
- テーブルスペースをインポートする場合、ページサイズはインポートするインスタンスのページサイズに一致する必要があります。
- DISCARD TABLESPACE がパーティション化されたテーブルでサポートされていないということは、トランスポータブルテーブルスペースも未サポートであることを意味します。パーティション化されたテーブルで ALTER TABLE ... DISCARD TABLESPACE を実行すると、次のエラーが返されます。ERROR 1031 (HY000): Table storage engine for 'part' doesn't have this option.
- DISCARD TABLESPACE は、foreign_key_checks が 1 に設定されている親/子 (主キー/外部キー) 関係を持つテーブルスペースをサポートしていません。親子関係のテーブルのテーブルスペースを破棄する前に、foreign_key_checks=0 を設定します。

- ALTER TABLE ... IMPORT TABLESPACE は、インポートされたデータに対して外部キー制約を課しません。テーブル間に外部キー制約がある場合、すべてのテーブルを同じ (論理上の) 時点でエクスポートしてください。
- ALTER TABLE ... IMPORT TABLESPACE では、テーブルスペースをインポートするための .cfg メタデータファイルは必要ありません。ただし、.cfg ファイルなしでインポートした場合は、メタデータのチェックは実行されず、次に類似した警告が発行されます。

```
Message: InnoDB: IO Read error: (2, No such file or directory) Error opening '.\
test\t.cfg', will attempt to import without schema verification
1 row in set (0.00 sec)
```

  .cfg ファイルなしでインポートする機能は、スキーマの不一致が予想されない場合に、使い勝手が高まる可能性があります。また、.cfg ファイルなしでインポートする機能は、メタデータが .ibd ファイルから収集できないクラッシュリカバリシナリオで役立つ可能性があります。
- MySQL 5.6 以降では、両方のサーバーが GA (一般提供) ステータスであり、両者のバージョンが同じシリーズである場合に、テーブルスペースファイルの別のサーバーからのインポートが機能します。そうでないと、インポート先のサーバーにファイルが作成されていなければなりません。
- レプリケーションシナリオでは、マスターとスレーブの両方で innodb_file_per_table が ON に設定されている必要があります。
- Windows では、InnoDB はデータベース、テーブルスペース、およびテーブル名を内部的に小文字で格納します。Linux や UNIX など、大文字と小文字を区別するオペレーティングシステムでインポートの問題を回避するには、すべてのデータベース、テーブルスペース、およびテーブルを小文字名を使用して作成します。これを遂行する便利な方法は、データベース、テーブルスペース、またはテーブルを作成する前に、my.cnf または my.ini ファイルの [mysqld] セクションに次の行を追加することです。

```
[mysqld]
lower_case_table_names=1
```

手順の例: あるサーバーから別のサーバーへのテーブルスペースのコピー (トランスポータブルテーブルスペース)

この手順では、MySQL の実行中のサーバーインスタンスから実行中の別のインスタンスへテーブルをコピーする方法について説明します。同じ手順は、微調整を加えると、同じインスタンスでテーブルの完全なリストアを実行するために使用できます。

1. ソースサーバーで、テーブルがまだ存在していない場合、テーブルを作成します。

```
mysql> use test;
mysql> CREATE TABLE t(c1 INT) engine=InnoDB;
```

2. 目的サーバーで、テーブルが存在していない場合、テーブルを作成します。

```
mysql> use test;
mysql> CREATE TABLE t(c1 INT) engine=InnoDB;
```

3. 目的サーバーで、既存のテーブルスペースを破棄します。(テーブルスペースをインポートする前に、InnoDB は受け取り側のテーブルにアタッチされたテーブルスペースを破棄します。)

```
mysql> ALTER TABLE t DISCARD TABLESPACE;
```

4. ソースサーバーでは、FLUSH TABLES ... FOR EXPORT を実行してテーブルを休止し、.cfg メタデータファイルを作成します。

```
mysql> use test;
mysql> FLUSH TABLES t FOR EXPORT;
```

   メタデータ (.cfg) ファイルは InnoDB データディレクトリに作成されます。

> **注記**
>
> FLUSH TABLES ... FOR EXPORT は MySQL 5.6.6 以降で使用できます。このステートメントは、サーバー稼働中にバイナリテーブルのコピーができるように、名前付きテーブルへの変更をディスクにフラッシュします。FLUSH TABLES ... FOR EXPORT が実行されると、InnoDB はテーブルと同じデータベースディレクトリに .cfg ファイルを作成します。.cfg ファイルには、テーブルスペースファイルをインポートするときのスキーマの検証に使用されるメタデータが含まれます。

5. .ibd ファイルおよび .cfg メタデータファイルをソースサーバーから目的サーバーにコピーします。例:

```
shell> scp /path/to/datadir/test/t.{ibd,cfg} destination-server:/path/to/datadir/test
```

> **注記**
>
> .ibd ファイルおよび .cfg ファイルは、次の手順で示すように、共有ロックを解放する前にコピーする必要があります。

6. ソースサーバーでは、UNLOCK TABLES を使用して、FLUSH TABLES ... FOR EXPORT によって取得されたロックを解放します。

```
mysql> use test;
mysql> UNLOCK TABLES;
```

7. 目的サーバーで、テーブルスペースをインポートします。

```
mysql> use test;
mysql> ALTER TABLE t IMPORT TABLESPACE;
```

> **注記**
>
> ALTER TABLE ... IMPORT TABLESPACE 機能は、インポートされたデータに対して外部キー制約を課しません。テーブル間に外部キー制約がある場合、すべてのテーブルを同じ (論理上の) 時点でエクスポートしてください。この場合、テーブルの更新を停止し、すべてのトランザクションをコミットし、テーブルで共有ロックを取得してから、エクスポート操作を実行します。

テーブルスペースのコピーの内部情報 (トランスポータブルテーブルスペース)

次の情報では、トランスポータブルテーブルスペースのコピー手順に関する内部情報とエラーログについて説明します。

ALTER TABLE ... DISCARD TABLESPACE が目的のインスタンスで実行された場合。

- テーブルは X モードでロックされています。
- テーブルスペースがテーブルから切り離されています。

FLUSH TABLES ... FOR EXPORT がソースインスタンスで実行された場合。

- エクスポートのためにフラッシュされたテーブルが共有モードでロックされています。
- パージコーディネータのスレッドが停止しています。
- ダーティーページがディスクに同期しています。
- テーブルのメタデータがバイナリの .cfg ファイルに書き込まれました。

この操作で予想されるエラーログメッセージです。

```
2013-07-18 14:47:31 34471 [Note] InnoDB: Sync to disk of '"test"."t"' started.
2013-07-18 14:47:31 34471 [Note] InnoDB: Stopping purge
2013-07-18 14:47:31 34471 [Note] InnoDB: Writing table metadata to './test/t.cfg'
2013-07-18 14:47:31 34471 [Note] InnoDB: Table '"test"."t"' flushed to disk
```

UNLOCK TABLES がソースインスタンスで実行された場合。

- バイナリの .cfg ファイルが削除されました。
- インポートされたテーブル (または複数のテーブル) の共有ロックが解放され、パージコーディネータのスレッドが再起動されました。

この操作で予想されるエラーログメッセージです。

```
2013-07-18 15:01:40 34471 [Note] InnoDB: Deleting the meta-data file './test/t.cfg'
2013-07-18 15:01:40 34471 [Note] InnoDB: Resuming purge
```

ALTER TABLE ... IMPORT TABLESPACE が目的のインスタンスで実行されると、インポートのアルゴリズムはインポートされたテーブルスペースごとに次の操作を実行します。

- テーブルスペースの各ページに破損があるかどうかをチェックします。

- 各ページのスペース ID とログシーケンス番号 (LSN) が更新されます。
- フラグが検証され、ヘッダーページの LSN が更新されます。
- B ツリーページが更新されます。
- ページの状態がディスクに書き込まれるように、この状態をダーティーに設定します。

この操作で予想されるエラーログメッセージです。

```
2013-07-18 15:15:01 34960 [Note] InnoDB: Importing tablespace for table 'test/t' that was exported from host 'ubuntu'
2013-07-18 15:15:01 34960 [Note] InnoDB: Phase I - Update all pages
2013-07-18 15:15:01 34960 [Note] InnoDB: Sync to disk
2013-07-18 15:15:01 34960 [Note] InnoDB: Sync to disk - done!
2013-07-18 15:15:01 34960 [Note] InnoDB: Phase III - Flush changes to disk
2013-07-18 15:15:01 34960 [Note] InnoDB: Phase IV - Flush complete
```

> **注記**
>
> テーブルスペースが破棄されたこと (目的のテーブルのテーブルスペースを破棄した場合) を伝える警告、および .ibd ファイルがないために統計値が計算できなかったことを伝えるメッセージも受け取る場合があります。
>
> ```
> 2013-07-18 15:14:38 34960 [Warning] InnoDB: Table "test"."t" tablespace is set as discarded.
> 2013-07-18 15:14:38 7f34d9a37700 InnoDB: cannot calculate statistics for table "test"."t" because the .ibd file is missing. For hel
> http://dev.mysql.com/doc/refman/5.7/en/innodb-troubleshooting.html
> ```

14.5.6 個別のテーブルスペースへの InnoDB Undo ログの格納

MySQL 5.6.3 以降は、システムテーブルスペースを除く 1 つ以上の個別の Undo テーブルスペースに InnoDB の Undo ログを格納することができます。このレイアウトは、Undo ログが システムテーブルスペースの一部であるデフォルト構成とは異なります。Undo ログの I/O パターンにより、これらのテーブルスペースは SSD ストレージへの移動に適した候補になります。一方、システムテーブルスペースはハードディスクストレージに維持します。ユーザーは、InnoDB の Undo ログを保持するために作成された個々のテーブルスペース、およびそのテーブルスペース内の各セグメントを削除できません。

これらのファイルで、システムテーブルスペース内で以前に実行された I/O 操作を処理するため、これらの新しいファイルを加えるように、システムテーブルスペースの定義を拡張します。

Undo ログはロールバックセグメントとしても知られています。

この機能には、次の新しい構成オプションまたは名前の変わった構成オプションが含まれます。

- innodb_undo_tablespaces。
- innodb_undo_directory。
- innodb_rollback_segments は innodb_undo_logs になります。互換性のため、古い名前も使用できます。

InnoDB の Undo ログ機能には 2 つの非動的起動変数 (innodb_undo_tablespaces および innodb_undo_directory) の設定を伴うため、この機能は MySQL インスタンスを初期化する場合にのみ有効化できます。

使用上の注意

この機能を使用するには、次の手順に従います。

1. Undo ログを保持するパスを決めます。MySQL 構成ファイルまたは起動スクリプトにある innodb_undo_directory オプションの引数としてそのパスを指定します。
2. innodb_undo_logs オプションにゼロでない開始値を指定します。比較的小さい値から始めて、パフォーマンスの効果を調べて徐々に値を増やすことができます。
3. innodb_undo_tablespaces オプションにゼロでない値を指定します。innodb_undo_logs 値で指定された複数の Undo ログは、この数の個々のテーブルスペース (.ibd ファイルで指定) に分割されます。この値は、MySQL インスタンスが有効である間は固定されるため、最適値が確かでない場合は、どちらかというと高く見積もります。
4. 構成ファイルまたは MySQL 起動スクリプトで選択した値を使用して、新しい MySQL インスタンスを作成します。本番サーバーに類似したデータ量の現実的なワークロードを使用します。また、トランスポータブルテーブルスペース機能を使用して、既存のデータベースのテーブルを新しく構成した MySQL インスタンスに

コピーします。詳細は、セクション14.5.5「テーブルスペースの別のサーバーへのコピー (トランスポータブルテーブルスペース)」を参照してください。

5. I/O が多いワークロードのパフォーマンスのベンチマーク。
6. 繰り返し、innodb_undo_logs の値を増やして、パフォーマンステストをやり直します。I/O パフォーマンスゲインが停止する値を見つけます。
7. これらのオプションに理想的な設定を使用して、新しい本番インスタンスを配備します。レプリケーション構成でスレーブサーバーとして設定するか、以前の本番インスタンスからデータを移動します。

パフォーマンスおよびスケーラビリティーに関する考慮事項

Undo ログを個々のファイルに保持すると、MySQL チームはこのトランザクションデータに関連した、I/O とメモリーの最適化を実装できます。たとえば、Undo データはディスクに書き込まれ、その上まれにしか使用されないため (クラッシュリカバリの場合にのみ)、ファイルシステムのメモリーキャッシュに保持しておく必要がなく、その結果、より多くのシステムメモリーを InnoDB のバッファープール に割り当てることができます。

InnoDB システムのテーブルスペースをハードドライブに保持し、テーブルごとのテーブルスペースを SSD に移動するという、典型的な SSD ベストプラクティスは、Undo 情報を個々のテーブルスペースファイルに移動することで役に立ちます。

内部情報

物理的なテーブルスペースファイルの名前は undoN です。ここで、N はスペース ID (頭のゼロを含む) です。

現在、個々の Undo テーブルスペースを含む MySQL インスタンスは、MySQL 5.5 や 5.1 などの以前のリリースにはダウングレードできません。

14.5.7 InnoDB ログファイルの数またはサイズの変更、および InnoDB テーブルスペースのサイズの変更

このセクションでは、InnoDB の Redo ログファイルの数またはサイズを変更する方法、および InnoDB のシステムテーブルスペースのサイズを増加または減少する方法について説明します。

InnoDB ログファイルの数またはサイズの変更

MySQL 5.6.7 以前で InnoDB の Redo ログファイルの数またはサイズを変更するには、次の手順を実行します。

1. innodb_fast_shutdown が 2 に設定されている場合は、innodb_fast_shutdown を 1 に設定します。

```
mysql> SET GLOBAL innodb_fast_shutdown = 1;
```

2. innodb_fast_shutdown が 2 に設定されていないことを確認したあとに、MySQL サーバーを停止し、エラーなしでシャットダウンされること (ログ内に未処理のトランザクションに関する情報が存在しないこと) を確認します。
3. シャットダウン中に何か問題が発生して、テーブルスペースをリカバリするために古いログファイルが必要となる場合に備えて、それらのログファイルを安全な場所にコピーします。
4. ログファイルディレクトリから古いログファイルを削除します。
5. my.cnf を編集して、ログファイルの構成を変更します。
6. MySQL を再起動します。mysqld によって、起動時に InnoDB ログファイルが存在しないことが表示され、新しいログファイルが作成されます。

MySQL 5.6.8 の時点では、InnoDB ログファイルの数またはサイズを変更する際に、innodb_fast_shutdown 設定が関連しなくなりました。さらに、古いログファイルを削除する必要もなくなりました。ただし、バックアップとして古いログファイルを安全な場所にコピーすることはあります。InnoDB のログファイルの数またはサイズを変更するには、次の手順を実行します。

1. MySQL サーバーを停止し、エラーなしでシャットダウンされることを確認します。
2. my.cnf を編集して、ログファイルの構成を変更します。ログファイルのサイズを変更するには、innodb_log_file_size を構成します。ログファイルの数を多くするには、innodb_log_files_in_group を構成します。
3. MySQL サーバーを再起動します。

InnoDB で innodb_log_file_size が Redo ログファイルのサイズと異なることが検出された場合は、ログチェックポイントが書き込まれ、古いログファイルが閉じられてから削除され、リクエストされたサイズで新しいファイルが作成され、その新しいログファイルが開かれます。

InnoDB テーブルスペースのサイズの増加

InnoDB システムテーブルスペースのサイズを大きくするもっとも簡単な方法は、最初から自動拡張として構成することです。テーブルスペース定義内の最後のデータファイルに autoextend 属性を指定します。これにより、InnoDB が領域を使い果たすと、そのファイルのサイズが自動的に 8M バイトずつ増加します。innodb_autoextend_increment システム変数の値を設定すると、増分のサイズを変更できます。このサイズは、M バイト単位で測定されます。

別のデータファイルを追加すると、システムテーブルスペースを定義された量だけ拡大できます。

1. MySQL サーバーをシャットダウンします。
2. 以前の最後のデータファイルが autoextend というキーワードを使用して定義されている場合は、実際に増加した大きさに基づいて、固定サイズが使用されるようにその定義を変更します。データファイルのサイズをチェックし、それを 1024 × 1024 バイト (= 1M バイト) にもっとも近い倍数に丸め、この丸められたサイズを innodb_data_file_path に明示的に指定します。
3. 新しいデータファイルを innodb_data_file_path の末尾に追加します。これにより、オプションでそのファイルが自動拡張になります。innodb_data_file_path で自動拡張として指定できるのは、最後のデータファイルのみです。
4. MySQL サーバーを再起動します。

たとえば、このテーブルスペースには、ibdata1 という 1 つの自動拡張データファイルしか存在しません。

```
innodb_data_home_dir =
innodb_data_file_path = /ibdata/ibdata1:10M:autoextend
```

このデータファイルが、時間をかけて 988M バイトまで増加したと仮定します。次に、固定サイズが使用されるように元のデータファイルを変更し、新しい自動拡張データファイルを追加したあとの構成行を示します。

```
innodb_data_home_dir =
innodb_data_file_path = /ibdata/ibdata1:988M;/disk2/ibdata2:50M:autoextend
```

新しいデータファイルをシステムテーブルスペース構成に追加する際に、そのファイル名が既存のファイルを参照していないことを確認してください。InnoDB は、サーバーの起動時にファイルを作成し、初期化します。

InnoDB テーブルスペースのサイズの減少

現在は、システムテーブルスペースからデータファイルを削除できません。システムテーブルスペースのサイズを小さくするには、次の手順を使用します。

1. mysqldump を使用して、MySQL データベース内に配置されている InnoDB テーブルを含む、すべての InnoDB テーブルをダンプします。5.6 の時点では、MySQL データベース内に 5 つの InnoDB テーブルが含まれています。

```
mysql> select table_name from information_schema.tables where table_schema='mysql' and engine='InnoDB';
+----------------------+
| table_name           |
+----------------------+
| innodb_index_stats   |
| innodb_table_stats   |
| slave_master_info    |
| slave_relay_log_info |
| slave_worker_info    |
+----------------------+
5 rows in set (0.00 sec)
```

2. サーバーを停止します。
3. ibdata および ib_log ファイルを含む、すべての既存のテーブルスペースファイル (*.ibd) を削除します。MySQL データベース内に配置されているテーブルの *.ibd ファイルも忘れずに削除してください。
4. InnoDB テーブルのすべての .frm ファイルを削除します。
5. 新しいテーブルスペースを構成します。
6. サーバーを再起動します。

7. ダンプファイルをインポートします。

> 注記
>
> データベースで InnoDB エンジンのみが使用されている場合は、すべてのデータベースをダンプし、サーバーを停止し、すべてのデータベースおよび InnoDB のログファイルを削除し、サーバーを再起動し、ダンプファイルをインポートした方が簡単な可能性があります。

14.5.8 共有テーブルスペースでの RAW ディスクパーティションの使用

InnoDB のシステムテーブルスペースでは、データファイルとして RAW ディスクパーティションを使用できます。この方法を使用すると、ファイルシステムのオーバーヘッドが発生せずに、Windows 上および一部の Linux と Unix 上でバッファーに入れられない I/O が有効になります。RAW パーティションを使用する場合と使用しない場合でテストを実行して、この変更によって実際にシステム上のパフォーマンスが改善されるかどうかを確認します。

RAW ディスクパーティションを使用する場合は、MySQL サーバーを実行しているユーザー ID がそのパーティションに対する読み取り権限および書き込み権限を持っていることを確認します。たとえば、mysql ユーザーとしてサーバーを実行する場合は、そのパーティションが mysql によって読み取り可能および書き込み可能である必要があります。--memlock オプションを付けてサーバーを実行する場合は、サーバーを root として実行する必要があるため、パーティションが root によって読み取り可能および書き込み可能である必要があります。

次で説明する手順には、オプションファイルの変更が伴います。追加情報については、セクション4.2.6「オプションファイルの使用」を参照してください。

Linux および Unix システムでの RAW ディスクパーティションの割り当て

1. 新しいデータファイルを作成する際は、innodb_data_file_path オプションのデータファイルサイズの直後に、newraw というキーワードを指定します。パーティションは、少なくとも指定したサイズと同じである必要があります。ディスク指定の 1M バイトは通常 1,000,000 バイトを意味するのに対して、InnoDB 内の 1M バイトは 1024 × 1024 バイトであることに注意してください。

```
[mysqld]
innodb_data_home_dir=
innodb_data_file_path=/dev/hdd1:3Gnewraw;/dev/hdd2:2Gnewraw
```

2. サーバーを再起動します。InnoDB によって newraw キーワードが認識され、新しいパーティションが初期化されます。ただし、まだ InnoDB テーブルを作成したり変更したりしないでください。そうしなければ、サーバーを次に再起動したときに InnoDB によってパーティションが再初期化され、変更がすべて失われます。(安全策として、InnoDB では、newraw を含むパーティションが指定されたときにユーザーがデータを更新することが回避されます。)

3. InnoDB によって新しいパーティションが初期化されたら、サーバーを停止し、データファイルの指定で newraw を raw に変更します。

```
[mysqld]
innodb_data_home_dir=
innodb_data_file_path=/dev/hdd1:3Graw;/dev/hdd2:2Graw
```

4. サーバーを再起動します。これにより、InnoDB で変更を行うことが許可されます。

Windows での RAW ディスクパーティションの割り当て

Windows システムでは、Linux および Unix システムで説明したものと同じ手順および付随するガイドラインが適用されます。ただし、Windows では innodb_data_file_path の設定がわずかに異なります。

1. 新しいデータファイルを作成する際は、innodb_data_file_path オプションのデータファイルサイズの直後に、newraw というキーワードを指定します。

```
[mysqld]
innodb_data_home_dir=
innodb_data_file_path=//./D::10Gnewraw
```

   //./ は、物理ドライブにアクセスするための Windows の構文 \\.\ に対応しています。前述の例では、D: がパーティションのドライブ文字です。

2. サーバーを再起動します。InnoDB によって newraw キーワードが認識され、新しいパーティションが初期化されます。

3. InnoDB によって新しいパーティションが初期化されたら、サーバーを停止し、データファイルの指定で newraw を raw に変更します。

```
[mysqld]
innodb_data_home_dir=
innodb_data_file_path=//./D::10Graw
```

4. サーバーを再起動します。これにより、InnoDB で変更を行うことが許可されます。

14.6 InnoDB テーブルの管理

14.6.1 InnoDB テーブルの作成

InnoDB テーブルを作成するには、特別な句を付けずに CREATE TABLE ステートメントを使用します。以前は、ENGINE=InnoDB 句が必要でしたが、InnoDB がデフォルトのストレージエンジンとなったため必要なくなりました。(デフォルトのストレージエンジンが MyISAM である MySQL 5.1 以前が実行されているサーバー上で、mysqldump またはレプリケーションを使用して CREATE TABLE ステートメントを再現する予定がある場合は、その句を引き続き使用できます。)

```
-- Default storage engine = InnoDB.
CREATE TABLE t1 (a INT, b CHAR (20), PRIMARY KEY (a));
-- Backward-compatible with older MySQL.
CREATE TABLE t2 (a INT, b CHAR (20), PRIMARY KEY (a)) ENGINE=InnoDB;
```

innodb_file_per_table の設定に応じて、InnoDB ではシステムテーブルスペース内またはテーブルごとの別個のテーブルスペース (.ibd ファイルで表されます) 内に、各テーブルおよび関連付けられた主キーインデックスが作成されます。MySQL では、MySQL データベースディレクトリの下にある test ディレクトリ内に、t1.frm および t2.frm ファイルが作成されます。内部的に、InnoDB 独自のデータディクショナリにテーブルのエントリが追加されます。このエントリには、データベース名が含まれます。たとえば、t1 テーブルが作成されるデータベースが test である場合、エントリは 'test/t1' となります。つまり、ほかのいくつかのデータベースに同じ名前 t1 のテーブルを作成でき、InnoDB 内部でテーブル名の競合は発生しません。

これらのテーブルのプロパティーを表示するには、SHOW TABLE STATUS ステートメントを発行します。

```
SHOW TABLE STATUS FROM test LIKE 't%' \G;
```

ステータスの出力には、これらの 1 番目のテーブルの行フォーマットプロパティーが「Compact」であることが表示されます。この設定は基本的な実験には適していますが、もっとも強力な InnoDB のパフォーマンス機能を活用するには、すぐに「Dynamic」や「Compressed」などのその他の行フォーマットの使用に進んでください。これらの値を使用するには、最初に少しの設定が必要です。

```
set global innodb_file_per_table=1;
set global innodb_file_format=barracuda;
CREATE TABLE t3 (a INT, b CHAR (20), PRIMARY KEY (a)) row_format=dynamic;
CREATE TABLE t4 (a INT, b CHAR (20), PRIMARY KEY (a)) row_format=compressed;
```

必ず、InnoDB テーブルごとに主キーを設定し、次のようなカラム (複数の場合あり) を指定してください。

- もっとも重要なクエリーで参照される。
- ブランクのままになっていない。
- 重複する値がない。
- 挿入後に値が変更されるとしても、きわめてまれである。

たとえば、人に関する情報を含むテーブルでは、複数の人が同じ名前を持つ可能性もあり、名字をブランクにしたり、名前を変更したりする人もいるため、(名、姓) 上には主キーを作成しません。制約が非常に多く、主キーとして使用する明確なカラムセットがないことも多い場合には、主キーの全部または一部として機能する数値 ID の新しいカラムを作成してください。行が挿入されると自動的に昇順の値が入力されるように、自動インクリメントカラムを宣言できます。

```
-- The value of ID can act like a pointer between related items in different tables.
CREATE TABLE t5 (id INT AUTO_INCREMENT, b CHAR (20), PRIMARY KEY (id));
-- The primary key can consist of more than one column. Any autoinc column must come first.
CREATE TABLE t6 (id INT AUTO_INCREMENT, a INT, b CHAR (20), PRIMARY KEY (id,a));
```

主キーを定義しなくてもテーブルは正常に機能しますが、主キーは多くのパフォーマンス要素に関連し、大きなテーブルや頻繁に使用されるテーブルにとって重要な設計要素です。常に CREATE TABLE ステートメントで指定することを習慣にしてください。(テーブルを作成し、データをロードしてから、ALTER TABLE を実行してあとで主キーを追加する場合の操作は、テーブルの作成時に主キーを定義するよりも大幅に時間がかかります。)

14.6.2 別のマシンへの InnoDB テーブルの移動またはコピー

このセクションでは、一部またはすべての InnoDB テーブルを別のサーバーに移動またはコピーするための方法について説明します。たとえば、MySQL インスタンス全体をより大規模で高速なサーバーに移動したり、MySQL インスタンス全体のクローンを新しいレプリケーションスレーブサーバーに作成したり、アプリケーションを開発およびテストするために各テーブルを別のサーバーにコピーしたり、レポートを生成するためにデータウェアハウスサーバーにコピーしたりする場合があります。

InnoDB テーブルを移動またはコピーするための方法は、次のとおりです。

- Transportable Tablespaces
- MySQL Enterprise Backup
- Copying Data Files (Cold Backup Method)
- Export and Import (mysqldump)

小文字の名前を使用したプラットフォーム間の移動またはコピー

Windows 上の InnoDB では常に、データベース名およびテーブル名が内部的に小文字で格納されます。バイナリ形式のデータベースを Unix から Windows に、または Windows から Unix に移動するには、すべてのデータベースおよびテーブルを小文字の名前を使用して作成します。これを実現する便利な方法は、データベースやテーブルを作成する前に、my.cnf または my.ini ファイルの [mysqld] セクションに次の行を追加することです。

```
[mysqld]
lower_case_table_names=1
```

トランスポータブルテーブルスペース

MySQL 5.6.6 で導入されたトランスポータブルテーブルスペース機能では、あるサーバーインスタンスから別のサーバーインスタンスにコピーするように InnoDB を準備する際に、FLUSH TABLES ... FOR EXPORT が使用されます。この機能を使用するには、各 InnoDB テーブルが独自のテーブルスペースを持つように、innodb_file_per_table を ON に設定した状態で InnoDB テーブルを作成する必要があります。使用法については、セクション14.5.5「テーブルスペースの別のサーバーへのコピー (トランスポータブルテーブルスペース)」を参照してください。

MySQL Enterprise Backup

MySQL Enterprise Backup 製品を使用すると、実行中の MySQL データベース (InnoDB および MyISAM テーブルを含む) を、データベースの整合性のあるスナップショットを生成しながら、操作の中断を最小限に抑えてバックアップできます。MySQL Enterprise Backup が InnoDB テーブルをコピーしている間は、InnoDB テーブルと MyISAM テーブルの両方に対する読み取りと書き込みを続行できます。MyISAM およびその他の InnoDB 以外のテーブルのコピー中は、これらのテーブルに対する (書き込みではなく) 読み取りが許可されます。さらに、MySQL Enterprise Backup では、圧縮バックアップファイルを作成したり、InnoDB テーブルのサブセットをバックアップしたりすることもできます。MySQL のバイナリログと組み合わせると、ポイントインタイムリカバリを実行できます。MySQL Enterprise Backup は、MySQL Enterprise サブスクリプションの一部として含まれています。

MySQL Enterprise Backup についての詳細は、セクション25.2「MySQL Enterprise Backup」を参照してください。

データファイルのコピー (コールドバックアップ方式)

単に、セクション14.16「InnoDB のバックアップとリカバリ」の「コールドバックアップ」で一覧表示した関連ファイルをすべてコピーするだけで、InnoDB データベースを移動できます。

MyISAM データファイルと同様に、InnoDB のデータファイルとログファイルにも、同じ浮動小数点数形式を持つすべてのプラットフォーム上でのバイナリ互換性があります。浮動小数点形式が異なっている場合でも、テーブル内で FLOAT または DOUBLE データ型を使用していなければ、手順は同じです。単に、関連するファイルをコピーするだけです。

.ibd ファイルの移植性に関する考慮事項

.ibd ファイルを移動またはコピーする際は、ソースシステムと宛先システムでデータベースディレクトリ名を同じにする必要があります。データベース名は、InnoDB の共有テーブルスペース内に格納されているテーブル定義に含まれています。テーブルスペースファイル内に格納されているトランザクション ID および ログシーケンス番号も、データベース間で異なります。

あるデータベースから別のデータベースに .ibd ファイルとそれに関連付けられたテーブルを移動するには、RENAME TABLE ステートメントを使用します。

```
RENAME TABLE db1.tbl_name TO db2.tbl_name;
```

.ibd ファイルの「クリーンな」バックアップがある場合は、次のように、そのバックアップが生成された MySQL インストールにリストアできます。

1. .ibd ファイルをコピーすると、テーブルスペース内に格納されたテーブル ID が変更されるため、それ以降はテーブルの削除または切り捨ては実行されなかったはずです。
2. 次の ALTER TABLE ステートメントを発行して、現在の .ibd ファイルを削除します。

   ```
   ALTER TABLE tbl_name DISCARD TABLESPACE;
   ```

3. バックアップ .ibd ファイルを適切なデータベースディレクトリにコピーします。
4. 次の ALTER TABLE ステートメントを発行して、このテーブルで新しい .ibd ファイルを使用するように InnoDB に指示します。

   ```
   ALTER TABLE tbl_name IMPORT TABLESPACE;
   ```

   > **注記**
   >
   > ALTER TABLE ... IMPORT TABLESPACE 機能は、インポートされたデータに対して外部キー制約を課しません。

このコンテキストでは、「クリーンな」 .ibd バックアップファイルとは、次の要件を満たすファイルです。

- .ibd ファイル内には、トランザクションによってコミットされていない変更はありません。
- .ibd ファイル内にマージされていない挿入バッファーエントリはありません。
- パージによって、.ibd ファイルから削除マークが付けられたすべてのインデックスレコードが削除されました。
- mysqld によって、.ibd ファイルの変更されたページがすべてバッファープールからファイルにフラッシュされました。

次の方法を使用すると、クリーンなバックアップ .ibd ファイルを作成できます。

1. mysqld サーバーからのすべてのアクティビティーを停止し、すべてのトランザクションをコミットします。
2. SHOW ENGINE INNODB STATUS でデータベース内にアクティブなトランザクションがないことが表示され、InnoDB のメインスレッドステータスが「Waiting for server activity」になるまで待機します。これにより、.ibd ファイルのコピーを作成できるようになります。

.ibd ファイルのクリーンなコピーを作成するためのもう 1 つの方法は、MySQL Enterprise Backup 製品を使用することです。

1. MySQL Enterprise Backup を使用して、InnoDB インストールをバックアップします。
2. 2 番目の mysqld サーバーをバックアップ上で起動します。そのサーバーで、バックアップ内の .ibd ファイルがクリーンアップされます。

エクスポートとインポート (mysqldump)

mysqldump を使用すると、あるマシン上でテーブルをダンプしてから、別のマシン上でそのダンプファイルをインポートできます。この方式を使用すれば、形式が異なっているかどうかや、テーブルに浮動小数点データが含まれているかどうかは関係ありません。

インポートトランザクションで生成される巨大なロールバックセグメント用の領域がテーブルスペースに十分にあると仮定すれば、この方式のパフォーマンスを向上させる方法の 1 つは、データのインポート時に自動コミットモードをオフにすることです。コミットは、テーブル全体またはテーブルのセグメントをインポートしたあとでのみ行なってください。

14.6.3 トランザクションを使用した DML 操作のグループ化

デフォルトでは、MySQL サーバーへの接続は、自動コミットモードが有効になっている状態で開始されるため、SQL ステートメントは実行するたびに自動的にコミットされます。一連の DML ステートメントを発行し、

すべてまとめてコミットまたはロールバックすることが標準操作となっているほかのデータベースシステムの使用経験がある場合は、この操作モードに馴染みがないかもしれません。

複数ステートメントのトランザクションを使用するには、SQL ステートメント SET autocommit = 0 を使用して自動コミットをオフにして、必要に応じて COMMIT または ROLLBACK を使用して各トランザクションを終了します。自動コミットをオンのままにするには、START TRANSACTION を使用して各トランザクションを開始し、COMMIT または ROLLBACK を使用して終了します。次の例は 2 つのトランザクションを表しています。1 番目はコミットされ、2 番目はロールバックされています。

```
shell> mysql test

mysql> CREATE TABLE customer (a INT, b CHAR (20), INDEX (a));
Query OK, 0 rows affected (0.00 sec)
mysql> -- Do a transaction with autocommit turned on.
mysql> START TRANSACTION;
Query OK, 0 rows affected (0.00 sec)
mysql> INSERT INTO customer VALUES (10, 'Heikki');
Query OK, 1 row affected (0.00 sec)
mysql> COMMIT;
Query OK, 0 rows affected (0.00 sec)
mysql> -- Do another transaction with autocommit turned off.
mysql> SET autocommit=0;
Query OK, 0 rows affected (0.00 sec)
mysql> INSERT INTO customer VALUES (15, 'John');
Query OK, 1 row affected (0.00 sec)
mysql> INSERT INTO customer VALUES (20, 'Paul');
Query OK, 1 row affected (0.00 sec)
mysql> DELETE FROM customer WHERE b = 'Heikki';
Query OK, 1 row affected (0.00 sec)
mysql> -- Now we undo those last 2 inserts and the delete.
mysql> ROLLBACK;
Query OK, 0 rows affected (0.00 sec)
mysql> SELECT * FROM customer;
+------+--------+
| a    | b      |
+------+--------+
|   10 | Heikki |
+------+--------+
1 row in set (0.00 sec)
mysql>
```

クライアント側言語でのトランザクション

PHP、Perl DBI、JDBC、ODBC などの API または MySQL の標準 C 呼び出しインタフェースでは、COMMIT などのトランザクション制御ステートメントを SELECT や INSERT などのその他の SQL ステートメントと同様の文字列として、MySQL サーバーに送信できます。一部の API では、別個の特別なトランザクションコミットおよびロールバックの関数やメソッドも提供されています。

14.6.4 MyISAM から InnoDB へのテーブルの変換

信頼性および拡張性を改善するために、既存のテーブルを使用するアプリケーションを InnoDB に変換する場合は、次のガイドラインおよびヒントを使用します。このセクションでは、このようなテーブルの大部分が当初は、以前のデフォルトの MyISAM であったことが前提となっています。

MyISAM のメモリー使用量の減少、InnoDB のメモリー使用量の増加

MyISAM テーブルから移行するときに、結果をキャッシュする際に必要でなくなったメモリーが解放されるように、key_buffer_size 構成オプションの値を小さくします。InnoDB テーブル用のキャッシュメモリー割り当てと同様の役割を担う innodb_buffer_pool_size 構成オプションの値を大きくします。InnoDB のバッファープールには、テーブルデータとインデックスデータの両方がキャッシュされるため、クエリーの検索速度を上げることと、再使用するためにクエリー結果をメモリー内に保持することの 2 つ役割があります。

- このオプションには、できるかぎり多くのメモリー (多くの場合、最大でサーバー上の物理メモリーの 80% まで) を割り当てます。
- オペレーティングシステムでその他のプロセス用のメモリーが不足し、スワップが発生し始めた場合は、innodb_buffer_pool_size の値を小さくします。スワップとは、キャッシュメモリーの利点が大幅に減少するような負荷の高い操作です。
- innodb_buffer_pool_size 値が数ギガバイト以上である場合は、innodb_buffer_pool_instances の値を大きくすることを検討してください。これを行うと、同時に数多くの接続がキャッシュにデータを読み込む高負荷のサーバーで役立ちます。

- 高負荷のサーバーでは、クエリーキャッシュをオフにしてベンチマークを実行します。InnoDB のバッファープールでも同様の利点が得られるため、クエリーキャッシュを使用すると、不必要にメモリーが停止する可能性があります。

長すぎるまたは短すぎるトランザクションの監視

MyISAM テーブルではトランザクションがサポートされていないため、autocommit 構成オプションと、COMMIT および ROLLBACK ステートメントに多くの注意が払われていない可能性があります。これらのキーワードは、複数のセッションが並列して InnoDB テーブルの読み取りおよび書き込みを行うことを許可する際に重要となります。これにより、書き込み負荷の高いワークロードで十分な拡張性の利点が得られます。

トランザクションが開いている間は、トランザクションの開始時に見られるようなデータのスナップショットがシステムで保持されます。これにより、未処理のトランザクションが動作し続けている間に、システムで数百万行の挿入、更新、および削除が行われると、相当なオーバーヘッドが発生する可能性があります。そのため、動作時間が長すぎるトランザクションは回避するように注意してください。

- インタラクティブな実験で mysql セッションを使用している場合は、完了後に必ず、(変更を完了させる場合は) COMMIT、または (変更を取り消す場合は) ROLLBACK を実行します。誤ってトランザクションが長時間開いたままになることを回避するには、インタラクティブなセッションを長時間開いたままにせず、閉じてください。
- アプリケーション内の任意のエラーハンドラでも、不完全な変更の ROLLBACK が実行されるか、完了した変更の COMMIT が実行されることを確認します。
- INSERT、UPDATE、および DELETE 操作は、ほとんどの変更は正常にコミットされ、ロールバックはまれにしか発生しないという見込みで、COMMIT よりも前に InnoDB テーブルに書き込まれるため、ROLLBACK は比較的負荷の高い操作です。大量のデータを使用して実験する際は、多数の行に変更を加えてから、それらの変更をロールバックすることは回避してください。
- 一連の INSERT ステートメントを使用して大量のデータをロードする際は、トランザクションが数時間存続することを回避するために、定期的に結果の COMMIT を実行します。データウェアハウスでの一般的なロード操作では、何か問題が発生した場合に、ユーザーは ROLLBACK を行うのではなく、TRUNCATE TABLE を実行し、最初からやり直します。

前述のヒントを使用すると、長すぎるトランザクション中に無駄になる可能性のあるメモリーおよびディスク容量を節約できます。トランザクションが本来よりも短い場合は、過剰な I/O が問題となります。MySQL では、COMMIT が実行されるたびに、各変更が安全にディスクに記録されていることが確認されます。これには、多少の I/O が伴います。

- InnoDB テーブル上のほとんどの操作では、autocommit=0 の設定を使用するようにしてください。効率性の観点から見ると、これにより、多数の連続した INSERT、UPDATE、または DELETE ステートメントを発行したときの不要な I/O が回避されます。安全性の観点から見ると、これにより、mysql コマンド行またはアプリケーションの例外ハンドラに誤りがあった場合に、ROLLBACK ステートメントを発行することで、失ったデータや文字化けしたデータをリカバリできます。
- InnoDB テーブルに autocommit=1 を設定することが適している状況は、レポートの生成または統計の分析を行うために一連のクエリーを実行するときです。このような状況では、COMMIT または ROLLBACK に関連する I/O ペナルティーが発生せず、InnoDB は自動的に読み取り専用のワークロードを最適化できます。
- 関連する一連の変更を行う場合は、最後に 1 回 COMMIT を実行して、これらのすべての変更を一度に完了させます。たとえば、情報の関連部分を複数のテーブルに挿入する場合は、すべての変更を行なったあとに、COMMIT を 1 回実行します。また、連続する多数の INSERT ステートメントを実行する場合は、すべてのデータがロードされたあとに、COMMIT を 1 回実行します。何百万もの INSERT ステートメントを実行する場合は、一万または一千レコードごとに COMMIT を発行することで、巨大なトランザクションを分割することがあります。
- SELECT ステートメントでもトランザクションが開かれるため、インタラクティブな mysql セッションで一部のレポートを実行したり、クエリーをデバッグしたりしたあとは、COMMIT を発行するか、または mysql セッションを閉じます。

デッドロックを心配しすぎないこと

MySQL のエラーログまたは SHOW ENGINE INNODB STATUS の出力に、「デッドロック」に言及する警告メッセージが表示されることがあります。デッドロックは、恐ろしい響きの名前にもかかわらず、InnoDB テーブルにとっては重大な問題でなく、修正アクションは何も必要ありません。2 つのトランザクションが複数のテーブルを変更し、そのテーブルに別々の順序でアクセスし始めると、各トランザクションが相互に待機し合って、どち

らも処理できない状態に達する可能性があります。すぐに MySQL によって、この状況が検出され、「小さい方の」トランザクションが取り消され (ロールバックされ)、他方が処理できるようになります。

アプリケーションには、このように強制的に取り消されたトランザクションを再開するためのエラー処理ロジックが必要です。以前と同じ SQL ステートメントを再発行するときは、元のタイミングの問題は適用されません。他方のトランザクションがすでに完了したため一方を処理できるか、他方のトランザクションがまだ処理中で、これが完了するまで一方が待機しているかのいずれかです。

デッドロックの警告が常に発生する場合は、アプリケーションコードを再確認して、一貫性のある方法で SQL 操作を再指示したり、トランザクションを短くしたりすることがあります。innodb_print_all_deadlocks オプションを有効にしてテストすれば、SHOW ENGINE INNODB STATUS 出力の最後の警告だけでなく、MySQL のエラーログにもすべてのデッドロックの警告を表示できます。

ストレージレイアウトの計画

InnoDB テーブルから最高のパフォーマンスを引き出すために、ストレージレイアウトに関連する数多くのパラメータを調整できます。

頻繁にアクセスされ、重要なデータが保持されている大きな MyISAM テーブルを変換する際は、innodb_file_per_table、innodb_file_format、innodb_page_size 構成オプション、および CREATE TABLE ステートメントの ROW_FORMAT と KEY_BLOCK_SIZE 句を調査および検討してください。

初期の実験時に、もっとも重要となる設定は innodb_file_per_table です。新しい InnoDB テーブルを作成する前に、このオプションを有効にすると、InnoDB のシステムテーブルスペースファイルを使用して、すべての InnoDB データ用のディスク領域が永続的に割り当てられなくなります。innodb_file_per_table を有効にすると、DROP TABLE および TRUNCATE TABLE を発行することで、要求どおりにディスク領域が解放されます。

既存テーブルの変換

InnoDB を使用するように InnoDB 以外のテーブルを変換するには、ALTER TABLE を使用します。

```
ALTER TABLE table_name ENGINE=InnoDB;
```

> **重要**
>
> mysql データベース内の MySQL システムテーブル (user や host など) を InnoDB 型に変換しないでください。これはサポートされていない操作です。システムテーブルの型は、必ず MyISAM にする必要があります。

テーブル構造のクローニング

古いテーブルと新しいテーブルを切り替える前に並列してテストする際に、ALTER TABLE 変換を行うのではなく、MyISAM テーブルのクローンである InnoDB テーブルを作成することがあります。

同じカラムとインデックスの定義を持つ空の InnoDB テーブルを作成します。show create table table_name\G を使用して、使用される完全な CREATE TABLE ステートメントを表示します。ENGINE 句を ENGINE=INNODB に変更します。

既存データの転送

前のセクションで示したように、作成された空の InnoDB テーブルに大量のデータを転送するには、INSERT INTO innodb_table SELECT * FROM myisam_table ORDER BY primary_key_columns を使用して行を挿入します。

データを挿入したあとに、InnoDB テーブル用のインデックスを作成することもできます。従来、新しいセカンダリインデックスを作成することは、InnoDB にとって低速な操作でしたが、現在は、インデックスの作成ステップで比較的小さいオーバーヘッドでデータがロードされたあとに、インデックスを作成できるようになりました。

副キー上に UNIQUE 制約がある場合は、インポート操作中に一意性チェックを一時的にオフにすることで、テーブルインポートの速度を上げることができます。

```
SET unique_checks=0;
... import operation ...
SET unique_checks=1;
```

大きいテーブルの場合、InnoDB はその挿入バッファーを使用して、一括してセカンダリインデックスレコードを書き込むことができるため、これにより、大量のディスク I/O が節約されます。データに重複キーが含まれない

ようにします。unique_checks では、ストレージエンジンが重複キーを無視することが許可されていますが、必須ではありません。

挿入プロセスをより適切に制御するために、大きなテーブルを分割して挿入することがあります。

```
INSERT INTO newtable SELECT * FROM oldtable
  WHERE yourkey > something AND yourkey <= somethingelse;
```

すべてのレコードが挿入されたあとに、テーブルの名前を変更できます。

ディスク I/O を削減するには、大きなテーブルの変換時に、最大で物理メモリーの 80% まで InnoDB バッファープールのサイズを大きくします。InnoDB ログファイルのサイズを大きくすることもできます。

ストレージ要件

すでに説明したように、この時点で、すでに innodb_file_per_table オプションが有効になっている必要があります。これにより、InnoDB テーブル内にデータの複数のコピーを一時的に作成している場合は、あとで不要なテーブルを削除することで、そのディスク領域をすべてリカバリできます。

MyISAM テーブルを直接変換するのか、クローンの InnoDB テーブルを作成するのかには関係なく、プロセス中に古いテーブルと新しいテーブルの両方を保持するのに十分なディスク領域があることを確認します。InnoDB テーブルには、MyISAM テーブルよりも多くのディスク領域が必要です。ALTER TABLE 操作によって領域が使い果たされると、ロールバックが開始されますが、ディスクバウンドの場合は、数時間かかる可能性があります。挿入の場合、InnoDB はバッチ内のインデックスにセカンダリインデックスレコードをマージする際に、挿入バッファーを使用します。これにより、大量のディスク I/O が節約されます。ロールバックでは、このようなメカニズムは使用されません。ロールバックは挿入よりも、30 倍長い時間がかかる可能性があります。

ランナウェイロールバックの場合は、データベースに貴重なデータがなければ、何百万ものディスク I/O 操作が完了するまで待機するのではなく、データベースプロセスを強制終了することをお勧めします。完全な手順については、セクション14.19.2「InnoDB のリカバリの強制的な実行」を参照してください。

テーブルごとの主キーの慎重な選択

PRIMARY KEY 句は、MySQL クエリーのパフォーマンスや、テーブルおよびインデックス用の領域使用量に影響を与える重要な要素です。おそらく、金融機関に電話をかけ、口座番号を求められた経験があるでしょう。その番号を持っていない場合は、自分自身を「一意に識別する」ために、多種多様な情報が求められます。主キーは、テーブル内の情報を問い合わせたり、変更したりする際に、すぐに仕事に取りかかるための一意の口座番号のようなものです。テーブル内のすべて行が主キー値を持っている必要があり、2 つの行が同じ主キー値を持つことはできません。

次に、主キーに関するいくつかのガイドラインに続き、さらに詳細な説明を示します。

- テーブルごとに PRIMARY KEY を宣言します。一般に、単一の行を検索するときに参照される WHERE 句内のカラムの中で、もっとも重要なものです。
- あとで ALTER TABLE ステートメントを使用して追加するのではなく、元の CREATE TABLE ステートメントで PRIMARY KEY 句を宣言します。
- カラムとそのデータ型は慎重に選択してください。文字または文字列のカラムよりも、数値のカラムを優先してください。
- 別の安定していて、一意で、非 NULL で、数値のカラムが使用できない場合は、自動インクリメントカラムを使用することを検討してください。
- 主キーカラムの値が変更されたかどうかが疑わしい場合にも、自動インクリメントは適切な選択です。主キーカラムの値を変更することは、負荷の高い操作であり、テーブル内および各セカンダリインデックス内でデータの再編成が伴う可能性があります。

主キーがまだ存在しないテーブルには、追加することを検討してください。計画されたテーブルの最大サイズに基づいて、現実的な最小の数値型を使用します。これにより、各行をわずかにコンパクトにすることができ、大きなテーブル用に相当な領域を節約できます。主キー値は、セカンダリインデックスが入力されるたびに繰り返されるため、テーブルが任意のセカンダリインデックスを持っている場合は、領域の節約も倍増します。小さな主キーを使用すると、ディスク上のデータサイズが削減されることに加えて、より多くのデータをバッファープール内に収容できるため、すべての種類の操作の速度が上がり、並列性が改善されます。

すでにテーブルの多少長いカラム (VARCHAR など) 上に主キーが存在する場合は、そのカラムがクエリーで参照されていなくても、新しい符号なし AUTO_INCREMENT カラムを追加し、主キーをそのカラムに切り替えることを検討してください。このような設計の変更によって、セカンダリインデックス内の相当な領域を節約できま

す。以前の主キーカラムを UNIQUE NOT NULL として指定すると、PRIMARY KEY 句と同じ制約を強制的に適用できます (つまり、これらのすべてのカラムにわたって重複する値や NULL 値を回避できます)。

関連する情報を複数のテーブルに分散させる場合は、一般に各テーブルで、その主キー用に同じカラムが使用されます。たとえば、人事部のデータベースには複数のテーブルが含まれ、各テーブルには従業員番号の主キーが含まれている場合があります。営業部のデータベースには、顧客番号の主キーを含むテーブルや、注文番号の主キーを含むテーブルが含まれている場合があります。主キーを使用した検索は非常に高速であるため、このようなテーブルには効率的な結合クエリーを構築できます。

PRIMARY KEY 句を完全に削除すると、MySQL によって自動的に非表示の主キーが作成されます。これは、必要以上に長くなる可能性のある 6 バイトの値であるため、領域が無駄になります。これは非表示であるため、クエリーで参照できません。

アプリケーションのパフォーマンスに関する考慮事項

InnoDB の追加の信頼性および拡張性機能を使用するには、同等の MyISAM テーブルよりも多くのディスクストレージが必要となります。領域の使用率を改善し、結果セットを処理する際の I/O およびメモリーの消費を削減し、インデックス検索を効率的に使用するクエリーの最適化計画を改善するために、カラムおよびインデックスの定義をわずかに変更することがあります。

主キーに数値の ID カラムを設定する場合 (特に、結合クエリーの場合) は、その値を使用して、その他の任意のテーブル内の関連する値と相互参照します。たとえば、入力として国名を受け入れ、同じ名前を検索するクエリーを実行するのではなく、国 ID を確認するための検索を 1 回実行してから、複数のテーブルにわたって関連情報を検索するための別のクエリー (または 1 回の結合クエリー) を実行します。顧客番号またはカタログ項目番号を数字の文字列として格納すると、数バイトを使い果たす可能性があるため、その代わりに、格納およびクエリー用に数値の ID に変換します。4 バイトの符号なし INT カラムでは、40 億を超える項目 (アメリカ合衆国での billion の意味: 10 億) にインデックスを付けることができます。さまざまな整数型の範囲については、セクション11.2.1「整数型 (真数値) - INTEGER、INT、SMALLINT、TINYINT、MEDIUMINT、BIGINT」を参照してください。

InnoDB テーブルに関連付けられたファイルの理解

InnoDB ファイルを使用する際は、MyISAM ファイルよりも多くの注意および計画が必要となります。

- InnoDB のシステムテーブルスペースを表す ibdata ファイルは削除しないでください。
- あるサーバーから別のサーバーに InnoDB テーブルをコピーするには、まず FLUSH TABLES ... FOR EXPORT ステートメントを発行してから、table_name.ibd ファイルとともに table_name.cfg ファイルをコピーする必要があります。

14.6.5 InnoDB での AUTO_INCREMENT 処理

InnoDB では、AUTO_INCREMENT カラムを含むテーブルに行を挿入する SQL ステートメントの拡張性およびパフォーマンスが大幅に改善される最適化が提供されています。InnoDB テーブルで AUTO_INCREMENT メカニズムを使用するには、テーブルで最大カラム値を取得するインデックス SELECT MAX(ai_col) ルックアップと同等の操作を実行できるように、AUTO_INCREMENT カラム ai_col をインデックスの一部として定義する必要があります。一般に、これはカラムをどこかのテーブルインデックスの 1 番目のカラムにすることで実現されます。

このセクションでは、InnoDB の自動インクリメントロックの元の (「従来の」) 実装に関する背景情報を提供し、構成可能なロックメカニズムについて説明し、このメカニズムを構成するためのパラメータを示し、その動作やレプリケーションとの相互作用について説明します。

14.6.5.1 従来の InnoDB の自動インクリメントロック

InnoDB の自動インクリメント処理の元の実装では、ステートメントベースレプリケーションや特定のリカバリシナリオでバイナリログを使用すると発生する問題を回避するために、次のような方針が使用されています。

InnoDB テーブルに AUTO_INCREMENT カラムを指定すると、InnoDB データディクショナリ内のテーブルハンドルに、カラムに新しい値を割り当てる際に使用される自動インクリメントカウンタと呼ばれる特別なカウンタが含まれます。このカウンタは、ディスク上には格納されず、メインメモリー内にのみ格納されます。

InnoDB では、ai_col という名前の AUTO_INCREMENT カラムを含むテーブル t に自動インクリメントカウンタを初期化するために、次のようなアルゴリズムが使用されます。サーバーの起動のあとで、テーブル t への最初の挿入をするために、InnoDB は次のステートメントと同等なものを実行します。

```
SELECT MAX(ai_col) FROM t FOR UPDATE;
```

InnoDB は、ステートメントで取得された値を増分し、それをテーブルのカラムおよび自動インクリメントカウンタに割り当てます。デフォルトでは、値が 1 ずつ増分されます。このデフォルトは、auto_increment_increment 構成の設定でオーバーライドできます。

テーブルが空の場合、InnoDB では値 1 が使用されます。このデフォルトは、auto_increment_offset 構成の設定でオーバーライドできます。

自動インクリメントカウンタが初期化される前に、SHOW TABLE STATUS ステートメントで t テーブルが調査される場合は、InnoDB によって値は初期化されますが、増分されず、後続の挿入で使用するために格納されます。この初期化では、テーブル上で通常の排他ロック読み取りが使用され、そのロックはトランザクションの最後まで存続します。

InnoDB は、新たに作成されたテーブル用に自動インクリメントカウンタを初期化するときと同じ手順に従います。

自動インクリメントカウンタが初期化されたあとに、AUTO_INCREMENT カラムの値を明示的に指定しない場合は、InnoDB によってカウンタが増分され、新しい値がカラムに割り当てられます。カラム値を明示的に指定する行を挿入するときに、その値が現在のカウンタ値よりも大きい場合は、カウンタが指定されたカラム値に設定されます。

ユーザーが INSERT で AUTO_INCREMENT カラムに NULL または 0 を指定すると、InnoDB では、値が指定されなかった場合と同様にその行が処理され、新しい値が生成されます。

カラムに負の値を割り当てる場合や、値が指定された整数型に格納できる最大整数よりも大きくなる場合は、自動インクリメントメカニズムの動作が定義されません。

自動インクリメントカウンタにアクセスするときに、InnoDB では、トランザクションの最後まででではなく、現在の SQL ステートメントの最後まで存続する特別なテーブルレベルの AUTO-INC ロックが使用されます。AUTO_INCREMENT カラムを含むテーブルへの挿入の並列性を改善するために、特別なロック解放方針が導入されました。それにもかかわらず、2 つのトランザクションが同時に AUTO-INC ロックを同じテーブル上で持つことはできません。これにより、AUTO-INC ロックが長時間保持されると、パフォーマンスが影響を受ける可能性があります。これは、あるテーブルから別のテーブルにすべての行を挿入する INSERT INTO t1 ... SELECT ... FROM t2 などのステートメントの場合に発生する可能性があります。

InnoDB では、サーバーが実行されていれば、インメモリーの自動インクリメントカウンタが使用されます。前に説明したように、サーバーが停止して再起動されると、テーブルへの最初の INSERT 時に、InnoDB によってテーブルごとにカウンタが再初期化されます。

サーバーが再起動されると、CREATE TABLE および ALTER TABLE ステートメントの AUTO_INCREMENT = N テーブルオプションの効果も取り消されます。このオプションを InnoDB テーブルで使用すると、初期のカウンタの値を設定したり、現在のカウンタ値を変更したりできます。

カウンタを使用して数値が生成されたトランザクションをロールバックすると、AUTO_INCREMENT カラムに割り当てられた一連の値でギャップが見つかることがあります。

14.6.5.2 構成可能な InnoDB の自動インクリメントロック

前のセクションで説明したように、InnoDB では AUTO_INCREMENT カラムを含むテーブルへの挿入を行う際に、AUTO-INC ロックと呼ばれる特殊なテーブルレベルロックが使用されます。このロックは通常、指定された一連の INSERT ステートメントに予測可能かつ繰り返し可能な順序で自動インクリメント番号が割り当てられるように、(トランザクションが終了するまででではなく) ステートメントが終了するまで保持されます。

ステートメントベースのレプリケーションの場合、これは、ある SQL ステートメントがスレーブサーバーで複製される際に、自動インクリメントカラムでマスターサーバーと同じ値が使用されることを意味します。複数の INSERT ステートメントの実行結果は決定的であり、マスター上と同じデータがスレーブで再生成されます。複数の INSERT ステートメントによって生成された自動インクリメント値がインターリーブされた場合は、2 つの並列 INSERT ステートメントの結果は非決定的であり、ステートメントベースのレプリケーションを使用してスレーブサーバーに伝搬される際の信頼性も低くなる可能性があります。

この点が明確になるように、次のテーブルを使用する例を考えてみましょう。

```
CREATE TABLE t1 (
  c1 INT(11) NOT NULL AUTO_INCREMENT,
  c2 VARCHAR(10) DEFAULT NULL,
  PRIMARY KEY (c1)
) ENGINE=InnoDB;
```

実行中のトランザクションが 2 つ存在しており、それぞれ AUTO_INCREMENT カラムを含むテーブル内に行を挿入しているものとします。1 つのトランザクションは 1000 行を挿入する INSERT ... SELECT ステートメント

を使用しており、もう 1 つのトランザクションは 1 行を挿入する単純な INSERT ステートメントを使用しています。

```
Tx1: INSERT INTO t1 (c2) SELECT 1000 rows from another table ...
Tx2: INSERT INTO t1 (c2) VALUES ('xxx');
```

InnoDB は、Tx1 の INSERT ステートメント内の SELECT から取得される行数を事前に知ることができないため、そのステートメントの処理を進める際に、自動インクリメント値を一度に 1 つずつ割り当てます。ステートメントの終了まで保持されるテーブルレベルロックが存在しているため、ある時点で実行可能な INSERT ステートメントはテーブル t1 を参照している 1 つのステートメントだけであり、複数ステートメントによって自動インクリメント番号の生成がインターリーブされることはありません。Tx1 の INSERT ... SELECT ステートメントで生成される自動インクリメント値は連続した番号となり、Tx2 の INSERT ステートメントで使用される (単一の) 自動インクリメント値は、どちらのステートメントが先に実行されるかに応じて、Tx1 で使用されるすべての値よりも小さいか大きい値になります。

(ステートメントベースのレプリケーション使用時やリカバリシナリオで) バイナリログから再現する際に SQL ステートメントが同じ順番で実行されるかぎり、その結果は、Tx1 と Tx2 が最初に実行されたときと同じになります。したがって、ステートメントの終了まで保持されるテーブルレベルロックが存在することで、自動インクリメントを使用する INSERT ステートメントをステートメントベースのレプリケーションで安全に使用できるようになります。ただし、このようなロックでは、複数のトランザクションで挿入ステートメントが同時に実行されるときの並列性および拡張性が制限されます。

前述の例でテーブルレベルロックが存在しなかった場合、Tx2 の INSERT で使用される自動インクリメントカラムの値は、ステートメントが実際に実行されるタイミングに応じて変更されます。Tx1 の INSERT の (実行前や完了後ではなく) 実行中に、Tx2 の INSERT が実行された場合、その 2 つの INSERT ステートメントで割り当てられる具体的な自動インクリメント値は非決定的となり、実行するたびに値が異なる可能性があります。

InnoDB では、行数が事前にわかっている場合は、INSERT ステートメントのクラスに対してテーブルレベル AUTO-INC ロックが使用されることを回避できますが、ステートメントベースのレプリケーションの決定的な実行および安全性は、引き続き保持されます。さらに、リカバリまたはレプリケーションの一部として SQL ステートメントを再現する際にバイナリログを使用しない場合は、並列性およびパフォーマンスをさらに改善するために、テーブルレベル AUTO-INC ロックの使用を完全に除去できますが、ステートメントで割り当てられた自動インクリメント数のギャップが許可され、並列実行されるステートメントで割り当てられた数がインターリーブされる可能性があるという犠牲が伴います。

ステートメントの処理開始時点で挿入行数がわかっているような INSERT ステートメントでは、InnoDB はロックを一切使用せずに必要な数の自動インクリメント値をすばやく割り当てます。ただし、テーブルレベル AUTO-INC ロックをすでに保持している並列セッションが存在しない場合に限ります (その別のステートメントが処理中に自動インクリメント値を 1 つずつ割り当てるため)。より正確に言えば、このような INSERT ステートメントは、ステートメントの完了まででではなく、割り当て処理の期間だけ保持される相互排他ロック (軽量ロック) の制御下で自動インクリメント値を取得します。

この新しいロックスキームを使用すると、拡張性を大幅に改善できますが、元のメカニズムと比べて、自動インクリメント値が割り当てられる方法にわずかな相違が散見されます。InnoDB での自動インクリメントの動作を説明するために、次の説明でいくつかの用語を定義し、サーバーの起動時に設定できる innodb_autoinc_lock_mode 構成パラメータのさまざまな設定を使用した InnoDB の動作について説明します。自動インクリメントロックの動作説明のあとで、追加の注意事項について説明します。

まず、いくつかの定義を次に示します。

- 「INSERT のような」ステートメント

  INSERT、INSERT ... SELECT、REPLACE、REPLACE ... SELECT、LOAD DATA など、テーブル内に新しい行を生成するすべてのステートメントです。

- 「単純挿入」

  挿入行数を事前に (ステートメントの初期処理時に) 決定できるステートメントです。これには、ネストしたサブクエリーを持たない単一行および複数行の INSERT および REPLACE ステートメントが含まれますが、INSERT ... ON DUPLICATE KEY UPDATE は含まれません。

- 「一括挿入」

  挿入行数 (および必要な自動インクリメント値の数) が事前にわからないステートメントです。これには、INSERT ... SELECT、REPLACE ... SELECT、および LOAD DATA ステートメントが含まれますが、単純な INSERT は含まれません。InnoDB は各行を処理する際に、AUTO_INCREMENT カラムの新しい値を一度に 1 つずつ割り当てます。

- 「混在モード挿入」

これらは、新しい行の一部 (全部ではない) の自動インクリメント値を指定する 「単純挿入」 ステートメントです。次の例を示します。c1 はテーブル t1 の AUTO_INCREMENT カラムです。

```
INSERT INTO t1 (c1,c2) VALUES (1,'a'), (NULL,'b'), (5,'c'), (NULL,'d');
```

INSERT ... ON DUPLICATE KEY UPDATE は別のタイプの 「混在モード挿入」 で、最悪の場合には実質 INSERT のあとに UPDATE を実行することに相当しますが、AUTO_INCREMENT カラムに割り当てられた値は、更新フェーズで使用される可能性も使用されない可能性もあります。

innodb_autoinc_lock_mode パラメータには、次の 3 つの設定を指定できます。

- innodb_autoinc_lock_mode = 0 (「従来」 ロックモード)

  このロックモードでは、innodb_autoinc_lock_mode が存在する前と同じ動作が提供されます。すべての「INSERT のような」ステートメントでは、特殊なテーブルレベル AUTO-INC ロックが取得され、ステートメントの終了まで保持されます。これにより、特定のステートメントによって割り当てられた自動インクリメント値が連続的になります。

  このロックモードの用途は次のとおりです。

  - 下位互換性。
  - パフォーマンスのテスト。
  - 「混在モード挿入」での問題の対処 (あとで説明するセマンティクスに相違がある可能性があるため)。

- innodb_autoinc_lock_mode = 1 (「連続」 ロックモード)

  これがデフォルトのロックモードです。このモードでは、「一括挿入」 は特殊な AUTO-INC テーブルレベルロックを使用し、そのロックをステートメントの終了まで保持します。これは、INSERT ... SELECT、REPLACE ... SELECT、LOAD DATA のすべてのステートメントに当てはまります。一度に実行できるステートメントは、AUTO-INC ロックを保持している 1 つのステートメントだけです。

  このロックモードでは、「単純挿入」 (のみ) が、自動インクリメント値の割り当てのときに軽量相互排他ロックが使用される新しいロックモデルを使用します。別のトランザクションがテーブルレベル AUTO-INC ロックを保持していないかぎり、AUTO-INC ロックは使用されません。別のトランザクションが AUTO-INC ロックを保持している場合、「単純挿入」は「一括ロック」と同様に、AUTO-INC ロックを待機します。

  このロックモードでは、行数が事前にわからない (したがってステートメントの処理中に自動インクリメント番号が割り当てられる) INSERT ステートメントが存在する場合には、任意の 「INSERT のような」 ステートメントによって割り当てられたすべての自動インクリメント値が必ず連続した値になるため、その処理は、ステートメントベースのレプリケーションで使用しても安全です。

  簡単に言えば、このロックモードの重要な効果は、拡張性の大幅な向上です。このモードは、ステートメントベースのレプリケーションで使用しても安全です。さらに、「従来」ロックモードの場合と同じく、任意のステートメントによって割り当てられた自動インクリメント番号が連続的になります。このモードでは 「従来」 モードと比較して、1 つの重要な例外を除けば、自動インクリメントを使用するステートメントのセマンティクスに変更点はありません。

  例外は「混在モード挿入」です。この挿入では、ユーザーは複数行の「単純挿入」で、明示的な値を全部ではなく、一部の行の AUTO_INCREMENT カラムに指定します。このような挿入の場合、InnoDB は挿入される行数よりも大きい自動インクリメント値を割り当てます。ただし、自動的に割り当てられる値はすべて連続的に生成されるため、直前に実行されたステートメントによって生成された自動インクリメント値よりも値が大きくなります。「余分」な番号は失われます。

- innodb_autoinc_lock_mode = 2 (「インターリーブ」 ロックモード)

  このロックモードでは、テーブルレベル AUTO-INC ロックを使用する 「INSERT のような」ステートメントは 1 つも存在しないため、複数のステートメントを同時に実行できます。これはもっとも高速で、もっとも拡張性の高いロックモードです。ただし、ステートメントベースのレプリケーションを使用する場合や、リカバリシナリオでバイナリログから SQL ステートメントを再現する際には、安全ではありません。

  このロックモードでは、自動インクリメント値は一意であり、並列実行されているすべての 「INSERT のような」ステートメントにわたって単調に増加することが保証されます。ただし、複数のステートメントが同時に番号を生成している (つまり番号の割り当てが複数のステートメント間でインターリーブされている) 可能性があるため、任意のステートメントによって挿入される行に対して生成された値が連続的でない可能性があります。

唯一のステートメントの実行が、挿入される行数が事前にわかっている「単純挿入」である場合は、「混在モード挿入」を除いて、単一のステートメントで生成される数にギャップがありません。ただし、「一括挿入」が実行されると、特定のステートメントで割り当てられた自動インクリメント値にギャップが発生する可能性があります。

innodb_autoinc_lock_mode によって提供される自動インクリメントロックモードには、次のように使用上の暗黙の前提がいくつかあります。

- レプリケーションでの自動インクリメントの使用

  ステートメントベースレプリケーションを使用している場合は、innodb_autoinc_lock_mode を 0 または 1 に設定し、マスターとそのスレーブで同じ値を使用してください。innodb_autoinc_lock_mode = 2 (「インターリーブ」)、またはマスターとスレーブが同じロックモードを使用しない構成を使用する場合は、マスターとスレーブで自動インクリメント値が同じになることは保証されません。

  行ベースレプリケーションは SQL ステートメントの実行順序に左右されない (混在形式は、ステートメントベースレプリケーションでは安全でないステートメントで行ベースレプリケーションを使用する) ため、行ベースまたは混在形式レプリケーションを使用している場合は、すべての自動インクリメントロックモードが安全です。

- 「失われた」自動インクリメント値とシーケンスギャップ

  すべてのロックモード (0、1、および 2) では、自動インクリメント値を生成したトランザクションがロールバックされると、これらの自動インクリメント値が「失われます」。「INSERT のような」ステートメントが完了したかどうか、およびそれを含むトランザクションがロールバックされたかどうかに関係なく、自動インクリメントカラムの値は一度生成されたら、ロールバックできません。このような失われた値は再使用されません。したがって、テーブルの AUTO_INCREMENT カラムに格納されている値にはギャップが存在する可能性があります。

- 「一括挿入」の自動インクリメント値のギャップ

  innodb_autoinc_lock_mode が 0 (「従来」) または 1 (「連続」) に設定されている場合、テーブルレベル AUTO-INC ロックがステートメントの終了まで保持され、同時に実行できるステートメントはこのような 1 つのステートメントだけであるため、任意のステートメントによって生成される自動インクリメント値は、ギャップのない連続的なものとなります。

  innodb_autoinc_lock_mode が 2 (「インターリーブ」) に設定されている場合、「一括挿入」によって生成された自動インクリメント値にギャップが存在する可能性がありますが、並列実行中の 「INSERT のような」ステートメントが存在する場合に限ります。

  一括挿入では、各ステートメントで必要となる自動インクリメント値の正確な数がわからず、過大評価される可能性があるため、ロックモードが 1 または 2 の場合は、連続したステートメント間でギャップが発生する可能性があります。

- 「混在モード挿入」によって割り当てられる自動インクリメント値

  「単純挿入」が (全部ではなく) 一部の結果行の自動インクリメント値を指定する「混在モード挿入」を検討します。このようなステートメントの動作は、ロックモード 0、1、および 2 で異なります。たとえば、c1 はテーブル t1 の AUTO_INCREMENT カラムで、自動生成されたシーケンス番号の最新値が 100 であるとします。次のような「混在モード挿入」ステートメントを検討します。

```
INSERT INTO t1 (c1,c2) VALUES (1,'a'), (NULL,'b'), (5,'c'), (NULL,'d');
```

  innodb_autoinc_lock_mode が 0 (「従来」) に設定されている場合、4 つの新しい行は次のようになります。

```
+-----+------+
| c1  | c2   |
+-----+------+
|   1 | a    |
| 101 | b    |
|   5 | c    |
| 102 | d    |
+-----+------+
```

  自動インクリメント値は、ステートメントの実行開始時に一度にすべての値が割り当てられるのではなく、一度に 1 つずつ割り当てられるため、次に使用可能な自動インクリメント値は 103 になります。この結果は、並列実行中の (任意の型の) 「INSERT のような」ステートメントが存在するかどうかに左右されません。

  innodb_autoinc_lock_mode が 1 (「連続」) に設定されている場合も、4 つの新しい行は次のようになります。

```
+-----+------+
| c1  | c2   |
+-----+------+
|   1 | a    |
| 101 | b    |
|   5 | c    |
| 102 | d    |
+-----+------+
```

ただし、この場合、ステートメントの処理時に自動インクリメント値が 4 つ割り当てられましたが、そのうちの 2 つだけが使用されたため、次に使用可能な自動インクリメント値は 103 ではなく、105 になります。この結果は、並列実行中の (任意の型の) 「INSERT のような」ステートメントが存在するかどうかに左右されません。

innodb_autoinc_lock_mode が 2 (「インターリーブ」) に設定されている場合、4 つの新しい行は次のようになります。

```
+-----+------+
| c1  | c2   |
+-----+------+
|   1 | a    |
|   x | b    |
|   5 | c    |
|   y | d    |
+-----+------+
```

x と y の値は一意であり、以前に生成されたどの行よりも大きくなります。ただし、x と y の具体的な値は、並列実行中のステートメントによって生成された自動インクリメント値の個数によって変わります。

最後に、生成された最新のシーケンス番号が値 4 だったときに、次のステートメントを発行した場合を検討します。

```
INSERT INTO t1 (c1,c2) VALUES (1,'a'), (NULL,'b'), (5,'c'), (NULL,'d');
```

どのように innodb_autoinc_lock_mode を設定しても、行 (NULL, 'b') に対して 5 が割り当てられ、行 (5, 'c') の挿入が失敗するため、このステートメントから重複キーエラー 23000 (Can't write; duplicate key in table) が生成されます。

14.6.6 InnoDB と FOREIGN KEY 制約

このセクションでは、InnoDB ストレージエンジンでの外部キー処理と、MySQL サーバーでの処理とを比較したときの相違点について説明します。

外部キーの定義

InnoDB テーブルの外部キー定義は、次のような条件の対象となります。

- InnoDB では、外部キーが任意のインデックスカラムまたはカラムのグループを参照することが許可されます。ただし、参照されるテーブルには、参照されるカラムが同じ順序で最初のカラムとして一覧表示されているインデックスが存在する必要があります。
- 現在、InnoDB ではユーザー定義のパーティションを持つテーブルの外部キーがサポートされていません。つまり、ユーザーがパーティション化した InnoDB テーブルには、外部キーで参照される外部キー参照またはカラムが含まれる可能性がありません。
- InnoDB では、外部キー制約が一意でないキーを参照することが許可されます。これは、標準 SQL の InnoDB 拡張です。

参照アクション

InnoDB テーブルの外部キーに関する参照アクションは、次のような条件の対象となります。

- SET DEFAULT は、MySQL サーバーで許可されていますが、InnoDB では無効として拒否されます。この句を使用した CREATE TABLE および ALTER TABLE ステートメントは、InnoDB テーブルで許可されていません。
- 同じ参照キー値を持つ複数の行が親テーブルにある場合、InnoDB は、同じキー値を持つほかの親の行が存在しないかのように、外部キーチェックで動作します。たとえば、RESTRICT 型の制約が定義されていて、複数の親の行を含む子の行が存在する場合は、これらの親の行のいずれかを削除することが InnoDB で許可されません。

- InnoDB では、外部キー制約に対応するインデックス内のレコードに基づいて、深さ優先アルゴリズムを使用したカスケード操作が実行されます。
- ON UPDATE CASCADE または ON UPDATE SET NULL は、カスケード中に以前に更新していた同じテーブルを更新するように再帰する場合、RESTRICT と同様に機能します。つまり、自己参照型 ON UPDATE CASCADE または ON UPDATE SET NULL 操作は使用できません。この目的は、カスケード更新で発生する無限ループを回避することです。反対に、自己参照型 ON DELETE SET NULL は、自己参照型 ON DELETE CASCADE と同様に動作できます。カスケード操作は、15 レベルよりも深くネストされる可能性がありません。
- 一般的な MySQL と同様に、多数の行を挿入、削除、または更新する SQL ステートメントでは、InnoDB によって UNIQUE および FOREIGN KEY 制約が 1 行ずつチェックされます。外部キーチェックの実行時に、InnoDB は、調査対象の子または親のレコード上に共有の行レベルロックを設定します。InnoDB では、即座に外部キー制約がチェックされ、そのチェックはトランザクションのコミットまで遅延されません。SQL 標準によると、デフォルトの動作は遅延チェックにするべきです。つまり、SQL ステートメント全体が処理されたあとにはじめて、制約がチェックされます。InnoDB で制約の遅延チェックが実装されるまで、外部キーを使用してそれ自体を参照するレコードを削除するなどの一部の操作が実行できません。

外部キーの使用法とエラー情報

外部キーおよびそれらの使用法に関する一般的な情報は、INFORMATION_SCHEMA.KEY_COLUMN_USAGE テーブルでクエリーを実行することで取得できます。InnoDB テーブルに固有の詳細な情報は、INNODB_SYS_FOREIGN および INNODB_SYS_FOREIGN_COLS テーブル、または INFORMATION_SCHEMA データベースで見つかります。セクション13.1.17.2「外部キー制約の使用」も参照してください。

SHOW ERRORS 以外でも、InnoDB テーブルが関与する外部キーエラー (通常、MySQL サーバーではエラー 150) の発生時に、SHOW ENGINE INNODB STATUS の出力をチェックすることで、最近の InnoDB 外部キーエラーの詳細な説明を取得できます。

14.6.7 InnoDB テーブル上の制限

> **警告**
>
> mysql データベース内の MySQL システムテーブルを MyISAM から InnoDB テーブルに変換しないでください。これはサポートされていない操作です。これを行うと、mysql_install_db プログラムを使用してバックアップから古いシステムテーブルをリストアするか、再生成するまで、MySQL は再起動されません。

> **警告**
>
> NFS ボリューム上でデータファイルやログファイルが使用されるように InnoDB を構成することは、適切ではありません。それ以外の場合は、ファイルがほかのプロセスによってロックされ、MySQL で使用できなくなる可能性があります。

最大と最小

- テーブルには、最大で 1017 個のカラムを含めることができます (MySQL 5.6.9 で、以前の 1000 個の制限から上昇されました)。
- テーブルには、最大で 64 個のセカンダリインデックスを含めることができます。
- デフォルトでは、単一カラムインデックスのインデックスキーを最大で 767 バイトにすることができます。インデックスキープリフィクスにも同じ長さ制限が適用されます。セクション13.1.13「CREATE INDEX 構文」を参照してください。たとえば、UTF-8 文字セットと文字ごとに最大 3 バイトを使用すると仮定すれば、TEXT または VARCHAR カラム上で 255 文字よりも長いカラムプリフィクスインデックスを使用すると、この制限に達する可能性があります。innodb_large_prefix 構成オプションを有効にすると、DYNAMIC および COMPRESSED 行フォーマットを使用する InnoDB テーブルで、この長さ制限が 3072 バイトに上昇します。

  許可されている最大値よりも長いインデックスプリフィクス長を使用しようとすると、エラーが生成されます。スレーブ上でも innodb_large_prefix オプションを設定できなく、この制限の影響を受ける可能性のある一意のインデックスをスレーブが持っている場合は、レプリケーション構成でこのようなエラーを回避するために、このオプションをマスター上で設定することを避けてください。
- InnoDB の内部的な最大キー長は 3500 バイトですが、MySQL 自体では 3072 バイトに制限されています。この制限は、複数カラムインデックス内の結合されたインデックスキーの長さに適用されます。

- MySQL インスタンスの作成時に innodb_page_size オプションを指定して、InnoDB のページサイズを 8K バイトまたは 4K バイトまで小さくすると、16K バイトのページサイズに対応する 3072 バイトの制限に基づいて、比例的にインデックスキーの最大長も短くなります。つまり、インデックスキーの最大長は、ページサイズが 8K バイトのときは 1536 バイト、ページサイズが 4K バイトのときは 768 バイトになります。

- 可変長カラム (VARBINARY、VARCHAR、BLOB、および TEXT) を除き、行の最大長はデータベースページの半分より少し短くなります。つまり、デフォルトページサイズの 16K バイトでは、行の最大長が約 8000 バイトになります。MySQL インスタンスの作成時に innodb_page_size オプションを指定してページサイズを小さくすると、行の最大長は、8K バイトのページでは 4000 バイト、4K バイトのページでは 2000 バイトになります。LONGBLOB および LONGTEXT カラムは 4G バイト未満である必要があり、BLOB および TEXT カラムを含む行全体の長さは 4G バイト未満である必要があります。

  行の長さが 1 ページの半分より短い場合は、行全体がそのページ内にローカルに格納されます。セクション 14.10.2「ファイル領域管理」で説明したように、半ページを超える行では、その行が半ページ以内に収まるように、可変長カラムが外部オフページストレージの対象として選択されます。

- InnoDB では内部的に 65,535 バイトを超える行サイズがサポートされていますが、MySQL 自体では、すべてのカラムを結合したサイズに 65,535 行のサイズ制限が課されています。

```
mysql> CREATE TABLE t (a VARCHAR(8000), b VARCHAR(10000),
    -> c VARCHAR(10000), d VARCHAR(10000), e VARCHAR(10000),
    -> f VARCHAR(10000), g VARCHAR(10000)) ENGINE=InnoDB;
ERROR 1118 (42000): Row size too large. The maximum row size for the
used table type, not counting BLOBs, is 65535. You have to change some
columns to TEXT or BLOBs
```

  セクションD.10.4「テーブルカラム数と行サイズの制限」を参照してください。

- 一部の古いオペレーティングシステムでは、ファイルは 2G バイトよりも小さくする必要があります。これは、InnoDB 自体の制限ではありません。ただし、大きいテーブルスペースが必要な場合は、1 つではなく複数の小さいデータファイルを使用して構成するか、より大きいデータファイルを作成する必要があります。

- InnoDB ログファイルを結合したサイズは、最大で 512G バイトまでにすることができます。

- テーブルスペースの最小サイズは、10M バイトをわずかに超える大きさです。テーブルスペースの最大サイズは、40 億データベースページ (64T バイト) です。これはテーブルの最大サイズでもあります。

- InnoDB のデフォルトのデータベースページサイズは 16K バイトですが、MySQL インスタンスの作成時に innodb_page_size オプションを指定すれば、ページサイズを 8K バイトまたは 4K バイトまで小さくすることができます。

  > 注記
  >
  > ページサイズを大きくすることは、サポートされている操作ではありません。InnoDB では、16K バイトを超えるページサイズでは通常な動作が保証されません。InnoDB をコンパイルまたは実行するときに、問題が発生する可能性があります。特に、Barracuda ファイル形式の ROW_FORMAT=COMPRESSED では、ページサイズが 16K バイトで、14 ビットのポインタを使用することが前提となっています。
  >
  > 特定の InnoDB ページサイズを使用している MySQL インスタンスは、別のページサイズを使用するインスタンスのデータファイルやログファイルを使用できません。この制限によって、16K バイト以外のページサイズがサポートされている MySQL 5.6 のデータを使用したリストアまたはダウングレード操作が影響を受ける可能性があります。

インデックスの型

- MySQL 5.6.4 以降では、InnoDB テーブルで FULLTEXT インデックスがサポートされています。詳細は、セクション14.2.13.3「FULLTEXT インデックス」を参照してください。

- InnoDB テーブルでは空間データ型がサポートされますが、そのインデックスはサポートされません。

InnoDB テーブル上の制約

- 各インデックスツリーにランダムダイブを行い、それに従ってインデックスカーディナリティーの見積もりを更新すると、(SHOW INDEX 出力の「Cardinality」カラムに表示されるように) ANALYZE TABLE でインデックスカーディナリティーが決定されます。これらは単なる見積もりであるため、ANALYZE TABLE を繰り返し実行すると、別の数値が生成される可能性があります。これによって ANALYZE TABLE の InnoDB テーブル上での速度は速くなりますが、すべての行が考慮されているわけではないため、100% 正確とは言えません。

セクション14.13.16.1「永続的オプティマイザ統計のパラメータの構成」で説明したように、innodb_stats_persistent 構成オプションをオンにすると、ANALYZE TABLE で収集された統計の正確性および安定性を向上させることができます。統計は定期的に再計算されないため、この設定を有効にした場合は、従来と同様に、インデックス付きカラムデータの主要な変更後 (サーバーの再起動後など) に、ANALYZE TABLE を実行することが重要です。

innodb_stats_persistent_sample_pages システム変数 (永続的な統計設定がオンになっている場合)、または innodb_stats_transient_sample_pages システム変数 (永続的な統計設定がオフになっている場合) を変更すると、ランダムダイブの数を変更できます。

MySQL では、結合の最適化時にのみインデックスカーディナリティーの見積もりが使用されます。一部の結合が適切に最適化されない場合は、ANALYZE TABLE を使用してみてください。ANALYZE TABLE では特定のテーブルに十分な値が生成されない場合は、特定のインデックスの使用を強制するクエリーとともに FORCE INDEX を使用するか、または MySQL でテーブルスキャンよりもインデックス検索が優先されるように max_seeks_for_key システム変数を設定してください。セクション5.1.4「サーバーシステム変数」、およびセクションB.5.6「オプティマイザ関連の問題」を参照してください。

- テーブル上でステートメントまたはトランザクションが実行されていて、ANALYZE TABLE のあとに同じテーブル上で 2 番目の ANALYZE TABLE 操作が実行されると、そのステートメントまたはトランザクションが完了するまで、2 番目の ANALYZE TABLE 操作はブロックされます。この動作が発生する原因は、ANALYZE TABLE の実行が完了すると、ANALYZE TABLE によって現在ロード中のテーブル定義に非推奨のマークが付けられるためです。新しいステートメントまたはトランザクション (2 番目の ANALYZE TABLE ステートメントを含む) は、新しいテーブル定義をテーブルキャッシュにロードする必要があります。この動作は、現在実行中のステートメントまたはトランザクションが完了し、古いテーブル定義がパージされるまで発生する可能性がありません。複数の並列テーブル定義をロードすることは、サポートされていません。

- SHOW TABLE STATUS では、テーブルで予約された物理サイズを除き、InnoDB テーブルに関する正確な統計が得られません。行カウントは、単に SQL 最適化で使用される概算見積もりです。

- 並列トランザクションでは同時にさまざまな数の行が「参照」される可能性があるため、InnoDB のテーブルには、行の内部的なカウントが保持されません。SELECT COUNT(*) FROM t ステートメントを処理するために、InnoDB ではテーブルのインデックスがスキャンされますが、インデックスが完全にバッファープール内にない場合は多少時間がかかります。テーブルが頻繁に変更されない場合は、MySQL クエリーキャッシュを使用することが適切な解決策となります。すばやくカウントするには、自分で作成したカウンタテーブルを使用し、実行される挿入および削除に応じてアプリケーションで更新できるようにする必要があります。概算の行カウントでは十分でない場合は、SHOW TABLE STATUS を使用できます。セクション8.5「InnoDB テーブルの最適化」を参照してください。

- Windows 上の InnoDB では常に、データベース名およびテーブル名が内部的に小文字で格納されます。バイナリ形式のデータベースを Unix から Windows に、または Windows から Unix に移動するには、すべてのデータベースおよびテーブルを小文字の名前を使用して作成します。

- 最大カラム値を取得するためにテーブルでのインデックス付きの SELECT MAX(ai_col) 検索と同等の操作を実行できるように、AUTO_INCREMENT カラム ai_col をインデックスの一部として定義する必要があります。一般に、これはカラムをどこかのテーブルインデックスの 1 番目のカラムにすることで実現されます。

- InnoDB は、テーブル上に事前に指定された AUTO_INCREMENT カラムの初期化中に、AUTO_INCREMENT カラムに関連付けられたインデックスの最後に排他ロックを設定します。自動インクリメントカウンタにアクセスするときに InnoDB では、トランザクション全体の最後までではなく、現在の SQL ステートメントの最後まで存続する特別な AUTO-INC テーブルロックモードが使用されます。AUTO-INC テーブルロックが保持されている間は、ほかのクライアントはそのテーブルに挿入できません。セクション14.6.5「InnoDB での AUTO_INCREMENT 処理」を参照してください。

- MySQL サーバーを再起動すると、AUTO_INCREMENT カラム用に生成されたが一度も格納されなかった古い値 (つまり、ロールバックされた古いトランザクション内で生成された値) が InnoDB で再使用される可能性があります。

- AUTO_INCREMENT 整数カラムの値を使い果たすと、後続の INSERT 操作で重複キーエラーが返されます。これは一般的な MySQL の動作であり、MyISAM の動作と似ています。

- DELETE FROM tbl_name はテーブルを再生成しませんが、その代わりにすべての行を 1 つずつ削除します。

- 現在、カスケードされた外部キーのアクションではトリガーがアクティブになっていません。

- 内部 InnoDB カラム (DB_ROW_ID、DB_TRX_ID、DB_ROLL_PTR、DB_MIX_ID など) の名前と一致するカラム名を持つテーブルを作成することはできません。サーバーはエラー 1005 をレポートし、エラーメッセージ内のエラー −1 を参照します。この制約は、大文字の名前を使用する場合にのみ適用されます。

ロックとトランザクション

- innodb_table_locks=1 (デフォルト) の場合、LOCK TABLES で各テーブル上に 2 つのロックが取得されます。MySQL レイヤーでのテーブルロックに加えて、InnoDB テーブルロックも取得されます。バージョン 4.1.2 よりも前の MySQL では、InnoDB テーブルロックが取得されませんでした。この古い動作は、innodb_table_locks=0 を設定すれば選択できます。InnoDB テーブルロックが取得されない場合は、テーブルの一部のレコードがほかのトランザクションによってロックされなくても、LOCK TABLES が完了します。

  MySQL 5.6 では、LOCK TABLES ... WRITE を使用して明示的にロックされたテーブルには、innodb_table_locks=0 が無効です。LOCK TABLES ... WRITE で暗黙的に (たとえば、トリガーを使用して)、または LOCK TABLES ... READ によって、読み取りまたは書き込み用にロックされたテーブルには有効です。

- トランザクションで保持されているすべての InnoDB ロックは、トランザクションがコミットまたは中止されると解放されます。したがって、autocommit=1 モードの InnoDB テーブル上で LOCK TABLES を呼び出しても、取得された InnoDB テーブルロックはすぐに解放されてしまうため、まったく意味がありません。

- LOCK TABLES では暗黙的な COMMIT および UNLOCK TABLES が実行されるため、トランザクションの実行中に追加のテーブルをロックできません。

- 並列データ変更トランザクションの 1023 個の制限は、MySQL 5.5 以上で上昇されました。現在、その制限は、Undo レコードが生成される 128 * 1023 個の並列トランザクションになりました。適切なトランザクション構造を変更する必要のある回避策 (より頻繁にコミットするなど) をすべて解除できます。

14.7 InnoDB 圧縮テーブル

圧縮するための SQL 構文および MySQL 構成オプションを使用すると、データが圧縮形式で格納されるテーブルを作成できます。圧縮を使用すると、生のパフォーマンスと拡張性の両方を改善する際に役立つことがあります。圧縮とは、ディスクとメモリー間で転送されるデータの量が少なくなり、ディスク上とメモリー内で占有される領域の量が少なくなることを意味します。インデックスデータも圧縮されるため、セカンダリインデックスを含むテーブルでは利点も増幅されます。SSD ストレージデバイスは、HDD デバイスよりも容量が小さくなる傾向があるため、圧縮が特に重要となる可能性があります。

14.7.1 テーブル圧縮の概要

プロセッサおよびキャッシュメモリーは、ディスクストレージデバイスよりも速度が上昇しているため、多くのワークロードがディスクバウンドになります。データ圧縮を使用すると、データベースのサイズが小さくなり、I/O が削減され、スループットが改善されますが、CPU 使用率が上昇するという少しの犠牲が伴います。圧縮は、頻繁に使用されるデータをメモリー内に保持するために十分な RAM が搭載されたシステム上で、読み取り負荷の高いアプリケーションを実行する際に、特に有効です。

ROW_FORMAT=COMPRESSED で作成された InnoDB テーブルでは、通常の 16K バイトのデフォルトよりも小さいページサイズをディスク上で使用できます。ページが小さいほど、ディスクから読み取られる I/O とディスクに書き込まれる I/O が少なくなるため、SSD デバイスを使用する際に、特に有効です。

ページサイズは、KEY_BLOCK_SIZE パラメータを使用して指定されます。ページサイズが異なる場合は、テーブルをシステムテーブルスペース内でなく、独自の .ibd ファイルに格納する必要があります。そのためには、innodb_file_per_table オプションを有効にする必要があります。圧縮レベルは、KEY_BLOCK_SIZE の値に関係なく同じです。KEY_BLOCK_SIZE に小さい値を指定するほど、徐々にページが小さくなるという I/O の利点が得られます。ただし、小さすぎる値を指定すると、各ページ内に複数の行を収容できるほど十分にデータ値を圧縮できない場合に、ページを再編成するための追加のオーバーヘッドが発生します。そのインデックスごとのキーカラムの長さに基づいて、どのくらい小さい KEY_BLOCK_SIZE をテーブルに指定できるのかについて、ハード制限が課されています。小さすぎる値を指定すると、CREATE TABLE または ALTER TABLE ステートメントが失敗します。

バッファープールには、圧縮済みデータが KEY_BLOCK_SIZE の値に基づいたページサイズの小さなページで保持されます。カラム値を抽出または更新すると、MySQL のバッファープールには、非圧縮データを含む 16K バイトのページも作成されます。バッファープール内では、非圧縮ページへの更新が同等の圧縮済みページに再度書き込まれます。圧縮済みページと非圧縮ページの両方の追加データが収容されるように、バッファーページのサイズを変更する必要がある場合もあります。ただし、非圧縮のページは、領域が必要になるとバッファープールから解放され、次のアクセス時に再度圧縮が解除されます。

14.7.2 テーブル圧縮の有効化

圧縮テーブルを作成する前に、innodb_file_per_table 構成オプションが有効になっていること、および innodb_file_format が Barracuda に設定されていることを確認してください。これらのパラメータは、MySQL 構

成ファイル my.cnf または my.ini で設定することも、MySQL サーバーをシャットダウンせずに SET ステートメントを使用して設定することもできます。

テーブルの圧縮を有効にするには、CREATE TABLE または ALTER TABLE ステートメントで ROW_FORMAT=COMPRESSED 句、KEY_BLOCK_SIZE 句、またはその両方を使用します。

圧縮テーブルを作成するには、次のようなステートメントを使用するとよいでしょう。

```
SET GLOBAL innodb_file_per_table=1;
SET GLOBAL innodb_file_format=Barracuda;
CREATE TABLE t1
 (c1 INT PRIMARY KEY)
 ROW_FORMAT=COMPRESSED
 KEY_BLOCK_SIZE=8;
```

- ROW_FORMAT=COMPRESSED を指定する場合は、KEY_BLOCK_SIZE を省略できます。innodb_page_size 値の半分であるデフォルトのページサイズ値が使用されます。
- KEY_BLOCK_SIZE を指定する場合は、ROW_FORMAT=COMPRESSED を省略できます。圧縮は自動的に有効になります。
- KEY_BLOCK_SIZE の最適な値を決定するには、一般に、この句にさまざまな値を指定した同じテーブルのコピーをいくつか作成してから、結果として生成される .ibd ファイルのサイズを計測し、現実的なワークロードで各動作のパフォーマンスを確認します。
- KEY_BLOCK_SIZE 値は、ヒントとして処理されます。InnoDB では、必要に応じて異なるサイズが使用される可能性があります。値 0 は、innodb_page_size 値の半分であるデフォルトの圧縮済みページサイズを表します。KEY_BLOCK_SIZE は、innodb_page_size 値以下にしかできません。innodb_page_size 値を超える値を指定した場合は、指定された値が無視され、警告が発行されます。また、KEY_BLOCK_SIZE は innodb_page_size 値の半分に設定されます。innodb_strict_mode=ON の場合、無効な KEY_BLOCK_SIZE 値を指定するとエラーが返されます。
- パフォーマンス関連の追加の構成オプションについては、セクション14.7.3「InnoDB テーブルの圧縮の調整」を参照してください。

InnoDB データページのデフォルトの非圧縮サイズは、16K バイトです。オプション値の組み合わせに応じて、MySQL ではテーブルの .ibd ファイルに対応した 1K バイト、2K バイト、4K バイト、8K バイト、または 16K バイトのページサイズが使用されます。実際の圧縮アルゴリズムは、KEY_BLOCK_SIZE 値の影響を受けません。この値によって、各圧縮済みチャンクの大きさが決定されるため、各圧縮済みページに詰め込むことができる行数が影響を受けます。

一般に、KEY_BLOCK_SIZE を InnoDB のページサイズに等しい値に設定しても、大量の圧縮は発生しません。たとえば、InnoDB のページサイズは 16K バイトであるため、一般に KEY_BLOCK_SIZE=16 を設定しても、大量の圧縮は発生しません。多くの場合、このような値で適切に圧縮されるため、この設定は多くの長い BLOB、VARCHAR、または TEXT カラムを持つテーブルで引き続き役立つことがあります。したがって、セクション14.7.5「InnoDB テーブルでの圧縮の動作」で説明したように、必要となるオーバーフローページが少なくなる可能性もあります。

テーブルのすべてのインデックス (クラスタ化されたインデックスを含む) は、CREATE TABLE または ALTER TABLE ステートメントで指定されたものと同じページサイズを使用して圧縮されます。ROW_FORMAT や KEY_BLOCK_SIZE などのテーブル属性は、InnoDB テーブルの CREATE INDEX 構文の一部ではないため、指定しても無視されます (ただし、SHOW CREATE TABLE ステートメントの出力には表示されます)。

圧縮テーブル上の制約

バージョン 5.1 よりも前の MySQL では圧縮テーブルを処理できないため、圧縮を使用するには、偶然に互換性の問題が発生することを回避するために、innodb_file_format=Barracuda 構成パラメータを指定する必要があります。

テーブルの圧縮は、InnoDB のシステムテーブルスペースでも使用できません。システムテーブルスペース (スペース 0、ibdata* ファイル) にはユーザーデータを含めることができますが、内部システム情報も含まれているため、圧縮されません。そのため、圧縮は独自のテーブルスペースに格納されているテーブル (およびインデックス) にのみ適用されます。つまり、innodb_file_per_table オプションが有効になっている状態で作成されます。

句の名前が ROW_FORMAT であるにもかかわらず、圧縮は個別の行にではなく、テーブル全体およびそれに関連付けられたすべてのインデックスに適用されます。

14.7.3 InnoDB テーブルの圧縮の調整

ほとんどの場合、InnoDB Data Storage and Compressionで説明した内部的な最適化によって、圧縮済みデータを使用してもシステムは適切に動作します。ただし、圧縮の効率性はデータの特性によって異なるため、圧縮テーブルのパフォーマンスに影響を与える決定を行うことができます。

- 圧縮するテーブル。
- 使用する圧縮済みページサイズ。
- 実行時のパフォーマンス特性 (システムでデータの圧縮および圧縮解除に要する時間など) に基づいて、バッファープールのサイズを調整するかどうか。ワークロードがデータウェアハウス (主にクエリー) または OLTP システム (クエリーと DML の混在) に似ているかどうか。
- システムの圧縮テーブル上で DML 操作が実行されているときに、データを配布する方法によって実行時に負荷の高い圧縮が失敗する場合は、追加の高度な構成オプションを調整することがあります。

このセクションのガイドラインを使用すると、このようなアーキテクチャー上および構成上の選択を行う際に役立ちます。長期間のテストを実施し、圧縮テーブルを本番環境に移行する準備ができたら、これらの選択を現実の状況で行なった場合の効率性を検証する方法について、セクション14.7.4「実行時の圧縮のモニタリング」を参照してください。

圧縮を使用するタイミング

一般に、圧縮は、適当な数の文字列カラムが含まれ、データの書き込みよりも読み取りの頻度の方がはるかに高いテーブルで最適に動作します。特定の状況で圧縮の利点が得られるかどうかを予測するための保証された方法はないため、必ず、代表的な構成で実行する特定のワークロードおよびデータセットをテストしてください。圧縮するテーブルを決定する際は、次の要素を検討してください。

データの特性と圧縮

データファイルのサイズを削減する際に圧縮の効率性の決定要因となるものは、データ自体の特性です。圧縮は、データのブロックで繰り返されるバイト文字列を識別することで動作していることを思い出してください。完全にランダム化されたデータは、最悪のケースです。多くの場合、一般的なデータには繰り返し値が含まれているため、効率的に圧縮されます。CHAR、VARCHAR、TEXT、または BLOB のいずれのカラムに定義されているのかに関係なく、多くの場合、文字列は効率的に圧縮されます。その一方で、一般に、ほとんどがバイナリデータ (整数または浮動小数) や以前に圧縮されたデータ (JPEG または PNG イメージなど) を含むテーブルは、大幅にまたはまったく効率的に圧縮されない可能性があります。

InnoDB テーブルごとに圧縮を有効にするかどうかを選択します。テーブルおよびそのすべてのインデックスでは、同じ (圧縮済み) ページサイズが使用されます。すべてのテーブルカラムのデータを含む主キー (クラスタ化) インデックスは、セカンダリインデックスよりも効率的に圧縮される可能性があります。長い行が存在する場合に圧縮を使用すると、セクション14.9.3「DYNAMIC および COMPRESSED 行フォーマット」で説明したように、長いカラム値が「オフページ」に格納される可能性があります。このようなオーバーフローページは、効率的に圧縮される可能性があります。これらの検討事項を考慮すると、多くのアプリケーションでは、一部のテーブルがその他よりも効率的に圧縮され、圧縮されたテーブルのサブセットを含むワークロードのみが最適に動作する場合もあります。

特定のテーブルを圧縮するかどうかを決定するには、実験を行います。非圧縮テーブルの .ibd ファイルのコピー上に、LZ77 圧縮 (gzip や WinZip など) が実装されたユーティリティーを使用すると、データを圧縮する際の効率性の概算見積もりを取得できます。MySQL ではページサイズ (デフォルトは 16K バイト) に基づいたチャンク単位でデータが圧縮されるため、MySQL で圧縮されたテーブルからは、ファイルベースの圧縮ツールよりも低い圧縮率が得られると予測できます。ページ形式には、ユーザーデータに加えて、圧縮されていない内部システムデータもいくつか含まれます。ファイルベースの圧縮ユーティリティーでは、さらに大きなデータチャンクを調査できるため、MySQL の各ページで見つかるよりも多くの繰り返し文字列が巨大なファイルで見つかる可能性があります。

特定のテーブル上で圧縮をテストするもう 1 つの方法は、いくつかのデータを非圧縮テーブルから同様の (同じインデックスをすべて含む) 圧縮テーブルにコピーし、結果として生成される .ibd ファイルのサイズを確認することです。例:

```
use test;
set global innodb_file_per_table=1;
set global innodb_file_format=Barracuda;
set global autocommit=0;

-- Create an uncompressed table with a million or two rows.
create table big_table as select * from information_schema.columns;
```

```
insert into big_table select * from big_table;
insert into big_table select * from big_table;
insert into big_table select * from big_table;
insert into big_table select * from big_table;
insert into big_table select * from big_table;
insert into big_table select * from big_table;
insert into big_table select * from big_table;
insert into big_table select * from big_table;
insert into big_table select * from big_table;
insert into big_table select * from big_table;
commit;
alter table big_table add id int unsigned not null primary key auto_increment;

show create table big_table\G

select count(id) from big_table;

-- Check how much space is needed for the uncompressed table.
\! ls -l data/test/big_table.ibd

create table key_block_size_4 like big_table;
alter table key_block_size_4 key_block_size=4 row_format=compressed;

insert into key_block_size_4 select * from big_table;
commit;

-- Check how much space is needed for a compressed table
-- with particular compression settings.
\! ls -l data/test/key_block_size_4.ibd
```

この実験では、次のような数値が生成されました。当然、テーブル構造やデータによって、数値が大幅に異なる可能性があります。

```
-rw-rw----  1 cirrus  staff  310378496 Jan  9 13:44 data/test/big_table.ibd
-rw-rw----  1 cirrus  staff  83886080 Jan  9 15:10 data/test/key_block_size_4.ibd
```

特定のワークロードで圧縮が効率的かどうかを確認するには:

- 単純なテストでは、その他の圧縮テーブルが含まれない MySQL インスタンスを使用して、INFORMATION_SCHEMA.INNODB_CMP テーブルに対してクエリーを実行します。
- 複数の圧縮テーブルが含まれるワークロードが関与するより詳細なテストでは、INFORMATION_SCHEMA.INNODB_CMP_PER_INDEX テーブルに対してクエリーを実行します。INNODB_CMP_PER_INDEX テーブルの統計を収集すると負荷が高くなるため、そのテーブルのクエリーを実行する前に、innodb_cmp_per_index_enabled 構成オプションを有効にする必要があります。このようなテストは、開発サーバーやクリティカルでないスレーブサーバーに限定されることもあります。
- テスト中の圧縮テーブルに対して、一般的な SQL ステートメントをいくつか実行します。
- INFORMATION_SCHEMA.INNODB_CMP または INFORMATION_SCHEMA.INNODB_CMP_PER_INDEX テーブルのクエリーを実行し、COMPRESS_OPS と COMPRESS_OPS_OK を比較することで、圧縮操作全体に対する正常な圧縮操作の比率を調査します。
- 圧縮操作が正常に完了した比率が高い場合は、そのテーブルが圧縮対象の候補である可能性が高くなります。
- 圧縮が失敗する比率が高い場合は、セクション14.7.6「OLTP ワークロードの圧縮」で説明したように、innodb_compression_level、innodb_compression_failure_threshold_pct、および innodb_compression_pad_pct_max オプションを調整すれば、さらに詳細なテストを試すことができます。

データベースの圧縮とアプリケーションの圧縮

アプリケーション内とテーブル内のどちらでデータを圧縮するかどうかを決定します。同じデータで両方のタイプの圧縮を使用しないでください。アプリケーション内でデータを圧縮し、その結果を圧縮テーブルに格納すると、追加の領域が節約される可能性は大幅に低くなり、二重圧縮によって単に CPU サイクルが無駄になるだけです。

データベース内での圧縮

これを有効にすると、MySQL テーブルの圧縮は自動的になり、すべてのカラムおよびインデックス値に適用されます。LIKE などの演算子を含むカラムも引き続きテストでき、インデックス値が圧縮されている場合でも、ソート操作でインデックスを引き続き使用できます。多くの場合、インデックスがデータベースの合計サイズの相当な割合を占めるため、圧縮を使用すると、ストレージ、I/O、またはプロセッサ時間が大幅に節約される可能性が

あります。圧縮および圧縮解除の操作は、予期される負荷を処理できるようにサイズ変更された強力なシステムとなる可能性が高いデータベースサーバー上で発生します。

アプリケーション内での圧縮

テキストなどのデータをアプリケーション内で圧縮してから、データベースに挿入する場合は、一部のカラムは圧縮されるが、その他は圧縮されないことで効率的に圧縮されないデータで、オーバーヘッドが節約される可能性があります。このアプローチでは、圧縮および圧縮解除用の CPU サイクルがデータベースサーバー上ではなく、クライアントマシン上で使用されるため、多数のクライアントが含まれる分散アプリケーションや、予備の CPU サイクルを備えたクライアントマシンに適している場合があります。

ハイブリッドアプローチ

当然、これらのアプローチは組み合わせることができます。一部のアプリケーションでは、いくつかの圧縮テーブルといくつかの非圧縮テーブルを使用することが適切である場合があります。一部のデータを外部で圧縮して (それを非圧縮テーブルに格納して)、アプリケーション内のその他のテーブル (の一部) を MySQL で圧縮できるようにすることが最適な方法である場合もあります。通常どおり、適切な決定に達するには、事前の設計および現実のテストが重要となります。

ワークロードの特性と圧縮

圧縮するテーブル (およびページサイズ) を選択することに加えて、ワークロードはもう 1 つのパフォーマンスの主要な決定要因でもあります。アプリケーションが更新ではなく、読み取りで占有されている場合は、圧縮済みデータ用に MySQL で保持されるページごとの「変更ログ」用の空き領域がインデックスページによって使い果たされたあとに、再編成および再圧縮する必要のあるページが少なくなります。更新によって、インデックスなしのカラムまたはそれらが含まれている BLOB や、偶然に「オフページ」に格納される大きな文字列が主に変更される場合は、圧縮のオーバーヘッドが許容可能になる可能性があります。単調に増加する主キーを使用する INSERT がテーブルへの唯一の変更であり、セカンダリインデックスがほとんどない場合は、インデックスページを再編成および再圧縮する必要もほとんどありません。MySQL では、非圧縮データを変更することで、「適切に」、圧縮済みページ上のデータに「削除マークを付け」てから削除できるため、テーブル上の DELETE 操作は比較的効率的に行われます。

環境によっては、データのロードに要する時間がリアルタイム検索と同じくらいに重要である場合があります。特にデータウェアハウス環境では、数多くのテーブルが読み取り専用または読み取りが大半になっている可能性があります。このような場合、結果として少数のディスク読み取りとストレージコストの節約が重要である場合を除いて、ロード時間が長くなるという点で圧縮の犠牲を払うことが許容できる場合と、許容できない場合があります。

本来は、データを圧縮および圧縮解除する際に CPU 時間を使用できるときに、圧縮が最適に動作します。そのため、ワークロードが CPU バウンドではなく、I/O バウンドである場合に、圧縮を使用することで全体的なパフォーマンスを改善できることがわかるでしょう。さまざまな圧縮構成でアプリケーションのパフォーマンスをテストする際は、計画した本番システム構成と同様のプラットフォーム上でテストしてください。

構成の特性と圧縮

データベースページのディスクからの読み取りとディスクへの書き込みは、システムパフォーマンスのもっとも低速な側面です。圧縮では、CPU 時間を使用してデータを圧縮および圧縮解除することで I/O の削減が試みられるため、プロセッササイクルと比べて、I/O が比較的少ないリソースであるときに、もっとも効率性が高くなります。

多くの場合、これは特に、高速のマルチコア CPU が搭載された複数ユーザー環境で動作しているときに当てはまります。圧縮テーブルのページがメモリー内にあるときは、MySQL では多くの場合、ページの非圧縮コピー用のバッファープール内で追加のメモリー (一般に 16K バイト) が使用されます。適応型 LRU アルゴリズムでは、I/O バウンドと CPU バウンドのどちらの方式でワークロードが動作しているのかに関係なく、考慮される圧縮済みページと非圧縮ページ間でメモリー使用のバランスを調整しようと試みられます。メモリーが非常に制約されている構成よりも、バッファープール専用のメモリーがより多く搭載された構成の方が、圧縮テーブルを使用するときに適切に動作する傾向があります。

圧縮済みページサイズの選択

圧縮済みページサイズの最適な設定は、テーブルおよびそのインデックスに含まれるデータの型および分布によって異なります。圧縮済みページのサイズは、常に最大のレコードサイズよりも大きくするようにしてください。そうでなければ、Compression of B-Tree Pagesで注記したように、操作に失敗する可能性があります。

圧縮済みページサイズの設定が大きすぎると、一部の領域が無駄になりますが、頻繁にページを圧縮する必要はなくなります。圧縮済みページサイズが小さすぎると、挿入時または更新時に時間のかかる再圧縮が必要になる

可能性があり、より頻繁な B ツリーノードの分割が必要になる可能性もあります。これにより、データファイルが大きくなり、インデックス作成の効率性が低くなります。

一般に、圧縮済みページサイズは 8K バイトまたは 4K バイトに設定されます。InnoDB テーブルの最大行サイズが約 8K とすれば、通常、KEY_BLOCK_SIZE=8 は安全な選択です。

14.7.4 実行時の圧縮のモニタリング

アプリケーション全体のパフォーマンス、CPU と I/O の使用率、およびディスクファイルのサイズは、アプリケーションでの圧縮の効率姓を示す適切な指標です。このセクションは、セクション14.7.3「InnoDB テーブルの圧縮の調整」に示したパフォーマンスチューニングのアドバイスに基づいて構成され、初期のテスト時には発生する可能性のない問題を見つける方法を示しています。

圧縮テーブルのパフォーマンス上の考慮事項をさらに深く掘り下げるには、例14.10「圧縮情報スキーマテーブルの使用」に記載した「情報スキーマ」テーブルを使用すれば、実行時に圧縮のパフォーマンスをモニターできます。これらのテーブルは、メモリーの内部使用および全体的に使用される圧縮の比率を反映しています。

INNODB_CMP テーブルには、使用中の圧縮済みページサイズ (KEY_BLOCK_SIZE) ごとに、圧縮アクティビティーに関する情報がレポートされます。これらのテーブル内の情報は、システム全体のものであり、データベース内のすべての圧縮テーブルにわたる圧縮の統計を集約したものです。このデータを使用すると、その他の圧縮テーブルがアクセスしていないときに、これらのテーブルを調査することでテーブルを圧縮するかどうかを決定する際に役立ちます。これには、サーバー上で比較的小さいオーバーヘッドが伴うため、圧縮失敗の全体的な効率性をチェックするために、本番環境サーバー上で定期的にクエリーを実行することがあります。

INNODB_CMP_PER_INDEX テーブルには、個別のテーブルおよびインデックスごとに、圧縮アクティビティーに関する情報がレポートされます。この情報は、圧縮の効率性を評価し、一度に 1 つのテーブルまたはインデックスのパフォーマンス問題を診断する際に、より的を絞ることができ、より役立ちます。(各 InnoDB テーブルはクラスタ化されたインデックスとして表されるため、このコンテキストでは、MySQL でテーブルとインデックス間で大きな区別が行われません。)INNODB_CMP_PER_INDEX テーブルには大量のオーバーヘッドが伴うため、さまざまなワークロード、データ、および圧縮設定の効果を分離して比較できる開発サーバーにより適しています。このモニタリングのオーバーヘッドが誤って課されることを防ぐには、INNODB_CMP_PER_INDEX テーブルのクエリーを実行する前に、innodb_cmp_per_index_enabled 構成オプションを有効にする必要があります。

考慮するべき主要な統計は、圧縮および圧縮解除操作の数、および実行に要する時間数です。B ツリーノードがいっぱいになって、変更後に圧縮済みデータを含めることができなくなると、MySQL によって B ツリーノードが分割されるため、「正常な」圧縮操作の数と、このような操作全体の数を比較します。INNODB_CMP および INNODB_CMP_PER_INDEX テーブル内の情報、およびアプリケーション全体のパフォーマンスとハードウェアリソースの使用率に基づいて、ハードウェア構成の変更を行なったり、バッファープールのサイズを調整したり、別のページサイズを選択したり、圧縮する別のテーブルセットを選択したりすることがあります。

圧縮および圧縮解除するために必要な CPU 時間の合計が大きい場合は、高速またはマルチコアの CPU に変更すると、同じデータ、アプリケーションのワークロード、および圧縮テーブルのセットを使用してパフォーマンスを改善する際に役立つことがあります。バッファープールのサイズを大きくすると、パフォーマンスの改善に役立つこともあります。これにより、より多くの非圧縮ページをメモリー内に滞在できるようになるため、圧縮形式でのみメモリー内に存在するページを圧縮解除する必要が少なくなります。

(アプリケーションでの INSERT、UPDATE、および DELETE 操作の数、およびデータベースのサイズと比較して) 圧縮操作全体の数が大きい場合は、効率的な圧縮としては、圧縮テーブルの一部が更新される頻度が高すぎることを示している可能性があります。その場合は、より大きなページサイズを選択するか、圧縮するテーブルをより慎重に選択してください。

「正常な」圧縮操作の数 (COMPRESS_OPS_OK) が圧縮操作の合計数 (COMPRESS_OPS) の高い比率を占めている場合は、システムが正常に実行されている可能性が高くなります。比率が低い場合は、MySQL によって理想よりも頻繁に、B ツリーノードの再編成、再圧縮、および分割が行われます。この場合、一部のテーブルの圧縮を回避するか、圧縮テーブルの一部で KEY_BLOCK_SIZE を大きくしてください。テーブルの圧縮をオフにすると、アプリケーション内での「圧縮失敗」の数が合計の 1% または 2% を上回る可能性があります。(このような失敗の比率は、データのロードなどの一時的な操作時には許容範囲内である場合もあります)。

14.7.5 InnoDB テーブルでの圧縮の動作

このセクションでは、InnoDB テーブルの圧縮に関する一部の内部実装について詳細に説明します。ここで示す情報は、パフォーマンスを調整する際に役立つことがありますが、圧縮の基本的な使用を理解する必要はありません。

圧縮アルゴリズム

一部のオペレーティングシステムでは、ファイルシステムのレベルで圧縮が実装されています。一般に、ファイルは、可変サイズのブロックに圧縮される固定サイズのブロックに分割されるため、簡単に断片化されます。ブロック内部で何かが変更されるたびに、ブロック全体が再圧縮されてからディスクに書き込まれます。これらのプロパティーを使用すると、この圧縮方法が更新の多いデータベースシステムでの使用には適さなくなります。

MySQL では、LZ77 圧縮アルゴリズムが実装されている有名な zlib ライブラリの支援を得て、圧縮が実装されています。この圧縮アルゴリズムは十分に発達し、強固であり、CPU の使用率とデータサイズの削減の両方の点で効率的です。このアルゴリズムは「損失なし」であるため、常に、元の非圧縮データを圧縮形式から再構築できます。LZ77 圧縮は、圧縮されるデータ内で繰り返される一連のデータを見つけることで動作します。データ内の値のパターンによって、圧縮の効率性が決定されますが、多くの場合、一般的なユーザーデータは 50% 以上圧縮されます。

アプリケーションで実行される圧縮や、その他の一部のデータベース管理システムの圧縮機能とは異なり、InnoDB の圧縮は、ユーザーデータとインデックスの両方に適用されます。多くの場合、インデックスがデータベースの合計サイズの 40-50% 以上を占める可能性があるため、この相違点は重要です。データセットの圧縮が正常に動作しているときは、InnoDB のデータファイル (.idb ファイル) のサイズが非圧縮サイズの 25% - 50%、場合によってはそれよりも小さくなります。ワークロードによっては、このようにデータベースを小さくすることにより、CPU 使用率を少し増加させるだけで I/O を削減してスループットを増加できます。innodb_compression_level 構成オプションを変更すると、圧縮のレベルと CPU のオーバーヘッド間のバランスを調整できます。

InnoDB データストレージと圧縮

InnoDB テーブル内のすべてのユーザーデータは、B ツリーインデックス (クラスタ化されたインデックス) を構成しているページに格納されます。その他の一部のデータベースシステムでは、このタイプのインデックスは「インデックス編成テーブル」と呼ばれます。インデックスノード内の各行には、(ユーザーが指定した、またはシステムで生成された) 主キーの値およびテーブルのその他のすべてのカラムが含まれています。

InnoDB テーブル内のセカンダリインデックスは、値のペア (インデックスキーと、クラスタ化されたインデックス内の行へのポインタ) を含む B ツリーでもあります。実際は、ポインタはテーブルの主キーの値であり、インデックスキーおよび主キー以外のカラムが必要な場合に、クラスタ化されたインデックスにアクセスする際に使用されます。常に、セカンダリインデックスのレコードは、B ツリーページ上に収容される必要があります。

次のセクションで説明するように、(クラスタ化インデックスとセカンダリインデックスの両方の) B ツリーノードの圧縮は、長い VARCHAR、BLOB、または TEXT カラムを格納するために使用されるオーバーフローの圧縮とは異なる方法で処理されます。

B ツリーページの圧縮

B ツリーページは頻繁に更新されるため、特別な処理が必要です。B ツリーノードが分割される回数を最小限にし、それらの内容を圧縮解除および再圧縮する必要性も最小限にすることが重要となります。

MySQL で使用される技術の 1 つでは、一部のシステム情報が非圧縮形式で B ツリーノード内に保持されるため、特定のインプレース更新が容易になります。たとえば、これにより、圧縮操作なしで行に削除のマークを付け、その行を削除できます。

さらに、MySQL では、インデックスページが変更されたときに、不要な圧縮解除および再圧縮を回避しようと試みられます。システムの各 B ツリーページ内には、ページに行われた変更を記録するための非圧縮の「変更ログ」が保持されます。小さいレコードの更新および挿入は、ページ全体を完全に再構築する必要なしで、この変更ログに書き込まれる場合があります。

変更ログ用の領域を使い果たすと、InnoDB によってページが圧縮解除され、変更が適用され、ページが再圧縮されます。再圧縮に失敗すると (圧縮の失敗と呼ばれる状況)、B ツリーノードが分割され、更新または挿入に成功するまでプロセスが繰り返されます。

OLTP アプリケーションなどで、書き込み負荷の高いワークロードでの頻繁な圧縮の失敗を回避するために、MySQL では、ページ内にいくつかの空のスペース (パディング) が予約されている場合があります。これにより、変更ログがより早く埋められ、分割を回避するための十分な空き領域がまだある間にページが再圧縮されます。各ページに残されるパディングスペースの量は、システムでページ分割の頻度が追跡されるにつれて変化します。圧縮テーブルへの書き込みが頻繁に行われる高負荷のサーバー上では、innodb_compression_failure_threshold_pct および innodb_compression_pad_pct_max 構成オプションを調整すると、このメカニズムを微調整できます。

一般に、MySQL では、InnoDB テーブル内の各 B ツリーページに 2 つ以上のレコードを収容できます。圧縮テーブルに対しては、この要件が緩和されました。B ツリーノードのリーフページには (主キーとセカンダリインデックスのどちらでも)、1 つのレコードのみが収容される必要がありますが、そのレコードはページごとの変更ログ

に非圧縮形式で収まる必要があります。innodb_strict_mode が ON の場合は、CREATE TABLE または CREATE INDEX の実行中に、MySQL によって行の最大サイズがチェックされます。行が収まらない場合は、「ERROR HY000: Too big row」というエラーメッセージが発行されます。

innodb_strict_mode が OFF のときにテーブルを作成した場合に、後続の INSERT または UPDATE ステートメントで圧縮済みページのサイズに収まらないインデックスエントリの作成が試みられると、その操作に失敗し、「ERROR 42000: Row size too large」というエラーが表示されます。(このエラーメッセージは、レコードが長すぎるインデックスの名前を示すものでも、その特定のインデックスページ上のインデックスレコードの長さや最大レコードサイズを示すものでもありません。)この問題を解決するには、ALTER TABLE を使用してテーブルを再構築し、より大きな圧縮済みページサイズ (KEY_BLOCK_SIZE) を選択して、任意のカラムプリフィクスのインデックスを短くするか、ROW_FORMAT=DYNAMIC または ROW_FORMAT=COMPACT を使用して圧縮を完全に無効にします。

BLOB、VARCHAR、および TEXT カラムの圧縮

InnoDB テーブルでは、主キーの一部ではない BLOB、VARCHAR、および TEXT カラムが、個別に割り当てられたオーバーフローページに格納される場合があります。このようなカラムは、オフページカラムと呼ばれています。これらの値は、オーバーフローページの片方向リストに格納されます。

ROW_FORMAT=DYNAMIC または ROW_FORMAT=COMPRESSED で作成されたテーブルでは、カラムの長さおよび行全体の長さによっては、BLOB、TEXT、または VARCHAR カラムの値が完全にオフページに格納される場合もあります。オフページに格納されるカラムでは、クラスタ化されたインデックスのレコードに、オーバーフローページへの 20 バイトのポインタのみがカラムごとに 1 つずつ含まれます。カラムがオフページに格納されるかどうかは、ページサイズおよび行の合計サイズによって異なります。行がクラスタ化されたインデックスのページ内に完全に収まらないほど長い場合は、クラスタ化されたインデックスページ上に行が収まるまで、MySQL によってオフページストレージに合った最長のカラムが選択されます。前述の注で示したように、行自体が圧縮済みページ上に収まらない場合は、エラーが発生します。

> **注記**
>
> ROW_FORMAT=DYNAMIC または ROW_FORMAT=COMPRESSED で作成されたテーブルでは、40 バイト以下の TEXT および BLOB カラムは、常にインラインに格納されます。

古いバージョンの MySQL で作成されたテーブルでは、ROW_FORMAT=REDUNDANT と ROW_FORMAT=COMPACT のみがサポートされている Antelope ファイル形式が使用されます。MySQL では、これらの形式で、クラスタ化されたインデックスレコード内に BLOB、VARCHAR、および TEXT カラムの最初の 768 バイトが主キーとともに格納されます。768 バイトのプリフィクスのあとには、残りのカラム値を含むオーバーフローページへの 20 バイトのポインタが続きます。

テーブルの形式が COMPRESSED である場合は、オーバーフローページに書き込まれるすべてのデータが「そのまま」圧縮されます。つまり、MySQL では、データ項目全体に zlib 圧縮アルゴリズムが適用されます。圧縮済みのオーバーフローページには、データ以外では特に、ページチェックサムを構成する非圧縮のヘッダーとトレーラ、および次のオーバーフローページへのリンクが含まれます。したがって、テキストデータを使用した場合に多く見られるように、データの圧縮性が高い場合は、長い BLOB、TEXT、または VARCHAR カラムで非常に大幅なストレージの節約が実現されます。一般に、JPEG などのイメージデータはすでに圧縮されているため、圧縮テーブルに格納される利点がほとんど得られません。領域の節約がほとんどない、またはまったくない場合は、二重圧縮によって CPU サイクルが無駄になる可能性があります。

オーバーフローページのサイズは、その他のページと同じです。カラムの合計長が 8K バイトのみである場合でも、オフページに格納される 10 個のカラムを含む行で、10 個のオーバーフローページが占有されます。非圧縮テーブルでは、10 個の非圧縮オーバーフローページで 160K バイトが占有されます。ページサイズが 8K の圧縮テーブルでは、80K バイトのみが占有されます。そのため、長いカラム値を含むテーブルでは、圧縮テーブル形式を使用すると効率性が高くなることが多くあります。

16K の圧縮済みページサイズを使用すると、多くの場合、このようなデータが効率的に圧縮されることで、必要なオーバーフローページが少なくなる可能性があるため、B ツリーノード自体のページ数は非圧縮形式の場合と同じであるにもかかわらず、BLOB、VARCHAR、または TEXT カラムのストレージおよび I/O コストを削減できます。

圧縮と InnoDB バッファープール

圧縮済みの InnoDB テーブル (1K、2K、4、または 8K) は、16K バイト (innodb_page_size が設定されている場合は、さらに小さいサイズ) の非圧縮ページに対応しています。ページ内のデータにアクセスするために、MySQL は、圧縮済みページがバッファープール内にすでに存在しない場合、そのページをディスクから読み取ってか

ら、その元の形式に圧縮解除します。このセクションでは、圧縮テーブルのページに関して、バッファープールが InnoDB で管理される方法について説明します。

I/O を最小限にして、ページを圧縮解除する必要性を削減するために、バッファープールに圧縮済み形式と非圧縮形式の両方のデータベースページが含まれることがあります。その他の必要なデータベースページ用の空き領域を作成するために、MySQL ではメモリー内に圧縮済みページを残しながら、バッファープールから非圧縮ページをエビクションできます。また、しばらくの間ページがアクセスされていない場合は、その他のデータ用に領域を解放するために、圧縮形式のページがディスクに書き込まれることもあります。したがって、そのときどきで、バッファープールに圧縮形式と非圧縮形式の両方のページが含まれている場合、圧縮形式のページのみが含まれている場合、どちらも含まれていない場合があります。

MySQL では、ホット (頻繁にアクセスされる) データがメモリー内に滞在する傾向となるように、最近もっとも使用されていない (LRU) リストを使用して、メモリー内に保持されるページおよび削除されるページが追跡されます。圧縮テーブルにアクセスすると、MySQL は適応型 LRU アルゴリズムを使用して、メモリー内の圧縮済みページと非圧縮ページの適切なバランスを実現します。この適応型アルゴリズムは、システムが I/O バウンドと CPU バウンドのどちらの方式で実行されているかどうかの影響を受けやすくなります。この目的は、CPU の負荷が高いときにページを圧縮解除するために要する処理時間が長くなりすぎることを回避すること、および (メモリー内にすでに存在する可能性のある) 圧縮済みページを圧縮解除するために使用できる予備のサイクルが CPU に備わっているときに過剰な I/O が発生することを回避することです。システムが I/O バウンドの場合、このアルゴリズムでは、その他のディスクページ用により多くの空き領域を作成することでメモリーが常駐になるように、ページの両方のコピーではなく、非圧縮コピーを削除することが優先されます。システムが CPU バウンドの場合、MySQL では、「ホット」ページ用に使用できるメモリーが多くなり、圧縮形式でのみメモリー内のデータを圧縮解除する必要性が少なくなるように、圧縮済みページと非圧縮ページの両方を削除することが優先されます。

圧縮と InnoDB の Redo ログファイル

圧縮済みページがデータファイルに書き込まれる前に、MySQL によってページのコピーが Redo ログに書き込まれます (最後にデータベースに書き込まれた以降に再圧縮された場合)。これは、zlib ライブラリがアップグレードされ、その変更によって圧縮済みデータとの互換性の問題が発生する可能性が低い場合でも、クラッシュリカバリ時に Redo ログを使用できるかどうかを確認するために行われます。したがって、圧縮の使用時に、ログファイルのサイズを多少大きくすること、またはより頻繁にチェックポイントを発生させる必要性を多少多くすることが要求される可能性があります。ログファイルのサイズを大きくする量またはチェックポイントの頻度を多くする数は、再構成および再圧縮が必要となる方法で圧縮済みページが変更される回数によって異なります。

圧縮テーブルでは、Redo ログおよびテーブルごとのテーブルスペースに使用されるファイル形式が MySQL 5.1 以前とは異なることに注意してください。MySQL Enterprise Backup 製品では、圧縮済み InnoDB テーブル用に、この最新の Barracuda ファイル形式がサポートされています。

14.7.6 OLTP ワークロードの圧縮

従来、InnoDB の圧縮機能は、データウェアハウス構成などで、主に読み取り専用または読み取りが大半のワークロードに対して使用することが推奨されていました。高速だが、比較的小型で高価である SSD ストレージデバイスを増加すると、OLTP ワークロードでも圧縮が魅力的なものとなります。高トラフィックでインタラクティブな Web サイトでは、頻繁に INSERT、UPDATE、および DELETE 操作を実行するアプリケーションで圧縮テーブルを使用することで、ストレージの要件および 1 秒あたりの I/O 操作 (IOPS) を削減できます。

MySQL 5.6 で導入された構成オプションを使用すると、書き込み負荷の高い操作のパフォーマンスおよび拡張性に重点を置いて、特定の MySQL インスタンスに合わせて圧縮の動作を調整できます。

- innodb_compression_level を使用すると、圧縮の程度を上げたり、下げたりできます。値を大きくすると、ストレージデバイス上に収容できるデータ量が多くなりますが、圧縮時の CPU オーバーヘッドも多くなるという犠牲が伴います。値を小さくすると、ストレージ領域がクリティカルでない場合に、CPU のオーバーヘッドを削減できます。それ以外の場合は、データが特に圧縮可能でないと予測されます。

- innodb_compression_failure_threshold_pct には、圧縮テーブルへの更新時に圧縮が失敗したときのカットオフポイントが指定されます。このしきい値を超えると、MySQL は、最大で innodb_compression_pad_pct_max で指定されたページサイズの割合まで空き領域の量を動的に調整することで、新しい各圧縮済みページ内に追加の空き領域を残し始めます。

- innodb_compression_pad_pct_max を使用すると、ページ全体を再度圧縮する必要なしで、変更を圧縮済み行に記録するための各ページ内に予約されている領域の最大量を調整できます。値を大きくすると、ページを再度圧縮せずに記録できる変更の量が多くなります。MySQL では、実行時に指定した割合の圧縮操作に「失敗した」ときにのみ、各圧縮テーブル内にあるページ用に可変量の空き領域が使用されますが、圧縮済みページを分割するために負荷の高い操作が必要となります。

圧縮済みデータを操作すると、圧縮済みと非圧縮の両方のバージョンのページが同時にメモリー内に保持されるため、OLTP スタイルのワークロードで圧縮を使用するときは、innodb_buffer_pool_size 構成オプションの値を大きくする準備をしてください。

14.7.7 SQL 圧縮構文の警告とエラー

Barracuda ファイル形式が有効になっていない場合に、CREATE TABLE または ALTER TABLE ステートメントで ROW_FORMAT=COMPRESSED または KEY_BLOCK_SIZE を指定すると、次のような警告が生成されます。SHOW WARNINGS ステートメントを使用すると、これらを表示できます。

レベル	コード	メッセージ
警告	1478	InnoDB: KEY_BLOCK_SIZE requires innodb_file_per_table.
警告	1478	InnoDB: KEY_BLOCK_SIZE requires innodb_file_format=1
警告	1478	InnoDB: ignoring KEY_BLOCK_SIZE=4.
警告	1478	InnoDB: ROW_FORMAT=COMPRESSED requires innodb_file_per_table.
警告	1478	InnoDB: assuming ROW_FORMAT=COMPACT.

注:

- デフォルトでは、これらのメッセージはエラーではなく、単なる警告です。テーブルは、オプションを指定しなかった場合と同様に、圧縮なしで作成されます。
- innodb_strict_mode を有効にすると、MySQL では、このような場合に警告ではなく、エラーが生成されます。現在の構成では圧縮テーブルの使用が許可されていないため、テーブルは作成されません。

「厳密でない」動作を使用すると、ソースデータベースに圧縮テーブルが含まれていない場合でも、圧縮テーブルがサポートされていないデータベースに mysqldump ファイルをインポートできます。この場合、MySQL は操作を回避する代わりに、ROW_FORMAT=COMPACT 内にテーブルを作成します。

新しいデータベースにダンプファイルをインポートし、元のデータベース内に存在するときにテーブルが再作成されるようにするには、サーバーで innodb_file_format および innodb_file_per_table 構成パラメータが適切に設定されていることを確認します。

KEY_BLOCK_SIZE 属性は、ROW_FORMAT が COMPRESSED として指定されているか、省略されている場合にのみ許可されます。その他の ROW_FORMAT とともに KEY_BLOCK_SIZE を指定すると、SHOW WARNINGS を使用して表示できる警告が生成されます。ただし、テーブルは非圧縮です。つまり、指定された KEY_BLOCK_SIZE は無視されます。

レベル	コード	メッセージ
警告	1478	InnoDB: ignoring KEY_BLOCK_SIZE=n unless ROW_FORMAT=COMPRESSED.

innodb_strict_mode が有効になっている状態で実行している場合は、COMPRESSED 以外の任意の ROW_FORMAT と KEY_BLOCK_SIZE を組み合わせると警告ではなく、エラーが生成され、テーブルは作成されません。

表14.2「CREATE TABLE および ALTER TABLE オプションの意味」では、CREATE TABLE および ALTER TABLE 上でさまざまなオプションが処理される方法が要約されています。

表 14.2 CREATE TABLE および ALTER TABLE オプションの意味

オプション	使用法	説明
ROW_FORMAT=REDUNDANT	MySQL 5.0.3 よりも前で使用されていたストレージフォーマット	ROW_FORMAT=COMPACT よりも効率性が低く、下位互換性を保つためのものです。
ROW_FORMAT=COMPACT	MySQL 5.0.3 以降でのデフォルトのストレージフォーマット	クラスタ化されたインデックスページに、768 バイトの長いカラム値のプリフィクスが格納され、残りのバイトはオーバーフローページに格納されます。
ROW_FORMAT=DYNAMIC	innodb_file_format=Barracuda とともにのみ使用可能	クラスタ化されたインデックスページ内に収まる場合は、そのページ内に値が保存されます。収まらない場合は、

オプション	使用法	説明
		オーバーフローページへの 20 バイトのポインタのみが (プリフィクスなしで) 格納されます。
ROW_FORMAT=COMPRESSED	innodb_file_format=Barracuda とともにのみ使用可能	zlib を使用してテーブルおよびインデックスをデフォルトの圧縮済みページサイズの 8K バイトに圧縮します。暗黙的に ROW_FORMAT=DYNAMIC を示します。
KEY_BLOCK_SIZE=n	innodb_file_format=Barracuda とともにのみ使用可能	1、2、4、8、または 16K バイトの圧縮済みページサイズを指定します。暗黙的に ROW_FORMAT=DYNAMIC および ROW_FORMAT=COMPRESSED を示します。

表14.3「InnoDB 厳密モードがオフになっているときの CREATE/ALTER TABLE の警告とエラー」では、CREATE TABLE または ALTER TABLE ステートメント上で、構成パラメータとオプションの特定の組み合わせで発生するエラー状況、およびオプションが SHOW TABLE STATUS の出力に表示される方法について簡単に説明しています。

innodb_strict_mode が OFF の場合、MySQL によってテーブルが作成または変更されますが、次に示すように特定の設定は無視されます。警告メッセージは、MySQL エラーログで確認できます。innodb_strict_mode が ON の場合、このような特定のオプションの組み合わせでエラーが生成され、テーブルは作成または変更されません。エラー状況の完全な説明を参照するには、次に示すように、SHOW ERRORS ステートメントを発行します。

```
mysql> CREATE TABLE x (id INT PRIMARY KEY, c INT)

-> ENGINE=INNODB KEY_BLOCK_SIZE=33333;

ERROR 1005 (HY000): Can't create table 'test.x' (errno: 1478)

mysql> SHOW ERRORS;
+-------+------+-----------------------------------------+
| Level | Code | Message                                 |
+-------+------+-----------------------------------------+
| Error | 1478 | InnoDB: invalid KEY_BLOCK_SIZE=33333.   |
| Error | 1005 | Can't create table 'test.x' (errno: 1478) |
+-------+------+-----------------------------------------+

2 rows in set (0.00 sec)
```

表 14.3 InnoDB 厳密モードがオフになっているときの CREATE/ALTER TABLE の警告とエラー

構文	警告またはエラーの状況	結果として SHOW TABLE STATUS に表示される ROW_FORMAT
ROW_FORMAT=REDUNDANT	なし	REDUNDANT
ROW_FORMAT=COMPACT	なし	COMPACT
ROW_FORMAT=COMPRESSED または ROW_FORMAT=DYNAMIC、または KEY_BLOCK_SIZE が指定されている	innodb_file_format=Barracuda と innodb_file_per_table の両方が有効になっていなければ無視されます。	COMPACT
無効な KEY_BLOCK_SIZE (1、2、4、8、または 16 以外) が指定されている	KEY_BLOCK_SIZE が無視されます。	リクエストされたもの、または COMPACT (デフォルト)
ROW_FORMAT=COMPRESSED および有効な KEY_BLOCK_SIZE が指定されている	なし。デフォルトの 8K ではなく、指定された KEY_BLOCK_SIZE が使用されます。	COMPRESSED
REDUNDANT、COMPACT、または DYNAMIC 行フォーマットを使用して KEY_BLOCK_SIZE が指定されている	KEY_BLOCK_SIZE が無視されます。	REDUNDANT、COMPACT、または DYNAMIC
ROW_FORMAT が REDUNDANT、COMPACT、DYNAMIC または COMPRESSED のいずれでもない	MySQL パーサーで認識される場合は無視されます。その他の場合は、エラーが発行されます。	COMPACT または該当なし

innodb_strict_mode が ON の場合、MySQL では無効な ROW_FORMAT または KEY_BLOCK_SIZE パラメータが拒否されます。以前のバージョンの MySQL との互換性を保つために、厳密モードはデフォルトで有効になっていません。その代わりに、MySQL では、無視された無効なパラメータに対応した警告 (エラーではない) が発行されます。

SHOW TABLE STATUS を使用しても、選択した KEY_BLOCK_SIZE を表示できないことに注意してください。SHOW CREATE TABLE ステートメントでは、(テーブルの作成時に無視された場合でも) KEY_BLOCK_SIZE が表示されます。テーブルの実際の圧縮済みページサイズは、MySQL では表示できません。

14.8 InnoDB のファイル形式管理

InnoDB が進化するにつれ、新機能をサポートするために、新しいディスク上のデータ構造が必要になる場合があります。圧縮テーブル (セクション14.7「InnoDB 圧縮テーブル」を参照してください) や、オフページに格納された長い可変長カラム (セクション14.9「InnoDB の行ストレージと行フォーマット」を参照してください) などの機能には、以前のバージョンの InnoDB とは互換性のないデータファイル形式が必要です。これらのどちらの機能にも、新しい Barracuda ファイル形式の使用が必要です。

> **注記**
>
> その他の新機能はすべて、元の Antelope ファイル形式と互換性があるため、Barracuda ファイル形式を必要としません。

このセクションでは、新しい InnoDB テーブルのためのファイル形式の有効化、MySQL リリース間での異なるファイル形式の互換性の確認、使用されているファイル形式の識別、ファイル形式のダウングレード、および将来使用される可能性のあるファイル形式名について説明します。

指定されたファイル形式　InnoDB は、アップグレードやダウングレードの状況、または異なるレベルの MySQL を実行している異機種混在システムで互換性を管理しやすくするために、指定されたファイル形式を使用します。現在は、Antelope および Barracuda ファイル形式がサポートされています。Barracuda は最新のファイル形式です。これは、圧縮テーブルや、より効率的な BLOB ストレージのための DYNAMIC 行フォーマットなどの重要な InnoDB 機能をサポートします。以前は名前が付いていなかった元の InnoDB ファイル形式は現在、Antelope と呼ばれます。InnoDB の将来のバージョンでは、動物の名前で識別される一連のファイル形式が昇順のアルファベット順に導入される可能性があります。

14.8.1 ファイル形式の有効化

innodb_file_format 構成パラメータは、新しい InnoDB テーブルに使用するファイル形式を定義します。このパラメータは、独自のテーブルスペースを持つテーブルにのみ適用できるため、innodb_file_per_table を有効にする必要があります。

innodb_file_format パラメータは現在、Antelope および Barracuda ファイル形式をサポートしています。テーブル圧縮や新しい DYNAMIC 行フォーマットなどの、Barracuda ファイル形式によってサポートされる機能を利用する新しいテーブルを作成するには、innodb_file_format を BARRACUDA に設定します。

MySQL 5.1 以前のリリースで組み込み InnoDB がデータベースにアクセスできなくなるような、Barracuda ファイル形式によってサポートされる新機能の使用を除外するには、innodb_file_format を省略するか、またはそれを Antelope に設定します。

innodb_file_format の値は、mysqld の起動時にコマンド行で設定するか、あるいはオプションファイル my.cnf (Unix オペレーティングシステム) または my.ini (Windows) で設定できます。また、SET GLOBAL ステートメントで動的に変更することもできます。

```
mysql> SET GLOBAL innodb_file_format=BARRACUDA;
Query OK, 0 rows affected (0.00 sec)
```

可能な場合、新しいテーブルには Barracuda 形式を使用することをお勧めしますが、MySQL 5.5 では、異なる MySQL リリースを含むレプリケーション構成との最大限の互換性のために、デフォルトのファイル形式は引き続き Antelope です。

14.8.2 ファイル形式の互換性の確認

InnoDB 1.1 には、新しいファイル形式を使用している InnoDB データファイルに対して古いリリースの MySQL サーバーを実行した場合に発生する可能性のあるクラッシュやデータ破損から保護するためのチェックがいくつか組み込まれています。これらのチェックは、サーバーが起動されたときと、テーブルに最初にアクセスしたと

きに実行されます。このセクションでは、これらのチェック、それらを制御する方法、および発生する可能性のあるエラーや警告の状態について説明します。

下位互換性

下位互換性の考慮事項は、最新バージョンの InnoDB (InnoDB Plugin、または InnoDB 1.1 を含む MySQL 5.5 以降) を古いバージョン (MySQL 5.1 以前、InnoDB Plugin ではなく組み込み InnoDB を含む) とともに使用している場合にのみ適用されます。互換性の問題が発生する可能性を最小限に抑えるために、MySQL 5.1 以前のすべてのデータベースサーバーを InnoDB Plugin に基づいて標準化できます。

一般に、新しいバージョンの InnoDB は、クラッシュ、ハングアップ、結果の誤り、破損などのリスクなしでは、以前のバージョンの InnoDB で安全に読み取りまたは書き込みができないテーブルまたはインデックスを作成する可能性があります。InnoDB 1.1 には、これらの状態から保護するとともに、データベースファイルや InnoDB のバージョン間での互換性の維持に役立つメカニズムが含まれています。このメカニズムを使用すると、下位互換性のないディスクファイルを作成する新機能の誤った使用を回避することによって、InnoDB リリースのいくつかの新機能 (パフォーマンス向上やバグ修正など) を利用しながら、引き続き以前のバージョンの InnoDB でデータベースを使用するオプションを保持できます。

あるバージョンの InnoDB によって特定のファイル形式がサポートされている場合は (その形式がデフォルトであるかどうかにかかわらず)、その形式または以前の形式を必要とするすべてのテーブルをクエリーして更新することができます。新機能を使用する新しいテーブルの作成だけは、有効になっている特定のファイル形式に基づいて制限されます。逆に、あるテーブルスペースに、現在実行中のソフトウェアでサポートされていないファイル形式を使用するテーブルまたはインデックスが含まれている場合は、読み取りアクセスであっても、そのテーブルスペースにはまったくアクセスできません。

InnoDB テーブルスペースを以前のファイル形式に「ダウングレード」するには、以前の形式を使用するテーブルスペース内の新しいテーブルにデータをコピーするしか方法がありません。これは、セクション14.8.4「ファイル形式のダウングレード」で説明されている ALTER TABLE ステートメントで実行できます。

既存の InnoDB テーブルスペースのファイル形式を判定するためのもっとも簡単な方法は、SHOW TABLE STATUS コマンドを使用するか、またはテーブル INFORMATION_SCHEMA.TABLES をクエリーして、そこに含まれているテーブルのプロパティーを検査することです。テーブルの Row_format が 'Compressed' または 'Dynamic' としてレポートされた場合、そのテーブルを含むテーブルスペースは Barracuda 形式を使用しています。それ以外の場合は、以前の InnoDB ファイル形式である Antelope を使用しています。

内部の詳細

InnoDB のテーブルごとのテーブルスペースファイル (*.ibd ファイルによって表されます) にはすべて、ファイル形式識別子のラベルが付けられます。システムテーブルスペース (ibdata ファイルによって表されます) には、InnoDB データベースファイルのグループで使用される「最高の」ファイル形式のタグが付けられ、そのファイルが開かれるときにこのタグがチェックされます。

圧縮テーブルを作成するか、または ROW_FORMAT=DYNAMIC でテーブルを作成すると、対応する .ibd ファイルのファイルヘッダーと InnoDB データディクショナリ内のテーブルタイプが Barracuda ファイル形式の識別子で更新されます。その時点から、このテーブルは、この新しいファイル形式をサポートしていない InnoDB のバージョンでは使用できなくなります。異常な動作から保護するために、InnoDB バージョン 5.0.21 以降では、テーブルが開かれるときに互換性チェックを実行します。

ib-file セットの定義

混乱を避けるために、この説明では「ib-file セット」という用語を、InnoDB が単位として管理する一連のオペレーティングシステムファイルを示すものとして定義します。ib-file セットには、次のファイルが含まれています。

- システムテーブルスペース (1 つ以上の ibdata ファイル)。これには、内部のシステム情報 (内部のカタログや Undo 情報を含む) が含まれるほか、ユーザーデータとインデックスも含まれる可能性があります。
- 0 個以上の単一テーブルのテーブルスペース (「file per table」 ファイルとも呼ばれ、*.ibd ファイルという名前が付けられます)。
- InnoDB ログファイル。通常は、ib_logfile0 と ib_logfile1 の 2 つです。クラッシュリカバリやバックアップに使用されます。

「ib-file セット」には、InnoDB テーブルに関するメタデータが含まれた、対応する .frm ファイルは含まれません。.frm ファイルは MySQL によって作成および管理され、InnoDB 内部のメタデータとの同期がとれなくなる場合があります。

1 つの「ib-file セット」に (場合によっては複数のデータベースの) 複数のテーブルを格納できます。(MySQL では、「データベース」は、ほかのシステムでは「スキーマ」または「カタログ」と呼ばれる、テーブルの論理的なコレクションです。)

14.8.2.1 InnoDB が起動されたときの互換性チェック

InnoDB は、ib-file セットを開いたときに発生する可能性のあるクラッシュまたはデータ破損を回避するために、その ib-file セット内で使用されているファイル形式を完全にサポートできることをチェックします。システムがクラッシュまたは「高速シャットダウン」(つまり、innodb_fast_shutdown が 0 より大きい) のあとに再起動された場合は、現在のソフトウェアにとっては「新しすぎる」形式のデータ構造 (Redo または Undo エントリや、二重書き込みページなど) がディスク上に存在する可能性があります。これらのデータ構造にアクセスすると、リカバリプロセス中に、データファイルの重大な破損が発生する場合があります。ファイル形式の起動チェックは、どのリカバリプロセスが開始されるよりも先に実行されるため、新しいテーブルとの一貫性の問題や MySQL サーバーの起動時の問題が回避されます。

バージョン InnoDB 1.0.1 から、システムテーブルスペースは、ib-file セットの一部であるいずれかのテーブルスペース内のいずれかのテーブルによって使用されている「最高の」ファイル形式の識別子またはタグを記録します。このファイル形式タグに対するチェックは、構成パラメータ innodb_file_format_check (デフォルトでは ON) によって制御されます。

システムテーブルスペース内のファイル形式タグが、現在実行中の特定のソフトウェアによってサポートされる最高のバージョンより新しいか、または高い値であり、かつ innodb_file_format_check が ON である場合は、サーバーが起動されたときに次のエラーが発行されます。

```
InnoDB: Error: the system tablespace is in a
file format that this version doesn't support
```

innodb_file_format をファイル形式名に設定することもできます。それにより、指定されたファイル形式が現在のソフトウェアでサポートされていない場合は InnoDB が起動されなくなります。また、「高位境界値」も指定した値に設定されます。innodb_file_format_check を設定する機能は、ib-file セット内のすべてのテーブルを手動で「ダウングレード」する場合 に (InnoDB の将来のリリースで) 役立ちます。それにより、あとで古いバージョンの InnoDB を使用して ib-file セットにアクセスする場合は、起動時のファイル形式のチェックを使用できます。

一部の限られた状況では、サーバーを起動し、「新しすぎる」形式 (使用しているソフトウェアではサポートされていない形式) の ib-file セットを使用することが必要になる場合があります。構成パラメータ innodb_file_format_check を OFF に設定すると、InnoDB はデータベースを開きますが、エラーログに次の警告メッセージを発行します。

```
InnoDB: Warning: the system tablespace is in a
file format that this version doesn't support
```

> **注記**
>
> これによりリカバリプロセスの実行が許可され、前回のシャットダウンがクラッシュまたは「高速シャットダウン」であった場合はデータベースが破損する可能性があるため、これは非常に危険な設定です。innodb_file_format_check を OFF に設定するのは、前回のシャットダウンが innodb_fast_shutdown=0 で実行されたことが確実である場合だけにし、基本的にリカバリプロセスが発生しないようにしてください。将来のリリースでは、このパラメータ設定は OFF から UNSAFE に名前が変更される可能性があります。(ただし、追加のファイル形式をサポートする InnoDB の新しいリリースが公開されるまでは、起動チェックを無効にしたとしても実際には「安全です」。)

パラメータ innodb_file_format_check は、データベースが開かれたときの動作にのみ影響を与え、そのあとには影響を与えません。逆に、パラメータ innodb_file_format (これは、特定の形式を有効にします) は、有効になっている形式で新しいテーブルを作成できるかどうかだけを決定し、データベースを開くことができるかどうかには影響を与えません。

ファイル形式タグは「高位境界値」であるため、「より高い」形式のテーブルが作成されるか、または既存のテーブルが読み取りまたは書き込みのためにアクセスされる (その形式がサポートされていると仮定します) と、サーバーが起動されたあとに増加します。実行中のソフトウェアがサポートしている形式より高い形式の既存のテーブルにアクセスした場合は、セクション14.8.2.2「テーブルが開かれたときの互換性チェック」で説明されているように、システムテーブルスペースのタグは更新されませんが、テーブルレベルの互換性チェックが適用されます (また、エラーが発行されます)。高位境界値が更新されると常に innodb_file_format_check の値も更新されるため、コマンド SELECT @@innodb_file_format_check; では、現在開いている ib-file セット内のテーブルによって使用され、現在実行中のソフトウェアによってサポートされることがわかっている最新のファイル形式の名前が表示されます。

この動作をもっとも適切に示すために、表14.4「InnoDB データファイルの互換性および関連する InnoDB パラメータ」で説明されているシナリオを考えてみます。ある将来のバージョンの InnoDB が Cheetah 形式をサポートし、ib-file セットがそのバージョンで使用されてきたとします。

表 14.4 InnoDB データファイルの互換性および関連する InnoDB パラメータ

innodb ファイル形式のチェック	innodb ファイル形式	ib-file セットで使用される最高のファイル形式	InnoDB によってサポートされる最高のファイル形式	結果
OFF	Antelope または Barracuda	Barracuda	Barracuda	データベースを開くことができます。Antelope または Barracuda ファイル形式を必要とするテーブルを作成できます
OFF	Antelope または Barracuda	Cheetah	Barracuda	データベースには「新しすぎる」形式のファイルが含まれているため、データベースを開くと警告が表示されます。Antelope または Barracuda ファイル形式のテーブルを作成できます。Cheetah 形式のテーブルにはアクセスできません
OFF	Cheetah	Barracuda	Barracuda	データベースを開くことができません。innodb_file_format を Cheetah に設定できません
ON	Antelope または Barracuda	Barracuda	Barracuda	データベースを開くことができます。Antelope または Barracuda ファイル形式のテーブルを作成できます
ON	Antelope または Barracuda	Cheetah	Barracuda	データベースには「新しすぎる」形式 (Cheetah) のファイルが含まれているため、データベースを開くことができません
ON	Cheetah	Barracuda	Barracuda	データベースを開くことができません。innodb_file_format を Cheetah に設定できません

14.8.2.2 テーブルが開かれたときの互換性チェック

テーブルがはじめてアクセスされると、InnoDB (InnoDB 1.0 の前のいくつかのリリースを含む) は、そのテーブルが格納されているテーブルスペースのファイル形式が完全にサポートされていることをチェックします。このチェックによって、チェックしないと、「新しすぎる」データ構造を使用しているテーブルが検出された場合に発生するクラッシュまたは破損が回避されます。

あるリリースによってサポートされるいずれかのファイル形式を使用しているすべてのテーブルの読み取りまたは書き込みが可能です (ユーザーに十分な権限があると仮定します)。システム構成パラメータ innodb_file_format を設定すると、特定のファイル形式が特定のリリースによってサポートされている場合でも、そのファイル形式を使用する新しいテーブルが作成されるのを防ぐことができます。このような設定は下位互換性を維持するために使用される可能性がありますが、サポートされているいずれかの形式を使用するテーブルへのアクセスは防げません。

指定されたファイル形式に示されているように、5.0.21 より古いバージョンの MySQL は、テーブルが作成されるときに新しいファイル形式が使用された場合、新しいバージョンによって作成されたデータベースファイルを確実に使用することができません。さまざまなエラー状態または破損を回避するために、InnoDB は、ファイルを開くときに (たとえば、テーブルへの最初のアクセス時に) ファイル形式の互換性をチェックします。現在実行中のバージョンの InnoDB が InnoDB データディクショナリ内のテーブルタイプで識別されるファイル形式をサポートしていない場合、MySQL は次のエラーをレポートします。

```
ERROR 1146 (42S02): Table 'test.t1' doesn't exist
```

InnoDB はまた、エラーログにもメッセージを書き込みます。

```
InnoDB: table test/t1: unknown table type 33
```

このテーブルタイプは、セクション14.8.3「使用されているファイル形式の識別」で説明されている、ファイル形式のバージョンを含むテーブルスペースフラグに等しいはずです。

MySQL 4.1 の前のバージョンの InnoDB では、データベースファイルにテーブル形式の識別子が含まれていませんでした。また、MySQL 5.0.21 の前のバージョンには、テーブル形式の互換性チェックが含まれていませんでした。そのため、「新しすぎる」形式のテーブルが 5.0.21 の前のバージョンの InnoDB で使用されている場合、正しい動作を保証するための方法はありません。

InnoDB 1.0 以降のファイル形式管理機能 (テーブルスペースのタグ付けおよび実行時チェック) を使用すると、InnoDB は、実行中のバージョンのソフトウェアがデータベース内の既存のテーブルを正しく処理できることをできるだけ早く確認できます。

InnoDB が、サポートしていない形式のファイルを含むデータベースを開くことを (パラメータ innodb_file_format_check を OFF に設定することによって) 許可した場合も、このセクションで説明したテーブルレベルのチェックが引き続き適用されます。

InnoDB Plugin を含む MySQL 5.1 より古いリリースの InnoDB では Barracuda ファイル形式のテーブルを含むデータベースファイルを使用しないようにすることを強くお勧めします。このようなテーブルは、セクション 14.8.4「ファイル形式のダウングレード」で説明されている手順を使用して Antelope 形式に「ダウングレード」できます。

14.8.3 使用されているファイル形式の識別

innodb_file_format 構成オプションを使用して別のファイル形式を有効にした場合、その変更は新しく作成されたテーブルにのみ適用されます。また、新しいテーブルを作成した場合、そのテーブルを含むテーブルスペースには、そのテーブルの機能をサポートするために必要な「もっとも早い」、または「もっとも単純な」ファイル形式のタグが付けられます。たとえば、Barracuda ファイル形式を有効にして、Dynamic または Compressed 行フォーマットを使用しない新しいテーブルを作成した場合、そのテーブルを含む新しいテーブルスペースには Antelope ファイル形式の使用のタグが付けられます。

特定のテーブルによって使用されるファイル形式を識別することは容易です。SHOW TABLE STATUS によってレポートされた行フォーマットが Compact または Redundant である場合、このテーブルは Antelope ファイル形式を使用します。SHOW TABLE STATUS によってレポートされた行フォーマットが Compressed または Dynamic である場合、このテーブルは Barracuda ファイル形式を使用します。

```
mysql> SHOW TABLE STATUS\G
*************************** 1. row ***************************
           Name: t1
         Engine: InnoDB
        Version: 10
     Row_format: Compact
           Rows: 0
 Avg_row_length: 0
    Data_length: 16384
Max_data_length: 0
   Index_length: 16384
      Data_free: 0
 Auto_increment: 1
    Create_time: 2014-11-03 13:32:10
    Update_time: NULL
     Check_time: NULL
      Collation: latin1_swedish_ci
       Checksum: NULL
 Create_options:
        Comment:
1 row in set (0.00 sec)
```

InnoDB INFORMATION_SCHEMA テーブルを使用して、特定のテーブルまたはテーブルスペースによって使用されるファイル形式を識別することもできます。例:

```
mysql> SELECT * FROM INFORMATION_SCHEMA.INNODB_SYS_TABLES WHERE NAME='test/t1'\G
*************************** 1. row ***************************
     TABLE_ID: 44
         NAME: test/t1
         FLAG: 1
       N_COLS: 6
        SPACE: 30
  FILE_FORMAT: Antelope
   ROW_FORMAT: Compact
ZIP_PAGE_SIZE: 0
1 row in set (0.00 sec)

mysql> SELECT * FROM INFORMATION_SCHEMA.INNODB_SYS_TABLESPACES WHERE NAME='test/t1'\G
*************************** 1. row ***************************
        SPACE: 30
         NAME: test/t1
         FLAG: 0
  FILE_FORMAT: Antelope
   ROW_FORMAT: Compact or Redundant
    PAGE_SIZE: 16384
ZIP_PAGE_SIZE: 0
1 row in set (0.00 sec)
```

14.8.4 ファイル形式のダウングレード

各 InnoDB テーブルスペースファイル (名前は *.ibd に一致します) には、そのテーブルとインデックスを作成するために使用されるファイル形式のタグが付けられます。テーブルスペースをダウングレードする方法として、テーブルとそのインデックスを再作成します。テーブルとそのインデックスを再作成するためのもっとも簡単な方法は、次のコマンドの使用です。

```
ALTER TABLE t ROW_FORMAT=COMPACT;
```

これを、ダウングレードする各テーブルに対して実行します。COMPACT 行フォーマットは、ファイル形式 Antelope を使用します。これは MySQL 5.0.3 で導入されました。

14.8.5 将来の InnoDB ファイル形式

MySQL 5.1 の標準の組み込み InnoDB によって使用されるファイル形式は、Antelope 形式です。InnoDB Plugin 1.0 で導入されたファイル形式は、Barracuda 形式です。追加の新しいファイル形式を必要とする新機能はまだ発表されていませんが、InnoDB ファイル形式のメカニズムでは将来の拡張が可能になっています。

完全性のために、将来のファイル形式に使用される可能性のあるファイル形式名を次に示します。Antelope、Barracuda、Cheetah、Dragon、Elk、Fox、Gazelle、Hornet、Impala、Jaguar、Kangaroo、Leopard、Moo および Zebra。これらのファイル形式は、内部の識別子 0 - 25 に対応しています。

14.9 InnoDB の行ストレージと行フォーマット

このセクションでは、テーブル圧縮や長いカラムのオフページストレージなどの特定の InnoDB 機能が、CREATE TABLE ステートメントの ROW_FORMAT 句によってどのように制御されるかについて説明します。ここでは、正しい行フォーマットを選択するための考慮事項、および MySQL リリース間の行フォーマットの互換性について説明します。

14.9.1 InnoDB 行ストレージの概要

行とそれに関連付けられたカラムのストレージが、クエリーや DML 操作のパフォーマンスに影響を与えます。1 つのディスクページに収まる行数が多いほど、クエリーやインデックスルックアップの処理は速くなり、InnoDB バッファープールで必要なキャッシュメモリーは減り、数値カラムや短い文字列カラムの更新された値を書き出すために必要な I/O も少なくなります。

各 InnoDB テーブル内のデータは、ページに分割されます。各テーブルを構成するページは、B ツリーインデックスと呼ばれるツリーデータ構造で配置されます。テーブルデータとセカンダリインデックスはどちらも、このタイプの構造を使用します。テーブル全体を表す B ツリーインデックスは、クラスタ化されたインデックスと呼ばれます。これは、主キーカラムに従って編成されます。インデックスデータ構造のノードには、その行内のすべてのカラム (クラスタ化されたインデックスの場合)、またはインデックスカラムと主キーカラム (セカンダリインデックスの場合) の値が含まれています。

可変長カラムは、このルールの例外です。B ツリーページに収めるには長すぎる BLOB や VARCHAR などのカラムは、オーバーフローページと呼ばれる、個別に割り当てられたディスクページに格納されます。このようなカラムを、オフページカラムと呼びます。これらのカラムの値は、オーバーフローページの片方向リンクリストに格納され、このような各カラムには 1 つ以上のオーバーフローページの独自のリストがあります。場合によっては、ストレージの浪費を避け、また個別のページを読み取る必要をなくすために、長いカラム値のすべてまたはプリフィクスが B ツリーに格納されます。

次のセクションでは、これらの可変長カラムを表す方法を制御するために CREATE TABLE および ALTER TABLE ステートメントで使用できる句である ROW_FORMAT と KEY_BLOCK_SIZE について説明します。これらの句を使用するには、innodb_file_per_table および innodb_file_format 構成オプションの設定の変更も必要になる可能性があります。

14.9.2 テーブルの行フォーマットの指定

テーブルの行フォーマットは、CREATE TABLE および ALTER TABLE ステートメントの ROW_FORMAT 句で指定します。例:

```
CREATE TABLE t1 (f1 int unsigned) ROW_FORMAT=DYNAMIC ENGINE=INNODB;
```

InnoDB ROW_FORMAT オプションには、COMPACT、REDUNDANT、DYNAMIC、および COMPRESSED が含まれます。InnoDB テーブルの場合、デフォルトでは、行は COMPACT フォーマット

(ROW_FORMAT=COMPACT) で格納されます。ROW_FORMAT テーブルオプションの詳細は、CREATE TABLE のドキュメントを参照してください。

InnoDB テーブルの物理的な行構造は、指定した ROW_FORMAT によって異なります。詳細は、セクション 14.2.13.7「物理的な行構造」を参照してください。

14.9.3 DYNAMIC および COMPRESSED 行フォーマット

このセクションでは、InnoDB テーブルの DYNAMIC および COMPRESSED 行フォーマットについて説明します。これらの種類のテーブルは、innodb_file_format 構成オプションが Barracuda に設定されている場合にのみ作成できます。(Barracuda ファイル形式では、COMPACT および REDUNDANT 行フォーマットも許可されます。)

テーブルが ROW_FORMAT=DYNAMIC または ROW_FORMAT=COMPRESSED で作成された場合、長いカラム値は完全にオフページに格納され、クラスタ化されたインデックスレコードにはオーバーフローページへの 20 バイトのポインタのみが含まれます。

カラムがオフページに格納されるかどうかは、ページサイズおよび行の合計サイズによって異なります。その行が長すぎる場合は、クラスタ化されたインデックスレコードが B ツリーページに収まるまで、InnoDB はオフページストレージのもっとも長いカラムを選択します。40 バイト以下の TEXT および BLOB カラムは、常にインラインに格納されます。

DYNAMIC 行フォーマットは、(COMPACT および REDUNDANT フォーマットと同様に) 収まる場合は行全体をインデックスノード内に格納する効率性を維持しますが、この新しいフォーマットでは、長いカラムの多数のデータバイトで B ツリーノードがいっぱいになる問題が回避されます。DYNAMIC フォーマットは、長いデータ値の一部がオフページに格納される場合は通常、すべての値をオフページに格納するのがもっとも効率的であるという考え方に基づいています。DYNAMIC フォーマットでは、短いカラムは B ツリーノード内に残る可能性があるため、特定の行に必要なオーバーフローページの数が最小限に抑えられます。

COMPRESSED 行フォーマットは、オフページストレージに関して DYNAMIC 行フォーマットと同様の内部の詳細を使用するほか、追加のストレージ、圧縮されるテーブルおよびインデックスデータからのパフォーマンスの考慮事項、および小さいページサイズを使用します。COMPRESSED 行フォーマットでは、オプション KEY_BLOCK_SIZE によって、クラスタ化されたインデックスに格納されるカラムデータの量、およびオーバーフローページに配置される量が制御されます。COMPRESSED 行フォーマットの詳細は、セクション 14.7「InnoDB 圧縮テーブル」を参照してください。

14.9.4 COMPACT および REDUNDANT 行フォーマット

InnoDB の早期のバージョンでは、データベースファイルに名前のないファイル形式 (現在は Antelope と呼ばれます) を使用していました。そのファイル形式では、テーブルは ROW_FORMAT=COMPACT または ROW_FORMAT=REDUNDANT で定義されます。InnoDB は、可変長カラム (BLOB や VARCHAR など) の最初の 768 バイトまでを B ツリーノード内のインデックスレコードに格納し、残りの部分はオーバーフローページに格納されます。

これらの以前のバージョンとの互換性を維持するために、最新の InnoDB で作成されたテーブルは、デフォルトで COMPACT 行フォーマットになります。新しい DYNAMIC および COMPRESSED 行フォーマットについては、セクション14.9.3「DYNAMIC および COMPRESSED 行フォーマット」を参照してください。

Antelope ファイル形式では、カラムの値が 768 バイト以下の場合、オーバーフローページは必要なく、値が B ツリーノード内に格納されるために I/O がある程度削減される可能性があります。これは、比較的短い BLOB の場合に適切に機能しますが、B ツリーノードがキー値ではなくデータでいっぱいになり、それによって効率が低下する可能性があります。多数の BLOB カラムを含むテーブルでは、B ツリーノードがデータでいっぱいになり、含まれる行数が少なすぎるために、行が短い場合や、カラム値がオフページに格納される場合に比べてインデックス全体の効率が低下することがあります。

14.10 InnoDB のディスク I/O とファイル領域管理

DBA は、I/O サブシステムが飽和状態にならないようにディスク I/O を管理するとともに、ストレージデバイスがいっぱいにならないようにディスク領域を管理する必要があります。ACID 設計モデルには、冗長に見える可能性はあっても、データの信頼性の確保に役立つ、ある一定の量の I/O が必要です。これらの制約の中で、InnoDB は、ディスク I/O の量を最小限に抑えるためにデータベースの動作やディスクファイルの編成を最適化しようとします。場合によって、I/O は、データベースがビジー状態でなくなるまで、または高速シャットダウンのあとのデータベースの再起動中など、すべてを一貫性のある状態に移行することが必要になるまで延期されます。

このセクションでは、デフォルトの種類の MySQL テーブル (InnoDB テーブルとも呼ばれます) での I/O とディスク領域に関する主な考慮事項について説明します。

- クエリーパフォーマンスを向上させるために使用されるバックグラウンド I/O の量の制御。
- 追加の I/O を削減する代わりに耐久性を向上させる機能の有効化または無効化。
- テーブルの、多数の小さなファイル、いくつかのより大きなファイル、またはその両方の組み合わせへの編成。
- Redo ログファイルのサイズと、ログファイルがいっぱいになったときに発生する I/O アクティビティーとのバランス。
- テーブルを最適なクエリーパフォーマンスのために再編成する方法。

14.10.1 InnoDB ディスク I/O

InnoDB は、可能であれば、I/O 操作を処理するための複数のスレッドを作成することによって非同期ディスク I/O を使用します。それにより、その I/O がまだ進行中の間もほかのデータベース操作を続行できるようにします。Linux および Windows プラットフォームでは、InnoDB は、使用可能な OS とライブラリ関数を使用して「ネイティブな」非同期 I/O を実行します。その他のプラットフォームの場合も、InnoDB は引き続き I/O スレッドを使用しますが、これらのスレッドが実際には I/O 要求の完了を待つ可能性があります。この手法は「シミュレートされた」非同期 I/O と呼ばれます。

先読み

InnoDB は、データがすぐに必要になる可能性が高いと判断できる場合は先読み操作を実行して、そのデータをメモリー内で使用できるようにバッファープールに移動します。連続したデータに対しては、いくつかの大きな読み取り要求を作成する方が、複数の拡散した小さな要求を作成するより効率的である場合があります。InnoDB には、次の 2 つの先読みに関する経験則があります。

- シーケンシャル先読みでは、テーブルスペース内のセグメントへのアクセスパターンがシーケンシャルであることに気付くと、InnoDB はデータベースページの読み取りのバッチを I/O システムにあらかじめ送信します。
- ランダム先読みでは、テーブルスペース内の一部の領域がバッファープールに完全に読み取られている最中であることに気付くと、InnoDB は残りの読み取りを I/O システムに送信します。

二重書き込みバッファー

InnoDB は、二重書き込みバッファーと呼ばれる構造に関連した斬新なファイルフラッシュ手法を使用しています。これはデフォルトで有効になっています (innodb_doublewrite=ON)。これにより、クラッシュや停電のあとのリカバリの安全性が高まるだけでなく、fsync() 操作の必要性が減るため、ほとんどの種類の Unix でパフォーマンスが向上します。

データファイルにページを書き込む前に、InnoDB はまず、それらのページを二重書き込みバッファーと呼ばれる連続したテーブルスペース領域に書き込みます。二重書き込みバッファーへの書き込みとフラッシュが完了したあとにはじめて、InnoDB はそれらのページをデータファイル内の適切な位置に書き込みます。ページ書き込みの最中にオペレーティングシステム、ストレージサブシステム、または mysqld プロセスのクラッシュ (それによる破損ページの状態) が発生した場合、InnoDB は、あとでリカバリ中にそのページの正常なコピーを二重書き込みバッファーから見つけることができます。

14.10.2 ファイル領域管理

構成ファイルで定義するデータファイルによって、InnoDB システムテーブルスペースが形成されます。これらのファイルは、テーブルスペースを形成するために論理的に連結されます。ストライピングは使用されません。現在、テーブルスペース内のどこにテーブルが割り当てられるかを定義することはできません。新しく作成されたテーブルスペースでは、InnoDB は最初のデータファイルから領域を割り当てます。

システムテーブルスペースの内部にすべてのテーブルおよびインデックスを格納することによって発生する問題を回避するために、innodb_file_per_table 構成オプションをオンにすることができます。このオプションは、新しく作成された各テーブルを個別のテーブルスペースファイル内に (拡張子 .ibd で) 格納します。この方法で格納されたテーブルの場合、ディスクファイル内の断片化は減少し、テーブルが切り捨てられると、その領域は InnoDB によって引き続きシステムテーブルスペース内に予約されるのではなく、オペレーティングシステムに返されます。

ページ、エクステント、セグメント、およびテーブルスペース

各テーブルスペースは、データベースページで構成されます。MySQL インスタンス内のテーブルスペースはすべて、同じページサイズを持っています。デフォルトでは、すべてのテーブルスペースが 16K バイトのページサイ

ズを持っています。このページサイズを 8K バイトまたは 4K バイトに減らすには、MySQL インスタンスを作成するときに innodb_page_size オプションを指定します。

これらのページは、サイズ 1M バイトのエクステント (連続した 64 個の 16K バイトページ、128 個の 8K バイトページ、または 256 個の 4K バイトページ) にグループ化されます。InnoDB では、テーブルスペース内部の「ファイル」をセグメントと呼びます。(これらのセグメントは、実際に多数のテーブルスペースセグメントが含まれているロールバックセグメントとは異なります。)

セグメントがテーブルスペース内部で拡張される場合、InnoDB は、そのセグメントに最初の 32 ページを一度に割り当てます。そのあと、InnoDB は、そのセグメントへのすべてのエクステントの割り当てを開始します。InnoDB は、データの良好な連続性を保証するために、大きなセグメントには 1 回につき最大 4 つのエクステントを追加できます。

InnoDB では、各インデックスに 2 つのセグメントが割り当てられます。1 つは B ツリーの非リーフノード用、もう 1 つはリーフノード用です。リーフノードをディスク上で連続した状態に維持すると、これらのリーフノードには実際のテーブルデータが含まれているため、シーケンシャル I/O 操作の性能が向上します。

テーブルスペース内の一部のページにはほかのページのビットマップが含まれているため、InnoDB テーブルスペース内のいくつかのエクステントは全体としてではなく、個々のページとしてのみセグメントに割り当てることができます。

SHOW TABLE STATUS ステートメントを発行することによってテーブルスペース内の使用可能な空き領域を求めると、InnoDB は、テーブルスペース内の確実に空いているエクステントをレポートします。InnoDB は、常にいくつかのエクステントをクリーンアップやその他の内部の目的のために予約します。これらの予約されたエクステントは空き領域に含まれません。

テーブルからデータを削除すると、InnoDB は、対応する B ツリーインデックスを短くします。解放された領域をほかのユーザーが使用できるようになるかどうかは、削除のパターンがテーブルスペースに対して個々のページまたはエクステントのどちらを解放するかによって異なります。テーブルを削除したりテーブルのすべての行を削除したりすると、その領域は確実にほかのユーザーに解放されますが、それらの削除された行は、それの行がトランザクションロールバックまたは一貫性読み取りに必要なくなったあと、しばらくして自動的に発生するパージ操作によってのみ物理的に削除されることに注意してください。(セクション14.2.12「InnoDB マルチバージョン」を参照してください。)

テーブルスペースに関する情報を表示するには、テーブルスペースモニターを使用します。セクション14.15「InnoDB モニター」を参照してください。

ページのテーブル行への関連付け

可変長カラム (VARBINARY、VARCHAR、BLOB、および TEXT) を除き、行の最大長はデータベースページの半分より少し短くなります。つまり、行の最大長は約 8000 バイトです。LONGBLOB および LONGTEXT カラムは 4G バイト未満である必要があり、BLOB および TEXT カラムを含む行全体の長さは 4G バイト未満である必要があります。

行の長さが 1 ページの半分より短い場合は、行全体がそのページ内にローカルに格納されます。それが 1 ページの半分を超える場合は、行が 1 ページの半分内に収まるまで、外部のオフページストレージとして可変長カラムが選択されます。オフページストレージとして選択されたカラムの場合、InnoDB は最初の 768 バイトをその行にローカルに格納し、残りを外部のオーバーフローページに格納します。このような各カラムには、オーバーフローページの独自のリストがあります。768 バイトのプリフィクスには、そのカラムの実際の長さを格納し、値の残りの部分が格納されているオーバーフローページリストを指す 20 バイトの値が付随します。

14.10.3 InnoDB チェックポイント

ログファイルを非常に大きくすると、チェックポイント設定中のディスク I/O が少なくなる可能性があります。ログファイルの合計サイズは多くの場合、バッファープールと同じか、またはそれより大きい設定が適切です。以前は、ログファイルが大きいとクラッシュリカバリに非常に長い時間がかかることがありましたが、MySQL 5.5 以降では、クラッシュリカバリのパフォーマンス向上により、クラッシュ後の起動を高速にして大きなログファイルを使用することが可能になっています。(厳密に言うと、このパフォーマンス向上は、InnoDB Plugin 1.0.7 以降を含む MySQL 5.1 で実現できます。この向上をデフォルトの InnoDB ストレージエンジンで実現できるのは MySQL 5.5 からです。)

チェックポイント処理の動作のしくみ

InnoDB は、ファジーチェックポイント設定と呼ばれるチェックポイントメカニズムを実装しています。InnoDB は、変更されたデータベースページをバッファープールから小さなバッチにフラッシュします。バッファープー

ルを 1 つのバッチにフラッシュする必要はありません。それを行うと、チェックポイント設定プロセス中にユーザーの SQL ステートメントの処理が中断されます。

クラッシュリカバリ中に、InnoDB は、ログファイルに書き込まれたチェックポイントラベルを探します。それは、そのラベルの前にあるデータベースへのすべての変更がデータベースのディスクイメージ内に存在することを知っています。次に、InnoDB はそのチェックポイントから前方にログファイルをスキャンしながら、ログに記録された変更をデータベースに適用します。

14.10.4 テーブルのデフラグ

セカンダリインデックスへのランダムな挿入やセカンダリインデックスからのランダムな削除によって、インデックスが断片化される場合があります。断片化とは、ディスク上のインデックスページの物理的な順序がページ上のレコードのインデックス順序とかけ離れているか、またはインデックスに割り当てられた 64 ページのブロック内に未使用のページが多数存在することを示します。

断片化の 1 つの現象として、あるテーブルが占めている領域が、本来占めている「はずの」領域より大きいことがあります。それが正確にどの程度かを判定するのは困難です。InnoDB のデータとインデックスはすべて B ツリー内に格納され、それらのフィルファクタは 50% から 100% まで変動する可能性があります。断片化の別の現象として、次のようなテーブルスキャンにかかる時間が、本来かかる「はずの」時間より長いことがあります。

```
SELECT COUNT(*) FROM t WHERE non_indexed_column <> 12345;
```

前のクエリーでは、MySQL が、大きなテーブルに対してもっとも遅いタイプのクエリーであるフルテーブルスキャンを実行する必要があります。

インデックススキャンを高速化するために、MySQL にテーブルを再構築させる次の「null」 ALTER TABLE 操作を定期的に実行できます。

```
ALTER TABLE tbl_name ENGINE=INNODB
```

MySQL 5.6.3 の時点では、ALTER TABLE tbl_name FORCE を使用して、テーブルを再構築する「null」変更操作を実行することもできます。以前は、FORCE オプションは認識されましたが、無視されました。

MySQL 5.6.17 の時点では、ALTER TABLE tbl_name ENGINE=INNODB と ALTER TABLE tbl_name FORCE の両方がオンライン DDL (ALGORITHM=COPY) を使用します。詳細は、セクション14.11.1「オンライン DDL の概要」を参照してください。

デフラグ操作を実行するための別の方法として、mysqldump を使用してテーブルをテキストファイルにダンプし、テーブルを削除してから、それをダンプファイルからリロードする方法があります。

インデックスへの挿入が常に昇順であり、かつレコードが末尾からしか削除されない場合は、InnoDB のファイル領域管理アルゴリズムにより、インデックス内の断片化は発生しないことが保証されます。

14.10.5 TRUNCATE TABLE によるディスク領域の再利用

InnoDB テーブルを切り捨てるときにオペレーティングシステムのディスク領域を再利用するには、そのテーブルが独自の .ibd ファイルに格納されている必要があります。独自の .ibd ファイルに格納されるテーブルの場合は、そのテーブルを作成するときに innodb_file_per_table を有効にする必要があります。さらに、切り捨てられるテーブルとその他のテーブルの間に外部キー制約が存在していてはいけません。そうしないと、TRUNCATE TABLE 操作は失敗します。ただし、同じテーブル内の 2 つのカラム間の外部キー制約は許可されます。

テーブルが切り捨てられると、そのテーブルが削除されて新しい .ibd ファイル内に再作成され、解放された領域はオペレーティングシステムに返されます。これは、InnoDB システムテーブルスペース内に格納されている InnoDB テーブル (innodb_file_per_table=OFF のときに作成されるテーブル) の切り捨てとは対照的です。この場合は、そのテーブルが切り捨てられたあと、解放された領域は InnoDB しか使用できません。

テーブルを切り捨て、そのディスク領域をオペレーティングシステムに返す機能はまた、物理バックアップを小さくすることもできます。システムテーブルスペースに格納されているテーブル (innodb_file_per_table=OFF のときに作成されるテーブル) の切り詰めでは、未使用領域のブロックがシステムテーブルスペース内に残されます。

14.11 InnoDB とオンライン DDL

オンライン DDL 機能は、MySQL 5.1 および MySQL 5.5 で使用可能な機能に基づいて構築されています。機能は、テーブルコピー動作を行わないように CREATE INDEX と DROP INDEX を最適化しました。MySQL 5.6 で導入されたオンライン DDL 機能は、ほかのタイプの多くの ALTER TABLE 操作を、テーブルコピー、DDL が進行中の DML 操作のブロック化、またはその両方を行わないように拡張しています。

オンライン DDL 機能には、次の利点があります。

- インデックスやカラム定義を変更する場合は常に、テーブルを数分または数時間にわたって使用できなくすることが現実的でないビジー状態の本番環境での応答性と可用性を向上させます。
- テーブルへのアクセスを完全にブロックするか (LOCK=EXCLUSIVE 句)、クエリーを許可するが、DML は許可しないか (LOCK=SHARED 句)、またはテーブルへの完全なクエリーおよび DML アクセスを許可するか (LOCK=NONE 句) どうかを選択することによって、DDL 操作中のパフォーマンスと並列性のバランスを調整できます。LOCK 句を省略するか、または LOCK=DEFAULT を指定すると、MySQL は、操作の種類に応じてできるだけ高い並列性を許可します。
- テーブルの新しいコピーを作成するのではなく、可能な場合はインプレースで変更を行うことによって、テーブルのコピーおよびすべてのセカンダリインデックスの再構築のためのディスク領域の使用量や I/O オーバーヘッドの一時的な増加が回避されます。

MySQL クラスタの NDB ストレージエンジンは、オンラインのテーブルスキーマ変更もサポートしていますが、InnoDB のオンライン操作に使用されるものとは互換性のない独自の構文を使用します。詳細は、セクション 13.1.7.2「MySQL Cluster での ALTER TABLE オンライン操作」を参照してください。

14.11.1 オンライン DDL の概要

従来より、InnoDB テーブルでの多くの DDL 操作は高いコストを必要としました。多くの ALTER TABLE 操作は、要求されたテーブルオプションとインデックスを使用して定義された新しい空のテーブルを作成してから、既存の行を新しいテーブルに 1 つずつコピーし、行が挿入されるたびにインデックスを更新することによって機能しました。元のテーブルのすべての行がコピーされたあと、古いテーブルが削除され、そのコピーの名前が元のテーブルの名前に変更されました。

MySQL 5.5、および InnoDB Plugin を含む MySQL 5.1 は、テーブルコピー動作を行わないように CREATE INDEX と DROP INDEX を最適化しました。その機能は、高速インデックス作成と呼ばれました。MySQL 5.6 は、ほかのタイプの多くの ALTER TABLE 操作を、テーブルのコピーを行わないように拡張しています。別の拡張では、テーブルが変更されている最中に SELECT クエリーや INSERT、UPDATE、および DELETE (DML) ステートメントの処理を続行できるようになります。この機能の組み合わせは現在、オンライン DDL と呼ばれます。

この新しいメカニズムはまた、一般に、セカンダリインデックスなしでテーブルを作成し、データがロードされたあとにセカンダリインデックスを追加することによって、テーブルとそれに関連付けられたインデックスを作成およびロードするプロセス全体を高速化できることも示しています。

CREATE INDEX または DROP INDEX コマンドで構文の変更は必要ありませんが、この操作のパフォーマンス、領域使用量、およびセマンティクスに影響を与える要因がいくつかあります (セクション14.11.9「オンライン DDL の制限」を参照してください)。

MySQL 5.6 のオンライン DDL 拡張によって、以前はテーブルコピー、テーブルでの DML 操作のブロック、またはその両方を必要とした多くの DDL 操作が改善されます。表14.5「DDL 操作のオンラインステータスのサマリー」は、ALTER TABLE ステートメントの各種類と、オンライン DDL 機能がそれぞれにどのように適用されるかを示しています。

ALTER TABLE のパーティション化句を除き、パーティション化された InnoDB テーブルに対するオンライン DDL 操作は、通常の InnoDB テーブルに適用されるのと同じルールに従います。詳細は、セクション 14.11.8「パーティション化された InnoDB テーブルに対するオンライン DDL」を参照してください。

- 「インプレース?」カラムは、どの操作の場合に ALGORITHM=INPLACE 句が許可されるかを示しています。推奨される値は「はい」です。
- 「テーブルをコピー?」カラムは、どの操作の場合にコストの高いテーブルコピー操作を回避できるかを示しています。推奨される値は「いいえ」です。ALGORITHM=INPLACE は許可されるが、ある程度の量のテーブルコピーが引き続き必要な操作がいくつかある点を除き、このカラムはほぼ、「インプレース?」カラムの反対です。
- 「並列 DML を許可?」カラムは、どの操作を完全にオンラインで実行できるかを示しています。推奨される値は「はい」です。DDL 中に完全な並列性が許可されることを表明するために LOCK=NONE を指定できますが、MySQL は、可能な場合は自動的にこのレベルの並列性を許可します。並列 DML が許可されている場合は、並列クエリーも常に許可されます。
- 「並列クエリーを許可?」カラムは、どの DDL 操作の場合に、その操作の進行中にテーブルに対するクエリーが許可されるかを示しています。推奨される値は「はい」です。並列クエリーは、すべてのオンライン DDL 操作中に許可されます。これは、参考のために、すべてのセルに表示されている「はい」で示されていま

す。DDL 中に並列クエリーが許可されることを表明するために LOCK=SHARED を指定できますが、MySQL は、可能な場合は自動的にこのレベルの並列性を許可します。

- 「注」カラムは、構成オプションの設定または DDL ステートメント内のほかの句によって答えが異なる場合など、ほかのカラムの「はい/いいえ」の値に例外がある場合はそれについて説明します。「はい*」と「いいえ*」の値は、これらの追加の注によって答えが異なることを示します。

表 14.5 DDL 操作のオンラインステータスのサマリー

操作	インプレース?	テーブルをコピー?	並列 DML を許可?	並列クエリーを許可?	メモ
CREATE INDEX、ADD INDEX	はい*	いいえ*	はい	はい	FULLTEXT インデックスにはいくつかの制限があります。次の行を参照してください。現在は、作成対象の同じインデックスも同じ ALTER TABLE ステートメント内の前の句によって削除された場合、この操作はインプレースではありません (つまり、テーブルをコピーします)。
ADD FULLTEXT INDEX	はい	いいえ*	いいえ	はい	ユーザーが指定した FTS_DOC_ID カラムがないかぎり、テーブルの最初の FULLTEXT インデックスの作成にはテーブルコピーが必要です。同じテーブル上の以降の FULLTEXT インデックスは、インプレースで作成できます。
DROP INDEX	はい	いいえ	はい	はい	データファイルではなく、.frm ファイルのみを変更します。
OPTIMIZE TABLE	はい	はい	はい	はい	MySQL 5.6.17 の時点では、ALGORITHM=INPLACE を使用します。old_alter_table=1 または mysqld --skip-new オプションが有効になっている場合は、ALGORITHM=COPY が使用されます。FULLTEXT インデックスを含むテーブルでは、オンライン DDL (ALGORITHM=INPLACE) を使用した OPTIMIZE TABLE はサポートされません。
カラムのデフォルト値を設定する	はい	いいえ	はい	はい	データファイルではなく、.frm ファイルのみを変更します。
カラムの自動インクリメント値を変更する	はい	いいえ	はい	はい	データファイルではなく、メモリーに格納された値を変更します。
外部キー制約を追加する	はい*	いいえ*	はい	はい	テーブルのコピーを行わないようにするには、制約の作成中に foreign_key_checks を無効にします。
外部キー制約を削除する	はい	いいえ	はい	はい	foreign_key_checks オプションを有効または無効にすることができます。
カラムを名前変更する	はい*	いいえ*	はい*	はい	並列 DML を許可するには、同じデータ型を維持し、カラム名のみを変更します。
カラムを追加する	はい	はい	はい*	はい	自動インクリメントカラムを追加する場合は、並列 DML が許可されません。ALGORITHM=INPLACE は許可されますが、データが大幅に再編成されるため、依然としてコストの高い操作です。

操作	インプレース?	テーブルをコピー?	並列 DML を許可?	並列クエリーを許可?	メモ
カラムを削除する	はい	はい	はい	はい	ALGORITHM=INPLACE は許可されますが、データが大幅に再編成されるため、依然としてコストの高い操作です。
カラムを並べ替える	はい	はい	はい	はい	ALGORITHM=INPLACE は許可されますが、データが大幅に再編成されるため、依然としてコストの高い操作です。
ROW_FORMAT プロパティーを変更する	はい	はい	はい	はい	ALGORITHM=INPLACE は許可されますが、データが大幅に再編成されるため、依然としてコストの高い操作です。
KEY_BLOCK_SIZE プロパティーを変更する	はい	はい	はい	はい	ALGORITHM=INPLACE は許可されますが、データが大幅に再編成されるため、依然としてコストの高い操作です。
カラム NULL を作成する	はい	はい	はい	はい	ALGORITHM=INPLACE は許可されますが、データが大幅に再編成されるため、依然としてコストの高い操作です。
カラム NOT NULL を作成する	はい*	はい	はい	はい	SQL_MODE に strict_all_tables または strict_all_tables が含まれている場合は、カラムに Null が含まれていると操作は失敗します。ALGORITHM=INPLACE は許可されますが、データが大幅に再編成されるため、依然としてコストの高い操作です。
カラムのデータ型を変更する	いいえ	はい	いいえ	はい	
主キーを追加する	はい*	はい	はい	はい	ALGORITHM=INPLACE は許可されますが、データが大幅に再編成されるため、依然としてコストの高い操作です。カラムを NOT NULL に変換する必要がある場合は、特定の状況では ALGORITHM=INPLACE が許可されません。例14.9「主キーの作成および削除」を参照してください。
主キーを削除して別の主キーを追加する	はい	はい	はい	はい	ALGORITHM=INPLACE は、同じ ALTER TABLE で新しい主キーを追加する場合にのみ許可されます。データが大幅に再編成されるため、これは依然としてコストの高い操作です。
主キーを削除する	いいえ	はい	いいえ	はい	同じ ALTER TABLE ステートメントで新しい主キーを追加することなく主キーを削除する場合は、制限が適用されます。
文字セットを変換する	いいえ	はい	いいえ	はい	新しい文字エンコーディングが別のものである場合は、テーブルを再構築します。
文字セットを指定する	いいえ	はい	いいえ	はい	新しい文字エンコーディングが別のものである場合は、テーブルを再構築します。
FORCE オプションを使用して再構築する	はい	はい	はい	はい	MySQL 5.6.17 の時点では、ALGORITHM=INPLACE を

操作	インプレース?	テーブルをコピー?	並列 DML を許可?	並列クエリーを許可?	メモ
					使用します。old_alter_table=1 または mysqld --skip-new オプションが有効になっている場合は、ALGORITHM=COPY が使用されます。FULLTEXT インデックスを含むテーブルでは、オンライン DDL (ALGORITHM=INPLACE) を使用したテーブル再構築はサポートされません。
「null」 ALTER TABLE ... ENGINE=INNODB を使用して再構築する	はい	はい	はい	はい	MySQL 5.6.17 の時点では、ALGORITHM=INPLACE を使用します。old_alter_table=1 または mysqld --skip-new オプションが有効になっている場合は、ALGORITHM=COPY が使用されます。FULLTEXT インデックスを含むテーブルでは、オンライン DDL (ALGORITHM=INPLACE) を使用したテーブル再構築はサポートされません。
テーブルレベルの永続的統計オプション (STATS_PERSISTENT、STATS_AUTO_RECALC STATS_SAMPLE_PAGES) を設定する	はい	いいえ	はい	はい	データファイルではなく、.frm ファイルのみを変更します。

次の各セクションでは、並列 DML、インプレース、またはその両方で実行できる主な操作のそれぞれについて、オンライン DDL に関連した基本的な構文と使用上の注意を示します。

セカンダリインデックス

- セカンダリインデックスを作成する: CREATE INDEX name ON table (col_list) または ALTER TABLE table ADD INDEX name (col_list)。(FULLTEXT インデックスの作成にはテーブルのロックが引き続き必要です。)
- セカンダリインデックスを削除する: DROP INDEX name ON table; または ALTER TABLE table DROP INDEX name

InnoDB テーブルでのセカンダリインデックスの作成および削除では、MySQL 5.5 や InnoDB Plugin を含む MySQL 5.1 と同様に、テーブルコピー動作がスキップされます。

MySQL 5.6 以降では、インデックスが作成または削除されている間も、そのテーブルの読み取りおよび書き込み操作は可能なままです。CREATE INDEX または DROP INDEX ステートメントは、インデックスの初期状態にテーブルの最新の内容が反映されるように、そのテーブルにアクセスしているすべてのトランザクションが完了したあとでのみ完了します。以前は、インデックスが作成または削除されている間にテーブルを変更すると、通常はデッドロックが発生し、それによりテーブルでの INSERT、UPDATE、または DELETE ステートメントが取り消されました。

カラムのプロパティー

- カラムのデフォルト値を設定する: ALTER TABLE tbl ALTER COLUMN col SET DEFAULT literal または ALTER TABLE tbl ALTER COLUMN col DROP DEFAULT

  カラムのデフォルト値は、InnoDB データディクショナリではなく、そのテーブルの .frm ファイルに格納されます。

- カラムの自動インクリメント値の変更: ALTER TABLE table AUTO_INCREMENT=next_value;

  特に、レプリケーションまたはシャーディングを使用した分散システムでは、テーブルの自動インクリメントカウンタを特定の値にリセットする場合があります。テーブルに挿入された次の行は、その自動インクリメントカラムの指定された値を使用します。この手法はまた、すべてのテーブルを定期的に空にしてリロードするデータウェアハウス環境でも使用できます。それにより、自動インクリメントのシーケンスを 1 から再開できます。

- カラムの名前変更: ALTER TABLE tbl CHANGE old_col_name new_col_name datatype

  同じデータ型と [NOT] NULL 属性を維持して、カラム名のみを変更する場合、この操作は常にオンラインで実行できます。

  この拡張の一部として、外部キー制約の一部であるカラムを名前変更できるようになりました。これは、以前は許可されていませんでした。外部キー定義は、新しいカラム名を使用するように自動的に更新されます。外部キーに参加しているカラムの名前変更は、ALTER TABLE のインプレースモードでのみ機能します。ALGORITHM=COPY 句を使用するか、またはその他の何らかの条件によってコマンドが内部的に ALGORITHM=COPY を使用した場合、ALTER TABLE ステートメントは失敗します。

外部キー

- 外部キー制約の追加または削除:

```
ALTER TABLE tbl1 ADD CONSTRAINT fk_name FOREIGN KEY index (col1) REFERENCES tbl2(col2) referential_actions;
ALTER TABLE tbl DROP FOREIGN KEY fk_name;
```

  外部キーの削除は、foreign_key_checks オプションが有効または無効になった状態でオンラインで実行できます。外部キーをオンラインで作成するには、foreign_key_checks が無効になっている必要があります。

  特定のテーブル上の外部キー制約の名前がわからない場合は、次のステートメントを発行し、各外部キーに対する CONSTRAINT 句で制約名を見つけます。

```
show create table table\G
```

  または、information_schema.table_constraints テーブルをクエリーし、constraint_name および constraint_type カラムを使用して外部キー名を識別します。

  この拡張の結果として、外部キーとそれに関連付けられたインデックスを 1 つのステートメントで削除することも可能になりました。これは以前、厳密な順序で並べられた個別のステートメントを必要としました。

```
ALTER TABLE table DROP FOREIGN KEY constraint, DROP INDEX index;
```

変更されるテーブル内に外部キーがすでに存在する (つまり、そのテーブルがいずれかの FOREIGN KEY ... REFERENCE 句を含む子テーブルである) 場合は、外部キーカラムに直接関連しない操作であっても、オンライン DDL 操作には次の追加の制限が適用されます。

- このような子テーブルでのオンライン DDL 操作中は、並列 DML が許可されません。(この制限はバグとして評価中であり、解除される可能性があります。)
- 親テーブルが変更されたために、CASCADE または SET NULL パラメータを使用した ON UPDATE または ON DELETE 句によって子テーブル内で関連する変更が発生した場合は、子テーブルに対する ALTER TABLE が別のトランザクションのコミットも待機する可能性があります。

同様に、あるテーブルが外部キー関係にある親テーブルである場合、そこには FOREIGN KEY 句が含まれていなくても、INSERT、UPDATE、または DELETE ステートメントによって子テーブル内で ON UPDATE または ON DELETE アクションが発生した場合は、そのテーブルが ALTER TABLE の完了を待機する可能性があります。

ALGORITHM=COPY に関する注意

ALGORITHM=COPY 句で実行される ALTER TABLE 操作はすべて、並列 DML 操作を妨げます。並列クエリーは、引き続き許可されます。つまり、テーブルコピー操作には常に、少なくとも LOCK=SHARED (クエリーを許可するが、DML は許可しない) の並列性の制限が含まれます。LOCK=EXCLUSIVE (DML とクエリーを妨げる) を指定することによって、このような操作の並列性をさらに制限できます。

並列 DML ではあるが、テーブルコピーが引き続き必要

その他の一部の ALTER TABLE 操作では、並列 DML は許可されますが、テーブルコピーが引き続き必要です。ただし、これらの操作のテーブルコピーは MySQL 5.5 以前の処理と比べて高速です。

- カラムの追加、削除、または並べ替え。
- 主キーの追加または削除。
- テーブルの ROW_FORMAT または KEY_BLOCK_SIZE プロパティーの変更。
- カラムの Null にできるステータスの変更。
- OPTIMIZE TABLE

- FORCE オプションを使用したテーブルの再構築
- 「null」 ALTER TABLE ... ENGINE=INNODB ステートメントを使用したテーブルの再構築

> 注記
>
> 新しいカラム、データ型、制約、インデックスなどによってデータベーススキーマが進化するにつれ、CREATE TABLE ステートメントを最新のテーブル定義が適用されるように維持してください。オンライン DDL のパフォーマンス向上があったとしても、スキーマの一部を作成し、そのあとに ALTER TABLE ステートメントを発行するより、最初から安定したデータベース構造を作成する方が効率的です。
>
> このガイドラインの主な例外は、多数の行を含むテーブル上のセカンダリインデックスに関するものです。通常は、セカンダリインデックスを除き、すべての詳細が指定された状態でテーブルを作成し、データをロードしてから、セカンダリインデックスを作成する方法がもっとも効率的です。初期のデータがクリーンであることがわかっていて、ロードプロセス中に一貫性チェックが必要ない場合は、外部キーでも同じ手法を使用できます (最初にデータをロードしてから、外部キーを設定します)。
>
> CREATE TABLE、CREATE INDEX、ALTER TABLE、および同様のステートメントのどのようなシーケンスでテーブルを作成した場合でも、ステートメント SHOW CREATE TABLE table\G (正式な形式には大文字の \G が必要です) を発行することによって、現在の形式のテーブルを再構築するために必要な SQL を取得できます。この出力には、場合によっては内部的に追加され、また新しいシステム上でのテーブルのクローニングや、同一の型を持つ外部キーカラムの設定を行うときに通常であれば省略する可能性がある数値精度、NOT NULL、CHARACTER SET などの句が示されます。

14.11.2 オンライン DDL でのパフォーマンスと並列性に関する考慮事項

オンライン DDL によって、パフォーマンス、並列性、可用性、スケーラビリティーなどの MySQL 操作のいくつかの側面が改善されます。

- テーブルでのクエリーや DML 操作は DDL の進行中も処理を続行できるため、そのテーブルにアクセスするアプリケーションの応答性が向上します。MySQL サーバー全体にわたってほかのリソースのロックや待機が削減されるため、変更されるテーブルには関連しない操作であっても、スケーラビリティーの向上がもたらされます。
- インプレース操作では、テーブルを再構築するためのディスク I/O や CPU サイクルが回避されるため、データベースにかかる全体的な負荷が最小限に抑えられるとともに、DDL 操作中に良好なパフォーマンスと高いスループットが維持されます。
- インプレース操作では、すべてのデータがコピーされる場合に比べてバッファープールに読み取られるデータが削減されるため、以前は DDL 操作のあとの一時的なパフォーマンス低下の原因になる可能性のあった、頻繁にアクセスされるデータのメモリーからのパージが回避されます。

オンライン操作に一時ファイルが必要な場合、InnoDB はそれらのファイルを元のテーブルを含むディレクトリではなく、一時ファイルディレクトリ内に作成します。このディレクトリがそのようなファイルを保持するほどに十分に大きくない場合は、tmpdir システム変数に別のディレクトリを設定する必要があることがあります。(セクションB.5.4.4「MySQL が一時ファイルを格納する場所」を参照してください。)

オンライン DDL のロックオプション

InnoDB テーブルが DDL 操作によって変更されている間は、その操作の内部動作や ALTER TABLE ステートメントの LOCK 句に応じて、そのテーブルはロックされる場合とされない場合があります。デフォルトでは、MySQL は DDL 操作中にできるだけ少ないロックを使用します。この句は、ロックを通常の場合より制限的にする (それによって並列 DML、または DML とクエリーを制限する) ためか、またはある操作に対して何らかの期待されるレベルのロックが確実に許可されるようにするために指定します。主キーの作成または削除中に LOCK 句がその特定の種類の DDL 操作では使用できないロックのレベル (LOCK=SHARED や LOCK=NONE など) を指定している場合は、この句が表明のように機能するため、このステートメントはエラーで失敗します。次のリストは、もっとも許容的な場合からもっとも制限的な場合までの LOCK 句のさまざまな可能性を示しています。

- LOCK=NONE を使用した DDL 操作では、クエリーと並列 DML の両方が許可されます。この句は、要求されたロックのタイプでこの種類の DDL 操作を実行できないと ALTER TABLE を失敗させるため、LOCK=NONE は、テーブルを完全に使用可能な状態に維持することが不可欠であり、かつそれができなければ DDL を取り消してもかまわない場合に指定します。たとえば、この句は、顧客のサインアップまたは購入に関連するテーブルの DDL で、コストの高い ALTER TABLE ステートメントの誤った発行によってこれらのテーブルが使用不可能にならないようにするために使用できます。

- LOCK=SHARED を使用した DDL 操作では、テーブルへの書き込み (つまり、DML 操作) がすべてブロックされますが、そのテーブル内のデータは読み取ることができます。この句は、要求されたロックのタイプでこの種類の DDL 操作を実行できないと ALTER TABLE を失敗させるため、LOCK=SHARED は、テーブルをクエリーに対して使用可能な状態に維持することが不可欠であり、かつそれができなければ DDL を取り消してもかまわない場合に指定します。たとえば、この句は、DDL が完了するまでデータロード操作を遅らせてもかまわないが、クエリーを長時間遅らせることはできないデータウェアハウス内のテーブルの DDL で使用できます。
- LOCK=DEFAULT を使用するか、または LOCK 句が省略された DDL 操作では、MySQL はその種類の操作で使用可能なもっとも低いレベルのロックを使用することにより、可能な場合は常に並列クエリー、DML、またはその両方を許可します。これは、そのテーブルのワークロードに基づいて可用性に関する問題が発生しないことがわかっている、事前に計画およびテストされた変更を行う場合に使用する設定です。
- LOCK=EXCLUSIVE を使用した DDL 操作では、クエリーと DML 操作の両方がブロックされます。この句は、要求されたロックのタイプでこの種類の DDL 操作を実行できないと ALTER TABLE を失敗させるため、LOCK=EXCLUSIVE は、主な関心事が DDL を可能性のある最短の時間で完了させることであり、かつテーブルにアクセスしようとするアプリケーションを待機させてもかまわない場合に指定します。LOCK=EXCLUSIVE はまた、テーブルへの予期しないアクセスを回避するために、サーバーがアイドル状態であると想定される場合にも使用できます。

InnoDB テーブルに対するオンライン DDL ステートメントは、DDL ステートメントの準備中に短時間だけテーブルへの排他的アクセスが必要なため、そのテーブルにアクセスしている現在実行中のトランザクションがコミットまたはロールバックするのを常に待機します。同様に、完了前にも、短時間だけテーブルへの排他的アクセスが必要です。そのため、オンライン DDL ステートメントは、その DDL が完了する前に、DDL の進行中に開始され、テーブルをクエリーまたは変更するすべてのトランザクションがコミットまたはロールバックするのを待機します。

並列 DML 操作によって行われた変更を記録したあと、最後にこれらの変更を適用するにはある程度の処理が必要であるため、オンライン DDL 操作には、ほかのセッションからのテーブルアクセスをブロックする古いスタイルのメカニズムに比べて全体的に長い時間がかかる可能性があります。raw パフォーマンスの低下は、そのテーブルを使用するアプリケーションの応答性の向上とバランスがとれています。テーブル構造を変更するための理想的な手法を評価する場合は、Web ページのロード時間などの要因に基づいて、エンドユーザーのパフォーマンスの認識を考慮してください。

新しく作成された InnoDB セカンダリインデックスには、CREATE INDEX または ALTER TABLE ステートメントが実行を完了した時点でのテーブル内のコミットされたデータのみが含まれています。コミットされていない値や古いバージョンの値、または削除対象としてマークされているが、まだ古いインデックスから削除されていない値は含まれていません。

インプレース DDL 操作とテーブルコピー DDL 操作のパフォーマンスの比較

オンライン DDL 操作の raw パフォーマンスは、その操作がインプレースで実行されるか、またはテーブル全体のコピーと再構築が必要かによってほとんど決定されます。インプレースで実行できる操作の種類や、テーブルコピー操作を行わないための何らかの要件を確認するには、表14.5「DDL 操作のオンラインステータスのサマリー」を参照してください。

インプレース DDL のパフォーマンス向上は、主キーのインデックスではなく、セカンダリインデックスに対する操作に適用されます。InnoDB テーブルの行は、主キーに基づいて編成されたクラスタ化されたインデックスに格納されます。これにより、一部のデータベースシステムで「インデックス編成テーブル」と呼ばれるものが形成されます。このテーブル構造は主キーにきわめて密接に結び付けられているため、主キーの再定義にはデータのコピーが引き続き必要です。

主キーに対する操作で ALGORITHM=INPLACE が使用される場合は、データが引き続きコピーされるにもかかわらず、次の理由で ALGORITHM=COPY を使用するより効率的です。

- ALGORITHM=INPLACE には、Undo ロギングやそれに関連する Redo ロギングが必要ありません。これらの操作は、ALGORITHM=COPY を使用する DDL ステートメントのオーバーヘッドを増やします。
- セカンダリインデックスエントリは事前にソートされているため、順番にロードできます。
- セカンダリインデックスへのランダムアクセス挿入は存在しないため、変更バッファーは使用されません。

オンライン DDL 操作の相対的なパフォーマンスを評価するには、現在のバージョンと以前のバージョンの MySQL を使用して、このような操作を大きな InnoDB テーブルで実行できます。また、すべてのパフォーマンステストを最新の MySQL バージョンで実行することもできます。つまり、old_alter_table システム変数を設定することにより、以前の DDL 動作をシミュレートして「前の」結果を求めます。セッション内でステートメント set old_alter_table=1 を発行し、DDL パフォーマンスを測定して「前の」数値を記録します。次に、set old_alter_table=0 を発行して新しい、高速な動作を再度有効にし、DDL 操作を再度実行して「あとの」数値を記録します。

DDL 操作がその変更をインプレースで行うか、またはテーブルコピーを実行するかの基本的な考え方については、コマンドが完了したあとに表示される「rows affected」の値を見てください。たとえば、さまざまなタイプの DDL 操作を実行したあとに表示される可能性のある行を次に示します。

- カラムのデフォルト値の変更 (非常に高速であり、テーブルデータにはまったく影響を与えません):

```
Query OK, 0 rows affected (0.07 sec)
```

- インデックスの追加 (時間はかかりますが、0 rows affected はテーブルがコピーされないことを示しています):

```
Query OK, 0 rows affected (21.42 sec)
```

- カラムのデータ型の変更 (かなりの時間がかかり、テーブルのすべての行の再構築が必要です):

```
Query OK, 1671168 rows affected (1 min 35.54 sec)
```

たとえば、大きなテーブルで DDL 操作を実行する前に、その操作が速いか遅いかを次のようにチェックできます。

1. テーブル構造をクローニングします。
2. クローニングされたテーブルに非常に少量のデータを移入します。
3. クローニングされたテーブルで DDL 操作を実行します。
4. 「rows affected」の値が 0 かどうかをチェックします。0 以外の値は、この操作にはテーブル全体の再構築が必要なため、特別な計画が必要になる可能性があることを示します。たとえば、この DDL 操作をスケジュールされたダウンタイムの期間中に、または各レプリケーションスレーブサーバー上で一度に 1 つずつ実行することができます。

MySQL 処理の削減をより深く理解するには、DDL 操作の前後に InnoDB に関連した performance_schema および INFORMATION_SCHEMA テーブルを検査して、物理的な読み取り、書き込み、メモリー割り当てなどの数を確認します。

14.11.3 オンライン DDL の SQL 構文

通常、InnoDB テーブルに対して ALTER TABLE ステートメントを使用する場合、オンライン DDL を有効にするために特殊なことをする必要はありません。インプレースで実行できるか、並列 DML を許可するか、またはその両方の DDL 操作の種類については、表14.5「DDL 操作のオンラインステータスのサマリー」を参照してください。一部の種類には、構成設定または ALTER TABLE 句の特定の組み合わせが必要になります。

ALTER TABLE ステートメントの LOCK および ALGORITHM 句を使用することによって、特定のオンライン DDL 操作のさまざまな側面を制御できます。これらの句はステートメントの最後に現れ、テーブルやカラムの指定とはカンマで区切られます。LOCK 句は、テーブルへの並列アクセスの程度を微調整するのに役立ちます。ALGORITHM 句は、主にパフォーマンス比較を目的にしているほか、既存の DDL コードで何らかの問題が発生した場合の古いテーブルコピー動作へのフォールバックとしても使用されます。例:

- テーブルの読み取り、書き込み、またはその両方を誤って不可能にしてしまうことがないようにするには、ALTER TABLE ステートメントで LOCK=NONE (読み取りと書き込みの両方を許可する) や LOCK=SHARED (読み取りを許可する) などの句を指定できます。要求されたレベルの並列性が使用できない場合、操作はただちに停止します。
- パフォーマンスを比較するには、old_alter_table 構成オプションを設定する代わりに、1 つのステートメントを ALGORITHM=INPLACE で、もう 1 つのステートメントを ALGORITHM=COPY で実行できます。
- テーブルをコピーした ALTER TABLE を実行することによってサーバーが結合されてしまう可能性をなくすには、インプレースメカニズムを使用できなければステートメントがただちに停止するように ALGORITHM=INPLACE を含めることができます。インプレースで実行できるか、またはできない DDL 操作のリストについては、表14.5「DDL 操作のオンラインステータスのサマリー」を参照してください。

LOCK 句の詳細は、セクション14.11.2「オンライン DDL でのパフォーマンスと並列性に関する考慮事項」を参照してください。オンライン DDL の使用の完全な例については、セクション14.11.5「オンライン DDL の例」を参照してください。

14.11.4 DDL ステートメントの結合または分離

オンライン DDL が導入される前は、多くの DDL 操作を 1 つの ALTER TABLE ステートメントに結合することが一般的な習慣でした。各 ALTER TABLE ステートメントにはテーブルのコピーと再構築が含まれていたため、テーブルに対するすべての変更を 1 回の再構築操作で実行できたことから、同じテーブルへのいくつかの変更を

一度に行う方が効率的でした。マイナス面としては、DDL 操作に関連する SQL コードが保守しにくく、別のスクリプトでの再利用も難しい点がありました。特定の変更が毎回異なっていたとすると、少し異なるシナリオごとに、新しい複雑な ALTER TABLE の構築が必要になる可能性があります。

インプレースで実行できる DDL 操作の場合は、表14.5「DDL 操作のオンラインステータスのサマリー」に示すように、効率を犠牲にすることなく、スクリプト作成や保守を容易にするために DDL 操作を個々の ALTER TABLE ステートメントに分離できるようになりました。たとえば、次のような複雑なステートメントを取り上げ、

```
alter table t1 add index i1(c1), add unique index i2(c2), change c4_old_name c4_new_name integer unsigned;
```

それを独立してテストおよび実行できる、次のようなより簡単な部分に分解することができます。

```
alter table t1 add index i1(c1);
alter table t1 add unique index i2(c2);
alter table t1 change c4_old_name c4_new_name integer unsigned not null;
```

複数の部分からなる ALTER TABLE ステートメントは、次の目的に引き続き使用できます。

- 特定のシーケンスで実行する必要のある操作。たとえば、インデックスの作成に続けて、そのインデックスを使用する外部キー制約を作成する場合など。
- グループとして成功または失敗するようにしたい、すべてが同じ特定の LOCK 句を使用している操作。
- インプレースで実行できない、つまり、引き続きテーブルをコピーして再構築する操作。
- 特殊なシナリオでの正確な下位互換性のために必要な場合に強制的にテーブルコピー動作を行うために、ALGORITHM=COPY または old_alter_table=1 を指定する操作。

14.11.5 オンライン DDL の例

次の各コード例は、そのパフォーマンス、並列性、およびスケーラビリティーが最新のオンライン DDL 拡張によって向上したいくつかの操作を示しています。

- 例14.1「オンライン DDL 実験のためのスキーマ設定コード」では、以降の例で使用される BIG_TABLE と SMALL_TABLE という名前のテーブルを設定します。
- 例14.2「CREATE INDEX および DROP INDEX の速度と効率」は、インデックスの作成および削除のパフォーマンスの側面を示しています。
- 例14.3「CREATE INDEX および DROP INDEX 中の並列 DML」は、DROP INDEX 操作中に実行されるクエリーと DML ステートメントを示しています。
- 例14.4「カラムの名前変更」は、カラムの名前変更の速度向上、および名前変更操作を行うときにデータ型を正確に同じ状態に維持するために必要な注意事項を示しています。
- 例14.5「外部キーの削除」は、外部キーがオンライン DDL でどのように機能するかを示しています。外部キーの操作には 2 つのテーブルが関連しているため、ロックに関する追加の考慮事項があります。そのため、外部キーを含むテーブルには、オンライン DDL 操作での制限が存在する場合があります。
- 例14.6「自動インクリメント値の変更」は、自動インクリメントカラムがオンライン DDL でどのように機能するかを示しています。自動インクリメントカラムを含むテーブルには、オンライン DDL 操作での制限が存在する場合があります。
- 例14.7「LOCK 句を使用した並列性の制御」は、オンライン DDL 操作の進行中の並列クエリーと DML 操作を許可または制限するためのオプションを示しています。これは、DDL ステートメントが待機するか、並列トランザクションが待機するか、またはデッドロックエラーのために並列トランザクションが DML ステートメントを取り消す可能性がある状況を示しています。
- 例14.8「オンライン DDL 実験のためのスキーマ設定コード」は、1 つのステートメントでの複数のインデックスの作成および削除を示しています。これは、インデックス操作ごとに個別のステートメントを使用するより効率的である場合があります。
- 例14.9「主キーの作成および削除」は、主キーはテーブルの作成時に定義する方が効率的であり、あとで追加した場合はコストがかなり高くなることを示しています。

例 14.1 オンライン DDL 実験のためのスキーマ設定コード

これらのデモで使用される初期のテーブルを設定するコードを次に示します。

```
/*
```

```
Setup code for the online DDL demonstration:
- Set up some config variables.
- Create 2 tables that are clones of one of the INFORMATION_SCHEMA tables
  that always has some data. The "small" table has a couple of thousand rows.
  For the "big" table, keep doubling the data until it reaches over a million rows.
- Set up a primary key for the sample tables, since we are demonstrating InnoDB aspects.
*/

set autocommit = 0;
set foreign_key_checks = 1;
set global innodb_file_per_table = 1;
set old_alter_table=0;
prompt mysql:

use test;

\! echo "Setting up 'small' table:"
drop table if exists small_table;
create table small_table as select * from information_schema.columns;
alter table small_table add id int unsigned not null primary key auto_increment;
select count(id) from small_table;

\! echo "Setting up 'big' table:"
drop table if exists big_table;
create table big_table as select * from information_schema.columns;
show create table big_table\G

insert into big_table select * from big_table;
insert into big_table select * from big_table;
insert into big_table select * from big_table;
insert into big_table select * from big_table;
insert into big_table select * from big_table;
insert into big_table select * from big_table;
insert into big_table select * from big_table;
insert into big_table select * from big_table;
insert into big_table select * from big_table;
insert into big_table select * from big_table;
commit;

alter table big_table add id int unsigned not null primary key auto_increment;
select count(id) from big_table;
```

このコードを実行すると、簡略化のために圧縮され、もっとも重要な点が太字で示された次の出力が得られます。

```
Setting up 'small' table:
Query OK, 0 rows affected (0.01 sec)

Query OK, 1678 rows affected (0.13 sec)
Records: 1678  Duplicates: 0  Warnings: 0

Query OK, 1678 rows affected (0.07 sec)
Records: 1678  Duplicates: 0  Warnings: 0

+-----------+
| count(id) |
+-----------+
|      1678 |
+-----------+
1 row in set (0.00 sec)

Setting up 'big' table:
Query OK, 0 rows affected (0.16 sec)

Query OK, 1678 rows affected (0.17 sec)
Records: 1678  Duplicates: 0  Warnings: 0

*************************** 1. row ***************************
       Table: big_table
Create Table: CREATE TABLE `big_table` (
  `TABLE_CATALOG` varchar(512) CHARACTER SET utf8 NOT NULL DEFAULT '',
  `TABLE_SCHEMA` varchar(64) CHARACTER SET utf8 NOT NULL DEFAULT '',
  `TABLE_NAME` varchar(64) CHARACTER SET utf8 NOT NULL DEFAULT '',
  `COLUMN_NAME` varchar(64) CHARACTER SET utf8 NOT NULL DEFAULT '',
  `ORDINAL_POSITION` bigint(21) unsigned NOT NULL DEFAULT '0',
  `COLUMN_DEFAULT` longtext CHARACTER SET utf8,
  `IS_NULLABLE` varchar(3) CHARACTER SET utf8 NOT NULL DEFAULT '',
  `DATA_TYPE` varchar(64) CHARACTER SET utf8 NOT NULL DEFAULT '',
  `CHARACTER_MAXIMUM_LENGTH` bigint(21) unsigned DEFAULT NULL,
  `CHARACTER_OCTET_LENGTH` bigint(21) unsigned DEFAULT NULL,
```

```
  `NUMERIC_PRECISION` bigint(21) unsigned DEFAULT NULL,
  `NUMERIC_SCALE` bigint(21) unsigned DEFAULT NULL,
  `DATETIME_PRECISION` bigint(21) unsigned DEFAULT NULL,
  `CHARACTER_SET_NAME` varchar(32) CHARACTER SET utf8 DEFAULT NULL,
  `COLLATION_NAME` varchar(32) CHARACTER SET utf8 DEFAULT NULL,
  `COLUMN_TYPE` longtext CHARACTER SET utf8 NOT NULL,
  `COLUMN_KEY` varchar(3) CHARACTER SET utf8 NOT NULL DEFAULT '',
  `EXTRA` varchar(30) CHARACTER SET utf8 NOT NULL DEFAULT '',
  `PRIVILEGES` varchar(80) CHARACTER SET utf8 NOT NULL DEFAULT '',
  `COLUMN_COMMENT` varchar(1024) CHARACTER SET utf8 NOT NULL DEFAULT ''
) ENGINE=InnoDB DEFAULT CHARSET=latin1
1 row in set (0.00 sec)

Query OK, 1678 rows affected (0.09 sec)
Records: 1678  Duplicates: 0  Warnings: 0

Query OK, 3356 rows affected (0.07 sec)
Records: 3356  Duplicates: 0  Warnings: 0

Query OK, 6712 rows affected (0.17 sec)
Records: 6712  Duplicates: 0  Warnings: 0

Query OK, 13424 rows affected (0.44 sec)
Records: 13424  Duplicates: 0  Warnings: 0

Query OK, 26848 rows affected (0.63 sec)
Records: 26848  Duplicates: 0  Warnings: 0

Query OK, 53696 rows affected (1.72 sec)
Records: 53696  Duplicates: 0  Warnings: 0

Query OK, 107392 rows affected (3.02 sec)
Records: 107392  Duplicates: 0  Warnings: 0

Query OK, 214784 rows affected (6.28 sec)
Records: 214784  Duplicates: 0  Warnings: 0

Query OK, 429568 rows affected (13.25 sec)
Records: 429568  Duplicates: 0  Warnings: 0

Query OK, 859136 rows affected (28.16 sec)
Records: 859136  Duplicates: 0  Warnings: 0

Query OK, 0 rows affected (0.03 sec)

Query OK, 1718272 rows affected (1 min 9.22 sec)
Records: 1718272  Duplicates: 0  Warnings: 0

+-----------+
| count(id) |
+-----------+
|   1718272 |
+-----------+
1 row in set (1.75 sec)
```

例 14.2 CREATE INDEX および DROP INDEX の速度と効率

次のステートメントシーケンスは、CREATE INDEX および DROP INDEX ステートメントの相対的な速度を示しています。小さなテーブルの場合は、速い方法と遅い方法のどちらを使用しても経過時間は 1 秒未満であるため、「rows affected」の出力を見て、どちらの操作がテーブル再構築を回避できるかを確認します。大きなテーブルの場合は、テーブル再構築のスキップによってかなりの時間が節約されるため、効率の違いは明らかです。

```
\! clear

\! echo "=== Create and drop index (small table, new/fast technique) ==="
\! echo
\! echo "Data size (kilobytes) before index created: "
\! du -k data/test/small_table.ibd
create index i_dtyp_small on small_table (data_type), algorithm=inplace;
\! echo "Data size after index created: "
\! du -k data/test/small_table.ibd
drop index i_dtyp_small on small_table, algorithm=inplace;

-- Compare against the older slower DDL.

\! echo "=== Create and drop index (small table, old/slow technique) ==="
\! echo
\! echo "Data size (kilobytes) before index created: "
\! du -k data/test/small_table.ibd
```

```
create index i_dtyp_small on small_table (data_type), algorithm=copy;
\! echo "Data size after index created: "
\! du -k data/test/small_table.ibd
drop index i_dtyp_small on small_table, algorithm=copy;

-- In the above example, we examined the "rows affected" number,
-- ideally looking for a zero figure. Let's try again with a larger
-- sample size, where we'll see that the actual time taken can
-- vary significantly.

\! echo "=== Create and drop index (big table, new/fast technique) ==="
\! echo
\! echo "Data size (kilobytes) before index created: "
\! du -k data/test/big_table.ibd
create index i_dtyp_big on big_table (data_type), algorithm=inplace;
\! echo "Data size after index created: "
\! du -k data/test/big_table.ibd
drop index i_dtyp_big on big_table, algorithm=inplace;

\! echo "=== Create and drop index (big table, old/slow technique) ==="
\! echo
\! echo "Data size (kilobytes) before index created: "
\! du -k data/test/big_table.ibd
create index i_dtyp_big on big_table (data_type), algorithm=copy;
\! echo "Data size after index created: "
\! du -k data/test/big_table.ibd
drop index i_dtyp_big on big_table, algorithm=copy;
```

このコードを実行すると、簡略化のために圧縮され、もっとも重要な点が太字で示された次の出力が得られます。

```
Query OK, 0 rows affected (0.00 sec)

=== Create and drop index (small table, new/fast technique) ===

Data size (kilobytes) before index created:
384  data/test/small_table.ibd
Query OK, 0 rows affected (0.04 sec)
Records: 0  Duplicates: 0  Warnings: 0

Data size after index created:
432  data/test/small_table.ibd
Query OK, 0 rows affected (0.02 sec)
Records: 0  Duplicates: 0  Warnings: 0

Query OK, 0 rows affected (0.00 sec)

=== Create and drop index (small table, old/slow technique) ===

Data size (kilobytes) before index created:
432  data/test/small_table.ibd
Query OK, 1678 rows affected (0.12 sec)
Records: 1678  Duplicates: 0  Warnings: 0

Data size after index created:
448  data/test/small_table.ibd
Query OK, 1678 rows affected (0.10 sec)
Records: 1678  Duplicates: 0  Warnings: 0

Query OK, 0 rows affected (0.00 sec)

=== Create and drop index (big table, new/fast technique) ===

Data size (kilobytes) before index created:
315392  data/test/big_table.ibd
Query OK, 0 rows affected (33.32 sec)
Records: 0  Duplicates: 0  Warnings: 0

Data size after index created:
335872  data/test/big_table.ibd
Query OK, 0 rows affected (0.02 sec)
Records: 0  Duplicates: 0  Warnings: 0

Query OK, 0 rows affected (0.00 sec)

=== Create and drop index (big table, old/slow technique) ===

Data size (kilobytes) before index created:
335872  data/test/big_table.ibd
Query OK, 1718272 rows affected (1 min 5.01 sec)
```

```
Records: 1718272  Duplicates: 0  Warnings: 0

Data size after index created:
348160  data/test/big_table.ibd
Query OK, 1718272 rows affected (46.59 sec)
Records: 1718272  Duplicates: 0  Warnings: 0
```

例 14.3 CREATE INDEX および DROP INDEX 中の並列 DML

CREATE INDEX および DROP INDEX と同時に実行されている DML ステートメント (挿入、更新、または削除) を示すために、次に、同じデータベースに接続された個別の mysql セッションで実行したいくつかのコードスニペットを示します。

```
/*
CREATE INDEX statement to run against a table while
insert/update/delete statements are modifying the
column being indexed.
*/

-- We'll run this script in one session, while simultaneously creating and dropping
-- an index on test/big_table.table_name in another session.

use test;
create index i_concurrent on big_table(table_name);
```

```
/*
DROP INDEX statement to run against a table while
insert/update/delete statements are modifying the
column being indexed.
*/

-- We'll run this script in one session, while simultaneously creating and dropping
-- an index on test/big_table.table_name in another session.

use test;
drop index i_concurrent on big_table;
```

```
/*
Some queries and insert/update/delete statements to run against a table
while an index is being created or dropped. Previously, these operations
would have stalled during the index create/drop period and possibly
timed out or deadlocked.
*/

-- We'll run this script in one session, while simultaneously creating and dropping
-- an index on test/big_table.table_name in another session.

-- In our test instance, that column has about 1.7M rows, with 136 different values.
-- Sample values: COLUMNS (20480), ENGINES (6144), EVENTS (24576), FILES (38912), TABLES (21504), VIEWS (10240).

set autocommit = 0;
use test;

select distinct character_set_name from big_table where table_name = 'FILES';
delete from big_table where table_name = 'FILES';
select distinct character_set_name from big_table where table_name = 'FILES';

-- I'll issue the final rollback interactively, not via script,
-- the better to control the timing.
-- rollback;
```

このコードを実行すると、簡略化のために圧縮され、もっとも重要な点が太字で示された次の出力が得られます。

```
mysql: source concurrent_ddl_create.sql
Database changed
Query OK, 0 rows affected (1 min 25.15 sec)
Records: 0  Duplicates: 0  Warnings: 0

mysql: source concurrent_ddl_drop.sql
Database changed
Query OK, 0 rows affected (24.98 sec)
Records: 0  Duplicates: 0  Warnings: 0

mysql: source concurrent_dml.sql
Query OK, 0 rows affected (0.00 sec)

Database changed
+--------------------+
```

```
| character_set_name |
+--------------------+
| NULL               |
| utf8               |
+--------------------+
2 rows in set (0.32 sec)

Query OK, 38912 rows affected (1.84 sec)

Empty set (0.01 sec)

mysql: rollback;
Query OK, 0 rows affected (1.05 sec)
```

例 14.4 カラムの名前変更

ALTER TABLE を使用したカラムの名前変更のデモを次に示します。新しい、高速な DDL メカニズムを使用して名前を変更してから、古い、低速な DDL メカニズムを (old_alter_table=1 とともに) 使用して元のカラム名をリストアします。

注:

- カラムの名前変更の構文にはデータ型の再指定も含まれるため、コストの高いテーブル再構築を回避するために、まったく同じデータ型を指定するよう十分に注意してください。この場合は、show create table table\G の出力をチェックし、元のカラム定義から CHARACTER SET や NOT NULL などの句をすべてコピーしました。
- この場合も、小さなテーブルのカラムの名前変更は十分高速であるため、新しい DDL メカニズムが古いメカニズムより効率的であることを確認するには 「rows affected」 の数値を検査する必要があります。大きなテーブルでは、経過時間の違いによって速度の向上が明らかになります。

```
/*
Run through a sequence of 'rename column' statements.
Because this operation involves only metadata, not table data,
it is fast for big and small tables, with new or old DDL mechanisms.
*/

\! clear

\! echo "Rename column (fast technique, small table):"
alter table small_table change `IS_NULLABLE` `NULLABLE` varchar(3) character set utf8 not null, algorithm=inplace;
\! echo "Rename back to original name (slow technique):"
alter table small_table change `NULLABLE` `IS_NULLABLE` varchar(3) character set utf8 not null, algorithm=copy;


\! echo "Rename column (fast technique, big table):"
alter table big_table change `IS_NULLABLE` `NULLABLE` varchar(3) character set utf8 not null, algorithm=inplace;
\! echo "Rename back to original name (slow technique):"
alter table big_table change `NULLABLE` `IS_NULLABLE` varchar(3) character set utf8 not null, algorithm=copy;
```

このコードを実行すると、簡略化のために圧縮され、もっとも重要な点が太字で示された次の出力が得られます。

```
Rename column (fast technique, small table):
Query OK, 0 rows affected (0.05 sec)

Query OK, 0 rows affected (0.13 sec)
Records: 0  Duplicates: 0  Warnings: 0

Rename back to original name (slow technique):
Query OK, 0 rows affected (0.00 sec)

Query OK, 1678 rows affected (0.35 sec)
Records: 1678  Duplicates: 0  Warnings: 0

Rename column (fast technique, big table):
Query OK, 0 rows affected (0.00 sec)

Query OK, 0 rows affected (0.11 sec)
Records: 0  Duplicates: 0  Warnings: 0

Rename back to original name (slow technique):
Query OK, 0 rows affected (0.00 sec)

Query OK, 1718272 rows affected (1 min 0.00 sec)
Records: 1718272  Duplicates: 0  Warnings: 0

Query OK, 0 rows affected (0.00 sec)
```

例 14.5 外部キーの削除

外部キーのデモ (外部キー制約の削除の速度の向上を含む) を次に示します。

```
/*
Demonstrate aspects of foreign keys that are or aren't affected by the DDL improvements.
- Create a new table with only a few values to serve as the parent table.
- Set up the 'small' and 'big' tables as child tables using a foreign key.
- Verify that the ON DELETE CASCADE clause makes changes ripple from parent to child tables.
- Drop the foreign key constraints, and optionally associated indexes. (This is the operation that is sped up.)
*/

\! clear

-- Make sure foreign keys are being enforced, and allow
-- rollback after doing some DELETEs that affect both
-- parent and child tables.
set foreign_key_checks = 1;
set autocommit = 0;

-- Create a parent table, containing values that we know are already present
-- in the child tables.
drop table if exists schema_names;
create table schema_names (id int unsigned not null primary key auto_increment, schema_name varchar(64) character set utf8 not null, index i_schema (schema

show create table schema_names\G
show create table small_table\G
show create table big_table\G

-- Creating the foreign key constraint still involves a table rebuild when foreign_key_checks=1,
-- as illustrated by the "rows affected" figure.
alter table small_table add constraint small_fk foreign key i_table_schema (table_schema) references schema_names(schema_name) on delete cascade;
alter table big_table add constraint big_fk foreign key i_table_schema (table_schema) references schema_names(schema_name) on delete cascade;

show create table small_table\G
show create table big_table\G

select schema_name from schema_names order by schema_name;
select count(table_schema) howmany, table_schema from small_table group by table_schema;
select count(table_schema) howmany, table_schema from big_table group by table_schema;

-- big_table is the parent table.
-- schema_names is the parent table.
-- big_table is the child table.
-- (One row in the parent table can have many "children" in the child table.)
-- Changes to the parent table can ripple through to the child table.
-- For example, removing the value 'test' from schema_names.schema_name will
-- result in the removal of 20K or so rows from big_table.

delete from schema_names where schema_name = 'test';

select schema_name from schema_names order by schema_name;
select count(table_schema) howmany, table_schema from small_table group by table_schema;
select count(table_schema) howmany, table_schema from big_table group by table_schema;

-- Because we've turned off autocommit, we can still get back those deleted rows
-- if the DELETE was issued by mistake.
rollback;

select schema_name from schema_names order by schema_name;
select count(table_schema) howmany, table_schema from small_table group by table_schema;
select count(table_schema) howmany, table_schema from big_table group by table_schema;

-- All of the cross-checking between parent and child tables would be
-- deadly slow if there wasn't the requirement for the corresponding
-- columns to be indexed!

-- But we can get rid of the foreign key using a fast operation
-- that doesn't rebuild the table.
-- If we didn't specify a constraint name when setting up the foreign key, we would
-- have to find the auto-generated name such as 'big_table_ibfk_1' in the
-- output from 'show create table'.

-- For the small table, we'll drop the foreign key and the associated index.
-- Having an index on a small table is less critical.

\! echo "DROP FOREIGN KEY and INDEX from small_table:"
alter table small_table drop foreign key small_fk, drop index small_fk;

-- For the big table, we'll drop the foreign key and leave the associated index.
```

```
-- If we are still doing queries that reference the indexed column, the index is
-- very important to avoid a full table scan of the big table.
\! echo "DROP FOREIGN KEY from big_table:"
alter table big_table drop foreign key big_fk;


show create table small_table\G
show create table big_table\G
```

このコードを実行すると、簡略化のために圧縮され、もっとも重要な点が太字で示された次の出力が得られます。

```
Query OK, 0 rows affected (0.00 sec)

Query OK, 0 rows affected (0.00 sec)

Query OK, 0 rows affected (0.01 sec)

Query OK, 4 rows affected (0.03 sec)
Records: 4  Duplicates: 0  Warnings: 0

*************************** 1. row ***************************
       Table: schema_names
Create Table: CREATE TABLE `schema_names` (
  `id` int(10) unsigned NOT NULL AUTO_INCREMENT,
  `schema_name` varchar(64) CHARACTER SET utf8 NOT NULL,
  PRIMARY KEY (`id`),
  KEY `i_schema` (`schema_name`)
) ENGINE=InnoDB AUTO_INCREMENT=8 DEFAULT CHARSET=latin1
1 row in set (0.00 sec)

*************************** 1. row ***************************
       Table: small_table
Create Table: CREATE TABLE `small_table` (
  `TABLE_CATALOG` varchar(512) CHARACTER SET utf8 NOT NULL DEFAULT '',
  `TABLE_SCHEMA` varchar(64) CHARACTER SET utf8 NOT NULL DEFAULT '',
  `TABLE_NAME` varchar(64) CHARACTER SET utf8 NOT NULL DEFAULT '',
  `COLUMN_NAME` varchar(64) CHARACTER SET utf8 NOT NULL DEFAULT '',
  `ORDINAL_POSITION` bigint(21) unsigned NOT NULL DEFAULT '0',
  `COLUMN_DEFAULT` longtext CHARACTER SET utf8,
  `IS_NULLABLE` varchar(3) CHARACTER SET utf8 NOT NULL,
  `DATA_TYPE` varchar(64) CHARACTER SET utf8 NOT NULL DEFAULT '',
  `CHARACTER_MAXIMUM_LENGTH` bigint(21) unsigned DEFAULT NULL,
  `CHARACTER_OCTET_LENGTH` bigint(21) unsigned DEFAULT NULL,
  `NUMERIC_PRECISION` bigint(21) unsigned DEFAULT NULL,
  `NUMERIC_SCALE` bigint(21) unsigned DEFAULT NULL,
  `DATETIME_PRECISION` bigint(21) unsigned DEFAULT NULL,
  `CHARACTER_SET_NAME` varchar(32) CHARACTER SET utf8 DEFAULT NULL,
  `COLLATION_NAME` varchar(32) CHARACTER SET utf8 DEFAULT NULL,
  `COLUMN_TYPE` longtext CHARACTER SET utf8 NOT NULL,
  `COLUMN_KEY` varchar(3) CHARACTER SET utf8 NOT NULL DEFAULT '',
  `EXTRA` varchar(30) CHARACTER SET utf8 NOT NULL DEFAULT '',
  `PRIVILEGES` varchar(80) CHARACTER SET utf8 NOT NULL DEFAULT '',
  `COLUMN_COMMENT` varchar(1024) CHARACTER SET utf8 NOT NULL DEFAULT '',
  `id` int(10) unsigned NOT NULL AUTO_INCREMENT,
  PRIMARY KEY (`id`)
) ENGINE=InnoDB AUTO_INCREMENT=1679 DEFAULT CHARSET=latin1
1 row in set (0.00 sec)

*************************** 1. row ***************************
       Table: big_table
Create Table: CREATE TABLE `big_table` (
  `TABLE_CATALOG` varchar(512) CHARACTER SET utf8 NOT NULL DEFAULT '',
  `TABLE_SCHEMA` varchar(64) CHARACTER SET utf8 NOT NULL DEFAULT '',
  `TABLE_NAME` varchar(64) CHARACTER SET utf8 NOT NULL DEFAULT '',
  `COLUMN_NAME` varchar(64) CHARACTER SET utf8 NOT NULL DEFAULT '',
  `ORDINAL_POSITION` bigint(21) unsigned NOT NULL DEFAULT '0',
  `COLUMN_DEFAULT` longtext CHARACTER SET utf8,
  `IS_NULLABLE` varchar(3) CHARACTER SET utf8 NOT NULL,
  `DATA_TYPE` varchar(64) CHARACTER SET utf8 NOT NULL DEFAULT '',
  `CHARACTER_MAXIMUM_LENGTH` bigint(21) unsigned DEFAULT NULL,
  `CHARACTER_OCTET_LENGTH` bigint(21) unsigned DEFAULT NULL,
  `NUMERIC_PRECISION` bigint(21) unsigned DEFAULT NULL,
  `NUMERIC_SCALE` bigint(21) unsigned DEFAULT NULL,
  `DATETIME_PRECISION` bigint(21) unsigned DEFAULT NULL,
  `CHARACTER_SET_NAME` varchar(32) CHARACTER SET utf8 DEFAULT NULL,
  `COLLATION_NAME` varchar(32) CHARACTER SET utf8 DEFAULT NULL,
  `COLUMN_TYPE` longtext CHARACTER SET utf8 NOT NULL,
  `COLUMN_KEY` varchar(3) CHARACTER SET utf8 NOT NULL DEFAULT '',
  `EXTRA` varchar(30) CHARACTER SET utf8 NOT NULL DEFAULT '',
```

```
  `PRIVILEGES` varchar(80) CHARACTER SET utf8 NOT NULL DEFAULT '',
  `COLUMN_COMMENT` varchar(1024) CHARACTER SET utf8 NOT NULL DEFAULT '',
  `id` int(10) unsigned NOT NULL AUTO_INCREMENT,
  PRIMARY KEY (`id`),
  KEY `big_fk` (`TABLE_SCHEMA`)
) ENGINE=InnoDB AUTO_INCREMENT=1718273 DEFAULT CHARSET=latin1
1 row in set (0.00 sec)

Query OK, 1678 rows affected (0.10 sec)
Records: 1678  Duplicates: 0  Warnings: 0

Query OK, 1718272 rows affected (1 min 14.54 sec)
Records: 1718272  Duplicates: 0  Warnings: 0

*************************** 1. row ***************************
       Table: small_table
Create Table: CREATE TABLE `small_table` (
  `TABLE_CATALOG` varchar(512) CHARACTER SET utf8 NOT NULL DEFAULT '',
  `TABLE_SCHEMA` varchar(64) CHARACTER SET utf8 NOT NULL DEFAULT '',
  `TABLE_NAME` varchar(64) CHARACTER SET utf8 NOT NULL DEFAULT '',
  `COLUMN_NAME` varchar(64) CHARACTER SET utf8 NOT NULL DEFAULT '',
  `ORDINAL_POSITION` bigint(21) unsigned NOT NULL DEFAULT '0',
  `COLUMN_DEFAULT` longtext CHARACTER SET utf8,
  `IS_NULLABLE` varchar(3) CHARACTER SET utf8 NOT NULL,
  `DATA_TYPE` varchar(64) CHARACTER SET utf8 NOT NULL DEFAULT '',
  `CHARACTER_MAXIMUM_LENGTH` bigint(21) unsigned DEFAULT NULL,
  `CHARACTER_OCTET_LENGTH` bigint(21) unsigned DEFAULT NULL,
  `NUMERIC_PRECISION` bigint(21) unsigned DEFAULT NULL,
  `NUMERIC_SCALE` bigint(21) unsigned DEFAULT NULL,
  `DATETIME_PRECISION` bigint(21) unsigned DEFAULT NULL,
  `CHARACTER_SET_NAME` varchar(32) CHARACTER SET utf8 DEFAULT NULL,
  `COLLATION_NAME` varchar(32) CHARACTER SET utf8 DEFAULT NULL,
  `COLUMN_TYPE` longtext CHARACTER SET utf8 NOT NULL,
  `COLUMN_KEY` varchar(3) CHARACTER SET utf8 NOT NULL DEFAULT '',
  `EXTRA` varchar(30) CHARACTER SET utf8 NOT NULL DEFAULT '',
  `PRIVILEGES` varchar(80) CHARACTER SET utf8 NOT NULL DEFAULT '',
  `COLUMN_COMMENT` varchar(1024) CHARACTER SET utf8 NOT NULL DEFAULT '',
  `id` int(10) unsigned NOT NULL AUTO_INCREMENT,
  PRIMARY KEY (`id`),
  KEY `small_fk` (`TABLE_SCHEMA`),
  CONSTRAINT `small_fk` FOREIGN KEY (`TABLE_SCHEMA`) REFERENCES `schema_names` (`schema_name`) ON DELETE CASCADE
) ENGINE=InnoDB AUTO_INCREMENT=1679 DEFAULT CHARSET=latin1
1 row in set (0.12 sec)

*************************** 1. row ***************************
       Table: big_table
Create Table: CREATE TABLE `big_table` (
  `TABLE_CATALOG` varchar(512) CHARACTER SET utf8 NOT NULL DEFAULT '',
  `TABLE_SCHEMA` varchar(64) CHARACTER SET utf8 NOT NULL DEFAULT '',
  `TABLE_NAME` varchar(64) CHARACTER SET utf8 NOT NULL DEFAULT '',
  `COLUMN_NAME` varchar(64) CHARACTER SET utf8 NOT NULL DEFAULT '',
  `ORDINAL_POSITION` bigint(21) unsigned NOT NULL DEFAULT '0',
  `COLUMN_DEFAULT` longtext CHARACTER SET utf8,
  `IS_NULLABLE` varchar(3) CHARACTER SET utf8 NOT NULL,
  `DATA_TYPE` varchar(64) CHARACTER SET utf8 NOT NULL DEFAULT '',
  `CHARACTER_MAXIMUM_LENGTH` bigint(21) unsigned DEFAULT NULL,
  `CHARACTER_OCTET_LENGTH` bigint(21) unsigned DEFAULT NULL,
  `NUMERIC_PRECISION` bigint(21) unsigned DEFAULT NULL,
  `NUMERIC_SCALE` bigint(21) unsigned DEFAULT NULL,
  `DATETIME_PRECISION` bigint(21) unsigned DEFAULT NULL,
  `CHARACTER_SET_NAME` varchar(32) CHARACTER SET utf8 DEFAULT NULL,
  `COLLATION_NAME` varchar(32) CHARACTER SET utf8 DEFAULT NULL,
  `COLUMN_TYPE` longtext CHARACTER SET utf8 NOT NULL,
  `COLUMN_KEY` varchar(3) CHARACTER SET utf8 NOT NULL DEFAULT '',
  `EXTRA` varchar(30) CHARACTER SET utf8 NOT NULL DEFAULT '',
  `PRIVILEGES` varchar(80) CHARACTER SET utf8 NOT NULL DEFAULT '',
  `COLUMN_COMMENT` varchar(1024) CHARACTER SET utf8 NOT NULL DEFAULT '',
  `id` int(10) unsigned NOT NULL AUTO_INCREMENT,
  PRIMARY KEY (`id`),
  KEY `big_fk` (`TABLE_SCHEMA`),
  CONSTRAINT `big_fk` FOREIGN KEY (`TABLE_SCHEMA`) REFERENCES `schema_names` (`schema_name`) ON DELETE CASCADE
) ENGINE=InnoDB AUTO_INCREMENT=1718273 DEFAULT CHARSET=latin1
1 row in set (0.01 sec)

+--------------------+
| schema_name        |
+--------------------+
| information_schema |
| mysql              |
| performance_schema |
```

```
| test               |
+--------------------+
4 rows in set (0.00 sec)

+---------+--------------------+
| howmany | table_schema       |
+---------+--------------------+
|     563 | information_schema |
|     286 | mysql              |
|     786 | performance_schema |
|      43 | test               |
+---------+--------------------+
4 rows in set (0.01 sec)

+---------+--------------------+
| howmany | table_schema       |
+---------+--------------------+
|  576512 | information_schema |
|  292864 | mysql              |
|  804864 | performance_schema |
|   44032 | test               |
+---------+--------------------+
4 rows in set (2.10 sec)

Query OK, 1 row affected (1.52 sec)

+--------------------+
| schema_name        |
+--------------------+
| information_schema |
| mysql              |
| performance_schema |
+--------------------+
3 rows in set (0.00 sec)

+---------+--------------------+
| howmany | table_schema       |
+---------+--------------------+
|     563 | information_schema |
|     286 | mysql              |
|     786 | performance_schema |
+---------+--------------------+
3 rows in set (0.00 sec)

+---------+--------------------+
| howmany | table_schema       |
+---------+--------------------+
|  576512 | information_schema |
|  292864 | mysql              |
|  804864 | performance_schema |
+---------+--------------------+
3 rows in set (1.74 sec)

Query OK, 0 rows affected (0.60 sec)

+--------------------+
| schema_name        |
+--------------------+
| information_schema |
| mysql              |
| performance_schema |
| test               |
+--------------------+
4 rows in set (0.00 sec)

+---------+--------------------+
| howmany | table_schema       |
+---------+--------------------+
|     563 | information_schema |
|     286 | mysql              |
|     786 | performance_schema |
|      43 | test               |
+---------+--------------------+
4 rows in set (0.01 sec)

+---------+--------------------+
| howmany | table_schema       |
+---------+--------------------+
|  576512 | information_schema |
|  292864 | mysql              |
```

```
|  804864 | performance_schema |
|   44032 | test               |
+---------+--------------------+
4 rows in set (1.59 sec)

DROP FOREIGN KEY and INDEX from small_table:
Query OK, 0 rows affected (0.02 sec)
Records: 0  Duplicates: 0  Warnings: 0

DROP FOREIGN KEY from big_table:
Query OK, 0 rows affected (0.02 sec)
Records: 0  Duplicates: 0  Warnings: 0

*************************** 1. row ***************************
       Table: small_table
Create Table: CREATE TABLE `small_table` (
  `TABLE_CATALOG` varchar(512) CHARACTER SET utf8 NOT NULL DEFAULT '',
  `TABLE_SCHEMA` varchar(64) CHARACTER SET utf8 NOT NULL DEFAULT '',
  `TABLE_NAME` varchar(64) CHARACTER SET utf8 NOT NULL DEFAULT '',
  `COLUMN_NAME` varchar(64) CHARACTER SET utf8 NOT NULL DEFAULT '',
  `ORDINAL_POSITION` bigint(21) unsigned NOT NULL DEFAULT '0',
  `COLUMN_DEFAULT` longtext CHARACTER SET utf8,
  `IS_NULLABLE` varchar(3) CHARACTER SET utf8 NOT NULL,
  `DATA_TYPE` varchar(64) CHARACTER SET utf8 NOT NULL DEFAULT '',
  `CHARACTER_MAXIMUM_LENGTH` bigint(21) unsigned DEFAULT NULL,
  `CHARACTER_OCTET_LENGTH` bigint(21) unsigned DEFAULT NULL,
  `NUMERIC_PRECISION` bigint(21) unsigned DEFAULT NULL,
  `NUMERIC_SCALE` bigint(21) unsigned DEFAULT NULL,
  `DATETIME_PRECISION` bigint(21) unsigned DEFAULT NULL,
  `CHARACTER_SET_NAME` varchar(32) CHARACTER SET utf8 DEFAULT NULL,
  `COLLATION_NAME` varchar(32) CHARACTER SET utf8 DEFAULT NULL,
  `COLUMN_TYPE` longtext CHARACTER SET utf8 NOT NULL,
  `COLUMN_KEY` varchar(3) CHARACTER SET utf8 NOT NULL DEFAULT '',
  `EXTRA` varchar(30) CHARACTER SET utf8 NOT NULL DEFAULT '',
  `PRIVILEGES` varchar(80) CHARACTER SET utf8 NOT NULL DEFAULT '',
  `COLUMN_COMMENT` varchar(1024) CHARACTER SET utf8 NOT NULL DEFAULT '',
  `id` int(10) unsigned NOT NULL AUTO_INCREMENT,
  PRIMARY KEY (`id`)
) ENGINE=InnoDB AUTO_INCREMENT=1679 DEFAULT CHARSET=latin1
1 row in set (0.00 sec)

*************************** 1. row ***************************
       Table: big_table
Create Table: CREATE TABLE `big_table` (
  `TABLE_CATALOG` varchar(512) CHARACTER SET utf8 NOT NULL DEFAULT '',
  `TABLE_SCHEMA` varchar(64) CHARACTER SET utf8 NOT NULL DEFAULT '',
  `TABLE_NAME` varchar(64) CHARACTER SET utf8 NOT NULL DEFAULT '',
  `COLUMN_NAME` varchar(64) CHARACTER SET utf8 NOT NULL DEFAULT '',
  `ORDINAL_POSITION` bigint(21) unsigned NOT NULL DEFAULT '0',
  `COLUMN_DEFAULT` longtext CHARACTER SET utf8,
  `IS_NULLABLE` varchar(3) CHARACTER SET utf8 NOT NULL,
  `DATA_TYPE` varchar(64) CHARACTER SET utf8 NOT NULL DEFAULT '',
  `CHARACTER_MAXIMUM_LENGTH` bigint(21) unsigned DEFAULT NULL,
  `CHARACTER_OCTET_LENGTH` bigint(21) unsigned DEFAULT NULL,
  `NUMERIC_PRECISION` bigint(21) unsigned DEFAULT NULL,
  `NUMERIC_SCALE` bigint(21) unsigned DEFAULT NULL,
  `DATETIME_PRECISION` bigint(21) unsigned DEFAULT NULL,
  `CHARACTER_SET_NAME` varchar(32) CHARACTER SET utf8 DEFAULT NULL,
  `COLLATION_NAME` varchar(32) CHARACTER SET utf8 DEFAULT NULL,
  `COLUMN_TYPE` longtext CHARACTER SET utf8 NOT NULL,
  `COLUMN_KEY` varchar(3) CHARACTER SET utf8 NOT NULL DEFAULT '',
  `EXTRA` varchar(30) CHARACTER SET utf8 NOT NULL DEFAULT '',
  `PRIVILEGES` varchar(80) CHARACTER SET utf8 NOT NULL DEFAULT '',
  `COLUMN_COMMENT` varchar(1024) CHARACTER SET utf8 NOT NULL DEFAULT '',
  `id` int(10) unsigned NOT NULL AUTO_INCREMENT,
  PRIMARY KEY (`id`),
  KEY `big_fk` (`TABLE_SCHEMA`)
) ENGINE=InnoDB AUTO_INCREMENT=1718273 DEFAULT CHARSET=latin1
1 row in set (0.00 sec)
```

例 14.6 自動インクリメント値の変更

テーブルカラムの自動インクリメントの下限値を増やすコードを次に示します。これは、この操作によってテーブル再構築がどのように回避されているか、さらには InnoDB の自動インクリメントカラムに関する興味深いその他のいくつかの事実を示しています。

```
/*
If this script is run after foreign_key.sql, the schema_names table is
already set up. But to allow this script to run multiple times without
running into duplicate ID errors, we set up the schema_names table
```

```
all over again.
*/

\! clear

\! echo "=== Adjusting the Auto-Increment Limit for a Table ==="
\! echo

drop table if exists schema_names;
create table schema_names (id int unsigned not null primary key auto_increment,
  schema_name varchar(64) character set utf8 not null, index i_schema (schema_name))
  as select distinct table_schema schema_name from small_table;

\! echo "Initial state of schema_names table. AUTO_INCREMENT is included in SHOW CREATE TABLE output."
\! echo "Note how MySQL reserved a block of IDs, but only needed 4 of them in this transaction, so the next inserted values would get IDs 8 and 9."
show create table schema_names\G
select * from schema_names order by id;

\! echo "Inserting even a tiny amount of data can produce gaps in the ID sequence."
insert into schema_names (schema_name) values ('eight'), ('nine');

\! echo "Bumping auto-increment lower limit to 20 (fast mechanism):"
alter table schema_names auto_increment=20, algorithm=inplace;

\! echo "Inserting 2 rows that should get IDs 20 and 21:"
insert into schema_names (schema_name) values ('foo'), ('bar');
commit;

\! echo "Bumping auto-increment lower limit to 30 (slow mechanism):"
alter table schema_names auto_increment=30, algorithm=copy;

\! echo "Inserting 2 rows that should get IDs 30 and 31:"
insert into schema_names (schema_name) values ('bletch'),('baz');
commit;

select * from schema_names order by id;

\! echo "Final state of schema_names table. AUTO_INCREMENT value shows the next inserted row would get ID=32."
show create table schema_names\G
```

このコードを実行すると、簡略化のために圧縮され、もっとも重要な点が太字で示された次の出力が得られます。

```
=== Adjusting the Auto-Increment Limit for a Table ===

Query OK, 0 rows affected (0.01 sec)

Query OK, 4 rows affected (0.02 sec)
Records: 4  Duplicates: 0  Warnings: 0

Initial state of schema_names table. AUTO_INCREMENT is included in SHOW CREATE TABLE output.
Note how MySQL reserved a block of IDs, but only needed 4 of them in this transaction, so the next inserted values would get IDs 8 and 9.
*************************** 1. row ***************************
       Table: schema_names
Create Table: CREATE TABLE `schema_names` (
  `id` int(10) unsigned NOT NULL AUTO_INCREMENT,
  `schema_name` varchar(64) CHARACTER SET utf8 NOT NULL,
  PRIMARY KEY (`id`),
  KEY `i_schema` (`schema_name`)
) ENGINE=InnoDB AUTO_INCREMENT=8 DEFAULT CHARSET=latin1
1 row in set (0.00 sec)

+----+--------------------+
| id | schema_name        |
+----+--------------------+
|  1 | information_schema |
|  2 | mysql              |
|  3 | performance_schema |
|  4 | test               |
+----+--------------------+
4 rows in set (0.00 sec)

Inserting even a tiny amount of data can produce gaps in the ID sequence.
Query OK, 2 rows affected (0.00 sec)
Records: 2  Duplicates: 0  Warnings: 0

Query OK, 0 rows affected (0.00 sec)

Bumping auto-increment lower limit to 20 (fast mechanism):
Query OK, 0 rows affected (0.01 sec)
```

```
Records: 0  Duplicates: 0  Warnings: 0

Inserting 2 rows that should get IDs 20 and 21:
Query OK, 2 rows affected (0.00 sec)
Records: 2  Duplicates: 0  Warnings: 0

Query OK, 0 rows affected (0.00 sec)

Query OK, 0 rows affected (0.00 sec)

Bumping auto-increment lower limit to 30 (slow mechanism):
Query OK, 8 rows affected (0.02 sec)
Records: 8  Duplicates: 0  Warnings: 0

Inserting 2 rows that should get IDs 30 and 31:
Query OK, 2 rows affected (0.00 sec)
Records: 2  Duplicates: 0  Warnings: 0

Query OK, 0 rows affected (0.01 sec)

+----+--------------------+
| id | schema_name        |
+----+--------------------+
|  1 | information_schema |
|  2 | mysql              |
|  3 | performance_schema |
|  4 | test               |
|  8 | eight              |
|  9 | nine               |
| 20 | foo                |
| 21 | bar                |
| 30 | bletch             |
| 31 | baz                |
+----+--------------------+
10 rows in set (0.00 sec)

Query OK, 0 rows affected (0.00 sec)

Final state of schema_names table. AUTO_INCREMENT value shows the next inserted row would get ID=32.
*************************** 1. row ***************************
       Table: schema_names
Create Table: CREATE TABLE `schema_names` (
  `id` int(10) unsigned NOT NULL AUTO_INCREMENT,
  `schema_name` varchar(64) CHARACTER SET utf8 NOT NULL,
  PRIMARY KEY (`id`),
  KEY `i_schema` (`schema_name`)
) ENGINE=InnoDB AUTO_INCREMENT=32 DEFAULT CHARSET=latin1
1 row in set (0.00 sec)
```

例 14.7 LOCK 句を使用した並列性の制御

この例は、ALTER TABLE ステートメントの LOCK 句を使用して、オンライン DDL 操作の進行中にテーブルへの並列アクセスを許可または拒否する方法を示しています。この句には、クエリーと DML ステートメントを許可するか (LOCK=NONE)、クエリーのみを許可するか (LOCK=SHARED)、または並列アクセスをまったく許可しない (LOCK=EXCLUSIVE) 設定があります。

ここでは、いずれかのセッションの待機中またはデッドロック中の動作を確認するために、LOCK 句の異なる値を使用して、1 つのセッションで連続した ALTER TABLE ステートメントを実行してインデックスを作成および削除します。前の例と同じ BIG_TABLE テーブルを使用し、約 170 万行から始めます。説明のために、IS_NULLABLE カラムに対してインデックス作成とクエリーを実行します。(ただし、実際には、固有の値が 2 つしかない非常に小さなカラムのインデックスを作成することはありえません。)

```
mysql: desc big_table;
+-------------------------+----------------------+------+-----+---------+----------------+
| Field                   | Type                 | Null | Key | Default | Extra          |
+-------------------------+----------------------+------+-----+---------+----------------+
| TABLE_CATALOG           | varchar(512)         | NO   |     |         |                |
| TABLE_SCHEMA            | varchar(64)          | NO   |     |         |                |
| TABLE_NAME              | varchar(64)          | NO   |     |         |                |
| COLUMN_NAME             | varchar(64)          | NO   |     |         |                |
| ORDINAL_POSITION        | bigint(21) unsigned | NO   |     | 0       |                |
| COLUMN_DEFAULT          | longtext             | YES  |     | NULL    |                |

| IS_NULLABLE             | varchar(3)           | NO   |     |         |                |
...
+-------------------------+----------------------+------+-----+---------+----------------+
21 rows in set (0.14 sec)

mysql: alter table big_table add index i1(is_nullable);
```

```
Query OK, 0 rows affected (20.71 sec)

mysql: alter table big_table drop index i1;
Query OK, 0 rows affected (0.02 sec)

mysql: alter table big_table add index i1(is_nullable), lock=exclusive;
Query OK, 0 rows affected (19.44 sec)

mysql: alter table big_table drop index i1;
Query OK, 0 rows affected (0.03 sec)

mysql: alter table big_table add index i1(is_nullable), lock=shared;
Query OK, 0 rows affected (16.71 sec)

mysql: alter table big_table drop index i1;
Query OK, 0 rows affected (0.05 sec)

mysql: alter table big_table add index i1(is_nullable), lock=none;
Query OK, 0 rows affected (12.26 sec)

mysql: alter table big_table drop index i1;
Query OK, 0 rows affected (0.01 sec)

... repeat statements like the above while running queries ...
... and DML statements at the same time in another session ...
```

DDL ステートメントを実行しているセッションでは、特別なことは何も発生しません。場合によっては、別のトランザクションが DDL 中にテーブルを変更したか、または DDL の前にテーブルをクエリーしたとき、そのトランザクションの完了を待機しているために ALTER TABLE に非常に長い時間がかかることがあります。

```
mysql: alter table big_table add index i1(is_nullable), lock=none;

Query OK, 0 rows affected (59.27 sec)

mysql: -- The previous ALTER took so long because it was waiting for all the concurrent
mysql: -- transactions to commit or roll back.

mysql: alter table big_table drop index i1;
Query OK, 0 rows affected (41.05 sec)

mysql: -- Even doing a SELECT on the table in the other session first causes
mysql: -- the ALTER TABLE above to stall until the transaction
mysql: -- surrounding the SELECT is committed or rolled back.
```

同時に実行されている別のセッションのログを次に示します。ここでは、前のリストに示されている DDL 操作の前、最中、およびあとにテーブルに対してクエリーと DML ステートメントを発行しています。この最初のリストは、クエリーのみを示しています。LOCK=NONE または LOCK=SHARED を使用して DDL 操作中にクエリーが許可されること、および ALTER TABLE ステートメントに LOCK=EXCLUSIVE が含まれている場合は DDL が完了するまでクエリーが待機することを予測しています。

```
mysql: show variables like 'autocommit';
+---------------+-------+
| Variable_name | Value |
+---------------+-------+
| autocommit    | ON    |
+---------------+-------+
1 row in set (0.01 sec)

mysql: -- A trial query before any ADD INDEX in the other session:
mysql: -- Note: because autocommit is enabled, each
mysql: -- transaction finishes immediately after the query.
mysql: select distinct is_nullable from big_table;
+-------------+
| is_nullable |
+-------------+
| NO          |
| YES         |
+-------------+
2 rows in set (4.49 sec)

mysql: -- Index is being created with LOCK=EXCLUSIVE on the ALTER statement.
mysql: -- The query waits until the DDL is finished before proceeding.
mysql: select distinct is_nullable from big_table;
+-------------+
| is_nullable |
+-------------+
| NO          |
| YES         |
+-------------+
```

```
2 rows in set (17.26 sec)

mysql: -- Index is being created with LOCK=SHARED on the ALTER statement.
mysql: -- The query returns its results while the DDL is in progress.
mysql: -- The same thing happens with LOCK=NONE on the ALTER statement.
mysql: select distinct is_nullable from big_table;
+-------------+
| is_nullable |
+-------------+
| NO          |
| YES         |
+-------------+
2 rows in set (3.11 sec)

mysql: -- Once the index is created, and with no DDL in progress,
mysql: -- queries referencing the indexed column are very fast:
mysql: select count(*) from big_table where is_nullable = 'YES';
+----------+
| count(*) |
+----------+
|   411648 |
+----------+
1 row in set (0.20 sec)

mysql: select distinct is_nullable from big_table;
+-------------+
| is_nullable |
+-------------+
| NO          |
| YES         |
+-------------+
2 rows in set (0.00 sec)
```

次に、この並列セッションで、DML ステートメントまたは DML ステートメントとクエリーの組み合わせを含むいくつかのトランザクションを実行します。テーブルへの予測可能かつ検証可能な変更を示すために、DELETE ステートメントを使用します。この部分にあるトランザクションは複数のステートメントにまたがる場合があるため、これらのテストは autocommit がオフになった状態で実行します。

```
mysql: set global autocommit = off;
Query OK, 0 rows affected (0.00 sec)

mysql: -- Count the rows that will be involved in our DELETE statements:
mysql: select count(*) from big_table where is_nullable = 'YES';
+----------+
| count(*) |
+----------+
|   411648 |
+----------+
1 row in set (0.95 sec)

mysql: -- After this point, any DDL statements back in the other session
mysql: -- stall until we commit or roll back.

mysql: delete from big_table where is_nullable = 'YES' limit 11648;
Query OK, 11648 rows affected (0.14 sec)

mysql: select count(*) from big_table where is_nullable = 'YES';
+----------+
| count(*) |
+----------+
|   400000 |
+----------+
1 row in set (1.04 sec)

mysql: rollback;
Query OK, 0 rows affected (0.09 sec)

mysql: select count(*) from big_table where is_nullable = 'YES';
+----------+
| count(*) |
+----------+
|   411648 |
+----------+
1 row in set (0.93 sec)

mysql: -- OK, now we're going to try that during index creation with LOCK=NONE.
mysql: delete from big_table where is_nullable = 'YES' limit 11648;
Query OK, 11648 rows affected (0.21 sec)
```

```
mysql: -- We expect that now there will be 400000 'YES' rows left:
mysql: select count(*) from big_table where is_nullable = 'YES';
+----------+
| count(*) |
+----------+
|   400000 |
+----------+
1 row in set (1.25 sec)

mysql: -- In the other session, the ALTER TABLE is waiting before finishing,
mysql: -- because _this_ transaction hasn't committed or rolled back yet.
mysql: rollback;
Query OK, 0 rows affected (0.11 sec)

mysql: select count(*) from big_table where is_nullable = 'YES';
+----------+
| count(*) |
+----------+
|   411648 |
+----------+
1 row in set (0.19 sec)

mysql: -- The ROLLBACK left the table in the same state we originally found it.
mysql: -- Now let's make a permanent change while the index is being created,
mysql: -- again with ALTER TABLE ... , LOCK=NONE.
mysql: -- First, commit so the DROP INDEX in the other shell can finish;
mysql: -- the previous SELECT started a transaction that accessed the table.
mysql: commit;
Query OK, 0 rows affected (0.00 sec)

mysql: -- Now we add the index back in the other shell, then issue DML in this one
mysql: -- while the DDL is running.
mysql: delete from big_table where is_nullable = 'YES' limit 11648;
Query OK, 11648 rows affected (0.23 sec)

mysql: commit;
Query OK, 0 rows affected (0.01 sec)

mysql: -- In the other shell, the ADD INDEX has finished.
mysql: select count(*) from big_table where is_nullable = 'YES';
+----------+
| count(*) |
+----------+
|   400000 |
+----------+
1 row in set (0.19 sec)

mysql: -- At the point the new index is finished being created, it contains entries
mysql: -- only for the 400000 'YES' rows left when all concurrent transactions are finished.
mysql:
mysql: -- Now we will run a similar test, while ALTER TABLE ... , LOCK=SHARED is running.
mysql: -- We expect a query to complete during the ALTER TABLE, but for the DELETE
mysql: -- to run into some kind of issue.
mysql: commit;
Query OK, 0 rows affected (0.00 sec)

mysql: -- As expected, the query returns results while the LOCK=SHARED DDL is running:
mysql: select count(*) from big_table where is_nullable = 'YES';
+----------+
| count(*) |
+----------+
|   400000 |
+----------+
1 row in set (2.07 sec)

mysql: -- The DDL in the other session is not going to finish until this transaction
mysql: -- is committed or rolled back. If we tried a DELETE now and it waited because
mysql: -- of LOCK=SHARED on the DDL, both transactions would wait forever (deadlock).
mysql: -- MySQL detects this condition and cancels the attempted DML statement.
mysql: delete from big_table where is_nullable = 'YES' limit 100000;
ERROR 1213 (40001): Deadlock found when trying to get lock; try restarting transaction
mysql: -- The transaction here is still going, so in the other shell, the ADD INDEX operation
mysql: -- is waiting for this transaction to commit or roll back.
mysql: rollback;
Query OK, 0 rows affected (0.00 sec)

mysql: -- Now let's try issuing a query and some DML, on one line, while running
mysql: -- ALTER TABLE ... , LOCK=EXCLUSIVE in the other shell.
mysql: -- Notice how even the query is held up until the DDL is finished.
mysql: -- By the time the DELETE is issued, there is no conflicting access
```

```
mysql: -- to the table and we avoid the deadlock error.
mysql: select count(*) from big_table where is_nullable = 'YES'; delete from big_table where is_nullable = 'YES' limit 100000;
+----------+
| count(*) |
+----------+
|   400000 |
+----------+

1 row in set (15.98 sec)

Query OK, 100000 rows affected (2.81 sec)

mysql: select count(*) from big_table where is_nullable = 'YES';
+----------+
| count(*) |
+----------+
|   300000 |
+----------+
1 row in set (0.17 sec)

mysql: rollback;
Query OK, 0 rows affected (1.36 sec)

mysql: select count(*) from big_table where is_nullable = 'YES';
+----------+
| count(*) |
+----------+
|   400000 |
+----------+
1 row in set (0.19 sec)

mysql: commit;
Query OK, 0 rows affected (0.00 sec)

mysql: -- Next, we try ALTER TABLE ... , LOCK=EXCLUSIVE in the other session
mysql: -- and only issue DML, not any query, in the concurrent transaction here.
mysql: delete from big_table where is_nullable = 'YES' limit 100000;
Query OK, 100000 rows affected (16.37 sec)

mysql: -- That was OK because the ALTER TABLE did not have to wait for the transaction
mysql: -- here to complete. The DELETE in this session waited until the index was ready.
mysql: select count(*) from big_table where is_nullable = 'YES';
+----------+
| count(*) |
+----------+
|   300000 |
+----------+
1 row in set (0.16 sec)

mysql: commit;
Query OK, 0 rows affected (0.00 sec)
```

前のリスト例では、次のことがわかりました。

- ALTER TABLE の LOCK 句は、ステートメントの残りの部分からカンマで区切られます。
- オンライン DDL 操作は、テーブルにアクセスする以前のいずれかのトランザクションがコミットまたはロールバックされるまで、開始前に待機する可能性があります。
- オンライン DDL 操作は、テーブルにアクセスするいずれかの並列トランザクションがコミットまたはロールバックされるまで、完了前に待機する可能性があります。
- ALTER TABLE ステートメントが LOCK=NONE または LOCK=SHARED を使用しているかぎり、オンライン DDL 操作が実行されている間の並列クエリーの動作は比較的単純です。
- autocommit がオンとオフのどちらになっているかに注意を払ってください。オフになっている場合は、テーブルで DDL 操作を実行する前にほかのセッションのトランザクション (クエリーだけであっても) を終了するときは注意してください。
- LOCK=SHARED では、クエリーと DML が混在した並列トランザクションでデッドロックエラーが発生する可能性があるため、DDL が完了したあとにこれらのトランザクションを再開する必要があります。
- LOCK=NONE では、並列トランザクションにクエリーと DML を自由に混在させることができます。DDL 操作は、並列トランザクションがコミットまたはロールバックされるまで待機します。
- LOCK=EXCLUSIVE では、並列トランザクションにクエリーと DML を自由に混在させることができますが、これらのトランザクションは、DDL 操作が完了するまで待機したあとでしかテーブルにアクセスできません。

例 14.8 オンライン DDL 実験のためのスキーマ設定コード

1 つの ALTER TABLE ステートメントでテーブル上に複数のインデックスを作成できます。テーブルのクラスタ化されたインデックスは 1 回しかスキャンする必要がない (ただし、データは新しいインデックスごとに個別にソートされます) ため、これはかなり効率的です。例:

```
CREATE TABLE T1(A INT PRIMARY KEY, B INT, C CHAR(1)) ENGINE=InnoDB;
INSERT INTO T1 VALUES (1,2,'a'), (2,3,'b'), (3,2,'c'), (4,3,'d'), (5,2,'e');
COMMIT;
ALTER TABLE T1 ADD INDEX (B), ADD UNIQUE INDEX (C);
```

カラム A に主キーを持つ上のステートメント CREATE TABLE T1 は、いくつかの行を挿入したあと、カラム B と C に 2 つの新しいインデックスを構築します。ALTER TABLE ステートメントの前に T1 に多数の行が挿入されていたとすると、このアプローチは、データをロードする前にすべてのセカンダリインデックスを作成するよりはるかに効率的です。

InnoDB セカンダリインデックスの削除にもテーブルデータのコピーは必要ないため、1 つの ALTER TABLE ステートメントまたは複数の DROP INDEX ステートメントで複数のインデックスを削除することは等しく効率的です。

```
ALTER TABLE T1 DROP INDEX B, DROP INDEX C;
```

または

```
DROP INDEX B ON T1;
DROP INDEX C ON T1;
```

例 14.9 主キーの作成および削除

InnoDB テーブルのクラスタ化されたインデックスの再構成には常に、テーブルデータのコピーが必要です。そのため、テーブルの再構築を回避するために、あとで ALTER TABLE ... ADD PRIMARY KEY を発行するのではなく、テーブルの作成時に主キーを定義することをお勧めします。

次の例のように、あとで PRIMARY KEY を定義した場合はデータがコピーされます。

```
CREATE TABLE T2 (A INT, B INT);
INSERT INTO T2 VALUES (NULL, 1);
ALTER TABLE T2 ADD PRIMARY KEY (B);
```

UNIQUE または PRIMARY KEY インデックスを作成したとき、MySQL は、いくつかの追加の作業を行う必要があります。UNIQUE インデックスの場合、MySQL は、テーブルに重複したキーの値が含まれていないことをチェックします。PRIMARY KEY インデックスの場合も、MySQL は、どの PRIMARY KEY カラムにも NULL が含まれていないことをチェックします。

ALGORITHM=COPY 句を使用して主キーを追加したとき、MySQL は実際には、関連付けられたカラム内の NULL 値をデフォルト値、つまり、数値の場合は 0、文字ベースのカラムや BLOB の場合は空の文字列、および DATETIME の場合は 0000-00-00 00:00:00 に変換します。これは非標準の動作であるため、これに依存しないようにすることをお勧めします。ALGORITHM=INPLACE を使用した主キーの追加は、SQL_MODE 設定に strict_trans_tables または strict_all_tables フラグが含まれている場合にのみ許可されます。SQL_MODE 設定が厳密である場合は、ADD PRIMARY KEY ... , ALGORITHM=INPLACE が許可されますが、要求された主キーカラムに NULL 値が含まれているとステートメントは引き続き失敗します。ALGORITHM=INPLACE の動作は、より標準に準拠しています。

次の例は、ADD PRIMARY KEY 句のいくつかの可能性を示しています。ALGORITHM=COPY 句を使用した場合、主キーカラム内に NULL 値が存在していても操作は成功します。データは暗黙のうちに変更され、それによって問題が発生する可能性があります。

```
mysql> CREATE TABLE add_pk_via_copy (c1 INT, c2 VARCHAR(10), c3 DATETIME);
Query OK, 0 rows affected (0.03 sec)

mysql> INSERT INTO add_pk_via_copy VALUES (1,'a','2014-11-03 11:01:37'),(NULL,NULL,NULL);
Query OK, 2 rows affected (0.00 sec)
Records: 2  Duplicates: 0  Warnings: 0

mysql> SET sql_mode = '';
Query OK, 0 rows affected (0.00 sec)

mysql> ALTER TABLE add_pk_via_copy ADD PRIMARY KEY (c1,c2,c3), ALGORITHM=COPY;
Query OK, 2 rows affected, 3 warnings (0.07 sec)
Records: 2  Duplicates: 0  Warnings: 3

mysql> SHOW WARNINGS;
+---------+------+-----------------------------------------+
| Level   | Code | Message                                 |
```

```
+---------+------+----------------------------------------+
| Warning | 1265 | Data truncated for column 'c1' at row 2 |
| Warning | 1265 | Data truncated for column 'c2' at row 2 |
| Warning | 1265 | Data truncated for column 'c3' at row 2 |
+---------+------+----------------------------------------+
3 rows in set (0.00 sec)

mysql> SELECT * FROM add_pk_via_copy;
+----+----+---------------------+
| c1 | c2 | c3                  |
+----+----+---------------------+
|  0 |    | 0000-00-00 00:00:00 |
|  1 | a  | 2014-11-03 11:01:37 |
+----+----+---------------------+
2 rows in set (0.00 sec)
```

ALGORITHM=INPLACE 句を使用した場合、この設定ではデータの整合性を高い優先度とみなしているため、操作はさまざまな理由で失敗する可能性があります。このステートメントは、SQL_MODE 設定が十分に「厳密」でない場合、または主キーカラムに NULL 値が含まれている場合にエラーを出力します。これらの両方の要件に対応すれば、ALTER TABLE 操作は成功します。

```
mysql> CREATE TABLE add_pk_via_inplace (c1 INT, c2 VARCHAR(10), c3 DATETIME);
Query OK, 0 rows affected (0.02 sec)

mysql> INSERT INTO add_pk_via_inplace VALUES (1,'a','2014-11-03 11:01:37'),(NULL,NULL,NULL);
Query OK, 2 rows affected (0.00 sec)
Records: 2  Duplicates: 0  Warnings: 0

mysql> SELECT * FROM add_pk_via_inplace;
+------+------+---------------------+
| c1   | c2   | c3                  |
+------+------+---------------------+
|    1 | a    | 2014-11-03 11:01:37 |
| NULL | NULL | NULL                |
+------+------+---------------------+
2 rows in set (0.00 sec)

mysql> SET sql_mode = '';
Query OK, 0 rows affected (0.00 sec)

mysql> ALTER TABLE add_pk_via_inplace ADD PRIMARY KEY (c1,c2,c3), ALGORITHM=INPLACE;
ERROR 1846 (0A000): ALGORITHM=INPLACE is not supported. Reason: cannot silently convert NULL values,
as required in this SQL_MODE. Try ALGORITHM=COPY.

mysql> SET sql_mode ='strict_trans_tables';
Query OK, 0 rows affected (0.00 sec)

mysql> ALTER TABLE add_pk_via_inplace ADD PRIMARY KEY (c1,c2,c3), ALGORITHM=INPLACE;
ERROR 1138 (22004): Invalid use of NULL value
mysql> DELETE FROM add_pk_via_inplace WHERE c1 IS NULL OR c2 IS NULL OR c3 IS NULL;
Query OK, 1 row affected (0.01 sec)

mysql> SELECT * FROM add_pk_via_inplace;
+------+------+---------------------+
| c1   | c2   | c3                  |
+------+------+---------------------+
|    1 | a    | 2014-11-03 11:01:37 |
+------+------+---------------------+
1 row in set (0.00 sec)

mysql> ALTER TABLE add_pk_via_inplace ADD PRIMARY KEY (c1,c2,c3), ALGORITHM=INPLACE;
Query OK, 0 rows affected (0.09 sec)
Records: 0  Duplicates: 0  Warnings: 0
```

主キーなしでテーブルを作成すると、InnoDB は、主キーを自動的に選択します。これは、NOT NULL カラムで定義された最初の UNIQUE キー、またはシステムで生成されたキーである場合があります。隠れた余分なカラムの不確実性や、それに対して領域要件が発生する可能性を排除するには、CREATE TABLE ステートメントの一部として PRIMARY KEY 句を指定します。

14.11.6 オンライン DDL の実装の詳細

InnoDB テーブルに対する各 ALTER TABLE 操作は、次のいくつかの側面によって制御されます。

- テーブルの物理表現への何らかの変更があるかどうか、またはそれが純粋に、テーブル自体を変更することなく実行できるメタデータへの変更であるかどうか。

- テーブル内のデータの量が同じままか、増えているか、または減っているか。
- テーブルデータ内の変更がクラスタ化されたインデックス、セカンダリインデックス、またはその両方に関連しているかどうか。
- 変更されるテーブルとその他のテーブルの間に何らかの外部キー関係が存在するかどうか。このメカニクスは、foreign_key_checks 構成オプションが有効または無効のどちらになっているかによって異なります。
- テーブルがパーティション化されているかどうか。ALTER TABLE のパーティション化句は 1 つ以上のテーブルに関連する低レベルの操作に変換され、これらの操作はオンライン DDL の通常のルールに従います。
- テーブルデータをコピーする必要があるかどうか、テーブルを「インプレース」で再編成できるかどうか、またはその両方の組み合わせか。
- テーブルに自動インクリメントカラムが含まれているかどうか。
- ベースとなるデータベース操作の性質、または ALTER TABLE ステートメントで指定した LOCK 句によって、どのようなレベルのロックが必要とされるか。

このセクションでは、これらの要因が InnoDB テーブルでのさまざまな種類の ALTER TABLE 操作にどのような影響を与えるかについて説明します。

オンライン DDL のエラー状態

オンライン DDL 操作が失敗する可能性がある主な理由を次に示します。

- LOCK 句が、特定のタイプの DDL 操作とは互換性のない低レベルのロック (SHARED または NONE) を指定している場合。
- DDL 操作の初期および最終フェーズ中に短時間だけ必要な、テーブルに対する排他的ロックの取得を待機している間にタイムアウトが発生した場合。
- インデックス作成中に MySQL がディスク上に一時的なソートファイルを書き込んでいる間に tmpdir ファイルシステムのディスク領域が不足した場合。
- ALTER TABLE に長い時間がかかりすぎたり、並列 DML によるテーブル変更が多すぎたりして、一時的なオンラインログのサイズが innodb_online_alter_log_max_size 構成オプションの値を超えた場合。この状態は DB_ONLINE_LOG_TOO_BIG エラーの原因になります。
- 並列 DML が、元のテーブル定義では許可されるが、新しいテーブル定義では許可されないテーブルに変更を加えた場合。この操作は、MySQL がいちばん最後に、並列 DML ステートメントからのすべての変更を適用しようとしたときにのみ失敗します。たとえば、一意のインデックスの作成中にカラムに重複した値を挿入したり、そのカラムでの主キーのインデックスの作成中にカラムに NULL 値を挿入したりすることがあります。並列 DML によって行われた変更が優先され、ALTER TABLE 操作は実質的にロールバックされます。

構成オプション innodb_file_per_table は InnoDB テーブルの表現に大きな影響を与えますが、そのオプションが有効または無効のどちらになっているかや、テーブルが物理的に独自の .ibd ファイル内またはシステムテーブルスペースの内部のどちらに配置されているかにかかわらず、すべてのオンライン DDL 操作が同様に適切に機能します。

InnoDB には、テーブル内のすべてのデータを表すクラスタ化されたインデックスと、クエリーを高速化するためのオプションのセカンダリインデックスという 2 つのタイプのインデックスがあります。クラスタ化されたインデックスにはその B ツリーノード内のデータ値が含まれているため、クラスタ化されたインデックスの追加または削除には、データのコピーおよびテーブルの新しいコピーの作成が含まれます。ただし、セカンダリインデックスには、インデックスキーと主キーの値のみが含まれます。このタイプのインデックスは、クラスタ化されたインデックス内のデータをコピーすることなく作成または削除できます。各セカンダリインデックスには主キー値のコピー (必要に応じて、クラスタ化されたインデックスにアクセスするために使用されます) が含まれているため、主キーの定義を変更すると、すべてのセカンダリインデックスも再作成されます。

セカンダリインデックスの削除は単純です。内部の InnoDB システムテーブルと MySQL データディクショナリテーブルが、このインデックスは存在しなくなったという事実を反映するように更新されるだけです。InnoDB は、このインデックスに使用されているストレージをそれが含まれていたテーブルスペースに返して、新しいインデックスまたは追加のテーブル行がこの領域を使用できるようにします。

既存のテーブルにセカンダリインデックスを追加するには、InnoDB はそのテーブルをスキャンし、メモリーバッファーや一時ファイルを使用して各行をセカンダリインデックスキーカラムの値で順番にソートします。次に、B ツリーがキー値の順序で構築されます。これは、行をインデックスにランダムな順序で挿入するより効率的です。B ツリーノードはいっぱいになると分割されるため、このようにしてインデックスを構築するとインデックスのフィルファクタが高くなり、以降のアクセスがより効率的になります。

主キーと副キーのインデックス

従来より、MySQL サーバーと InnoDB はそれぞれ、テーブルおよびインデックス構造に関する独自のメタデータを保持しています。MySQL サーバーがこれらの情報を、トランザクションメカニズムによって保護されない .frm ファイル内に格納するのに対して、InnoDB は、独自のデータディクショナリをシステムテーブルスペースの一部として保持しています。DDL 操作が途中でクラッシュまたはその他の予期しないイベントによって中断された場合は、メタデータがこれらの 2 つの場所の間で整合性がない状態のままになり、起動エラーや、変更の途中であったテーブルへのアクセス不可などの問題が発生する可能性があります。InnoDB は現在、デフォルトのストレージエンジンであるため、このような問題への対処は高い優先度を持っています。これらの DDL 操作への拡張により、このような問題が発生する可能性のある期間が削減されます。

14.11.7 オンライン DDL でのクラッシュリカバリの動作のしくみ

ALTER TABLE ステートメントの実行中にサーバーがクラッシュしてもデータは失われませんが、クラッシュリカバリのプロセスは、クラスタ化されたインデックスの場合とセカンダリインデックスの場合で異なります。

InnoDB セカンダリインデックスの作成中にサーバーがクラッシュした場合、MySQL はリカバリ時に、部分的に作成されたインデックスをすべて削除します。ALTER TABLE または CREATE INDEX ステートメントを再実行する必要があります。

InnoDB のクラスタ化されたインデックスの作成中にクラッシュが発生した場合は、テーブル内のデータをまったく新しいクラスタ化されたインデックスにコピーする必要があるため、リカバリはより複雑です。すべての InnoDB テーブルが、クラスタ化されたインデックスとして格納されることに注意してください。次の説明では、テーブルとクラスタ化されたインデックスという言葉を区別なく使用しています。

MySQL は、既存のデータを元の InnoDB テーブルから目的のインデックス構造を持つ一時テーブルにコピーすることによって、新しいクラスタ化されたインデックスを作成します。データがこの一時テーブルに完全にコピーされたら、元のテーブルの名前が別の一時テーブル名に変更されます。新しいクラスタ化されたインデックスで構成される一時テーブルの名前が元のテーブルの名前に変更され、元のテーブルはデータベースから削除されます。

新しいクラスタ化されたインデックスの作成中にシステムクラッシュが発生した場合、データは失われませんが、このプロセス中に存在する一時テーブルを使用してリカバリプロセスを完了する必要があります。クラスタ化されたインデックスを再作成したり、大きなテーブルで主キーを再定義したり、あるいはこの操作中にシステムクラッシュが発生したりすることはまれであるため、このマニュアルではこのシナリオからのリカバリに関する情報は提供していません。

14.11.8 パーティション化された InnoDB テーブルに対するオンライン DDL

ALTER TABLE のパーティション化句を除き、パーティション化された InnoDB テーブルに対するオンライン DDL 操作は、通常の InnoDB テーブルに適用されるのと同じルールに従います。オンライン DDL のルールは、表 14.5「DDL 操作のオンラインステータスのサマリー」で概説されています。

ALTER TABLE のパーティション化句は、通常のパーティション化されていない InnoDB テーブルと同じ内部のオンライン DDL API を経由せず、ALGORITHM=DEFAULT および LOCK=DEFAULT との組み合わせでのみ許可されます。

ALTER TABLE ステートメントで ALTER TABLE のパーティション化句を使用した場合、パーティション化されたテーブルは、ALTER TABLE COPY アルゴリズムを使用して「再パーティション化」されます。つまり、新しいパーティション化されたテーブルは、新しいパーティション化スキームで作成されます。新しく作成されたテーブルには ALTER TABLE ステートメントによって適用されたすべての変更が含まれ、テーブルデータが新しいテーブル構造にコピーされます。

ALTER TABLE のパーティション化句を使用してテーブルのパーティション化を変更しない場合、または ALTER TABLE ステートメントでほかの何らかのパーティション管理を実行した場合、ALTER TABLE は、各テーブルパーティションで INPLACE アルゴリズムを使用します。ただし、INPLACE ALTER TABLE 操作が各パーティションで実行されると、複数のパーティションで実行されている操作のために、システムリソースへの要求が増加することに注意してください。

ALTER TABLE ステートメントのパーティション化句が、通常のパーティション化されていない InnoDB テーブルと同じ内部のオンライン DDL API を経由しないにもかかわらず、MySQL は引き続き、可能な場合はデータコピーとロックを最小限に抑えようとします。

- RANGE または LIST によってパーティション化されたテーブルに対する ADD PARTITION および DROP PARTITION では、どの既存のデータもコピーされません。
- TRUNCATE PARTITION では、すべてのタイプのパーティション化されたテーブルについて、どの既存のデータもコピーされません。

- 並列クエリーは、HASH または LIST によってパーティション化されたテーブルに対する ADD PARTITION および COALESCE PARTITION 中に許可されます。MySQL は、共有ロックを保持している間にデータをコピーします。
- REORGANIZE PARTITION、REBUILD PARTITION、あるいは LINEAR HASH または LIST によってパーティション化されたテーブルに対する ADD PARTITION または COALESCE PARTITION では、並列クエリーが許可されます。影響を受けるパーティションからのデータは、テーブルレベルの共有メタデータ (読み取り) ロックを保持している間にコピーされます。

> 注記
>
> InnoDB のパーティション化されたテーブルでは、全文検索 (FTS) と外部キーはサポートされていません。詳細は、セクション12.9.5「全文制限」およびセクション19.6.2「ストレージエンジンに関連するパーティショニング制限」を参照してください。

14.11.9 オンライン DDL の制限

オンライン DDL 操作を実行する場合は、次の制限を考慮に入れてください。

- テーブルをコピーするオンライン DDL 操作中に、ファイルは一時ディレクトリ (Unix では $TMPDIR、Windows では %TEMP%、または --tmpdir 構成変数で指定されたディレクトリ) に書き込まれます。各一時ファイルは、新しいテーブルまたはインデックス内に 1 つのカラムを保持できるだけの十分な大きさを持ち、最終的なテーブルまたはインデックスにマージされたらすぐに削除されます。
- どちらも同じインデックスを指定する DROP INDEX および ADD INDEX 句を含む ALTER TABLE ステートメントは、高速インデックス作成ではなくテーブルコピーを使用します。
- TEMPORARY TABLE でインデックスを作成した場合は、高速インデックス作成が使用されるのではなく、テーブルがコピーされます。これは MySQL Bug #39833 としてレポートされています。
- InnoDB は、外部キーに必要なインデックスをユーザーが削除しようとしたときにエラー事例を処理します。エラー 1553 に関連した詳細は、セクション14.19.5「InnoDB のエラーコード」を参照してください。
- ALTER TABLE の句 LOCK=NONE は、テーブル上に ON...CASCADE または ON...SET NULL 制約が存在する場合は許可されません。
- オンライン DDL の各 ALTER TABLE ステートメント中に、LOCK 句には関係なく、テーブルに対する排他的ロック (LOCK=EXCLUSIVE 句で指定されるのと同じ種類のロック) を必要とする短い期間が最初と最後に存在します。そのため、そのテーブル上で挿入、更新、削除、または SELECT ... FOR UPDATE を実行している長時間実行されるトランザクションが存在する場合は、オンライン DDL 操作が開始前に待機する可能性があります。また、ALTER TABLE の進行中に同様の長時間実行されるトランザクションが開始された場合は、オンライン DDL 操作が完了前に待機する可能性があります。
- オンライン ALTER TABLE 操作の実行時に、ALTER TABLE 操作を実行しているスレッドは、その同じテーブルに対してほかの接続スレッドから同時に実行された DML 操作の「オンラインログ」を適用します。これらの DML 操作が適用されると、重複したキーエントリのエラー (ERROR 1062 (23000): 重複したエントリ) が発生する可能性があります。これは、重複したエントリが一時的なだけで、「オンラインログ」のあとの方のエントリによって元に戻されるとしても同じです。これは、トランザクション中は制約を保持する必要のある、InnoDB での外部キー制約チェックの考え方に似ています。
- InnoDB テーブルに対する OPTIMIZE TABLE は、テーブルを再構築して、インデックス統計を更新し、クラスタ化されたインデックス内の未使用領域を解放するための ALTER TABLE 操作にマップされます。5.6.17 より前は、この操作に対するオンライン DDL のサポートはありません。主キーに現れる順序でキーが挿入されるため、セカンダリインデックスはそれほど効率的に作成されません。5.6.17 の時点では、InnoDB の通常のテーブルとパーティション化されたテーブルを再構築するためのオンライン DDL のサポートの追加によって、OPTIMIZE TABLE がサポートされます。詳細は、セクション14.11.1「オンライン DDL の概要」を参照してください。
- MySQL 5.6 より前に作成された InnoDB テーブルは、一時的なカラム (DATE、DATETIME、または TIMESTAMP) を含み、かつ ALTER TABLE ... ALGORITHM=COPY を使用して再構築されていないテーブルに対する ALTER TABLE ... ALGORITHM=INPLACE をサポートしていません。この場合は、ALTER TABLE ... ALGORITHM=INPLACE 操作によって次のエラーが返されます。

```
ERROR 1846 (0A000): ALGORITHM=INPLACE is not supported.
Reason: Cannot change column type INPLACE. Try ALGORITHM=COPY.
```

14.12 InnoDB の起動オプションおよびシステム変数

- true または false であるシステム変数は、サーバー起動時に変数の名前を指定することで有効にすることができ、--skip- プリフィクスを使用することで無効にすることができます。たとえば、InnoDB 適応型ハッシュインデックスを有効または無効にするには、コマンド行で --innodb_adaptive_hash_index または --skip-innodb_adaptive_hash_index を使用するか、オプションファイルで innodb_adaptive_hash_index または skip-innodb_adaptive_hash_index を使用します。
- 数値が指定されるシステム変数は、コマンド行で --var_name=value として指定するか、オプションファイルで var_name=value として指定できます。
- 多くのシステム変数は、実行時に変更できます (セクション5.1.5.2「動的システム変数」を参照してください)。
- GLOBAL および SESSION 変数スコープ修飾子については、SET ステートメントのドキュメントを参照してください。
- 特定のオプションでは、InnoDB データファイルの場所およびレイアウトが制御されます。セクション14.3「InnoDB の構成」では、これらのオプションを使用する方法について説明します。
- 初期段階では使用しないような一部のオプションは、マシンの処理能力やデータベースのワークロードに基づいて、InnoDB のパフォーマンス特性を調整する際に役立ちます。
- オプションおよびシステム変数の指定に関する詳細は、セクション4.2.3「プログラムオプションの指定」を参照してください。

表 14.6 InnoDB オプション/変数のリファレンス

名前	コマンド行	オプションファイル	システム変数	ステータス変数	変数スコープ	動的
daemon_memcached_enable_binlog	はい	はい	はい		グローバル	いいえ
daemon_memcached_engine_lib_name	はい	はい	はい		グローバル	いいえ
daemon_memcached_engine_lib_path	はい	はい	はい		グローバル	いいえ
daemon_memcached_option	はい	はい	はい		グローバル	いいえ
daemon_memcached_r_batch_size	はい	はい	はい		グローバル	いいえ
daemon_memcached_w_batch_size	はい	はい	はい		グローバル	いいえ
foreign_key_checks			はい		両方	はい
have_innodb			はい		グローバル	いいえ
ignore_builtin_innodb	はい	はい	はい		グローバル	いいえ
innodb	はい	はい				
innodb_adaptive_flushing	はい	はい	はい		グローバル	はい
innodb_adaptive_flushing_lwm	はい	はい	はい		グローバル	はい
innodb_adaptive_hash_index	はい	はい	はい		グローバル	はい
innodb_adaptive_max_sleep_delay	はい	はい	はい		グローバル	はい
innodb_additional_mem_pool_size	はい	はい	はい		グローバル	いいえ
innodb_api_bk_commit_interval	はい	はい	はい		グローバル	はい
innodb_api_disable_rowlock	はい	はい	はい		グローバル	いいえ
innodb_api_enable_binlog	はい	はい	はい		グローバル	いいえ
innodb_api_enable_mdl	はい	はい	はい		グローバル	いいえ
innodb_api_trx_level	はい	はい	はい		グローバル	はい
innodb_autoextend_increment	はい	はい	はい		グローバル	はい
innodb_autoinc_lock_mode	はい	はい	はい		グローバル	いいえ
Innodb_available_undo_logs				はい	グローバル	いいえ
Innodb_buffer_pool_bytes_data				はい	グローバル	いいえ
Innodb_buffer_pool_bytes_dirty				はい	グローバル	いいえ
innodb_buffer_pool_dump_at_shutdown	はい	はい	はい		グローバル	はい
innodb_buffer_pool_dump_now	はい	はい	はい		グローバル	はい

名前	コマンド行	オプションファイル	システム変数	ステータス変数	変数スコープ	動的
Innodb_buffer_pool_dump_status				はい	グローバル	いい
innodb_buffer_pool_filename	はい	はい	はい		グローバル	はい
innodb_buffer_pool_instances	はい	はい	はい		グローバル	いい
innodb_buffer_pool_load_abort	はい	はい	はい		グローバル	はい
innodb_buffer_pool_load_at_startup	はい	はい	はい		グローバル	いい
innodb_buffer_pool_load_now	はい	はい	はい		グローバル	はい
Innodb_buffer_pool_load_status				はい	グローバル	いい
Innodb_buffer_pool_pages_data				はい	グローバル	いい
Innodb_buffer_pool_pages_dirty				はい	グローバル	いい
Innodb_buffer_pool_pages_flushed				はい	グローバル	いい
Innodb_buffer_pool_pages_free				はい	グローバル	いい
Innodb_buffer_pool_pages_latched				はい	グローバル	いい
Innodb_buffer_pool_pages_misc				はい	グローバル	いい
Innodb_buffer_pool_pages_total				はい	グローバル	いい
Innodb_buffer_pool_read_ahead				はい	グローバル	いい
Innodb_buffer_pool_read_ahead_evicted				はい	グローバル	いい
Innodb_buffer_pool_read_requests				はい	グローバル	いい
Innodb_buffer_pool_reads				はい	グローバル	いい
innodb_buffer_pool_size	はい	はい	はい		グローバル	いい
Innodb_buffer_pool_wait_free				はい	グローバル	いい
Innodb_buffer_pool_write_requests				はい	グローバル	いい
innodb_change_buffer_max_size	はい	はい	はい		グローバル	はい
innodb_change_buffering	はい	はい	はい		グローバル	はい
innodb_checksum_algorithm	はい	はい	はい		グローバル	はい
innodb_checksums	はい	はい	はい		グローバル	いい
innodb_cmp_per_index_enabled	はい	はい	はい		グローバル	はい
innodb_commit_concurrency	はい	はい	はい		グローバル	はい
innodb_compression_failure_threshold_pct	はい	はい	はい		グローバル	はい
innodb_compression_level	はい	はい	はい		グローバル	はい
innodb_compression_pad_pct_max	はい	はい	はい		グローバル	はい
innodb_concurrency_tickets	はい	はい	はい		グローバル	はい
innodb_data_file_path	はい	はい	はい		グローバル	いい
Innodb_data_fsyncs				はい	グローバル	いい
innodb_data_home_dir	はい	はい	はい		グローバル	いい
Innodb_data_pending_fsyncs				はい	グローバル	いい
Innodb_data_pending_reads				はい	グローバル	いい
Innodb_data_pending_writes				はい	グローバル	いい
Innodb_data_read				はい	グローバル	いい
Innodb_data_reads				はい	グローバル	いい
Innodb_data_writes				はい	グローバル	いい
Innodb_data_written				はい	グローバル	いい
Innodb_dblwr_pages_written				はい	グローバル	いい
Innodb_dblwr_writes				はい	グローバル	いい

名前	コマンド行	オプションファイル	システム変数	ステータス変数	変数スコープ	動的
innodb_disable_sort_file_cache	はい	はい	はい		グローバル	はい
innodb_doublewrite	はい	はい	はい		グローバル	いいえ
innodb_fast_shutdown	はい	はい	はい		グローバル	はい
innodb_file_format	はい	はい	はい		グローバル	はい
innodb_file_format_check	はい	はい	はい		グローバル	いいえ
innodb_file_format_max	はい	はい	はい		グローバル	はい
innodb_file_per_table	はい	はい	はい		グローバル	はい
innodb_flush_log_at_timeout			はい		グローバル	はい
innodb_flush_log_at_trx_commit	はい	はい	はい		グローバル	はい
innodb_flush_method	はい	はい	はい		グローバル	いいえ
innodb_flush_neighbors	はい	はい	はい		グローバル	はい
innodb_flushing_avg_loops	はい	はい	はい		グローバル	はい
innodb_force_load_corrupted	はい	はい	はい		グローバル	いいえ
innodb_force_recovery	はい	はい	はい		グローバル	いいえ
innodb_ft_aux_table			はい		グローバル	はい
innodb_ft_cache_size	はい	はい	はい		グローバル	いいえ
innodb_ft_enable_diag_print	はい	はい	はい		グローバル	はい
innodb_ft_enable_stopword	はい	はい	はい		グローバル	はい
innodb_ft_max_token_size	はい	はい	はい		グローバル	いいえ
innodb_ft_min_token_size	はい	はい	はい		グローバル	いいえ
innodb_ft_num_word_optimize	はい	はい	はい		グローバル	はい
innodb_ft_result_cache_limit	はい	はい	はい		グローバル	はい
innodb_ft_server_stopword_table	はい	はい	はい		グローバル	はい
innodb_ft_sort_pll_degree	はい	はい	はい		グローバル	いいえ
innodb_ft_total_cache_size	はい	はい	はい		グローバル	いいえ
innodb_ft_user_stopword_table	はい	はい	はい		両方	はい
Innodb_have_atomic_builtins				はい	グローバル	いいえ
innodb_io_capacity	はい	はい	はい		グローバル	はい
innodb_io_capacity_max	はい	はい	はい		グローバル	はい
innodb_large_prefix	はい	はい	はい		グローバル	はい
innodb_lock_wait_timeout	はい	はい	はい		両方	はい
innodb_locks_unsafe_for_binlog	はい	はい	はい		グローバル	いいえ
innodb_log_buffer_size	はい	はい	はい		グローバル	いいえ
innodb_log_compressed_pages	はい	はい	はい		グローバル	はい
innodb_log_file_size	はい	はい	はい		グローバル	いいえ
innodb_log_files_in_group	はい	はい	はい		グローバル	いいえ
innodb_log_group_home_dir	はい	はい	はい		グローバル	いいえ
Innodb_log_waits				はい	グローバル	いいえ
Innodb_log_write_requests				はい	グローバル	いいえ
Innodb_log_writes				はい	グローバル	いいえ
innodb_lru_scan_depth	はい	はい	はい		グローバル	はい
innodb_max_dirty_pages_pct	はい	はい	はい		グローバル	はい
innodb_max_dirty_pages_pct_lwm	はい	はい	はい		グローバル	はい

名前	コマンド行	オプションファイル	システム変数	ステータス変数	変数スコープ	動的
innodb_max_purge_lag	はい	はい	はい		グローバル	はい
innodb_max_purge_lag_delay	はい	はい	はい		グローバル	はい
innodb_mirrored_log_groups	はい	はい	はい		グローバル	いい
innodb_monitor_disable	はい	はい	はい		グローバル	はい
innodb_monitor_enable	はい	はい	はい		グローバル	はい
innodb_monitor_reset	はい	はい	はい		グローバル	はい
innodb_monitor_reset_all	はい	はい	はい		グローバル	はい
Innodb_num_open_files				はい	グローバル	いい
innodb_old_blocks_pct	はい	はい	はい		グローバル	はい
innodb_old_blocks_time	はい	はい	はい		グローバル	はい
innodb_online_alter_log_max_size	はい	はい	はい		グローバル	はい
innodb_open_files	はい	はい	はい		グローバル	いい
innodb_optimize_fulltext_only	はい	はい	はい		グローバル	はい
Innodb_os_log_fsyncs				はい	グローバル	いい
Innodb_os_log_pending_fsyncs				はい	グローバル	いい
Innodb_os_log_pending_writes				はい	グローバル	いい
Innodb_os_log_written				はい	グローバル	いい
Innodb_page_size				はい	グローバル	いい
innodb_page_size	はい	はい	はい		グローバル	いい
Innodb_pages_created				はい	グローバル	いい
Innodb_pages_read				はい	グローバル	いい
Innodb_pages_written				はい	グローバル	いい
innodb_print_all_deadlocks	はい	はい	はい		グローバル	はい
innodb_purge_batch_size	はい	はい	はい		グローバル	はい
innodb_purge_threads	はい	はい	はい		グローバル	いい
innodb_random_read_ahead	はい	はい	はい		グローバル	はい
innodb_read_ahead_threshold	はい	はい	はい		グローバル	はい
innodb_read_io_threads	はい	はい	はい		グローバル	いい
innodb_read_only	はい	はい	はい		グローバル	いい
innodb_replication_delay	はい	はい	はい		グローバル	はい
innodb_rollback_on_timeout	はい	はい	はい		グローバル	いい
innodb_rollback_segments	はい	はい	はい		グローバル	はい
Innodb_row_lock_current_waits				はい	グローバル	いい
Innodb_row_lock_time				はい	グローバル	いい
Innodb_row_lock_time_avg				はい	グローバル	いい
Innodb_row_lock_time_max				はい	グローバル	いい
Innodb_row_lock_waits				はい	グローバル	いい
Innodb_rows_deleted				はい	グローバル	いい
Innodb_rows_inserted				はい	グローバル	いい
Innodb_rows_read				はい	グローバル	いい
Innodb_rows_updated				はい	グローバル	いい
innodb_sort_buffer_size	はい	はい	はい		グローバル	いい
innodb_spin_wait_delay	はい	はい	はい		グローバル	はい

名前	コマンド行	オプションファイル	システム変数	ステータス変数	変数スコープ	動的
innodb_stats_auto_recalc	はい	はい	はい		グローバル	はい
innodb_stats_method	はい	はい	はい		グローバル	はい
innodb_stats_on_metadata	はい	はい	はい		グローバル	はい
innodb_stats_persistent	はい	はい	はい		グローバル	はい
innodb_stats_persistent_sample_pages	はい	はい	はい		グローバル	はい
innodb_stats_sample_pages	はい	はい	はい		グローバル	はい
innodb_stats_transient_sample_pages	はい	はい	はい		グローバル	はい
innodb-status-file	はい	はい				
innodb_status_output	はい	はい	はい		グローバル	はい
innodb_status_output_locks	はい	はい	はい		グローバル	はい
innodb_strict_mode	はい	はい	はい		両方	はい
innodb_support_xa	はい	はい	はい		両方	はい
innodb_sync_array_size	はい	はい	はい		グローバル	いいえ
innodb_sync_spin_loops	はい	はい	はい		グローバル	はい
innodb_table_locks	はい	はい	はい		両方	はい
innodb_thread_concurrency	はい	はい	はい		グローバル	はい
innodb_thread_sleep_delay	はい	はい	はい		グローバル	はい
Innodb_truncated_status_writes				はい	グローバル	いいえ
innodb_undo_directory	はい	はい	はい		グローバル	いいえ
innodb_undo_logs	はい	はい	はい		グローバル	はい
innodb_undo_tablespaces	はい	はい	はい		グローバル	いいえ
innodb_use_native_aio	はい	はい	はい		グローバル	いいえ
innodb_use_sys_malloc	はい	はい	はい		グローバル	いいえ
innodb_version			はい		グローバル	いいえ
innodb_write_io_threads	はい	はい	はい		グローバル	いいえ
timed_mutexes	はい	はい	はい		グローバル	はい
unique_checks			はい		両方	はい

InnoDB コマンドオプション

- --ignore-builtin-innodb

コマンド行形式	--ignore-builtin-innodb
非推奨	はい
システム変数	ignore_builtin_innodb
スコープ	グローバル
動的	いいえ
型	ブール

MySQL 5.1 では、このオプションを使用すると、サーバーは組み込み InnoDB が存在しない場合と同様に動作し、代わりに InnoDB Plugin を使用できました。MySQL 5.6 では、InnoDB がデフォルトのストレージエンジンとなり、InnoDB Plugin は使用されないため、このオプションは無効です。MySQL 5.6.5 の時点では、無視されます。

- --innodb[=value]

コマンド行形式	--innodb[=value]
非推奨	5.6.21

型	列挙
デフォルト	ON
有効な値	OFF ON FORCE

サーバーが InnoDB サポートでコンパイルされた場合に、InnoDB ストレージエンジンのロードを制御します。このオプションの形式はトライステートであり、指定可能な値は OFF、ON、または FORCE です。セクション5.1.8.1「プラグインのインストールおよびアンインストール」を参照してください。

InnoDB を無効にするには、--innodb=OFF または --skip-innodb を使用します。この場合、デフォルトのストレージエンジンは InnoDB であるため、--default-storage-engine および --default-tmp-storage-engine を使用して、永続テーブルと TEMPORARY テーブルの両方についてデフォルトを別のエンジンに設定しないかぎりサーバーは開始しません。

MySQL 5.6.21 の時点では、--innodb=OFF および --skip-innodb オプションが非推奨となり、使用すると警告が発生します。これらのオプションは、今後の MySQL リリースで削除されます。

- --innodb-status-file

コマンド行形式	--innodb-status-file
型	ブール
デフォルト	OFF

InnoDB が MySQL データディレクトリに innodb_status.pid という名前のファイルを作成するかどうかを制御します。有効にすると、InnoDB は定期的に SHOW ENGINE INNODB STATUS の出力をこのファイルに書き込みます。

このファイルはデフォルトでは作成されません。これを作成するには、--innodb-status-file=1 オプションを付けて mysqld を起動します。このファイルは、通常のシャットダウン中に削除されます。

- --skip-innodb

InnoDB ストレージエンジンを無効にします。--innodb の説明を参照してください。

InnoDB システム変数

- daemon_memcached_enable_binlog

コマンド行形式	--daemon-memcached-enable-binlog=#
導入	5.6.6
システム変数	daemon_memcached_enable_binlog
スコープ	グローバル
動的	いいえ
型	ブール
デフォルト	false

このオプションの使用法の詳細は、セクション14.18「InnoDB と memcached の統合」を参照してください。

- daemon_memcached_engine_lib_name

コマンド行形式	--daemon-memcached-engine-lib-name=library
導入	5.6.6
システム変数	daemon_memcached_engine_lib_name
スコープ	グローバル
動的	いいえ

型	ファイル名
デフォルト	innodb_engine.so

InnoDB memcached プラグインを実装する共有ライブラリを指定します。

このオプションの使用法の詳細は、セクション14.18「InnoDB と memcached の統合」を参照してください。

- daemon_memcached_engine_lib_path

コマンド行形式	--daemon-memcached-engine-lib-path=directory
導入	5.6.6
システム変数	daemon_memcached_engine_lib_path
スコープ	グローバル
動的	いいえ
型	ディレクトリ名
デフォルト	NULL

InnoDB memcached プラグインを実装する共有ライブラリを含むディレクトリのパスです。デフォルト値は、MySQL プラグインディレクトリを表す NULL です。MySQL プラグインディレクトリの外部に配置されている別のストレージエンジンの memcached プラグインを指定していなければ、このパラメータを変更する必要はないはずです。

このオプションの使用法の詳細は、セクション14.18「InnoDB と memcached の統合」を参照してください。

- daemon_memcached_option

コマンド行形式	--daemon-memcached-option=options
導入	5.6.6
システム変数	daemon_memcached_option
スコープ	グローバル
動的	いいえ
型	文字列
デフォルト	

起動時に、空白文字で区切られた memcached オプションをベースとなる memcached メモリーオブジェクトのキャッシュデーモンに渡すために使用されます。たとえば、memcached が待機するポートを変更したり、同時接続の最大数を削減したり、鍵と値のペアの最大メモリーサイズを変更したり、エラーログに関するメッセージのデバッグを有効にしたりします。

このオプションの使用法の詳細は、セクション14.18「InnoDB と memcached の統合」を参照してください。memcached のオプションについては、memcached のマニュアルページを参照してください。

- daemon_memcached_r_batch_size

コマンド行形式	--daemon-memcached-r-batch-size=#
導入	5.6.6
システム変数	daemon_memcached_r_batch_size
スコープ	グローバル
動的	いいえ
型	数値
デフォルト	1

COMMIT を実行して新しいトランザクションを開始する前に、実行される memcached 読み取り操作 (get) の数を指定します。daemon_memcached_w_batch_size の対の片方です。

この値は、SQL ステートメントを使用してテーブルに行われた変更がすぐに memcached 操作に表示されるように、デフォルトで 1 に設定されています。ベースとなるテーブルが memcached インタフェースからのみア

クセスされているシステム上で、頻繁なコミットによるオーバーヘッドを削減するために、これを大きくすることがあります。大きすぎる値を設定すると、Undo データまたは Redo データの量によっては、長時間実行されるトランザクションの場合と同様に、一部のストレージでオーバーヘッドが発生する可能性があります。

このオプションの使用法の詳細は、セクション14.18「InnoDB と memcached の統合」を参照してください。

- daemon_memcached_w_batch_size

コマンド行形式	--daemon-memcached-w-batch-size=#
導入	5.6.6
システム変数	daemon_memcached_w_batch_size
スコープ	グローバル
動的	いいえ
型	数値
デフォルト	1

COMMIT を実行して新しいトランザクションを開始する前に、実行される memcached 書き込み操作 (add、set、incr など) の数を指定します。daemon_memcached_r_batch_size の対の一方です。

この値は、格納されるデータはすべて停止に備えて保持しておくことが重要であり、すぐにコミットされるべきであるという仮定に基づいて、デフォルトで 1 に設定されています。クリティカルでないデータを格納するときは、頻繁なコミットによるオーバーヘッドを削減するために、この値を大きくすることがあります。ただし、クラッシュ時に、コミットされていない最後の N-1 回の書き込み操作が失われる可能性があります。

このオプションの使用法の詳細は、セクション14.18「InnoDB と memcached の統合」を参照してください。

- ignore_builtin_innodb

コマンド行形式	--ignore-builtin-innodb
非推奨	はい
システム変数	ignore_builtin_innodb
スコープ	グローバル
動的	いいえ
型	ブール

このセクションの前半の 「InnoDB コマンドオプション」の下にある --ignore-builtin-innodb の説明を参照してください。

- innodb_adaptive_flushing

コマンド行形式	--innodb-adaptive-flushing=#
システム変数	innodb_adaptive_flushing
スコープ	グローバル
動的	はい
型	ブール
デフォルト	ON

ワークロードに基づいて、InnoDB バッファープール内のダーティーページをフラッシュする比率を動的に調整するかどうかを指定します。フラッシュ比率を動的に調整する目的は、I/O アクティビティーのバーストを回避することです。この設定はデフォルトで有効になっています。詳細は、セクション14.13.1.2「InnoDB バッファープールのフラッシュの頻度の構成」を参照してください。一般的な I/O チューニングのアドバイスについては、セクション8.5.7「InnoDB ディスク I/O の最適化」を参照してください。

- innodb_adaptive_flushing_lwm

コマンド行形式	--innodb-adaptive-flushing-lwm=#
導入	5.6.6

システム変数	innodb_adaptive_flushing_lwm
スコープ	グローバル
動的	はい
型	数値
デフォルト	10
最小値	0
最大値	70

適応型フラッシュが有効になっている Redo ログ容量の割合を表す低位境界値です。

- innodb_adaptive_hash_index

コマンド行形式	--innodb-adaptive-hash-index=#
システム変数	innodb_adaptive_hash_index
スコープ	グローバル
動的	はい
型	ブール
デフォルト	ON

InnoDB 適応型ハッシュインデックスが有効と無効のどちらになっているのかを示します。ワークロードに応じて、適応型ハッシュインデックスの作成を動的に有効または無効にして、クエリーのパフォーマンスを改善することが望ましい場合があります。適応型ハッシュインデックスがすべてのワークロードに役立つとは限らないため、現実的なワークロードを使用して、有効と無効の両方でベンチマークを実施してください。詳細は、セクション14.2.13.6「適応型ハッシュインデックス」を参照してください。

この変数はデフォルトで有効になっています。SET GLOBAL ステートメントを使用すると、サーバーを再起動せずに、このパラメータを変更できます。この設定を変更するには、SUPER 権限が必要です。また、サーバーの起動時に --skip-innodb_adaptive_hash_index を使用すると、無効にすることができます。

適応型ハッシュインデックスを無効にすると、すぐにハッシュテーブルが空になります。ハッシュテーブルが空になっても通常の操作は続行でき、ハッシュテーブルを使用していた実行中のクエリーは、代わりにインデックスの B ツリーに直接アクセスします。適応型ハッシュインデックスを再度有効にすると、通常の操作時にハッシュテーブルが再度移入されます。

- innodb_adaptive_max_sleep_delay

コマンド行形式	--innodb-adaptive-max-sleep-delay=#
導入	5.6.3
システム変数	innodb_adaptive_max_sleep_delay
スコープ	グローバル
動的	はい
型	数値
デフォルト	150000
最小値	0
最大値	1000000

現在のワークロードに応じて、InnoDB によって自動的に innodb_thread_sleep_delay の値が上下に調整されるようにします。ゼロ以外の値に指定すると、最大で innodb_adaptive_max_sleep_delay オプションで指定された最大値まで、自動的に innodb_thread_sleep_delay 値の動的な調整が行われます。値はマイクロ秒数を表しています。このオプションは、InnoDB スレッド数が 16 個を上回る高負荷のシステムで役立つことがあります。(実際には、同時接続数が数百または数千になる MySQL システムの大部分の変数です。)

詳細は、セクション14.13.5「InnoDB のスレッド並列性の構成」を参照してください。

- innodb_additional_mem_pool_size

コマンド行形式	--innodb-additional-mem-pool-size=#

非推奨	5.6.3
システム変数	innodb_additional_mem_pool_size
スコープ	グローバル
動的	いいえ
型	数値
デフォルト	8388608
最小値	2097152
最大値	4294967295

データディクショナリ情報およびその他の内部データ構造を格納する際に InnoDB で使用されるメモリープールのサイズ (バイト単位) です。アプリケーションに存在するテーブル数が多いほど、ここで割り当てるメモリー量も多くなります。このプール内のメモリーが InnoDB によって使い果たされると、オペレーティングシステムからのメモリーの割り当てが開始され、MySQL エラーログに警告メッセージが書き込まれます。デフォルトの値は 8M バイトです。

この変数は、InnoDB の内部メモリーアロケータに関連します。これは、innodb_use_sys_malloc が有効になっている場合は使用されません。MySQL 5.6.3 の時点では、innodb_additional_mem_pool_size は非推奨となり、今後の MySQL リリースで削除される予定です。

- innodb_api_bk_commit_interval

コマンド行形式	--innodb-api-bk-commit-interval=#
導入	5.6.7
システム変数	innodb_api_bk_commit_interval
スコープ	グローバル
動的	はい
型	数値
デフォルト	5
最小値	1
最大値	1073741824

InnoDB memcached インタフェースが使用されるアイドル状態の接続が自動コミットされる頻度 (秒単位) です。このオプションの使用法の詳細は、セクション14.18「InnoDB と memcached の統合」を参照してください。

- innodb_api_disable_rowlock

コマンド行形式	--innodb-api-disable-rowlock=#
導入	5.6.6
システム変数	innodb_api_disable_rowlock
スコープ	グローバル
動的	いいえ
型	ブール
デフォルト	OFF

この変数を使用すると、InnoDB memcached で DML 操作が実行されるときに、行ロックが無効になります。デフォルトでは、innodb_api_disable_rowlock は OFF に設定されており、memcached が get および set 操作の行ロックをリクエストします。innodb_api_disable_rowlock を ON に設定すると、memcached は行ロックの代わりに、テーブルロックをリクエストします。

innodb_api_disable_rowlock オプションは動的ではありません。これは mysqld コマンド行で指定するか、または MySQL 構成ファイルに入力する必要があります。構成は、MySQL サーバーが起動されるたびに行うプラグインのインストール時に有効になります。

- innodb_api_enable_binlog

コマンド行形式	--innodb-api-enable-binlog=#
導入	5.6.6
システム変数	innodb_api_enable_binlog
スコープ	グローバル
動的	いいえ
型	ブール
デフォルト	OFF

MySQL バイナリログとともに、InnoDB memcached プラグインを使用できます。このオプションの使用法の詳細は、セクション14.18「InnoDB と memcached の統合」を参照してください。

- innodb_api_enable_mdl

コマンド行形式	--innodb-api-enable-mdl=#
導入	5.6.6
システム変数	innodb_api_enable_mdl
スコープ	グローバル
動的	いいえ
型	ブール
デフォルト	OFF

InnoDB memcached プラグインで使用されるテーブルをロックします。これにより、SQL インタフェースから DDL によって削除または変更できなくなります。このオプションの使用法の詳細は、セクション14.18「InnoDB と memcached の統合」を参照してください。

- innodb_api_trx_level

コマンド行形式	--innodb-api-trx-level=#
導入	5.6.6
システム変数	innodb_api_trx_level
スコープ	グローバル
動的	はい
型	数値
デフォルト	0

memcached インタフェースで処理されたクエリー上のトランザクション分離レベルを制御できます。このオプションの使用法の詳細は、セクション14.18「InnoDB と memcached の統合」を参照してください。よく聞く名前に対応する定数は、次のとおりです。

- 0 = READ UNCOMMITTED
- 1 = READ COMMITTED
- 2 = REPEATABLE READ
- 3 = SERIALIZABLE

- innodb_autoextend_increment

コマンド行形式	--innodb-autoextend-increment=#
システム変数	innodb_autoextend_increment
スコープ	グローバル
動的	はい
型	数値

デフォルト (≥ 5.6.6)	64
デフォルト (≤ 5.6.5)	8
最小値	1
最大値	1000

InnoDB の自動拡張システムテーブルスペースファイルがいっぱいになったときに、そのサイズを拡張する際の増分サイズ (M バイト単位) です。デフォルト値は、MySQL 5.6.6 の時点では 64、それよりも前では 8 です。この変数によって、innodb_file_per_table=1 を使用した場合に作成されるテーブルごとのテーブルスペースファイルは影響を受けません。innodb_autoextend_increment の値には関係なく、これらのファイルは自動拡張されます。拡張は少量で始まり、その後の拡張は増分が 4MB で発生します。

- innodb_autoinc_lock_mode

コマンド行形式	--innodb-autoinc-lock-mode=#
システム変数	innodb_autoinc_lock_mode
スコープ	グローバル
動的	いいえ
型	数値
デフォルト	1
有効な値	0 1 2

自動インクリメント値を生成する際に使用されるロックモードです。許可される値は、「従来」を表す 0、「連続」 を表す 1、または 「インターリーブ」 を表す 2 です。セクション14.6.5「InnoDB での AUTO_INCREMENT 処理」では、これらのモードの特性について説明します。

この変数のデフォルトは、1 (「連続」ロックモード) です。

- innodb_buffer_pool_dump_at_shutdown

コマンド行形式	--innodb-buffer-pool-dump-at-shutdown=#
導入	5.6.3
システム変数	innodb_buffer_pool_dump_at_shutdown
スコープ	グローバル
動的	はい
型	ブール
デフォルト	OFF

次回再起動時のウォームアッププロセスの時間を短縮するために、MySQL サーバーのシャットダウン時に、InnoDB のバッファープールにキャッシュされるページを記録するかどうかを指定します。一般に、innodb_buffer_pool_load_at_startup と組み合わせて使用されます。

関連情報については、セクション14.13.1.5「再起動を高速化するための InnoDB バッファープールのプリロード」を参照してください。

- innodb_buffer_pool_dump_now

コマンド行形式	--innodb-buffer-pool-dump-now=#
導入	5.6.3
システム変数	innodb_buffer_pool_dump_now
スコープ	グローバル
動的	はい
型	ブール

デフォルト	OFF

InnoDB のバッファープールにキャッシュされるページをすぐに記録します。一般に、innodb_buffer_pool_load_now と組み合わせて使用されます。

関連情報については、セクション14.13.1.5「再起動を高速化するための InnoDB バッファープールのプリロード」を参照してください。

- innodb_buffer_pool_filename

コマンド行形式	--innodb-buffer-pool-filename=file
導入	5.6.3
システム変数	innodb_buffer_pool_filename
スコープ	グローバル
動的	はい
型	ファイル名
デフォルト	ib_buffer_pool

innodb_buffer_pool_dump_at_shutdown または innodb_buffer_pool_dump_now で生成されるテーブルスペース ID およびページ ID のリストを保持するファイルの名前を指定します。テーブルスペース ID およびページ ID は、space, page_id という形式で保存されます。デフォルトでは、このファイルは InnoDB データディレクトリに配置されます。

関連情報については、セクション14.13.1.5「再起動を高速化するための InnoDB バッファープールのプリロード」を参照してください。

- innodb_buffer_pool_instances

コマンド行形式	--innodb-buffer-pool-instances=#
システム変数	innodb_buffer_pool_instances
スコープ	グローバル
動的	いいえ
型	数値
デフォルト (その他, ≥ 5.6.6)	8
デフォルト (Windows, 32 ビットプラットフォーム, ≥ 5.6.6)	(autosized)
デフォルト (≤ 5.6.5)	1
最小値	1
最大値	64

InnoDB のバッファープールが分割される領域の数です。バッファープールが数 G バイトの範囲にあるシステムでは、バッファープールを個別のインスタンスに分割すると、キャッシュされたページに対して異なるスレッドが読み取りおよび書き込みを行うときの競合が減るため、並列性が向上する場合があります。バッファープールに格納される各ページまたはバッファープールから読み取られる各ページは、ハッシュ関数を使用して、バッファープールインスタンスのいずれかにランダムに割り当てられます。各バッファープールは、独自の空きリスト、フラッシュリスト、LRU、およびバッファープールに接続されたその他のすべてのデータ構造を管理し、独自のバッファープール相互排他ロックによって保護されます。

このオプションは、innodb_buffer_pool_size を 1G バイト以上のサイズに設定した場合にのみ有効になります。指定した合計サイズは、すべてのバッファープール間で分割されます。最高の効率を得るには、innodb_buffer_pool_instances と innodb_buffer_pool_size の組み合わせを、各バッファープールインスタンスが少なくとも 1G バイトになるように指定します。

MySQL 5.6.6 より前では、デフォルトは 1 です。MySQL 5.6.6 の時点では、デフォルトは 8 です。ただし、32 ビットの Windows システムでは、デフォルトは innodb_buffer_pool_size の値に依存します。

- innodb_buffer_pool_size が 1.3G バイトよりも大きい場合は、innodb_buffer_pool_instances のデフォルトが innodb_buffer_pool_size/128M バイトになり、チャンクごとに個別のメモリー割り当てリクエストを持ちま

す。32 ビット版 Windows で単一のバッファープールで必要となる連続したアドレス空間を割り当てることができないという重大なリスクが存在する境界として、1.3G バイトが選択されました。

- それ以外の場合、デフォルトは 1 です。

- innodb_buffer_pool_load_abort

コマンド行形式	--innodb-buffer-pool-load-abort=#
導入	5.6.3
システム変数	innodb_buffer_pool_load_abort
スコープ	グローバル
動的	はい
型	ブール
デフォルト	OFF

innodb_buffer_pool_load_at_startup または innodb_buffer_pool_load_now でトリガーされる InnoDB のバッファープールの内容をリストアするプロセスを中断します。

関連情報については、セクション14.13.1.5「再起動を高速化するための InnoDB バッファープールのプリロード」を参照してください。

- innodb_buffer_pool_load_at_startup

コマンド行形式	--innodb-buffer-pool-load-at-startup=#
導入	5.6.3
システム変数	innodb_buffer_pool_load_at_startup
スコープ	グローバル
動的	いいえ
型	ブール
デフォルト	OFF

MySQL サーバーの起動時に、以前に保持されたときと同じページをロードすることで、InnoDB のバッファープールが自動的にウォームアップされるように指定します。一般に、innodb_buffer_pool_dump_at_shutdown と組み合わせて使用されます。

関連情報については、セクション14.13.1.5「再起動を高速化するための InnoDB バッファープールのプリロード」を参照してください。

- innodb_buffer_pool_load_now

コマンド行形式	--innodb-buffer-pool-load-now=#
導入	5.6.3
システム変数	innodb_buffer_pool_load_now
スコープ	グローバル
動的	はい
型	ブール
デフォルト	OFF

サーバーの再起動を待機せずにデータページのセットをロードすることで、InnoDB のバッファープールをすぐにウォームアップします。ベンチマーク時にキャッシュメモリーを既知の状態に戻したり、レポートやメンテナンスのためにクエリーを実行したあとに、MySQL サーバーの通常のワークロードを再開する準備をしたりする際に役立ちます。

関連情報については、セクション14.13.1.5「再起動を高速化するための InnoDB バッファープールのプリロード」を参照してください。

- innodb_buffer_pool_size

コマンド行形式	--innodb-buffer-pool-size=#
システム変数	innodb_buffer_pool_size
スコープ	グローバル
動的	いいえ
型	数値
デフォルト	134217728
最小値	5242880
最大値 (64 ビットプラットフォーム)	2**64-1
最大値 (32 ビットプラットフォーム)	2**32-1

InnoDB がテーブルおよびインデックスのデータをキャッシュするメモリー領域であるバッファープールのサイズ (バイト単位) です。デフォルト値は 128M バイトです。最大値は、CPU アーキテクチャーによって異なります。最大値は、32 ビットシステムでは 4294967295 (2^{32}-1)、64 ビットシステムでは 18446744073709551615 (2^{64}-1) です。32 ビットシステムでは、CPU アーキテクチャーおよびオペレーティングシステムに、指定された最大値よりも小さい実用的な最大サイズが課されている可能性があります。バッファープールのサイズが 1G バイトよりも大きい場合に、innodb_buffer_pool_instances を 1 よりも大きい値に設定すると、高負荷のサーバーで拡張性を改善できます。

この値を大きく設定するほど、テーブル内の同じデータに複数回アクセスするために必要なディスク I/O が少なくなります。専用のデータベースサーバーでは、これを最大でマシンの物理メモリーサイズの 80% まで設定することがあります。次のようなその他の問題が発生した場合は、この値を小さくする準備をしてください。

- 物理メモリーが競合すると、オペレーティングシステムでページングが発生する可能性があります。
- InnoDB では、割り当てられた領域の合計が指定されたサイズよりも約 10% 大きくなるように、バッファーおよび制御構造用に追加のメモリーが予約されています。
- アドレス空間は隣接しているはすです。これにより、Windows システムで特定のアドレスにロードする DLL に関する問題が発生する可能性があります。
- バッファープールを初期化する時間は、ほぼそのサイズに比例しています。大規模なインストールでは、この初期化時間が重要となる場合もあります。たとえば、最新の Linux x86_64 サーバーで 10G バイトのバッファープールを初期化するには、約 6 秒かかります。セクション8.9.1「InnoDB バッファープール」を参照してください。

- innodb_change_buffer_max_size

コマンド行形式	--innodb-change-buffer-max-size=#
導入	5.6.2
システム変数	innodb_change_buffer_max_size
スコープ	グローバル
動的	はい
型	数値
デフォルト	25
最小値	0
最大値	50

バッファープールの合計サイズの割合として示した、InnoDB の変更バッファーの最大サイズです。この値は、MySQL サーバーで頻繁に挿入、更新、および削除アクティビティーが発生する場合は大きくし、MySQL サーバーでレポート用に使用されるデータが変更されない場合は小さくするとよいでしょう。一般的な I/O チューニングのアドバイスについては、セクション8.5.7「InnoDB ディスク I/O の最適化」を参照してください。

- innodb_change_buffering

コマンド行形式	--innodb-change-buffering=#
システム変数	innodb_change_buffering
スコープ	グローバル
動的	はい
型	列挙
デフォルト	all
有効な値	inserts deletes purges changes all none

InnoDB がバッファリングの変更 (I/O 操作を連続して実行できるように、セカンダリインデックスへの書き込み操作を遅延させる最適化) を実行するかどうかを指定します。許可される値は inserts (挿入操作のバッファリング)、deletes (削除操作のバッファリング。厳密に言えば、パージ操作時にあとで削除するインデックスレコードにマークを付ける書き込み)、changes (挿入操作および削除マーク操作のバッファリング)、purges (パージ操作のバッファリング。削除されたインデックスエントリのガベージコレクションが最終的に実行される書き込み)、all (挿入、削除マーク、パージ操作のバッファリング)、および none (操作のバッファリングなし) です。デフォルトは all です。詳細については、セクション14.13.4「InnoDB 変更バッファリングの構成」を参照してください。一般的な I/O チューニングのアドバイスについては、セクション8.5.7「InnoDB ディスク I/O の最適化」を参照してください。

- innodb_checksum_algorithm

コマンド行形式	--innodb-checksum-algorithm=#
導入	5.6.3
システム変数	innodb_checksum_algorithm
スコープ	グローバル
動的	はい
型	列挙
デフォルト (≥ 5.6.7)	innodb
デフォルト (5.6.6)	crc32
デフォルト (≤ 5.6.5)	innodb
有効な値	innodb crc32 none strict_innodb strict_crc32 strict_none

InnoDB の各テーブルスペースの各ディスクブロックに格納されているチェックサムを生成および検証する方法を指定します。

MySQL 5.6.3 の時点で innodb_checksums オプションは、innodb_checksum_algorithm で置き換えられました。次の値は、互換性を保つために提供されています。

- innodb_checksum_algorithm=innodb は innodb_checksums=ON と同じです。

- innodb_checksum_algorithm=none は innodb_checksums=OFF と同じです。

競合を回避するには、構成ファイルおよび MySQL 起動スクリプトから innodb_checksums への参照を削除します。

値 innodb は、すべての MySQL バージョンとの下位互換性があります。値 crc32 では、より高速に、変更されたすべてのブロックのチェックサムを計算し、ディスク読み取りごとにチェックサムをチェックするアルゴリズムが使用されます。値 none では、ブロックデータに基づいて値が計算されるのではなく、チェックサムフィールドに定数値が書き込まれます。テーブルスペース内のブロックは、古い値、新しい値、およびチェックサムなしの値を混在させて使用でき、データが更新されるにつれ徐々に更新されます。テーブルスペース内のブロックが crc32 アルゴリズムを使用するように変更されたあとは、関連付けられたテーブルを以前のバージョンの MySQL で読み取ることはできません。

strict_* 形式の機能は、innodb、crc32、および none と同じです。ただし、InnoDB は、同じテーブルスペース内でチェックサム値の混在が発生した場合に停止します。これらのオプションを完全に新しいインスタンスで使用するだけで、はじめてでもすべてのテーブルスペースを設定できます。strict_* 設定では、ディスクの読み取り時に新しいチェックサム値と古いチェックサム値の両方を受け入れるために、その両方を計算する必要がないため、多少高速になります。

次の表には、none、innodb、crc32 オプション値、およびそれぞれに対応する strict_ オプション値間の相違点を示します。none、innodb、および crc32 では、特定のタイプのチェックサム値が各データブロックに書き込まれますが、互換性を保つために、読み取り操作中にブロックを検証する際に、その他のチェックサム値のいずれかが受け入れられます。strict_ 形式の各パラメータでは、1 種類のチェックサムのみが認識されます。これにより、検証が高速になりますが、インスタンス内のすべての InnoDB データファイルが同じ innodb_checksum_algorithm 値で作成される必要があります。

表 14.7 innodb_checksum_algorithm で許可される設定

値	生成されるチェックサム (書き込み時)	許可されるチェックサム (読み取り時)
none	定数。	none、innodb、または crc32 で生成されるチェックサムのいずれか。
innodb	ソフトウェアで InnoDB の元のアルゴリズムを使用して計算されたチェックサム。	none、innodb、または crc32 で生成されるチェックサムのいずれか。
crc32	crc32 アルゴリズムを使用して計算されたチェックサム (ハードウェアの支援を得て実行される可能性もあります)。	none、innodb、または crc32 で生成されるチェックサムのいずれか。
strict_none	定数	none で生成されるチェックサムのみ。
strict_innodb	ソフトウェアで InnoDB の元のアルゴリズムを使用して計算されたチェックサム。	innodb で生成されるチェックサムのみ。
strict_crc32	crc32 アルゴリズムを使用して計算されたチェックサム (ハードウェアの支援を得て実行される可能性もあります)。	crc32 で生成されるチェックサムのみ。

innodb_checksum_algorithm のデフォルト値は MySQL 5.6.6 で innodb から crc32 に変更されましたが、以前の MySQL バージョンへのダウングレード中の InnoDB データファイルの互換性向上のため、および MySQL Enterprise Backup で使用するために 5.6.7 で innodb に戻されました。検出された制限には、次のものが含まれます。

- CRC32 チェックサムを含む .ibd ファイルは、5.6.3 より前の MySQL バージョンへのダウングレード中に問題が発生する可能性があります。MySQL 5.6.3 以降では、ディスクからブロックを読み取るとき、そのブロックの新しいチェックサム値と古いチェックサム値のどちらも正しいとして認識します。それにより、アルゴリズムの設定には関係なく、アップグレードおよびダウングレード中にそのデータブロックの互換性を保証します。新しいチェックサム値で書き込まれたデータが 5.6.3 より前のレベルの MySQL によって処理された場合は、破損しているとしてレポートされる可能性があります。

- 3.8.0 までのバージョンの MySQL Enterprise Backup は、CRC32 チェックサムを使用するテーブルスペースのバックアップをサポートしていません。MySQL Enterprise Backup は、CRC32 チェックサムのサポートを 3.8.1 で (いくつかの制限付きで) 追加しています。詳細は、MySQL Enterprise Backup 3.8.1 の変更履歴を参照してください。

crc32 チェックサムアルゴリズムに関する追加情報については、セクション14.13.15「チェックサムの高速化のための CRC32 チェックサムアルゴリズムの使用」を参照してください。

- innodb_checksums

コマンド行形式	--innodb-checksums
非推奨	5.6.3
システム変数	innodb_checksums
スコープ	グローバル
動的	いいえ
型	ブール
デフォルト	ON

InnoDB では、ディスクから読み取られるすべてのテーブルスペースページ上でチェックサム検証を使用することで、ハードウェアの障害やデータファイルの破損に対する追加のフォールトトレランスを実現できます。この検証はデフォルトで有効になっています。特殊な状況 (ベンチマークの実行時など) では、このような追加の安全機能は --skip-innodb-checksums を使用して無効にすることができます。innodb_checksum_algorithm を使用すると、チェックサムを計算する方法を指定できます。

MySQL 5.6.3 以降では、このオプションは非推奨となり、innodb_checksum_algorithm で置き換えられました。innodb_checksum_algorithm=innodb は innodb_checksums=ON (デフォルト) と同じです。innodb_checksum_algorithm=none は innodb_checksums=OFF と同じです。innodb_checksum_algorithm との競合を回避するために、構成ファイルおよび起動スクリプトからすべての innodb_checksums オプションを削除してください。innodb_checksums=OFF によって自動的に innodb_checksum_algorithm=none が設定され、innodb_checksums=ON は無視され、innodb_checksum_algorithm のその他の設定でオーバーライドされます。

- innodb_cmp_per_index_enabled

コマンド行形式	--innodb-cmp-per-index-enabled=#
導入	5.6.7
システム変数	innodb_cmp_per_index_enabled
スコープ	グローバル
動的	はい
型	ブール
デフォルト	OFF
有効な値	OFF ON

INFORMATION_SCHEMA.INNODB_CMP_PER_INDEX テーブルでインデックスごとの圧縮関連の統計を有効にします。これらの統計を収集すると負荷が高くなるため、このオプションは、InnoDB の圧縮済みテーブルに関連するパフォーマンスチューニング時に開発、テスト、またはスレーブのインスタンス上でのみ有効にしてください。

- innodb_commit_concurrency

コマンド行形式	--innodb-commit-concurrency=#
システム変数	innodb_commit_concurrency
スコープ	グローバル
動的	はい
型	数値
デフォルト	0
最小値	0

最大値	1000

同時にコミットできるスレッドの数です。値を 0 (デフォルト) にすると、任意の数のトランザクションを同時にコミットすることが許可されます。

innodb_commit_concurrency の値は、実行時にゼロからゼロ以外 (またはその逆) に変更できません。ゼロ以外の値から別のゼロ以外の値に変更することはできます。

- innodb_compression_failure_threshold_pct

コマンド行形式	--innodb-compression-failure-threshold-pct=#
導入	5.6.7
システム変数	innodb_compression_failure_threshold_pct
スコープ	グローバル
動的	はい
型	数値
デフォルト	5
最小値	0
最大値	100

高負荷での圧縮の失敗を回避するために、圧縮されたページ内のパディングの追加が MySQL で開始されるカットオフポイントを設定します。値をゼロにすると、圧縮の効率性をモニターするメカニズムが無効になり、パディングの量が動的に調整されます。

- innodb_compression_level

コマンド行形式	--innodb-compression-level=#
導入	5.6.7
システム変数	innodb_compression_level
スコープ	グローバル
動的	はい
型	数値
デフォルト	6
最小値	0
最大値	9

InnoDB の圧縮されたテーブルおよびインデックスで使用される zlib 圧縮のレベルを指定します。

- innodb_compression_pad_pct_max

コマンド行形式	--innodb-compression-pad-pct-max=#
導入	5.6.7
システム変数	innodb_compression_pad_pct_max
スコープ	グローバル
動的	はい
型	数値
デフォルト	50
最小値	0
最大値	75

圧縮された各ページ内の空き領域として予約できる最大の割合を指定します。これにより、圧縮されたテーブルまたはインデックスが更新され、データが再度圧縮される可能性があるときに、ページ内のデータおよび変更ログを再編成する余地が得られます。innodb_compression_failure_threshold_pct がゼロ以外の値に設定され、圧縮エラーの比率がカットオフポイントを超えたときにのみ適用されます。

- innodb_concurrency_tickets

コマンド行形式	--innodb-concurrency-tickets=#
システム変数	innodb_concurrency_tickets
スコープ	グローバル
動的	はい
型	数値
デフォルト (≥ 5.6.6)	5000
デフォルト (≤ 5.6.5)	500
最小値	1
最大値	4294967295

同時に InnoDB に入ることができるスレッドの数を決定します。スレッドが InnoDB に入ろうとしたときに、すでにスレッド数が並列実行の制限に達している場合は、そのスレッドがキューに配置されます。スレッドが InnoDB に入ることが許可されている場合は、innodb_concurrency_tickets の値に等しい数の「空きチケット」が付与され、スレッドはそのチケットを使い果たすまで自由に InnoDB に出入りすることができます。それ以降は、スレッドが次に InnoDB に入ろうとしたときに、再度並列実行チェックの対象となります (キューに入る対象となる可能性もあります)。デフォルト値は、MySQL 5.6.6 の時点では 5000、それよりも前では 500 です。

innodb_concurrency_tickets 値を小さくすると、1、2 行しか処理する必要のない小規模なトランザクションと、多数の行を処理する大規模なトランザクションが競合する可能性が高くなります。innodb_concurrency_tickets 値を小さくする欠点は、大規模なトランザクションが完了するまでに何回もキュー間をループする必要があるため、タスクを完了するために必要な時間が長くなる点です。

innodb_concurrency_tickets 値を大きくすると、大規模なトランザクションで (innodb_thread_concurrency で制御される) キューの終了時の位置を待機する時間が短くなり、行を取得する時間が長くなります。また、大規模なトランザクションでは、タスクを完了するために必要なキューとの間の移動も少なくなります。innodb_concurrency_tickets 値を大きくする欠点は、同時に実行する大規模なトランザクションの数が非常に多くなることで、小規模なトランザクションが実行されるまでの待機時間が長くなるため、枯渇する可能性がある点です。

innodb_thread_concurrency 値をゼロ以外にすると、大規模なトランザクションと小規模なトランザクション間の適切なバランスを見つけるために、innodb_concurrency_tickets 値を上下に調整する必要がある場合があります。SHOW ENGINE INNODB STATUS レポートには、キューを通過する現時点で実行中のトランザクション用に残されているチケットの数が表示されます。このデータは、INFORMATION_SCHEMA.INNODB_TRX テーブルの TRX_CONCURRENCY_TICKETS カラムから取得することもできます。

詳細は、セクション14.13.5「InnoDB のスレッド並列性の構成」を参照してください。

- innodb_data_file_path

コマンド行形式	--innodb-data-file-path=name
システム変数	innodb_data_file_path
スコープ	グローバル
動的	いいえ
型	文字列
デフォルト (≥ 5.6.7)	ibdata1:12M:autoextend
デフォルト (≤ 5.6.6)	ibdata1:10M:autoextend

InnoDB の各データファイルへのパスとそれらのサイズです。各データファイルへの完全ディレクトリパスは、ここに指定された各パスに innodb_data_home_dir を結合することで形成されます。サイズ値に K、M、または G を追加することで、ファイルサイズが K バイト、M バイト、または G バイト (1024M バイト) で指定されます。データファイルのサイズをキロバイト (K バイト) で指定する場合は、1024 の倍数で指定してください。それ以外の場合は、K バイト値はもっとも近いメガバイト (M バイト) の境界で丸められます。ファイルサイズの合計は、わずかに 10M バイトを上回る大きさにする必要があります。innodb_data_file_path を指定しない場合は、デフォルトの動作で、ibdata1 という名前の単一の自動拡張データファイルが 12M バイトをわずかに上回る大きさで作成されます。各ファイルのサイズ制限は、オペレーティングシステムによって決定されます。大きいファイルをサポートするオペレーティングシステムでは、4G バイトよりも大きいファイルサイズを設定で

きます。データファイルとして生のディスクパーティションを使用することもできます。InnoDB のテーブルスペースファイルの構成についての詳細は、セクション14.3「InnoDB の構成」を参照してください。

- innodb_data_home_dir

コマンド行形式	--innodb-data-home-dir=path
システム変数	innodb_data_home_dir
スコープ	グローバル
動的	いいえ
型	ディレクトリ名

システムテーブルスペース内のすべての InnoDB のデータファイルのディレクトリパスに共通する部分です。この設定によって、innodb_file_per_table を有効にしたときの file-per-table テーブルスペースの場所は影響を受けません。デフォルト値は、MySQL の data ディレクトリです。値を空の文字列として指定した場合は、innodb_data_file_path 内で完全なファイルパスを使用できます。

- innodb_disable_sort_file_cache

コマンド行形式	--innodb-disable-sort-file-cache=#
導入	5.6.4
システム変数	innodb_disable_sort_file_cache
スコープ	グローバル
動的	はい
型	ブール
デフォルト	OFF

この変数を有効にすると、マージソート一時ファイル用のオペレーティングシステムファイルシステムのキャッシュが無効になります。その結果、このようなファイルが O_DIRECT の同等のものとともに開きます。この変数は MySQL 5.6.4 で追加されました。

- innodb_doublewrite

コマンド行形式	--innodb-doublewrite
システム変数	innodb_doublewrite
スコープ	グローバル
動的	いいえ
型	ブール
デフォルト	ON

この変数を有効にすると (デフォルト)、InnoDB にすべてのデータが 2 回 (まず二重書き込みバッファー、次に実際のデータファイルに) 格納されます。データの整合性や失敗の可能性の懸念より、ベンチマークや最高のパフォーマンスが必要なケースでは、--skip-innodb_doublewrite を使用すれば、この変数を無効にすることができます。

- innodb_fast_shutdown

コマンド行形式	--innodb-fast-shutdown[=#]
システム変数	innodb_fast_shutdown
スコープ	グローバル
動的	はい
型	数値
デフォルト	1
有効な値	0 1 2

InnoDB のシャットダウンモードです。この値を 0 にすると、InnoDB は低速シャットダウン、完全なパージ、および挿入バッファーのマージを実行してから、シャットダウンします。この値を 1 (デフォルト) にすると、InnoDB はシャットダウン時に、これらの操作をスキップします。このプロセスは、高速シャットダウンと呼ばれます。この値を 2 にすると、InnoDB は MySQL がクラッシュした場合と同様に、そのログをフラッシュし、コールドシャットダウンを実行します。コミットされていないトランザクションは失われませんが、クラッシュリカバリ操作によって次回の起動時間が長くなります。

低速シャットダウンには数分間かかる可能性があり、大量のデータがバッファーに存在する極端なケースでは、数時間かかる可能性もあります。MySQL のメジャーリリース間でアップグレードまたはダウングレードを行う前には、アップグレードプロセスによってファイル形式が更新される場合に備えて、すべてのデータファイルが完全に準備されるように、低速シャットダウン技術を使用してください。

データが破損するリスクがある場合に、完全な最速のシャットダウンを行うには、緊急事態またはトラブルシューティングの状況で innodb_fast_shutdown=2 を使用してください。

- innodb_file_format

コマンド行形式	--innodb-file-format=#
システム変数	innodb_file_format
スコープ	グローバル
動的	はい
型	文字列
デフォルト	Antelope
有効な値	Antelope Barracuda

新しい InnoDB テーブルで使用されるファイル形式です。現在は、Antelope および Barracuda がサポートされています。これは、独自のテーブルスペースを持つテーブルにのみ適用されるため、これを有効にするには、innodb_file_per_table が有効になっている必要があります。テーブルの圧縮などの特定の InnoDB 機能を使用するには、Barracuda ファイル形式が必要です。

InnoDB のテーブル (ALGORITHM=COPY) を再作成する ALTER TABLE 操作では、現在の innodb_file_format 設定が使用される (前述の状況が適用される) ことに注意してください。

- innodb_file_format_check

コマンド行形式	--innodb-file-format-check=#
システム変数	innodb_file_format_check
スコープ	グローバル
動的	いいえ
型	ブール
デフォルト	ON

この変数をサーバーの起動時に 1 または 0 に設定すると、InnoDB がシステムテーブルスペースのファイル形式タグ (Antelope や Barracuda など) をチェックするかどうかを有効または無効にすることができます。チェックされたタグが最新バージョンの InnoDB でサポートされているよりも大きい場合は、エラーが発生し、InnoDB は起動されません。このタグの方が大きくない場合は、InnoDB によって innodb_file_format_max の値がファイル形式タグに設定されます。

> **注記**
>
> デフォルト値が ON または OFF と表示されることがあるにもかかわらず、このオプションを構成ファイルまたはコマンド行でオンまたはオフに切り替えるには、常に数値 1 または 0 を使用します。

- innodb_file_format_max

コマンド行形式	--innodb-file-format-max=#
システム変数	innodb_file_format_max

スコープ	グローバル
動的	はい
型	文字列
デフォルト	Antelope
有効な値	Antelope Barracuda

サーバーの起動時に InnoDB によって、この変数の値がシステムテーブルスペースのファイル形式タグ (Antelope や Barracuda など) に設定されます。サーバーで「大きい」ファイル形式のテーブルが作成されたり、開かれたりすると、innodb_file_format_max の値がそのファイル形式に設定されます。

- innodb_file_per_table

コマンド行形式	--innodb-file-per-table
システム変数	innodb_file_per_table
スコープ	グローバル
動的	はい
型	ブール
デフォルト (≥ 5.6.6)	ON
デフォルト (≤ 5.6.5)	OFF

innodb_file_per_table が有効になっている (5.6.6 以上のデフォルト) 場合、InnoDB では、新たに作成された各テーブルのデータおよびインデックスがシステムテーブルスペースではなく、個別の .ibd ファイルに格納されます。これらの InnoDB テーブル用のストレージは、テーブルが削除されたり、切り捨てられたりすると再利用されます。このように設定すると、テーブルの圧縮などのその他のいくつかの InnoDB 機能が有効になります。このような機能、および file-per-table テーブルスペースを使用する利点および欠点についての詳細は、セクション14.5.2「InnoDB File-Per-Table モード」を参照してください。

ALTER TABLE でテーブルが再作成されるケース (ALGORITHM=COPY) では、innodb_file_per_table を有効にすることは、ALTER TABLE 操作によって InnoDB テーブルがシステムテーブルスペースから個々の .ibd ファイルに移動されることも意味することに注意してください。

innodb_file_per_table を無効にすると、InnoDB ではすべてのテーブルおよびインデックス用のデータが、システムテーブルスペースを構成する ibdata ファイルに格納されます。このように設定すると、DROP TABLE や TRUNCATE TABLE などの操作で、ファイルシステム操作のパフォーマンスオーバーヘッドが削減されます。これは、ストレージデバイス全体が MySQL データ専用になっているサーバー環境に最適です。システムテーブルスペースは縮小されず、インスタンス内のすべてのデータベースにわたって共有されないため、innodb_file_per_table=OFF のときは、領域が制約されているシステムで一時データを大量にロードすることは回避してください。このような場合は、領域を再利用するためにインスタンス全体を削除できるように、個別のインスタンスを設定します。

MySQL 5.6.6 の時点では、デフォルトで innodb_file_per_table が有効になっています。それよりも前では、無効になっています。MySQL 5.5 または 5.1 との下位互換性が懸念事項となっている場合は、これを無効にすることを検討してください。これにより、ALTER TABLE によって InnoDB テーブルがシステムテーブルスペースから個々の .ibd ファイルに移動することが回避されます。

innodb_file_per_table は動的であり、SET GLOBAL を使用して ON または OFF に設定できます。このパラメータは、MySQL 構成ファイル (my.cnf または my.ini) でも設定できますが、このためにはサーバーをシャットダウンしてから再起動する必要があります。

このパラメータの値を動的に変更するには、SUPER 権限が必要です。動的に変更すると、すべての接続の操作がすぐに影響を受けます。

- innodb_flush_log_at_timeout

導入	5.6.6
システム変数	innodb_flush_log_at_timeout
スコープ	グローバル
動的	はい

型	数値
デフォルト	1
最小値	1
最大値	2700

ログを N 秒ごとに書き込んで、フラッシュします。innodb_flush_log_at_timeout は MySQL 5.6.6 で導入されました。フラッシュを減らし、バイナリロググループのコミット時のパフォーマンスへの影響を回避するために、フラッシュ間のタイムアウト期間を長くすることができます。MySQL 5.6.6 よりも前では、フラッシュの頻度は 1 秒ごとに 1 回でした。innodb_flush_log_at_timeout のデフォルト設定も 1 秒ごとに 1 回です。

- innodb_flush_log_at_trx_commit

コマンド行形式	--innodb-flush-log-at-trx-commit[=#]
システム変数	innodb_flush_log_at_trx_commit
スコープ	グローバル
動的	はい
型	列挙
デフォルト	1
有効な値	0 1 2

コミット操作に対する厳密な ACID コンプライアンスと、コミット関連の I/O 操作が再編成およびバッチ処理されるときに実現可能な高いパフォーマンスとの間のバランスを制御します。デフォルト値を変更するとパフォーマンスを改善できますが、クラッシュ時にトランザクションが最大で 1 秒間失われる可能性があります。

- 完全に ACID コンプライアンスに従うには、デフォルト値の 1 を使用する必要があります。この値を使用すると、トランザクションコミットのたびに、InnoDB のログバッファーの内容がログファイルに書き込まれ、ログファイルがディスクにフラッシュされます。
- 値を 0 にすると、約 1 秒ごとに 1 回、InnoDB のログバッファーの内容がログファイルに書き込まれ、ログファイルがディスクにフラッシュされます。ログバッファーからログファイルに書き込みは、トランザクションコミット時には実行されません。プロセスのスケジューリングの問題が原因で、1 秒ごとに 1 回のフラッシュが毎秒発生する 100% の保証はありません。ディスク操作へのフラッシュは約 1 秒ごとに 1 回しか発生しないため、任意の mysqld プロセスがクラッシュすると、トランザクションが最大で 1 秒間失われる可能性があります。
- 値を 2 にすると、トランザクションコミットのたびに、InnoDB のログバッファーの内容がログファイルに書き込まれ、約 1 秒ごとに 1 回ログファイルがディスクにフラッシュされます。プロセスのスケジューリングの問題が原因で、1 秒ごとに 1 回のフラッシュが毎秒発生する 100% の保証はありません。ディスク操作へのフラッシュは約 1 秒ごとに 1 回しか発生しないため、オペレーティングシステムがクラッシュしたり、停電が発生したりすると、トランザクションが最大で 1 秒間失われる可能性があります。
- MySQL 5.6.6 の時点では、InnoDB でのログフラッシュの頻度が innodb_flush_log_at_timeout で制御されます。これにより、ログフラッシュの頻度を N 秒間に設定できます (ここで、N は 1 ... 2700 で、デフォルト値は 1 です)。ただし、任意の mysqld プロセスがクラッシュすると、トランザクションが最大で N 秒間消失する可能性があります。
- innodb_flush_log_at_trx_commit の設定とは関係なく、DDL の変更やその他の InnoDB アクティビティーによって、InnoDB のログがフラッシュされます。
- InnoDB のクラッシュリカバリは、innodb_flush_log_at_trx_commit の設定に関係なく機能します。トランザクションは完全に適用されるか、完全に消去されるかのいずれかです。

トランザクションで InnoDB が使用されるレプリケーションセットアップの持続性および一貫性を保つ場合:

- バイナリロギングが有効になっている場合は、sync_binlog=1 を設定します。

- 常に innodb_flush_log_at_trx_commit=1 を設定します。

> **注意**
>
> 多くのオペレーティングシステムや一部のディスクハードウェアは、ディスクへのフラッシュ操作を行なったと欺きます。フラッシュが行われていなくても、行われたと mysqld に通知される可能性があります。そのため、1 を設定してもトランザクションの持続性は保証されず、最悪のケースでは、停電によって InnoDB のデータが破損する可能性もあります。バッテリーバックアップのディスクキャッシュを SCSI ディスクコントローラ内やディスク自体で使用すると、ファイルフラッシュの速度が上がり、操作が安全になります。ハードウェアキャッシュ内でディスク書き込みのキャッシュを無効にするために、Unix コマンド hdparm を使用してみたり、ハードウェアベンダー固有のその他のコマンドを使用したりすることもできます。

- innodb_flush_method

コマンド行形式	--innodb-flush-method=name
システム変数	innodb_flush_method
スコープ	グローバル
動的	いいえ
型	文字列
デフォルト (Windows)	async_unbuffered
デフォルト (Unix)	fsync
有効な値 (Unix, ≥ 5.6.7)	fsync O_DSYNC O_DIRECT O_DIRECT_NO_FSYNC
有効な値 (Unix, ≤ 5.6.6)	fsync O_DSYNC O_DIRECT

InnoDB のデータファイルおよびログファイルにデータをフラッシュする際に使用される方法を定義します。これにより、I/O スループットが影響を受ける可能性があります。この変数は、Unix および Linux システムでのみ構成可能です。Windows システムでは、フラッシュ方法は常に async_unbuffered であり、変更できません。

innodb_flush_method オプションの内容は、次のとおりです。

- fsync: InnoDB は fsync() システムコールを使用して、データファイルとログファイルの両方をフラッシュします。fsync はデフォルト設定です。
- O_DSYNC: InnoDB は、O_SYNC を使用してログファイルを開いてフラッシュし、fsync() を使用してデータファイルをフラッシュします。さまざまな種類の Unix で問題が発生しているため、InnoDB では直接 O_DSYNC が使用されません。
- O_DIRECT: InnoDB は、O_DIRECT (Solaris では directio()) を使用してデータファイルを開き、fsync() を使用してデータファイルとログファイルの両方をフラッシュします。このオプションは、一部の GNU/Linux バージョン、FreeBSD、および Solaris で使用可能です。
- O_DIRECT_NO_FSYNC: InnoDB は、I/O のフラッシュ時に O_DIRECT を使用しますが、後続の fsync() システムコールはスキップします。この設定は、一部のタイプのファイルシステムには適していますが、その他には適していません。たとえば、XFS には適していません。たとえば、使用中のファイルシステムですべてのファイルメタデータを保持するために、fsync() が必要であるのかどうかが不明な場合は、代わりに O_DIRECT を使用してください。このオプションは MySQL 5.6.7 (Bug #11754304、Bug #45892) で導入されました。

各設定によるパフォーマンスへの影響度は、ハードウェア構成およびワークロードによって異なります。使用する設定を決定したり、デフォルト設定のままにするかどうかを決定したりするには、特定の構成でベンチ

マークを実施します。設定ごとに fsync() 呼び出しの全体数を確認するには、Innodb_data_fsyncs ステータス変数を調査します。ワークロードに読み取り操作と書き込み操作を混在させると、一部の設定での実行が影響を受ける可能性があります。たとえば、ハードウェア RAID コントローラおよびバッテリーでバックアップされる書き込みキャッシュが搭載されたシステムでは、InnoDB のバッファープールとオペレーティングシステムのファイルシステムキャッシュ間での二重バッファリングを回避する際に、O_DIRECT が役立つことがあります。InnoDB のデータファイルとログファイルが SAN 上に配置されている一部のシステムでは、大部分の SELECT ステートメントを含む読み取り負荷の高いワークロードで、デフォルト値または O_DSYNC の速度が速くなる可能性があります。このパラメータは、必ず、本番環境が反映されたハードウェアおよびワークロードでテストしてください。一般的な I/O チューニングのアドバイスについては、セクション8.5.7「InnoDB ディスク I/O の最適化」を参照してください。

- innodb_flush_neighbors

コマンド行形式	--innodb-flush-neighbors
導入	5.6.3
システム変数	innodb_flush_neighbors
スコープ	グローバル
動的	はい
型	列挙
デフォルト	1
有効な値	0 1 2

InnoDB のバッファープールからページをフラッシュすると、同じエクステント内のその他のダーティーページもフラッシュされるかどうかを指定します。

- デフォルト値の 1 では、バッファープールから同じエクステント内の連続するダーティーページがフラッシュされます。
- 0 を設定すると、innodb_flush_neighbors がオフになり、その他のダーティーページはバッファープールからフラッシュされません。
- 2 を設定すると、同じエクステント内のダーティーページがバッファープールからフラッシュされます。

テーブルデータが従来の HDD ストレージデバイスに格納されている場合は、1 回の操作でこのような隣接ページをフラッシュすると、さまざまな時間に個々のページをフラッシュする場合と比較して、(主にディスクシーク操作の) I/O オーバーヘッドが削減されます。テーブルデータが SSD 上に格納されている場合は、シーク時間が重要な要素ではないため、この設定をオフにすれば、書き込み操作を分散できます。一般的な I/O チューニングのアドバイスについては、セクション8.5.7「InnoDB ディスク I/O の最適化」を参照してください。

- innodb_flushing_avg_loops

コマンド行形式	--innodb-flushing-avg-loops=#
導入	5.6.6
システム変数	innodb_flushing_avg_loops
スコープ	グローバル
動的	はい
型	数値
デフォルト	30
最小値	1
最大値	1000

InnoDB が以前に計算されたフラッシュ状態のスナップショットを保持する繰り返しの数です。これにより、適応型フラッシュがワークロードの変更に対応する速度が制御されます。この値を大きくすると、ワークロードが変化するにつれて、フラッシュ操作の速度が円滑かつ徐々に変化します。この値を小さくすると、適応型フ

ラッシュがワークロードの変化にすばやく適応します。これにより、ワークロードが突然に増減した場合に、フラッシュアクティビティーが急増する可能性があります。

- innodb_force_load_corrupted

コマンド行形式	--innodb-force-load-corrupted
導入	5.6.3
システム変数	innodb_force_load_corrupted
スコープ	グローバル
動的	いいえ
型	ブール
デフォルト	OFF

InnoDB の起動時に、破損マークが付けられたテーブルをロードできます。トラブルシューティング時に、何も対処しなければアクセスできないデータをリカバリする際にのみ使用してください。トラブルシューティングが完了したら、この設定をオフに戻して、サーバーを再起動します。

- innodb_force_recovery

コマンド行形式	--innodb-force-recovery=#
システム変数	innodb_force_recovery
スコープ	グローバル
動的	いいえ
型	数値
デフォルト	0
最小値	0
最大値	6

クラッシュリカバリモードです。一般に、重大なトラブルシューティングの状況でのみ変更されます。指定可能な値は 0 から 6 までです。これらの値の意味および innodb_force_recovery に関する重要な情報については、セクション14.19.2「InnoDB のリカバリの強制的な実行」を参照してください。

> **警告**
>
> 緊急状況でのみ、この変数を 0 よりも大きい値に設定してください。これにより、InnoDB を起動し、テーブルをダンプできるようになります。安全対策として、innodb_force_recovery を 0 よりも大きくすると、InnoDB で INSERT、UPDATE、または DELETE 操作が回避されます。また、5.6.15 の時点では、innodb_force_recovery の設定を 4 よりも大きくすると、InnoDB が読み取り専用モードになります。
>
> このような制約のために、--relay-log-info-repository=TABLE や --master-info-repository=TABLE などのレプリケーションオプションによって InnoDB 内のテーブルに情報が格納されると、レプリケーション管理コマンドに失敗し、エラーが発生する可能性があります。

- innodb_ft_aux_table

導入	5.6.4
システム変数	innodb_ft_aux_table
スコープ	グローバル
動的	はい

型	文字列

FULLTEXT インデックスを含む InnoDB テーブルの修飾名を指定します。この変数は診断のために使用され、実行時にのみ設定できます。例:

```
mysql> set global innodb_ft_aux_table = 'test/t1';
```

この変数を起動時に設定しようとすると、「mysqld: option '--innodb-ft-aux-table' cannot take an argument」エラーが発生し、起動が中止されます。この変数を db_name/table_name 形式の名前に設定すると、INFORMATION_SCHEMA テーブル INNODB_FT_INDEX_TABLE、INNODB_FT_INDEX_CACHE、INNODB_FT_CONFIG、INNODB_FT_DELETED、および INNODB_FT_BEING_DELETED に、指定されたテーブルの検索インデックスに関する情報が表示されます。

- innodb_ft_cache_size

コマンド行形式	--innodb-ft-cache-size=#
導入	5.6.4
システム変数	innodb_ft_cache_size
スコープ	グローバル
動的	いいえ
型	数値
デフォルト (≥ 5.6.10)	8000000
デフォルト (≥ 5.6.4, ≤ 5.6.9)	32000000
最小値	1600000
最大値	80000000

InnoDB の FULLTEXT 検索インデックスのキャッシュ用に割り当てられたメモリー量 (バイト単位) です。InnoDB の FULLTEXT インデックスの作成時に、この量の解析済みドキュメントがメモリー内に保持されます。innodb_ft_cache_size のサイズ制限に達すると、インデックスの挿入および更新のみがディスクにコミットされます。innodb_ft_cache_size では、キャッシュサイズがテーブルごとに定義されます。すべてのテーブルにグローバルな制限を設定する方法については、innodb_ft_total_cache_size を参照してください。

- innodb_ft_enable_diag_print

コマンド行形式	--innodb-ft-enable-diag-print=#
導入	5.6.4
システム変数	innodb_ft_enable_diag_print
スコープ	グローバル
動的	はい
型	ブール
デフォルト (≥ 5.6.7)	OFF
デフォルト (≤ 5.6.6)	ON

追加の全文検索 (FTS) 診断の出力を有効にするかどうかを指定します。このオプションは、主に高度な FTS デバッグのために使用され、大部分のユーザーには関心がないものです。出力はエラーログに記録され、次のような情報が含まれています。

- FTS インデックス同期の進行状況 (FTS キャッシュ制限に達したとき)。例:

```
FTS SYNC for table test, deleted count: 100 size: 10000 bytes
SYNC words: 100
```

- FTS 最適化の進行状況。例:

```
FTS start optimize test
FTS_OPTIMIZE: optimize "mysql"
FTS_OPTIMIZE: processed "mysql"
```

- FTS インデックス構築の進行状況。例:

```
Number of doc processed: 1000
```

- FTS クエリーでは、クエリー解析のツリー、単語の重み、クエリーの処理時間、およびメモリーの使用状況が出力されます。例:

```
FTS Search Processing time: 1 secs: 100 millisec: row(s) 10000
Full Search Memory: 245666 (bytes),  Row: 10000
```

- innodb_ft_enable_stopword

コマンド行形式	--innodb-ft-enable-stopword=#
導入	5.6.4
システム変数	innodb_ft_enable_stopword
スコープ	グローバル
動的	はい
型	ブール
デフォルト	ON

インデックスの作成時に、一連のストップワードが InnoDB の FULLTEXT インデックスに関連付けられることを指定します。innodb_ft_user_stopword_table オプションが設定されている場合は、そのテーブルからストップワードが取得されます。そうでなければ、innodb_ft_server_stopword_table オプションが設定されている場合は、そのテーブルからストップワードが取得されます。それ以外の場合は、組み込みのデフォルトストップワードセットが使用されます。

- innodb_ft_max_token_size

コマンド行形式	--innodb-ft-max-token-size=#
導入	5.6.4
システム変数	innodb_ft_max_token_size
スコープ	グローバル
動的	いいえ
型	数値
デフォルト	84
最小値	10
最大値	252

InnoDB の FULLTEXT インデックスに格納されている単語の最大長です。この値に制限を設定すると、実在の単語ではなく、検索語句になる可能性の低い英字の任意のコレクションや長いキーワードが省略されることで、インデックスのサイズが削減されるため、クエリーの速度が上がります。

- innodb_ft_min_token_size

コマンド行形式	--innodb-ft-min-token-size=#
導入	5.6.4
システム変数	innodb_ft_min_token_size
スコープ	グローバル
動的	いいえ
型	数値
デフォルト	3
最小値	0
最大値	16

InnoDB の FULLTEXT インデックスに格納されている単語の最小長です。この値を大きくすると、検索のコンテキストで重要となる可能性の低い一般的な単語 (「a」 や 「to」 などの英単語) が省略されることで、イン

デックスのサイズが削減されるため、クエリーの速度が上がります。内容で CJK (中国語、日本語、韓国語) 文字セットが使用されている場合は、値 1 を指定します。

- innodb_ft_num_word_optimize

コマンド行形式	--innodb-ft-num-word-optimize=#
導入	5.6.4
システム変数	innodb_ft_num_word_optimize
スコープ	グローバル
動的	はい
型	数値
デフォルト	2000

InnoDB の FULLTEXT インデックスでの各 OPTIMIZE TABLE 操作時に処理される単語数です。全文検索インデックスを含むテーブルへの一括挿入または一括更新操作では、すべての変更を組み込むために大量のインデックスのメンテナンスが必要となる可能性があるため、それぞれが最後に終了した場所から再開する一連の OPTIMIZE TABLE ステートメントを実行するとよいでしょう。

- innodb_ft_result_cache_limit

コマンド行形式	--innodb-ft-result-cache-limit=#
導入	5.6.13
システム変数	innodb_ft_result_cache_limit
スコープ	グローバル
動的	はい
型	数値
デフォルト	2000000000
最小値	1000000
最大値 (Windows, ≥ 5.6.13, ≤ 5.6.16)	2**32-1
最大値 (Unix, 64 ビットプラットフォーム, ≥ 5.6.13, ≤ 5.6.16)	2**64-1
最大値 (Unix, 32 ビットプラットフォーム, ≥ 5.6.13, ≤ 5.6.16)	2**32-1
最大値 (≥ 5.6.17)	2**32-1

FTS クエリーごとまたはスレッドごとに、(バイト単位で定義された) InnoDB の FULLTEXT 検索 (FTS) クエリー結果のキャッシュ制限です。中間および最終の InnoDB FTS クエリー結果は、メモリー内で処理されます。InnoDB の FTS クエリー結果が非常に大きい (何百万や何億もの行数など) 場合に、過剰なメモリー消費を回避するには、innodb_ft_result_cache_limit を使用して InnoDB の FTS クエリー結果のキャッシュにサイズ制限を課します。メモリーは、FTS クエリーの処理時に必要に応じて割り当てられます。結果のキャッシュサイズ制限に達すると、クエリーで最大限に許可されるメモリー量を超えたことを示すエラーが返されます。

MySQL 5.6.17 の時点では、すべてのプラットフォームタイプおよびプラットフォームビットサイズに対応した innodb_ft_result_cache_limit の最大値は、2**32-1 です。Bug #71554。

- innodb_ft_server_stopword_table

コマンド行形式	--innodb-ft-server-stopword-table=db_name/table_name
導入	5.6.4
システム変数	innodb_ft_server_stopword_table
スコープ	グローバル
動的	はい
型	文字列

デフォルト	NULL

このオプションは、すべての InnoDB テーブルに対応した独自の InnoDB の FULLTEXT インデックスストップワードリストを指定する際に使用されます。特定の InnoDB テーブルに独自のストップワードリストを構成するには、innodb_ft_user_stopword_table を使用します。

db_name/table_name の形式で、innodb_ft_server_stopword_table をストップワードリストを含むテーブルの名前に設定します。

innodb_ft_server_stopword_table を構成する前に、ストップワードテーブルが存在する必要があります。FULLTEXT インデックスを作成する前に、innodb_ft_enable_stopword を有効にして、innodb_ft_server_stopword_table オプションを構成する必要あります。

ストップワードテーブルは、VALUE という名前の単一の VARCHAR カラムを含む InnoDB テーブルにする必要があります。

詳細は、セクション12.9.4「全文ストップワード」を参照してください。

- innodb_ft_sort_pll_degree

コマンド行形式	--innodb-ft-sort-pll-degree=#
導入	5.6.4
システム変数	innodb_ft_sort_pll_degree
スコープ	グローバル
動的	いいえ
型	数値
デフォルト	2
最小値	1
最大値	32

検索インデックスの構築時に、InnoDB の FULLTEXT インデックス内のテキストのインデックス作成およびトークン化を行う際に、並列して使用されるスレッド数です。使用法の追加情報については、innodb_sort_buffer_size を参照してください。

- innodb_ft_total_cache_size

コマンド行形式	--innodb-ft-total-cache-size=#
導入	5.6.13
システム変数	innodb_ft_total_cache_size
スコープ	グローバル
動的	いいえ
型	数値
デフォルト	640000000
最小値	32000000
最大値	1600000000

すべてのテーブルに対応した InnoDB の FULLTEXT 検索インデックスキャッシュ用に割り当てられた合計メモリー量 (バイト単位) です。それぞれが全文検索インデックスを持つ多数のテーブルを作成すると、使用可能なメモリーの大部分が消費される可能性があります。innodb_ft_total_cache_size では、過剰なメモリー消費を回避する際に役立つ、すべての全文検索インデックスに対応したグローバルなメモリー制限が定義されます。インデックス操作でグローバルな制限に達すると、強制的な同期がトリガーされます。

- innodb_ft_user_stopword_table

コマンド行形式	--innodb-ft-user-stopword-table=db_name/table_name
導入	5.6.4
システム変数	innodb_ft_user_stopword_table

スコープ	グローバル、セッション
動的	はい
型	文字列
デフォルト	NULL

このオプションは、特定のテーブルに独自の InnoDB の FULLTEXT インデックスストップワードリストを指定する際に使用されます。すべての InnoDB テーブル用に独自のストップワードリストを構成するには、innodb_ft_server_stopword_table を使用します。

db_name/table_name の形式で、innodb_ft_user_stopword_table をストップワードリストを含むテーブルの名前に設定します。

innodb_ft_user_stopword_table を構成する前に、ストップワードテーブルが存在する必要があります。FULLTEXT インデックスを作成する前に、innodb_ft_enable_stopword を有効にして、innodb_ft_user_stopword_table オプションを構成する必要あります。

ストップワードテーブルは、VALUE という名前の単一の VARCHAR カラムを含む InnoDB テーブルにする必要があります。

詳細は、セクション12.9.4「全文ストップワード」を参照してください。

- innodb_io_capacity

コマンド行形式	--innodb-io-capacity=#
システム変数	innodb_io_capacity
スコープ	グローバル
動的	はい
型	数値
デフォルト	200
最小値	100
最大値 (64 ビットプラットフォーム)	2**64-1
最大値 (32 ビットプラットフォーム)	2**32-1

innodb_io_capacity パラメータは、InnoDB バックグラウンドタスクで実行される I/O アクティビティー (バッファープールからのページのフラッシュや挿入バッファーからのデータのマージなど) に上限を設定します。デフォルト値は 200 です。高い I/O レートを処理できる高負荷のシステムでは、サーバーの起動時に大きい値を設定すると、サーバーが高いレートの行変更に関連付けられたバックグラウンドメンテナンス作業を処理できるようになります。

innodb_io_capacity の制限は、すべてのバッファープールインスタンスに対する合計の制限です。ダーティーページがフラッシュされるとき、innodb_io_capacity 制限は、バッファープールインスタンス間で均等に分割されます。

個別の 5400 RPM または 7200 RPM ドライブが搭載されたシステムでは、元のデフォルトの 100 まで値を小さくするとよいでしょう。

このパラメータはほぼ、システムが 1 秒あたりに実行できる I/O 操作の数に設定するようにしてください。理想的には、この設定はできるだけ小さく保ちます。ただし、これらのバックグラウンドアクティビティーに

遅延が発生するほど小さくしないでください。値が大きすぎる場合は、データがバッファープールおよび挿入バッファーから瞬時に削除されるため、キャッシュを使用する重要な利点が得られません。

この値は、約 100 IOPS を実行できる旧世代のディスクドライブで実現可能な IOPS (I/O Operations Per Second) の推定比率を表します。現在のデフォルトの 200 は、最新のストレージデバイスがさらに高い I/O レートを処理できることを反映しています。

一般に、InnoDB の I/O で使用されるドライブ (特に、高い数値の IOPS を処理できる高速ドライブ) の数に応じて、値を大きくすることができます。たとえば、InnoDB 用に複数のディスクまたはソリッドステートディスクを使用するシステムでは、このパラメータを制御する機能の利点が得られる可能性が高くなります。

非常に大きい数値も指定できますが、実際にはこのような大きな値にすると、利点を得られるとしてもわずかです。たとえば、100 万は非常に大きい値と考えられます。

innodb_io_capacity の値は、innodb_io_capacity_max で定義された最大値まで、100 以上の任意の数値に設定できます。デフォルト値は 200 です。このパラメータの値は MySQL オプションファイル (my.cnf または my.ini) で設定するか、あるいは SET GLOBAL コマンド (これには SUPER 権限が必要です) で動的に変更できます。

このオプションに関する詳細なガイドラインについては、セクション14.13.8「InnoDB マススレッドの I/O レートの構成」を参照してください。InnoDB の I/O パフォーマンスに関する一般的な情報については、セクション8.5.7「InnoDB ディスク I/O の最適化」を参照してください。

- innodb_io_capacity_max

コマンド行形式	--innodb-io-capacity-max=#
導入	5.6.6
システム変数	innodb_io_capacity_max
スコープ	グローバル
動的	はい
型	数値
デフォルト (Windows, 64 ビットプラットフォーム)	2000
デフォルト (Unix, 64 ビットプラットフォーム)	see description
デフォルト (32 ビットプラットフォーム)	see description
最小値	2000
最大値 (Windows, 64 ビットプラットフォーム)	2**32-1
最大値 (Unix, 64 ビットプラットフォーム)	2**64-1
最大値 (32 ビットプラットフォーム)	2**32-1

緊急時に innodb_io_capacity の設定を拡張する際に、InnoDB で許可されている上限です。起動時に innodb_io_capacity の設定を指定し、innodb_io_capacity_max に値を指定しない場合は、innodb_io_capacity_max のデフォルト値が innodb_io_capacity の 2 倍となり、下限が 2000 となります。また、2000 は初期のデフォルト innodb_io_capacity_max 構成値です。

innodb_io_capacity_max 設定は、すべてのバッファープールインスタンスに対する合計の制限です。

MySQL 5.6 開発中の短期間は、この変数は innodb_max_io_capacity と呼ばれていました。MySQL 5.6.7 では、innodb_io_capacity オプションとの関係を強調するために、innodb_io_capacity_max という名前に変更されました。

- innodb_large_prefix

コマンド行形式	--innodb-large-prefix
導入	5.6.3
システム変数	innodb_large_prefix

スコープ	グローバル
動的	はい
型	ブール
デフォルト	OFF

このオプションを有効にすると、DYNAMIC および COMPRESSED 行フォーマットを使用する InnoDB テーブルで、767 バイトよりも長い (最大で 3072 バイトの) インデックスキープリフィクスが許可されます。(このようなテーブルの作成には、innodb_file_format=barracuda および innodb_file_per_table=true のオプション値も必要になります。)さまざまな設定でインデックスキープリフィクスに関連付けられた関連性の最大値については、セクション14.6.7「InnoDB テーブル上の制限」を参照してください。

REDUNDANT および COMPACT 行フォーマットを使用したテーブルでは、このオプションによってキープリフィクスの長さは影響を受けません。

- innodb_lock_wait_timeout

コマンド行形式	--innodb-lock-wait-timeout=#
システム変数	innodb_lock_wait_timeout
スコープ	グローバル、セッション
動的	はい
型	数値
デフォルト	50
最小値	1
最大値	1073741824

行ロックが解除されるまで InnoDB トランザクションが待機する時間の長さ (秒単位) です。デフォルト値は 50 秒です。別の InnoDB トランザクションでロックされている行へのアクセスを試みるトランザクションは、行への書き込みアクセスを最大でこの秒数間待機してから、次のエラーを発行します。

```
ERROR 1205 (HY000): Lock wait timeout exceeded; try restarting transaction
```

ロック待機のタイムアウトが発生すると、(トランザクション全体ではなく) 現在のステートメントがロールバックされます。トランザクション全体をロールバックするには、--innodb_rollback_on_timeout オプションを付けてサーバーを起動します。セクション14.19.4「InnoDB のエラー処理」も参照してください。

高度にインタラクティブなアプリケーションまたは OLTP システムでは、ユーザーのフィードバックをすばやく表示したり、あとで処理するために更新をキューに入れたりするために、この値を小さくするとよいでしょう。長時間実行されるバックエンド操作 (その他の大規模な挿入操作や更新操作が完了するまで待機するデータウェアハウスでの変換ステップなど) では、この値を大きくするとよいでしょう。

innodb_lock_wait_timeout は InnoDB の行ロックにのみ適用されます。MySQL のテーブルロックは InnoDB 内部では発生せず、このタイムアウトはテーブルロックの待機には適用されません。

デッドロックは InnoDB によってすぐに検出され、デッドロックになったトランザクションのいずれかがロールバックされるため、デッドロックにはロック待機のタイムアウト値が適用されません。

innodb_lock_wait_timeout は、実行時に SET GLOBAL または SET SESSION ステートメントとともに設定できます。GLOBAL 値を変更するには、SUPER 権限が必要です。これを変更すると、それ以降に接続するすべてのクライアントの操作が影響を受けます。任意のクライアントが innodb_lock_wait_timeout の SESSION 設定を変更でき、そのクライアントのみが影響を受けます。

- innodb_locks_unsafe_for_binlog

コマンド行形式	--innodb-locks-unsafe-for-binlog
非推奨	5.6.3
システム変数	innodb_locks_unsafe_for_binlog
スコープ	グローバル
動的	いいえ
型	ブール

デフォルト	OFF

この変数によって、InnoDB が検索およびインデックススキャンでギャップロックを使用する方法が影響を受けます。MySQL 5.6.3 の時点では、innodb_locks_unsafe_for_binlog は非推奨となり、今後の MySQL リリースで削除される予定です。

通常、InnoDB では、インデックス行ロックとギャップロックを組み合わせた「ネクストキーロック」と呼ばれるアルゴリズムが使用されます。InnoDB は、テーブルインデックスを検索またはスキャンするときに、生成されたインデックスレコード上に共有ロックまたは排他ロックを設定するという方法で、行レベルロックを実行します。したがって、行レベルロックは、実際にはインデックスレコードロックです。さらに、あるインデックスレコードに対するネクストキーロックによって、そのインデックスレコードの前の「ギャップ」も影響を受けます。つまり、ネクストキーロックは、インデックスレコードロックと、そのインデックスレコードの前のギャップに対するギャップロックとを組み合わせたものです。あるセッションがインデックス内のレコード R 上に共有ロックまたは排他ロックを持っている場合は、別のセッションがインデックスの順番で R の直前にあるギャップに新しいインデックスレコードを挿入できません。セクション14.2.6「InnoDB のレコード、ギャップ、およびネクストキーロック」を参照してください。

innodb_locks_unsafe_for_binlog の値はデフォルトで 0 (無効) になっていますが、これは、ギャップロックが有効であることを意味します。InnoDB はネクストキーロックを使用して、検索およびインデックススキャンを実行します。この変数を有効にするには、値を 1 に設定します。これにより、ギャップロックが無効になります。InnoDB はインデックスレコードロックのみを使用して、検索およびインデックススキャンを実行します。

innodb_locks_unsafe_for_binlog を有効にしても、外部キー制約チェックや重複キーチェックでのギャップロックの使用は無効になりません。

innodb_locks_unsafe_for_binlog を有効にした場合の影響は、トランザクション分離レベルを READ COMMITTED に設定した場合の影響に似ていますが、同じではありません。

- innodb_locks_unsafe_for_binlog を有効にすることはグローバルな設定であるため、すべてのセッションが影響を受けます。その一方で、分離レベルは、すべてのセッションに対してグローバルに設定することも、セッションごとに個別に設定することもできます。
- innodb_locks_unsafe_for_binlog はサーバー起動時にしか設定できないのに対して、分離レベルは起動時に設定することも、実行時に変更することもできます。

したがって、READ COMMITTED では innodb_locks_unsafe_for_binlog よりも細かく柔軟な制御が提供されます。ギャップロックに対する分離レベルの影響に関する追加の詳細については、セクション13.3.6「SET TRANSACTION 構文」を参照してください。

ギャップロックが無効になるとほかのセッションが新しい行をギャップに挿入できるため、innodb_locks_unsafe_for_binlog を有効にすると、ファントムの問題が発生する可能性があります。child テーブルの id カラム上にインデックスがあり、識別子の値が 100 よりも大きいすべての行をテーブルから読み取り、選択された行の一部のカラムをあとで更新するという意図でロックすると仮定します。

```
SELECT * FROM child WHERE id > 100 FOR UPDATE;
```

クエリーでは、id が 100 より大きい最初のレコードからインデックスがスキャンされます。その範囲内のインデックスレコード上に設定されたロックによって、ギャップへの挿入がロックアウトされていない場合は、別のセッションがそのテーブルに新しい行を挿入できます。したがって、同じトランザクション内で同じ SELECT を再度実行すると、クエリーから返された結果セット内に新しい行を見つけることができます。これは、データベースに新しい項目が追加された場合は、InnoDB で直列化可能性が保証されないことも意味します。したがって、innodb_locks_unsafe_for_binlog が有効な場合に InnoDB によって保証される最大の分離レベルは、READ COMMITTED になります。(競合直列化可能性は引き続き保証されます。)ファントムの追加情報については、セクション14.2.7「ネクストキーロックによるファントム問題の回避」を参照してください。

innodb_locks_unsafe_for_binlog を有効にした場合には、次のような影響も発生します。

- UPDATE または DELETE ステートメントでは、InnoDB は更新または削除の対象となる行に対してのみ、ロックを保持します。一致しなかった行のレコードロックは、MySQL による WHERE 条件の評価後に解除されます。これにより、デッドロックの可能性が大幅に低くなりますが、まだ発生する可能性はあります。
- UPDATE ステートメントである行がすでにロックされていた場合、InnoDB は 「半一貫性」 読み取りを実行し、最後にコミットされたバージョンを MySQL に返すため、MySQL はその行が UPDATE の WHERE 条件に一致するかどうかを判断できます。その行が一致した場合 (その行を更新する必要がある場合)、MySQL は

その行を再度読み取り、InnoDB は今度はその行をロックするか、その行のロックが解除されるまで待機します。

次のような例について、このテーブルから検討します。

```
CREATE TABLE t (a INT NOT NULL, b INT) ENGINE = InnoDB;
INSERT INTO t VALUES (1,2),(2,3),(3,2),(4,3),(5,2);
COMMIT;
```

この場合は、テーブルにインデックスが設定されていないため、検索およびインデックススキャンでは、非表示のクラスタ化されたインデックスを使用してレコードのロックが行われます (セクション14.2.13.2「クラスタインデックスとセカンダリインデックス」を参照してください)。

あるクライアントが次のステートメントを使用して、UPDATE を実行すると仮定します。

```
SET autocommit = 0;
UPDATE t SET b = 5 WHERE b = 3;
```

また、2 番目のクライアントが 1 番目のクライアントの実行後に次のステートメントを実行することで、UPDATE を実行すると仮定します。

```
SET autocommit = 0;
UPDATE t SET b = 4 WHERE b = 2;
```

InnoDB は各 UPDATE を実行する際に、まず各行の排他ロックを取得し、次にその行を変更するかどうかを判断します。InnoDB がその行を変更せず、かつ innodb_locks_unsafe_for_binlog が有効な場合は、そのロックが解除されます。それ以外の場合、トランザクションが終了するまで InnoDB はそのロックを保持します。これにより、トランザクション処理が次のような影響を受けます。

innodb_locks_unsafe_for_binlog が無効な場合は次のように、最初の UPDATE は X ロックを取得し、そのいずれも解除しません。

```
x-lock(1,2); retain x-lock
x-lock(2,3); update(2,3) to (2,5); retain x-lock
x-lock(3,2); retain x-lock
x-lock(4,3); update(4,3) to (4,5); retain x-lock
x-lock(5,2); retain x-lock
```

次のように、2 番目の UPDATE は (1 番目の更新がすべての行のロックを保持しているため)、ロックを取得しようとしてもすぐにブロックされ、1 番目の UPDATE がコミットまたはロールバックを実行するまで続行されません。

```
x-lock(1,2); block and wait for first UPDATE to commit or roll back
```

innodb_locks_unsafe_for_binlog が有効な場合は次のように、最初の UPDATE は X ロックを取得したあとに、変更されない行のロックを解除します。

```
x-lock(1,2); unlock(1,2)
x-lock(2,3); update(2,3) to (2,5); retain x-lock
x-lock(3,2); unlock(3,2)
x-lock(4,3); update(4,3) to (4,5); retain x-lock
x-lock(5,2); unlock(5,2)
```

2 番目の UPDATE では次のように、InnoDB は 「半一貫性」 読み取りを行い、最後にコミットされたバージョンを MySQL に返すため、MySQL はその行が UPDATE の WHERE 条件に一致するかどうかを判断できます。

```
x-lock(1,2); update(1,2) to (1,4); retain x-lock
x-lock(2,3); unlock(2,3)
x-lock(3,2); update(3,2) to (3,4); retain x-lock
x-lock(4,3); unlock(4,3)
x-lock(5,2); update(5,2) to (5,4); retain x-lock
```

- innodb_log_buffer_size

コマンド行形式	--innodb-log-buffer-size=#
システム変数	innodb_log_buffer_size
スコープ	グローバル
動的	いいえ
型	数値

デフォルト	8388608
最小値	262144
最大値	4294967295

ディスク上のログファイルに書き込む際に InnoDB で使用されるバッファーのサイズ (バイト単位) です。デフォルトの値は 8M バイトです。ログバッファーを大きくすると、トランザクションがコミットする前にディスクにログを書き込まなくても、大規模なトランザクションを実行できます。したがって、多数の行を更新、挿入、または削除するトランザクションの場合、ログバッファーを大きくすると、ディスク I/O を節約できます。一般的な I/O チューニングのアドバイスについては、セクション8.5.7「InnoDB ディスク I/O の最適化」を参照してください。

- innodb_log_compressed_pages

コマンド行形式	--innodb-log-compressed-pages=#
導入	5.6.11
システム変数	innodb_log_compressed_pages
スコープ	グローバル
動的	はい
型	ブール
デフォルト	ON

再圧縮されたページのイメージが InnoDB の Redo ログに格納されるかどうかを指定します。

この変数は MySQL 5.6.11 で追加されました。

- innodb_log_file_size

コマンド行形式	--innodb-log-file-size=#
システム変数	innodb_log_file_size
スコープ	グローバル
動的	いいえ
型	数値
デフォルト (≥ 5.6.8)	50331648
デフォルト (≤ 5.6.7)	5242880
最小値	1048576
最大値 (≥ 5.6.3)	512GB / innodb_log_files_in_group
最大値 (≤ 5.6.2)	4GB / innodb_log_files_in_group

ロググループ内の各ログファイルのサイズ (バイト単位) です。ログファイルを結合したサイズ (innodb_log_file_size * innodb_log_files_in_group) は、512G バイトよりもわずかに小さい最大値を上回ることができません。たとえば、255G バイトのログファイルのペアを使用すれば、制限に近づくことはできますが、上回ることはできません。デフォルト値は 48M バイトです。適切な値の範囲は、1M バイトから 1/バッファープールの N 番目のサイズまでです。ここで、N はグループ内のログファイルの数です。値を大きくするほど、バッファープール内で必要となるチェックポイントフラッシュアクティビティーの数が少なくなるため、ディスク I/O を節約できます。また、ログファイルを大きくすると、クラッシュリカバリの速度が遅くなります。ただし、MySQL 5.5 以上ではリカバリのパフォーマンスが改善されているため、ログファイルのサイズに対する考慮事項も少なくなります。一般的な I/O チューニングのアドバイスについては、セクション8.5.7「InnoDB ディスク I/O の最適化」を参照してください。

> **重要**
>
> Bug #69477 が原因で、外部に格納された大きな BLOB フィールドに対する Redo ログの書き込みによって、最新のチェックポイントが上書きされる可能性があります。このバグに対処するために MySQL 5.6.20 で導入されたパッチを適用すれば、BLOB で書き込まれる Redo ログのサイズが Redo ログファイルサイズの 10% に制限されます。この制限の結果として、innodb_log_file_size は、テーブルの行で見つかった最大の BLOB データサイズの 10 倍よりも大きい値に、その他の変数の長さフィールド

> (VARCHAR、VARBINARY、および TEXT 型のフィールド) の長さを加えた値に設定されるはずです。
>
> MySQL 5.6.22 では、Redo ログ BLOB の書き込み制限は 合計 Redo ログサイズ (innodb_log_file_size * innodb_log_files_in_group) の 10% に緩められました。(Bug #19498877)

- innodb_log_files_in_group

コマンド行形式	--innodb-log-files-in-group=#
システム変数	innodb_log_files_in_group
スコープ	グローバル
動的	いいえ
型	数値
デフォルト	2
最小値	2
最大値	100

ロググループ内のログファイルの数です。InnoDB はファイルに輪状に書き込みをします。デフォルト (推奨) 値は 2 です。これらのファイルの場所は、innodb_log_group_home_dir で指定されます。ログファイルを結合したサイズ (innodb_log_file_size * innodb_log_files_in_group) は、最大で 512G バイトにすることができます。

- innodb_log_group_home_dir

コマンド行形式	--innodb-log-group-home-dir=path
システム変数	innodb_log_group_home_dir
スコープ	グローバル
動的	いいえ
型	ディレクトリ名

InnoDB の Redo ログファイルへのディレクトリパスです。この数は、innodb_log_files_in_group で指定されます。どの InnoDB ログ変数も指定しない場合は、デフォルトで、MySQL データディレクトリ内に ib_logfile0 および ib_logfile1 という名前の 2 つのファイルが作成されます。これらのサイズは、innodb_log_file_size システム変数のサイズで指定されます。

- innodb_lru_scan_depth

コマンド行形式	--innodb-lru-scan-depth=#
導入	5.6.3
システム変数	innodb_lru_scan_depth
スコープ	グローバル
動的	はい
型	数値
デフォルト	1024
最小値	100
最大値 (64 ビットプラットフォーム)	2**64-1
最大値 (32 ビットプラットフォーム)	2**32-1

InnoDB のバッファープールでのフラッシュ操作のアルゴリズムおよびヒューリスティクスに影響を与えるパラメータです。主に、I/O インテンシブなワークロードを調整するパフォーマンスの専門家が関心を持つものです。バッファープールインスタンスごとに、page_cleaner スレッドがフラッシュするダーティーページを検索する際に、どのくらいの深さまでバッファープール LRU リストをスキャンするのかが指定されます。これは、1 秒ごとに 1 回実行されるバックグラウンド操作です。一般的なワークロードで予備の I/O 容量を持っている場合は、この値を大きくします。書き込みが集中するワークロードで I/O 容量がいっぱいになった場合

は、この値を小さくします (特に大きなバッファープールを持っている場合)。一般的な I/O チューニングのアドバイスについては、セクション8.5.7「InnoDB ディスク I/O の最適化」を参照してください。

- innodb_max_dirty_pages_pct

コマンド行形式	--innodb-max-dirty-pages-pct=#
システム変数	innodb_max_dirty_pages_pct
スコープ	グローバル
動的	はい
型	数値
デフォルト	75
最小値	0
最大値	99

InnoDB は、ダーティーページの割合がこの値を超えないように、バッファープールからデータをフラッシュしようと試みます。0 から 99 までの範囲内の整数を指定します。デフォルト値は 75 です。

innodb_max_dirty_pages_pct 設定は、フラッシュアクティビティーのターゲットを確立します。フラッシュの頻度には影響を与えません。フラッシュの頻度の管理については、セクション14.13.1.2「InnoDB バッファープールのフラッシュの頻度の構成」を参照してください。

この変数に関する追加情報については、セクション14.13.1.6「InnoDB バッファープールのフラッシュのチューニング」を参照してください。一般的な I/O チューニングのアドバイスについては、セクション8.5.7「InnoDB ディスク I/O の最適化」を参照してください。

- innodb_max_dirty_pages_pct_lwm

コマンド行形式	--innodb-max-dirty-pages-pct-lwm=#
導入	5.6.6
システム変数	innodb_max_dirty_pages_pct_lwm
スコープ	グローバル
動的	はい
型	数値
デフォルト	0
最小値	0
最大値	99

ダーティーページの比率を制御するために事前フラッシュが有効になっている場合に、ダーティーページの割合を表す低位境界値です。デフォルトの 0 では、事前フラッシュの動作が完全に無効になります。この変数に関する追加情報については、セクション14.13.1.6「InnoDB バッファープールのフラッシュのチューニング」を参照してください。

- innodb_max_purge_lag

コマンド行形式	--innodb-max-purge-lag=#
システム変数	innodb_max_purge_lag
スコープ	グローバル
動的	はい
型	数値
デフォルト	0
最小値	0

最大値	4294967295

この変数は、パージ操作が遅れたときに、INSERT、UPDATE、および DELETE 操作を遅延させる方法を制御します (セクション14.2.12「InnoDB マルチバージョン」を参照してください)。デフォルト値は 0 (遅延なし) です。

InnoDB トランザクションシステムでは、UPDATE または DELETE 操作で削除のマークが付けられたインデックスレコードを含むトランザクションのリストが保持されます。purge_lag の値は、このリストの長さを表しています。purge_lag が innodb_max_purge_lag を超えると、各 INSERT、UPDATE、および DELETE 操作が遅延します。

purge_lag が非常に大きくなるような極端な状況で、過剰な遅延を回避するには、innodb_max_purge_lag_delay 構成オプションを設定すれば、遅延の量に上限を設定できます。遅延は、パージバッチの開始時に計算されます。

トランザクションが小規模 (100 バイト程度のサイズ) であり、許可されている未パージの InnoDB テーブル行が 100M バイトであると仮定すると、問題のあるワークロードに適した一般的な設定が 100 万になる可能性があります。

遅延の値は、InnoDB Monitor 出力の TRANSACTIONS セクションに履歴リストの長さとして表示されます。たとえば、出力に次の行が含まれている場合は、遅延の値が 20 です。

```
------------
TRANSACTIONS
------------
Trx id counter 0 290328385
Purge done for trx's n:o < 0 290315608 undo n:o < 0 17
History list length 20
```

一般的な I/O チューニングのアドバイスについては、セクション8.5.7「InnoDB ディスク I/O の最適化」を参照してください。

- innodb_max_purge_lag_delay

コマンド行形式	--innodb-max-purge-lag-delay=#
導入	5.6.5
システム変数	innodb_max_purge_lag_delay
スコープ	グローバル
動的	はい
型	数値
デフォルト	0
最小値	0

innodb_max_purge_lag 構成オプションで課された遅延の最大遅延をミリ秒単位で指定します。ゼロ以外の値は、innodb_max_purge_lag の値に基づいて、公式から計算された遅延期間への上限を表します。デフォルトのゼロは、遅延間隔に上限が課されていないことを意味します。

一般的な I/O チューニングのアドバイスについては、セクション8.5.7「InnoDB ディスク I/O の最適化」を参照してください。

- innodb_mirrored_log_groups

効果はありません。この変数は MySQL 5.6.11 の時点で非推奨となり、将来の MySQL リリースで削除される予定です。

- innodb_monitor_disable

コマンド行形式	--innodb-monitor-disable=[counter|module|pattern|all]
導入	5.6.2
システム変数	innodb_monitor_disable
スコープ	グローバル
動的	はい

型	文字列

INFORMATION_SCHEMA.INNODB_METRICS テーブルで 1 つ以上のカウンタをオフにします。使用法については、セクション21.29.19「INFORMATION_SCHEMA INNODB_METRICS テーブル」を参照してください。

- innodb_monitor_enable

コマンド行形式	--innodb-monitor-enable=[counter|module|pattern|all]
導入	5.6.2
システム変数	innodb_monitor_enable
スコープ	グローバル
動的	はい
型	文字列

INFORMATION_SCHEMA.INNODB_METRICS テーブルで 1 つ以上のカウンタをオンにします。使用法については、セクション21.29.19「INFORMATION_SCHEMA INNODB_METRICS テーブル」を参照してください。

- innodb_monitor_reset

コマンド行形式	--innodb-monitor-reset=[counter|module|pattern|all]
導入	5.6.2
システム変数	innodb_monitor_reset
スコープ	グローバル
動的	はい
型	文字列

INFORMATION_SCHEMA.INNODB_METRICS テーブルで 1 つ以上のカウンタに対応するカウント値をゼロにリセットします。使用法については、セクション21.29.19「INFORMATION_SCHEMA INNODB_METRICS テーブル」を参照してください。

- innodb_monitor_reset_all

コマンド行形式	--innodb-monitor-reset-all=[counter|module|pattern|all]
導入	5.6.2
システム変数	innodb_monitor_reset_all
スコープ	グローバル
動的	はい
型	文字列

INFORMATION_SCHEMA.INNODB_METRICS テーブルで 1 つ以上のカウンタに対応するすべての値 (最小、最大など) をリセットします。使用法については、セクション21.29.19「INFORMATION_SCHEMA INNODB_METRICS テーブル」を参照してください。

- innodb_old_blocks_pct

コマンド行形式	--innodb-old-blocks-pct=#
システム変数	innodb_old_blocks_pct
スコープ	グローバル
動的	はい
型	数値
デフォルト	37
最小値	5
最大値	95

古いブロックサブリストで使用される InnoDB のバッファープールの概算割合を指定します。値の範囲は 5 から 95 です。デフォルト値は 37 (つまり、プールの 3/8) です。多くの場合、innodb_old_blocks_time と組み

合わせて使用されます。詳細は、セクション14.13.1.3「バッファープールをスキャンに耐えられるようにする」を参照してください。LRU アルゴリズムやエビクションポリシーなどのバッファープール管理については、セクション8.9.1「InnoDB バッファープール」を参照してください。

- innodb_old_blocks_time

コマンド行形式	--innodb-old-blocks-time=#
システム変数	innodb_old_blocks_time
スコープ	グローバル
動的	はい
型	数値
デフォルト (≥ 5.6.6)	1000
デフォルト (≤ 5.6.5)	0
最小値	0
最大値	2**32-1

ゼロ以外の値にすると、バッファープールがテーブルの完全スキャン時などの短期間でのみ参照されるデータでいっぱいになることから保護されます。この値を大きくすると、テーブルの完全スキャンがバッファープール内にキャッシュされたデータとやりとりすることからさらに保護されます。

最初のアクセス後に、古いサブリストに挿入されるブロックが新しいサブリストに移動するまでに、そこに滞在する必要のある期間をミリ秒 (ms) 単位で指定します。値を 0 にすると、古いサブリストに挿入されたブロックは、挿入後にどのくらいの期間でアクセスが発生するのかには関係なく、最初のアクセスの直後に新しいサブリストに移動します。この値が 0 より大きい場合、ブロックは最初のアクセス後、少なくともそのミリ秒でアクセスが発生するまで、古いサブリストに残ります。たとえば、1000 の値では、ブロックは最初のアクセス後、それらが新しいサブリストに移動される資格を得るまで、1 秒間古いサブリストにとどまります。

デフォルト値は、MySQL 5.6.6 の時点では 1000、それよりも前では 0 です。

多くの場合、この変数は innodb_old_blocks_pct と組み合わせて使用されます。詳細は、セクション14.13.1.3「バッファープールをスキャンに耐えられるようにする」を参照してください。LRU アルゴリズムやエビクションポリシーなどのバッファープール管理については、セクション8.9.1「InnoDB バッファープール」を参照してください。

- innodb_online_alter_log_max_size

コマンド行形式	--innodb-online-alter-log-max-size=#
導入	5.6.6
システム変数	innodb_online_alter_log_max_size
スコープ	グローバル
動的	はい
型	数値
デフォルト	134217728
最小値	65536
最大値	2**64-1

InnoDB テーブルに対するオンライン DDL 操作時に使用される一時ログファイルのサイズに上限を指定します。作成されるインデックスまたは変更されるテーブルごとに、このようなログファイルが 1 つ存在します。このログファイルには、DDL 操作時にテーブルで挿入、更新、または削除されたデータが格納されます。一時ログファイルは、innodb_sort_buffer_size の値で必要になったときに、最大で innodb_online_alter_log_max_size で指定された最大値まで拡張されます。一時ログファイルがサイズの上限を超えた場合は、ALTER TABLE 操作に失敗し、コミットされていない並列 DML 操作がすべてロールバックされます。したがって、このオプションの値を大きくすると、オンライン DDL 操作時に実行できる DML 数は多くなりますが、ログからデータを適用するためにテーブルがロックされると、DDL 操作の終了時の期間も長くなります。

- innodb_open_files

コマンド行形式	--innodb-open-files=#

システム変数	innodb_open_files
スコープ	グローバル
動的	いいえ
型	数値
デフォルト (≥ 5.6.6)	-1 (autosized)
デフォルト (≤ 5.6.5)	300
最小値	10
最大値	4294967295

この変数は、複数の InnoDB テーブルスペースを使用する場合にのみ関連します。MySQL で一度に開いたままにできる .ibd ファイルの最大数が指定されます。最小値は 10 です。MySQL 5.6.6 の時点では、innodb_file_per_table が無効になっている場合のデフォルト値は 300 です。それ以外の場合は、300 よりも大きい値および table_open_cache です。5.6.6 よりも前のデフォルト値は 300 です。

.ibd ファイルで使用されるファイルディスクリプタは、InnoDB テーブルでのみ使用されます。それらは、--open-files-limit サーバーオプションによって指定されたものからは独立していて、テーブルキャッシュの操作に影響を与えません。一般的な I/O チューニングのアドバイスについては、セクション8.5.7「InnoDB ディスク I/O の最適化」を参照してください。

- innodb_optimize_fulltext_only

コマンド行形式	--innodb-optimize-fulltext-only=#
導入	5.6.4
システム変数	innodb_optimize_fulltext_only
スコープ	グローバル
動的	はい
型	ブール
デフォルト	OFF

InnoDB テーブルでの OPTIMIZE TABLE ステートメントの動作方法を変更します。FULLTEXT インデックスを含む InnoDB テーブルのメンテナンス操作時に、一時的に有効にするために使用されます。

デフォルトでは、OPTIMIZE TABLE はテーブルのクラスタ化されたインデックス内のデータを再構成します。このオプションを有効にすると、OPTIMIZE TABLE はこのようなテーブルデータの再構成をスキップし、その代わりに FULLTEXT インデックス用に新たに追加、削除、および更新されたトークンデータを処理します。InnoDB テーブルの FULLTEXT インデックスについての詳細は、セクション14.2.13.3「FULLTEXT インデックス」を参照してください。

- innodb_page_size

コマンド行形式	--innodb-page-size=#k
導入	5.6.4
システム変数	innodb_page_size
スコープ	グローバル
動的	いいえ
型	列挙
デフォルト	16384
有効な値	4k 8k 16k 4096 8192

	16384

MySQL インスタンス内のすべての InnoDB テーブルスペースのページサイズを指定します。この値は、インスタンスが作成されたあとに、定数が残っている場合に設定されます。ページサイズは、値 16k (デフォルト)、8k、または 4k を使用して指定できます。また、バイト単位 (4096、8192、16384) でページサイズを指定することもできます。

最大のページサイズでは、デフォルトが広範囲のワークロード (特に、テーブルスキャンを伴うクエリーや一括更新を伴う DML 操作) に適しています。ページサイズが小さいほど、多くの小規模な書き込みを伴う OLTP ワークロードの効率性が高くなる可能性があります。その一方で、単一のページに数多くの行が含まれる場合は、競合の問題が発生する可能性もあります。ページを小さくすると、一般に小さなブロックサイズが使用される SSD ストレージデバイスの効率性が高くなる可能性もあります。InnoDB のページサイズをストレージデバイスのブロックサイズに近づけると、ディスクに再度書き込まれる未変更データの量が最小限になります。一般的な I/O チューニングのアドバイスについては、セクション8.5.7「InnoDB ディスク I/O の最適化」を参照してください。

- innodb_print_all_deadlocks

コマンド行形式	--innodb-print-all-deadlocks=#
導入	5.6.2
システム変数	innodb_print_all_deadlocks
スコープ	グローバル
動的	はい
型	ブール
デフォルト	OFF

このオプションを有効にすると、mysqld エラーログに、InnoDB のユーザートランザクション内のすべてのデッドロックに関する情報が記録されます。それ以外の場合は、SHOW ENGINE INNODB STATUS コマンドを使用すると、最後のデッドロックに関する情報のみが表示されます。不定期に発生する InnoDB のデッドロックは、InnoDB によってすぐに状況が検出され、自動的にトランザクションのいずれかがロールバックされるため、必ずしも問題になるとは限りません。アプリケーションにロールバックを検出し、その操作を再試行するための適切なエラー処理ロジックが存在しない場合は、デッドロックが発生する原因についてトラブルシューティングを行う際に、このオプションを使用するとよいでしょう。多数のデッドロックが発生する場合は、各トランザクションが同じ順序でテーブルにアクセスするように (これにより、デッドロックの状況が回避されます)、複数のテーブルに対して DML または SELECT ... FOR UPDATE ステートメントを発行するトランザクションを再構築する必要があることを示している可能性があります。

- innodb_purge_batch_size

コマンド行形式	--innodb-purge-batch-size=#
システム変数	innodb_purge_batch_size
スコープ	グローバル
動的	はい
型	数値
デフォルト (≥ 5.6.3)	300
デフォルト (≤ 5.6.2)	20
最小値	1
最大値	5000

Redo ログレコードの単位で表現される変更の粒度です。これにより、パージ操作がトリガーされ、変更されたバッファープールブロックがディスクにフラッシュされます。このオプションは、innodb_purge_threads=n 設定と組み合わせてパフォーマンスを調整するために使用されるため、一般的なユーザーは変更する必要がありません。

- innodb_purge_threads

コマンド行形式	--innodb-purge-threads=#

システム変数	innodb_purge_threads
スコープ	グローバル
動的	いいえ
型	数値
デフォルト (≥ 5.6.5)	1
デフォルト (≤ 5.6.4)	0
最小値 (≥ 5.6.5)	1
最小値 (≤ 5.6.4)	0
最大値 (≥ 5.6.2)	32
最大値 (≤ 5.6.1)	1

InnoDB のパージ操作専用のバックグラウンドスレッドの数です。MySQL 5.6.5 での新しいデフォルトの最小値である 1 は、パージ操作がマスタースレッドの一部としてではなく、常にバックグラウンドスレッドで実行されることを表します。ゼロ以外の値にすると、1 つ以上のバックグラウンドスレッドでパージ操作が実行されるため、InnoDB 内の内部競合を削減でき、拡張性が改善されます。この値を 1 よりも大きくすると、数多くの個別のパージスレッドが作成されるため、複数のテーブル上で DML 操作が実行されるシステムの効率性を改善できます。最大値は 32 です。

- innodb_random_read_ahead

コマンド行形式	--innodb-random-read-ahead=#
導入	5.6.3
システム変数	innodb_random_read_ahead
スコープ	グローバル
動的	はい
型	ブール
デフォルト	OFF

InnoDB の I/O を最適化するために、ランダムな先読み技術を有効にします。ランダム先読み機能は、InnoDB Plugin (バージョン 1.0.4) から削除されたため、InnoDB Plugin が InnoDB の「組み込みの」バージョンになった時点では MySQL 5.5.0 に含まれていませんでした。ランダム先読みは、innodb_random_read_ahead 構成オプション (デフォルトでは無効になっています) とともに、MySQL 5.1.59 および 5.5.16 以降でふたたび導入されました。

さまざまなタイプの先読みリクエストに関するパフォーマンスの考慮事項についての詳細は、セクション 14.13.1.1「InnoDB バッファープールのプリフェッチ (先読み) の構成」を参照してください。一般的な I/O チューニングのアドバイスについては、セクション8.5.7「InnoDB ディスク I/O の最適化」を参照してください。

- innodb_read_ahead_threshold

コマンド行形式	--innodb-read-ahead-threshold=#
システム変数	innodb_read_ahead_threshold
スコープ	グローバル
動的	はい
型	数値
デフォルト	56
最小値	0
最大値	64

バッファープールにページをプリフェッチする際に InnoDB で使用される線形の先読みの感度を制御します。InnoDB が少なくとも innodb_read_ahead_threshold ページをエクステント (64 ページ) から連続して読み取る場合は、次のエクステント全体の非同期読み取りが開始されます。許可される値の範囲は 0 から 64 までです。MySQL 5.6.1 の時点では、値を 0 にすると先読みが無効になります。5.6.1 よりも前では、値を 0 にすると、64 ページのエクステントの境界ページを読み取るときに、先読みがトリガーされました。デフォルトの 56

では、InnoDB は次のエクステント全体の非同期読み取りを開始するために、少なくとも 56 ページをエクステントから連続して読み取る必要があります。

この先読みメカニズムによって読み取られるページ数、およびその中でアクセスされずにバッファープールから削除される数を把握しておくと、innodb_read_ahead_threshold パラメータを微調整する際に役立ちます。MySQL 5.5 の時点では、SHOW ENGINE INNODB STATUS の出力に、Innodb_buffer_pool_read_ahead および Innodb_buffer_pool_read_ahead_evicted グローバルステータス変数からのカウンタ情報が表示されます。これらの変数は、先読みリクエストによってバッファープールに格納されるページの数、およびこのようなページの中で、個別にアクセスされずにバッファープールからエビクションされる数を示します。これらのカウンタには、最後にサーバーが再起動された以降のグローバルな値が表示されます。

SHOW ENGINE INNODB STATUS には、先読みページが読み取られる比率、およびこのようなページがアクセスされずに削除される比率も表示されます。1 秒ごとの平均値は、最後に SHOW ENGINE INNODB STATUS が呼び出された以降に収集された統計に基づいて計算され、出力の BUFFER POOL AND MEMORY セクションに表示されます。

詳細は、セクション14.13.1.1「InnoDB バッファープールのプリフェッチ (先読み) の構成」を参照してください。一般的な I/O チューニングのアドバイスについては、セクション8.5.7「InnoDB ディスク I/O の最適化」を参照してください。

- innodb_read_io_threads

コマンド行形式	--innodb-read-io-threads=#
システム変数	innodb_read_io_threads
スコープ	グローバル
動的	いいえ
型	数値
デフォルト	4
最小値	1
最大値	64

InnoDB での読み取り操作で使用される I/O スレッドの数です。デフォルト値は 4 です。書き込みスレッドで対応するものは、innodb_write_io_threads です。詳細は、セクション14.13.6「InnoDB バックグラウンド I/O スレッドの数の構成」を参照してください。一般的な I/O チューニングのアドバイスについては、セクション8.5.7「InnoDB ディスク I/O の最適化」を参照してください。

> **注記**
>
> Linux システムでは、デフォルトの innodb_read_io_threads 設定で複数 (一般には 12 台よりも多く) の MySQL サーバーを実行すると、innodb_write_io_threads および Linux の aio-max-nr 設定がシステムの制限を超過する可能性があります。理想的には aio-max-nr 設定を大きくします。回避策として、MySQL 構成オプションの一方または両方の設定を小さくするとよいでしょう。

- innodb_read_only

コマンド行形式	--innodb-read-only=#
導入	5.6.7
システム変数	innodb_read_only
スコープ	グローバル
動的	いいえ
型	ブール
デフォルト	OFF

サーバーを読み取り専用モードで起動します。読み取り専用メディア上のデータベースアプリケーションまたはデータセットを配布するために使用されます。複数のインスタンス間で同じデータディレクトリを共有する際に、データウェアハウスで使用することもできます。使用手順については、セクション14.3.1「読み取り専用操作用の InnoDB の構成」を参照してください。

- innodb_replication_delay

コマンド行形式	--innodb-replication-delay=#
システム変数	innodb_replication_delay
スコープ	グローバル
動的	はい
型	数値
デフォルト	0
最小値	0
最大値	4294967295

innodb_thread_concurrency に達した場合のスレーブサーバー上のレプリケーションスレッドの遅延 (ミリ秒単位) です。

- innodb_rollback_on_timeout

コマンド行形式	--innodb-rollback-on-timeout
システム変数	innodb_rollback_on_timeout
スコープ	グローバル
動的	いいえ
型	ブール
デフォルト	OFF

MySQL 5.6 では、InnoDB はデフォルトで、トランザクションタイムアウト時に最後のステートメントのみをロールバックします。--innodb_rollback_on_timeout を指定すると、トランザクションタイムアウトによって、InnoDB はトランザクション全体を中止してロールバックします (MySQL 4.1 と同じ動作です)。

- innodb_rollback_segments

コマンド行形式	--innodb-rollback-segments=#
導入	5.6.2
システム変数	innodb_rollback_segments
スコープ	グローバル
動的	はい
型	数値
デフォルト	128
最小値	1
最大値	128

トランザクション内で InnoDB が使用するシステムテーブルスペースにあるロールバックセグメントの数を定義します。この設定は引き続き有効ですが、innodb_undo_logs で置き換えられます。

- innodb_sort_buffer_size

コマンド行形式	--innodb-sort-buffer-size=#
導入	5.6.4
システム変数	innodb_sort_buffer_size
スコープ	グローバル
動的	いいえ
型	数値
デフォルト	1048576
最小値 (≥ 5.6.5)	65536
最小値 (5.6.4)	524288

最大値	67108864

InnoDB インデックスの作成時に、データを格納する際に使用されるソートバッファーのサイズを指定します。指定されたサイズでは、内部ソート用にメモリーに入力され、ディスクに書き込まれたデータの量が定義されます。これは、「実行」と呼ばれることもあります。マージフェーズ時に、指定されたサイズのバッファーのペアが「読み取られ」、マージされます。設定を大きくするほど、「実行」数およびマージ数が少なくなります。このことは、チューニングの観点から理解する際に重要です。

このソート領域は、後続のインデックスのメンテナンス操作時ではなく、インデックスの作成時のマージソートでのみ使用されます。インデックスの作成が完了すると、バッファーの割り当てが解除されます。

このオプションの値では、オンライン DDL の操作時に並列 DML を記録するために、一時ログファイルが拡張される量も制御されます。

この設定が構成可能になる前は、サイズが 1048576 バイト (1M バイト) にハードコーディングされていました。その値は、現在もデフォルトのままです。

インデックスを作成する ALTER TABLE または CREATE TABLE ステートメントの実行時に、それぞれが、このオプションで定義されたサイズを持つ 3 つのバッファーが割り当てられます。さらに、ポインタ上でソートを実行できるように、ソートバッファー内の行に補助ポインタが割り当てられます (これは、ソート操作時の行の移動とは異なります)。

一般的なソート操作では、次のような公式を使用して、メモリーの消費を見積もることができます。

```
(6 /*FTS_NUM_AUX_INDEX*/ *
(3*@@GLOBAL.innodb_sort_buffer_size) + 2 * (
@@GLOBAL.innodb_sort_buffer_size/dict_index_get_min_size(index)*/)
* 8 /*64-bit sizeof *buf->tuples*/")
```

「@@GLOBAL.innodb_sort_buffer_size/dict_index_get_min_size(index)」 は、保持される最大のタプル数を示します。「2 * (@@GLOBAL.innodb_sort_buffer_size/*dict_index_get_min_size(index)*/) * 8 /*64-bit size of *buf->tuples*/ は、割り当てられた補助ポインタ数を示します」。

> **注記**
>
> 32 ビットの場合は、8 の代わりに 4 で乗算します。

全文インデックスでの並列ソートでは、innodb_ft_sort_pll_degree の設定で乗算します。

```
(6 /*FTS_NUM_AUX_INDEX*/ @@GLOBAL.innodb_ft_sort_pll_degree)
```

- innodb_spin_wait_delay

コマンド行形式	--innodb-spin-wait-delay=#
システム変数	innodb_spin_wait_delay
スコープ	グローバル
動的	はい
型	数値
デフォルト	6
最小値	0
最大値	4294967295

スピンロックでのポーリング間の最大遅延です。このメカニズムの低レベルの実装は、ハードウェアとオペレーティングシステムの組み合わせによって異なるため、遅延は一定の時間間隔に対応しません。デフォルト値は 6 です。詳細は、セクション14.13.10「スピンロックのポーリングの構成」を参照してください。

- innodb_stats_auto_recalc

コマンド行形式	--innodb-stats-auto-recalc=#
導入	5.6.6
システム変数	innodb_stats_auto_recalc
スコープ	グローバル

動的	はい
型	ブール
デフォルト	ON

テーブル内のデータが大幅に変更されたあとは、InnoDB によって自動的に永続的統計が再計算されます。現在のしきい値は、テーブル内の行の 10% です。この設定は、innodb_stats_persistent オプションが有効になっている場合や、CREATE TABLE または ALTER TABLE ステートメントで STATS_PERSISTENT=1 句が有効になっている場合に、作成されたテーブルに適用されます。統計を生成するためにサンプルとして取得されるデータの量は、innodb_stats_persistent_sample_pages 構成オプションで制御されます。

innodb_stats_auto_recalc に関する追加情報については、セクション14.13.16.1「永続的オプティマイザ統計のパラメータの構成」を参照してください。

- innodb_stats_method

コマンド行形式	--innodb-stats-method=name
導入	5.6.2
システム変数	innodb_stats_method
スコープ	グローバル
動的	はい
型	列挙
デフォルト	nulls_equal
有効な値	nulls_equal nulls_unequal nulls_ignored

InnoDB テーブルのインデックス値の分布に関する統計を収集するときに、サーバーが NULL 値を処理する方法です。この変数は、nulls_equal、nulls_unequal、および nulls_ignored の 3 つの値を指定できます。nulls_equal の場合、すべての NULL インデックス値を同等として扱い、NULL 値の数とサイズが同等の単一値グループを生成します。nulls_unequal の場合、NULL 値同士を同等として扱わず、それぞれの NULL はサイズが 1 の別個のグループを生成します。nulls_ignored の場合、NULL 値は無視されます。

テーブル統計を生成するために使用する方法は、セクション8.3.7「InnoDB および MyISAM インデックス統計コレクション」に記載されているように、オプティマイザがクエリー実行のためのインデックスを選択する方法に影響を与えます。

- innodb_stats_on_metadata

コマンド行形式	--innodb-stats-on-metadata
システム変数	innodb_stats_on_metadata
スコープ	グローバル
動的	はい
型	ブール
デフォルト (≥ 5.6.6)	OFF
デフォルト (≤ 5.6.5)	ON

この変数を有効にすると、SHOW TABLE STATUS や SHOW INDEX などのメタデータステートメントが実行されるときや、INFORMATION_SCHEMA テーブル TABLES または STATISTICS にアクセスするときに、InnoDB によって統計が更新されます。(これらの更新は、ANALYZE TABLE で実行されるものに似ています。)無効にすると、これらの操作時に InnoDB によって統計が更新されません。この設定を無効のままにする

と、多数のテーブルまたはインデックスを持つスキーマのアクセス速度を改善できます。InnoDB テーブルが関与するクエリーの実行計画の安定性も改善できます。

設定を変更するには、SET GLOBAL innodb_stats_on_metadata=mode ステートメントを発行します。ここで、mode は ON と OFF のいずれか (または 1 と 0 のいずれか) です。この設定を変更するには、SUPER 権限が必要です。変更すると、すべての接続の操作がすぐに影響を受けます。

MySQL 5.6.6 の時点では、この変数はデフォルトで無効になっています。それよりも前では、有効になっています。

- innodb_stats_persistent

コマンド行形式	--innodb-stats-persistent=setting
導入	5.6.6
システム変数	innodb_stats_persistent
スコープ	グローバル
動的	はい
型	ブール
デフォルト	ON
有効な値	OFF ON 0 1

InnoDB インデックスの統計がディスクに保持されるかどうかを指定します。それ以外の場合は、頻繁に統計が再計算される可能性があります。これにより、クエリーの実行計画が変化する可能性があります。テーブルが作成されると、この設定が各テーブルとともに格納されます。テーブルを作成する前にグローバルレベルで innodb_stats_persistent を設定することも、CREATE TABLE および ALTER TABLE ステートメントで STATS_PERSISTENT 句を使用して、システム全体の設定をオーバーライドし、個々のテーブルの永続的統計を構成することもできます。

このオプションについての詳細は、セクション14.13.16.1「永続的オプティマイザ統計のパラメータの構成」を参照してください。

- innodb_stats_persistent_sample_pages

コマンド行形式	--innodb-stats-persistent-sample-pages=#
導入	5.6.2
システム変数	innodb_stats_persistent_sample_pages
スコープ	グローバル
動的	はい
型	数値
デフォルト	20

インデックス付きカラムのカーディナリティーやその他の統計 (ANALYZE TABLE で計算された統計など) を見積もるときに、サンプルとして取得されるインデックスページの数です。値を大きくすると、クエリーの実行計画を改善するインデックス統計の精度が改善されますが、InnoDB テーブルに対する ANALYZE TABLE の実行時に I/O が増加することになります。追加情報については、セクション14.13.16.1「永続的オプティマイザ統計のパラメータの構成」を参照してください。

> **注記**
>
> innodb_stats_persistent_sample_pages に大きな値を設定すると、ANALYZE TABLE の実行時間が長くなる可能性があります。アクセスされるデータベースページの数を

> 見積もる方法については、セクション14.13.17「InnoDB テーブルに対する ANALYZE TABLE の複雑さの推定」を参照してください。

このオプションは、テーブルで innodb_stats_persistent 設定がオンになっている場合にのみ適用されます。このオプションがテーブルでオフになっている場合は、代わりに innodb_stats_transient_sample_pages 設定が適用されます。

- innodb_stats_sample_pages

コマンド行形式	--innodb-stats-sample-pages=#
非推奨	5.6.3
システム変数	innodb_stats_sample_pages
スコープ	グローバル
動的	はい
型	数値
デフォルト	8
最小値	1
最大値	2**64-1

非推奨です。代わりに innodb_stats_transient_sample_pages を使用してください。

- innodb_stats_transient_sample_pages

コマンド行形式	--innodb-stats-transient-sample-pages=#
導入	5.6.2
システム変数	innodb_stats_transient_sample_pages
スコープ	グローバル
動的	はい
型	数値
デフォルト	8

インデックス付きカラムのカーディナリティーやその他の統計 (ANALYZE TABLE で計算された統計など) を見積もるときに、サンプルとして取得されるインデックスページの数です。デフォルト値は 8 です。値を大きくすると、インデックス統計の精度が改善されます。これにより、クエリーの実行計画を改善できますが、InnoDB テーブルを開くときや統計を再計算するときに I/O が増加するという犠牲が伴います。追加情報については セクション14.13.16.2「非永続的オプティマイザ統計のパラメータの構成」を参照してください。

> 注記
>
> innodb_stats_transient_sample_pages に大きな値を設定すると、ANALYZE TABLE の実行時間が長くなる可能性があります。アクセスされるデータベースページの数を見積もる方法については、セクション14.13.17「InnoDB テーブルに対する ANALYZE TABLE の複雑さの推定」を参照してください。

このオプションは、テーブルで innodb_stats_persistent 設定がオフになっている場合にのみ適用されます。このオプションがテーブルでオンになっている場合は、代わりに innodb_stats_persistent_sample_pages 設定が適用されます。innodb_stats_sample_pages オプションの場所を取得します。

- innodb_status_output

コマンド行形式	--innodb-status-output
導入	5.6.16
システム変数	innodb_status_output
スコープ	グローバル
動的	はい
型	ブール
デフォルト	OFF

標準 InnoDB Monitor の定期的な出力を有効または無効にする際に使用されます。また、InnoDB Lock Monitor の定期的な出力を有効または無効にする際に、innodb_status_output_locks と組み合わせて使用されます。追加情報については、セクション14.15「InnoDB モニター」を参照してください。

- innodb_status_output_locks

コマンド行形式	--innodb-status-output-locks
導入	5.6.16
システム変数	innodb_status_output_locks
スコープ	グローバル
動的	はい
型	ブール
デフォルト	OFF

InnoDB Lock Monitor の定期的な出力を有効または無効にする際に使用されます。innodb_status_output と組み合わせて使用する必要があります。追加情報については、セクション14.15「InnoDB モニター」を参照してください。

- innodb_strict_mode

コマンド行形式	--innodb-strict-mode=#
システム変数	innodb_strict_mode
スコープ	グローバル、セッション
動的	はい
型	ブール
デフォルト	OFF

innodb_strict_mode を ON にすると、InnoDB は特定の条件に対応した警告ではなく、エラーを返します。デフォルト値は OFF です。

厳密モードは、SQL 内の無視できる誤字や構文エラー、または操作モードと SQL ステートメントのさまざまな組み合わせによる意図しないその他の結果から保護する際に役立ちます。innodb_strict_mode を ON にすると、InnoDB は警告を発行して、指定されたステートメントを処理する (意図しない動作が伴う可能性があります) のではなく、特定のケースでエラー状況が発生します。これは、MySQL で受け入れられる SQL 構文を制御し、警告なしでエラーを無視するのか、入力構文とデータ値を検証するのかを決定する MySQL の sql_mode と類似しています。

innodb_strict_mode の設定によって、CREATE TABLE、ALTER TABLE、および CREATE INDEX ステートメントの構文エラー処理が影響を受けます。innodb_strict_mode では、選択したページサイズに対してレコードが大きくなりすぎることが原因で、INSERT または UPDATE が失敗しないように、レコードサイズのチェックも有効になります。

オラクルでは、CREATE TABLE、ALTER TABLE、および CREATE INDEX ステートメントで ROW_FORMAT および KEY_BLOCK_SIZE 句を使用する際に、innodb_strict_mode を有効にすることが推奨されています。innodb_strict_mode を OFF にすると、InnoDB は競合する句を無視し、テーブルまたはインデックスを作成し、メッセージログに警告のみが表示されます。結果として生成されるテーブルでは、圧縮されたテーブルを作成しようとしても圧縮されないなど、意図したものとは異なる動作が発生する可能性があります。innodb_strict_mode を ON にすると、このような問題が発生するとすぐにエラーが生成され、テーブルまたはインデックスは作成されないため、後続のトラブルシューティングセッションが回避されます。

mysqld の起動時にコマンド行で、または my.cnf または my.ini 構成ファイルで、innodb_strict_mode の ON と OFF を切り替えることができます。SET [GLOBAL|SESSION] innodb_strict_mode=mode ステートメントを使用すれば、実行時に innodb_strict_mode を有効または無効にすることもできます。ここで、mode は ON と OFF のいずれかです。GLOBAL 値を変更するには、SUPER 権限が必要です。これを変更すると、それ以降に接続するすべてのクライアントの操作が影響を受けます。任意のクライアントが innodb_strict_mode の SESSION 設定を変更でき、そのクライアントのみが設定の影響を受けます。

- innodb_support_xa

コマンド行形式	--innodb-support-xa

システム変数	innodb_support_xa
スコープ	グローバル、セッション
動的	はい
型	ブール
デフォルト	TRUE

XA トランザクションの 2 相コミットで InnoDB のサポートを有効にします。これにより、トランザクションの準備時に追加のディスクフラッシュが発生します。この設定はデフォルトです。XA メカニズムは内部で使用されるため、バイナリログがオンになっていて、複数のスレッドからのデータの変更が許可されている任意のサーバーで重要となります。オフにすると、ライブデータベースがコミットするときとは異なる順序で、トランザクションがバイナリログに書き込まれる可能性があります。これにより、ディザスタリカバリ時やレプリケーションスレーブでバイナリログが再現されるときに、異なるデータが生成される可能性があります。1 つのスレッドしかデータを変更できない例外的な設定を使用している場合を除いて、レプリケーションマスターサーバーではオフにしないでください。

1 つのスレッドからのデータ変更のみが許可されているサーバーでは、InnoDB テーブルのパフォーマンスを改善するために、このオプションをオフにすることが安全であり、推奨されています。たとえば、レプリケーション SQL スレッドのみがデータを変更するレプリケーションスレーブでは、オフにすることができます。

また、安全なバイナリロギングまたはレプリケーションで必要でなく、外部の XA トランザクションマネージャーを使用しない場合でも、このオプションをオフにすることができます。

- innodb_sync_array_size

コマンド行形式	--innodb-sync-array-size=#
導入	5.6.3
システム変数	innodb_sync_array_size
スコープ	グローバル
動的	いいえ
型	数値
デフォルト	1
最小値	1
最大値	1024

大量の待機中スレッドを含むワークロードの並列性を高くするために、スレッドの調整に使用される内部データ構造を分割します。この設定は MySQL インスタンスの起動時に構成する必要があり、あとで変更することはできません。大量の (一般に 768 を超える) 待機中スレッドが頻繁に生成されるワークロードでは、このオプション値を大きくすることをお勧めします。

- innodb_sync_spin_loops

コマンド行形式	--innodb-sync-spin-loops=#
システム変数	innodb_sync_spin_loops
スコープ	グローバル
動的	はい
型	数値
デフォルト	30
最小値	0
最大値	4294967295

スレッドが中断される前に、InnoDB 相互排他ロックが開放されるまでスレッドが待機する回数です。デフォルト値は 30 です。

- innodb_table_locks

コマンド行形式	--innodb-table-locks

システム変数	innodb_table_locks
スコープ	グローバル、セッション
動的	はい
型	ブール
デフォルト	TRUE

autocommit = 0 の場合、InnoDB は LOCK TABLES の要求を受け入れます。MySQL はすべてのスレッドがテーブルに対するすべてのロックを解放するまで、LOCK TABLES ... WRITE から戻りません。innodb_table_locks のデフォルト値は 1 です。これは、autocommit = 0. の場合、LOCK TABLES によって InnoDB がテーブルを内部的にロックすることを意味します。

MySQL 5.6 では、LOCK TABLES ... WRITE を使用して明示的にロックされたテーブルには、innodb_table_locks = 0 が無効です。LOCK TABLES ... WRITE で暗黙的に (たとえば、トリガーを使用して)、または LOCK TABLES ... READ によって、読み取りまたは書き込み用にロックされたテーブルには有効です。

- innodb_thread_concurrency

コマンド行形式	--innodb-thread-concurrency=#
システム変数	innodb_thread_concurrency
スコープ	グローバル
動的	はい
型	数値
デフォルト	0
最小値	0
最大値	1000

InnoDB は、この変数で指定された制限以下の数のオペレーティングシステムスレッドを同時に InnoDB 内部に保持しようと試みます (InnoDB では、ユーザートランザクションを処理する際にオペレーティングシステムのスレッドが使用されます)。スレッド数がこの制限に達すると、それ以降のスレッドは実行されるまで、「先入れ先出し」 (FIFO) キュー内で待機状態になります。ロックを待機しているスレッドは、並列実行中のスレッドの数にカウントされません。

この変数の範囲は 0 から 1000 までです。値 0 (デフォルト) は、無制限の並列性 (並列性チェックなし) と解釈されます。スレッドの並列性チェックを無効にすると、InnoDB は必要な数だけのスレッドを作成できます。値を 0 にすると、InnoDB 内部のクエリーおよび SHOW ENGINE INNODB STATUS 出力の ROW OPERATIONS セクションにあるキューカウンタ内のクエリーも無効になります。

MySQL インスタンスとその他のアプリケーションで CPU リソースを共有している場合や、ワークロードまたは並列ユーザー数が増加している場合は、この変数を設定することを検討してください。適切な設定は、ワークロード、コンピューティング環境、および実行中の MySQL のバージョンによって異なります。最適なパフォーマンスを実現する設定を決定するには、広範囲の値をテストする必要があります。innodb_thread_concurrency は動的な変数であるため、これを使用すれば、ライブテストシステムでさまざまな設定を試みることができます。特定の設定でパフォーマンスが低下した場合は、すぐに innodb_thread_concurrency を 0 に戻してください。

次のガイドラインに従うと、適切な設定を見つけて保持する際に役立ちます。

- ワークロードに対する並列ユーザースレッドの数が 64 よりも少ない場合は、innodb_thread_concurrency=0 を設定します。
- ワークロードの負荷が常に高い場合や、ときどき急上昇する場合は、初めに innodb_thread_concurrency=128 を設定してから、最適なパフォーマンスが実現されるスレッド数が見つかるまで、値を 96、80、64 と小さくしてください。たとえば、システムに通常は 40 - 50 のユーザーが存在しますが、定期的にその数が 60、70、さらには 200 まで上昇すると仮定します。パフォーマンスは、並列ユーザーが 80 のときは安定していますが、この数を上回ると低下が見られ始めます。この場合、パフォーマンスへの影響を回避するには、innodb_thread_concurrency=80 を設定します。
- InnoDB でユーザースレッド用に特定の数を上回る vCPU (たとえば、20 個の vCPU) が使用されないようにする場合は、innodb_thread_concurrency をその数 (またはパフォーマンスの結果によっては、さらに小さい数) に設定します。MySQL をその他のアプリケーションから分離することが目的である場合は、排他的に

mysqld プロセスを vCPU にバインドすることを検討してみてください。ただし、mysqld プロセスの負荷が常に高いわけではない場合は、排他的なバインドによって最適なハードウェアの使用が実現されない可能性があります。この場合、mysqld プロセスを vCPU にバインドしても、その他のアプリケーションも vCPU の一部または全部を使用できます。

> **注記**
>
> オペレーティングシステムの観点から見ると、mysqld プロセスをバインドするよりも、リソース管理ソリューションを使用して (使用可能な場合)、アプリケーション間で CPU 時間を共有する方法を管理する方が適切な場合があります。たとえば、その他のクリティカルなプロセスが実行されていない場合は、vCPU 時間の 90% を特定のアプリケーションに割り当て、その他のクリティカルなプロセスが実行されている場合は、その値を 40% に戻します。

- innodb_thread_concurrency の値が大きすぎると、システム内部およびリソース上の競合が増加するため、パフォーマンスが低下する可能性があります。
- 場合によっては、最適な innodb_thread_concurrency が vCPU の数よりも小さくなる可能性もあります。
- 定期的にシステムをモニターし、分析してください。ワークロード、ユーザー数、またはコンピューティング環境を変更するために、innodb_thread_concurrency 設定の調整が必要なことがあります。

関連情報については、セクション14.13.5「InnoDB のスレッド並列性の構成」を参照してください。

- innodb_thread_sleep_delay

コマンド行形式	--innodb-thread-sleep-delay=#
システム変数	innodb_thread_sleep_delay
スコープ	グローバル
動的	はい
型	数値
デフォルト	10000
最小値	0
最大値 (64 ビットプラットフォーム, ≤ 5.6.16)	18446744073709551615
最大値 (32 ビットプラットフォーム, ≤ 5.6.16)	4294967295
最大値 (≥ 5.6.17)	1000000

InnoDB キューに参加するまでに、InnoDB スレッドがスリープ状態になる期間 (マイクロ秒単位) です。デフォルト値は 10000 です。0 の値はスリープを無効にします。MySQL 5.6.3 以降では、構成オプション innodb_adaptive_max_sleep_delay を innodb_thread_sleep_delay で許可される最大の値に設定でき、InnoDB によって自動的に、現在のスレッドスケジューリングアクティビティーに応じて innodb_thread_sleep_delay が上下に調整されます。この動的な調整は、システムにかかる負荷が軽い期間や、システムがほぼ容量いっぱいで動作している期間に、スレッドスケジューリングメカニズムがスムーズに機能するのに役立ちます。

詳細は、セクション14.13.5「InnoDB のスレッド並列性の構成」を参照してください。

- innodb_undo_directory

コマンド行形式	--innodb-undo-directory=dir_name
導入	5.6.3
システム変数	innodb_undo_directory
スコープ	グローバル
動的	いいえ
型	ディレクトリ名

デフォルト	.

InnoDB が Undo ログ用に個別のテーブルスペースを作成するディレクトリの相対パスまたは絶対パスです。一般に、これらのログを別のストレージデバイス上に配置する際に使用されます。innodb_undo_logs および innodb_undo_tablespaces と組み合わせて使用すると、システムテーブルスペース外の Undo ログのディスクレイアウトが決定されます。デフォルト値の . は、InnoDB がデフォルトでその他のログファイルを作成するディレクトリと同じであることを表します。

- innodb_undo_logs

コマンド行形式	--innodb-undo-logs=#
導入	5.6.3
システム変数	innodb_undo_logs
スコープ	グローバル
動的	はい
型	数値
デフォルト	128
最小値	0
最大値	128

トランザクション内で InnoDB が使用するシステムテーブルスペースにあるロールバックセグメントの数を定義します。この設定は、Undo ログに関連する相互排他競合が観察された場合のパフォーマンスのチューニングに適切です。innodb_rollback_segments 設定から置き換えられました。アクティブな Undo ログではなく、使用可能な Undo ログの合計数については、Innodb_available_undo_logs ステータス変数を参照してください。

トランザクション内で使用されるロールバックセグメントの数は大きくしたり、小さくしたりできますが、システム内に物理的に存在するロールバックセグメントの数は減少しません。したがって、あとで必要でなくなるロールバックセグメントが割り当てられることを回避するために、このパラメータは小さい値から始めて徐々に大きくするとよいでしょう。innodb_undo_logs が設定されていない場合は、デフォルトの最大値が 128 になります。ロールバックセグメントの管理については、セクション14.2.12「InnoDB マルチバージョン」を参照してください。

- innodb_undo_tablespaces

コマンド行形式	--innodb-undo-tablespaces=#
導入	5.6.3
システム変数	innodb_undo_tablespaces
スコープ	グローバル
動的	いいえ
型	数値
デフォルト	0
最小値	0
最大値	126

ゼロ以外の innodb_undo_logs 設定を使用する場合に、Undo ログが分割されるテーブルスペースファイルの数です。デフォルトでは、すべての Undo ログはシステムテーブルスペースの一部であり、システムテーブルスペースには、innodb_undo_tablespaces で構成されたものに加えて、常に 1 つの Undo テーブルスペースが含まれています。長時間実行されるトランザクション中に Undo ログが大きくなる可能性があるため、Undo ログを複数のテーブルスペースに分割すると、任意の 1 つのテーブルスペースの最大サイズが削減されます。テーブルスペースファイルは、innodb_undo_directory で定義された場所に、undoN 形式の名前で作成されます。ここで N は、先頭のゼロを含む一連の連続する整数です。Undo テーブルスペースファイルのデフォルトサイズは 10M です。はじめて InnoDB を初期化するときは、innodb_undo_tablespaces の数を設定する必要があります。最初にデータベースを作成したときに指定したときよりも多くの数の Undo テーブルスペースを持つ InnoDB を再起動しようとすると、起動に失敗し、InnoDB で予期された数の Undo テーブルスペースが見つからなかったことを示すエラーが表示されます。

- innodb_use_native_aio

コマンド行形式	--innodb-use-native-aio=#
システム変数	innodb_use_native_aio
スコープ	グローバル
動的	いいえ
型	ブール
デフォルト	ON

Linux の非同期 I/O サブシステムを使用するかどうかを指定します。この変数は Linux システムにのみ適用され、サーバーの実行中は変更できません。このオプションはデフォルトで有効になっているため、通常は触れる必要がありません。

MySQL 5.5 の時点では、Windows システム上の InnoDB に備わっている非同期 I/O 機能が Linux システムでも使用できます。(その他の Unix に似たシステムでは、引き続き同期 I/O 呼び出しが使用されます。)この機能によって、I/O 負荷の高いシステムの拡張性が改善されます。一般に、SHOW ENGINE INNODB STATUS\G コマンドの出力に数多くの中断された読み取り/書き込みが表示されます。

大量の InnoDB I/O スレッドとともに実行すると (特に、同じサーバーマシン上で複数のこのようなインスタンスを実行すると)、Linux システムの能力制限を超える可能性があります。この場合、次のエラーを受信する可能性があります。

```
EAGAIN: The specified maxevents exceeds the user's limit of available events.
```

一般に、/proc/sys/fs/aio-max-nr により大きな制限を記述すれば、このエラーに対処できます。

ただし、OS の非同期 I/O サブシステムの問題によって InnoDB が起動しない場合は、innodb_use_native_aio=0 を無効にして (オプションファイルで innodb_use_native_aio=0 を使用します)、サーバーを起動してください。また、InnoDB で tmpdir の場所、tmpfs ファイルシステム、および tmpfs 上で AIO がサポートされていない Linux カーネルなどを組み合わせた潜在的な問題が検出された場合に、このオプションが起動中に自動的にオフになる可能性もあります。

- innodb_use_sys_malloc

コマンド行形式	--innodb-use-sys-malloc=#
非推奨	5.6.3
システム変数	innodb_use_sys_malloc
スコープ	グローバル
動的	いいえ
型	ブール
デフォルト	ON

InnoDB がオペレーティングシステムのメモリーアロケータを使用するのか (ON)、独自のものを使用するのか (OFF) を指定します。デフォルト値は ON です。詳細は、セクション14.13.3「InnoDB のためのメモリーアロケータの構成」を参照してください。

MySQL 5.6.3 の時点では、innodb_use_sys_malloc は非推奨となり、今後の MySQL リリースで削除される予定です。

- innodb_version

InnoDB のバージョン番号です。5.6.11 以降では、InnoDB ごとに個別の番号付与が廃止され、この値は version 変数の番号と同じです。

- innodb_write_io_threads

コマンド行形式	--innodb-write-io-threads=#
システム変数	innodb_write_io_threads
スコープ	グローバル
動的	いいえ

型	数値
デフォルト	4
最小値	1
最大値	64

InnoDB の書き込み操作で使用される I/O スレッドの数です。デフォルト値は 4 です。読み取りスレッドで対応するものは、innodb_read_io_threads です。詳細は、セクション14.13.6「InnoDB バックグラウンド I/O スレッドの数の構成」を参照してください。一般的な I/O チューニングのアドバイスについては、セクション8.5.7「InnoDB ディスク I/O の最適化」を参照してください。

> **注記**
>
> Linux システムでは、デフォルトの innodb_read_io_threads 設定で複数 (一般には 12 台よりも多く) の MySQL サーバーを実行すると、innodb_write_io_threads および Linux の aio-max-nr 設定がシステムの制限を超過する可能性があります。理想的には aio-max-nr 設定を大きくします。回避策として、MySQL 構成オプションの一方または両方の設定を小さくするとよいでしょう。

また、ディスクへのバイナリログの同期を制御する sync_binlog の値も考慮に入れるようにしてください。

一般的な I/O チューニングのアドバイスについては、セクション8.5.7「InnoDB ディスク I/O の最適化」を参照してください。

14.13 InnoDB のパフォーマンス

このセクションでは、パフォーマンスとスケーラビリティーに関連した InnoDB の機能および拡張に関する情報を提供します。

InnoDB テーブルをチューニングするためのヒントとガイドラインは、最適化の章に示されています。セクション8.5「InnoDB テーブルの最適化」を参照してください。

14.13.1 InnoDB バッファープールの構成

このセクションでは、InnoDB バッファープールのパフォーマンス関連の構成情報について説明します。詳細は、セクション8.9.1「InnoDB バッファープール」を参照してください。

14.13.1.1 InnoDB バッファープールのプリフェッチ (先読み) の構成

先読み要求とは、バッファープール内の複数のページがすぐに必要になるという予測のもとに、非同期にこれらのページのプリフェッチを行う I/O 要求のことです。これらの要求によって、すべてのページが 1 つのエクステント内に移動されます。InnoDB は、I/O パフォーマンスを向上させるために、次の 2 つの先読みアルゴリズムを使用します。

線形先読みは、順次にアクセスされているバッファープール内のページに基づいて、どのページがすぐに必要になる可能性があるかを予測する手法です。構成パラメータ innodb_read_ahead_threshold を使用して、非同期読み取り要求をトリガーするために必要な順次ページアクセスの数を調整することにより、InnoDB がいつ先読み操作を実行するかを制御します。このパラメータが追加される前、InnoDB は現在のエクステントの最後のページを読み取るときに、次のエクステント全体に対する非同期プリフェッチ要求を発行するかどうかを推測するだけでした。

構成パラメータ innodb_read_ahead_threshold は、順次ページアクセスのパターンの検出において InnoDB がどれだけ早く反応するかを制御します。エクステントからシーケンシャルに読み取られるページの数が innodb_read_ahead_threshold 以上である場合、InnoDB は、次のエクステント全体の非同期先読み操作を開始します。これは 0 から 64 までの任意の値に設定できます。デフォルト値は 56 です。この値が大きいほど、アクセスパターンのチェックは厳密になります。たとえば、この値を 48 に設定すると、InnoDB は、現在のエクステント内の 48 ページが順次にアクセスされた場合にのみ線形先読み要求をトリガーします。この値が 8 である場合、InnoDB は、エクステント内の 8 ページが順次にアクセスされただけでも非同期先読みをトリガーします。このパラメータの値は MySQL 構成ファイルで設定するか、または SET GLOBAL コマンド (これには SUPER 権限が必要です) で動的に変更できます。

ランダム先読みは、すでにバッファープール内に存在するページに基づいて、これらのページが読み取られた順序には関係なく、ページがいつ必要になる可能性があるかを予測する手法です。同じエクステントからの 13 個の連続したページがバッファープール内に見つかった場合、InnoDB は、そのエクステントの残りのページのプリフェッチを行う要求を非同期に発行します。

ランダム先読み機能は、InnoDB Plugin (バージョン 1.0.4) から削除されたため、InnoDB Plugin が InnoDB の「組み込みの」バージョンになった時点では MySQL 5.5.0 に含まれていませんでした。ランダム先読みは、innodb_random_read_ahead 構成オプション (デフォルトでは無効になっています) とともに、MySQL 5.1.59 および 5.5.16 以降でふたたび導入されました。この機能を有効にするには、構成変数 innodb_random_read_ahead を ON に設定します。

SHOW ENGINE INNODB STATUS コマンドは、先読みアルゴリズムの有効性を評価するのに役立つ統計を表示します。統計には、Innodb_buffer_pool_read_ahead および Innodb_buffer_pool_read_ahead_evicted グローバルステータス変数のカウンタ情報が含まれます。これらの情報は、innodb_random_read_ahead 設定を微調整する場合に役立つことがあります。

また、MySQL 5.6 でランダム先読み機能が復活したことにより、SHOW ENGINE INNODB STATUS コマンドには Innodb_buffer_pool_read_ahead_rnd がふたたび含まれています。Innodb_buffer_pool_read_ahead は、その現在の名前を維持しています。(以前のリリースでは、Innodb_buffer_pool_read_ahead_seq として示されていました。)

I/O パフォーマンスの詳細は、セクション8.5.7「InnoDB ディスク I/O の最適化」およびセクション8.11.3「ディスク I/O の最適化」を参照してください。

14.13.1.2 InnoDB バッファープールのフラッシュの頻度の構成

InnoDB は、バッファープールからのダーティーページ (変更されたが、まだデータベースファイルに書き込まれていないページ) のフラッシュなどの特定のタスクをバックグラウンドで実行します。InnoDB は現在、バッファープール内のダーティーページの割合 (%) が innodb_max_dirty_pages_pct を超えた場合にバッファープールページをフラッシュします。

InnoDB は、Redo ログ生成の速度とフラッシュの現在の頻度に基づいて、フラッシュの必要な頻度を推定するアルゴリズムを使用します。その目的は、バッファーのフラッシュアクティビティーが常に、バッファープールを「クリーンな」状態に維持する必要性に対応するようにして、全体的なパフォーマンスを平滑化することにあります。フラッシュ頻度の自動的な調整は、通常の読み取りおよび書き込みアクティビティーに使用可能な I/O 容量がバッファープールの過剰なフラッシュによって制限されるような場合、スループットの突然の低下を回避するのに役立つことがあります。

InnoDB は、そのログファイルを循環的に使用します。ログファイルのある部分を再利用する前に、InnoDB は、Redo エントリがログファイルのその部分に含まれているダーティーバッファープールページをすべてディスクにフラッシュします。このプロセスは、シャープチェックポイントと呼ばれます。書き込みの多いワークロードでは、そのすべてがログファイルに書き込まれる多くの Redo 情報が生成されます。ログファイル内の使用可能なすべての領域が使い果たされると、シャープチェックポイントが発生するため、スループットが一時的に低下します。この状況は、innodb_max_dirty_pages_pct に達していなくても発生する場合があります。

InnoDB は、バッファープール内のダーティーページの数と、Redo が生成されている割合を測定することによってこのようなシナリオを回避する、経験則に基づいたアルゴリズムを使用します。これらの数値に基づいて、InnoDB は、バッファープールから毎秒フラッシュするダーティーページの数を決定します。この自己適応型のアルゴリズムは、ワークロードの突然の変化に対処できます。

内部のベンチマークでも示されているように、このアルゴリズムは一定期間にわたってスループットを維持するだけでなく、全体的なスループットも大幅に向上させることができます。

適応型フラッシュはワークロードの I/O パターンに大きな影響を与える場合があるため、innodb_adaptive_flushing 構成パラメータを使用して、この機能を無効にすることができます。innodb_adaptive_flushing のデフォルト値は TRUE であり、適応型フラッシュアルゴリズムが有効になります。このパラメータの値は MySQL オプションファイル (my.cnf または my.ini) で設定するか、あるいは SET GLOBAL コマンド (これには SUPER 権限が必要です) で動的に変更できます。

InnoDB の I/O パフォーマンスの詳細は、セクション8.5.7「InnoDB ディスク I/O の最適化」を参照してください。

14.13.1.3 バッファープールをスキャンに耐えられるようにする

InnoDB は、LRU アルゴリズムを厳密に使用するのではなく、バッファープールに読み取られたあと二度とアクセスされないデータの量を最小限に抑えるための手法を使用します。目標は、先読みやフルテーブルスキャンによって、その後アクセスされるかどうかわからない新しいブロックが読み取られた場合でも、頻繁にアクセスされるページ (「ホットページ」) が確実にバッファープール内に残るようにすることです。

新しく読み取られたブロックは、LRU リストの途中に挿入されます。新しく読み取られたページはすべて、デフォルトでは LRU リストの末尾から 3/8 にあたる場所に挿入されます。これらのページは、はじめてバッファー

プール内でアクセスされたときに、リストの前面 (直近で使用された端) に移動されます。そのため、アクセスされることがないページは決して LRU リストの前面の部分には移動されず、厳密な LRU アプローチの場合より早く「古く」なります。この配置では、LRU リストが 2 つのセグメントに分割されます。つまり、挿入ポイントの下流にあるページは「古い」とみなされ、LRU のエビクションの望ましい対象になります。

InnoDB バッファープールの内部動作や、その LRU の置き換えアルゴリズムの詳細については、セクション 8.9.1「InnoDB バッファープール」を参照してください。

LRU リスト内の挿入ポイントを制御したり、InnoDB が同じ最適化をテーブルまたはインデックススキャンによってバッファープールに読み取られたブロックにも適用するかどうかを選択したりできます。構成パラメータ innodb_old_blocks_pct は、LRU リスト内の「古い」ブロックの割合 (%) を制御します。innodb_old_blocks_pct のデフォルト値は 37 であり、元の固定された 3/8 の比率に対応します。この値の範囲は、5 (バッファープール内の新しいページが非常に早く古くなります) から 95 (バッファープールの 5% しかホットページとして予約されないため、アルゴリズムがなじみのある LRU の方法に近くなります) までです。

バッファープールを先読みによって混乱した状態にならないように維持する最適化は、テーブルまたはインデックススキャンによる同様の問題も回避できます。これらのスキャンでは通常、データページはすばやく連続して数回アクセスされ、それ以降は二度とアクセスされません。構成パラメータ innodb_old_blocks_time は、あるページにはじめてアクセスしたあと、そのページが LRU リストの前面 (直近で使用された端) に移動されることなくアクセス可能になっている時間ウィンドウ (ミリ秒単位) を指定します。MySQL 5.6.6 より前は、innodb_old_blocks_time のデフォルト値は 0 であり、はじめてバッファープール内でアクセスされたときにページをバッファープールのリストの直近で使用された端に移動するという元の動作に対応します。この値を大きくすると、より多くのブロックがバッファープールから早く古くなる可能性があります。MySQL 5.6.6 の時点で、innodb_old_blocks_time のデフォルト値は、標準でのパフォーマンスを向上させるために 1000 に増やされました。

innodb_old_blocks_pct と innodb_old_blocks_time はどちらも動的かつグローバルであり、MySQL オプションファイル (my.cnf または my.ini) で指定するか、あるいは SET GLOBAL コマンドで実行時に変更できます。この設定を変更するには、SUPER 権限が必要です。

これらのパラメータを設定した場合の効果の測定に役立つように、SHOW ENGINE INNODB STATUS コマンドは追加の統計をレポートします。BUFFER POOL AND MEMORY セクションは次のようになります。

```
Total memory allocated 1107296256; in additional pool allocated 0
Dictionary memory allocated 80360
Buffer pool size   65535
Free buffers       0
Database pages     63920
Old database pages 23600
Modified db pages  34969
Pending reads 32
Pending writes: LRU 0, flush list 0, single page 0
Pages made young 414946, not young 2930673
1274.75 youngs/s, 16521.90 non-youngs/s
Pages read 486005, created 3178, written 160585
2132.37 reads/s, 3.40 creates/s, 323.74 writes/s
Buffer pool hit rate 950 / 1000, young-making rate 30 / 1000 not 392 / 1000
Pages read ahead 1510.10/s, evicted without access 0.00/s
LRU len: 63920, unzip_LRU len: 0
I/O sum[43690]:cur[221], unzip sum[0]:cur[0]
```

- Old database pages は、LRU リストの「古い」セグメント内のページの数です。
- Pages made young と not young はそれぞれ、新しくなった「古い」ページの総数となっていないページの総数です。
- youngs/s と non-young/s はそれぞれ、このコマンドの最後の呼び出しのあと、「古い」ページへのページアクセスによってこのようなページが新しくなった割合となっていない割合です。
- young-making rate と not は、「古い」ページへのアクセスだけでなく、全体的なバッファープールアクセスの点から見た場合、同じ割合を示しています。

> **注記**
>
> InnoDB モニターの出力で示される 1 秒あたりの平均は、現在の時間と InnoDB モニターの出力が最後に出力された時間の間の経過時間に基づいています。

これらのパラメータの効果はハードウェア構成、使用しているデータ、およびワークロードの詳細によって大幅に異なる場合があるため、パフォーマンスが重要な環境や本番環境でこれらの設定を変更する前には、常にベンチマークによってその有効性を確認してください。

ほとんどのアクティビティーが、大規模なスキャンにつながる定期的なバッチレポートクエリーを含む OLTP タイプである混在ワークロード環境では、バッチの実行中に innodb_old_blocks_time の値を設定すると、通常のワークロードのワーキングセットをバッファープール内に維持するのに役立つ場合があります。

バッファープール内に完全には収まらない大きなテーブルをスキャンする場合は、innodb_old_blocks_pct を小さな値に設定すると、1 回しか読み取られないデータがバッファープールの大きな部分を消費することはなくなります。たとえば、innodb_old_blocks_pct=5 を設定すると、1 回しか読み取られないこのデータがバッファープールの 5% に制限されます。

メモリーに収まる小さなテーブルをスキャンする場合は、バッファープール内でページを移動するためのオーバーヘッドが低いため、innodb_old_blocks_pct をデフォルト値のままにするか、あるいは場合によっては (innodb_old_blocks_pct=50 などと) 増やすこともできます。

innodb_old_blocks_time パラメータの効果は、比較的効果の小さい innodb_old_blocks_pct パラメータに比べて予測が困難であり、ワークロードによる変動も大きくなります。最適な値に到達するには、innodb_old_blocks_pct の調整によるパフォーマンス向上が不十分な場合は独自のベンチマークを実施してください。

InnoDB バッファープールの詳細は、セクション8.9.1「InnoDB バッファープール」を参照してください。

14.13.1.4 複数のバッファープールインスタンスの使用

バッファープールが数 G バイトの範囲にあるシステムでは、バッファープールを個別のインスタンスに分割すると、キャッシュされたページに対して異なるスレッドが読み取りおよび書き込みを行うときの競合が減るため、並列性が向上する場合があります。この機能は通常、バッファープールのサイズが数 G バイトの範囲にあるシステムを対象にしています。複数のバッファープールインスタンスは innodb_buffer_pool_instances 構成オプションを使用して構成され、また innodb_buffer_pool_size 値を調整することもできます。

InnoDB バッファープールが大きい場合は、メモリーから取得することによって多くのデータ要求を満足できます。複数のスレッドが一度にバッファープールにアクセスしようとした場合は、ボトルネックが発生する可能性があります。この競合を最小限に抑えるために、複数のバッファープールを有効にすることができます。バッファープールに格納されるか、またはバッファープールから読み取られる各ページは、ハッシュ関数を使用して、いずれかのバッファープールにランダムに割り当てられます。各バッファープールは、独自の空きリスト、フラッシュリスト、LRU、およびバッファープールに接続されたその他のすべてのデータ構造を管理し、独自のバッファープール相互排他ロックによって保護されます。

複数のバッファープールインスタンスを有効にするには、innodb_buffer_pool_instances 構成オプションを 1 (デフォルト) より大きく 64 (最大) までの値に設定します。このオプションは、innodb_buffer_pool_size を 1G バイト以上のサイズに設定した場合にのみ有効になります。指定した合計サイズは、すべてのバッファープール間で分割されます。最高の効率を得るには、innodb_buffer_pool_instances と innodb_buffer_pool_size の組み合わせを、各バッファープールインスタンスが少なくとも 1G バイトになるように指定します。

InnoDB バッファープールの詳細は、セクション8.9.1「InnoDB バッファープール」を参照してください。

14.13.1.5 再起動を高速化するための InnoDB バッファープールのプリロード

特に、大きなInnoDB バッファープールを持つインスタンスの場合は、サーバーを再起動したあとの非常に長いウォームアップ期間を回避するために、サーバーのシャットダウン時に InnoDB バッファープールの状態を保存し、サーバーの起動時にバッファープールを同じ状態にリストアすることができます。

ビジー状態のサーバーを再起動したあと、通常は、InnoDB バッファープール内にあったディスクページがメモリーに戻されるにつれ (同じデータへのクエリーや更新などが実行されるにつれ)、スループットが着実に向上するウォームアップ期間が存在します。バッファープールをシャットダウン前の状態にリストアする機能により、DML 操作での対応する行へのアクセスを待つのではなく、再起動前にバッファープール内にあったディスクページをただちにリロードできるため、ウォームアップ期間が短くなります。I/O 要求を大きなバッチで実行できるため、I/O 全体がより高速になります。ページのロードはバックグラウンドで実行されるため、データベースの起動が遅れることはありません。

バッファープールの状態をシャットダウン時に保存し、起動時にリストアすることに加え、サーバーの実行中にも、バッファープールの状態をいつでも保存およびリストアすることができます。たとえば、定常状態のワークロードの下で安定したスループットに達したあとに、バッファープールの状態を保存できます。その操作の期間中にのみ必要なデータページをバッファープールに移動するレポートまたは保守ジョブを実行したあとや、標準的でないワークロードを使用したその他の一定期間のあとに、バッファープールの以前の状態をリストアできます。

バッファープール自体のサイズは何 G バイトもある可能性がありますが、InnoDB がディスク上に保存するデータはそれに比べて非常に少量です。該当するページを見つけるために必要なテーブルスペース ID とページ ID だけがディスクに保存されます。この情報は、INNODB_BUFFER_PAGE_LRU INFORMATION_SCHEMA

テーブルから取得されます。デフォルトでは、テーブルスペース ID とページ ID のデータは、InnoDB データディレクトリに保存される ib_buffer_pool という名前のファイル内に保存されます。このファイル名は、innodb_buffer_pool_filename 構成パラメータを使用して変更できます。

ベースとなるメカニズムには、ダンプおよびロード操作を実行するためにディスパッチされるバックグラウンドスレッドが含まれています。

圧縮テーブルからのディスクページは、その圧縮された形式でバッファープールにロードされます。DML 操作の過程でページの内容がアクセスされると、圧縮解除が通常どおりに実行されます。圧縮解除は CPU を大量に消費するプロセスであるため、並列性としては、バッファープールのリストア操作を実行する単一スレッドではなく、接続スレッドのいずれかで操作を実行する方が効率的です。

デフォルトでは、バッファープールの状態は、InnoDB データディレクトリに保存される ib_buffer_pool という名前のファイル内に保存されます。このファイル名は、innodb_buffer_pool_filename 構成パラメータを使用して変更できます。

バッファープールの状態の保存

サーバーのシャットダウン時に InnoDB バッファープールの状態を保存するには、次のステートメントを発行します。

```
SET GLOBAL innodb_buffer_pool_dump_at_shutdown=ON;
```

MySQL サーバーの実行中に InnoDB バッファープールの状態を保存するには、次のステートメントを発行します。

```
SET GLOBAL innodb_buffer_pool_dump_now=ON;
```

バッファープールの状態のリストア

サーバーの起動時に InnoDB バッファープールの状態をリストアするには、サーバーを起動するときに --innodb_buffer_pool_load_at_startup オプションを指定します。

```
mysqld --innodb_buffer_pool_load_at_startup=ON;
```

MySQL の実行中に InnoDB バッファープールの状態をリストアするには、次のステートメントを発行します。

```
SET GLOBAL innodb_buffer_pool_load_now=ON;
```

バッファープールのダンプの進行状況の表示

InnoDB バッファープールの状態をディスクに保存しているときに進行状況を表示するには、次のいずれかのオプションを使用します。

```
SHOW STATUS LIKE 'innodb_buffer_pool_dump_status';
```

または

```
SELECT variable_value FROM information_schema.global_status WHERE
variable_name = 'INNODB_BUFFER_POOL_DUMP_STATUS';
```

操作がまだ開始されていない場合は、「not started」が返されます。操作が完了している場合は、完了時間が出力されます (たとえば、Finished at 110505 12:18:02)。操作が進行中である場合は、ステータス情報が表示されます (たとえば、Dumping buffer pool 5/7, page 237/2873)。

バッファープールのロードの進行状況の表示

InnoDB バッファープールをロードしているときに進行状況を表示するには、次のいずれかのオプションを使用します。

```
SHOW STATUS LIKE 'innodb_buffer_pool_load_status';
```

または

```
SELECT variable_value FROM information_schema.global_status WHERE
variable_name = 'INNODB_BUFFER_POOL_LOAD_STATUS';
```

操作がまだ開始されていない場合は、「not started」が返されます。操作が完了している場合は、完了時間が出力されます (たとえば、Finished at 110505 12:23:24)。操作が進行中である場合は、ステータス情報が表示されます (たとえば、Loaded 123/22301 pages)。

バッファープールのロードの中止

バッファープールのロード操作を中止するには、次のステートメントを発行します。

```
SET innodb_buffer_pool_load_abort=ON;
```

14.13.1.6 InnoDB バッファープールのフラッシュのチューニング

構成オプション innodb_flush_neighbors および innodb_lru_scan_depth を使用すると、InnoDB バッファープールに対するフラッシュプロセスの特定の側面を微調整できます。これらのオプションは主に、書き込みの多いワークロードに役立ちます。負荷の高い DML アクティビティーでは、それが十分に積極的でないとフラッシュが遅延して、バッファープールで過剰なメモリーが使用される場合があります。または、そのメカニズムが積極的すぎると、フラッシュによるディスク書き込みによって I/O 容量が飽和する場合があります。理想的な設定は、ワークロード、データのアクセスパターン、およびストレージ構成 (たとえば、データが HDD または SSD デバイスのどちらに格納されているか) によって異なります。

ワークロードの負荷が常時高いシステム、またはワークロードが大幅に変動するシステムでは、複数の構成オプション innodb_adaptive_flushing_lwm、innodb_max_dirty_pages_pct_lwm、innodb_io_capacity_max、および innodb_flushing_avg_loops を使用して InnoDB テーブルに対するフラッシュ動作を微調整できます。これらのオプションは、innodb_adaptive_flushing オプションによって使用される計算式に入力されます。

innodb_adaptive_flushing、innodb_io_capacity、および innodb_max_dirty_pages_pct オプションは、次のオプション innodb_adaptive_flushing_lwm、innodb_io_capacity_max、および innodb_max_dirty_pages_pct_lwm によって制限または拡張されます。

- InnoDB 適応型フラッシュメカニズムは、すべての場合に適切なわけではありません。これがもっとも大きな利点をもたらすのは、Redo ログがいっぱいになるおそれがある場合です。innodb_adaptive_flushing_lwm オプションは、Redo ログ容量の「低位境界値」の割合 (%) を指定します。そのしきい値を超えると、InnoDB は、innodb_adaptive_flushing オプションで指定されていない場合でも適応型フラッシュを有効にします。
- フラッシュアクティビティーが大幅に遅延した場合、InnoDB は、innodb_io_capacity で指定されているより積極的にフラッシュできます。innodb_io_capacity_max は、I/O のスパイクによってサーバーのすべての容量が消費されてしまわないように、このような緊急の状況で使用される I/O 容量の上限を表します。
- InnoDB は、ダーティーページの割合 (%) が innodb_max_dirty_pages_pct の値を超えないように、バッファープールからデータをフラッシュしようとします。innodb_max_dirty_pages_pct のデフォルト値は 75 です。

  > 注記
  >
  > innodb_max_dirty_pages_pct 設定は、フラッシュアクティビティーのターゲットを確立します。フラッシュの頻度には影響を与えません。フラッシュの頻度の管理については、セクション14.13.1.2「InnoDB バッファープールのフラッシュの頻度の構成」を参照してください。

  innodb_max_dirty_pages_pct_lwm オプションは、ダーティーページの比率を制御するために事前フラッシュを有効にし、理想的にはダーティーページの割合 (%) が innodb_max_dirty_pages_pct に達しないようにするダーティーページの割合 (%) を表す、「低位境界値」の値を指定します。innodb_max_dirty_pages_pct_lwm=0 の値を指定すると、「事前フラッシュ」動作が無効になります。

上で参照されたオプションのほとんどは、書き込み負荷の高いワークロードを長期間にわたって実行しており、ディスクへの書き込みを待機している変更に追い付くためにロード時間の削減がほとんどないサーバーにもっとも適しています。

innodb_flushing_avg_loops は、InnoDB が以前に計算されたフラッシュ状態のスナップショットを保持する反復の数を定義します。これは、適応型フラッシュがフォアグラウンドの負荷の変化にどれだけすばやく応答するかを制御します。innodb_flushing_avg_loops に大きな値を設定すると、InnoDB が以前に計算されたスナップショットをより長く保持するため、適応型フラッシュはよりゆっくり応答します。大きな値ではまた、フォアグラウンド作業とバックグラウンド作業の間の正のフィードバックも削減されますが、大きな値を設定する場合は、InnoDB の Redo ログの使用率が 75% (非同期のフラッシュが開始されるハードコードされた制限) に達しないようにすること、および innodb_max_dirty_pages_pct 設定によってダーティーページの数がそのワークロードに適したレベルに維持されるようにすることが重要です。

ワークロードに一貫性があり、innodb_log_file_size が大きく、かつスパイクが小さいために Redo ログ領域の使用率が 75% に達しないシステムでは、フラッシュをできるだけ滑らかな状態に維持するために大きな innodb_flushing_avg_loops 値を使用するようにしてください。負荷のスパイクが極端なシステム、またはログファイルでは多くの領域が提供されないシステムでは、より小さな innodb_flushing_avg_loops 値を検討してくだ

さい。この値を小さくすると、フラッシュで負荷を密接に追跡できるため、Redo ログ領域の使用率が 75% に達しないようにするのに役立ちます。

14.13.2 InnoDB 相互排他ロックおよび読み取り/書き込みロックの実装

MySQL および InnoDB では、実行の複数のスレッドが共有データ構造にアクセスします。InnoDB は、これらのアクセスを相互排他ロックと読み取り/書き込みロックの独自の実装と同期させます。InnoDB は従来、InnoDB 相互排他ロックを使用して読み取り/書き込みロックの内部状態を保護してきました。Unix および Linux プラットフォームでは、IEEE Std 1003.1c (POSIX.1c) にあるように、InnoDB 相互排他ロックの内部状態は Pthreads 相互排他ロックによって保護されます。

多くのプラットフォームには、相互排他ロックと読み取り/書き込みロックを実装するためのより効率的な方法が存在します。アトミック操作を使用すると、多くの場合、Pthreads より効率的に複数のスレッドのアクションを同期させることができます。ロックを取得または解放する各操作をより少ない CPU 命令で実行できるため、共有データ構造へのアクセスのためにスレッドが競合している場合の浪費時間が少なくなります。マルチコアプラットフォーム上では、これがさらにスケーラビリティーを向上させます。

InnoDB は、以前に使用されていた Pthreads のアプローチを使用する代わりに、アトミックメモリーアクセスのための GNU コンパイラコレクション (GCC) によって提供される組み込み関数を使用して相互排他ロックと読み取り/書き込みロックを実装します。より具体的には、GCC バージョン 4.1.2 以降でコンパイルされた InnoDB は、pthread_mutex_t の代わりにアトミックビルトインを使用して InnoDB 相互排他ロックおよび読み取り/書き込みロックを実装します。

32 ビットの Microsoft Windows では、InnoDB は、手で書かれたアセンブラ命令を使用して (読み取り/書き込みロックではなく) 相互排他ロックを実装していました。Microsoft Windows 2000 からは、GCC によって提供される組み込み関数と同様の、インターロックされた変数アクセスのための関数が使用できます。Windows 2000 以降では、InnoDB はインターロックされた関数を使用します。手で書かれた古いアセンブラコードとは異なり、新しい実装では、読み取り/書き込みロックと 64 ビットプラットフォームがサポートされます。

Solaris 10 ではアトミック操作のためのライブラリ関数が導入され、InnoDB は、デフォルトでこれらの関数を使用します。アトミックメモリーアクセスのための GNU コンパイラコレクション (GCC) によって提供される組み込み関数をサポートしていないコンパイラを備えた Solaris 10 上で MySQL がコンパイルされた場合、InnoDB はライブラリ関数を使用します。

この変更によって、マルチコアシステム上の InnoDB のスケーラビリティーが向上します。この機能は、それがサポートされているプラットフォーム上で標準で有効になります。パフォーマンスの向上を利用するためにパラメータやオプションを設定する必要はありません。アトミックメモリーアクセスのための GCC、Windows、または Solaris 関数が使用できないプラットフォームでは、InnoDB は、相互排他ロックと読み取り/書き込みロックを実装するための従来の Pthreads の方法を使用します。

MySQL が起動すると、InnoDB は、アトミックメモリーアクセスが相互排他ロックに使用されるか、相互排他ロックと読み取り/書き込みロックに使用されるか、またはどちらにも使用されないかを示すメッセージをログファイルに書き込みます。適切なツールを使用して InnoDB が構築されており、かつターゲット CPU が必要なアトミック操作をサポートしている場合、InnoDB は、相互排他ロックに組み込み関数を使用します。さらに、スレッド識別子 (pthread_t) で比較およびスワップ操作を使用できる場合、InnoDB は、読み取り/書き込みロックのための命令も使用します。

> **注記**
>
> ソースからビルドしている場合は、そのビルドプロセスがプラットフォームの機能を正しく利用していることを確認してください。

ロックのパフォーマンスへの影響の詳細は、セクション8.10「ロック操作の最適化」を参照してください。

14.13.3 InnoDB のためのメモリーアロケータの構成

InnoDB が開発されたとき、オペレーティングシステムに付属のメモリーアロケータや実行時ライブラリは多くの場合、パフォーマンスとスケーラビリティーに欠けていました。その時点では、マルチコア CPU 用にチューニングされたメモリーアロケータライブラリは存在しませんでした。そのため、InnoDB は、mem サブシステム内に独自のメモリーアロケータを実装しました。このアロケータは単一の相互排他ロックによって保護されており、これがボトルネックになる可能性があります。InnoDB はまた、システムアロケータ (malloc および free) の周りに、同様に単一の相互排他ロックによって保護されているラッパーインタフェースも実装しています。

今日、マルチコアシステムがより広範囲に使用可能になるにつれ、またオペレーティングシステムが成熟するにつれ、オペレーティングシステムに付属のメモリーアロケータに対して大幅な機能強化が行われてきました。新しいメモリーアロケータは以前のものより性能が向上し、またよりスケーラブルになりました。主要な高性能メ

モリーアロケータには、Hoard、libumem、mtmalloc、ptmalloc、tbbmalloc、および TCMalloc が含まれます。ほとんどのワークロードは、特にメモリーの割り当てと解放が頻繁に行われる (マルチテーブル結合などの) 場合、内部の InnoDB 固有のメモリーアロケータではなく、より高度にチューニングされたメモリーアロケータを使用するとメリットがあります。

InnoDB が独自のメモリーアロケータまたはオペレーティングシステムのアロケータのどちらを使用するかは、MySQL オプションファイル (my.cnf または my.ini) 内のシステム構成パラメータ innodb_use_sys_malloc の値を設定することによって制御できます。ON または 1 (デフォルト) に設定されている場合、InnoDB はメモリープールを自ら管理するのではなく、ベースとなるシステムの malloc および free 関数を使用します。このパラメータは動的ではなく、システムが起動された場合にのみ有効になります。InnoDB メモリーアロケータを引き続き使用するには、innodb_use_sys_malloc を 0 に設定します。

InnoDB メモリーアロケータが無効になっている場合、InnoDB は、パラメータ innodb_additional_mem_pool_size の値を無視します。InnoDB メモリーアロケータは、システムメモリーアロケータにフォールバックしなくても割り当て要求を満たせるように、追加のメモリープールを使用します。InnoDB メモリーアロケータが無効になっている場合、このような割り当て要求はすべて、システムメモリーアロケータによって満たされます。

動的リンクを使用する Unix ライクなシステムでは、メモリーアロケータの置き換えは、環境変数 LD_PRELOAD または LD_LIBRARY_PATH がそのアロケータを実装している動的ライブラリを指すようにするだけの簡単な操作で済むことがあります。ほかのシステムでは、ある程度の再リンクが必要になる可能性があります。選択したメモリーアロケータライブラリのドキュメントを参照してください。

システムメモリーアロケータが使用されている (innodb_use_sys_malloc が ON である) 場合、InnoDB はすべてのメモリー使用を追跡することができないため、SHOW ENGINE INNODB STATUS コマンドの出力内のセクション「BUFFER POOL AND MEMORY」の「Total memory allocated」にはバッファープールの統計のみが含まれます。mem サブシステムまたは ut_malloc を使用して割り当てられたメモリーはすべて除外されます。

> **注記**
>
> innodb_use_sys_malloc と innodb_additional_mem_pool_size は MySQL 5.6.3 では非推奨であり、将来のリリースで削除される予定です。

InnoDB のメモリー使用のパフォーマンスへの影響の詳細は、セクション8.9「バッファリングとキャッシュ」を参照してください。

14.13.4 InnoDB 変更バッファリングの構成

テーブルに対して INSERT、UPDATE、および DELETE 操作が実行されると、インデックス付きカラムの値 (特に副キーの値) は多くの場合、ソートされた順番にならないため、セカンダリインデックスを最新の状態にするために大量の I/O が必要になります。InnoDB は、挿入バッファーを備えています。これは、関連するページがバッファープール内にない場合にセカンダリインデックスエントリへの変更をキャッシュすることにより、そのページをディスクから読み取らないことで I/O 操作を回避するものです。バッファリングされた変更は、そのページがバッファープールにロードされたときにマージされ、更新されたページはあとで通常のメカニズムを使用してディスクにフラッシュされます。InnoDB のメインスレッドは、それらのバッファリングされた変更を、サーバーがほぼアイドル状態にあるときと低速シャットダウン中にマージします。

これにより、ディスクの読み取りや書き込みが少なくなる場合があるため、この機能は、I/O に依存するワークロード (たとえば、一括挿入などの大量の DML 操作を含むアプリケーション) にとってもっとも価値があります。

ただし、挿入バッファーはバッファープールの一部を占有するため、データページをキャッシュするために使用できるメモリーが減少します。ワーキングセットがバッファープールにほぼ収まる場合や、テーブルのセカンダリインデックスが比較的少ない場合は、挿入バッファリングを無効にすると役立つことがあります。ワーキングセットがバッファープールに完全に収まる場合は、挿入バッファリングはバッファープール内にないページにしか適用されないため、追加で課せられるオーバーヘッドはありません。

InnoDB がどの程度挿入バッファリングを実行するかは、システム構成パラメータ innodb_change_buffering を使用して制御できます。挿入操作、削除操作 (インデックスレコードが最初に削除対象としてマークされるとき)、およびパージ操作 (インデックスレコードが物理的に削除されるとき) でのバッファリングを有効または無効にすることができます。更新操作は、挿入と削除の組み合わせとして表されます。MySQL 5.5 以降では、デフォルト値が inserts から all に変更されました。

innodb_change_buffering の許可される値は次のとおりです。

- all

  デフォルト値: バッファーの挿入、削除のマーキング操作、およびパージ。

- none

  どの操作もバッファリングしません。

- inserts

  挿入操作をバッファリングします。

- deletes

  削除のマーキング操作をバッファリングします。

- changes

  挿入と削除のマーキングの両方をバッファリングします。

- purges

  バックグラウンドで実行される物理的な削除操作をバッファリングします。

このパラメータの値は MySQL オプションファイル (my.cnf または my.ini) で設定するか、あるいは SET GLOBAL コマンド (これには SUPER 権限が必要です) で動的に変更できます。この設定を変更すると、新しい操作のバッファリングに影響を与えます。すでにバッファリングされたエントリのマージは影響を受けません。

INSERT、UPDATE、および DELETE ステートメントの高速化の詳細は、セクション8.2.2「DML ステートメントの最適化」を参照してください。

14.13.5 InnoDB のスレッド並列性の構成

InnoDB は、オペレーティングシステムスレッドを使用して、ユーザートランザクションからの要求を処理します。(トランザクションは、コミットまたはロールバックする前に、InnoDB に多数の要求を発行する可能性があります。)コンテキストスイッチングが効率的な、マルチコアプロセッサを備えた最新のオペレーティングシステムおよびサーバーでは、並列スレッドの数を制限することなく、ほとんどのワークロードが適切に動作します。MySQL 5.5 以降でのスケーラビリティーの向上によって、InnoDB の内部の並列実行中のスレッドの数を制限する必要性は低下します。

スレッド間のコンテキストスイッチングを最小限に抑えることが役立つ状況では、InnoDB はいくつかの手法を使用して、並列実行中のオペレーティングシステムスレッドの数 (したがって、一度に処理される要求の数) を制限できます。InnoDB がユーザーセッションからの新しい要求を受信したとき、並列実行中のスレッドの数が事前に定義された制限に達している場合、その新しい要求は再試行の前に短時間だけスリープします。スリープのあとに再スケジュールできない要求は先入れ先出しキューに入れられ、最終的に処理されます。ロックを待機しているスレッドは、並列実行中のスレッドの数にカウントされません。

並列スレッドの数は、構成パラメータ innodb_thread_concurrency を設定することによって制限できます。実行中のスレッドの数がこの制限に達すると、追加のスレッドはキューに入れられる前に、構成パラメータ innodb_thread_sleep_delay で設定されたマイクロ秒数だけスリープします。

以前は、innodb_thread_sleep_delay の最適な値を見つけるには実験が必要であり、その最適な値もワークロードによって変化する可能性がありました。MySQL 5.6.3 以降では、構成オプション innodb_adaptive_max_sleep_delay を innodb_thread_sleep_delay に許可するもっとも大きな値に設定することができ、InnoDB が、現在のスレッドスケジューリングアクティビティーに応じて innodb_thread_sleep_delay を上または下に自動的に調整します。この動的な調整は、システムにかかる負荷が軽い期間や、システムがほぼ容量いっぱいで動作している期間に、スレッドスケジューリングメカニズムがスムーズに機能するのに役立ちます。

innodb_thread_concurrency のデフォルト値や、並列スレッドの数に対する暗黙的なデフォルトの制限は、MySQL および InnoDB のさまざまなリリースで変更されてきました。現在、デフォルトでは並列実行中のスレッドの数に対して制限がないように、innodb_thread_concurrency のデフォルト値は 0 です。

InnoDB がスレッドをスリープさせるのは、並列スレッドの数が制限されている場合だけであることに注意してください。スレッドの数に対して制限がない場合は、すべてが均等に競合してスケジュールされます。つまり、innodb_thread_concurrency が 0 である場合は、innodb_thread_sleep_delay の値は無視されます。

スレッドの数に対して制限がある (innodb_thread_concurrency > 0 である) 場合、InnoDB は、1 つの SQL ステートメントの実行中に発行された複数の要求が innodb_thread_concurrency で設定された制限に従うことなく InnoDB に入ることを許可することによって、コンテキストスイッチングのオーバーヘッドを削減します。SQL ステートメント (結合など) は InnoDB 内の複数の行操作で構成されている可能性があるため、InnoDB は、スレッドが最小限のオーバーヘッドで繰り返しスケジュールされることを許可する指定された数の「チケット」を割り当てます。

新しい SQL ステートメントが開始されたとき、スレッドにはチケットがないため、innodb_thread_concurrency に従う必要があります。スレッドが InnoDB に入ることを許可されると、そのスレッドには、行操作を実行するためにあとで InnoDB に入るときに使用できる複数のチケットが割り当てられます。それらのチケットが使い果たされた場合、スレッドは削除され、ふたたび innodb_thread_concurrency に従います。それにより、そのスレッドが、待機中のスレッドの先入れ先出しキューに戻される可能性があります。スレッドがふたたび InnoDB に入ることを許可されると、チケットが再度割り当てられます。割り当てられるチケットの数は、グローバルオプション innodb_concurrency_tickets (デフォルトでは 5000、5.6.6 より前は 500) で指定されます。ロックを待機しているスレッドには、そのロックが使用可能になるとチケットが 1 つ与えられます。

これらの変数の正しい値は、環境やワークロードによって異なります。アプリケーションでどのような値が機能するかを確認するには、さまざまな値を試してください。並列実行中のスレッドの数を制限する前に、マルチコアおよびマルチプロセッサコンピュータ上の InnoDB のパフォーマンスを向上させる可能性のある構成オプション (innodb_adaptive_hash_index など) を確認してください。

MySQL のスレッド処理に関する一般的なパフォーマンス情報については、セクション8.11.5.1「MySQL のクライアント接続のためのスレッドの使用方法」を参照してください。

14.13.6 InnoDB バックグラウンド I/O スレッドの数の構成

InnoDB は、さまざまなタイプの I/O 要求を処理するためにバックグラウンドスレッドを使用します。構成パラメータ innodb_read_io_threads および innodb_write_io_threads を使用して、データページに対する読み取りおよび書き込み I/O を処理するバックグラウンドスレッドの数を構成できます。これらのパラメータはそれぞれ、読み取りおよび書き込み要求に使用されるバックグラウンドスレッドの数を示します。これらは、サポートされるすべてのプラットフォームで有効です。これらのパラメータの値は、MySQL オプションファイル (my.cnf または my.ini) で設定できます。動的に変更することはできません。これらのパラメータのデフォルト値は 4 であり、許可される値の範囲は 1-64 です。

この変更の目的は、InnoDB をハイエンドのシステム上でよりスケーラブルにすることです。各バックグラウンドスレッドは、保留中の I/O 要求を最大 256 個処理できます。バックグラウンド I/O の主なソースは、先読み要求です。InnoDB は、バックグラウンドスレッドのほとんどが均等に作業を共有するような方法で、受信要求の負荷のバランスをとろうとします。InnoDB はまた、要求を 1 つに合体できる可能性を増やすために、読み取り要求を同じエクステントから同じスレッドに割り当てようとします。ハイエンドの I/O サブシステムを使用しており、SHOW ENGINE INNODB STATUS で 64 × innodb_read_io_threads 個を超える保留中の読み取り要求が表示される場合は、innodb_read_io_threads の値を増やすと役立つことがあります。

InnoDB の I/O パフォーマンスの詳細は、セクション8.5.7「InnoDB ディスク I/O の最適化」を参照してください。

14.13.7 グループコミット

InnoDB では、ACID 準拠のほかのデータベースエンジンと同様に、トランザクションの Redo ログをそれがコミットされる前にフラッシュします。従来より、InnoDB はグループコミット機能を使用してこのような複数のフラッシュ要求を 1 つにグループ化し、コミットごとに 1 つのフラッシュを回避してきました。グループコミットでは、ほぼ同じ時間にコミットする複数のユーザートランザクションのコミットアクションを実行するために InnoDB がログファイルに 1 つの書き込みを発行するため、スループットが大幅に向上します。

InnoDB でのグループコミットは MySQL 4.x まで機能し、InnoDB Plugin を含む MySQL 5.1、および MySQL 5.5 以降でふたたび機能しています。MySQL 5.0 での分散トランザクションおよび 2 フェーズコミット (2PC) に対するサポートの導入は、InnoDB のグループコミット機能に混乱をもたらしました。現在、この問題は解決されています。

InnoDB 内部のグループコミット機能は、MySQL の 2 フェーズコミットプロトコルとともに機能します。グループコミット機能の再有効化によって、MySQL バイナリログおよび InnoDB ログファイル内のコミットの順序が以前と同じであることが完全に保証されます。これは、InnoDB 1.0.4 (つまり、MySQL 5.1 の InnoDB Plugin) 以降で MySQL Enterprise Backup 製品を使用することが完全に安全であることを示しています。

グループコミットは透過的です。この大幅なパフォーマンス向上を利用するために何もする必要はありません。

COMMIT やその他のトランザクション操作のパフォーマンスの詳細は、セクション8.5.2「InnoDB トランザクション管理の最適化」を参照してください。

14.13.8 InnoDB マスタースレッドの I/O レートの構成

InnoDB でのマスタースレッドは、さまざまなタスクをバックグラウンドで実行するスレッドです。これらのタスクは、そのほとんどがダーティーページのバッファープールからのフラッシュや、変更の挿入バッファーから適切なセカンダリインデックスへの書き込みなどの I/O 関連のタスクです。マスタースレッドは、これらのタスク

を、サーバーの通常の動作に悪影響を与えない方法で実行しようとします。つまり、使用可能な空き I/O 帯域幅を推定し、自身のアクティビティーをチューニングしてこの空き容量を利用しようとします。従来より、InnoDB は、サーバーの合計 I/O 容量として 100 IOPs (1 秒あたりの入力/出力操作数) のハードコードされた値を使用してきました。

パラメータ innodb_io_capacity は、InnoDB で使用できる全体的な I/O 容量を示します。このパラメータはほぼ、システムが 1 秒あたりに実行できる I/O 操作の数に設定するようにしてください。この値は、システム構成によって異なります。innodb_io_capacity が設定されている場合、マスタースレッドは設定済みの値に基づいて、バックグラウンドタスクに使用できる I/O 帯域幅を推定します。この値を 100 に設定すると、従来の動作に戻ります。

innodb_io_capacity の値は、100 以上の任意の数値に設定できます。デフォルト値は 200 であり、標準的な最新の I/O デバイスのパフォーマンスが MySQL の初期のころより向上していることを反映しています。通常、消費者レベルのストレージデバイス (最大 7200 RPM のハードドライブなど) には、以前のデフォルトである 100 近辺の値が適しています。より高速なハードドライブ、RAID 構成、および SSD は、値を大きくするとメリットがあります。

innodb_io_capacity 設定は、すべてのバッファープールインスタンスに対する合計の制限です。ダーティーページがフラッシュされるとき、innodb_io_capacity 制限は、バッファープールインスタンス間で均等に分割されます。詳細は、innodb_io_capacity システム変数の説明を参照してください。

このパラメータの値は MySQL オプションファイル (my.cnf または my.ini) で設定するか、あるいは SET GLOBAL コマンド (これには SUPER 権限が必要です) で動的に変更できます。

以前、InnoDB マスタースレッドは、必要なパージ操作もすべて実行していました。MySQL 5.6.5 以降では、これらの I/O 操作はほかのバックグラウンドスレッドに移行され、そのスレッドの数は innodb_purge_threads 構成オプションによって制御されます。

InnoDB の I/O パフォーマンスの詳細は、セクション8.5.7「InnoDB ディスク I/O の最適化」を参照してください。

14.13.9 InnoDB スピンループでの PAUSE 命令の使用

InnoDB 内部の同期には、スピンループがよく使用されます。InnoDB は待機中に、InnoDB プロセスやスレッドがオペレーティングシステムによって再スケジュールされないように、命令の緊密なループを繰り返し実行します。スピンループの実行が速すぎると、システムリソースが浪費され、それによってトランザクションスループットのパフォーマンスが低下します。最新のプロセッサは、スピンループで使用する PAUSE 命令を実装しているため、プロセッサをより効率的にすることができます。

InnoDB は、このような命令が使用可能なすべてのプラットフォーム上のスピンループで PAUSE 命令を使用します。この手法により、CPU に依存するワークロードの全体的なパフォーマンスが向上するだけでなく、スピンループの実行中の消費電力が最小限に抑えられるという利点も追加されます。

このパフォーマンス向上を利用するために何もする必要はありません。

InnoDB のロック操作に関するパフォーマンスの考慮事項については、セクション8.10「ロック操作の最適化」を参照してください。

14.13.10 スピンロックのポーリングの構成

InnoDB の多くの相互排他ロックや読み書きロックは短時間だけ予約されます。マルチコアシステムでは、スレッドがスリープの前に相互排他ロックまたは読み書きロックをしばらくの間取得できるかどうかを連続して確認すると、より効率的になる場合があります。このポーリング期間中に相互排他ロックまたは読み書きロックが使用可能になった場合、そのスレッドは同じタイムスライス内でただちに続行できます。ただし、共有オブジェクトの複数のスレッドが頻繁にポーリングしすぎると、「キャッシュのピンポン」が発生し、各プロセッサが互いのキャッシュの一部を無効にする場合があります。InnoDB は、以降の各ポーリング間にランダムな時間待つことによってこの問題を最小限に抑えます。この遅延は、ビジーループとして実装されます。

相互排他ロックまたは読み書きロックのテスト間の最大の遅延は、パラメータ innodb_spin_wait_delay を使用して制御できます。遅延ループの期間は、C コンパイラやターゲットプロセッサによって異なります。(100MHz Pentium の時代、この遅延の単位は 1 マイクロ秒でした。)すべてのプロセッサコアが高速なキャッシュメモリーを共有するシステムでは、この最大の遅延を短くするか、または innodb_spin_wait_delay=0 を設定してビジーループを完全に無効にすることができます。複数のプロセッサチップを備えたシステムでは、キャッシュを無効にすると重大な影響を与える場合があるため、最大の遅延を増やすことができます。

innodb_spin_wait_delay のデフォルト値は 6 です。スピン待ちの遅延は、MySQL オプションファイル (my.cnf または my.ini) で指定したり、コマンド SET GLOBAL innodb_spin_wait_delay=delay を使用して実行時に変更した

りできる動的なグローバルパラメータです。ここで、delay は望ましい最大の遅延です。この設定を変更するには、SUPER 権限が必要です。

InnoDB のロック操作に関するパフォーマンスの考慮事項については、セクション8.10「ロック操作の最適化」を参照してください。

14.13.11 InnoDB の MySQL パフォーマンススキーマとの統合

MySQL 5.5 を含む InnoDB 1.1 から、MySQL パフォーマンススキーマ機能を使用して InnoDB の特定の内部操作をプロファイルできます。このタイプのチューニングは主に、MySQL のパフォーマンスの制限をプッシュしたり、MySQL のソースコードを読んだり、パフォーマンスのボトルネックを克服するための最適化方法を評価したりする上級ユーザー向けです。DBA はまた、この機能を容量計画に使用することにより、標準的なワークロードのときに CPU、RAM、およびディスクストレージの特定の組み合わせでパフォーマンスのボトルネックが発生するかどうかを確認し、発生する場合は、システムの一部の容量を増やすことでパフォーマンスを向上させることができるかどうかを判断することもできます。

この機能を使用して InnoDB のパフォーマンスを検査するには:

- セクション22.2「パフォーマンススキーマ構成」で説明されているように、パフォーマンススキーマ機能が使用可能で、かつ有効になっている状態で MySQL 5.5 以降を実行している必要があります。パフォーマンススキーマ機能によってある程度のパフォーマンスオーバーヘッドが導入されるため、この機能は本番システムではなく、テストまたは開発システムで使用するようにしてください。
- たとえば、performance_schema データベース内のテーブルをクエリーする方法など、パフォーマンススキーマ機能の使用方法に全般的に精通している必要があります。
- 次の種類の InnoDB オブジェクトは、対応する performance_schema テーブルをクエリーすることによって検査します。InnoDB に関連した項目はすべて、EVENT_NAME カラム内に部分文字列 innodb を含んでいます。

  *_instances テーブルの定義については、セクション22.9.3「パフォーマンススキーマインスタンステーブル」を参照してください。*_summary_* テーブルの定義については、セクション22.9.9「パフォーマンススキーマサマリーテーブル」を参照してください。thread テーブルの定義については、セクション22.9.10「パフォーマンススキーマのその他のテーブル」を参照してください。*_current_* および *_history_* テーブルの定義については、セクション22.9.4「パフォーマンススキーマ待機イベントテーブル」を参照してください。

  - mutex_instances テーブル内の相互排他ロック。(ここには、InnoDB バッファープールに関連した相互排他ロックと読み書きロックは含まれていません。同じことが、SHOW ENGINE INNODB MUTEX コマンドの出力にも適用されます。)
  - rwlock_instances テーブル内の読み書きロック。
  - file_instances、file_summary_by_event_name、および file_summary_by_instance テーブル内のファイル I/O 操作。
  - PROCESSLIST テーブル内のスレッド。
- パフォーマンステスト中に、events_waits_current および events_waits_history_long テーブル内のパフォーマンスデータを検査します。InnoDB 関連のオブジェクトに特に関心がある場合は、句 WHERE EVENT_NAME LIKE '%innodb%' を使用してそれらのエントリだけを確認します。それ以外の場合は、MySQL サーバー全体のパフォーマンス統計を検査します。

MySQL パフォーマンススキーマの詳細は、第22章「MySQL パフォーマンススキーマ」を参照してください。

14.13.12 複数のロールバックセグメントによるスケーラビリティーの向上

MySQL 5.5 を含む InnoDB 1.1 から、並列トランザクションの制限が大幅に拡張されたため、大容量システムに影響を与えていた InnoDB のロールバックセグメントでのボトルネックが解消されました。この制限は、何らかのデータを変更する並列トランザクションに適用されます。読み取り専用トランザクションは、その最大数に対してカウントされません。

1 つのロールバックセグメントが現在 128 個のセグメントに分割され、そのそれぞれで、書き込みを実行するトランザクションを最大 1023 個サポートできるため、合計で約 128K 個の並列トランザクションがサポートされます。元のトランザクション制限は 1023 でした。

各トランザクションはいずれかのロールバックセグメントに割り当てられ、存続期間中は、そのロールバックセグメントに結び付けられたままになります。この拡張により、スケーラビリティー (並列トランザクションの数の増加) とパフォーマンス (各種のトランザクションがロールバックセグメントにアクセスした場合の競合の減少) の両方が向上します。

この機能を利用するために新しいデータベースまたはテーブルを作成したり、何かを再構成したりする必要はありません。MySQL 5.1 以前からのアップグレードの前、またはその後しばらくして低速シャットダウンを実行する必要があります。低速シャットダウンを実行したあとにはじめて再起動すると、システムテーブルスペース内の必要な変更が InnoDB によって自動的に行われます。

ワークロードが 1023 個の並列トランザクションの元の制限によって制約されなかった場合は、構成オプション innodb_rollback_segments を設定することによって、MySQL インスタンスまたはセッション内で使用されるロールバックセグメントの数を減らすことができます。

高いトランザクション負荷の下での InnoDB のパフォーマンスの詳細は、セクション8.5.2「InnoDB トランザクション管理の最適化」を参照してください。

14.13.13 InnoDB のパージスケジューリングの構成

InnoDB が自動的に実行するパージ操作 (ガベージコレクションの一種) は現在、マススレッドではなく、1 つ以上の個別のスレッドで実行されます。この変更により、メインのデータベース操作がバックグラウンドで実行される保守作業とは独立に実行されるため、スケーラビリティーが向上します。

この機能を制御するには、構成オプション innodb_purge_threads の値を増やします。DML アクションが 1 つのテーブルまたは少数のテーブルに集中している場合は、ビジーテーブルにアクセスするためにスレッドが互いに競合することのないように、この設定を低い値に維持します。DML 操作が多数のテーブルに分散している場合は、この設定を増やします。その最大値は 32 です。

関連する別の構成オプションとして、デフォルト値が 20 で、最大値が 5000 である innodb_purge_batch_size があります。このオプションは主に、パージ操作の実験やチューニングを対象にしており、標準的なユーザーにとって魅力的なものではありません。

InnoDB の I/O パフォーマンスの詳細は、セクション8.5.7「InnoDB ディスク I/O の最適化」を参照してください。

14.13.14 InnoDB の読み取り専用トランザクションの最適化

MySQL 5.6.4 の時点では、InnoDB は、読み取り専用であることがわかっているトランザクションのトランザクション ID (TRX_ID フィールド) の設定に関連したオーバーヘッドを回避できます。トランザクション ID は、書き込み操作または ロック読み取り (SELECT ... FOR UPDATE など) を実行する可能性のあるトランザクションにのみ必要です。不必要なトランザクション ID を削除すると、クエリーや DML ステートメントによって読み取りビューが構築されるたびに参照される内部データ構造のサイズが削減されます。

現在、InnoDB は、次の場合に読み取り専用トランザクションを検出します。

- トランザクションが START TRANSACTION READ ONLY ステートメントで開始された場合。この場合は、データベース (InnoDB、MyISAM、またはその他のタイプのテーブル) に対して変更を行おうとするとエラーが発生し、そのトランザクションは読み取り専用状態のままになります。

  ```
  ERROR 1792 (25006): Cannot execute statement in a READ ONLY transaction.
  ```

  ただし、読み取り専用トランザクションでのセッション固有の一時テーブルの変更や、それらのテーブルに対するロッククエリーの発行は、その変更やロックがほかのどのトランザクションにも表示されないため引き続き可能です。

- autocommit 設定がオンになっているため、トランザクションが 1 つのステートメントであることが保証され、そのトランザクションを構成している 1 つのステートメントが「非ロック」の SELECT ステートメントである場合。つまり、FOR UPDATE または LOCK IN SHARED MODE 句を使用しない SELECT です。

そのため、レポートジェネレータなどの読み取りの多いアプリケーションの場合は、InnoDB のクエリーを START TRANSACTION READ ONLY および COMMIT の内部にグループ化するか、SELECT ステートメントを実行する前に autocommit 設定をオンにするか、または単純にどの DML ステートメントにもクエリーが組み込まれないようにすることにより、それらのクエリーのシーケンスをチューニングできます。

START TRANSACTION および autocommit については、セクション13.3.1「START TRANSACTION、COMMIT、および ROLLBACK 構文」を参照してください。

> **注記**
>
> 自動コミット、非ロック、および読み取り専用 (AC-NL-RO) として承認されたトランザクションは、InnoDB の特定の内部データ構造から除外されるため、SHOW ENGINE

> INNODB STATUS の出力には表示されません。これらのトランザクションは、情報スキーマでのみ表示されます。

14.13.15 チェックサムの高速化のための CRC32 チェックサムアルゴリズムの使用

MySQL 5.6.3 で導入された CRC32 チェックサムアルゴリズムは、ブロックを 1 回につき 32 ビットスキャンします。これは、ブロックを 1 回につき 8 ビットスキャンする InnoDB のチェックサムアルゴリズムに対する改善です。CRC32 チェックサムアルゴリズムは、innodb_checksum_algorithm=crc32 を設定することによって有効にすることができます。

> 注記
>
> 同様に MySQL 5.6.3 で導入された innodb_checksum_algorithm 構成パラメータによって、innodb_checksums 構成パラメータが置き換えられます。詳細は、innodb_checksum_algorithm のドキュメントを参照してください。

CRC32 アルゴリズムが有効になっている場合、InnoDB によってディスクに書き込まれるデータブロックのチェックサムフィールドには、以前とは異なる値が含まれています。テーブルスペース内のブロックが CRC32 チェックサムアルゴリズムを使用するように変更されたあと、関連付けられたテーブルを以前のバージョンの MySQL で読み取ることはできません。

新しい MySQL インスタンスを設定するときに、すべての InnoDB データが CRC32 チェックサムアルゴリズムを使用して作成されている場合は、innodb_checksum_algorithm=strict_crc32 設定を使用できます。これは、古い値と新しい値の両方をサポートするための余分なチェックサム計算を実行しないため、crc32 設定より高速である可能性があります。

innodb_checksum_algorithm のデフォルト値は MySQL 5.6.6 で innodb から crc32 に変更されましたが、以前の MySQL バージョンへのダウングレード中の InnoDB データファイルの互換性向上のため、および MySQL Enterprise Backup で使用するために 5.6.7 で innodb に戻されました。検出された制限には、次のものが含まれます。

- CRC32 チェックサムを含む .ibd ファイルは、5.6.3 より前の MySQL バージョンへのダウングレード中に問題が発生する可能性があります。MySQL 5.6.3 以降では、ディスクからブロックを読み取るとき、そのブロックの新しいチェックサム値と古いチェックサム値のどちらも正しいとして認識します。それにより、アルゴリズムの設定には関係なく、アップグレードおよびダウングレード中にそのデータブロックの互換性を保証します。新しいチェックサム値で書き込まれたデータが 5.6.3 より前のレベルの MySQL によって処理された場合は、破損しているとしてレポートされる可能性があります。
- 3.8.0 までのバージョンの MySQL Enterprise Backup は、CRC32 チェックサムを使用するテーブルスペースのバックアップをサポートしていません。MySQL Enterprise Backup は、CRC32 チェックサムのサポートを 3.8.1 で (いくつかの制限付きで) 追加しています。詳細は、MySQL Enterprise Backup 3.8.1 の変更履歴を参照してください。

14.13.16 オプティマイザ統計

14.13.16.1 永続的オプティマイザ統計のパラメータの構成

計画安定性は、もっとも大きく、かつもっとも重要なクエリーの望ましい目標です。InnoDB は、オプティマイザがもっとも効率的なクエリー実行計画を容易に見つけることができるように、常に InnoDB テーブルごとの統計を計算してきました。これらの統計を永続的なものにすることが可能になったため、特定のクエリーのインデックス使用状況や結合順序が変更される可能性は低くなります。

この機能はデフォルトでオンになっており、構成オプション innodb_stats_persistent で有効になります。

innodb_stats_persistent_sample_pages 構成オプションを設定することによって、統計を収集するために実行されるサンプリングの量を制御します。

構成オプション innodb_stats_auto_recalc は、テーブルに (行の 10% を超える) 大幅な変更が加えられた場合は常に統計を自動的に計算するかどうかを決定します。

> 注記
>
> 統計の自動再計算 (これは、バックグラウンドで実行されます) には非同期の性質があるため、innodb_stats_auto_recalc が有効になっていたとしても、テーブルの 10% を超

> える部分に影響を与える DML 操作を実行した直後に統計が再計算されるとは限りません。場合によっては、統計の再計算が数秒遅れる可能性があります。テーブルの大きな部分を変更した直後に最新の統計が必要な場合は、統計の同期的な (フォアグラウンド) 再計算を開始するために ANALYZE TABLE を実行してください。

innodb_stats_auto_recalc が無効になっている場合は、インデックス付きカラムへの大幅な変更を行なったあと、該当する各テーブルに対して ANALYZE TABLE ステートメントを発行することによってオプティマイザ統計の精度を確保してください。このステートメントは、代表的なデータがテーブルにロードされたあとにセットアップスクリプトで実行したり、インデックス付きカラムの内容が DML 操作によって大幅に変更されたあとに定期的に、または低アクティビティーの時間帯にスケジュールに従って実行したりすることができます。既存のテーブルに新しいインデックスが追加された場合は、innodb_stats_auto_recalc の値には関係なく、インデックス統計が計算されて innodb_index_stats テーブルに追加されます。

> 注意
>
> 新しいインデックスが作成されたときに統計が確実に収集されるように、innodb_stats_auto_recalc オプションを有効にするか、または永続的統計モードが有効になっているときに新しいインデックスを作成するたびに ANALYZE TABLE を実行してください。

テーブルを作成する前にグローバルレベルで innodb_stats_persistent、innodb_stats_auto_recalc オプションを設定するか、または CREATE TABLE および ALTER TABLE ステートメントで STATS_PERSISTENT、STATS_AUTO_RECALC、および STATS_SAMPLE_PAGES 句を使用して、システム全体の設定をオーバーライドし、個々のテーブルの永続的統計を構成することができます。

以前は、これらの統計は各サーバーの再起動やその他の一部の操作のあとにクリアされ、テーブルが次回アクセスされたときに再計算されていました。これらの統計は、次回には異なる推定値が生成される可能性のあるランダムなサンプリング手法を使用して計算されるため、実行計画において異なる選択が行われ、そのためにクエリーパフォーマンスが変動します。

定期的に消去される以前の統計収集方法に戻すには、コマンド ALTER TABLE tbl_name STATS_PERSISTENT=0 を実行します。関連情報については、セクション14.13.16.2「非永続的オプティマイザ統計のパラメータの構成」を参照してください。

InnoDB オプティマイザ統計でサンプリングされるページの数の構成

MySQL クエリーオプティマイザは、インデックスの相対的な選択性に基づいて、キー分布に関する推定された統計を使用して実行計画のためのインデックスを選択します。ANALYZE TABLE などの操作を行うと、InnoDB は、インデックスのカーディナリティーを推定するためにテーブル上の各インデックスからランダムなページをサンプリングします。(この手法は、ランダムダイブと呼ばれます。)

統計の推定値の品質を制御する (それにより、クエリーオプティマイザへの情報を改善する) ために、実行時に設定できるパラメータ innodb_stats_persistent_sample_pages を使用して、サンプリングされるページの数を変更できます。

innodb_stats_persistent_sample_pages のデフォルト値は 20 です。一般的なガイドラインとして、次の問題が発生した場合は、このパラメータを変更することを考慮してください。

1. EXPLAIN の出力で示されているように、統計の精度が十分でないため、オプティマイザが次善の計画を選択する。統計の精度は、インデックスの実際のカーディナリティー (インデックスカラムに対して SELECT DISTINCT を実行することにより返されます) を、mysql.innodb_index_stats 永続的統計テーブルに示されている推定値と比較することによってチェックできます。

   統計の精度が十分でないことが確認された場合は、統計の推定値が十分な精度になるまで innodb_stats_persistent_sample_pages の値を増やすようにしてください。ただし、innodb_stats_persistent_sample_pages を大きくしすぎると、ANALYZE TABLE の実行が遅くなる可能性があります。

2. ANALYZE TABLE が遅すぎる。この場合は、ANALYZE TABLE の実行時間が許容可能になるまで innodb_stats_persistent_sample_pages を減らすようにしてください。ただし、この値を小さくしすぎると、精度の低い統計および次善のクエリー実行計画という最初の問題につながる可能性があります。

   統計の精度と ANALYZE TABLE の実行時間のバランスをとることができない場合は、ANALYZE TABLE の複雑さを減らすためにテーブル内のインデックス付きカラムの数を減らすか、またはパーティションの数を制限することを考慮してください。また、主キーカラムは一意でない各インデックスに付加されるため、テーブルの主キー内のカラム数について考慮することも重要です。

関連情報については、セクション14.13.17「InnoDB テーブルに対する ANALYZE TABLE の複雑さの推定」を参照してください。

InnoDB 永続的統計テーブル

永続的統計機能は、innodb_table_stats および innodb_index_stats という名前の、mysql データベース内の内部的に管理されているテーブルに依存します。これらのテーブルは、すべてのインストール、アップグレード、およびソースからのビルド手順で自動的に設定されます。

表 14.8 innodb_table_stats のカラム

カラム名	説明
database_name	データベース名
table_name	テーブル名、パーティション名、またはサブパーティション名
last_update	InnoDB が最後にこの行を更新した時間を示すタイムスタンプ
n_rows	テーブル内の行数
clustered_index_size	プライマリインデックスのサイズ (ページ数)
sum_of_other_index_sizes	その他の (プライマリ以外の) インデックスの合計サイズ (ページ数)

表 14.9 innodb_index_stats のカラム

カラム名	説明
database_name	データベース名
table_name	テーブル名、パーティション名、またはサブパーティション名
index_name	インデックス名
last_update	InnoDB が最後にこの行を更新した時間を示すタイムスタンプ
stat_name	stat_value カラムに値がレポートされている統計の名前
stat_value	stat_name カラムで名前が指定されている統計の値
sample_size	stat_value カラムに示されている推定値のサンプリングされるページの数
stat_description	stat_name カラムで名前が指定されている統計の説明

次の例に示すように、innodb_table_stats テーブルと innodb_index_stats テーブルのどちらにも、InnoDB が最後にインデックス統計を更新した時間を示す last_update カラムが含まれています。

```
mysql> select * from innodb_table_stats \G
*************************** 1. row ***************************
        database_name: sakila
           table_name: actor
          last_update: 2014-05-28 16:16:44
              n_rows: 200
  clustered_index_size: 1
sum_of_other_index_sizes: 1
...
```

```
mysql> select * from innodb_index_stats \G
*************************** 1. row ***************************
  database_name: sakila
     table_name: actor
     index_name: PRIMARY
    last_update: 2014-05-28 16:16:44
      stat_name: n_diff_pfx01
     stat_value: 200
    sample_size: 1
    ...
```

innodb_table_stats および innodb_index_stats テーブルは通常のテーブルであるため、手動で更新できます。統計を手動で更新する機能により、データベースを変更することなく、特定のクエリー最適化計画やテスト代替計画を強制的に実行することが可能になります。統計を手動で更新した場合は、更新された統計が MySQL でリロードされるように、FLUSH TABLE tbl_name コマンドを発行します。

InnoDB 永続的統計テーブルの例

innodb_table_stats テーブルには、テーブルごとに 1 行が含まれています。収集されるデータを次の例に示します。

テーブル t1 には、プライマリインデックス (カラム a、b)、セカンダリインデックス (カラム c、d)、および一意のインデックス (カラム e、f) が含まれています。

```
CREATE TABLE t1 (
a INT, b INT, c INT, d INT, e INT, f INT,
PRIMARY KEY (a, b), KEY i1 (c, d), UNIQUE KEY i2uniq (e, f)
) ENGINE=INNODB;
```

5 行のサンプルデータを挿入したあと、テーブルは次のようになります。

```
mysql> SELECT * FROM t1;
+---+---+------+------+------+------+
| a | b | c    | d    | e    | f    |
+---+---+------+------+------+------+
| 1 | 1 |   10 |   11 |  100 |  101 |
| 1 | 2 |   10 |   11 |  200 |  102 |
| 1 | 3 |   10 |   11 |  100 |  103 |
| 1 | 4 |   10 |   12 |  200 |  104 |
| 1 | 5 |   10 |   12 |  100 |  105 |
+---+---+------+------+------+------+
5 rows in set (0.00 sec)
```

統計をただちに更新するには、ANALYZE TABLE を実行します (innodb_stats_auto_recalc が有効になっている場合、変更されるテーブル行の 10% のしきい値に達したと仮定すると、統計は数秒以内に自動的に更新されます)。

```
mysql> ANALYZE TABLE t1;
+---------+---------+----------+----------+
| Table   | Op      | Msg_type | Msg_text |
+---------+---------+----------+----------+
| test.t1 | analyze | status   | OK       |
+---------+---------+----------+----------+
1 row in set (0.02 sec)
```

テーブル t1 のテーブル統計には、InnoDB が最後にテーブル統計を更新した時間 (2014-03-14 14:36:34)、テーブル内の行数 (5)、クラスタ化されたインデックスのサイズ (1 ページ)、およびほかのインデックスの合計サイズ (2 ページ) が示されます。

```
mysql> SELECT * FROM mysql.innodb_table_stats WHERE table_name like 't1'\G
*************************** 1. row ***************************
           database_name: test
              table_name: t1
             last_update: 2014-03-14 14:36:34
                   n_rows: 5
     clustered_index_size: 1
sum_of_other_index_sizes: 2
1 row in set (0.00 sec)
```

innodb_index_stats テーブルには、インデックスごとに複数の行が含まれています。innodb_index_stats テーブル内の各行は、stat_name カラムで名前が指定され、stat_description カラムで説明されている特定のインデックス統計に関連したデータを示します。例:

```
mysql> SELECT index_name, stat_name, stat_value, stat_description
    -> FROM mysql.innodb_index_stats WHERE table_name like 't1';
+------------+--------------+------------+--------------------------------+
| index_name | stat_name    | stat_value | stat_description               |
+------------+--------------+------------+--------------------------------+
| PRIMARY    | n_diff_pfx01 |          1 | a                              |
| PRIMARY    | n_diff_pfx02 |          5 | a,b                            |
| PRIMARY    | n_leaf_pages |          1 | Number of leaf pages in the index |
| PRIMARY    | size         |          1 | Number of pages in the index   |
| i1         | n_diff_pfx01 |          1 | c                              |
| i1         | n_diff_pfx02 |          2 | c,d                            |
| i1         | n_diff_pfx03 |          2 | c,d,a                          |
| i1         | n_diff_pfx04 |          5 | c,d,a,b                        |
| i1         | n_leaf_pages |          1 | Number of leaf pages in the index |
| i1         | size         |          1 | Number of pages in the index   |
| i2uniq     | n_diff_pfx01 |          2 | e                              |
| i2uniq     | n_diff_pfx02 |          5 | e,f                            |
| i2uniq     | n_leaf_pages |          1 | Number of leaf pages in the index |
| i2uniq     | size         |          1 | Number of pages in the index   |
+------------+--------------+------------+--------------------------------+
14 rows in set (0.00 sec)
```

stat_name カラムには、次のタイプの統計が示されます。

- size: stat_name=size である場合、stat_value カラムには、インデックス内のページの総数が表示されます。

- n_leaf_pages: stat_name=n_leaf_pages である場合、stat_value カラムには、インデックス内のリーフページの数が表示されます。
- n_diff_pfxNN: stat_name=n_diff_pfx01 である場合、stat_value カラムには、インデックスの最初のカラム内の固有の値の数が表示されます。stat_name=n_diff_pfx02 である場合、stat_value カラムには、インデックスの最初の 2 つのカラム内の固有の値の数が表示されます。以下も同様です。さらに、stat_name=n_diff_pfxNN である場合、stat_description カラムには、カウントされるインデックスカラムのカンマ区切りリストが示されます。

カーディナリティーデータを提供する n_diff_pfxNN の統計をさらに詳細に示すために、t1 テーブルの例を考えてみます。次に示すように、t1 テーブルは、プライマリインデックス (カラム a、b)、セカンダリインデックス (カラム c、d)、および一意のインデックス (カラム e、f) で作成されます。

```
CREATE TABLE t1 (
  a INT, b INT, c INT, d INT, e INT, f INT,
  PRIMARY KEY (a, b), KEY i1 (c, d), UNIQUE KEY i2uniq (e, f)
) ENGINE=INNODB;
```

5 行のサンプルデータを挿入したあと、テーブルは次のようになります。

```
mysql> SELECT * FROM t1;
+---+---+------+------+------+------+
| a | b | c    | d    | e    | f    |
+---+---+------+------+------+------+
| 1 | 1 |   10 |   11 |  100 |  101 |
| 1 | 2 |   10 |   11 |  200 |  102 |
| 1 | 3 |   10 |   11 |  100 |  103 |
| 1 | 4 |   10 |   12 |  200 |  104 |
| 1 | 5 |   10 |   12 |  100 |  105 |
+---+---+------+------+------+------+
5 rows in set (0.00 sec)
```

stat_name LIKE 'n_diff%' である index_name、stat_name、stat_value、および stat_description をクエリーすると、次の結果セットが返されます。

```
mysql> SELECT index_name, stat_name, stat_value, stat_description
    -> FROM mysql.innodb_index_stats
    -> WHERE table_name like 't1' AND stat_name LIKE 'n_diff%';
+------------+--------------+------------+------------------+
| index_name | stat_name    | stat_value | stat_description |
+------------+--------------+------------+------------------+
| PRIMARY    | n_diff_pfx01 |          1 | a                |
| PRIMARY    | n_diff_pfx02 |          5 | a,b              |
| i1         | n_diff_pfx01 |          1 | c                |
| i1         | n_diff_pfx02 |          2 | c,d              |
| i1         | n_diff_pfx03 |          2 | c,d,a            |
| i1         | n_diff_pfx04 |          5 | c,d,a,b          |
| i2uniq     | n_diff_pfx01 |          2 | e                |
| i2uniq     | n_diff_pfx02 |          5 | e,f              |
+------------+--------------+------------+------------------+
8 rows in set (0.00 sec)
```

PRIMARY インデックスの場合は、2 つの n_diff% 行があります。行数は、インデックス内のカラム数に等しくなります。

> **注記**
>
> 一意でないインデックスの場合は、InnoDB によって主キーのカラムが付加されます。

- index_name=PRIMARY および stat_name=n_diff_pfx01 である場合、stat_value は 1 です。これは、インデックスの最初のカラム (カラム a) 内に固有の値が 1 つ存在することを示します。カラム a 内の固有の値の数は、テーブル t1 内のカラム a のデータを表示することによって確認されます。ここには、固有の値が 1 つ存在します (1)。カウントされるカラム (a) は、結果セットの stat_description カラムに示されています。
- index_name=PRIMARY および stat_name=n_diff_pfx02 である場合、stat_value は 5 です。これは、インデックスの 2 つのカラム (a,b) 内に固有の値が 5 つ存在することを示します。カラム a および b 内の固有の値の数は、テーブル t1 内のカラム a および b のデータを表示することによって確認されます。ここには、固有の値が 5 つ存在します: (1,1)、(1,2)、(1,3)、(1,4)、および (1,5)。カウントされるカラム (a,b) は、結果セットの stat_description カラムに示されています。

セカンダリインデックス (i1) の場合は、4 つの n_diff% 行があります。セカンダリインデックスとして定義されているカラムは 2 つ (c,d) しかありませんが、セカンダリインデックスの n_diff% 行は 4 つあります。これは、一意でないインデックスにはすべて InnoDB サフィクスによって主キーが付加されるためです。その結果、セカンダリインデックスカラム (c,d) と主キーカラム (a,b) の両方を反映して、2 つではなく 4 つの n_diff% 行があります。

- index_name=i1 および stat_name=n_diff_pfx01 である場合、stat_value は 1 です。これは、インデックスの最初のカラム (カラム c) 内に固有の値が 1 つ存在することを示します。カラム c 内の固有の値の数は、テーブル t1 内のカラム c のデータを表示することによって確認されます。ここには、固有の値が 1 つ存在します: (10)。カウントされるカラム (c) は、結果セットの stat_description カラムに示されています。

- index_name=i1 および stat_name=n_diff_pfx02 である場合、stat_value は 2 です。これは、インデックスの最初の 2 つのカラム (c,d) 内に固有の値が 2 つ存在することを示します。カラム c および d 内の固有の値の数は、テーブル t1 内のカラム c および d のデータを表示することによって確認されます。ここには、固有の値が 2 つ存在します: (10,11) および (10,12)。カウントされるカラム (c,d) は、結果セットの stat_description カラムに示されています。

- index_name=i1 および stat_name=n_diff_pfx03 である場合、stat_value は 2 です。これは、インデックスの最初の 3 つのカラム (c,d,a) 内に固有の値が 2 つ存在することを示します。カラム c、d、および a 内の固有の値の数は、テーブル t1 内のカラム c、d、および a のデータを表示することによって確認されます。ここには、固有の値が 2 つ存在します: (10,11,1) および (10,12,1)。カウントされるカラム (c,d,a) は、結果セットの stat_description カラムに示されています。

- index_name=i1 および stat_name=n_diff_pfx04 である場合、stat_value は 5 です。これは、インデックスの 4 つのカラム (c,d,a,b) 内に固有の値が 5 つ存在することを示します。カラム c、d、a、および b 内の固有の値の数は、テーブル t1 内のカラム c、d、a、および b のデータを表示することによって確認されます。ここには、固有の値が 5 つ存在します: (10,11,1,1)、(10,11,1,2)、(10,11,1,3)、(10,12,1,4)、および (10,12,1,5)。カウントされるカラム (c,d,a,b) は、結果セットの stat_description カラムに示されています。

一意のインデックス (i2uniq) の場合は、2 つの n_diff% 行があります。

- index_name=i2uniq および stat_name=n_diff_pfx01 である場合、stat_value は 2 です。これは、インデックスの最初のカラム (カラム e) 内に固有の値が 2 つ存在することを示します。カラム e 内の固有の値の数は、テーブル t1 内のカラム e のデータを表示することによって確認されます。ここには、固有の値が 2 つ存在します: (100) および (200)。カウントされるカラム (e) は、結果セットの stat_description カラムに示されています。

- index_name=i2uniq および stat_name=n_diff_pfx02 である場合、stat_value は 5 です。これは、インデックスの 2 つのカラム (e,f) 内に固有の値が 5 つ存在することを示します。カラム e および f 内の固有の値の数は、テーブル t1 内のカラム e および f のデータを表示することによって確認されます。ここには、固有の値が 5 つ存在します: (100,101)、(200,102)、(100,103)、(200,104)、および (100,105)。カウントされるカラム (e,f) は、結果セットの stat_description カラムに示されています。

innodb_index_stats テーブルを使用したインデックスサイズの取得

テーブル、パーティション、またはサブパーティションのインデックスのサイズは、innodb_index_stats テーブルを使用して取得できます。次の例では、テーブル t1 のインデックスサイズが取得されています。テーブル t1 の定義および対応するインデックス統計については、InnoDB 永続的統計テーブルの例を参照してください。

```
mysql> SELECT SUM(stat_value) pages, index_name,
  -> SUM(stat_value)*@@innodb_page_size size
  -> FROM mysql.innodb_index_stats WHERE table_name='t1'
  -> AND stat_name = 'size' GROUP BY index_name;
+-------+------------+-------+
| pages | index_name | size  |
+-------+------------+-------+
|     1 | PRIMARY    | 16384 |
|     1 | i1         | 16384 |
|     1 | i2uniq     | 16384 |
+-------+------------+-------+
3 rows in set (0.00 sec)
```

パーティションまたはサブパーティションの場合は、変更された WHERE 句を含む同じクエリーを使用してインデックスサイズを取得できます。たとえば、次のクエリーは、テーブル t1 のパーティションのインデックスサイズを取得します。

```
mysql> SELECT SUM(stat_value) pages, index_name,
  -> SUM(stat_value)*@@innodb_page_size size
  -> FROM mysql.innodb_index_stats WHERE table_name like 't1#P%'
  -> AND stat_name = 'size' GROUP BY index_name;
```

14.13.16.2 非永続的オプティマイザ統計のパラメータの構成

MySQL 5.6.6 の時点では、オプティマイザ統計はデフォルトで永続的であり、innodb_stats_persistent 構成オプションで有効になります。このセクションでは、innodb_stats_persistent=OFF であるとき、または STATS_PERSISTENT=0 で個々のテーブルが作成または変更されるときに適用される非永続的オプティマイザ統

計の構成について説明します。永続的オプティマイザ統計については、セクション14.13.16.1「永続的オプティマイザ統計のパラメータの構成」を参照してください。

MySQL クエリーオプティマイザは、インデックスの相対的な選択性に基づいて、キー分布に関する推定された統計を使用して実行計画のためのインデックスを選択します。特定の操作を行うと、InnoDB は、インデックスのカーディナリティーを推定するためにテーブル上の各インデックスからランダムなページをサンプリングします。(この手法は、ランダムダイブと呼ばれます。)これらの操作には、ANALYZE TABLE ステートメント、SHOW TABLE STATUS ステートメント、および再起動後のテーブルへのはじめてのアクセスが含まれます。

統計の推定値の品質を制御する (それにより、クエリーオプティマイザへの情報を改善する) ために、パラメータ innodb_stats_transient_sample_pages を使用して、サンプリングされるページの数を変更できます。サンプリングされるページのデフォルト数は 8 です。これは、正確な推定値を生成するには十分ではなく、クエリーオプティマイザによる不適切なインデックス選択につながる可能性があります。この手法は、大きなテーブルや、結合で使用されるテーブルの場合に特に重要です。このようなテーブルに対する不必要なフルテーブルスキャンが、パフォーマンスの重大な問題になる場合があります。このようなクエリーのチューニングに関するヒントについては、セクション8.2.1.20「フルテーブルスキャンを回避する方法」を参照してください。

グローバルパラメータ innodb_stats_transient_sample_pages は、実行時に設定できます。

> 注記
>
> innodb_stats_persistent=0 である場合は、innodb_stats_transient_sample_pages の値がすべての InnoDB テーブルおよびインデックスのインデックスサンプリングに影響を与えます。インデックスのサンプルサイズを変更すると、次の潜在的に重大な影響が発生します。
>
> - 1 や 2 などの小さな値では、カーディナリティーの不正確な推定値が生成される可能性があります。
> - innodb_stats_transient_sample_pages 値を大きくすると、必要なディスク読み取りが増える可能性があります。8 よりはるかに大きい値 (たとえば、100) では、テーブルを開いたり、SHOW TABLE STATUS を実行したりするための速度が大幅に遅くなる可能性があります。
> - オプティマイザが、インデックスの選択性の異なる推定値に基づいて、非常に異なるクエリー計画を選択する可能性があります。

SHOW TABLE STATUS や SHOW INDEX などのメタデータステートメントの実行時や、INFORMATION_SCHEMA.TABLES または INFORMATION_SCHEMA.STATISTICS テーブルへのアクセス時に統計が更新されるようにするには、ステートメント SET GLOBAL innodb_stats_on_metadata=ON (または 0) を実行します。

> 注記
>
> 永続的オプティマイザ統計が MySQL 5.6.6 でデフォルトで有効になったとき、innodb_stats_on_metadata のデフォルト設定は OFF に変更されました。この変数を有効にすると、多数のテーブルまたはインデックスを含むスキーマへのアクセス速度が低下したり、InnoDB テーブルに関するクエリーの実行計画の安定性が低下したりする可能性があります。

--auto-rehash 設定がオン (デフォルト) の状態で mysql クライアントが起動されると、すべての InnoDB テーブルが開かれ、関連付けられたすべてのインデックスの統計がふたたび推定されます。mysql クライアントの起動時間を改善するために、--disable-auto-rehash オプションを使用して auto-rehash をオフにすることができます。auto-rehash 機能は、対話ユーザーのためのデータベース、テーブル、およびカラム名の自動名前補完を有効にします。

あるシステムで innodb_stats_transient_sample_pages のどのような値が最適に機能したとしても、このオプションを設定し、その値のままにします。過剰な I/O を必要とせずに、データベース内のすべてのテーブルに対して適度に正確な推定値を生成する値を選択してください。統計は ANALYZE TABLE の実行時以外のさまざまな時間に自動的に再計算されるため、インデックスのサンプルサイズを増やし、ANALYZE TABLE を実行してから、サンプルサイズをふたたび減らしても意味がありません。innodb_stats_transient_sample_pages の大きな値で実行されている ANALYZE によって計算されたより正確な統計があとで消去される可能性があります。

小さなテーブルは一般に、大きなテーブルに比べて、必要なインデックスサンプルが少なくなります。データベースに多数の大きなテーブルが含まれている場合は、ほとんどが小さなテーブルである場合より大きな innodb_stats_transient_sample_pages 値を使用することを考慮してください。

14.13.17 InnoDB テーブルに対する ANALYZE TABLE の複雑さの推定

InnoDB テーブルに対する ANALYZE TABLE の複雑さは、次のものに依存します。

- innodb_stats_persistent_sample_pages で定義される、サンプリングされるページの数。
- テーブル内のインデックス付きカラムの数
- パーティションの数。テーブルにパーティションが存在しない場合、パーティションの数は 1 であるとみなされます。

これらのパラメータを使用すると、ANALYZE TABLE の複雑さを推定するための概略の計算式は次のようになります。

innodb_stats_persistent_sample_pages の値 * テーブル内のインデックス付きカラムの数 * パーティションの数

通常は、この結果の値が大きいほど、ANALYZE TABLE の実行時間も大きくなります。

> **注記**
>
> innodb_stats_persistent_sample_pages は、グローバルレベルでサンプリングされるページの数を定義します。個々のテーブルのサンプリングされるページの数を設定するには、CREATE TABLE または ALTER TABLE で STATS_SAMPLE_PAGES オプションを使用します。詳細は、セクション14.13.16.1「永続的オプティマイザ統計のパラメータの構成」を参照してください。
>
> innodb_stats_persistent=OFF である場合、サンプリングされるページの数は innodb_stats_transient_sample_pages で定義されます。詳細は、セクション 14.13.16.2「非永続的オプティマイザ統計のパラメータの構成」を参照してください。

ANALYZE TABLE の複雑さを推定するためのより詳細なアプローチを示すために、次の例を考えてみます。

ビッグオー表記では、ANALYZE TABLE の複雑さは次のように記述されます。

```
O(n_sample
 * (n_cols_in_uniq_i
    + n_cols_in_non_uniq_i
    + n_cols_in_pk * (1 + n_non_uniq_i))
 * n_part)
```

ここでは:

- n_sample は、サンプリングされるページの数 (innodb_stats_persistent_sample_pages で定義されます)
- n_cols_in_uniq_i は、すべての一意のインデックス内のすべてのカラムの総数 (主キーカラムはカウントしない)
- n_cols_in_non_uniq_i は、すべての一意でないインデックス内のすべてのカラムの総数
- n_cols_in_pk は、主キー内のカラム数 (主キーが定義されていない場合、InnoDB は単一カラムの主キーを内部的に作成します)
- n_non_uniq_i は、テーブル内の一意でないインデックスの数
- n_part は、パーティションの数。パーティションが定義されていない場合、そのテーブルは単一パーティションであるとみなされます。

ここで、主キー (2 つのカラム)、一意のインデックス (2 つのカラム)、および 2 つの一意でないインデックス (それぞれ 2 つのカラム) を持つ次のテーブル (テーブル t) を考えてみます。

```
CREATE TABLE t (
 a INT,
 b INT,
 c INT,
 d INT,
 e INT,
 f INT,
 g INT,
 h INT,
 PRIMARY KEY (a, b),
 UNIQUE KEY i1uniq (c, d),
 KEY i2nonuniq (e, f),
 KEY i3nonuniq (g, h)
);
```

上で説明したアルゴリズムに必要なカラムとインデックスデータについて、テーブル t の mysql.innodb_index_stats 永続的インデックス統計テーブルにクエリーします。n_diff_pfx% の統計には、各インデックスに対してカウントされるカラムが示されます。たとえば、カラム a および b は、主キーのインデックスに対してカウントされます。一意でないインデックスの場合は、ユーザー定義のカラムに加えて、主キーカラム (a,b) がカウントされます。

> **注記**
>
> InnoDB 永続的統計テーブルの詳細は、セクション14.13.16.1「永続的オプティマイザ統計のパラメータの構成」を参照してください。

```
SELECT index_name, stat_name, stat_description
FROM mysql.innodb_index_stats
WHERE
database_name='test' AND
table_name='t' AND
stat_name like 'n_diff_pfx%';

+------------+--------------+------------------+
| index_name | stat_name    | stat_description |
+------------+--------------+------------------+
| PRIMARY    | n_diff_pfx01 | a                |
| PRIMARY    | n_diff_pfx02 | a,b              |
| i1uniq     | n_diff_pfx01 | c                |
| i1uniq     | n_diff_pfx02 | c,d              |
| i2nonuniq  | n_diff_pfx01 | e                |
| i2nonuniq  | n_diff_pfx02 | e,f              |
| i2nonuniq  | n_diff_pfx03 | e,f,a            |
| i2nonuniq  | n_diff_pfx04 | e,f,a,b          |
| i3nonuniq  | n_diff_pfx01 | g                |
| i3nonuniq  | n_diff_pfx02 | g,h              |
| i3nonuniq  | n_diff_pfx03 | g,h,a            |
| i3nonuniq  | n_diff_pfx04 | g,h,a,b          |
+------------+--------------+------------------+
```

上に示したインデックス統計データとテーブル定義に基づいて、次の値を確認できます。

- n_cols_in_uniq_i (すべての一意のインデックス内のすべてのカラムの総数、主キーカラムはカウントしない) は 2 (c および d)
- n_cols_in_non_uniq_i (すべての一意でないインデックス内のすべてのカラムの総数) は 4 (e、f、g、および h)
- n_cols_in_pk (主キー内のカラム数) は 2 (a および b)
- n_non_uniq_i (テーブル内の一意でないインデックスの数) は 2 (i2nonuniq および i3nonuniq)
- n_part (パーティションの数) は 1。

これで、スキャンされるリーフページの数を決定するために innodb_stats_persistent_sample_pages * (2 + 4 + 2 * (1 + 2)) * 1 を計算できます。innodb_stats_persistent_sample_pages が 20 のデフォルト値に設定されており、かつページサイズがデフォルトの 16 KiB (innodb_page_size=16384) である場合は、テーブル t に対して 20 * 12 * 16384 バイト、つまり約 4 MiB が読み取られると推定できます。

> **注記**
>
> 一部のリーフページはすでにバッファープール内にキャッシュされている可能性があるため、4 MiB のすべてがディスクから読み取られるとは限りません。

14.14 InnoDB INFORMATION_SCHEMA テーブル

このセクションでは、InnoDB INFORMATION_SCHEMA テーブルについて、その使用例とともに説明します。

InnoDB INFORMATION_SCHEMA テーブルは、InnoDB ストレージエンジンのさまざまな側面に関するメタデータ、ステータス情報、および統計を提供します。INFORMATION_SCHEMA データベースで SHOW TABLES ステートメントを発行することによって、InnoDB INFORMATION_SCHEMA テーブルのリストを表示できます。

```
mysql> SHOW TABLES FROM INFORMATION_SCHEMA LIKE 'INNODB%';
```

テーブル定義については、セクション21.29「InnoDB の INFORMATION_SCHEMA テーブル」を参照してください。MySQL INFORMATION_SCHEMA データベースに関連した一般的な情報については、第21章「INFORMATION_SCHEMA テーブル」を参照してください。

14.14.1 圧縮に関する InnoDB INFORMATION_SCHEMA テーブル

圧縮が全体としてどれだけ適切に機能しているかを把握するための、圧縮に関する InnoDB INFORMATION_SCHEMA テーブルのペアとして、次の 2 つがあります。

- INNODB_CMP および INNODB_CMP_RESET には、圧縮操作の数および圧縮の実行に費やされた時間に関する情報が含まれています。
- INNODB_CMPMEM および INNODB_CMP_RESET には、圧縮のためにメモリーが割り当てられる方法に関する情報が含まれています。

14.14.1.1 INNODB_CMP および INNODB_CMP_RESET

INNODB_CMP および INNODB_CMP_RESET テーブルには、セクション14.7「InnoDB 圧縮テーブル」で説明されている圧縮テーブルに関連した操作に関するステータス情報が含まれています。PAGE_SIZE カラムは、圧縮されたページサイズをレポートします。

これらの 2 つのテーブルの内容は同じですが、INNODB_CMP_RESET から読み取ると、圧縮および圧縮解除操作に関する統計がリセットされます。たとえば、INNODB_CMP_RESET の出力を 60 分に 1 回アーカイブした場合は、1 時間ごとの統計が表示されます。INNODB_CMP の出力を (INNODB_CMP_RESET を読み取らないように注意して) モニターした場合は、InnoDB が起動されてからの累積された統計が表示されます。

テーブル定義については、セクション21.29.1「INFORMATION_SCHEMA INNODB_CMP および INNODB_CMP_RESET テーブル」を参照してください。

14.14.1.2 INNODB_CMPMEM および INNODB_CMPMEM_RESET

INNODB_CMPMEM および INNODB_CMPMEM_RESET テーブルには、バッファープール内に存在する圧縮ページに関するステータス情報が含まれています。圧縮テーブルおよびバッファープールの使用の詳細は、セクション14.7「InnoDB 圧縮テーブル」を参照してください。INNODB_CMP および INNODB_CMP_RESET テーブルでは、圧縮に関するより役立つ統計が提供されます。

内部の詳細

InnoDB は、バディーアロケータシステムを使用して、1K バイトから 16K バイトまでのさまざまなサイズのページに割り当てられたメモリーを管理します。ここで説明されている 2 つのテーブルの各行は、1 つのページサイズに対応します。

INNODB_CMPMEM および INNODB_CMPMEM_RESET テーブルの内容は同じですが、INNODB_CMPMEM_RESET から読み取ると、再配置操作に関する統計がリセットされます。たとえば、INNODB_CMPMEM_RESET の出力を 60 分に 1 回アーカイブした場合は、1 時間ごとの統計が表示されます。INNODB_CMPMEM_RESET を読み取らないように注意して、代わりに INNODB_CMPMEM の出力をモニターした場合は、InnoDB が起動されてからの累積された統計が表示されます。

テーブル定義については、セクション21.29.3「INFORMATION_SCHEMA INNODB_CMPMEM および INNODB_CMPMEM_RESET テーブル」を参照してください。

14.14.1.3 圧縮情報スキーマテーブルの使用

例 14.10 圧縮情報スキーマテーブルの使用

圧縮テーブルを含むデータベースからのサンプル出力を次に示します (セクション14.7「InnoDB 圧縮テーブル」、INNODB_CMP、INNODB_CMP_PER_INDEX、および INNODB_CMPMEM を参照してください)。

次の表は、軽いワークロード下にある INFORMATION_SCHEMA.INNODB_CMP の内容を示しています。バッファープールに含まれている唯一の圧縮ページサイズは 8K です。カラム COMPRESS_TIME および UNCOMPRESS_TIME が 0 であるため、ページの圧縮または圧縮解除で消費された時間は統計がリセットされてから 1 秒未満でした。

page size	compress ops	compress ops ok	compress time	uncompress ops	uncompress time
1024	0	0	0	0	0
2048	0	0	0	0	0
4096	0	0	0	0	0
8192	1048	921	0	61	0

page size	compress ops	compress ops ok	compress time	uncompress ops	uncompress time
16384	0	0	0	0	0

INNODB_CMPMEM によると、バッファープール内には 6169 個の圧縮された 8K バイトページが存在します。割り当てられているほかのブロックサイズは 64 バイトだけです。INNODB_CMPMEM 内のもっとも小さい PAGE_SIZE は、対応する圧縮解除されたページがバッファープール内に存在しない圧縮ページのブロックディスクリプタとして使用されます。このようなページが 5910 個存在することがわかります。また、間接的には、259 (6169-5910) 個の圧縮ページもバッファープール内に圧縮解除された形式で存在することがわかります。

次の表は、軽いワークロード下にある INFORMATION_SCHEMA.INNODB_CMPMEM の内容を示しています。圧縮ページのためのメモリーアロケータの断片化のために、一部のメモリー SUM(PAGE_SIZE*PAGES_FREE)=6784 は使用できません。これは、小さなメモリー割り当て要求が、バディーアロケーションシステムを使用して (メインのバッファープールから割り当てられる 16K ブロックから始めて) より大きなブロックを分割することによって満たされるためです。断片化がこのように少ないのは、より大きな隣接した空きブロックを形成するために、割り当てられた一部のブロックが再配置 (コピー) されたためです。この SUM(PAGE_SIZE*RELOCATION_OPS) バイトのコピーで消費された時間は 1 秒未満でした ((SUM(RELOCATION_TIME)=0))。

page size	pages used	pages free	relocation ops	relocation time
64	5910	0	2436	0
128	0	1	0	0
256	0	0	0	0
512	0	1	0	0
1024	0	0	0	0
2048	0	1	0	0
4096	0	1	0	0
8192	6169	0	5	0
16384	0	0	0	0

14.14.2 InnoDB INFORMATION_SCHEMA トランザクションおよびロックテーブル

3 つの InnoDB INFORMATION_SCHEMA テーブルを使用すると、トランザクションのモニタリングや、可能性のあるロックの問題の診断が容易になります。これらの 3 つのテーブルは、INNODB_TRX、INNODB_LOCKS、および INNODB_LOCK_WAITS です。

- INNODB_TRX: 現在 InnoDB の内部で実行中のすべてのトランザクションに関する情報が含まれています。これには、そのトランザクションがロックを待機しているかどうか、そのトランザクションがいつ開始されたか、そのトランザクションが実行している特定の SQL ステートメントなどが含まれます。

- INNODB_LOCKS: 別のトランザクションがロックを解放するのを待機している InnoDB 内の各トランザクション (INNODB_TRX.TRX_STATE='LOCK WAIT') は、ただ 1 つの「ブロックしているロック要求」によってブロックされます。そのブロックしているロック要求は、互換性がないモードにある別のトランザクションによって保持されている行ロックまたはテーブルロックに対するものです。待機しているトランザクションまたはブロックされているトランザクションは、ほかのトランザクションがコミットまたはロールバックして、要求されたロックを解放するまで処理を続行できません。INNODB_LOCKS には、ブロックされているトランザクションごとに、そのトランザクションが要求し、かつ待機している各ロックを記述した 1 行が含まれています。INNODB_LOCKS にはまた、ロックを保持しているトランザクションの状態 ('RUNNING'、'LOCK WAIT'、'ROLLING BACK'、または 'COMMITTING') にかかわらず、別のトランザクションをブロックしているロックごとの 1 行も含まれています。トランザクションをブロックしているロックは常に、要求されたロックのモードとは互換性のないモード (読み取りと書き込み、共有と排他的など) に保持されています。

- INNODB_LOCK_WAITS: このテーブルを使用すると、特定のロックをどのトランザクションが待機しているか、または特定のトランザクションがどのロックを待機しているかがわかります。このテーブルには、ブロックされているトランザクションごとに、そのトランザクションが要求したロックと、その要求をブロックしているロックを示す 1 つ以上の行が含まれています。REQUESTED_LOCK_ID はトランザクションが要求しているロックを示し、BLOCKING_LOCK_ID は、最初のトランザクションの続行を妨げている (別のトランザクションによって保持されている) ロックを示します。ブロックされている特定のどのトランザクションでも、INNODB_LOCK_WAITS 内のすべての行が REQUESTED_LOCK_ID については同じ値を、また BLOCKING_LOCK_ID については異なる値を持っています。

14.14.2.1 InnoDB トランザクションおよびロックテーブルの使用例

例 14.11 ブロックしているトランザクションの識別

別のトランザクションをどのトランザクションがブロックしているかを識別できると役立つ場合があります。INFORMATION_SCHEMA テーブルを使用すると、別のトランザクションをどのトランザクションが待機しているかや、どのリソースが要求されているかを見つけることができます。

3 人のユーザーが同時に実行している次のシナリオがあるとします。各ユーザー (またはセッション) は MySQL スレッドに対応し、あるトランザクションを別のトランザクションのあとに実行します。これらのユーザーが次のコマンドを発行したが、まだだれも自分のトランザクションをコミットしていないときのシステムの状態を考えてみてください。

- ユーザー A:

```
BEGIN;
SELECT a FROM t FOR UPDATE;
SELECT SLEEP(100);
```

- ユーザー B:

```
SELECT b FROM t FOR UPDATE;
```

- ユーザー C:

```
SELECT c FROM t FOR UPDATE;
```

このシナリオでは、次のクエリーを使用して、だれがだれを待機しているかを確認できます。

```
SELECT r.trx_id waiting_trx_id,
       r.trx_mysql_thread_id waiting_thread,
       r.trx_query waiting_query,
       b.trx_id blocking_trx_id,
       b.trx_mysql_thread_id blocking_thread,
       b.trx_query blocking_query
  FROM       information_schema.innodb_lock_waits w
  INNER JOIN information_schema.innodb_trx b  ON
    b.trx_id = w.blocking_trx_id
 INNER JOIN information_schema.innodb_trx r  ON
   r.trx_id = w.requesting_trx_id;
```

waiting trx id	waiting thread	waiting query	blocking trx id	blocking thread	blocking query
A4	6	SELECT b FROM t FOR UPDATE	A3	5	SELECT SLEEP(100)
A5	7	SELECT c FROM t FOR UPDATE	A3	5	SELECT SLEEP(100)
A5	7	SELECT c FROM t FOR UPDATE	A4	6	SELECT b FROM t FOR UPDATE

上の結果では、「waiting query」または「blocking query」でユーザーを識別できます。次のことがわかります。

- ユーザー B (trx id 'A4'、スレッド 6) とユーザー C (trx id 'A5'、スレッド 7) はどちらも、ユーザー A (trx id 'A3'、スレッド 5) を待機しています。
- ユーザー C は、ユーザー A のほかにユーザー B を待機しています。

テーブル INNODB_TRX、INNODB_LOCKS、および INNODB_LOCK_WAITS 内のベースとなるデータを確認できます。

次の表は、INFORMATION_SCHEMA.INNODB_TRX の内容のいくつかのサンプルを示しています。

trx id	trx state	trx started	trx requested lock id	trx wait started	trx weight	trx mysql thread id	trx query
A3	RUNNING	2008-01-15 16:44:54	NULL	NULL	2	5	SELECT SLEEP(100)
A4	LOCK WAIT	2008-01-15 16:45:09	A4:1:3:2	2008-01-15 16:45:09	2	6	SELECT b FROM t FOR UPDATE
A5	LOCK WAIT	2008-01-15 16:45:14	A5:1:3:2	2008-01-15 16:45:14	2	7	SELECT c FROM t FOR UPDATE

次の表は、INFORMATION_SCHEMA.INNODB_LOCKS の内容のいくつかのサンプルを示しています。

lock id	lock trx id	lock mode	lock type	lock table	lock index	lock space	lock page	lock rec	lock data
A3:1:3:2	A3	X	RECORD	`test`.`t`	`PRIMARY`	1	3	2	0x0200
A4:1:3:2	A4	X	RECORD	`test`.`t`	`PRIMARY`	1	3	2	0x0200
A5:1:3:2	A5	X	RECORD	`test`.`t`	`PRIMARY`	1	3	2	0x0200

次の表は、INFORMATION_SCHEMA.INNODB_LOCK_WAITS の内容のいくつかのサンプルを示しています。

requesting trx id	requested lock id	blocking trx id	blocking lock id
A4	A4:1:3:2	A3	A3:1:3:2
A5	A5:1:3:2	A3	A3:1:3:2
A5	A5:1:3:2	A4	A4:1:3:2

例 14.12 情報スキーマテーブル内のトランザクションデータのより複雑な例

場合によっては、内部の InnoDB ロック情報を MySQL で保持されているセッションレベルの情報と関連付けたいことがあります。たとえば、特定の InnoDB トランザクション ID について、ロックを保持しているために別のトランザクションをブロックしている可能性があるユーザーの対応する MySQL セッション ID と名前を知りたいことがあります。

INFORMATION_SCHEMA テーブルからの次の出力は、ある程度負荷の高いシステムから取得されました。

次の表からわかるように、実行中のトランザクションが複数存在します。

次の INNODB_LOCKS および INNODB_LOCK_WAITS テーブルは、次のことを示しています。

- トランザクション 77F (INSERT を実行中) は、トランザクション 77E、77D、および 77B がコミットするのを待機しています。
- トランザクション 77E (INSERT を実行中) は、トランザクション 77D および 77B がコミットするのを待機しています。
- トランザクション 77D (INSERT を実行中) は、トランザクション 77B がコミットするのを待機しています。
- トランザクション 77B (INSERT を実行中) は、トランザクション 77A がコミットするのを待機しています。
- トランザクション 77A は実行中であり、現在 SELECT を実行しています。
- トランザクション E56 (INSERT を実行中) は、トランザクション E55 がコミットするのを待機しています。
- トランザクション E55 (INSERT を実行中) は、トランザクション 19C がコミットするのを待機しています。
- トランザクション 19C は実行中であり、現在 INSERT を実行しています。

2 つのテーブル INNODB_TRX.TRX_QUERY と PROCESSLIST.INFO に示されているクエリー間に不整合が存在する可能性があることに注意してください。いずれかの特定のスレッドについて、そのスレッドの現在のトランザクション ID や、そのトランザクションで実行されているクエリーがこれらの 2 つのテーブルで異なる可能性があります。説明については、PROCESSLIST データとの不整合の可能性を参照してください。

次の表は、重いワークロードを実行しているシステム内の INFORMATION_SCHEMA.PROCESSLIST の内容を示しています。

ID	USER	HOST	DB	COMMAND	TIME	STATE	INFO
384	root	localhost	test	Query	10	update	insert into t2 values …
257	root	localhost	test	Query	3	update	insert into t2 values …
130	root	localhost	test	Query	0	update	insert into t2 values …
61	root	localhost	test	Query	1	update	insert into t2 values …
8	root	localhost	test	Query	1	update	insert into t2 values …

ID	USER	HOST	DB	COMMAND	TIME	STATE	INFO
4	root	localhost	test	Query	0	preparing	SELECT * FROM processlist
2	root	localhost	test	Sleep	566		NULL

次の表は、重いワークロードを実行しているシステム内の INFORMATION_SCHEMA.INNODB_TRX の内容を示しています。

trx id	trx state	trx started	trx requested lock id	trx wait started	trx weight	trx mysql thread id	trx query
77F	LOCK WAIT	2008-01-15 13:10:16	77F:806	2008-01-15 13:10:16	1	876	insert into t09 (D, B, C) values …
77E	LOCK WAIT	2008-01-15 13:10:16	77E:806	2008-01-15 13:10:16	1	875	insert into t09 (D, B, C) values …
77D	LOCK WAIT	2008-01-15 13:10:16	77D:806	2008-01-15 13:10:16	1	874	insert into t09 (D, B, C) values …
77B	LOCK WAIT	2008-01-15 13:10:16	77B:733:12:1	2008-01-15 13:10:16	4	873	insert into t09 (D, B, C) values …
77A	RUN-NING	2008-01-15 13:10:16	NULL	NULL	4	872	select b, c from t09 where …
E56	LOCK WAIT	2008-01-15 13:10:06	E56:743:6:2	2008-01-15 13:10:06	5	384	insert into t2 values …
E55	LOCK WAIT	2008-01-15 13:10:06	E55:743:38:2	2008-01-15 13:10:13	965	257	insert into t2 values …
19C	RUN-NING	2008-01-15 13:09:10	NULL	NULL	2900	130	insert into t2 values …
E15	RUN-NING	2008-01-15 13:08:59	NULL	NULL	5395	61	insert into t2 values …
51D	RUN-NING	2008-01-15 13:08:47	NULL	NULL	9807	8	insert into t2 values …

次の表は、重いワークロードを実行しているシステム内の INFORMATION_SCHEMA.INNODB_LOCK_WAITS の内容を示しています。

requesting trx id	requested lock id	blocking trx id	blocking lock id
77F	77F:806	77E	77E:806
77F	77F:806	77D	77D:806
77F	77F:806	77B	77B:806
77E	77E:806	77D	77D:806
77E	77E:806	77B	77B:806
77D	77D:806	77B	77B:806
77B	77B:733:12:1	77A	77A:733:12:1
E56	E56:743:6:2	E55	E55:743:6:2
E55	E55:743:38:2	19C	19C:743:38:2

次の表は、重いワークロードを実行しているシステム内の INFORMATION_SCHEMA.INNODB_LOCKS の内容を示しています。

lock id	lock trx id	lock mode	lock type	lock table	lock index	lock space	lock page	lock rec	lock data
77F:806	77F	AUTO_INC	TABLE	`test`.`t09`	NULL	NULL	NULL	NULL	NULL
77E:806	77E	AUTO_INC	TABLE	`test`.`t09`	NULL	NULL	NULL	NULL	NULL

lock id	lock trx id	lock mode	lock type	lock table	lock index	lock space	lock page	lock rec	lock data
77D:806	77D	AUTO_INC	TABLE	`test`.`t09`	NULL	NULL	NULL	NULL	NULL
77B:806	77B	AUTO_INC	TABLE	`test`.`t09`	NULL	NULL	NULL	NULL	NULL
77B:733:12:1	77B	X	RECORD	`test`.`t09`	`PRIMARY`	733	12	1	supremum pseudo-record
77A:733:12:1	77A	X	RECORD	`test`.`t09`	`PRIMARY`	733	12	1	supremum pseudo-record
E56:743:6:2	E56	S	RECORD	`test`.`t2`	`PRIMARY`	743	6	2	0, 0
E55:743:6:2	E55	X	RECORD	`test`.`t2`	`PRIMARY`	743	6	2	0, 0
E55:743:38:2	E55	S	RECORD	`test`.`t2`	`PRIMARY`	743	38	2	1922, 1922
19C:743:38:2	19C	X	RECORD	`test`.`t2`	`PRIMARY`	743	38	2	1922, 1922

14.14.2.2 INNODB_LOCKS と INNODB_LOCK_WAITS のデータ

トランザクションがテーブル内の行を更新するか、または SELECT FOR UPDATE でロックする場合、InnoDB はその行に関するロックのリストまたはキューを確立します。同様に、テーブルレベルのロックの場合、InnoDB はテーブルに関するロックのリストを保持します。2 番目のトランザクションが、互換性がないモードにある以前のトランザクションによってすでにロックされている行の更新またはテーブルのロックを行おうとした場合、InnoDB はその行に対するロック要求を対応するキューに追加します。トランザクションによって取得されるロックの場合は、その行またはテーブルのロックキューに以前に入力された互換性のないロック要求をすべて削除する (これらのロックを保持または要求しているトランザクションがコミットまたはロールバックする) 必要があります。

トランザクションは、異なる行またはテーブルに対する任意の数のロック要求を保持できます。トランザクションはいつでも、別のトランザクションによって保持されているロックを要求できますが、そのロックは、その別のトランザクションによってブロックされます。要求しているトランザクションは、ブロックしているロックを保持するトランザクションがコミットまたはロールバックするのを待機する必要があります。ロックを待機していないトランザクションは、'RUNNING' 状態にあります。ロックを待機しているトランザクションは、'LOCK WAIT' 状態にあります。

INNODB_LOCKS テーブルは、'LOCK WAIT' トランザクションごとに、その続行を妨げているすべてのロック要求を示す 1 つ以上の行を保持しています。このテーブルにはまた、特定の行またはテーブルに対して保留されているロックのキュー内の各ロックを記述した 1 行も含まれています。INNODB_LOCK_WAITS テーブルは、ほかのトランザクションによって要求されたロックを、あるトランザクションによってすでに保持されているどのロックがブロックしているかを示します。

14.14.2.3 InnoDB トランザクションおよびロックテーブルのデータ永続性および一貫性

トランザクションおよびロックテーブル (INNODB_TRX、INNODB_LOCKS、および INNODB_LOCK_WAITS) によって公開されるデータは、すばやく変更されるデータへの参照を表します。これは、アプリケーションで開始された更新が発生した場合にのみデータが変更されるほかの (ユーザー) テーブルとは異なります。ベースとなるデータはシステムで管理される内部データであり、非常にすばやく変更されることがあります。

パフォーマンス上の理由から、また InnoDB トランザクションとロック中の INFORMATION_SCHEMA テーブルの間の JOIN が誤って解釈される可能性を最小限に抑えるために、いずれかのテーブルに対して SELECT が発行されると常に、InnoDB は必要なトランザクションおよびロック情報を中間バッファーに収集します。このバッファーは、最後に読み取られてから 0.1 秒を超える時間が経過した場合にのみリフレッシュされます。3 つのテーブルを満たすために必要なデータは原子的に、かつ整合性を保ってフェッチされ、このグローバルな内部バッファー内に保存されて、ポイントインタイム「スナップショット」を形成します。複数のテーブルアクセスが 0.1 秒以内に発生した場合は (MySQL がこれらのテーブル間の結合を処理する場合は、ほぼ間違いなく発生します)、クエリーを満たすために同じスナップショットが使用されます。

これらのいずれかのテーブルを 1 つのクエリーにまとめる JOIN を実行した場合は、3 つのテーブルのデータが同じスナップショットから取得されるため、正しい結果が返されます。バッファーはこれらのテーブルのクエリーごとにはリフレッシュされないため、これらのテーブルに対して 10 分の 1 秒以内に個別のクエリーを発行した場合は、クエリーごとの結果が同じになります。これに対して、10 分の 1 秒を超える時間離れて発行された同じ

テーブルまたは異なるテーブルの 2 つの個別のクエリーでは、データが異なるスナップショットから取得されるため、異なる結果が表示される可能性があります。

トランザクションおよびロックデータが収集されている間 InnoDB は一時的に停止する必要があるため、これらのテーブルのクエリーを頻繁に実行しすぎると、ほかのユーザーから見たパフォーマンスに悪影響を与える場合があります。

これらのテーブルには機密情報 (少なくとも INNODB_LOCKS.LOCK_DATA および INNODB_TRX.TRX_QUERY) が含まれているため、セキュリティー上の理由から、そこからの SELECT を許可されるのは PROCESS 権限を持つユーザーだけです。

PROCESSLIST データとの不整合の可能性

セクション14.14.2.3「InnoDB トランザクションおよびロックテーブルのデータ永続性および一貫性」で説明されているように、InnoDB トランザクションおよびロックテーブル (INNODB_TRX、INNODB_LOCKS、および INNODB_LOCK_WAITS) を満たすデータは自動的にフェッチされ、「ポイントインタイム」スナップショットを提供する中間バッファーに保存されます。同じスナップショットからクエリーされた場合、データは 3 つのすべてのテーブルにわたって整合性があります。ただし、ベースとなるデータが非常にすばやく変更されるため、同様にすばやく変更されるほかのデータへの同様の参照が同期していない可能性があります。そのため、InnoDB トランザクションおよびロックテーブル内のデータを PROCESSLIST テーブル内のデータと比較する場合は注意してください。PROCESSLIST テーブルからのデータは、ロックおよびトランザクションに関するデータと同じスナップショットからは取得されません。1 つの SELECT (たとえば、INNODB_TRX と PROCESSLIST の結合) を発行した場合でも、一般に、これらのテーブルの内容には整合性がありません。INNODB_TRX が PROCESSLIST 内には存在しない行を参照したり、INNODB_TRX.TRX_QUERY に示されている、トランザクションの現在実行中の SQL クエリーが PROCESSLIST.INFO 内のものとは異なっていたりする可能性があります。

14.14.3 InnoDB INFORMATION_SCHEMA システムテーブル

MySQL 5.6 の時点では、InnoDB INFORMATION_SCHEMA システムテーブルを使用して、InnoDB によって管理されているスキーマオブジェクトに関するメタデータを抽出できます。この情報は、通常の InnoDB テーブルとは異なり直接クエリーできない InnoDB 内部システムテーブル (InnoDB データディクショナリとも呼ばれます) から取得されます。従来より、このタイプの情報は、セクション14.15「InnoDB モニター」の手法を使用して、InnoDB モニターを設定し、SHOW ENGINE INNODB STATUS コマンドからの出力を解析することによって取得します。InnoDB INFORMATION_SCHEMA テーブルのインタフェースを使用すると、SQL を使用してこのデータをクエリーできます。

対応する内部システムテーブルが存在しない INNODB_SYS_TABLESTATS を除き、InnoDB INFORMATION_SCHEMA システムテーブルは、メモリー内にキャッシュされているメタデータからではなく、内部の InnoDB システムテーブルから直接読み取られたデータで移入されます。

InnoDB INFORMATION_SCHEMA システムテーブルには、下に一覧表示されているテーブルが含まれます。INNODB_SYS_DATAFILES と INNODB_SYS_TABLESPACES は、InnoDB file-per-table テーブルスペース (.ibd ファイル) を MySQL データディレクトリ以外の場所に作成できるようにする、CREATE TABLE ステートメントの DATA DIRECTORY='directory' 句に対するサポートの導入とともに MySQL 5.6.6 で追加されました。

```
mysql> SHOW TABLES FROM INFORMATION_SCHEMA LIKE 'INNODB_SYS%';
+---------------------------------------------+
| Tables_in_information_schema (INNODB_SYS%) |
+---------------------------------------------+
| INNODB_SYS_DATAFILES                        |
| INNODB_SYS_TABLESTATS                       |
| INNODB_SYS_FOREIGN                          |
| INNODB_SYS_COLUMNS                          |
| INNODB_SYS_INDEXES                          |
| INNODB_SYS_FIELDS                           |
| INNODB_SYS_TABLESPACES                      |
| INNODB_SYS_FOREIGN_COLS                     |
| INNODB_SYS_TABLES                           |
+---------------------------------------------+
9 rows in set (0.00 sec)
```

これらのテーブル名は、提供されるデータのタイプを示しています。

- INNODB_SYS_TABLES は、InnoDB データディクショナリの SYS_TABLES テーブル内の情報と同等の、InnoDB テーブルに関するメタデータを提供します。
- INNODB_SYS_COLUMNS は、InnoDB データディクショナリの SYS_COLUMNS テーブル内の情報と同等の、InnoDB テーブルカラムに関するメタデータを提供します。

- INNODB_SYS_INDEXES は、InnoDB データディクショナリの SYS_INDEXES テーブル内の情報と同等の、InnoDB インデックスに関するメタデータを提供します。
- INNODB_SYS_FIELDS は、InnoDB データディクショナリの SYS_FIELDS テーブル内の情報と同等の、InnoDB インデックスのキーカラム (フィールド) に関するメタデータを提供します。
- INNODB_SYS_TABLESTATS は、インメモリーデータ構造から取得された InnoDB テーブルに関する低レベルのステータス情報のビューを提供します。対応する内部 InnoDB システムテーブルはありません。
- INNODB_SYS_DATAFILES は、InnoDB データディクショナリの SYS_DATAFILES テーブル内の情報と同等の、InnoDB テーブルスペースのデータファイルパス情報を提供します。
- INNODB_SYS_TABLESPACES は、InnoDB データディクショナリの SYS_TABLESPACES テーブル内の情報と同等の、InnoDB テーブルスペースに関するメタデータを提供します。
- INNODB_SYS_FOREIGN は、InnoDB データディクショナリの SYS_FOREIGN テーブル内の情報と同等の、InnoDB テーブルで定義された外部キーに関するメタデータを提供します。
- INNODB_SYS_FOREIGN_COLS は、InnoDB データディクショナリの SYS_FOREIGN_COLS テーブル内の情報と同等の、InnoDB テーブルで定義された外部キーのカラムに関するメタデータを提供します。

InnoDB INFORMATION_SCHEMA システムテーブルを TABLE_ID、INDEX_ID、SPACE などのフィールドを通して結合することにより、調査またはモニターしたいオブジェクトの使用可能なすべてのデータを容易に取得できます。

各テーブルのカラムについては、InnoDB INFORMATION_SCHEMA のドキュメントを参照してください。

例 14.13 InnoDB INFORMATION_SCHEMA システムテーブル

この例では、単純なテーブル (t1) を 1 つのインデックス (i1) で使用して、InnoDB INFORMATION_SCHEMA システムテーブル内に見つかったメタデータのタイプを示します。

1. テストデータベースとテーブル t1 を作成します。

```
mysql> CREATE DATABASE test;

mysql> USE test;

mysql> CREATE TABLE t1 (
col1 INT,
col2 CHAR(10),
col3 VARCHAR(10))
ENGINE = InnoDB;

mysql> CREATE INDEX i1 ON t1(col1);
```

2. テーブル t1 を作成したあと、INNODB_SYS_TABLES をクエリーして test/t1 のメタデータを見つけます。

```
mysql> SELECT * FROM INFORMATION_SCHEMA.INNODB_SYS_TABLES WHERE NAME='test/t1' \G

*************************** 1. row ***************************
     TABLE_ID: 71
         NAME: test/t1
         FLAG: 1
       N_COLS: 6
        SPACE: 57
  FILE_FORMAT: Antelope
   ROW_FORMAT: Compact
ZIP_PAGE_SIZE: 0
10 rows in set (0.00 sec)
```

テーブル t1 の TABLE_ID は 71 です。FLAG フィールドは、テーブルの形式とストレージの特性に関するビットレベルの情報を提供します。6 つのカラムがあり、そのうちの 3 つが InnoDB によって作成された非表示のカラム (DB_ROW_ID、DB_TRX_ID、および DB_ROLL_PTR) です。このテーブルの SPACE の ID は 57 です (0 の値は、テーブルがシステムテーブルスペース内に存在することを示します)。FILE_FORMAT は Antelope であり、ROW_FORMAT は Compact です。ZIP_PAGE_SIZE は、Compressed 行フォーマットのテーブルにのみ適用されます。

3. INNODB_SYS_TABLES からの TABLE_ID 情報を使用して、このテーブルのカラムに関する情報を取得するために INNODB_SYS_COLUMNS テーブルをクエリーします。

```
mysql> SELECT * FROM INFORMATION_SCHEMA.INNODB_SYS_COLUMNS where TABLE_ID = 71 \G
*************************** 1. row ***************************
TABLE_ID: 71
    NAME: col1
```

```
    POS: 0
  MTYPE: 6
 PRTYPE: 1027
    LEN: 4
*************************** 2. row ***************************
TABLE_ID: 71
   NAME: col2
    POS: 1
  MTYPE: 2
 PRTYPE: 524542
    LEN: 10
*************************** 3. row ***************************
TABLE_ID: 71
   NAME: col3
    POS: 2
  MTYPE: 1
 PRTYPE: 524303
    LEN: 10
3 rows in set (0.00 sec)
```

INNODB_SYS_COLUMNS は、TABLE_ID とカラム NAME に加えて、各カラムの序数位置 (POS) (0 から始まり順次に増分します)、カラム MTYPE または「メインの型」 (6 = INT、2 = CHAR、1 = VARCHAR)、PRTYPE または「正確な型」 (MySQL のデータ型、文字セットコード、および NULL 可能性を表すビットを含むバイナリ値)、およびカラムの長さ (LEN) を提供します。

4. ふたたび INNODB_SYS_TABLES からの TABLE_ID 情報を使用して、テーブル t1 に関連付けられたインデックスに関する情報を取得するために INNODB_SYS_INDEXES をクエリーします。

```
mysql> SELECT * FROM INFORMATION_SCHEMA.INNODB_SYS_INDEXES WHERE TABLE_ID = 71 \G
*************************** 1. row ***************************
INDEX_ID: 111
   NAME: GEN_CLUST_INDEX
TABLE_ID: 71
   TYPE: 1
N_FIELDS: 0
 PAGE_NO: 3
  SPACE: 57
*************************** 2. row ***************************
INDEX_ID: 112
   NAME: i1
TABLE_ID: 71
   TYPE: 0
N_FIELDS: 1
 PAGE_NO: 4
  SPACE: 57
2 rows in set (0.00 sec)
```

INNODB_SYS_INDEXES は、2 つのインデックスのデータを返します。最初のインデックスは GEN_CLUST_INDEX です。これは、テーブルにユーザー定義のクラスタ化されたインデックスが存在しない場合に InnoDB によって作成されたクラスタ化されたインデックスです。2 番目のインデックス (i1) は、ユーザー定義のセカンダリインデックスです。

INDEX_ID は、インスタンス内のすべてのデータベースにわたって一意であるインデックスの識別子です。TABLE_ID は、そのインデックスが関連付けられているテーブルを識別します。インデックスの TYPE 値は、インデックスのタイプ (1 = クラスタ化されたインデックス、0 = セカンダリインデックス) を示します。N_FILEDS 値は、このインデックスを構成するフィールドの数です。PAGE_NO はインデックスの B ツリーのルートページ番号であり、SPACE はインデックスが存在するテーブルスペースの ID です。0 以外の値は、そのインデックスがシステムテーブルスペース内に存在しないことを示します。

5. INNODB_SYS_INDEXES からの INDEX_ID 情報を使用して、インデックス i1 のフィールドに関する情報を取得するために INNODB_SYS_FIELDS をクエリーします。

```
mysql> SELECT * FROM INFORMATION_SCHEMA.INNODB_SYS_FIELDS where INDEX_ID = 112 \G
*************************** 1. row ***************************
INDEX_ID: 112
   NAME: col1
    POS: 0
1 row in set (0.00 sec)
```

INNODB_SYS_FIELDS は、インデックス付きフィールドの NAME と、インデックス内のその序数位置を提供します。インデックス (i1) が複数のフィールドで定義されている場合、INNODB_SYS_FIELDS は、各インデックス付きフィールドのメタデータを提供します。

6. INNODB_SYS_TABLES からの SPACE 情報を使用して、このテーブルのテーブルスペースに関する情報を取得するために INNODB_SYS_TABLESPACES テーブルをクエリーします。

```
mysql> SELECT * FROM INFORMATION_SCHEMA.INNODB_SYS_TABLESPACES WHERE SPACE = 57 \G
*************************** 1. row ***************************
       SPACE: 57
        NAME: test/t1
        FLAG: 0
 FILE_FORMAT: Antelope
  ROW_FORMAT: Compact or Redundant
   PAGE_SIZE: 16384
ZIP_PAGE_SIZE: 0
1 row in set (0.00 sec)
```

INNODB_SYS_TABLESPACES は、テーブルスペースの SPACE ID および関連付けられたテーブルの NAME に加えて、テーブルスペースの形式とストレージの特性に関するビットレベルの情報であるテーブルスペースの FLAG データを提供します。また、テーブルスペースの FILE_FORMAT、ROW_FORMAT、PAGE_SIZE、および ZIP_PAGE_SIZE データも提供されます (ZIP_PAGE_SIZE は、Compressed 行フォーマットのテーブルスペースに適用されます)。

7. ふたたび INNODB_SYS_TABLES からの SPACE 情報を使用して、このテーブルスペースのデータファイルの場所を取得するために INNODB_SYS_DATAFILES をクエリーします。

```
mysql> SELECT * FROM INFORMATION_SCHEMA.INNODB_SYS_DATAFILES WHERE SPACE = 57 \G
*************************** 1. row ***************************
SPACE: 57
 PATH: ./test/t1.ibd
1 row in set (0.01 sec)
```

データファイルは、MySQL の data ディレクトリの下の test ディレクトリにあります。file-per-table テーブルスペースが CREATE TABLE ステートメントの DATA DIRECTORY 句を使用して MySQL データディレクトリ以外の場所に作成された場合、テーブルスペースの PATH は完全修飾のディレクトリパスになります。

8. 最後の手順として、テーブル t1 (TABLE_ID = 71) に行を挿入し、INNODB_SYS_TABLESTATS テーブル内のデータを表示します。このテーブル内のデータは、InnoDB テーブルのクエリー時に使用するインデックスを決定するために MySQL オプティマイザによって使用されます。この情報は、インメモリーデータ構造から取得されます。対応する内部 InnoDB システムテーブルはありません。

```
mysql> INSERT INTO t1 VALUES(5, 'abc', 'def');
Query OK, 1 row affected (0.06 sec)

mysql> SELECT * FROM INFORMATION_SCHEMA.INNODB_SYS_TABLESTATS where TABLE_ID = 71 \G
*************************** 1. row ***************************
        TABLE_ID: 71
            NAME: test/t1
STATS_INITIALIZED: Initialized
        NUM_ROWS: 1
 CLUST_INDEX_SIZE: 1
 OTHER_INDEX_SIZE: 0
 MODIFIED_COUNTER: 1
         AUTOINC: 0
       REF_COUNT: 1
1 row in set (0.00 sec)
```

STATS_INITIALIZED フィールドは、このテーブルの統計が収集されているかどうかを示します。NUM_ROWS は、現在の推定されるテーブル内の行数です。CLUST_INDEX_SIZE および OTHER_INDEX_SIZE フィールドはそれぞれ、テーブルのクラスタ化されたインデックスとセカンダリインデックスを格納するディスク上のページの数をレポートします。MODIFIED_COUNTER 値は、外部キーからの DML 操作およびカスケード操作によって変更された行数を示します。AUTOINC 値は、自動インクリメントベースの操作に対して発行される次の番号です。テーブル t1 では自動インクリメントカラムが定義されていないため、この値は 0 です。REF_COUNT 値はカウンタです。このカウンタが 0 に達すると、テーブルキャッシュからテーブルメタデータを削除できることを示します。

例 14.14 外部キーの INFORMATION_SCHEMA システムテーブル

INNODB_SYS_FOREIGN および INNODB_SYS_FOREIGN_COLS テーブルは、外部キー関係に関するデータを提供します。この例では、外部キー関係を持つ親テーブルと子テーブルを使用して、INNODB_SYS_FOREIGN および INNODB_SYS_FOREIGN_COLS テーブル内に見つかったデータを示します。

1. テストデータベースおよび親テーブルと子テーブルを作成します。

```
mysql> CREATE DATABASE test;

mysql> USE test;

mysql> CREATE TABLE parent (id INT NOT NULL,
```

```
  -> PRIMARY KEY (id)) ENGINE=INNODB;

mysql> CREATE TABLE child (id INT, parent_id INT,
  -> INDEX par_ind (parent_id),
  -> CONSTRAINT fk1
  -> FOREIGN KEY (parent_id) REFERENCES parent(id)
  -> ON DELETE CASCADE) ENGINE=INNODB;
```

2. 親テーブルと子テーブルが作成されたら、INNODB_SYS_FOREIGN をクエリーして、test/child と test/parent の外部キー関係の外部キーデータを見つけます。

```
mysql> SELECT * FROM INFORMATION_SCHEMA.INNODB_SYS_FOREIGN \G
*************************** 1. row ***************************
      ID: test/fk1
FOR_NAME: test/child
REF_NAME: test/parent
  N_COLS: 1
    TYPE: 1
1 row in set (0.00 sec)
```

   メタデータには、子テーブルで定義された CONSTRAINT として指定されている外部キー ID (fk1) が含まれています。FOR_NAME は、外部キーが定義されている子テーブルの名前です。REF_NAME は、親テーブル (「参照される」テーブル) の名前です。N_COLS は、外部キーのインデックス内のカラム数です。TYPE は、外部キーカラムに関する追加情報を提供するビットフラグを表す数値です。この場合、TYPE 値は 1 です。これは、外部キーに対して ON DELETE CASCADE オプションが指定されたことを示します。TYPE 値の詳細は、INNODB_SYS_FOREIGN テーブルの定義を参照してください。

3. 外部キー ID を使用して、この外部キーのカラムに関するデータを表示するために INNODB_SYS_FOREIGN_COLS をクエリーします。

```
mysql> SELECT * FROM INFORMATION_SCHEMA.INNODB_SYS_FOREIGN_COLS WHERE ID = 'test/fk1' \G
*************************** 1. row ***************************
         ID: test/fk1
FOR_COL_NAME: parent_id
REF_COL_NAME: id
        POS: 0
1 row in set (0.00 sec)
```

   FOR_COL_NAME は子テーブル内の外部キーカラムの名前であり、REF_COL_NAME は親テーブル内の参照されるカラムの名前です。POS 値は、外部キーのインデックス内のキーフィールドの序数位置です (0 から始まります)。

例 14.15 InnoDB INFORMATION_SCHEMA システムテーブルの結合

この例では、employees サンプルデータベース内のテーブルに関するファイル形式、行フォーマット、ページサイズ、およびインデックスサイズ情報を収集するために 3 つの InnoDB INFORMATION_SCHEMA システムテーブル (INNODB_SYS_TABLES、INNODB_SYS_TABLESPACES、および INNODB_SYS_TABLESTATS) を結合する方法を示します。

クエリー文字列を短くするために、次のテーブル名のエイリアスが使用されます。

- INFORMATION_SCHEMA.INNODB_SYS_TABLES: a
- INFORMATION_SCHEMA.INNODB_SYS_TABLESPACES: b
- INFORMATION_SCHEMA.INNODB_SYS_TABLESTATS: c

圧縮テーブルに対応するために、IF() 制御フロー関数が使用されています。テーブルが圧縮されている場合、インデックスサイズは PAGE_SIZE ではなく、ZIP_PAGE_SIZE を使用して計算されます。バイト単位でレポートされる CLUST_INDEX_SIZE および OTHER_INDEX_SIZE を 1024*1024 で割ると、M バイト (MB) 単位のインデックスサイズが得られます。MB 値は、ROUND() 関数を使用して小数点以下 0 桁に丸められます。

```
mysql> SELECT a.NAME, a.FILE_FORMAT, a.ROW_FORMAT,
 @page_size :=
  IF(a.ROW_FORMAT='Compressed',
   b.ZIP_PAGE_SIZE, b.PAGE_SIZE)
   AS page_size,
  ROUND((@page_size * c.CLUST_INDEX_SIZE)
   /(1024*1024)) AS pk_mb,
  ROUND((@page_size * c.OTHER_INDEX_SIZE)
   /(1024*1024)) AS secidx_mb
FROM INFORMATION_SCHEMA.INNODB_SYS_TABLES a
INNER JOIN INFORMATION_SCHEMA.INNODB_SYS_TABLESPACES b on a.NAME = b.NAME
INNER JOIN INFORMATION_SCHEMA.INNODB_SYS_TABLESTATS c on b.NAME = c.NAME
```

```
WHERE a.NAME LIKE 'employees/%'
ORDER BY a.NAME DESC;
+-----------------------+-------------+------------+-----------+-------+-----------+
| NAME                  | FILE_FORMAT | ROW_FORMAT | page_size | pk_mb | secidx_mb |
+-----------------------+-------------+------------+-----------+-------+-----------+
| employees/titles      | Antelope    | Compact    |     16384 |    20 |        11 |
| employees/salaries    | Antelope    | Compact    |     16384 |    91 |        33 |
| employees/employees   | Antelope    | Compact    |     16384 |    15 |         0 |
| employees/dept_manager | Antelope   | Compact    |     16384 |     0 |         0 |
| employees/dept_emp    | Antelope    | Compact    |     16384 |    12 |        10 |
| employees/departments | Antelope    | Compact    |     16384 |     0 |         0 |
+-----------------------+-------------+------------+-----------+-------+-----------+
6 rows in set (0.01 sec)
```

14.14.4 InnoDB INFORMATION_SCHEMA FULLTEXT インデックステーブル

MySQL 5.6.4 での InnoDB テーブルに対する FULLTEXT インデックスサポートの導入により、INFORMATION_SCHEMA データベースに次のテーブルが追加されました。

```
mysql> SHOW TABLES FROM INFORMATION_SCHEMA LIKE 'INNODB_FT%';
+-------------------------------------------+
| Tables_in_INFORMATION_SCHEMA (INNODB_FT%) |
+-------------------------------------------+
| INNODB_FT_CONFIG                          |
| INNODB_FT_BEING_DELETED                   |
| INNODB_FT_DELETED                         |
| INNODB_FT_DEFAULT_STOPWORD                |
| INNODB_FT_INDEX_TABLE                     |
| INNODB_FT_INDEX_CACHE                     |
+-------------------------------------------+
6 rows in set (0.00 sec)
```

テーブルの概要

- INNODB_FT_CONFIG: FULLTEXT インデックスに関するメタデータと、InnoDB テーブルに対するそれに関連する処理を表示します。
- INNODB_FT_BEING_DELETED: OPTIMIZE TABLE の保守操作中にのみ使用される INNODB_FT_DELETED テーブルのスナップショットを提供します。OPTIMIZE TABLE が実行されると、INNODB_FT_BEING_DELETED テーブルは空になり、INNODB_FT_DELETED テーブルから DOC_ID が削除されます。INNODB_FT_BEING_DELETED の内容は一般に有効期間が短いため、モニタリングやデバッグでのこのテーブルの有用性は限られます。FULLTEXT インデックスを持つテーブルでの OPTIMIZE TABLE の実行の詳細は、セクション12.9.6「MySQL の全文検索の微調整」を参照してください。
- INNODB_FT_DELETED: InnoDB テーブルの FULLTEXT インデックスから削除された行を記録します。InnoDB FULLTEXT インデックスに対する DML 操作中にコストの高いインデックス再編成が行われないようにするために、新しく削除された単語に関する情報は個別に格納され、テキスト検索を実行すると検索結果からフィルタで除外され、OPTIMIZE TABLE テーブルを実行したときにのみメインの検索インデックスから削除されます。
- INNODB_FT_DEFAULT_STOPWORD: FULLTEXT インデックスを作成するときにデフォルトで使用されるストップワードのリストを保持します。

  INNODB_FT_DEFAULT_STOPWORD テーブルについては、セクション12.9.4「全文ストップワード」を参照してください。
- INNODB_FT_INDEX_TABLE: FULLTEXT インデックスに対するテキスト検索を処理するために使用される逆インデックスに関するデータが含まれています。
- INNODB_FT_INDEX_CACHE: FULLTEXT インデックス内の新しく挿入された行に関するトークン情報が含まれています。DML 操作中の負荷の大きなインデックスの再編成を避けるために、新しくインデックスが付けられた単語に関する情報は個別に格納され、OPTIMIZE TABLE の実行時、サーバーのシャットダウン時、またはキャッシュサイズが innodb_ft_cache_size や innodb_ft_total_cache_size で定義された制限を超えたときにのみ、メインの検索インデックスと組み合わされます。

> **注記**
>
> INNODB_FT_DEFAULT_STOPWORD テーブルを除き、innodb_ft_aux_table 構成変数を FULLTEXT インデックスを含むテーブルの名前 (database_name/table_name) に設定する必要があります。そうしないと、InnoDB FULLTEXT インデックスの INFORMATION_SCHEMA テーブルが空で表示されます。

例 14.16 InnoDB FULLTEXT インデックスの INFORMATION_SCHEMA テーブル

この例では、FULLTEXT インデックスを含むテーブルを使用して、FULLTEXT インデックスの INFORMATION_SCHEMA テーブルに含まれているデータを示します。

1. FULLTEXT インデックスを含むテーブルを作成し、一部のデータを挿入します。

```
mysql> CREATE TABLE articles (
      id INT UNSIGNED AUTO_INCREMENT NOT NULL PRIMARY KEY,
      title VARCHAR(200),
      body TEXT,
      FULLTEXT (title,body)
     ) ENGINE=InnoDB;

INSERT INTO articles (title,body) VALUES
    ('MySQL Tutorial','DBMS stands for DataBase ...'),
    ('How To Use MySQL Well','After you went through a ...'),
    ('Optimizing MySQL','In this tutorial we will show ...'),
    ('1001 MySQL Tricks','1. Never run mysqld as root. 2. ...'),
    ('MySQL vs. YourSQL','In the following database comparison ...'),
    ('MySQL Security','When configured properly, MySQL ...');
```

2. innodb_ft_aux_table 変数を FULLTEXT インデックスを含むテーブルの名前に設定します。INNODB_FT_DEFAULT_STOPWORD テーブルを除き、この変数が設定されていないと InnoDB FULLTEXT INFORMATION_SCHEMA テーブルが空で表示されます。

```
SET GLOBAL innodb_ft_aux_table = 'test/articles';
```

3. INNODB_FT_INDEX_CACHE テーブルをクエリーします。これにより、FULLTEXT インデックス内の新しく挿入された行に関する情報が示されます。DML 操作中にコストの高いインデックス再編成が行われないようにするために、新しく挿入された行のデータは、OPTIMIZE TABLE が実行されるまで (あるいは、サーバーがシャットダウンされるか、またはキャッシュの制限を超えるまで) FULLTEXT インデックスキャッシュ内に残ります。

```
mysql> SELECT * FROM INFORMATION_SCHEMA.INNODB_FT_INDEX_CACHE LIMIT 5;
+-----------+--------------+-------------+-----------+--------+----------+
| WORD      | FIRST_DOC_ID | LAST_DOC_ID | DOC_COUNT | DOC_ID | POSITION |
+-----------+--------------+-------------+-----------+--------+----------+
| 1001      |            5 |           5 |         1 |      5 |        0 |
| after     |            3 |           3 |         1 |      3 |       22 |
| comparison |            6 |           6 |         1 |      6 |       44 |
| configured |            7 |           7 |         1 |      7 |       20 |
| database  |            2 |           6 |         2 |      2 |       31 |
+-----------+--------------+-------------+-----------+--------+----------+
5 rows in set (0.00 sec)
```

4. innodb_optimize_fulltext_only を有効にし、FULLTEXT インデックスを含むテーブルに対して OPTIMIZE TABLE を実行します。この操作により、FULLTEXT インデックスキャッシュの内容がメインの FULLTEXT インデックスにフラッシュされます。innodb_optimize_fulltext_only は、InnoDB テーブルでの OPTIMIZE TABLE ステートメントの動作方法を変更するものであり、FULLTEXT インデックスを含む InnoDB テーブルでの保守操作中に一時的に有効にすることを目的にしています。

```
mysql> SET GLOBAL innodb_optimize_fulltext_only=ON;
Query OK, 0 rows affected (0.00 sec)

mysql> OPTIMIZE TABLE articles;
+---------------+----------+----------+----------+
| Table         | Op       | Msg_type | Msg_text |
+---------------+----------+----------+----------+
| test.articles | optimize | status   | OK       |
+---------------+----------+----------+----------+
1 row in set (0.03 sec)
```

5. INNODB_FT_INDEX_TABLE テーブルにクエリーして、メインの FULLTEXT インデックス内のデータに関する情報 (FULLTEXT インデックスキャッシュからフラッシュされたばかりのデータに関する情報を含む) を表示します。

```
mysql> SELECT * FROM INFORMATION_SCHEMA.INNODB_FT_INDEX_TABLE LIMIT 5;
+-----------+--------------+-------------+-----------+--------+----------+
| WORD      | FIRST_DOC_ID | LAST_DOC_ID | DOC_COUNT | DOC_ID | POSITION |
+-----------+--------------+-------------+-----------+--------+----------+
| 1001      |            5 |           5 |         1 |      5 |        0 |
| after     |            3 |           3 |         1 |      3 |       22 |
| comparison |            6 |           6 |         1 |      6 |       44 |
| configured |            7 |           7 |         1 |      7 |       20 |
```

```
| database   |          2 |          6 |       2 |     2 |      31 |
+------------+------------+------------+---------+-------+---------+
5 rows in set (0.00 sec)
```

OPTIMIZE TABLE 操作によって FULLTEXT インデックスキャッシュがフラッシュされたため、INNODB_FT_INDEX_CACHE テーブルは空になっています。

```
mysql> SELECT * FROM INFORMATION_SCHEMA.INNODB_FT_INDEX_CACHE LIMIT 5;
Empty set (0.00 sec)
```

6. test/articles テーブルからいくつかのレコードを削除します。

```
mysql> DELETE FROM test.articles WHERE id < 4;
Query OK, 3 rows affected (0.11 sec)
```

7. INNODB_FT_DELETED テーブルをクエリーします。このテーブルには、FULLTEXT インデックスから削除された行が記録されます。DML 操作中にコストの高いインデックス再編成が行われないようにするために、新しく削除されたレコードに関する情報は個別に格納され、テキスト検索を実行すると検索結果からフィルタで除外され、OPTIMIZE TABLE を実行するとメインの検索インデックスから削除されます。

```
mysql> SELECT * FROM INFORMATION_SCHEMA.INNODB_FT_DELETED;
+--------+
| DOC_ID |
+--------+
|      2 |
|      3 |
|      4 |
+--------+
3 rows in set (0.00 sec)
```

8. OPTIMIZE TABLE を実行して、削除されたレコードを消去します。

```
mysql> OPTIMIZE TABLE articles;
+---------------+----------+----------+----------+
| Table         | Op       | Msg_type | Msg_text |
+---------------+----------+----------+----------+
| test.articles | optimize | status   | OK       |
+---------------+----------+----------+----------+
1 row in set (0.10 sec)
```

INNODB_FT_DELETED テーブルが空で表示されるようになります。

```
mysql> SELECT * FROM INFORMATION_SCHEMA.INNODB_FT_DELETED;
Empty set (0.00 sec)
```

9. INNODB_FT_CONFIG テーブルをクエリーします。このテーブルには、FULLTEXT インデックスに関するメタデータとそれに関連する処理が含まれています。

   - optimize_checkpoint_limit は、OPTIMIZE TABLE の実行が停止するまでの秒数です。
   - synced_doc_id は、発行される次の DOC_ID です。
   - stopword_table_name は、ユーザー定義のストップワードテーブルに対する database/table の名前です。ユーザー定義のストップワードテーブルがない場合、このフィールドは空です。
   - use_stopword は、ストップワードテーブルを使用するかどうかを示します。これは、FULLTEXT インデックスの作成時に定義されます。

```
mysql> SELECT * FROM INFORMATION_SCHEMA.INNODB_FT_CONFIG;
+---------------------------+-------+
| KEY                       | VALUE |
+---------------------------+-------+
| optimize_checkpoint_limit | 180   |
| synced_doc_id             | 8     |
| stopword_table_name       |       |
| use_stopword              | 1     |
+---------------------------+-------+
4 rows in set (0.00 sec)
```

14.14.5 InnoDB INFORMATION_SCHEMA バッファープールテーブル

InnoDB INFORMATION_SCHEMA バッファープールテーブルは、バッファープールのステータス情報、および InnoDB バッファープール内のページに関するメタデータを提供します。これらのテーブルは MySQL 5.6.2、で導入され、あとで MySQL 5.5 (MySQL 5.5.28) および MySQL 5.1 (MySQL 5.1.66) にバックポートされました。

InnoDB INFORMATION_SCHEMA バッファープールテーブルには、下に一覧表示されているものが含まれます。

```
mysql> SHOW TABLES FROM INFORMATION_SCHEMA LIKE 'INNODB_BUFFER%';
+-----------------------------------------------+
| Tables_in_INFORMATION_SCHEMA (INNODB_BUFFER%) |
+-----------------------------------------------+
| INNODB_BUFFER_PAGE_LRU                        |
| INNODB_BUFFER_PAGE                            |
| INNODB_BUFFER_POOL_STATS                      |
+-----------------------------------------------+
3 rows in set (0.00 sec)
```

テーブルの概要

- INNODB_BUFFER_PAGE: InnoDB バッファープール内の各ページに関する情報を保持します。
- INNODB_BUFFER_PAGE_LRU: InnoDB バッファープール内のページに関する情報、特に、いっぱいになったときにバッファープールからどのページを削除するかを決定する LRU リスト内の各ページの順序を保持します。INNODB_BUFFER_PAGE_LRU テーブルには、INNODB_BUFFER_PAGE テーブルと同じカラムがありますが、INNODB_BUFFER_PAGE_LRU テーブルには BLOCK_ID カラムではなく LRU_POSITION カラムがある点が異なります。
- INNODB_BUFFER_POOL_STATS: バッファープールのステータス情報を提供します。同じ情報のほとんどは、SHOW ENGINE INNODB STATUS の出力で提供されるか、または InnoDB バッファープールのサーバーステータス変数を使用して取得できます。

> **警告**
>
> INNODB_BUFFER_PAGE テーブルまたは INNODB_BUFFER_PAGE_LRU テーブルのクエリーによって、大幅なパフォーマンスオーバーヘッドが導入される場合があります。クエリーによって発生する可能性のあるパフォーマンスへの影響を認識し、かつそれが許容可能であると判断していないかぎり、これらのテーブルを本番システムではクエリーしないでください。パフォーマンスへの影響を回避するために、調査しようとしている問題をテストインスタンスで再現し、テストインスタンスでクエリーを実行してください。

例 14.17 INNODB_BUFFER_PAGE テーブル内のシステムデータのクエリー

このクエリーは、TABLE_NAME 値が NULL であるか、あるいはそのテーブル名にスラッシュ「/」またはピリオド「.」を含む (これはユーザー定義のテーブルを示します) ページを除外することによって、システムデータを含むページの概数を提供します。

```
SELECT COUNT(*) FROM INFORMATION_SCHEMA.INNODB_BUFFER_PAGE
WHERE TABLE_NAME IS NULL OR (INSTR(TABLE_NAME, '/') = 0 AND INSTR(TABLE_NAME, '.') = 0);
+----------+
| COUNT(*) |
+----------+
|     1320 |
+----------+
1 row in set (0.12 sec)
```

このクエリーは、システムデータを含むページの概数、バッファープールページの総数、およびシステムデータを含むページの概略の割合 (%) を返します。

```
SELECT
(SELECT COUNT(*) FROM INFORMATION_SCHEMA.INNODB_BUFFER_PAGE
WHERE TABLE_NAME IS NULL OR (INSTR(TABLE_NAME, '/') = 0 AND INSTR(TABLE_NAME, '.') = 0)
) AS system_pages,
(
SELECT COUNT(*)
FROM INFORMATION_SCHEMA.INNODB_BUFFER_PAGE
) AS total_pages,
(
SELECT ROUND((system_pages/total_pages) * 100)
) AS system_page_percentage;
+--------------+-------------+------------------------+
| system_pages | total_pages | system_page_percentage |
+--------------+-------------+------------------------+
|         1320 |        8192 |                     16 |
+--------------+-------------+------------------------+
1 row in set (0.15 sec)
```

バッファープール内のシステムデータのタイプは、PAGE_TYPE 値をクエリーすることによって確認できます。たとえば、次のクエリーは、システムデータを含むページ間の 8 つの個別の PAGE_TYPE 値を返します。

```
mysql> SELECT DISTINCT PAGE_TYPE FROM INFORMATION_SCHEMA.INNODB_BUFFER_PAGE
WHERE TABLE_NAME IS NULL OR (INSTR(TABLE_NAME, '/') = 0 AND INSTR(TABLE_NAME, '.') = 0);
+------------------+
| PAGE_TYPE        |
+------------------+
| SYSTEM           |
| IBUF_BITMAP      |
| UNDO_LOG         |
| UNKNOWN          |
| FILE_SPACE_HEADER |
| INODE            |
| ALLOCATED        |
| TRX_SYSTEM       |
+------------------+
8 rows in set (0.05 sec)
```

例 14.18 INNODB_BUFFER_PAGE テーブル内のユーザーデータのクエリー

このクエリーは、TABLE_NAME 値が NOT NULL および NOT LIKE '%INNODB_SYS_TABLES%' であるページをカウントすることによって、ユーザーデータを含むページの概数を提供します。

```
mysql> SELECT COUNT(*) FROM INFORMATION_SCHEMA.INNODB_BUFFER_PAGE
WHERE TABLE_NAME IS NOT NULL AND TABLE_NAME NOT LIKE '%INNODB_SYS_TABLES%';
+----------+
| COUNT(*) |
+----------+
|     6872 |
+----------+
1 row in set (0.06 sec)
```

このクエリーは、ユーザーデータを含むページの概数、バッファープールページの総数、およびユーザーデータを含むページの概略の割合 (%) を返します。

```
mysql> SELECT
(SELECT COUNT(*) FROM INFORMATION_SCHEMA.INNODB_BUFFER_PAGE
WHERE TABLE_NAME IS NOT NULL AND (INSTR(TABLE_NAME, '/') > 0 OR INSTR(TABLE_NAME, '.') > 0)
) AS user_pages,
(
SELECT COUNT(*)
FROM information_schema.INNODB_BUFFER_PAGE
) AS total_pages,
(
SELECT ROUND((user_pages/total_pages) * 100)
) AS user_page_percentage;
+------------+-------------+----------------------+
| user_pages | total_pages | user_page_percentage |
+------------+-------------+----------------------+
|       6872 |        8192 |                   84 |
+------------+-------------+----------------------+
1 row in set (0.08 sec)
```

このクエリーは、バッファープール内のページを含むユーザー定義のテーブルを識別します。

```
mysql> SELECT DISTINCT TABLE_NAME FROM INFORMATION_SCHEMA.INNODB_BUFFER_PAGE
WHERE TABLE_NAME IS NOT NULL AND (INSTR(TABLE_NAME, '/') > 0 OR INSTR(TABLE_NAME, '.') > 0)
AND TABLE_NAME NOT LIKE '`mysql`.`innodb_%';
+------------------------+
| TABLE_NAME             |
+------------------------+
| `employees`.`salaries` |
| `employees`.`employees` |
+------------------------+
2 rows in set (0.09 sec)
```

例 14.19 INNODB_BUFFER_PAGE テーブル内のインデックスデータのクエリー

インデックスページに関する情報を取得するには、そのインデックスの名前を使用して INDEX_NAME カラムをクエリーします。たとえば、次のクエリーは、employees.salaries テーブルで定義されている emp_no インデックスのページの数とページの合計データサイズを返します。

```
mysql> SELECT INDEX_NAME, COUNT(*) AS Pages,
ROUND(SUM(IF(COMPRESSED_SIZE = 0, @@GLOBAL.innodb_page_size, COMPRESSED_SIZE))/1024/1024)
AS 'Total Data (MB)'
FROM INFORMATION_SCHEMA.INNODB_BUFFER_PAGE
WHERE INDEX_NAME='emp_no' AND TABLE_NAME = '`employees`.`salaries`';
```

```
+------------+-------+-----------------+
| INDEX_NAME | Pages | Total Data (MB) |
+------------+-------+-----------------+
| emp_no     |  1756 |              27 |
+------------+-------+-----------------+
1 row in set (0.07 sec)
```

このクエリーは、employees.salaries テーブルで定義されているすべてのインデックスのページの数とページの合計データサイズを返します。

```
mysql> SELECT INDEX_NAME, COUNT(*) AS Pages,
ROUND(SUM(IF(COMPRESSED_SIZE = 0, @@GLOBAL.innodb_page_size, COMPRESSED_SIZE))/1024/1024)
AS 'Total Data (MB)'
FROM INFORMATION_SCHEMA.INNODB_BUFFER_PAGE
WHERE TABLE_NAME = '`employees`.`salaries`'
GROUP BY INDEX_NAME;
+------------+-------+-----------------+
| INDEX_NAME | Pages | Total Data (MB) |
+------------+-------+-----------------+
| emp_no     |  1756 |              27 |
| PRIMARY    |  4838 |              76 |
+------------+-------+-----------------+
2 rows in set (0.12 sec)
```

例 14.20 INNODB_BUFFER_PAGE_LRU テーブル内の LRU_POSITION データのクエリー

INNODB_BUFFER_PAGE_LRU テーブルは、InnoDB バッファープール内のページに関する情報、特に、いっぱいになったときにバッファープールからどのページを削除するかを決定する各ページの順序を保持しています。このページの定義は、このテーブルには BLOCK_ID カラムの代わりに LRU_POSITION カラムがある点を除き、INNODB_BUFFER_PAGE の場合と同じです。

このクエリーは、employees.employees テーブルの各ページによって占有されている LRU リスト内の特定の場所にある位置の数をカウントします。

```
mysql> SELECT COUNT(LRU_POSITION) FROM INFORMATION_SCHEMA.INNODB_BUFFER_PAGE_LRU
WHERE TABLE_NAME="`employees`.`employees`" AND LRU_POSITION < 3072;
+--------------------+
| COUNT(LRU_POSITION) |
+--------------------+
|                275 |
+--------------------+
1 row in set (0.04 sec)
```

例 14.21 INNODB_BUFFER_POOL_STATS テーブルのクエリー

INNODB_BUFFER_POOL_STATS テーブルは、SHOW ENGINE INNODB STATUS および InnoDB バッファープールのステータス変数と同様の情報を提供します。

```
mysql> SELECT * FROM INFORMATION_SCHEMA.INNODB_BUFFER_POOL_STATS \G
*************************** 1. row ***************************
                         POOL_ID: 0
                       POOL_SIZE: 8192
                    FREE_BUFFERS: 1024
                  DATABASE_PAGES: 7029
              OLD_DATABASE_PAGES: 2574
          MODIFIED_DATABASE_PAGES: 0
             PENDING_DECOMPRESS: 0
                PENDING_READS: 0
              PENDING_FLUSH_LRU: 0
             PENDING_FLUSH_LIST: 0
               PAGES_MADE_YOUNG: 173
           PAGES_NOT_MADE_YOUNG: 3721891
          PAGES_MADE_YOUNG_RATE: 0
      PAGES_MADE_NOT_YOUNG_RATE: 0
            NUMBER_PAGES_READ: 1075
         NUMBER_PAGES_CREATED: 12594
         NUMBER_PAGES_WRITTEN: 13525
             PAGES_READ_RATE: 0
           PAGES_CREATE_RATE: 0
          PAGES_WRITTEN_RATE: 0
            NUMBER_PAGES_GET: 27873240
                    HIT_RATE: 0
    YOUNG_MAKE_PER_THOUSAND_GETS: 0
NOT_YOUNG_MAKE_PER_THOUSAND_GETS: 0
       NUMBER_PAGES_READ_AHEAD: 576
```

```
      NUMBER_READ_AHEAD_EVICTED: 0
            READ_AHEAD_RATE: 0
       READ_AHEAD_EVICTED_RATE: 0
              LRU_IO_TOTAL: 0
            LRU_IO_CURRENT: 0
          UNCOMPRESS_TOTAL: 0
        UNCOMPRESS_CURRENT: 0
1 row in set (0.00 sec)
```

比較のために、同じデータセットに基づいた SHOW ENGINE INNODB STATUS の出力および InnoDB バッファープールのステータス変数の出力を次に示します。

SHOW ENGINE INNODB STATUS の出力の詳細は、セクション14.15.3「InnoDB 標準モニターおよびロックモニターの出力」を参照してください。

```
mysql> SHOW ENGINE INNODB STATUS \G
...
----------------------
BUFFER POOL AND MEMORY
----------------------
Total memory allocated 137363456; in additional pool allocated 0
Dictionary memory allocated 99725
Buffer pool size   8192
Free buffers       1024
Database pages     7029
Old database pages 2574
Modified db pages  0
Pending reads 0
Pending writes: LRU 0, flush list 0, single page 0
Pages made young 173, not young 3721891
0.00 youngs/s, 0.00 non-youngs/s
Pages read 1075, created 12594, written 13525
0.00 reads/s, 0.00 creates/s, 0.00 writes/s
No buffer pool page gets since the last printout
Pages read ahead 0.00/s, evicted without access 0.00/s, Random read ahead 0.00/s
LRU len: 7029, unzip_LRU len: 0
I/O sum[0]:cur[0], unzip sum[0]:cur[0]
...
```

ステータス変数の説明については、セクション5.1.6「サーバーステータス変数」を参照してください。

```
mysql> SHOW STATUS LIKE 'Innodb_buffer%';
+---------------------------------------+-------------+
| Variable_name                         | Value       |
+---------------------------------------+-------------+
| Innodb_buffer_pool_dump_status        | not started |
| Innodb_buffer_pool_load_status        | not started |
| Innodb_buffer_pool_pages_data         | 7029        |
| Innodb_buffer_pool_bytes_data         | 115163136   |
| Innodb_buffer_pool_pages_dirty        | 0           |
| Innodb_buffer_pool_bytes_dirty        | 0           |
| Innodb_buffer_pool_pages_flushed      | 13525       |
| Innodb_buffer_pool_pages_free         | 1024        |
| Innodb_buffer_pool_pages_misc         | 139         |
| Innodb_buffer_pool_pages_total        | 8192        |
| Innodb_buffer_pool_read_ahead_rnd     | 0           |
| Innodb_buffer_pool_read_ahead         | 576         |
| Innodb_buffer_pool_read_ahead_evicted | 0           |
| Innodb_buffer_pool_read_requests      | 27873240    |
| Innodb_buffer_pool_reads              | 500         |
| Innodb_buffer_pool_wait_free          | 0           |
| Innodb_buffer_pool_write_requests     | 11966441    |
+---------------------------------------+-------------+
17 rows in set (0.00 sec)
```

14.14.6 InnoDB INFORMATION_SCHEMA メトリックテーブル

MySQL 5.6.2 で導入された INNODB_METRICS テーブルは、InnoDB のパフォーマンスおよびリソース関連のすべてのカウンタを 1 つの INFORMATION_SCHEMA テーブルに統合します。

INNODB_METRICS テーブルのカラムを次の例に示します。各カラムについては、セクション21.29.19「INFORMATION_SCHEMA INNODB_METRICS テーブル」を参照してください。

```
mysql> SELECT * FROM INFORMATION_SCHEMA.INNODB_METRICS WHERE NAME="dml_inserts" \G
*************************** 1. row ***************************
           NAME: dml_inserts
```

```
     SUBSYSTEM: dml
         COUNT: 46273
     MAX_COUNT: 46273
     MIN_COUNT: NULL
     AVG_COUNT: 492.2659574468085
   COUNT_RESET: 46273
MAX_COUNT_RESET: 46273
MIN_COUNT_RESET: NULL
AVG_COUNT_RESET: NULL
  TIME_ENABLED: 2014-11-28 16:07:53
 TIME_DISABLED: NULL
  TIME_ELAPSED: 94
    TIME_RESET: NULL
        STATUS: enabled
          TYPE: status_counter
       COMMENT: Number of rows inserted
1 row in set (0.00 sec)
```

カウンタの有効化、無効化、およびリセット

次の構成オプションを使用して、カウンタを有効または無効にしたり、リセットしたりすることができます。

- innodb_monitor_enable: 1 つ以上のカウンタを有効にします。

  ```
  SET GLOBAL innodb_monitor_enable = [counter-name|module_name|pattern|all];
  ```

- innodb_monitor_disable: 1 つ以上のカウンタを無効にします。

  ```
  SET GLOBAL innodb_monitor_disable = [counter-name|module_name|pattern|all];
  ```

- innodb_monitor_reset: 1 つ以上のカウンタのカウント値を 0 にリセットします。

  ```
  SET GLOBAL innodb_monitor_reset = [counter-name|module_name|pattern|all];
  ```

- innodb_monitor_reset_all: 1 つ以上のカウンタのすべての値をリセットします。innodb_monitor_reset_all を使用する前にカウンタを無効にする必要があります。

  ```
  SET GLOBAL innodb_monitor_reset_all = [counter-name|module_name|pattern|all];
  ```

MySQL サーバーの構成ファイルを使用して、起動時にカウンタおよびカウンタモジュールを有効にすることもできます。たとえば、log モジュール、metadata_table_handles_opened および metadata_table_handles_closed カウンタを有効にするには、my.cnf 構成ファイルの [mysqld] セクション内に次の行を入力します。

```
[mysqld]
innodb_monitor_enable = module_recovery,metadata_table_handles_opened,metadata_table_handles_closed
```

構成ファイルで複数のカウンタまたはモジュールを有効にする場合は、上の例に示すように、innodb_monitor_enable 構成オプションに続けて、カウンタおよびモジュール名をカンマで区切って指定する必要があります。構成ファイルで使用できるのは、innodb_monitor_enable オプションだけです。無効化とリセットの構成オプションは、コマンド行でのみサポートされます。

> **注記**
>
> 各カウンタによってサーバーにはある程度の実行時オーバーヘッドが発生するため、通常は、実験やベンチマーク中にテストおよび開発サーバー上で多くのカウンタを有効にし、本番サーバー上でカウンタを有効にするのは、既知の問題を診断するか、または特定のサーバーやワークロードのボトルネックになる可能性のある側面をモニターする場合だけにしてください。

カウンタ

INNODB_METRICS テーブルで表されるカウンタは変更される可能性があるため、最新のリストを取得するには、実行中の MySQL サーバーをクエリーします。下のリストは、MySQL 5.6.23 の時点で使用可能なカウンタを示しています。

デフォルトで有効になっているカウンタは、SHOW ENGINE INNODB STATUS によって使用されるカウンタに対応しています。SHOW ENGINE INNODB STATUS によって使用されるカウンタは常にシステムレベルで「オン」の状態ですが、INNODB_METRICS テーブルのこれらのカウンタは必要に応じて無効にすることができます。また、カウンタのステータスも永続的ではありません。特に指定されていないかぎり、カウンタは、サーバーが再起動されるとデフォルトの有効または無効のステータスに戻ります。

INNODB_METRICS テーブルへの追加または変更によって影響を受けるプログラムを実行する場合は、アップグレードの前にリリースノートを確認し、新しいリリースの INNODB_METRICS テーブルをクエリーすることをお勧めします。

```
mysql> SELECT name, subsystem, status FROM INFORMATION_SCHEMA.INNODB_METRICS ORDER BY NAME;
+----------------------------------------+--------------------+----------+
| name                                   | subsystem          | status   |
+----------------------------------------+--------------------+----------+
| adaptive_hash_pages_added              | adaptive_hash_index | disabled |
| adaptive_hash_pages_removed            | adaptive_hash_index | disabled |
| adaptive_hash_rows_added               | adaptive_hash_index | disabled |
| adaptive_hash_rows_deleted_no_hash_entry | adaptive_hash_index | disabled |
| adaptive_hash_rows_removed             | adaptive_hash_index | disabled |
| adaptive_hash_rows_updated             | adaptive_hash_index | disabled |
| adaptive_hash_searches                 | adaptive_hash_index | enabled  |
| adaptive_hash_searches_btree           | adaptive_hash_index | disabled |
| buffer_data_reads                      | buffer             | enabled  |
| buffer_data_written                    | buffer             | enabled  |
| buffer_flush_adaptive                  | buffer             | disabled |
| buffer_flush_adaptive_pages            | buffer             | disabled |
| buffer_flush_adaptive_total_pages      | buffer             | disabled |
| buffer_flush_avg_page_rate             | buffer             | disabled |
| buffer_flush_background                | buffer             | disabled |
| buffer_flush_background_pages          | buffer             | disabled |
| buffer_flush_background_total_pages    | buffer             | disabled |
| buffer_flush_batches                   | buffer             | disabled |
| buffer_flush_batch_num_scan            | buffer             | disabled |
| buffer_flush_batch_pages               | buffer             | disabled |
| buffer_flush_batch_rescan              | buffer             | disabled |
| buffer_flush_batch_scanned             | buffer             | disabled |
| buffer_flush_batch_scanned_per_call    | buffer             | disabled |
| buffer_flush_batch_total_pages         | buffer             | disabled |
| buffer_flush_lsn_avg_rate              | buffer             | disabled |
| buffer_flush_neighbor                  | buffer             | disabled |
| buffer_flush_neighbor_pages            | buffer             | disabled |
| buffer_flush_neighbor_total_pages      | buffer             | disabled |
| buffer_flush_n_to_flush_requested      | buffer             | disabled |
| buffer_flush_pct_for_dirty             | buffer             | disabled |
| buffer_flush_pct_for_lsn               | buffer             | disabled |
| buffer_flush_sync                      | buffer             | disabled |
| buffer_flush_sync_pages                | buffer             | disabled |
| buffer_flush_sync_total_pages          | buffer             | disabled |
| buffer_flush_sync_waits                | buffer             | disabled |
| buffer_LRU_batches                     | buffer             | disabled |
| buffer_LRU_batch_num_scan              | buffer             | disabled |
| buffer_LRU_batch_pages                 | buffer             | disabled |
| buffer_LRU_batch_scanned               | buffer             | disabled |
| buffer_LRU_batch_scanned_per_call      | buffer             | disabled |
| buffer_LRU_batch_total_pages           | buffer             | disabled |
| buffer_LRU_get_free_search             | Buffer             | disabled |
| buffer_LRU_search_num_scan             | buffer             | disabled |
| buffer_LRU_search_scanned              | buffer             | disabled |
| buffer_LRU_search_scanned_per_call     | buffer             | disabled |
| buffer_LRU_single_flush_failure_count  | Buffer             | disabled |
| buffer_LRU_single_flush_num_scan       | buffer             | disabled |
| buffer_LRU_single_flush_scanned        | buffer             | disabled |
| buffer_LRU_single_flush_scanned_per_call | buffer           | disabled |
| buffer_LRU_unzip_search_num_scan       | buffer             | disabled |
| buffer_LRU_unzip_search_scanned        | buffer             | disabled |
| buffer_LRU_unzip_search_scanned_per_call | buffer           | disabled |
| buffer_pages_created                   | buffer             | enabled  |
| buffer_pages_read                      | buffer             | enabled  |
| buffer_pages_written                   | buffer             | enabled  |
| buffer_page_read_blob                  | buffer_page_io     | disabled |
| buffer_page_read_fsp_hdr               | buffer_page_io     | disabled |
| buffer_page_read_ibuf_bitmap           | buffer_page_io     | disabled |
| buffer_page_read_ibuf_free_list        | buffer_page_io     | disabled |
| buffer_page_read_index_ibuf_leaf       | buffer_page_io     | disabled |
| buffer_page_read_index_ibuf_non_leaf   | buffer_page_io     | disabled |
| buffer_page_read_index_inode           | buffer_page_io     | disabled |
| buffer_page_read_index_leaf            | buffer_page_io     | disabled |
| buffer_page_read_index_non_leaf        | buffer_page_io     | disabled |
| buffer_page_read_other                 | buffer_page_io     | disabled |
| buffer_page_read_system_page           | buffer_page_io     | disabled |
| buffer_page_read_trx_system            | buffer_page_io     | disabled |
| buffer_page_read_undo_log              | buffer_page_io     | disabled |
| buffer_page_read_xdes                  | buffer_page_io     | disabled |
| buffer_page_read_zblob                 | buffer_page_io     | disabled |
| buffer_page_read_zblob2                | buffer_page_io     | disabled |
| buffer_page_written_blob               | buffer_page_io     | disabled |
```

```
| buffer_page_written_fsp_hdr             | buffer_page_io      | disabled |
| buffer_page_written_ibuf_bitmap         | buffer_page_io      | disabled |
| buffer_page_written_ibuf_free_list      | buffer_page_io      | disabled |
| buffer_page_written_index_ibuf_leaf     | buffer_page_io      | disabled |
| buffer_page_written_index_ibuf_non_leaf | buffer_page_io      | disabled |
| buffer_page_written_index_inode         | buffer_page_io      | disabled |
| buffer_page_written_index_leaf          | buffer_page_io      | disabled |
| buffer_page_written_index_non_leaf      | buffer_page_io      | disabled |
| buffer_page_written_other               | buffer_page_io      | disabled |
| buffer_page_written_system_page         | buffer_page_io      | disabled |
| buffer_page_written_trx_system          | buffer_page_io      | disabled |
| buffer_page_written_undo_log            | buffer_page_io      | disabled |
| buffer_page_written_xdes                | buffer_page_io      | disabled |
| buffer_page_written_zblob               | buffer_page_io      | disabled |
| buffer_page_written_zblob2              | buffer_page_io      | disabled |
| buffer_pool_bytes_data                  | buffer              | enabled  |
| buffer_pool_bytes_dirty                 | buffer              | enabled  |
| buffer_pool_pages_data                  | buffer              | enabled  |
| buffer_pool_pages_dirty                 | buffer              | enabled  |
| buffer_pool_pages_free                  | buffer              | enabled  |
| buffer_pool_pages_misc                  | buffer              | enabled  |
| buffer_pool_pages_total                 | buffer              | enabled  |
| buffer_pool_reads                       | buffer              | enabled  |
| buffer_pool_read_ahead                  | buffer              | enabled  |
| buffer_pool_read_ahead_evicted          | buffer              | enabled  |
| buffer_pool_read_requests               | buffer              | enabled  |
| buffer_pool_size                        | server              | enabled  |
| buffer_pool_wait_free                   | buffer              | enabled  |
| buffer_pool_write_requests              | buffer              | enabled  |
| compression_pad_decrements              | compression         | disabled |
| compression_pad_increments              | compression         | disabled |
| compress_pages_compressed               | compression         | disabled |
| compress_pages_decompressed             | compression         | disabled |
| ddl_background_drop_indexes             | ddl                 | disabled |
| ddl_background_drop_tables              | ddl                 | disabled |
| ddl_online_create_index                 | ddl                 | disabled |
| ddl_pending_alter_table                 | ddl                 | disabled |
| dml_deletes                             | dml                 | enabled  |
| dml_inserts                             | dml                 | enabled  |
| dml_reads                               | dml                 | enabled  |
| dml_updates                             | dml                 | enabled  |
| file_num_open_files                     | file_system         | enabled  |
| ibuf_merges                             | change_buffer       | enabled  |
| ibuf_merges_delete                      | change_buffer       | enabled  |
| ibuf_merges_delete_mark                 | change_buffer       | enabled  |
| ibuf_merges_discard_delete              | change_buffer       | enabled  |
| ibuf_merges_discard_delete_mark         | change_buffer       | enabled  |
| ibuf_merges_discard_insert              | change_buffer       | enabled  |
| ibuf_merges_insert                      | change_buffer       | enabled  |
| ibuf_size                               | change_buffer       | enabled  |
| icp_attempts                            | icp                 | disabled |
| icp_match                               | icp                 | disabled |
| icp_no_match                            | icp                 | disabled |
| icp_out_of_range                        | icp                 | disabled |
| index_page_discards                     | index               | disabled |
| index_page_merge_attempts               | index               | disabled |
| index_page_merge_successful             | index               | disabled |
| index_page_reorg_attempts               | index               | disabled |
| index_page_reorg_successful             | index               | disabled |
| index_page_splits                       | index               | disabled |
| innodb_activity_count                   | server              | enabled  |
| innodb_background_drop_table_usec       | server              | disabled |
| innodb_checkpoint_usec                  | server              | disabled |
| innodb_dblwr_pages_written              | server              | enabled  |
| innodb_dblwr_writes                     | server              | enabled  |
| innodb_dict_lru_usec                    | server              | disabled |
| innodb_ibuf_merge_usec                  | server              | disabled |
| innodb_log_flush_usec                   | server              | disabled |
| innodb_master_active_loops              | server              | disabled |
| innodb_master_idle_loops                | server              | disabled |
| innodb_master_purge_usec                | server              | disabled |
| innodb_master_thread_sleeps             | server              | disabled |
| innodb_mem_validate_usec                | server              | disabled |
| innodb_page_size                        | server              | enabled  |
| innodb_rwlock_s_os_waits                | server              | enabled  |
| innodb_rwlock_s_spin_rounds             | server              | enabled  |
| innodb_rwlock_s_spin_waits              | server              | enabled  |
| innodb_rwlock_x_os_waits                | server              | enabled  |
| innodb_rwlock_x_spin_rounds             | server              | enabled  |
| innodb_rwlock_x_spin_waits              | server              | enabled  |
```

```
| lock_deadlocks                          | lock              | enabled  |
| lock_rec_locks                          | lock              | disabled |
| lock_rec_lock_created                   | lock              | disabled |
| lock_rec_lock_removed                   | lock              | disabled |
| lock_rec_lock_requests                  | lock              | disabled |
| lock_rec_lock_waits                     | lock              | disabled |
| lock_row_lock_current_waits             | lock              | enabled  |
| lock_row_lock_time                      | lock              | enabled  |
| lock_row_lock_time_avg                  | lock              | enabled  |
| lock_row_lock_time_max                  | lock              | enabled  |
| lock_row_lock_waits                     | lock              | enabled  |
| lock_table_locks                        | lock              | disabled |
| lock_table_lock_created                 | lock              | disabled |
| lock_table_lock_removed                 | lock              | disabled |
| lock_table_lock_waits                   | lock              | disabled |
| lock_timeouts                           | lock              | enabled  |
| log_checkpoints                         | recovery          | disabled |
| log_lsn_buf_pool_oldest                 | recovery          | disabled |
| log_lsn_checkpoint_age                  | recovery          | disabled |
| log_lsn_current                         | recovery          | disabled |
| log_lsn_last_checkpoint                 | recovery          | disabled |
| log_lsn_last_flush                      | recovery          | disabled |
| log_max_modified_age_async              | recovery          | disabled |
| log_max_modified_age_sync               | recovery          | disabled |
| log_num_log_io                          | recovery          | disabled |
| log_pending_checkpoint_writes           | recovery          | disabled |
| log_pending_log_writes                  | recovery          | disabled |
| log_waits                               | recovery          | enabled  |
| log_writes                              | recovery          | enabled  |
| log_write_requests                      | recovery          | enabled  |
| metadata_mem_pool_size                  | metadata          | enabled  |
| metadata_table_handles_closed           | metadata          | disabled |
| metadata_table_handles_opened           | metadata          | disabled |
| metadata_table_reference_count          | metadata          | disabled |
| os_data_fsyncs                          | os                | enabled  |
| os_data_reads                           | os                | enabled  |
| os_data_writes                          | os                | enabled  |
| os_log_bytes_written                    | os                | enabled  |
| os_log_fsyncs                           | os                | enabled  |
| os_log_pending_fsyncs                   | os                | enabled  |
| os_log_pending_writes                   | os                | enabled  |
| os_pending_reads                        | os                | disabled |
| os_pending_writes                       | os                | disabled |
| purge_del_mark_records                  | purge             | disabled |
| purge_dml_delay_usec                    | purge             | disabled |
| purge_invoked                           | purge             | disabled |
| purge_resume_count                      | purge             | disabled |
| purge_stop_count                        | purge             | disabled |
| purge_undo_log_pages                    | purge             | disabled |
| purge_upd_exist_or_extern_records       | purge             | disabled |
| trx_active_transactions                 | transaction       | disabled |
| trx_commits_insert_update               | transaction       | disabled |
| trx_nl_ro_commits                       | transaction       | disabled |
| trx_rollbacks                           | transaction       | disabled |
| trx_rollbacks_savepoint                 | transaction       | disabled |
| trx_rollback_active                     | transaction       | disabled |
| trx_ro_commits                          | transaction       | disabled |
| trx_rseg_current_size                   | transaction       | disabled |
| trx_rseg_history_len                    | transaction       | enabled  |
| trx_rw_commits                          | transaction       | disabled |
| trx_undo_slots_cached                   | transaction       | disabled |
| trx_undo_slots_used                     | transaction       | disabled |
+-----------------------------------------+-------------------+----------+
214 rows in set (0.00 sec)
```

カウンタモジュール

モジュール名は INNODB_METRICS テーブルの SUBSYSTEM カラムの値に対応しますが、まったく同じではありません。カウンタの個別の有効化、無効化、またはリセットではなく、モジュール名を使用すると、特定のサブシステムのすべてのカウンタをすばやく有効または無効にしたり、リセットしたりすることができます。たとえば、dml サブシステムに関連付けられたすべてのカウンタを有効にするには、module_dml を使用します。

```
mysql> SET GLOBAL innodb_monitor_enable = module_dml;

mysql> SELECT name, subsystem, status FROM INFORMATION_SCHEMA.INNODB_METRICS
WHERE subsystem ='dml';
+-------------+-----------+---------+
| name        | subsystem | status  |
+-------------+-----------+---------+
```

```
| dml_reads   | dml       | enabled |
| dml_inserts | dml       | enabled |
| dml_deletes | dml       | enabled |
| dml_updates | dml       | enabled |
+-------------+-----------+---------+
4 rows in set (0.01 sec)
```

innodb_monitor_enable および関連する構成オプションで module_name に使用できる値を、対応する SUBSYSTEM の名前とともに次に示します。

- module_metadata (サブシステム = metadata)
- module_lock (サブシステム = lock)
- module_buffer (サブシステム = buffer)
- module_buf_page (サブシステム = buffer_page_io)
- module_os (サブシステム = os)
- module_trx (サブシステム = transaction)
- module_purge (サブシステム = purge)
- module_compress (サブシステム = compression)
- module_file (サブシステム = file_system)
- module_index (サブシステム = index)
- module_adaptive_hash (サブシステム = adaptive_hash_index)
- module_ibuf_system (サブシステム = change_buffer)
- module_srv (サブシステム = server)
- module_ddl (サブシステム = ddl)
- module_dml (サブシステム = dml)
- module_log (サブシステム = recovery)
- module_icp (サブシステム = icp)

例 14.22 INNODB_METRICS テーブルのカウンタの操作

この例では、カウンタの有効化、無効化、およびリセットと、INNODB_METRICS テーブル内のカウンタデータのクエリーを示します。

1. 単純な InnoDB テーブルを作成します。

```
mysql> USE test;
Database changed

mysql> CREATE TABLE t1 (c1 INT) ENGINE=INNODB;
Query OK, 0 rows affected (0.02 sec)
```

2. dml_inserts カウンタを有効にします。

```
mysql> SET GLOBAL innodb_monitor_enable = dml_inserts;
Query OK, 0 rows affected (0.01 sec)
```

dml_inserts カウンタの説明は、INNODB_METRICS テーブルの COMMENT カラムで見つけることができます。

```
mysql> SELECT NAME, COMMENT FROM INFORMATION_SCHEMA.INNODB_METRICS WHERE NAME="dml_inserts";
+-------------+-------------------------+
| NAME        | COMMENT                 |
+-------------+-------------------------+
| dml_inserts | Number of rows inserted |
+-------------+-------------------------+
1 row in set (0.00 sec)
```

3. dml_inserts カウンタデータを取得するために INNODB_METRICS テーブルをクエリーします。DML 操作が実行されていないため、カウンタ値は 0 または NULL です。TIME_ENABLED および TIME_ELAPSED 値は、このカウンタが最後に有効になった時間と、この時間から経過した秒数を示します。

```
mysql>  SELECT * FROM INFORMATION_SCHEMA.INNODB_METRICS WHERE NAME="dml_inserts" \G
*************************** 1. row ***************************
        NAME: dml_inserts
   SUBSYSTEM: dml
       COUNT: 0
   MAX_COUNT: 0
   MIN_COUNT: NULL
   AVG_COUNT: 0
 COUNT_RESET: 0
MAX_COUNT_RESET: 0
MIN_COUNT_RESET: NULL
AVG_COUNT_RESET: NULL
  TIME_ENABLED: 2014-12-04 14:18:28
 TIME_DISABLED: NULL
  TIME_ELAPSED: 28
   TIME_RESET: NULL
      STATUS: enabled
        TYPE: status_counter
     COMMENT: Number of rows inserted
1 row in set (0.01 sec)
```

4. テーブルに 3 行のデータを挿入します。

```
mysql> INSERT INTO t1 values(1);
Query OK, 1 row affected (0.00 sec)

mysql> INSERT INTO t1 values(2);
Query OK, 1 row affected (0.00 sec)

mysql> INSERT INTO t1 values(3);
Query OK, 1 row affected (0.00 sec)
```

5. dml_inserts カウンタデータを取得するために再度 INNODB_METRICS テーブルをクエリーします。COUNT、MAX_COUNT、AVG_COUNT、COUNT_RESET など、いくつかのカウンタ値が増分されています。これらの値の説明については、INNODB_METRICS テーブルの定義を参照してください。

```
mysql>  SELECT * FROM INFORMATION_SCHEMA.INNODB_METRICS WHERE NAME="dml_inserts"\G
*************************** 1. row ***************************
        NAME: dml_inserts
   SUBSYSTEM: dml
       COUNT: 3
   MAX_COUNT: 3
   MIN_COUNT: NULL
   AVG_COUNT: 0.046153846153846156
 COUNT_RESET: 3
MAX_COUNT_RESET: 3
MIN_COUNT_RESET: NULL
AVG_COUNT_RESET: NULL
  TIME_ENABLED: 2014-12-04 14:18:28
 TIME_DISABLED: NULL
  TIME_ELAPSED: 65
   TIME_RESET: NULL
      STATUS: enabled
        TYPE: status_counter
     COMMENT: Number of rows inserted
1 row in set (0.00 sec)
```

6. dml_inserts カウンタをリセットし、dml_inserts カウンタデータを取得するために再度 INNODB_METRICS テーブルをクエリーします。COUNT_RESET や MAX_RESET などの、前にレポートされた「%_RESET」値が 0 に戻っています。カウンタが有効になった時点から累積してデータを収集する COUNT、MAX_COUNT、AVG_COUNT などの値はリセットの影響を受けません。

```
mysql> SET GLOBAL innodb_monitor_reset = dml_inserts;
Query OK, 0 rows affected (0.00 sec)

mysql> SELECT * FROM INFORMATION_SCHEMA.INNODB_METRICS WHERE NAME="dml_inserts"\G
*************************** 1. row ***************************
        NAME: dml_inserts
   SUBSYSTEM: dml
       COUNT: 3
   MAX_COUNT: 3
   MIN_COUNT: NULL
   AVG_COUNT: 0.03529411764705882
 COUNT_RESET: 0
MAX_COUNT_RESET: 0
MIN_COUNT_RESET: NULL
AVG_COUNT_RESET: 0
  TIME_ENABLED: 2014-12-04 14:18:28
```

```
 TIME_DISABLED: NULL
  TIME_ELAPSED: 85
    TIME_RESET: 2014-12-04 14:19:44
        STATUS: enabled
          TYPE: status_counter
       COMMENT: Number of rows inserted
1 row in set (0.00 sec)
```

7. すべてのカウンタ値をリセットするには、まずそのカウンタを無効にする必要があります。カウンタを無効にすると、STATUS 値が disbaled に設定されます。

```
mysql> SET GLOBAL innodb_monitor_disable = dml_inserts;
Query OK, 0 rows affected (0.00 sec)

mysql> SELECT * FROM INFORMATION_SCHEMA.INNODB_METRICS WHERE NAME="dml_inserts"\G
*************************** 1. row ***************************
           NAME: dml_inserts
      SUBSYSTEM: dml
          COUNT: 3
      MAX_COUNT: 3
      MIN_COUNT: NULL
      AVG_COUNT: 0.030612244897959183
    COUNT_RESET: 0
MAX_COUNT_RESET: 0
MIN_COUNT_RESET: NULL
AVG_COUNT_RESET: 0
   TIME_ENABLED: 2014-12-04 14:18:28
  TIME_DISABLED: 2014-12-04 14:20:06
   TIME_ELAPSED: 98
     TIME_RESET: NULL
         STATUS: disabled
           TYPE: status_counter
        COMMENT: Number of rows inserted
1 row in set (0.00 sec)
```

> **注記**
>
> カウンタおよびモジュール名にはワイルドカードマッチングがサポートされています。たとえば、dml_inserts カウンタの完全な名前を指定する代わりに、「dml_i%」を指定できます。また、ワイルドカードマッチングを使用して、複数のカウンタまたはモジュールを一度に有効または無効にしたり、リセットしたりすることもできます。たとえば、「dml_%」で始まるすべてのカウンタを有効または無効にしたり、リセットしたりするには、「dml_%」を指定します。

8. カウンタが無効になったら、innodb_monitor_reset_all オプションを使用して、すべてのカウンタ値をリセットできます。すべての値が 0 または NULL に設定されます。

```
mysql> SET GLOBAL innodb_monitor_reset_all = dml_inserts;
Query OK, 0 rows affected (0.00 sec)

mysql> SELECT * FROM INFORMATION_SCHEMA.INNODB_METRICS WHERE NAME="dml_inserts"\G
*************************** 1. row ***************************
           NAME: dml_inserts
      SUBSYSTEM: dml
          COUNT: 0
      MAX_COUNT: NULL
      MIN_COUNT: NULL
      AVG_COUNT: NULL
    COUNT_RESET: 0
MAX_COUNT_RESET: NULL
MIN_COUNT_RESET: NULL
AVG_COUNT_RESET: NULL
   TIME_ENABLED: NULL
  TIME_DISABLED: NULL
   TIME_ELAPSED: NULL
     TIME_RESET: NULL
         STATUS: disabled
           TYPE: status_counter
        COMMENT: Number of rows inserted
1 row in set (0.01 sec)
```

14.15 InnoDB モニター

InnoDB モニターは、InnoDB の内部状態に関する情報を提供します。この情報は、パフォーマンスチューニングに役立ちます。

14.15.1 InnoDB モニターのタイプ

InnoDB モニターには、次の 4 つのタイプがあります。

- InnoDB 標準モニターは、次のタイプの情報を表示します。
  - アクティブな各トランザクションによって保持されているテーブルおよびレコードロック。
  - トランザクションのロック待機。
  - スレッドのセマフォー待機。
  - 保留中のファイル I/O 要求。
  - バッファープールの統計。
  - メインの InnoDB スレッドのパージおよび挿入バッファーマージアクティビティー。
- InnoDB ロックモニターは InnoDB 標準モニターに似ていますが、広範囲にわたるロック情報も提供します。
- InnoDB テーブルスペースモニターは、共有テーブルスペース内のファイルセグメントのリストを出力したり、テーブルスペースの割り当てデータ構造を検証したりします。
- InnoDB テーブルモニターは、InnoDB 内部データディクショナリの内容を出力します。

> **注記**
>
> テーブルスペースモニターとテーブルモニターは非推奨であり、将来の MySQL リリースで削除される予定です。テーブルモニターと同様の情報は、InnoDB INFORMATION_SCHEMA テーブルから取得できます。セクション21.29「InnoDB の INFORMATION_SCHEMA テーブル」を参照してください。

InnoDB モニターの詳細は、次を参照してください。

- Mark Leith: InnoDB テーブルおよびテーブルスペースモニターに関する記事

14.15.2 InnoDB モニターの有効化

InnoDB モニターでの定期的な出力を有効にすると、InnoDB は、その出力を mysqld サーバーの標準エラー出力 (stderr) に書き込みます。この場合、クライアントには出力が送信されません。オンに切り替えられると、InnoDB モニターは約 15 秒に 1 回データを出力します。サーバーの出力は通常、エラーログに送信されます (セクション5.2.2「エラーログ」を参照してください)。このデータは、パフォーマンスチューニングに役立ちます。Windows では、出力をエラーログにではなくウィンドウに送信したい場合は、コンソールウィンドウのコマンドプロンプトから --console オプションを使用してサーバーを起動します。

InnoDB は、バッファーオーバーフローの可能性を回避するために、診断の出力を stdout または固定サイズのメモリーバッファーにではなく、stderr またはファイルに送信します。副作用として、SHOW ENGINE INNODB STATUS の出力が MySQL データディレクトリ内のステータスファイルに 15 秒に 1 回書き込まれます。このファイルの名前は innodb_status.pid です。ここで、pid はサーバープロセス ID です。InnoDB は、正常なシャットダウンのときにこのファイルを削除します。異常なシャットダウンが発生した場合は、これらのステータスファイルのインスタンスが存在する可能性があるため、手動で削除する必要があります。削除する前に、これらのファイルを検査して、異常なシャットダウンの原因に関する有効な情報が含まれているかどうかを確認することをお勧めします。innodb_status.pid ファイルは、構成オプション innodb-status-file=1 が設定されている場合にのみ作成されます。

出力の生成によってパフォーマンスはある程度低下するため、InnoDB モニターは、実際にモニター情報の確認が必要な場合にのみ有効にするようにしてください。また、関連付けられたテーブルを作成することによってモニターの出力を有効にした場合は、あとでテーブルを削除することを忘れると、エラーログがきわめて大きくなることがあります。

> **注記**
>
> トラブルシューティングを支援するために、InnoDB は、特定の状況で InnoDB 標準モニターの出力を一時的に有効にします。詳細は、セクション14.19「InnoDB のトラブルシューティング」を参照してください。

各モニターは、タイムスタンプとモニター名が含まれたヘッダーで始まります。例:

```
=====================================
```

```
2014-10-16 16:28:15 7feee43c5700 INNODB MONITOR OUTPUT
=====================================
```

ロックモニターでは、追加のロック情報が付加された同じ出力が生成されるため、InnoDB 標準モニターのヘッダー (INNODB MONITOR OUTPUT) はロックモニターにも使用されます。

InnoDB モニターでの定期的な出力を有効にするには、CREATE TABLE ステートメントを使用して、そのモニターに関連付けられた特別な名前の付いた InnoDB テーブルを作成します。たとえば、InnoDB 標準モニターを有効にするには、innodb_monitor という名前の InnoDB テーブルを作成します。

CREATE TABLE 構文の使用は、MySQL の SQL パーサーを経由して InnoDB エンジンにコマンドを渡すための方法にすぎません。重要なのは、テーブル名と、それが InnoDB テーブルであるということだけです。テーブルの構造は関係ありません。サーバーをシャットダウンした場合は、サーバーを再起動しても、モニターは自動的に再開されません。モニターテーブルを削除し、新しい CREATE TABLE ステートメントを発行してモニターを開始します。

> **注記**
>
> InnoDB モニターを有効にするための CREATE TABLE の方法は非推奨であり、将来のリリースで削除される可能性があります。MySQL 5.6.16 の時点では、innodb_status_output および innodb_status_output_locks システム変数を使用して InnoDB 標準モニターおよび InnoDB ロックモニターを有効にすることができます。

InnoDB モニターを無効および有効にするには、PROCESS 権限が必要です。

InnoDB 標準モニターの有効化

InnoDB 標準モニターでの定期的な出力を有効にするには、innodb_monitor テーブルを作成します。

```
CREATE TABLE innodb_monitor (a INT) ENGINE=INNODB;
```

InnoDB 標準モニターを無効にするには、そのテーブルを削除します。

```
DROP TABLE innodb_monitor;
```

MySQL 5.6.16 の時点では、innodb_status_output システム変数を ON に設定することによって InnoDB 標準モニターを有効にすることもできます。

```
set GLOBAL innodb_status_output=ON;
```

InnoDB 標準モニターを無効にするには、innodb_status_output を OFF に設定します。

サーバーをシャットダウンすると、innodb_status_output 変数がデフォルトの OFF 値に設定されます。

オンデマンドでの InnoDB 標準モニターの出力の取得

InnoDB 標準モニターでの定期的な出力を有効にする代わりに、出力をクライアントプログラムにフェッチする SHOW ENGINE INNODB STATUS SQL ステートメントを使用して、オンデマンドで InnoDB 標準モニターの出力を取得できます。mysql 対話型クライアントを使用している場合は、通常のセミコロンのステートメントターミネータを \G に置き換えると、出力が読み取りやすくなります。

```
mysql> SHOW ENGINE INNODB STATUS\G
```

InnoDB ロックモニターの有効化

InnoDB ロックモニターでの定期的な出力を有効にするには、innodb_lock_monitor テーブルを作成します。

```
CREATE TABLE innodb_lock_monitor (a INT) ENGINE=INNODB;
```

InnoDB ロックモニターを無効にするには、そのテーブルを削除します。

```
DROP TABLE innodb_lock_monitor;
```

MySQL 5.6.16 の時点では、innodb_status_output および innodb_status_output_locks システム変数の両方を ON に設定することによって InnoDB ロックモニターを有効にすることもできます。ロックモニターの出力は InnoDB 標準モニターの出力とともに出力されるため、ロックモニターの出力を有効にするには、両方のモニターを有効にする必要があります。

```
set GLOBAL innodb_status_output=ON;
set GLOBAL innodb_status_output_locks=ON;
```

サーバーをシャットダウンすると、innodb_status_output および innodb_status_output_locks 変数がデフォルトの OFF 値に設定されます。

InnoDB ロックモニターを無効にするには、innodb_status_output_locks を OFF に設定します。InnoDB 標準モニターも無効にするには、innodb_status_output を OFF に設定します。

InnoDB テーブルスペースモニターの有効化

InnoDB テーブルスペースモニターでの定期的な出力を有効にするには、innodb_tablespace_monitor テーブルを作成します。

```
CREATE TABLE innodb_tablespace_monitor (a INT) ENGINE=INNODB;
```

標準の InnoDB テーブルスペースモニターを無効にするには、そのテーブルを削除します。

```
DROP TABLE innodb_tablespace_monitor;
```

> **注記**
>
> テーブルスペースモニターは非推奨であり、将来の MySQL リリースで削除される予定です。

InnoDB テーブルモニターの有効化

InnoDB テーブルモニターでの定期的な出力を有効にするには、innodb_table_monitor テーブルを作成します。

```
CREATE TABLE innodb_table_monitor (a INT) ENGINE=INNODB;
```

InnoDB テーブルモニターを無効にするには、そのテーブルを削除します。

```
DROP TABLE innodb_table_monitor;
```

> **注記**
>
> テーブルスペースモニターは非推奨であり、将来の MySQL リリースで削除される予定です。

14.15.3 InnoDB 標準モニターおよびロックモニターの出力

ロックモニターは、追加のロック情報を含んでいる点を除き、標準モニターと同じです。どちらのモニターの定期的な出力を有効にしても、同じ出力ストリームが有効になりますが、ロックモニターが有効になっている場合は、そのストリームに追加の情報が含まれます。たとえば、InnoDB 標準モニターおよび InnoDB ロックモニターを有効にすると、1 つの出力ストリームが有効になります。ロックモニターを無効にするまで、そのストリームには追加のロック情報が含まれます。

InnoDB モニターの出力例 (MySQL 5.6.22 の時点):

```
mysql> SHOW ENGINE INNODB STATUS\G
*************************** 1. row ***************************
  Type: InnoDB
  Name:
Status:
=====================================
2014-10-17 10:33:50 7f47bcd64700 INNODB MONITOR OUTPUT
=====================================
Per second averages calculated from the last 6 seconds
-----------------
BACKGROUND THREAD
-----------------
srv_master_thread loops: 167 srv_active, 0 srv_shutdown, 3023 srv_idle
srv_master_thread log flush and writes: 3190
----------
SEMAPHORES
----------
OS WAIT ARRAY INFO: reservation count 1040
OS WAIT ARRAY INFO: signal count 959
Mutex spin waits 677, rounds 20336, OS waits 644
RW-shared spins 180, rounds 5400, OS waits 180
RW-excl spins 0, rounds 6420, OS waits 214
Spin rounds per wait: 30.04 mutex, 30.00 RW-shared, 6420.00 RW-excl
------------------------
LATEST FOREIGN KEY ERROR
------------------------
2014-10-17 09:51:31 7f47bcde6700 Transaction:
TRANSACTION 436786, ACTIVE 0 sec inserting
mysql tables in use 1, locked 1
4 lock struct(s), heap size 1184, 3 row lock(s), undo log entries 3
MySQL thread id 1, OS thread handle 0x7f47bcde6700, query id 96 localhost
```

```
root update
INSERT INTO child VALUES
    (NULL, 1)
   , (NULL, 2)
   , (NULL, 3)
   , (NULL, 4)
   , (NULL, 5)
   , (NULL, 6)
Foreign key constraint fails for table `mysql`.`child`:
,
  CONSTRAINT `child_ibfk_1` FOREIGN KEY (`parent_id`) REFERENCES `parent`
  (`id`)
 ON DELETE CASCADE ON UPDATE CASCADE
Trying to add in child table, in index `par_ind` tuple:
DATA TUPLE: 2 fields;
 0: len 4; hex 80000003; asc     ;;
 1: len 4; hex 80000003; asc     ;;

But in parent table `mysql`.`parent`, in index `PRIMARY`,
the closest match we can find is record:
PHYSICAL RECORD: n_fields 3; compact format; info bits 0
 0: len 4; hex 80000004; asc     ;;
 1: len 6; hex 00000006aa26; asc      &;;
 2: len 7; hex 9d000001610137; asc     a 7;;
------------------------
LATEST DETECTED DEADLOCK
------------------------
2014-10-17 09:52:38 7f47bcde6700
*** (1) TRANSACTION:
TRANSACTION 436801, ACTIVE 12 sec starting index read
mysql tables in use 1, locked 1
LOCK WAIT 2 lock struct(s), heap size 360, 1 row lock(s)
MySQL thread id 2, OS thread handle 0x7f47bcda5700, query id 102 localhost
root updating
DELETE FROM t WHERE i = 1
*** (1) WAITING FOR THIS LOCK TO BE GRANTED:
RECORD LOCKS space id 3693 page no 3 n bits 72 index `GEN_CLUST_INDEX` of
table `mysql`.`t` trx id 436801 lock_mode X waiting
Record lock, heap no 2 PHYSICAL RECORD: n_fields 4; compact format; info
bits 0
 0: len 6; hex 000000003a00; asc     : ;;
 1: len 6; hex 00000006aa3f; asc      ?;;
 2: len 7; hex ad0000021d0110; asc        ;;
 3: len 4; hex 80000001; asc     ;;

*** (2) TRANSACTION:
TRANSACTION 436800, ACTIVE 34 sec starting index read
mysql tables in use 1, locked 1
4 lock struct(s), heap size 1184, 3 row lock(s)
MySQL thread id 1, OS thread handle 0x7f47bcde6700, query id 103 localhost
root updating
DELETE FROM t WHERE i = 1
*** (2) HOLDS THE LOCK(S):
RECORD LOCKS space id 3693 page no 3 n bits 72 index `GEN_CLUST_INDEX` of
table `mysql`.`t` trx id 436800 lock mode S
Record lock, heap no 1 PHYSICAL RECORD: n_fields 1; compact format; info
bits 0 0: len 8; hex 73757072656d756d; asc supremum;;

Record lock, heap no 2 PHYSICAL RECORD: n_fields 4; compact format; info
bits 0
 0: len 6; hex 000000003a00; asc     : ;;
 1: len 6; hex 00000006aa3f; asc      ?;;
 2: len 7; hex ad0000021d0110; asc        ;;
 3: len 4; hex 80000001; asc     ;;

*** (2) WAITING FOR THIS LOCK TO BE GRANTED:
RECORD LOCKS space id 3693 page no 3 n bits 72 index `GEN_CLUST_INDEX` of
table `mysql`.`t` trx id 436800 lock_mode X waiting
Record lock, heap no 2 PHYSICAL RECORD: n_fields 4; compact format; info
bits 0
 0: len 6; hex 000000003a00; asc     : ;;
 1: len 6; hex 00000006aa3f; asc      ?;;
 2: len 7; hex ad0000021d0110; asc        ;;
 3: len 4; hex 80000001; asc     ;;

*** WE ROLL BACK TRANSACTION (1)
------------
TRANSACTIONS
------------
Trx id counter 437661
```

```
Purge done for trx's n:o < 437657 undo n:o < 0 state: running but
idle History list length 371
LIST OF TRANSACTIONS FOR EACH SESSION:
---TRANSACTION 0, not started
MySQL thread id 10, OS thread handle 0x7f47bcd64700, query id 1001 localhost
root init
SHOW ENGINE INNODB STATUS
---TRANSACTION 436801, not started
MySQL thread id 2, OS thread handle 0x7f47bcda5700, query id 102 localhost
root ceaning up
---TRANSACTION 437660, ACTIVE 0 sec inserting
mysql tables in use 1, locked 1
43 lock struct(s), heap size 6544, 6474 row lock(s), undo log entries 7124
MySQL thread id 14, OS thread handle 0x7f47bcde6700, query id 1000 localhost
root update
INSERT INTO `dept_emp` VALUES (100258,'d002','1994-03-21','9999-01-01'),
(100259, 'd005','1998-11-04','9999-01-01'),(100259,'d008','1988-02-03',
'1998-11-04'),(100 260,'d005','1998-09-18','9999-01-01'),(100261,'d004',
'1989-03-11','9999-01-01'), (100262,'d008','1996-08-12','9999-01-01'),
(100263,'d002','1998-06-24','1998-10-0 5'),(100264,'d005','1989-11-09',
'9999-01-01'),(100265,'d001','1992-06-27','9999- 01-01'),(100266,'d009',
'1990-09-10','9999-01-01'),(100267,'d009','1992-04-14','9 999-01-01'),
(100268,'d005','1998-05-01','2000-04-07'),(100269,'d007','1994-01-02',
'1999-09-18'),(100269,'d009','1999-09-
--------
FILE I/O
--------
I/O thread 0 state: waiting for completed aio requests (insert buffer thread)
I/O thread 1 state: waiting for completed aio requests (log thread)
I/O thread 2 state: waiting for completed aio requests (read thread)
I/O thread 3 state: waiting for completed aio requests (read thread)
I/O thread 4 state: waiting for completed aio requests (read thread)
I/O thread 5 state: waiting for completed aio requests (read thread)
I/O thread 6 state: waiting for completed aio requests (write thread)
I/O thread 7 state: waiting for completed aio requests (write thread)
I/O thread 8 state: waiting for completed aio requests (write thread)
I/O thread 9 state: waiting for completed aio requests (write thread)
Pending normal aio reads: 0 [0, 0, 0, 0] , aio writes: 0 [0, 0, 0, 0] ,
 ibuf aio reads: 0, log i/o's: 0, sync i/o's: 0
Pending flushes (fsync) log: 0; buffer pool: 0
344 OS file reads, 45666 OS file writes, 4030 OS fsyncs
0.00 reads/s, 0 avg bytes/read, 202.80 writes/s, 48.33 fsyncs/s
-------------------------------------
INSERT BUFFER AND ADAPTIVE HASH INDEX
-------------------------------------
Ibuf: size 1, free list len 0, seg size 2, 0 merges
merged operations:
 insert 0, delete mark 0, delete 0
discarded operations:
 insert 0, delete mark 0, delete 0
Hash table size 4425293, node heap has 143 buffer(s)
137083.82 hash searches/s, 2495.92 non-hash searches/s
---
LOG
---
Log sequence number 3091027710
Log flushed up to   3090240098
Pages flushed up to 3074432960
Last checkpoint at  3050856266
0 pending log writes, 0 pending chkp writes
1187 log i/o's done, 14.67 log i/o's/second
----------------------
BUFFER POOL AND MEMORY
----------------------
Total memory allocated 2197815296; in additional pool allocated 0
Dictionary memory allocated 155455
Buffer pool size   131071
Free buffers       92158
Database pages     38770
Old database pages 14271
Modified db pages  619
Pending reads 0
Pending writes: LRU 0, flush list 0, single page 0
Pages made young 4, not young 0
0.00 youngs/s, 0.00 non-youngs/s
Pages read 322, created 38448, written 42083
0.00 reads/s, 222.30 creates/s, 159.47 writes/s
Buffer pool hit rate 1000 / 1000, young-making rate 0 / 1000 not 0 / 1000
Pages read ahead 0.00/s, evicted without access 0.00/s, Random read ahead
0.00/s
```

```
LRU len: 38770, unzip_LRU len: 0
I/O sum[0]:cur[0], unzip sum[0]:cur[0]
----------------------
INDIVIDUAL BUFFER POOL INFO
----------------------
---BUFFER POOL 0
Buffer pool size   65536
Free buffers       46120
Database pages     19345
Old database pages 7121
Modified db pages  291
Pending reads 0
Pending writes: LRU 0, flush list 0, single page 0
Pages made young 3, not young 0
0.00 youngs/s, 0.00 non-youngs/s
Pages read 163, created 19182, written 21149
0.00 reads/s, 103.48 creates/s, 83.15 writes/s
Buffer pool hit rate 1000 / 1000, young-making rate 0 / 1000 not 0 / 1000
Pages read ahead 0.00/s, evicted without access 0.00/s, Random read ahead
0.00/s
LRU len: 19345, unzip_LRU len: 0
I/O sum[0]:cur[0], unzip sum[0]:cur[0]
---BUFFER POOL 1
Buffer pool size   65535
Free buffers       46038
Database pages     19425
Old database pages 7150
Modified db pages  328
Pending reads 0
Pending writes: LRU 0, flush list 0, single page 0
Pages made young 1, not young 0
0.00 youngs/s, 0.00 non-youngs/s
Pages read 159, created 19266, written 20934
0.00 reads/s, 118.81 creates/s, 76.32 writes/s
Buffer pool hit rate 1000 / 1000, young-making rate 0 / 1000 not 0 / 1000
Pages read ahead 0.00/s, evicted without access 0.00/s, Random read ahead
0.00/s
LRU len: 19425, unzip_LRU len: 0
I/O sum[0]:cur[0], unzip sum[0]:cur[0]
--------------
ROW OPERATIONS
--------------
0 queries inside InnoDB, 0 queries in queue
0 read views open inside InnoDB
Main thread process no. 54607, id 139946075744000, state: sleeping
Number of rows inserted 12163964, updated 0, deleted 3, read 4
67807.03 inserts/s, 0.00 updates/s, 0.00 deletes/s, 0.00 reads/s
----------------------------
END OF INNODB MONITOR OUTPUT
============================
```

SHOW ENGINE INNODB STATUS ステートメントを使用して生成された場合、InnoDB 標準モニターの出力は 1M バイトに制限されます。この制限は、サーバーのエラー出力に書き込まれる出力には適用されません。

出力のセクションに関するいくつかの注意点:

Status

このセクションは、タイムスタンプ、モニター名、および 1 秒あたりの平均の基になる秒数を示します。この秒数は、現在の時間と InnoDB モニターの出力が最後に出力された時間の間の経過時間です。

BACKGROUND THREAD

srv_master_thread 行は、メインのバックグラウンドスレッドによって実行された作業を示します。

SEMAPHORES

このセクションは、セマフォーを待機しているスレッド、およびスレッドが相互排他ロックまたは読み書きロックセマフォーでスピンまたは待機を必要とした回数に関する統計をレポートします。多数のスレッドがセマフォーを待機している場合は、ディスク I/O または InnoDB 内部の競合の問題の結果である可能性があります。競合は、クエリーの高い並列性、またはオペレーティングシステムのスレッドスケジューリングでの問題が原因である場合があります。このような状況では、innodb_thread_concurrency システム変数をデフォルト値より小さい値に設定すると役立つことがあります。Spin rounds per wait 行は、相互排他ロックでの OS ウェイトあたりのスピンロックラウンドの数を示します。

LATEST FOREIGN KEY ERROR

このセクションは、最新の外部キー制約エラーに関する情報を提供します。このようなエラーが発生していない場合は存在しません。その内容には、失敗したステートメントのほか、失敗した制約や、参照されるテーブルと参照するテーブルに関する情報が含まれます。

LATEST DETECTED DEADLOCK

このセクションは、最新のデッドロックに関する情報を提供します。デッドロックが発生していない場合は存在しません。その内容には、関連しているトランザクション、各トランザクションが実行しようとしていたステートメント、それぞれが保持しているロックと必要なロック、およびデッドロックを解消するために InnoDB がロールバックすることを決定したトランザクションが示されます。このセクションでレポートされるロックモードについては、セクション14.2.3「InnoDB のロックモード」で説明されています。

TRANSACTIONS

このセクションでロック待機がレポートされている場合は、アプリケーションでロック競合が発生している可能性があります。この出力はまた、トランザクションデッドロックの原因の追跡にも役立つことがあります。

FILE I/O

このセクションは、InnoDB がさまざまなタイプの I/O を実行するために使用するスレッドに関する情報を提供します。このうちの最初の数行は、InnoDB の一般的な処理に専用に使用されます。この内容には、保留中の I/O 操作や I/O パフォーマンスの統計に関する情報も表示されます。

これらのスレッドの数は、innodb_read_io_threads および innodb_write_io_threads パラメータによって制御されます。セクション14.12「InnoDB の起動オプションおよびシステム変数」を参照してください。

INSERT BUFFER AND ADAPTIVE HASH INDEX

このセクションは、InnoDB 挿入バッファーおよびアダプティブハッシュインデックスのステータスを示します。(セクション14.2.13.5「挿入バッファー」およびセクション14.2.13.6「適応型ハッシュインデックス」を参照してください。)その内容には、それぞれに対して実行された操作の数のほか、ハッシュインデックスのパフォーマンスの統計が含まれます。

LOG

このセクションには、InnoDB のログに関する情報が表示されます。その内容には、現在のログシーケンス番号、ログがディスクにフラッシュされた範囲、および InnoDB が最後にチェックポイントを取得した位置が含まれます。(セクション14.10.3「InnoDB チェックポイント」を参照してください。)このセクションには、保留中の書き込みや書き込みパフォーマンスの統計に関する情報も表示されます。

BUFFER POOL AND MEMORY

このセクションは、読み取られたページと書き込まれたページに関する統計を提供します。これらの数値から、現在クエリーが実行しているデータファイル I/O 操作の数を計算できます。

バッファープールの操作の詳細は、セクション8.9.1「InnoDB バッファープール」を参照してください。

ROW OPERATIONS

このセクションは、メインスレッドが実行している内容 (各タイプの行操作の数とパフォーマンスレートを含む) を示します。

14.15.4 InnoDB テーブルスペースモニターの出力

> 注記
>
> InnoDB テーブルスペースモニターは非推奨であり、将来のリリースで削除される可能性があります。

InnoDB テーブルスペースモニターは、共有テーブルスペース内のファイルセグメントに関する情報を出力したり、テーブルスペースの割り当てデータ構造を検証したりします。テーブルスペースモニターでは、innodb_file_per_table オプションで作成された file-per-table テーブルスペースは記載されません。

InnoDB テーブルスペースモニターの出力例:

```
=====================================================
090408 21:28:09 INNODB TABLESPACE MONITOR OUTPUT
=====================================================
FILE SPACE INFO: id 0
size 13440, free limit 3136, free extents 28
```

```
not full frag extents 2: used pages 78, full frag extents 3
first seg id not used 0 23845
SEGMENT id 0 1 space 0; page 2; res 96 used 46; full ext 0
fragm pages 32; free extents 0; not full extents 1: pages 14
SEGMENT id 0 2 space 0; page 2; res 1 used 1; full ext 0
fragm pages 1; free extents 0; not full extents 0: pages 0
SEGMENT id 0 3 space 0; page 2; res 1 used 1; full ext 0
fragm pages 1; free extents 0; not full extents 0: pages 0
...
SEGMENT id 0 15 space 0; page 2; res 160 used 160; full ext 2
fragm pages 32; free extents 0; not full extents 0: pages 0
SEGMENT id 0 488 space 0; page 2; res 1 used 1; full ext 0
fragm pages 1; free extents 0; not full extents 0: pages 0
SEGMENT id 0 17 space 0; page 2; res 1 used 1; full ext 0
fragm pages 1; free extents 0; not full extents 0: pages 0
...
SEGMENT id 0 171 space 0; page 2; res 592 used 481; full ext 7
fragm pages 16; free extents 0; not full extents 2: pages 17
SEGMENT id 0 172 space 0; page 2; res 1 used 1; full ext 0
fragm pages 1; free extents 0; not full extents 0: pages 0
SEGMENT id 0 173 space 0; page 2; res 96 used 44; full ext 0
fragm pages 32; free extents 0; not full extents 1: pages 12
...
SEGMENT id 0 601 space 0; page 2; res 1 used 1; full ext 0
fragm pages 1; free extents 0; not full extents 0: pages 0
NUMBER of file segments: 73
Validating tablespace
Validation ok
---------------------------------------
END OF INNODB TABLESPACE MONITOR OUTPUT
=======================================
```

テーブルスペースモニターの出力には、共有テーブルスペースに関する全体的な情報のあとに、テーブルスペース内のセグメントごとの内訳を含むリストが含まれます。

デフォルトのページサイズを使用したこの例では、テーブルスペースが、それぞれ 16K バイトであるデータベースページで構成されています。これらのページは、サイズが 1M バイト (64 個の連続したページ) のエクステントにグループ化されています。

全体的なテーブルスペース情報を表示する出力の最初の部分の形式は次のとおりです。

```
FILE SPACE INFO: id 0
size 13440, free limit 3136, free extents 28
not full frag extents 2: used pages 78, full frag extents 3
first seg id not used 0 23845
```

全体的なテーブルスペース情報には、次の値が含まれます。

- id: テーブルスペース ID。0 の値は、共有テーブルスペースを示します。
- size: 現在のテーブルスペースサイズ (ページ数)。
- free limit: 空きリストが初期化されていない最小のページ番号。この制限の位置にあるページ、またはそれより上のページは空いています。
- free extents: 空きエクステントの数。
- not full frag extents、used pages: 完全にはいっぱいになっていないフラグメントエクステントの数、およびこれらのエクステント内の割り当てられたページの数。
- full frag extents: 完全にいっぱいになっているフラグメントエクステントの数。
- first seg id not used: 使用されていない最初のセグメント ID。

個々のセグメント情報の形式は次のとおりです。

```
SEGMENT id 0 15 space 0; page 2; res 160 used 160; full ext 2
fragm pages 32; free extents 0; not full extents 0: pages 0
```

セグメント情報には、次の値が含まれます。

id: セグメント ID。

space、page: テーブルスペース番号、およびこのセグメントの「i ノード」が格納されているテーブルスペース内のページ。0 のテーブルスペース番号は、共有テーブルスペースを示します。InnoDB は、i ノードを使用して

テーブルスペース内のセグメントを追跡します。セグメントに関して表示されるその他のフィールド (id や res など) は、i ノード内の情報から取得されます。

res: このセグメントに割り当てられている (予約されている) ページの数。

used: このセグメントで使用されている割り当てられたページの数。

full ext: このセグメントに割り当てられたエクステントのうち、完全に使用されているものの数。

fragm pages: このセグメントに割り当てられた最初のページの数。

free extents: このセグメントに割り当てられたエクステントのうち、完全に未使用であるものの数。

not full extents: このセグメントに割り当てられたエクステントのうち、部分的に使用されているものの数。

pages: いっぱいになっていないエクステント内で使用されているページの数。

セグメントが拡張される場合、そのセグメントは 1 ページとして開始されますが、InnoDB は最初の数ページ (最大 32 ページ、これは fragm pages 値です) を一度に割り当てます。そのあと、InnoDB は完全なエクステントを割り当てます。InnoDB は、データの良好な連続性を保証するために、大きなセグメントには 1 回につき最大 4 つのエクステントを追加できます。

前に示されたセグメントの例では、32 個のフラグメントページに加え、いっぱいになったエクステントが 2 つあり (それぞれ 64 ページ)、割り当てられた 160 ページのうち合計 160 ページが使用されています。次のセグメントには、32 個のフラグメントページと、14 ページを使用している部分的にいっぱいになったエクステントが 1 つあり、割り当てられた 96 ページのうち合計 46 ページが使用されています。

```
SEGMENT id 0 1 space 0; page 2; res 96 used 46; full ext 0
fragm pages 32; free extents 0; not full extents 1: pages 14
```

エクステントの割り当てのあとに個々のページのいくつかが解放された場合は、エクステントが割り当てられているセグメントの fragm pages 値が 32 より小さい可能性があります。

14.15.5 InnoDB テーブルモニターの出力

> **注記**
>
> InnoDB テーブルモニターは非推奨であり、将来のリリースで削除される可能性があります。同様の情報は、InnoDB INFORMATION_SCHEMA テーブルから取得できます。セクション21.29「InnoDB の INFORMATION_SCHEMA テーブル」を参照してください。

InnoDB テーブルモニターは、InnoDB 内部データディクショナリの内容を出力します。

この出力には、テーブルごとに 1 つのセクションが含まれます。SYS_FOREIGN および SYS_FOREIGN_COLS セクションは、外部キーに関する情報を保持する内部データディクショナリテーブルのためのものです。また、テーブルモニターのテーブルや、ユーザーが作成した各 InnoDB テーブルのためのセクションもあります。test データベース内に次の 2 つのテーブルが作成されたとします。

```
CREATE TABLE parent
(
  par_id    INT NOT NULL,
  fname      CHAR(20),
  lname      CHAR(20),
  PRIMARY KEY (par_id),
  UNIQUE INDEX (lname, fname)
) ENGINE = INNODB;

CREATE TABLE child
(
  par_id      INT NOT NULL,
  child_id    INT NOT NULL,
  name        VARCHAR(40),
  birth       DATE,
  weight      DECIMAL(10,2),
  misc_info   VARCHAR(255),
  last_update TIMESTAMP,
  PRIMARY KEY (par_id, child_id),
  INDEX (name),
  FOREIGN KEY (par_id) REFERENCES parent (par_id)
    ON DELETE CASCADE
    ON UPDATE CASCADE
```

```
) ENGINE = INNODB;
```

この場合、テーブルモニターの出力は次のようになります (形式を少し変更しました)。

```
===========================================
090420 12:09:32 INNODB TABLE MONITOR OUTPUT
===========================================
--------------------------------------
TABLE: name SYS_FOREIGN, id 0 11, columns 7, indexes 3, appr.rows 1
 COLUMNS: ID: DATA_VARCHAR DATA_ENGLISH len 0;
          FOR_NAME: DATA_VARCHAR DATA_ENGLISH len 0;
          REF_NAME: DATA_VARCHAR DATA_ENGLISH len 0;
          N_COLS: DATA_INT len 4;
          DB_ROW_ID: DATA_SYS prtype 256 len 6;
          DB_TRX_ID: DATA_SYS prtype 257 len 6;
 INDEX: name ID_IND, id 0 11, fields 1/6, uniq 1, type 3
  root page 46, appr.key vals 1, leaf pages 1, size pages 1
  FIELDS:  ID DB_TRX_ID DB_ROLL_PTR FOR_NAME REF_NAME N_COLS
 INDEX: name FOR_IND, id 0 12, fields 1/2, uniq 2, type 0
  root page 47, appr.key vals 1, leaf pages 1, size pages 1
  FIELDS:  FOR_NAME ID
 INDEX: name REF_IND, id 0 13, fields 1/2, uniq 2, type 0
  root page 48, appr.key vals 1, leaf pages 1, size pages 1
  FIELDS:  REF_NAME ID
--------------------------------------
TABLE: name SYS_FOREIGN_COLS, id 0 12, columns 7, indexes 1, appr.rows 1
 COLUMNS: ID: DATA_VARCHAR DATA_ENGLISH len 0;
          POS: DATA_INT len 4;
          FOR_COL_NAME: DATA_VARCHAR DATA_ENGLISH len 0;
          REF_COL_NAME: DATA_VARCHAR DATA_ENGLISH len 0;
          DB_ROW_ID: DATA_SYS prtype 256 len 6;
          DB_TRX_ID: DATA_SYS prtype 257 len 6;
 INDEX: name ID_IND, id 0 14, fields 2/6, uniq 2, type 3
  root page 49, appr.key vals 1, leaf pages 1, size pages 1
  FIELDS:  ID POS DB_TRX_ID DB_ROLL_PTR FOR_COL_NAME REF_COL_NAME
--------------------------------------
TABLE: name test/child, id 0 14, columns 10, indexes 2, appr.rows 201
 COLUMNS: par_id: DATA_INT DATA_BINARY_TYPE DATA_NOT_NULL len 4;
          child_id: DATA_INT DATA_BINARY_TYPE DATA_NOT_NULL len 4;
          name: DATA_VARCHAR prtype 524303 len 40;
          birth: DATA_INT DATA_BINARY_TYPE len 3;
          weight: DATA_FIXBINARY DATA_BINARY_TYPE len 5;
          misc_info: DATA_VARCHAR prtype 524303 len 255;
          last_update: DATA_INT DATA_UNSIGNED DATA_BINARY_TYPE DATA_NOT_NULL len 4;
          DB_ROW_ID: DATA_SYS prtype 256 len 6;
          DB_TRX_ID: DATA_SYS prtype 257 len 6;
 INDEX: name PRIMARY, id 0 17, fields 2/9, uniq 2, type 3
  root page 52, appr.key vals 201, leaf pages 5, size pages 6
  FIELDS:  par_id child_id DB_TRX_ID DB_ROLL_PTR name birth weight misc_info last_update
 INDEX: name name, id 0 18, fields 1/3, uniq 3, type 0
  root page 53, appr.key vals 210, leaf pages 1, size pages 1
  FIELDS:  name par_id child_id
 FOREIGN KEY CONSTRAINT test/child_ibfk_1: test/child ( par_id )
           REFERENCES test/parent ( par_id )
--------------------------------------
TABLE: name test/innodb_table_monitor, id 0 15, columns 4, indexes 1, appr.rows 0
 COLUMNS: i: DATA_INT DATA_BINARY_TYPE len 4;
          DB_ROW_ID: DATA_SYS prtype 256 len 6;
          DB_TRX_ID: DATA_SYS prtype 257 len 6;
 INDEX: name GEN_CLUST_INDEX, id 0 19, fields 0/4, uniq 1, type 1
  root page 193, appr.key vals 0, leaf pages 1, size pages 1
  FIELDS:  DB_ROW_ID DB_TRX_ID DB_ROLL_PTR i
--------------------------------------
TABLE: name test/parent, id 0 13, columns 6, indexes 2, appr.rows 299
 COLUMNS: par_id: DATA_INT DATA_BINARY_TYPE DATA_NOT_NULL len 4;
          fname: DATA_CHAR prtype 524542 len 20;
          lname: DATA_CHAR prtype 524542 len 20;
          DB_ROW_ID: DATA_SYS prtype 256 len 6;
          DB_TRX_ID: DATA_SYS prtype 257 len 6;
 INDEX: name PRIMARY, id 0 15, fields 1/5, uniq 1, type 3
  root page 50, appr.key vals 299, leaf pages 2, size pages 3
  FIELDS:  par_id DB_TRX_ID DB_ROLL_PTR fname lname
 INDEX: name lname, id 0 16, fields 2/3, uniq 2, type 2
  root page 51, appr.key vals 300, leaf pages 1, size pages 1
  FIELDS:  lname fname par_id
 FOREIGN KEY CONSTRAINT test/child_ibfk_1: test/child ( par_id )
           REFERENCES test/parent ( par_id )
---------------------------------
END OF INNODB TABLE MONITOR OUTPUT
=================================
```

テーブルモニターの出力には、テーブルごとに、そのテーブルに関する一般的な情報と、そのカラム、インデックス、および外部キーに関する固有の情報を表示するセクションが含まれます。

テーブルごとの一般的な情報には、テーブル名 (内部テーブルを除き、db_name/tbl_name の形式)、その ID、カラムとインデックスの数、および概略の行数が含まれます。

テーブルセクションの COLUMNS の部分には、テーブル内の各カラムが一覧表示されます。カラムごとの情報には、その名前とデータ型の特性が示されます。InnoDB によって、DB_ROW_ID (行 ID)、DB_TRX_ID (トランザクション ID)、DB_ROLL_PTR (ロールバック/Undo データへのポインタ) などのいくつかの内部カラムが追加されます。

- DATA_xxx: これらのシンボルはデータ型を示します。特定のカラムについて、複数の DATA_xxx シンボルが存在する可能性があります。
- prtype: そのカラムの「正確な」型。このフィールドには、そのカラムのデータ型、文字セットコード、NULL 可能性、符号の有無、それがバイナリ文字列であるかどうかなどの情報が含まれます。このフィールドについては、innobase/include/data0type.h ソースファイルで説明されています。
- len: カラムの長さ (バイト単位)。

テーブルセクションの各 INDEX の部分は、1 つのテーブルインデックスの名前と特性を提供します。

- name: インデックス名。この名前が PRIMARY である場合、そのインデックスは主キーです。この名前が GEN_CLUST_INDEX である場合、そのインデックスは、テーブル定義に主キーや NULL 以外の一意のインデックスが含まれていないときに自動的に作成されるクラスタ化されたインデックスです。セクション 14.2.13.2「クラスタインデックスとセカンダリインデックス」を参照してください。
- id: インデックス ID。
- fields: インデックス内のフィールドの数。m/n という形式の値です。
  - m はユーザー定義のカラムの数です。つまり、CREATE TABLE ステートメントのインデックス定義に表示されるカラム数です。
  - n は、内部的に追加されたものを含むインデックスカラムの総数です。クラスタ化されたインデックスの場合、この総数には、テーブル定義内のほかのカラムに加えて、内部的に追加されたすべてのカラムが含まれます。セカンダリインデックスの場合、この総数には、セカンダリインデックスには含まれていない主キーのカラムが含まれます。
- uniq: インデックス値を一意に特定するために十分な先頭フィールドの数。
- type: インデックスタイプ。これはビットフィールドです。たとえば、1 はクラスタ化されたインデックスを示し、2 は一意のインデックスを示すため、クラスタ化されたインデックス (常に一意の値を含みます) の type 値は 3 になります。type 値が 0 であるインデックスは、クラスタ化されたインデックスでも一意のインデックスでもありません。これらのフラグ値は、innobase/include/dict0mem.h ソースファイルで定義されています。
- root page: インデックスのルートページ番号。
- appr. key vals: 概略のインデックスカーディナリティー。
- leaf pages: インデックス内のリーフページの概数。
- size pages: インデックス内の概略のページの総数。
- FIELDS: インデックス内のフィールドの名前。自動的に生成されたクラスタ化されたインデックスの場合、このフィールドリストは、内部の DB_ROW_ID (行 ID) フィールドで始まります。クラスタ化されたインデックスには、主キーを構成するフィールドに続けて、常に DB_TRX_ID と DB_ROLL_PTR が内部的に追加されます。セカンダリインデックスの場合、最後の数フィールドは、セカンダリインデックスには含まれていない主キーのフィールドです。

テーブルセクションの最後には、そのテーブルに適用される FOREIGN KEY 定義が一覧表示されます。この情報は、そのテーブルが、参照するテーブルまたは参照されるテーブルのどちらであっても表示されます。

14.16 InnoDB のバックアップとリカバリ

安全なデータベース管理の鍵は、定期的なバックアップを作成することです。データボリューム、MySQL サーバーの数、およびデータベースワークロードに応じて、次の手法を単独で、または組み合わせて使用できます。

すなわち、MySQL Enterprise Backup を使用したホットバックアップ、MySQL サーバーのシャットダウン中にファイルをコピーすることによるコールドバックアップ、迅速な操作 (特にリストア) のための物理バックアップ、小さなデータボリュームのため、またはスキーマオブジェクトの構造を記録するための mysqldump を使用した論理バックアップです。

ホットバックアップ

MySQL Enterprise Backup コンポーネントの一部である mysqlbackup コマンドを使用すると、実行中の MySQL インスタンス (InnoDB および MyISAM テーブルを含む) を、データベースの整合性のあるスナップショットを生成しながら、操作の中断を最小限に抑えてバックアップできます。mysqlbackup が InnoDB テーブルをコピーしている間、InnoDB テーブルと MyISAM テーブルの両方に対する読み取りと書き込みは続行できます。MyISAM テーブルのコピー中は、これらのテーブルに対する (書き込みではなく) 読み取りが許可されます。MySQL Enterprise Backup はまた、圧縮バックアップファイルを作成したり、テーブルやデータベースのサブセットをバックアップしたりすることもできます。MySQL のバイナリログと組み合わせると、ユーザーはポイントインタイムリカバリを実行できます。MySQL Enterprise Backup は、MySQL Enterprise サブスクリプションの一部です。詳細は、セクション25.2「MySQL Enterprise Backup」を参照してください。

コールドバックアップ

MySQL サーバーをシャットダウンできる場合は、InnoDB がそのテーブルを管理するために使用するすべてのファイルで構成されるバイナリバックアップを作成できます。次の手順を使用します。

1. MySQL サーバーの低速シャットダウンを実行し、そのサーバーがエラーなく停止したことを確認します。
2. すべての InnoDB データファイル (ibdata ファイルおよび .ibd ファイル) を安全な場所にコピーします。
3. InnoDB テーブルのすべての .frm ファイルを安全な場所にコピーします。
4. すべての InnoDB ログファイル (ib_logfile ファイル) を安全な場所にコピーします。
5. 1 つまたは複数の my.cnf 構成ファイルを安全な場所にコピーします。

代替バックアップの種類

今説明したバイナリバックアップの作成に加えて、mysqldump を使用してテーブルのダンプを定期的に作成します。バイナリファイルは、気付かないうちに破損することがあります。ダンプされたテーブルは人間が読むことのできるテキストファイルに格納されるため、テーブルの破損を見つけることが容易になります。また、形式が単純であるため、重大なデータ破損につながる可能性も少なくなります。mysqldump には、ほかのクライアントをロックすることなく、整合性のあるスナップショットを作成するための --single-transaction オプションも用意されています。セクション7.3.1「バックアップポリシーの確立」を参照してください。

レプリケーションは InnoDB テーブルと連携して動作するため、MySQL のレプリケーション機能を使用して、データベースのコピーを高可用性が必要なデータベースサイトに保持できます。

リカバリの実行

InnoDB データベースをバイナリバックアップが作成された時点から現時点にリカバリするには、バックアップを作成する前であっても、バイナリロギングをオンにして MySQL サーバーを実行する必要があります。バックアップをリストアしたあとにポイントインタイムリカバリを実現するには、バックアップが作成されたあとに発生した変更をバイナリログから適用できます。セクション7.5「バイナリログを使用したポイントインタイム (増分) リカバリ」を参照してください。

MySQL サーバーのクラッシュからリカバリするための唯一の要件は、そのサーバーの再起動です。InnoDB はログを自動的にチェックし、データベースの現時点へのロールフォワードを実行します。InnoDB は、クラッシュの時点で存在していたコミットされていないトランザクションを自動的にロールバックします。リカバリ中に、mysqld は次のような出力を表示します。

```
InnoDB: Database was not shut down normally.
InnoDB: Starting recovery from log files...
InnoDB: Starting log scan based on checkpoint at
InnoDB: log sequence number 0 13674004
InnoDB: Doing recovery: scanned up to log sequence number 0 13739520
InnoDB: Doing recovery: scanned up to log sequence number 0 13805056
InnoDB: Doing recovery: scanned up to log sequence number 0 13870592
InnoDB: Doing recovery: scanned up to log sequence number 0 13936128
...
```

```
InnoDB: Doing recovery: scanned up to log sequence number 0 20555264
InnoDB: Doing recovery: scanned up to log sequence number 0 20620800
InnoDB: Doing recovery: scanned up to log sequence number 0 20664692
InnoDB: 1 uncommitted transaction(s) which must be rolled back
InnoDB: Starting rollback of uncommitted transactions
InnoDB: Rolling back trx no 16745
InnoDB: Rolling back of trx no 16745 completed
InnoDB: Rollback of uncommitted transactions completed
InnoDB: Starting an apply batch of log records to the database...
InnoDB: Apply batch completed
InnoDB: Started
mysqld: ready for connections
```

データベースが破損するか、またはディスク障害が発生した場合は、バックアップを使用してリカバリを実行する必要があります。破損の場合は、まず、破損していないバックアップを見つけます。ベースバックアップをリストアしたあと、mysqlbinlog および mysql を使用してバイナリログファイルからポイントインタイムリカバリを実行することにより、バックアップが作成されたあとに発生した変更をリストアします。

データベースの破損の程度によっては、1 つまたは複数の破損したテーブルをダンプして削除し、再作成するだけで十分である場合があります。テーブルが破損しているかどうかは、CHECK TABLE SQL ステートメントを使用してチェックできます。ただし、当然ながら、CHECK TABLE が可能性のあるすべての種類の破損を検出することはできません。テーブルスペースモニターを使用すると、テーブルスペースファイル内部のファイル領域管理の整合性をチェックできます。

場合によっては、見た目はデータベースページの破損だが、実際にはオペレーティングシステムによる独自のファイルキャッシュの破損であり、ディスク上のデータは正常であることがあります。まず、コンピュータを再起動してみることが最善です。それにより、データベースページの破損に見えたエラーが解消される可能性があります。InnoDB の一貫性の問題のために MySQL を引き続き起動できない場合は、セクション14.19.2「InnoDB のリカバリの強制的な実行」を参照して、データをダンプできる診断モードでインスタンスを起動する手順を調べてください。

14.16.1 InnoDB のリカバリプロセス

InnoDB のクラッシュリカバリは、次のいくつかのステップで構成されます。

- Redo ログの適用: Redo ログの適用は最初のステップであり、初期化中の、まだどの接続も受け入れていないときに実行されます。シャットダウンまたはクラッシュの時点で、バッファープールからテーブルスペース (ibdata* および *.ibd ファイル) にすべての変更がフラッシュされた場合は、Redo ログの適用をスキップできます。起動時に Redo ログファイルが見つからない場合、InnoDB は Redo ログの適用をスキップします。

  ある程度のデータ損失が許容可能な場合でも、リカバリプロセスを高速化するために Redo ログを削除することはお勧めできません。Redo ログの削除は、innodb_fast_shutdown が 0 または 1 に設定された状態でクリーンシャットダウンが実行されたあとのオプションにすぎないと考えてください。

- 未完了のトランザクションのロールバック: クラッシュまたは高速シャットダウンの時点でアクティブであったすべてのトランザクションが対象です。未完了のトランザクションをロールバックするためにかかる時間は、サーバーの負荷に応じて、そのトランザクションが中断される前にアクティブであった期間の 3 または 4 倍になる場合があります。

  ロールバックされている最中のトランザクションを取り消すことはできません。極端なケースとして、トランザクションのロールバックに膨大な時間がかかると予測される場合は、innodb_force_recovery の設定を 3 以上にして InnoDB を起動した方が速いことがあります。詳細は、セクション14.19.2「InnoDB のリカバリの強制的な実行」を参照してください。

- 挿入バッファーのマージ: インデックスページがバッファープールに読み取られたときに、挿入バッファー (システムテーブルスペースの一部) からセカンダリインデックスのリーフページに変更を適用します。

- パージ: どのアクティブなトランザクションにも表示されなくなった、削除のマークが付いたレコードを削除します。

Redo ログの適用に続く各ステップは (書き込みのロギングを除き) Redo ログには依存しないため、通常の処理では並列に実行されます。これらのうち、クラッシュリカバリに固有なのは未完了のトランザクションのロールバックだけです。挿入バッファーのマージとパージは、通常の処理中に実行されます。

Redo ログの適用のあと、InnoDB は、ダウンタイムを短縮するために接続をできるだけ早く受け入れようとします。クラッシュリカバリの一部として、InnoDB は、サーバーのクラッシュ時にコミットされていなかったか、または XA PREPARE 状態にあったすべてのトランザクションをロールバックします。このロールバックは、新しい接続からのトランザクションと並列に実行されているバックグラウンドスレッドによって実行されます。新し

い接続では、ロールバック操作が完了するまで、リカバリされるトランザクションとのロック競合が発生する可能性があります。

ほとんどの状況では、負荷の高いアクティビティーの最中に MySQL サーバーが予期せず強制終了された場合でも、リカバリプロセスが自動的に実行されるため、DBA からのアクションは必要ありません。ハードウェア障害や重大なシステムエラーのために InnoDB データが破損した場合は、MySQL が起動を拒否する可能性があります。その場合は、このような問題をトラブルシューティングする手順について、セクション14.19.2「InnoDB のリカバリの強制的な実行」を参照してください。

バイナリログおよび InnoDB のクラッシュリカバリについては、セクション5.2.4「バイナリログ」を参照してください。

14.17 InnoDB と MySQL レプリケーション

MySQL レプリケーションは、MyISAM テーブルに対して機能するのと同じように、InnoDB テーブルに対して機能します。また、スレーブ上のストレージエンジンがマスター上の元のストレージエンジンと同じではない状況でレプリケーションを使用することもできます。たとえば、マスター上の InnoDB テーブルへの変更をスレーブ上の MyISAM テーブルにレプリケートできます。

マスターに対して新しいスレーブを設定するには、InnoDB テーブルスペースとログファイルに加えて InnoDB テーブルの .frm ファイルのコピーを作成し、それらのコピーをスレーブに移動します。innodb_file_per_table オプションが有効になっている場合は、.ibd ファイルもコピーします。これを行うための正しい手順については、セクション14.16「InnoDB のバックアップとリカバリ」を参照してください。

マスターまたは既存のスレーブを停止することなく新しいスレーブを作成するには、MySQL Enterprise Backup 製品を使用します。マスターまたは既存のスレーブをシャットダウンできる場合は、InnoDB テーブルスペースとログファイルのコールドバックアップを作成し、それを使用してスレーブを設定します。

マスター上で失敗したトランザクションがレプリケーションに影響を与えることはありません。MySQL レプリケーションは、データを変更する SQL ステートメントが MySQL によって書き込まれたバイナリログに基づいています。失敗したトランザクション (たとえば、外部キーの違反のため、またはロールバックされているため) はバイナリログに書き込まれないため、スレーブには送信されません。セクション13.3.1「START TRANSACTION、COMMIT、および ROLLBACK 構文」を参照してください。

レプリケーションと CASCADE　マスター上の InnoDB テーブルに関するカスケードアクションは、外部キー関係を共有しているテーブルがマスターとスレーブの両方で InnoDB を使用している場合にのみスレーブにレプリケートされます。これは、ステートメントベースのレプリケーションと行ベースのレプリケーションのどちらを使用している場合にも当てはまります。レプリケーションを開始したあと、次の CREATE TABLE ステートメントを使用してマスター上に 2 つのテーブルを作成したとします。

```
CREATE TABLE fc1 (
  i INT PRIMARY KEY,
  j INT
) ENGINE = InnoDB;

CREATE TABLE fc2 (
  m INT PRIMARY KEY,
  n INT,
  FOREIGN KEY ni (n) REFERENCES fc1 (i)
    ON DELETE CASCADE
) ENGINE = InnoDB;
```

このスレーブでは InnoDB のサポートが有効になっていないと仮定します。この場合、スレーブ上にテーブルは作成されますが、それらのテーブルは MyISAM ストレージエンジンを使用し、FOREIGN KEY オプションは無視されます。次に、マスター上のテーブルに何行か挿入します。

```
master> INSERT INTO fc1 VALUES (1, 1), (2, 2);
Query OK, 2 rows affected (0.09 sec)
Records: 2  Duplicates: 0  Warnings: 0

master> INSERT INTO fc2 VALUES (1, 1), (2, 2), (3, 1);
Query OK, 3 rows affected (0.19 sec)
Records: 3  Duplicates: 0  Warnings: 0
```

この時点では、次に示すように、マスターとスレーブの両方で、テーブル fc1 には 2 行が含まれ、テーブル fc2 には 3 行が含まれています。

```
master> SELECT * FROM fc1;
+---+------+
| i | j    |
```

```
+---+------+
| 1 |    1 |
| 2 |    2 |
+---+------+
2 rows in set (0.00 sec)

master> SELECT * FROM fc2;
+---+------+
| m | n    |
+---+------+
| 1 |    1 |
| 2 |    2 |
| 3 |    1 |
+---+------+
3 rows in set (0.00 sec)

slave> SELECT * FROM fc1;
+---+------+
| i | j    |
+---+------+
| 1 |    1 |
| 2 |    2 |
+---+------+
2 rows in set (0.00 sec)

slave> SELECT * FROM fc2;
+---+------+
| m | n    |
+---+------+
| 1 |    1 |
| 2 |    2 |
| 3 |    1 |
+---+------+
3 rows in set (0.00 sec)
```

ここで、マスター上で次の DELETE ステートメントを実行したとします。

```
master> DELETE FROM fc1 WHERE i=1;
Query OK, 1 row affected (0.09 sec)
```

カスケードのために、マスター上のテーブル fc2 には 1 行しか含まれなくなりました。

```
master> SELECT * FROM fc2;
+---+---+
| m | n |
+---+---+
| 2 | 2 |
+---+---+
1 row in set (0.00 sec)
```

ただし、スレーブでは fc1 に対する DELETE によって fc2 から行が削除されないため、カスケードはスレーブに伝播されません。スレーブの fc2 のコピーには引き続き、最初に挿入されたすべての行が含まれています。

```
slave> SELECT * FROM fc2;
+---+---+
| m | n |
+---+---+
| 1 | 1 |
| 3 | 1 |
| 2 | 2 |
+---+---+
3 rows in set (0.00 sec)
```

この違いは、カスケード削除が、実際には InnoDB ストレージエンジンによって内部的に処理されることから来ています。つまり、どの変更も口グに記録されません。

14.18 InnoDB と memcached の統合

memcached デーモンは、MySQL データベースサーバーの前面のインメモリーキャッシュレイヤーとして頻繁に使用されます。InnoDB memcached プラグインの導入により、MySQL は memcached プロトコルおよびクライアントライブラリを使用して、InnoDB テーブルに直接アクセスできるようになりました。

InnoDB memcached プラグインは、InnoDB テーブルへのデータの自動保管およびテーブルからのデータの自動取得が可能な組み込み型の memcached デーモンを提供し、MySQL Server を高速な「キー/値ストア」に変換させます。クエリーを SQL で作成する代わりに、SQL 構文解析およびクエリー最適化プランの作成のパフォーマンスオーバーヘッドを回避する、単純な get、set、および increment 操作を実行できます。また、簡便性、複雑

なクエリー、一括操作、アプリケーション互換性、および従来のデータベースソフトウェアが持つその他の強みを活かすために、SQL を使用して同じ InnoDB テーブルにアクセスすることもできます。

この「NoSQL スタイルの」インタフェースは、memcached API を使用してデータベース操作を高速化し、InnoDB がそのバッファープールメカニズムを使用してメモリーキャッシュを処理します。ADD、SET、INCR などの memcached 操作によって変更されたデータは、変更バッファリング、二重書き込みバッファー、クラッシュリカバリなどの InnoDB メカニズムを使用してディスクに格納されます。memcached の簡便性と InnoDB の信頼性および一貫性の組み合わせにより、セクション14.18.1「InnoDB と memcached の組み合わせの利点」で説明されている両方の優れた点がユーザーに提供されます。コンポーネントを互いに組み合わせる方法に関する詳細なアーキテクチャーについては、セクション14.18.2「InnoDB および memcached の統合のアーキテクチャー」を参照してください。

14.18.1 InnoDB と memcached の組み合わせの利点

このセクションでは、セクション14.18「InnoDB と memcached の統合」で導入された InnoDB テーブルへの memcached インタフェースの利点について説明します。InnoDB テーブルと memcached を組み合わせることで、いずれかを単独で使用した場合を上回る利点が得られます。

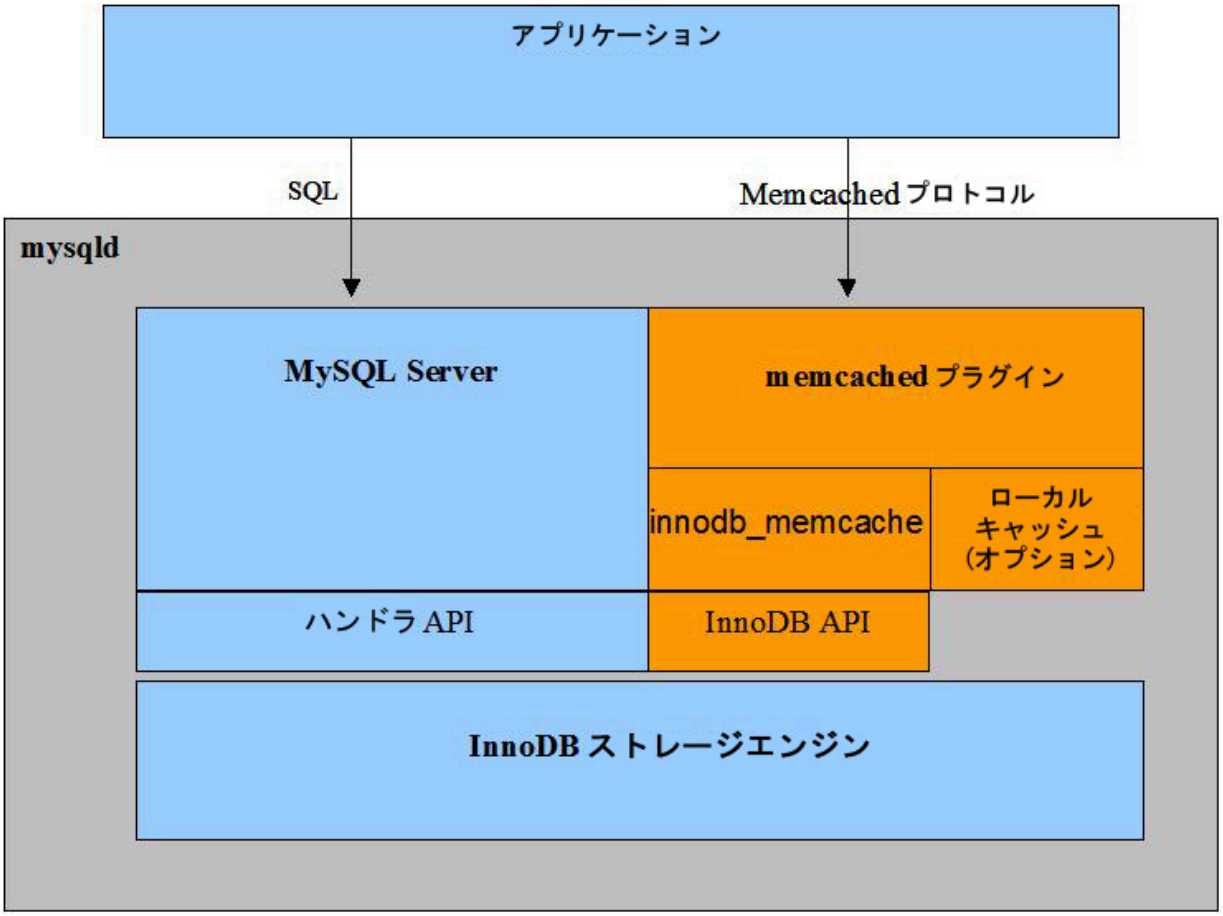
- InnoDB ストレージエンジンに直接アクセスすることによって、SQL の構文解析およびプランニングのオーバーヘッドを回避できます。
- memcached を MySQL Server と同じプロセス空間で実行することにより、リクエストを受け渡すことによるネットワークオーバーヘッドが回避されます。
- memcached プロトコルを使用して書き込まれるデータは透過的に InnoDB テーブルに書き込まれ、MySQL SQL レイヤーを経由しません。書き込みの頻度を制御することで、重要ではないデータを更新するときに本来のパフォーマンスを高めることができます。
- memcached プロトコルを介してリクエストされるデータは透過的に InnoDB テーブルから照会され、MySQL SQL レイヤーを経由しません。
- 同じデータに対する後続のリクエストは InnoDB バッファープールから提供されます。バッファープールはインメモリーキャッシュを処理します。使い慣れた InnoDB 構成オプションを使用して、大量のデータを処理する操作のパフォーマンスをチューニングできます。
- アプリケーションのタイプに応じて、非構造化データまたは構造化データを使用できます。データ用にまったく新しいテーブルを作るか、1 つ以上の既存のテーブルに NoSQL スタイルの処理をマップできます。
- InnoDB は複数カラムの値を単一の memcached 項目値に連結したり分解したりできるため、アプリケーションで必要な文字列の構文解釈および連結の量が削減されます。たとえば、文字列値 2|4|6|8 が memcached キャッシュに格納されている場合、InnoDB はその値を区切り文字に基づいて分割し、結果を 4 つの数値カラムに格納します。
- メモリーとディスク間の転送は自動的に処理されるため、アプリケーションロジックが簡素化されます。
- データは MySQL データベースに格納されることで、クラッシュ、機能停止、および破損から保護されます。
- レポート作成、分析、アドホッククエリー、一括ロード、マルチステップのトランザクション計算、論理和や論理積などの set 操作、および SQL の表現性と柔軟性に適したその他の操作のために、引き続き SQL を介して基礎テーブルにアクセスできます。
- この機能を MySQL レプリケーションと組み合わせてマスターサーバー上で使用すると、NoSQL データの高可用性を保証できます。
- memcached と MySQL を統合することにより、手間をかけずにインメモリーデータを永続的なものにでき、これを非常に多くの種類のデータに使用できます。データがすぐに消去される心配がなく、より多くの add、incr、および類似の書き込み操作をアプリケーションに配置できます。キャッシュされたデータへの更新を失わずに memcached サーバーを停止および起動できます。予期しない機能停止を防ぐために、InnoDB のクラッシュリカバリ手順、レプリケーション手順、およびバックアップ手順を利用できます。
- InnoDB が高速な主キー検索を実行する方法では、通常 memcached の単一項目クエリーに適合します。memcached プラグインによって使用される、低レベルの直接的なデータベースアクセスパスは、キー/値の検索では同等の SQL クエリーよりもはるかに効率的です。
- 複雑なデータ構造、バイナリファイル、またはコードブロックも格納可能な文字列に変換できる memcached のシリアライズ機能によって、このようなオブジェクトをデータベースに格納する簡単な方法が提供されます。

- 基礎データに SQL 経由でアクセスできるため、memcached データについてレポートを生成したり、複数のキーで検索または更新したり、AVG() や MAX() などの関数を呼び出したりできます。これらすべての操作は、スタンドアロンの memcached ではコストがかかったり複雑になったりします。
- 起動時に memcached にデータを手動でロードする必要はありません。アプリケーションによって特定のキーがリクエストされると、値はデータベースから自動的に取得され、InnoDB バッファープールを使用してメモリーにキャッシュされます。
- memcached は CPU の消費が比較的少なく、メモリーフットプリントを管理しやすいため、同じシステム上で MySQL インスタンスとともに快適に実行できます。
- 通常の InnoDB テーブルで使用されるメカニズムによってデータ完全性が強制されるため、古くなった memcached データや、キーが見つからない場合にデータベースにクエリーを実行するフォールバックロジックについて懸念する必要はありません。

14.18.2 InnoDB および memcached の統合のアーキテクチャー

このセクションでは memcached デーモンが MySQL Server にどのように統合されるかについて説明します。これは、このアプローチと NoSQL コンポーネントまたはインタフェースを MySQL バックエンドに結合するほかの技法とを比べてその優劣を理解するのに役立ちます。

memcached は MySQL Server と統合すると、MySQL プラグインデーモンとして実装され、InnoDB ストレージエンジンに直接アクセスして SQL レイヤーをバイパスします。



現在のリリースで提供される機能:

- mysqld のデーモンプラグインとしての memcached: mysqld および memcached は両方とも同じプロセス空間で実行し、非常に短い待機時間でデータにアクセスします。
- SQL パーサー、オプティマイザ、さらにハンドラ API レイヤーもバイパスして、InnoDB テーブルに直接アクセスします。
- テキストベースプロトコルとバイナリプロトコルの両方からなる標準の memcached プロトコル。InnoDB + memcached の組み合わせは、memcapable コマンドの 55 個すべての互換性テストに合格しています。

- 複数カラムのサポート: 複数のカラムをキー/値ストアの「値」部分にマップでき、カラム値はユーザー指定の区切り文字によって区切られます。
- デフォルトでは、memcached プロトコルを使用して、InnoDB からデータを直接読み取ったり書き込んだりし、MySQL で InnoDB バッファープールを介してインメモリーキャッシュを管理します。デフォルト設定は、データベースアプリケーションに対する予想外の動作を最小限に抑えた信頼性の高い組み合わせになっています。たとえばデフォルト設定では、データベース側でデータがコミットされなかったり、memcached get リクエストに対して古くなったデータが返されたりすることがないようになっています。
- 上級ユーザーは、従来の memcached サーバーとしてシステムを構成し、すべてのデータを memcached デフォルトエンジン (メモリー) にのみキャッシュするか、「memcached デフォルトエンジン」 (メモリーキャッシュ) と InnoDB memcached エンジン (バックエンド永続ストレージとしての InnoDB) の組み合わせを使用できます。
- innodb_api_bk_commit_interval、daemon_memcached_r_batch_size、および daemon_memcached_w_batch_size 構成オプションを使用して、InnoDB と memcached の操作間のデータがやり取りされる頻度を制御できます。最大限の信頼性を得るには、両方のバッチサイズオプションのデフォルト値を 1 にします。
- MySQL 構成変数 daemon_memcached_option から memcached 構成オプションを指定できます。たとえば、memcached が待機するポートを変更したり、同時接続の最大数を削減したり、鍵と値のペアの最大メモリーサイズを変更したり、エラーログに関するメッセージのデバッグを有効にしたりします。
- 構成オプション innodb_api_trx_level を使用すると、ユーザーは memcached インタフェースによって処理されるクエリーのトランザクション分離レベルを制御できます。memcached にはトランザクションの概念がありませんが、このプロパティーを使用すると、memcached がインタフェースするテーブルと同じテーブル上で DML ステートメントを発行した場合、SQL ステートメントによって発生した変更が memcached で表示されるまでの速さを制御できます。デフォルトでは、これは READ UNCOMMITTED に設定されています。
- 別の構成オプションは innodb_api_enable_mdl です。「MDL」は「メタデータロック」を表します。これは基本的に MySQL レベルでテーブルをロックするため、マップされたテーブルを、SQL インタフェース経由で DDL によってドロップしたり変更したりできません。ロックがなければテーブルを MySQL レイヤーからドロップできますが、memcached またはほかのユーザーがテーブルの使用を停止するまで InnoDB ストレージ内に保持されます。

memcached のスタンドアロンでの使用と InnoDB との併用の違い

セクション16.6「MySQL と memcached の併用」に記載されているように、MySQL ユーザーは、memcached を MySQL とともに使用することをすでに熟知している場合があります。このセクションでは、セクション内の情報の類似点および相違点と、MySQL に組み込まれている memcached の InnoDB 統合機能を使用する場合について説明します。各項目の最初のリンクから、従来の memcached サーバーに関する関連情報にアクセスできます。

- インストール: memcached ライブラリは MySQL Server に付属しているため、インストールおよびセットアップは簡単明瞭です。使用する memcached 用のテーブルをセットアップするための SQL スクリプトを実行し、1 回かぎりの install plugin ステートメントを発行して memcached を使用可能にし、MySQL 構成ファイルまたは起動オプションに、別のポートを使用するなどの必要な memcached オプションを追加します。通常の memcached 配布をインストールして、memcp、memcat、memcapable などの追加のユーティリティーを取得する場合もあります。
- 配備: 通常は、能力の低い多数の memcached サーバーを実行します。InnoDB + memcached の組み合わせでは、データベースと memcached サーバーの比率が 1:1 になるため、通常の配備では、MySQL をすでに実行している中程度あるいは高い能力を持つ少数のサーバーマシンが含まれます。このサーバー構成の利点は、多数のサーバーにわたって未使用メモリーを活用したり、検索を分散させたりすることよりも、個々のデータベースサーバーそれぞれの効率を高めることの方が上回ります。デフォルト構成では、memcached にはメモリーがほとんど使用されず、インメモリー検索は InnoDB バッファープールから提供され、このバッファープールは、最近使用されたデータと頻繁に使用されるデータを自動的にキャッシュします。従来の MySQL Server インスタンスと同じように、innodb_buffer_pool_size 構成オプションの値を (OS レベルでのページングを発生させない) 実用的なできるだけ高い値に維持し、これによりできるだけ多くのワークロードがメモリー内で実行されるようにします。
- 期限切れ: デフォルトでは (キャッシュポリシー innodb_only を使用)、InnoDB テーブルから最新データが常に返されるため、期限切れオプションは事実上影響ありません。キャッシュポリシーを caching または cache-only に変更した場合、期限切れオプションは通常どおり機能しますが、リクエストされたデータが、メモリーキャッシュ内で期限切れになる前に基礎テーブル内で更新された場合、データが古くなっている可能性があります。

- ネームスペース: memcached は 1 つの巨大なディレクトリに似ており、ここではファイルが互いに矛盾しないように、プリフィクスとサフィクスを使用した複雑な名前を指定する場合があります。InnoDB と memcached による統合サーバーでは、キーに同じ命名規則を使用できますが、追加の規則が 1 つあります。@@table_id.key.table_id という形式のキー名は、innodb_memcache.containers テーブルからのマッピングデータを使用して、特定のテーブルを参照するようにデコードされます。key は指定されたテーブル内で参照されるか、このテーブルに書き込まれます。

  @@ 表記は、get、add、および set 関数に対する個別の呼び出しにのみ機能し、incr や delete などのほかの関数には機能しません。セッション内の後続の memcached 操作に対してデフォルトテーブルを指定するには、@@ 表記とテーブル ID を使用し、キーの部分は指定せずに get リクエストを実行します。例:

  ```
  get @@table_x
  ```

  後続の get、set、incr、delete およびその他の操作は、innodb_memcache.containers.name カラム内で table_x によって指定されたテーブルを使用します。

- ハッシュおよび配布: キャッシュポリシー innodb_only を使用したデフォルト構成は、レプリケーションスレーブサーバーのセットなど、サーバー上ですべてのデータが使用できる従来の配備構成に適しています。

  データをシャード構成のように物理的に分割する場合、InnoDB および memcached を組み合わせたサーバーを実行するいくつかのマシンでデータを分割でき、従来の memcached ハッシュメカニズムを使用するとリクエストを特定のマシンに送信できます。MySQL 側では、通常、memcached への add リクエストによってすべてのデータを挿入するため、適切なサーバー上のデータベース内に適切な値が保管されます。

  これらのタイプの配備のベストプラクティスが、引き続き体系化されています。

- メモリーの使用: デフォルトでは (キャッシュポリシー innodb_only を使用)、memcached プロトコルは InnoDB テーブル間で情報をやり取りし、memcached のメモリー使用量を増加および減少させるのではなく、固定サイズの InnoDB バッファープールでインメモリー検索を処理します。相対的には、memcached 側ではメモリーをほとんど使用しません。

  キャッシュポリシーを caching または cache-only に切り換えた場合、memcached メモリー使用量の通常のルールが適用されます。memcached データ値のメモリーは、「スラブ」を利用して割り当てられます。memcached で使用されるスラブサイズおよび最大メモリーを制御できます。

  いずれの場合も、たとえば telnet セッションを介した標準プロトコル経由でアクセスされる、使い慣れた統計システムを使用して、統合された memcached デーモンをモニターおよびトラブルシューティングできます。統合型デーモンとともに追加のユーティリティーが含まれているわけではないため、memcached-tool スクリプトを使用するには、完全な memcached 配布をインストールします。

- スレッドの使用: MySQL スレッドおよび memcached スレッドは同一サーバー上に共存する必要があるため、オペレーティングシステムでスレッドに課されるすべての制限は、この合計の数値に適用されます。

- ログの使用: memcached デーモンは MySQL Server とともに実行し、stderr に書き込むため、ロギングのための -v、-vv、および -vvv オプションによって、これらの出力が MySQL エラーログに書き込まれます。

- memcached 操作: get、set、add、delete などのよく使用される操作を使用できます。シリアライズ (複雑なデータ構造を表す正確な文字列書式) は言語インタフェースによって異なります。

- MySQL フロントエンドとしての memcached の使用: InnoDB を memcached と統合させるということは、まさにこのことです。これらのコンポーネントを一緒に配置することで、アプリケーションのパフォーマンスが改善されます。InnoDB によってメモリーとディスク間のデータ転送が処理されるため、アプリケーションのロジックが簡素化されます。

- ユーティリティー: MySQL Server は libmemcached ライブラリを含んでいますが、追加のコマンド行ユーティリティーを含んでいません。memcp、memcat、memcapable コマンドなどのコマンドを取得するには、完全な memcached 配布をインストールします。memrm および memflush がキャッシュから項目を削除すると、項目はベースとなる InnoDB テーブルからも削除されます。

- プログラミングインタフェース: いつも使用している言語と同じ言語である、C および C++、Java、Perl、Python、PHP、および Ruby を使用して、InnoDB および memcached の組み合わせから MySQL Server にアクセスできます。ほかの memcached サーバーと同じように、サーバーホスト名およびポートを指定します。デフォルトでは、統合化された memcached サーバーは、通常と同じポートである 11211 をリッスンします。テキストプロトコルとバイナリプロトコルの両方を使用できます。memcached 関数の動作を実行時にカスタマイズできます。シリアライズ (複雑なデータ構造を表す正確な文字列書式) は言語インタフェースによって異なります。

- よくある質問: MySQL では、複数のリリースで、広範囲にわたる memcached FAQ が用意されています。MySQL 5.6 でも回答はおおむね変わっていませんが、InnoDB テーブルを memcached データのストレージメディアとして使用することで、この組み合わせを読み取り専用キャッシュとしてではなく、書き込み処理が多いアプリケーションに使用できるようになっています。

この機能のしくみについて詳しくは、セクション14.18.7「InnoDB memcached プラグインの内部構造」を参照してください。

14.18.3 InnoDB Memcached プラグインの概要

このセクションでは、MySQL Server 上で InnoDB と memcached の統合をアクティブ化するステップについて説明します。memcached デーモンは、ネットワークトラフィックを回避して待機時間を最小限に抑えるよう MySQL Server と強固に統合されているため、この機能を使用する個々の MySQL インスタンス上でこのプロセスを実行します。

> **注記**
>
> memcached インタフェースをすべてのデータ用にセットアップする前に、セクション14.18.4「InnoDB memcached プラグインのセキュリティーに関する考慮事項」を調べて、不正アクセスを防ぐために必要なセキュリティー手順を理解します。

14.18.3.1 InnoDB memcached プラグインの前提条件

プラグインと内部テーブルをセットアップする前に、必要な前提条件ソフトウェアをサーバーが持っていることを確認します。

プラットフォームサポート

memcached デーモンプラグインは、現在、Linux、Solaris、および OS X プラットフォームでのみサポートされています。

ソフトウェアの前提条件

memcached で必要なため、libevent がインストールされている必要があります。このライブラリを取得する方法は、MySQL インストールプログラムを使用するか、ソースから構築するかによって異なります。

- MySQL インストールプログラムを使用してインストールする場合、libevent ライブラリはインストールに含まれません。オペレーティングシステムのインストール方法を使用して、libevent 1.4.3 以降をインストールします。たとえば、オペレーティングシステムによっては、apt-get、yum、または port install コマンドを使用する場合もあります。たとえば、Ubuntu Linux で次のようにします。

```
sudo apt-get install libevent-dev
```

- ソースコードリリースからインストールする場合、libevent 1.4.3 がパッケージにバンドルされており、MySQL ソースコードディレクトリの最上位レベルに配置されています。バンドルされたバージョンの libevent を使用する場合、アクションは不要です。ローカルシステムバージョンの libevent を使用する場合、-DWITH_LIBEVENT ビルドオプションを system または yes に設定して MySQL をビルドする必要があります。

MySQL をソースからビルドするときの前提条件

MySQL Server をビルドする場合、-DWITH_INNODB_MEMCACHED=ON を指定してビルドする必要があります。このビルドオプションでは、MySQL プラグインディレクトリ (plugin_dir) 内に、InnoDB memcached を実行するために必要な 2 つの共有ライブラリを生成します。

- libmemcached.so: MySQL に対する memcached デーモンプラグイン。
- innodb_engine.so: memcached に対する InnoDB API プラグイン。

14.18.3.2 InnoDB memcached プラグインのインストールおよび構成

必須テーブルのセットアップ

InnoDB テーブルと対話できるように memcached プラグインを構成するには、innodb_memcached_config.sql 構成スクリプトを実行して、バックグラウンドで使用される必要なテーブルをインストールします。

```
mysql> source MYSQL_HOME/share/innodb_memcached_config.sql
```

これは 1 回かぎりの操作です。あとで memcached サポートを無効化してから再度有効化した場合、テーブルはそのまま残ります。これらのテーブルのレイアウトと目的については、セクション14.18.7「InnoDB memcached プラグインの内部構造」を参照してください。

デーモンプラグインのインストール

デーモンプラグインをアクティブ化するには、ほかの MySQL プラグインをインストールする場合と同様に、install plugin ステートメントを使用します。

```
mysql> install plugin daemon_memcached soname "libmemcached.so";
```

プラグインがこのようにインストールされると、プラグインは MySQL Server がブートまたは再起動されるたびに自動的にアクティブ化されます。

デーモンプラグインの無効化

プラグイン構成に大きな変更を行う場合、プラグインをオフにすることが必要な場合があります。これを実行するには、次のステートメントを発行します。

```
mysql> uninstall plugin daemon_memcached;
```

これをふたたび有効にするには、前述の install plugin ステートメントを発行します。プラグインがこの方法で再起動される場合、以前のすべての構成設定、内部テーブル、およびデータが保持されます。

プラグインの有効化および無効化についてのその他の情報は、セクション5.1.8.1「プラグインのインストールおよびアンインストール」を参照してください。

memcached 構成オプションの指定

memcached に固有の構成パラメータがある場合、それらを mysqld コマンド行に指定するか、MySQL 構成ファイルに入力し、daemon_memcached_option MySQL 構成オプションに対する引数としてエンコードします。memcached 構成オプションは、MySQL Server が開始されるたびに実行されるプラグインのインストール時に有効になります。

たとえば、memcached がデフォルトポート 11211 ではなくポート 11222 をリッスンするするようにする場合、MySQL 構成オプション daemon_memcached_option に -p11222 を追加します。

```
mysqld .... --daemon_memcached_option="-p11222"
```

daemon_memcached_option 文字列に、ほかの memcached コマンド行オプションを追加できます。ほかの構成オプションは次のとおりです。

- daemon_memcached_engine_lib_name (デフォルトは innodb_engine.so)
- daemon_memcached_engine_lib_path (デフォルトは NULL、プラグインディレクトリを表す)。
- daemon_memcached_r_batch_size、読み取り操作 (get) のバッチコミットサイズ。これは memcached の読み取り操作を何回実行したあとにシステムが自動的にコミットを実行するかを指定します。デフォルトでは、これは 1 に設定されるため、memcached または SQL によってデータが更新されたかどうかに関係なく、すべての get リクエストは InnoDB テーブル内のコミット済みの最新データにアクセスできます。この値が 1 より大きい場合、読み取り操作のカウンタは get 呼び出しのたびに増分されます。flush_all 呼び出しは、読み取りおよび書き込みカウンタを両方ともリセットします。
- daemon_memcached_w_batch_size、すべての書き込み操作 (set、replace、append、prepend、incr、decr など) のバッチコミット。デフォルトでは、これは 1 に設定されるため、機能停止時にコミット済みでないデータは失われず、基礎テーブルへのすべての SQL クエリーは最新のデータにアクセスできます。この値が 1 より大きいとき、書き込み操作のカウンタは、すべての add、set、incr、decr、および delete 呼び出しのたびに 1 回増分されます。flush_all 呼び出しは、読み取りおよび書き込みカウンタを両方ともリセットします。

デフォルトでは、最初の 2 つの構成オプションを変更する必要はありません。これらのオプションによって、memcached のために別のストレージエンジン (NDB memcached エンジンなど) をロードできます。

繰り返しになりますが、これらの構成パラメータは、MySQL 構成ファイルまたは MySQL ブートコマンド行で指定します。これらは、memcached プラグインをロードしたときに有効になります。

サマリー

これで、すべてセットアップできました。memcached インタフェースを介して InnoDB テーブルと直接対話できます。この機能が正常に動作していることを検証するには、セクション14.18.3.3「InnoDB および memcached 設定の検証」を参照してください。

14.18.3.3 InnoDB および memcached 設定の検証

すべてのセットアップが完了したので、InnoDB と memcached の組み合わせについて実験できます。

Unix、Linux、または OS X コマンドシェル使用した例を示します。

```
# Point memcached-related commands at the memcached attached to the mysqld process.
export MEMCACHED_SERVERS=127.0.0.1:11211
# Store the contents of a modestly sized text file in memcached, with the data passed
# to MySQL and stored in a table. The key is the basename of the file, 'mime.types'.
memcp /etc/apache2/mime.types
# Retrieve the data we just stored, from the memory cache.
memcat mime.types
```

telnet を使用して memcached コマンドを送信し、ASCII プロトコルを介して結果を受け取る例を示します。

```
telnet 127.0.0.1 11211
set a11 10 0 9
123456789
STORED
get a11
VALUE a11 0 9
123456789
END
quit
```

すべての同じデータが MySQL に格納されたことを確認するには、MySQL Server に接続して次のコマンドを発行します。

```
mysql> select * from test.demo_test;
```

ここで MySQL Server をシャットダウンすると、統合済みの memcached サーバーも停止します。さらに memcached データにアクセスしようとすると、今度は接続エラーで失敗します。通常であれば、memcached データはこの時点で消失し、memcached の再起動時にデータをメモリーにロードするようアプリケーションロジックを記述します。ただし、MySQL と memcached の統合により、このプロセスが自動化されます。

- MySQL Server を再起動します。
- install plugin ステートメントを実行して InnoDB memcached プラグインを再起動します。
- ここで、すべての memcat コマンドまたは get 操作により、以前の memcached セッションで格納されていたキー/値のペアがふたたび戻されます。キーのリクエスト時、関連付けられた値がメモリーキャッシュにまだ存在しない場合、MySQL テーブル (デフォルトでは test.demo_test) から自動的にクエリーされます。

14.18.4 InnoDB memcached プラグインのセキュリティーに関する考慮事項

> **注意**
>
> InnoDB memcached プラグインを本番サーバーに配備する前に (MySQL インスタンスに機密情報が格納されている場合はテストサーバーも含む)、このセクションを参照してください。

memcached はデフォルトで認証メカニズムを使用せず、オプションの SASL 認証が従来の DBMS セキュリティー対策ほど強くないため、InnoDB memcached プラグインを使用する MySQL インスタンスには機密情報以外のデータのみを保持するようにし、この構成を使用するサーバーへの潜在的な侵入を防止します。このようなサーバーへの memcached アクセスはインターネットからは許可せず、ファイアウォールで保護されたイントラネット内部からのみとし、ユーザーがメンバーシップを制限できるサブネットからのアクセスが適しています。

14.18.4.1 SASL による memcached インタフェースのパスワード保護

SASL サポートにより、memcached クライアントを介した未認証アクセスから MySQL データベースを保護する機能を使用できます。このセクションでは、このオプションを使用可能にするステップを説明します。このよう

なサポートを可能にするステップは、従来の memcached サーバーで SASL を使用可能にするためのステップとほとんど同じです。

バックグラウンド情報

SASL とは「Simple Authentication and Security Layer」を意味し、接続ベースのプロトコルに対して認証サポートを追加するための規格です。memcached には 1.4.3 リリースから SASL サポートが追加されました。

SASL 認証はバイナリプロトコルでのみサポートされます。

InnoDB + memcached の組み合わせの場合、memcached データを格納するテーブルは、container システムテーブルに登録する必要があります。また、memcached クライアントはこのような登録済みのテーブルにのみアクセスできます。 memcached プラグインに登録されたテーブルに、DBA のアクセス制限を追加できますが、memcached アプリケーション経由でのアクセス可能なユーザーの制御はできません。このため、memcached に関連付けられた InnoDB テーブルにアクセス可能なユーザーを制御するための (SASL 経由の) 手段が提供されています。

次のセクションでは、SASL 対応の InnoDB memcached プラグインをビルド、有効化、およびテストする方法を示しています。

InnoDB Memcached プラグインの SASL をビルドおよび使用可能にするためのステップ

デフォルトでは、SASL 対応の InnoDB memcached は、SASL ライブラリを使用した memcached のビルドに依存しているため、リリースパッケージに含まれていません。この機能を有効にするには、MySQL ソースをダウンロードし、SASL ライブラリをダウンロードしたあとに InnoDB memcached プラグインを再ビルドします。

1. 最初に、SASL 開発ライブラリおよびユーティリティーライブラリを取得します。たとえば、Ubuntu では、次のようにしてこれらのライブラリを取得できます。

   ```
   sudo apt-get -f install libsasl2-2 sasl2-bin libsasl2-2 libsasl2-dev libsasl2-modules
   ```

2. 次に、ENABLE_MEMCACHED_SASL=1 を cmake オプションに追加することによって、SASL 機能を有する InnoDB memcached プラグイン (共有ライブラリ) をビルドします。さらに、memcached はテストに簡単に使用できる単純な平文パスワードサポートを提供しています。これを有効にするには、オプション ENABLE_MEMCACHED_SASL_PWDB=1 を設定します。

   全体として、次の 3 つのオプションを cmake に追加します。

   ```
   cmake ... -DWITH_INNODB_MEMCACHED=1
    -DENABLE_MEMCACHED_SASL=1 -DENABLE_MEMCACHED_SASL_PWDB=1
   ```

3. 3 番目のステップでは、セクション14.18.3「InnoDB Memcached プラグインの概要」で説明したように、InnoDB memcached プラグインを以前のようにインストールします。

4. 以前説明したように、memcached には SASL による単純な平文パスワードサポートが提供されており、このデモで使用します。

   a. testname という名前のユーザーと、そのパスワード testpasswd をファイル内に作成します。

      ```
      echo "testname:testpasswd:::::::" >/home/jy/memcached-sasl-db
      ```

   b. 環境変数 MEMCACHED_SASL_PWDB を設定することによって memcached にこの情報を通知します。

      ```
      export MEMCACHED_SASL_PWDB=/home/jy/memcached-sasl-db
      ```

   c. さらに、これが平文パスワードであることも memcached に通知します。

      ```
      echo "mech_list: plain" > /home/jy/work2/msasl/clients/memcached.conf
      export SASL_CONF_PATH=/home/jy/work2/msasl/clients/memcached.conf
      ```

5. 次に、サーバーをリブートし、daemon_memcached_option オプション -S を追加して SASL を有効にします。

   ```
   mysqld ... --daemon_memcached_option="-S"
   ```

6. これでセットアップは完了しました。これをテストするには、この SASL 対応 libmemcached のような SASL 対応クライアントが必要な場合もあります。

```
memcp --servers=localhost:11211 --binary  --username=testname
 --password=testpasswd myfile.txt

memcat --servers=localhost:11211 --binary --username=testname
 --password=testpasswd myfile.txt
```

適切なユーザー名またはパスワードがないと、前述の操作はエラーメッセージ memcache error AUTHENTICATION FAILURE で拒否されます。それ以外の場合、操作は成功です。また、memcached-sasl-db ファイルに設定された平文パスワードを調査して、これを検証することもできます。

memcached での SASL 認証を検査する方法はほかにもあります。ただし、上に記載されたものがもっとも簡単です。

14.18.5 InnoDB memcached インタフェース用のアプリケーションの作成

一般的に、InnoDB memcached インタフェース用のアプリケーションの作成には、MySQL または memcached API を使用する既存コードの再作成または改変が含まれます。

- 多くの memcached サーバーを低性能のマシン上で実行させる代わりに、MySQL Server と同数の memcached サーバーを用意し、十分な量のディスクストレージおよびメモリーを持つ比較的高性能のマシン上で実行させます。memcached API とともに動作する既存のコードを一部再利用する場合もありますが、サーバー構成が異なるため、一部の改変が必要になる場合があります。
- このインタフェースから格納されたデータはすべて、VARCHAR、TEXT、または BLOB カラムに入るため、数値的な操作を行うために変換する必要があります。この変換は、アプリケーション側で行うことも、クエリー内で CAST() 関数を使用して行うこともできます。
- データベースバックグラウンドから使用する場合、多くのカラムを備えた汎用の SQL テーブルを使用する場合があります。memcached コードによってアクセスするテーブルは、データ値を保持するカラムが 2 つか 3 つ、または 1 つのみの場合もあります。
- コードの重要なセクションからパフォーマンスを引き出すために、単一行のクエリー、挿入、更新、または削除を実行するアプリケーションの一部を改変することがあります。クエリー (読み取り) および DML (書き込み) の両方の操作は、memcached インタフェースを介して実行すると大幅に高速化できます。書き込み速度の向上は読み取り速度の向上より大きいため、ロギングを行なったり、Web サイトでの対話を記録したりするコードの改変に的を絞るのがよい場合もあります。

次のセクションでは、これらの側面をより詳細に見ていきます。

14.18.5.1 memcached アプリケーションに対する既存の MySQL スキーマの改変

memcached インタフェースを使用するために既存の MySQL スキーマまたはアプリケーションを改変する場合、memcached アプリケーションの次の点を検討してください。

- スペースまたは改行文字は ASCII プロトコルで区切り文字として使用されるため、memcached のキーにこれらの文字を含めることはできません。スペースを含む検索値を使用する場合は、add()、set()、get() などの呼び出しでこれらをキーとして使用する前に、スペースのない値に変換またはハッシュします。これらの文字は、理論上、バイナリプロトコルを使用するプログラム内でキーとして許可されますが、さまざまなクライアントとの互換性を確保するために、キーに使用される文字を常に制限します。
- InnoDB テーブルに短い数値の主キーカラムがある場合、整数を文字列値に変換すると、memcached の一意の検索キーとして使用できます。memcached サーバーが複数のアプリケーションで使用される場合、または複数の InnoDB テーブルとともに使用される場合、一意性を保つために名前の変更を検討してください。たとえば、数値の前にテーブル名やデータベース名とテーブル名を付けます。

  > **注記**
  >
  > MySQL 5.6.14 以降では、InnoDB memcached プラグインは、主キーとして INTEGER が定義されたマップ済み InnoDB テーブル上での挿入および読み取りをサポートします。

- memcached インタフェースからクエリーを実行したり格納したりしたデータについて、パーティション化されたテーブルは使用できません。
- memcached プロトコルは数値を文字列として渡します。ベースとなる InnoDB テーブルに数値を格納する場合、たとえば、SUM() や AVG() などの SQL 関数に使用できるカウンタを実装するには、次のようにします。

- 予想される最大数のすべての桁 (さらに該当する場合はマイナス符号、小数点またはその両方に対する追加の文字) を保持するために十分な文字がある VARCHAR カラムを使用します。
- カラム値を使用して計算を実行するすべてのクエリー内で、CAST() 関数を使用して、文字列から整数またはその他の数値型に変換します。例:

```
-- Alphabetic entries are returned as zero.
select cast(c2 as unsigned integer) from demo_test;
-- Since there could be numeric values of 0, can't disqualify them.
-- Test the string values to find the ones that are integers, and average only those.
select avg(cast(c2 as unsigned integer)) from demo_test
  where c2 between '0' and '9999999999';
-- Views let you hide the complexity of queries. The results are already converted;
-- no need to repeat conversion functions and WHERE clauses each time.
create view numbers as select c1 key, cast(c2 as unsigned integer) val
  from demo_test where c2 between '0' and '9999999999';
select sum(val) from numbers;
```

  結果セット内のアルファベットの値は、CAST() の呼び出しによって 0 に変換されます。結果セット内の行数に依存する AVG() などの関数を使用する場合、数値以外の値を取り除くための WHERE 句を含めるようにします。

- キーとして使用する InnoDB カラムが 250 バイトよりも長くなる可能性がある場合、250 バイト未満の値になるようにハッシュします。
- memcached インタフェースで既存のテーブルを使用するには、そのテーブルのためのエントリを innodb_memcache.containers テーブル内に定義します。memcached を介して中継されるリクエストでテーブルをデフォルトに指定する場合、name カラムに値 default を指定し、その後 MySQL Server を再起動して変更を有効にします。異なる種類の memcached データに対して複数のテーブルを使用している場合、ユーザーの選択した name 値を使用して innodb_memcache.containers テーブル内に複数のエントリをセットアップし、次にアプリケーション内で get @@name または set @@name の形式で memcached リクエストを発行し、memcached API を経由する後続のリクエストについて使用されるテーブルを切り換えます。

  事前定義された test.demo_test テーブル以外のテーブルを使用する例については、例14.23「InnoDB + memcached アプリケーションのためのテーブルおよびカラムマッピングの指定」を参照してください。そのようなテーブルの必要なレイアウトとカラムの意味については、セクション14.18.7「InnoDB memcached プラグインの内部構造」を参照してください。

- memcached のキー/値のペアによって複数の MySQL カラム値を使用するには、MySQL テーブルに関連付けられた innodb_memcache.containers エントリ内で、value_columns フィールドに、カンマ、セミコロン、スペース、またはパイプ文字で区切られた複数のカラム名を指定します。たとえば、col1,col2,col3 または col1|col2|col3 のようになります。

  カラム値を、パイプ文字を区切り文字として使用した単一文字列になるように連結したあとで、その文字列を memcached の add または set 呼び出しに渡します。文字列はさまざまなカラムに自動的にアンパックされます。個々の get 呼び出しは、やはりパイプ区切り文字によって区切られている複数のカラム値を含む単一文字列を返します。アプリケーション言語に応じた適切な構文を使用して、これらの値をアンパックします。

例 14.23 InnoDB + memcached アプリケーションのためのテーブルおよびカラムマッピングの指定

ここで、データ操作のために InnoDB memcached プラグインを使用する MySQL アプリケーションのためにユーザー独自のデータを使用する方法の例を示します。

最初に、一部の国別データを保持するテーブルをセットアップします。データは、人口、メートル法で示した面積、右または左のどちら側を運転するかを示す 'R' または 'L' です。

```
use test;

CREATE TABLE `multicol` (
  `country` varchar(128) NOT NULL DEFAULT '',
  `population` varchar(10) DEFAULT NULL,
  `area_sq_km` varchar(9) DEFAULT NULL,
  `drive_side` varchar(1) DEFAULT NULL,
  `c3` int(11) DEFAULT NULL,
  `c4` bigint(20) unsigned DEFAULT NULL,
  `c5` int(11) DEFAULT NULL,
  PRIMARY KEY (`country`),
) ENGINE=InnoDB DEFAULT CHARSET=latin1;
```

ここで、InnoDB memcached プラグインがこのテーブルにアクセスする方法を認識するように、このテーブルのディスクリプタを作成します。

- CONTAINERS テーブル内のサンプルエントリは、name カラムが 'aaa' で、別の識別子 'bbb' をセットアップします。使用するすべての memcached アプリケーションに対して単一のマスターテーブルを作成した場合、ID を 'default' にして、テーブルを切り換えるための @@ リクエストをスキップします。
- test.multicol テーブルを指定します。スキーマ名は 1 つのカラムに格納され、テーブル名は別のカラムに格納されます。
- キーカラムは一意の country 値です。このカラムは上記のテーブルを作成するときに主キーに指定されたため、ここでインデックス名 'PRIMARY' も指定します。
- 1 つの複合データを単一カラムに保持する代わりにデータを 3 つのテーブルカラムに分割するため、値を格納または取得するときに使用されるカラムのカンマ区切りリストを指定します。
- さらに、フラグ、期限切れ、および CAS 値については、サンプルテーブル demo.test の設定に基づく対応するカラムを指定します。これらの値は、InnoDB memcached プラグインを使用するアプリケーションでは通常はあまり重要ではありません。MySQL がデータの同期を保持するため、データが期限切れになったり古くなったりすることを心配する必要がないためです。

```
insert into innodb_memcache.containers
  (name,db_schema,db_table,key_columns,value_columns,flags,cas_column,
  expire_time_column,unique_idx_name_on_key)
values
  ('bbb','test','multicol','country','population,area_sq_km,drive_side',
  'c3','c4','c5','PRIMARY');

commit;
```

ここで、プログラムからこのテーブルにアクセスする方法を示すサンプル Python プログラムを示します。

- すべてのデータ操作は memcached インタフェースを介して実行されるため、データベース許可は不要です。知る必要があるのは、memcached デーモンがローカルシステム上でリスニングするポート番号のみです。
- 任意のいくつかの国のサンプル値をロードします。(ウィキペディアからの面積および人口の数値。)
- プログラムで multicol テーブルを使用するために、@@ 表記を使用してダミーの GET または SET リクエストを行う switch_table() 関数を呼び出します。リクエスト内の名前は bbb で、これは innodb_memcache.containers.name に格納されている値です。(実際のアプリケーションでは、さらに記述的な名前を使用します。この例では、GET @@... リクエストでテーブル名でなくテーブル識別子を指定することを示しています。
- データを挿入してクエリーを実行するためのユーティリティー関数は、ADD または SET リクエストによって MySQL に送信するために Python データ構造をパイプ区切り値に変換し、GET リクエストによって返されるパイプ区切り値をアンパックするための方法を示します。この特別な処理は、単一の memcached 値を複数の MySQL テーブルカラムにマッピングする場合にのみ必要です。

```
import sys, os
import memcache

def connect_to_memcached():
  memc = memcache.Client(['127.0.0.1:11211'], debug=0);
  print "Connected to memcached."
  return memc

def banner(message):
  print
  print "=" * len(message)
  print message
  print "=" * len(message)

country_data = [
("Canada","34820000","9984670","R"),
("USA","314242000","9826675","R"),
("Ireland","6399152","84421","L"),
("UK","62262000","243610","L"),
("Mexico","113910608","1972550","R"),
("Denmark","5543453","43094","R"),
("Norway","5002942","385252","R"),
("UAE","8264070","83600","R"),
("India","1210193422","3287263","L"),
("China","1347350000","9640821","R"),
]

def switch_table(memc,table):
  key = "@@" + table
```

```
  print "Switching default table to '" + table + "' by issuing GET for '" + key + "'."
  result = memc.get(key)

def insert_country_data(memc):
  banner("Inserting initial data via memcached interface")
  for item in country_data:
    country = item[0]
    population = item[1]
    area = item[2]
    drive_side = item[3]

    key = country
    value = "|".join([population,area,drive_side])
    print "Key = " + key
    print "Value = " + value

    if memc.add(key,value):
      print "Added new key, value pair."
    else:
      print "Updating value for existing key."
      memc.set(key,value)

def query_country_data(memc):
  banner("Retrieving data for all keys (country names)")
  for item in country_data:
    key = item[0]
    result = memc.get(key)
    print "Here is the result retrieved from the database for key " + key + ":"
    print result
    (m_population, m_area, m_drive_side) = result.split("|")
    print "Unpacked population value: " + m_population
    print "Unpacked area value      : " + m_area
    print "Unpacked drive side value: " + m_drive_side

if __name__ == '__main__':

  memc = connect_to_memcached()
  switch_table(memc,"bbb")
  insert_country_data(memc)
  query_country_data(memc)

  sys.exit(0)
```

ここに示すいくつかの SQL クエリーは、スクリプトが実行されたあとの MySQL データの状態を示し、SQL を介して同じデータに直接アクセスしたり、適切な MySQL コネクタまたは API を使用する任意の言語で記述されたアプリケーションからアクセスしたりする方法を示します。

テーブルディスクリプタ 'bbb' があるため、memcached リクエスト GET @bbb を発行すると、multicol テーブルに切り換えることができます。

```
mysql: use innodb_memcache;
Database changed

mysql: select * from containers;
+------+-----------+-----------+-------------+---------------------------------+-------+------------+-------------------+-----------------------+
| name | db_schema | db_table  | key_columns | value_columns                   | flags | cas_column | expire_time_column | unique_idx_name_on_key |
+------+-----------+-----------+-------------+---------------------------------+-------+------------+-------------------+-----------------------+
| aaa  | test      | demo_test | c1          | c2                              | c3    | c4         | c5                | PRIMARY               |
| bbb  | test      | multicol  | country     | population,area_sq_km,drive_side | c3    | c4         | c5                | PRIMARY               |
+------+-----------+-----------+-------------+---------------------------------+-------+------------+-------------------+-----------------------+
2 rows in set (0.01 sec)
```

スクリプトを実行したあと、データは multicol テーブル内にあり、従来の MySQL クエリーまたは DML ステートメントから入手できます。

```
mysql: use test;
Database changed

mysql: select * from multicol;
+---------+------------+------------+------------+------+------+------+
| country | population | area_sq_km | drive_side | c3   | c4   | c5   |
+---------+------------+------------+------------+------+------+------+
| Canada  | 34820000   | 9984670    | R          |    0 |   11 |    0 |
| China   | 1347350000 | 9640821    | R          |    0 |   20 |    0 |
| Denmark | 5543453    | 43094      | R          |    0 |   16 |    0 |
| India   | 1210193422 | 3287263    | L          |    0 |   19 |    0 |
| Ireland | 6399152    | 84421      | L          |    0 |   13 |    0 |
| Mexico  | 113910608  | 1972550    | R          |    0 |   15 |    0 |
| Norway  | 5002942    | 385252     | R          |    0 |   17 |    0 |
```

```
| UAE     | 8264070    | 83600      | R          |    0 |   18 |    0 |
| UK      | 62262000   | 243610     | L          |    0 |   14 |    0 |
| USA     | 314242000  | 9826675    | R          |    0 |   12 |    0 |
+---------+------------+------------+------------+------+------+------+
10 rows in set (0.00 sec)

mysql: desc multicol;
+------------+---------------------+------+-----+---------+-------+
| Field      | Type                | Null | Key | Default | Extra |
+------------+---------------------+------+-----+---------+-------+
| country    | varchar(128)        | NO   | PRI |         |       |
| population | varchar(10)         | YES  |     | NULL    |       |
| area_sq_km | varchar(9)          | YES  |     | NULL    |       |
| drive_side | varchar(1)          | YES  |     | NULL    |       |
| c3         | int(11)             | YES  |     | NULL    |       |
| c4         | bigint(20) unsigned | YES  |     | NULL    |       |
| c5         | int(11)             | YES  |     | NULL    |       |
+------------+---------------------+------+-----+---------+-------+
7 rows in set (0.01 sec)
```

数値として扱われるカラムの長さを定義する場合、必要な桁、小数点、符号文字、先行ゼロなどを保持する十分なサイズを考慮します。VARCHAR などの文字列カラム内で値が長すぎると、その値は一部の文字を削除して切り捨てられ、不適切な数値が生成されることがあります。

SQL クエリーを使用して、country キーカラムだけでなく任意のカラムを使用して計算およびテストを行うと、レポートを作成できます。(これらの例では少数の国のデータのみが使用されているため、数値は例示のみを目的としています。)ここで、右側を運転する国の平均人口と、名前が「U」で始まる国の平均面積を求めます。

```
mysql: select avg(population) from multicol where drive_side = 'R';
+------------------+
| avg(population)  |
+------------------+
| 261304724.7142857 |
+------------------+
1 row in set (0.00 sec)

mysql: select sum(area_sq_km) from multicol where country like 'U%';
+-----------------+
| sum(area_sq_km) |
+-----------------+
|        10153885 |
+-----------------+
1 row in set (0.00 sec)
```

population および area_sq_km カラムは厳密に型指定された数値データでなく文字データを格納するため、avg() や sum() などの関数は、最初にそれぞれの値を数値に変換します。この手法は < や > などの演算子では機能せず、 たとえば、文字ベースの値を比較するとき、9 > 1000 となり、これは ORDER BY population DESC などの句で予期される結果ではありません。もっとも正確な型処理を行うには、数値カラムを適切な型にキャストするビューに対してクエリーを実行します。この技法では、非常に単純な SELECT * クエリーをデータベースアプリケーションから発行でき、キャスト、フィルタ、および並べ替えがすべて正しくなります。ここで、人口の上位 3 か国を降順で求めるためにクエリーを実行するビューを作成します。結果は常に multicol テーブルからの最新データを反映し、人口と面積は常に数値として扱われます。

```
mysql: create view populous_countries as
  select
    country,
    cast(population as unsigned integer) population,
    cast(area_sq_km as unsigned integer) area_sq_km,
    drive_side from multicol
  order by cast(population as unsigned integer) desc
  limit 3;
Query OK, 0 rows affected (0.01 sec)

mysql: select * from populous_countries;
+---------+------------+------------+------------+
| country | population | area_sq_km | drive_side |
+---------+------------+------------+------------+
| China   | 1347350000 |    9640821 | R          |
| India   | 1210193422 |    3287263 | L          |
| USA     |  314242000 |    9826675 | R          |
+---------+------------+------------+------------+
3 rows in set (0.00 sec)

mysql: desc populous_countries;
+------------+---------------------+------+-----+---------+-------+
| Field      | Type                | Null | Key | Default | Extra |
+------------+---------------------+------+-----+---------+-------+
```

```
| country    | varchar(128)        | NO   |     |         |       |
| population | bigint(10) unsigned | YES  |     | NULL    |       |
| area_sq_km | int(9) unsigned     | YES  |     | NULL    |       |
| drive_side | varchar(1)          | YES  |     | NULL    |       |
+------------+---------------------+------+-----+---------+-------+
4 rows in set (0.02 sec)
```

14.18.5.2 統合型の memcached デーモンのための既存の memcached アプリケーションの改変

MySQL 統合を使用するために既存の memcached アプリケーションを改変するときは、MySQL および InnoDB テーブルの次の側面について検討してください。

- 数バイトよりも長いキー値を持つ場合、InnoDB テーブルの主キーに対して自動インクリメントする数値カラムを使用し、memcached のキー値を保持するカラムに一意のセカンダリインデックスを作成する方が効率的な場合もあります。これは、主キー値が (自動インクリメント値を使用したときのように) ソートされた順に追加される場合、大規模な挿入で InnoDB のパフォーマンスが最大になり、主キー値が各セカンダリインデックスに複製され、主キーが長い文字列値の場合に不要なスペースを占有する可能性があるためです。
- いくつかの異なる種類の情報を memcached に格納するとき、データの種類ごとに別個の InnoDB テーブルをセットアップすることがあります。innodb_memcache.containers テーブル内に追加のテーブル識別子を定義し、@@table_id.key の表記を使用して、項目を異なるテーブルに格納したり取得したりします。項目を物理的に分割することで、各テーブルの特性を調整すると、最適なスペース使用率、パフォーマンス、および信頼性を得ることができます。たとえば、ブログ投稿を保持するテーブルに対して圧縮を有効にし、サムネイル画像を保持するテーブルを有効にしない場合があります。非常に重要なデータが保持されているテーブルは、別のテーブルよりも頻繁にバックアップする場合もあります。SQL を介してレポートを生成するために頻繁に使用されるテーブルで、追加のセカンダリインデックスを作成する場合もあります。
- できれば、memcached インタフェースで使用する安定したテーブル定義セットをセットアップし、永続的に保存します。containers テーブルへの変更は、テーブルに対してクエリーが次回実行されたときに有効になります。そのテーブル内のエントリは起動時に処理され、認識されていないテーブル ID が @@ 表記によってリクエストされるたびに参照されます。そのため、新しいエントリは、関連付けられたテーブル ID を使用しようとするとただちに表示されますが、既存のエントリではエントリが有効になる前にサーバーの再起動が必要です。
- デフォルトのキャッシュポリシー innodb_only を使用するとき、add()、set()、incr() などの呼び出しは成功しますが、while expecting 'STORED', got unexpected response 'NOT_STORED などのデバッグメッセージが発生します。これは、innodb_only 構成では、新しい値と更新された値が、メモリーキャッシュに保存されることなく InnoDB テーブルに直接送信されるためです。

14.18.5.3 InnoDB memcached プラグインのパフォーマンスのチューニング

InnoDB と memcached を組み合わせて使用すると、すべてのデータを即座にまたはしばらくしてからディスクに書き込むため、パフォーマンスが、memcached のみを使用する場合よりも若干低くなります。InnoDB memcached プラグインのチューニングでは、同等の SQL 操作よりも高いパフォーマンスを達成するよう設定してください。

ベンチマークでは、クエリーおよび DML 操作 (挿入、更新、および削除) は両方とも、従来の SQL よりも memcached インタフェースを経由した方が高速になります。通常は DML 操作の方が速度が大きく向上します。したがって、memcached インタフェースを使用するために最初に改変した方がよいアプリケーションのタイプは、書き込み処理の多いアプリケーションです。また、以前信頼性が優先されていなかった場合、高速かつ軽量なメカニズムを使用した書き込み処理の多いアプリケーションタイプでは、MySQL をデータストアとして使用する場合もあります。

SQL クエリーの改変

単純な GET リクエストスタイルのもっとも適合するクエリーのタイプは、単一句または AND 条件のセットを WHERE 句に指定したものです。

```
SQL:
select col from tbl where key = 'key_value';

memcached:
GET key_value

SQL:
select col from tbl where col1 = val1 and col2 = val2 and col3 = val3;

memcached:
# Since you must always know these 3 values to look up the key,
```

```
# combine them into a unique string and use that as the key
# for all ADD, SET, and GET operations.
key_value = val1 + ":" + val2 + ":" + val3
GET key_value

SQL:
select 'key exists!' from tbl
  where exists (select col1 from tbl where key = 'key_value') limit 1;

memcached:
# Test for existence of key by asking for its value and checking if the call succeeds,
# ignoring the value itself. For existence checking, you typically only store a very
# short value such as "1".
GET key_value
```

システムメモリーの利用

最適なパフォーマンスを得るには、InnoDB memcached プラグインを、典型的なデータベースサーバーのように構成されたマシン上に配備し、特に、innodb_buffer_pool_size 構成オプションを使用して、大部分のシステム RAM を InnoDB バッファープールに割り振ったデータベースサーバー上に配備します。複数ギガバイトのバッファープールを備えたシステムで、ほとんどの操作がメモリーにキャッシュされているデータを使用する場合、スループットが最大になるように innodb_buffer_pool_instances 構成オプションの値を高くします。

冗長 I/O の削減

InnoDB には、クラッシュ時の高い信頼性と、高書き込みワークロード時の I/O オーバーヘッドの量とのバランスを選択できる設定があります。たとえば、構成オプション innodb_doublewrite=0 および innodb_flush_log_at_trx_commit=2 を設定します。innodb_flush_method オプションの設定を変えて、パフォーマンスを測定します。サーバーのバイナリログがチューニングされていない場合、innodb_support_xa=0 の設定を使用します。

テーブル操作の I/O を削減したりチューニングしたりするその他の方法については、セクション8.5.7「InnoDB ディスク I/O の最適化」を参照してください。

トランザクションオーバーヘッドの削減

構成オプション daemon_memcached_r_batch_size および daemon_memcached_w_batch_size に対するデフォルト値 1 では、結果に対する最大限の信頼性と、保管または更新されるデータの安全性を確保します。

アプリケーションのタイプによっては、これらの設定の 1 つまたは両方を増やすと、頻繁なコミット操作によるオーバーヘッドを削減できる場合もあります。データ処理の多いシステムでは、SQL を介して行なったデータへの変更が memcached に即座に (つまり、get 操作があと N 回処理されるまで) 表示されなくなることを確認し、daemon_memcached_r_batch_size を増やすこともできます。すべての書き込み操作を確実に保管する必要があるデータ処理の場合、daemon_memcached_w_batch_size の設定は 1 のままにします。統計分析にのみ使用する大量の更新を処理する場合は、クラッシュ時に最後の N 回の更新が失われても大きな問題はないため、この設定を増やすこともできます。

たとえば、通行量の多い橋を渡る交通量をモニターし、1 日に約 100,000 台の車両を記録するシステムについて考えてみます。交通パターンを分析するために、単に異なるタイプの車両を数えるだけのアプリケーションの場合、daemon_memcached_w_batch_size を 1 から 100 に変更すると、コミット操作の I/O オーバーヘッドを 99% 削減できます。予想外の機能停止の場合、最大 100 件のレコードが失われても、許容誤差の可能性があります。一方、各車両に対する自動料金徴収をアプリケーションで実行している場合、daemon_memcached_w_batch_size の設定を 1 のままにし、料金徴収の記録を即座にディスクに保存する方がよい場合もあります。

InnoDB がディスク上で memcached キー値を編成する方法が原因で、大量のキーが作成される場合、データ項目をアプリケーション内でキー値でソートし、ソートした順序で追加した方が、任意の順序でキーを作成するよりも高速になります。

memslap コマンドは、さまざまな構成のベンチマークで便利です。これは通常の memcached 配布の一部ですが MySQL Server に含まれていません。また、独自のベンチマークで使用できるサンプルのキー/値のペアの生成に使用できます。詳細は、libmemcached のコマンド行ユーティリティーを参照してください。

14.18.5.4 InnoDB memcached プラグインのトランザクション動作の制御

従来の memcached とは異なり、InnoDB + memcached の組み合わせを使用すると、add、set、incr などの呼び出しを使用して生成されたデータ値の「永続性」を制御できます。MySQL はデータの永続性と一貫性に高い優先順位を置いているため、デフォルトでは、memcached インタフェース経由で書き込まれるすべてのデータは常にディスクに格納され、get の呼び出しは、ディスクの最新の値を常に返します。このデフォルト設定では、本来の

パフォーマンスを可能なかぎり高めることはできませんが、InnoDB テーブルの従来の SQL インタフェースと比べると、非常に高速です。

この機能を使用していくことで、機能が停止した場合に一部の更新済みの値を失ったり、古くなったデータを返したりするリスクを認識し、重要ではないデータの永続性の設定を緩和できるようになります。

コミットの頻度

永続性と本来のパフォーマンスを両立させる 1 つの条件は、新しいデータや変更されたデータがコミットされる頻度です。重要なデータの場合、データをただちにコミットして、クラッシュまたは機能停止の発生時にデータを保護する必要があります。クラッシュ後にリセットされるカウンタや、数秒分損失しても支障がない口ギングデータなどのあまり重要ではないデータの場合、コミットの頻度を下げて本来のスループットを高めた方がよいこともあります。

memcached 操作によって、ベースとなる InnoDB テーブル内で挿入、更新、または削除を行う場合、その変更は即座に (daemon_memcached_w_batch_size=1 の場合) またはしばらくしてから (この構成オプション値が 1 より大きいとき) 基礎テーブルにコミットされます。いずれの場合も、変更はロールバックできません。混雑時の高 I/O オーバーヘッドを回避するために daemon_memcached_w_batch_size=1 の値を増やすと、ワークロードの減少時にコミットの頻度が非常に低下する可能性があります。安全策として、バックグラウンドスレッドで、memcached API 経由で行なった変更を一定の間隔で自動的にコミットします。間隔は innodb_api_bk_commit_interval 構成オプションで制御され、デフォルトは 5 秒です。

memcached 操作によって、ベースとなる InnoDB テーブルに挿入または更新が発生した場合、変更されたデータはほかの memcached リクエストによってすぐに表示されます。これは、新しい値は MySQL 側でまだコミットされていなくてもメモリーキャッシュ内に残っているためです。

トランザクションの分離

get や incr などの memcached 操作によって、ベースとなる InnoDB テーブルでクエリーまたは DML 操作を実行する場合、テーブルに書き込まれた最新データの表示、コミットされたデータのみの表示、ほかのトランザクション分離レベルの表示を制御できます。この機能は innodb_api_trx_level 構成オプションを使用して制御します。このオプションで指定される数値は、REPEATABLE READ のようなわかりやすい分離レベル名に対応します。全リストについては、innodb_api_trx_level オプションの説明を参照してください。

分離レベルの厳密性が高くなると、取得したデータが突然ロールバックしたり変更したりして後続のクエリーで異なる値が表示されることがなくなります。ただし、厳密性が高いと ロックによるオーバーヘッドが大きくなり待機が発生します。長期間にわたるトランザクションを使用しない NoSQL スタイルのアプリケーションでは、通常はデフォルトの分離レベルのままにするか、低い厳密性に切り換えることができます。

memcached DML 操作の行ロックの無効化

innodb_api_disable_rowlock オプションでは、InnoDB memcached が DML 操作を実行するときに行ロックを無効化できます。デフォルトでは、innodb_api_disable_rowlock は OFF に設定されており、memcached が get および set 操作の行ロックをリクエストします。innodb_api_disable_rowlock を ON に設定すると、memcached は行ロックの代わりに、テーブルロックをリクエストします。

innodb_api_disable_rowlock オプションは動的ではありません。mysqld コマンド行で起動時に指定するか、MySQL 構成ファイルに入力する必要があります。

DDL の許可または禁止

デフォルトでは、InnoDB memcached プラグインによって使用されるテーブル上で ALTER TABLE などの DDL 操作を実行できます。スループットの高いアプリケーションでのテーブル使用時の潜在的な速度低下を回避するには、innodb_api_enable_mdl 構成オプションを起動時にオンにすると、テーブル上の DDL 操作を無効にできます。このオプションは、memcached インタフェースおよび SQL の両方で同じ基礎テーブルにアクセスする場合はあまり適切ではありません。このオプションでは、レポート作成クエリーを実行するシステムの構成で重要になる、テーブルの CREATE INDEX ステートメントをブロックするためです。

ディスク、メモリー、または両方へのデータの格納

テーブル innodb_memcache.cache_policies は、memcached から書き込まれるデータが、ディスク上に格納されるか (デフォルト、innodb_only)、従来の memcached のようにメモリーのみに格納されるか (cache-only)、または両方に格納されるか (caching) を指定します。

caching 設定では、memcached がメモリー内からキーを検出できない場合、InnoDB テーブル内から値を検索します。caching 設定で get 呼び出しから返された値は、ディスク上の InnoDB テーブルで更新され、メモリーキャッシュでまだ期限切れとなっていない場合、値が最新でないことがあります。

キャッシュポリシーは、get、set (incr および decr を含む)、delete、および flush 操作で個々に設定できます。例:

- get および set 操作では、(caching 設定を使用して) テーブルおよび memcached メモリーキャッシュへのクエリー実行または更新を同時に実行しながら、delete、flush、またはこれらの両方を (cache_only 設定を使用して) インメモリーコピー上でのみ動作できます。こうすることで、項目を削除またはフラッシュすると単にキャッシュで期限切れになり、項目が次回要求されたときに最新の値が InnoDB テーブルから返されます。

```
mysql> desc innodb_memcache.cache_policies;
+--------------+---------------------------------------------------------+------+-----+---------+-------+
| Field        | Type                                                    | Null | Key | Default | Extra |
+--------------+---------------------------------------------------------+------+-----+---------+-------+
| policy_name  | varchar(40)                                             | NO   | PRI | NULL    |       |
| get_policy   | enum('innodb_only','cache_only','caching','disabled') | NO   |     | NULL    |       |
| set_policy   | enum('innodb_only','cache_only','caching','disabled') | NO   |     | NULL    |       |
| delete_policy | enum('innodb_only','cache_only','caching','disabled') | NO   |     | NULL    |       |
| flush_policy | enum('innodb_only','cache_only','caching','disabled') | NO   |     | NULL    |       |
+--------------+---------------------------------------------------------+------+-----+---------+-------+

mysql> select * from innodb_memcache.cache_policies;
+--------------+-------------+-------------+---------------+--------------+
| policy_name  | get_policy  | set_policy  | delete_policy | flush_policy |
+--------------+-------------+-------------+---------------+--------------+
| cache_policy | innodb_only | innodb_only | innodb_only   | innodb_only  |
+--------------+-------------+-------------+---------------+--------------+

mysql> update innodb_memcache.cache_policies set set_policy = 'caching'
   -> where policy_name = 'cache_policy';
```

cache_policies 値は起動時にのみ読み取られ、memcached プラグインの動作と強固に統合されています。このテーブル中の値を変更したあと、プラグインをアンインストールし再インストールします。

```
mysql> uninstall plugin daemon_memcached;
Query OK, 0 rows affected (2.00 sec)
mysql> install plugin daemon_memcached soname "libmemcached.so";
Query OK, 0 rows affected (0.00 sec)
```

14.18.5.5 memcached 操作に合わせた DML ステートメントの改変

ベンチマークでは、InnoDB memcached プラグインは、クエリーを高速化するだけではなく DML 操作 (挿入、更新、および削除) を高速化することが示唆されています。初期の開発作業では、書き込み処理の多い I/O バウンドのアプリケーションに集中し、書き込み処理の多い新しい種類のアプリケーションで MySQL を使用する機会を検討する方がよい場合もあります。

- ```
  INSERT INTO t1 (key,val) VALUES (some_key,some_value);
  SELECT val FROM t1 WHERE key = some_key;
  UPDATE t1 SET val = new_value WHERE key = some_key;
  UPDATE t1 SET val = val + x WHERE key = some_key;
  DELETE FROM t1 WHERE key = some_key;
  ```

  単一行の DML ステートメントは memcached 操作に変換するもっとも単純な種類のステートメントで、INSERT は add に、UPDATE は set、incr、または decr に、DELETE は delete になります。memcached インタフェースを介して発行すると、テーブル内でキーは一意であるため、これらの操作は 1 行のみに影響を与えることができます。

  前述の SQL の例で、t1 は innodb_memcache.containers テーブルの構成設定に基づき InnoDB memcached プラグインによって現在使用されているテーブルを参照し、key は key_columns の下にリストされているカラムを参照し、val は value_columns の下にリストされているカラムを参照します。

- ```
  TRUNCATE TABLE t1;
  DELETE FROM t1;
  ```

  前のステップのように、t1 が memcached 操作についてのテーブルとして構成されている場合に、flush_all 操作に相当します。テーブル中のすべての行を削除します。

14.18.5.6 ベースとなる InnoDB テーブルでの DML および DDL ステートメントの実行

標準の SQL インタフェースを介して InnoDB テーブル (デフォルトは test.demo_test) にアクセスできます。ただし、いくつかの制約があります。

- memcached インタフェース経由でもアクセスしているテーブルで SQL 経由でクエリーを実行する場合、memcached 操作は、すべての書き込み操作のあとではなく定期的にコミットするよう構成できます。この動作は、daemon_memcached_w_batch_size オプションによって制御されます。このオプションが 1 より大きい値に設定されている場合、挿入されたばかりの行を検出するには READ UNCOMMITTED クエリーを使用します。

```
mysql> set session TRANSACTION ISOLATION LEVEL read uncommitted;
Query OK, 0 rows affected (0.00 sec)

mysql> select * from demo_test;
+------+------+------+------+-----------+------+------+------+------+------+------+
| cx   | cy   | c1   | cz   | c2        | ca   | CB   | c3   | cu   | c4   | C5   |
+------+------+------+------+-----------+------+------+------+------+------+------+
| NULL | NULL | a11  | NULL | 123456789 | NULL | NULL |   10 | NULL |    3 | NULL |
+------+------+------+------+-----------+------+------+------+------+------+------+
1 row in set (0.00 sec)
```

- memcached インタフェース経由でもアクセスしているテーブルを SQL 経由で変更する場合、memcached 操作を、すべての読み取り操作のあとではなく定期的に新しいトランザクションを開始するよう構成できます。この動作は、daemon_memcached_r_batch_size オプションによって制御されます。このオプションが 1 より大きい値に設定されている場合、...

- InnoDB テーブルは、トランザクション内のすべての操作で IS (共有インテンション) または IX (排他インテンション) でロックされます。daemon_memcached_r_batch_size および daemon_memcached_w_batch_size をデフォルト値の 1 よりも著しく増加させた場合、テーブルはほとんどの場合、各操作間で意図的にロックされ、ユーザーはテーブル上で DDL ステートメントを実行できなくなります。

14.18.6 レプリケーションでの InnoDB memcached プラグインの使用

InnoDB memcached デーモンプラグインは MySQL バイナリログをサポートしているため、memcached インタフェース経由でマスターサーバーに実行した更新は、バックアップ、読み取り処理の多いワークロードのバランスの確保、および高可用性のためにレプリケーションできます。すべての memcached コマンドは、バイナリロギング用にサポートされています。

スレーブサーバー上で InnoDB memcached プラグインをセットアップする必要はありません。この構成の主な利点は、マスター上での書き込みスループットの増加です。レプリケーションメカニズムの速度は影響を受けません。

次のセクションでは、バイナリログ機能を使用して、InnoDB memcached プラグインを MySQL レプリケーションと一緒に使用する方法を示します。セクション14.18.3「InnoDB Memcached プラグインの概要」に記載されている基本設定がすでに実行されていることを前提としています。

innodb_api_enable_binlog による InnoDB Memcached バイナリログの有効化:

- InnoDB memcached プラグインを MySQL バイナリログと一緒に使用するには、マスターサーバー上で innodb_api_enable_binlog 構成オプションを有効にします。このオプションはサーバーのブート時にのみ設定できます。また、--log-bin オプションを使用して、マスターサーバー上で MySQL バイナリログを有効にする必要もあります。これらのオプションを、my.cnf などのサーバー構成ファイルに追加したり、mysqld コマンド行に追加したりできます。

```
mysqld ... --log-bin --innodb_api_enable_binlog=1
```

- 次に、セクション17.1.1「レプリケーションのセットアップ方法」に記載されているようにマスターサーバーおよびスレーブサーバーを構成します。

- mysqldump を使用してマスターデータスナップショットを作成し、これをスレーブサーバーに同期します。

```
master shell: mysqldump --all-databases --lock-all-tables > dbdump.db
slave shell: mysql < dbdump.db
```

- マスターサーバー上で、show master status を発行してマスターバイナリログ座標を取得します。

```
mysql> show master status;
```

- スレーブサーバー上で、change master to ステートメントを使用して、上記の座標でスレーブサーバーをセットアップします。

```
mysql> CHANGE MASTER TO
       MASTER_HOST='localhost',
       MASTER_USER='root',
       MASTER_PASSWORD='',
       MASTER_PORT = 13000,
       MASTER_LOG_FILE='0.000001,
       MASTER_LOG_POS=114;
```

- 次に、スレーブを開始します。

```
mysql> start slave;
```

次のような出力がエラーログから出力される場合、スレーブはレプリケーションの準備ができています。

```
111002 18:47:15 [Note] Slave I/O thread: connected to master 'root@localhost:13000',
replication started in log '0.000001' at position 114
```

memcached telnet インタフェースによるテスト

上記のレプリケーション設定を備えたサーバーをテストするには、memcached telnet インタフェースを使用し、さらに SQL を使用してマスターサーバーおよびスレーブサーバーにクエリーを実行して、結果を検証します。

ここでの構成設定 SQL で、memcached によって使用される test データベース内に 1 つのサンプルテーブル demo_test が作成されます。このデフォルトテーブルをデモ用に使用します。

- set を使用して、キー test1、値 t1、およびフラグ 10 を持つレコードを挿入します。

```
telnet 127.0.0.1 11211
Trying 127.0.0.1...
Connected to 127.0.0.1.
Escape character is '^]'.
set test1 10 0 2
t1
STORED
```

マスターサーバーで、行が挿入されたことを確認できます。c1 はキーにマップされ、c2 は値にマップされ、c3 はフラグ、c4 は cas 値、および c5 は期限です。

```
mysql> select * from test.demo_test;
```

c1	c2	c3	c4	c5
test1	t1	10	2	0

```
1 row in set (0.00 sec)
```

スレーブサーバーでは、レプリケーションによって同じレコードが挿入されたことがわかります。

```
mysql> select * from test.demo_test;
```

c1	c2	c3	c4	c5
test1	t1	10	2	0

```
1 row in set (0.00 sec)
```

- set コマンドを使用して、キー test1 を新しい値 new に更新します。

```
Connected to 127.0.0.1.
Escape character is '^]'.
set test1 10 0 3
new
STORED
```

スレーブサーバーで、更新がレプリケーションされます (cas 値も更新されます)。

```
mysql> select * from test.demo_test;
```

c1	c2	c3	c4	c5
test1	new	10	3	0

1 row in set (0.00 sec)

- delete コマンドでレコードを削除します。

```
Connected to 127.0.0.1.
Escape character is '^]'.
delete test1
DELETED
```

削除がスレーブにレプリケーションされると、スレーブ上のレコードも削除されます。

```
mysql> select * from test.demo_test;
Empty set (0.00 sec)
```

- flush_all コマンドでテーブルを削除します。

最初に、telnet でマスターサーバーに接続して 2 つのレコードを挿入します。

```
Connected to 127.0.0.1.
Escape character is '^]'
set test2 10 0 5
again
STORED
set test3 10 0 6
again1
STORED
```

スレーブサーバーで、これらの 2 つのレコードがレプリケーションされることを確認します。

```
mysql> select * from test.demo_test;
```

c1	c2	c3	c4	c5
test2	again	10	5	0
test3	again1	10	6	0

2 rows in set (0.00 sec)

telnet インタフェースで flush_all を呼び出してテーブルを削除します。

```
Connected to 127.0.0.1.
Escape character is '^]'.
flush_all
OK
```

次に、削除操作がスレーブサーバーにレプリケーションされたことを確認します。

```
mysql> select * from test.demo_test;
Empty set (0.00 sec)
```

すべての memcached コマンドは、レプリケーションでサポートされます。

InnoDB Memcached Binlog に関する注意点

Binlog 形式:

- ほとんどの memcached 操作は (挿入、削除、更新に類似した) DML ステートメントにマップされます。MySQL Server によって実際に処理される SQL ステートメントがないため、memcached コマンド (flush_all 以外) は行ベースのレプリケーション (RBR) ロギングを使用します。これはサーバーのすべての binlog_format 設定とは無関係です。
- memcached flush_all コマンドは TRUNCATE TABLE コマンドにマップされます。DDL コマンドはステートメントベースのロギングのみ使用できるため、この flush_all コマンドは TRUNCATE TABLE ステートメントを送信することでレプリケーションされます。

トランザクション:

- トランザクションの概念は、これまで通常は memcached アプリケーションの一部をなすものではありませんでした。ここでは daemon_memcached_r_batch_size および daemon_memcached_w_batch_size を使用して、パフォーマンスを考慮した読み取りおよび書き込みトランザクションのバッチサイズを制御します。これらの設定はレプリケーションに影響せず、基礎テーブル上でのすべての SQL 操作は実行完了後すぐにレプリケーションされます。
- daemon_memcached_w_batch_size のデフォルト値は 1 であるため、memcached の書き込み操作は即時にコミットされます。このデフォルト設定は、マスターサーバーとスレーブサーバーに表示されるデータの不整合を回避するために、ある程度のパフォーマンスオーバーヘッドを発生させます。レプリケートされたレコードは、常にスレーブサーバー上でただちに使用可能になります。daemon_memcached_w_batch_size を 1 より大きく設定した場合、memcached インタフェースを介して挿入または更新されたレコードは、マスターサーバー上にすぐには表示されません。これらのレコードがコミットされる前にレコードをマスターサーバー上に表示するには、set transaction isolation level read uncommitted を発行します。

14.18.7 InnoDB memcached プラグインの内部構造

InnoDB memcached プラグイン用の InnoDB API

InnoDB memcached エンジンは InnoDB API を介して InnoDB にアクセスします。ほとんどの API は、組み込まれた InnoDB から直接採用されます。InnoDB API 関数は、「コールバック関数」として InnoDB memcached に渡されます。InnoDB API 関数は InnoDB テーブルに直接アクセスし、TRUNCATE TABLE 操作を除いてほとんどが DML 操作です。

次にリストされたすべての memcached コマンドは、 InnoDB memcached API によって実装されます。次の表は、それぞれの memcached コマンドが DML 操作にどのようにマップされるかの概要を示しています。

表 14.10 memcached コマンドおよび関連付けられた DML 操作

memcached コマンド	DML 操作
get	読み取り/フェッチコマンド
set	検索後に挿入または更新を実行する (キーが存在するかどうかに依存する)
add	検索後に挿入または更新を実行する
replace	検索後に更新を実行する
append	検索後に更新を実行する (更新前に結果の後ろにデータを付加する)
prepend	検索後に更新を実行する (更新前に結果の前にデータを付加する)
incr	検索後に更新を実行する
decr	検索後に更新を実行する
delete	検索後に削除を実行する
flush_all	テーブルを削除する

InnoDB memcached プラグインで使用される基礎テーブル

このセクションでは、InnoDB memcached プラグインによって使用される基礎テーブルについて説明します。

innodb_memcached_config.sql 構成スクリプトは InnoDB memcached プラグインによって必要となる 3 つのテーブルをインストールします。テーブルは専用の innodb_memcache データベースに作成されます。

```
mysql> USE innodb_memcache;
Database changed
mysql> SHOW TABLES;
+---------------------------+
| Tables_in_innodb_memcache |
+---------------------------+
| cache_policies            |
| config_options            |
| containers                |
+---------------------------+
3 rows in set (0.01 sec)
```

containers テーブル

containers テーブルは 3 つのテーブルの中でもっとも重要です。このテーブル内のエントリは、memcached 値を格納するために使用される InnoDB テーブル用の「コンテナ」です。コンテナは、InnoDB テーブルのカラムを下

記の表で示す値にマップします。このマッピングは memcached が InnoDB テーブルと一緒に機能するために必要です。

containers テーブルには test.demo_test テーブル用のデフォルトエントリがあります。InnoDB memcached プラグインを独自の InnoDB テーブルと一緒に使用するには、使用するテーブルのエントリを containers テーブルに追加する必要があります。

表 14.11 containers のカラム

カラム	説明
name	コンテナに付与された名前。
db_schema	InnoDB テーブルが存在するデータベースの名前。これは必須の値です。
db_table	memcached 値を格納する InnoDB テーブルの名前。これは必須の値です。
key_columns	memcached 操作のための検索キー値を格納する InnoDB テーブルのカラム。これは必須の値です。
value_columns	memcached データを格納する InnoDB テーブル内のカラム (1 つ以上)。innodb_memcached.config_options テーブルで指定された区切り文字を使用して複数のカラムを指定できます。デフォルトでは、区切り文字はパイプ文字 (「|」) です。複数カラムを指定するには、定義された区切り文字でカラムを区切ります。たとえば、col1|col2|col3 となります。これは必須の値です。
flags	memcached のフラグ (メインの値と一緒に格納および取得されるユーザー定義の数値) として使用される InnoDB テーブル内のカラムを指定します。memcached 値が複数カラムにマップされる場合、フラグ値はいくつかの操作 (incr や prepend など) のカラム指定子として使用でき、その結果、指定されたカラムで操作が実行されます。たとえば、3 つのカラムに値をマップしていて、これらのカラムのうちの 1 つでインクリメント操作が実行する場合に、これらの操作に使用するカラムを指定するには flags を使用できます。flags カラムを使用しない場合は、未使用であることを示すためにその値を 0 に設定します。
cas_column	比較およびスワップ (cas) 値を格納する InnoDB テーブル内のカラム。cas_column 値と expire_time_column 値は、memcached が別のサーバーへのリクエストをハッシュし、データをメモリーにキャッシュする方法に関係します。InnoDB memcached プラグインは単一の memcached デーモンに強固に統合され、インメモリーキャッシュメカニズムは MySQL およびバッファープールによって処理されるため、このタイプの配備でこれらのカラムはほとんど必要ありません。これらのカラムを使用しない場合、値を 0 に設定してカラムが未使用であることを示します。
expire_time_column	有効期限の値を格納する InnoDB テーブル内のカラム。cas_column 値と expire_time_column 値は、memcached が別のサーバーへのリクエストをハッシュし、データをメモリーにキャッシュする方法に関係します。InnoDB memcached プラグインは単一の memcached デーモンに強固に統合され、インメモリーキャッシュメカニズムは MySQL およびバッファープールによって処理されるため、このタイプの配備でこれらのカラムはほとんど必要ありません。これらのカラムを使用しない場合、値を 0 に設定してカラムが未使用であることを示します。
unique_idx_name_on_key	キーカラムのインデックスの名前。これは一意のインデックスである必要があります。これは主キーまたはセカンダリインデックスにできます。できれば、

カラム	説明
	キーカラムを InnoDB テーブルの主キーにしてください。こうすることで、このカラムに対してセカンダリインデックスを使用するための検索ステップが省かれます。memcached 参照のためのカバリングインデックスは作成できません。キーカラムおよび値カラムの両方に複合セカンダリインデックスを定義しようとすると、InnoDB はエラーを返します。

containers テーブルカラムの制約

- db_schema、db_name、key_columns、value_columns、および unique_idx_name_on_key の値を指定する必要があります。そうしない場合、セットアップは失敗します。flags、cas_column、および expire_time_column が使用されない場合、これらに 0 を指定します。そうしないと、セットアップが失敗する場合があります。
- key_columns: memcached で強制される、memcached キーの最大長は 250 文字です。最大長を超えるマップ済みのキーが使用された場合、操作は失敗します。マップ済みのキーは、Null 以外の CHAR または VARCHAR タイプである必要があります。
- value_columns: CHAR、VARCHAR、または BLOB カラムにマップされる必要があります。長さに制約はなく、値を NULL に指定できます。
- cas_column: cas 値は 64 ビットの整数です。これは少なくとも 8 バイトの BIGINT にマップされる必要があります。このカラムを使用しない場合は、未使用であることを示すためにその値を 0 に設定します。
- expiration_time_column: 少なくとも 4 バイトの INTEGER にマップされる必要があります。有効期限は、Unix 時間の 32 ビット整数 (1970 年 1 月 1 日からの秒数の 32 ビット値) として、または現在時間から開始する秒数として定義されます。後者の場合、秒数は 60*60*24*30 (30 日間の秒数) を超えないようにしてください。クライアントによって送信された数値の方が大きい場合、サーバーは現在時間からのオフセットではなく実際の Unix 時間の値とみなします。このカラムを使用しない場合は、未使用であることを示すためにその値を 0 に設定します。
- flags: 少なくとも 32 ビットの INTEGER にマップする必要があり、NULL に指定できます。このカラムを使用しない場合は、未使用であることを示すためにその値を 0 に設定します。

カラム制約を強制するために、プラグインのロード時に事前検査が行われます。不一致が見つかった場合、プラグインはロードされません。

cache_policies テーブル

cache_policies テーブルは、InnoDB memcached セットアップ用のキャッシュポリシーを定義します。単一のキャッシュポリシー内で、get、set、delete、および flush 操作に対する個々のポリシーを指定できます。すべての操作のデフォルト設定は innodb_only です。

- innodb_only: InnoDB を memcached のデータストアとして使用します。
- cache-only: 従来の memcached エンジンをデータストアとして使用します。
- innodb_only: InnoDB と従来の memcached エンジンの両方をデータストアとして使用します。この状況で、memcached がメモリー内からキーを検出できない場合、InnoDB テーブル内から値を検索します。
- disable: キャッシュを無効にします。

表 14.12 cache_policies カラム

カラム	説明
policy_name	キャッシュポリシーの名前。デフォルトのキャッシュポリシー名は cache_policy です。
get_policy	get 操作のキャッシュポリシー。有効な値は、innodb_only、cache-only、caching、または disabled です。デフォルト設定は innodb_only です。
set_policy	set 操作のキャッシュポリシー。有効な値は、innodb_only、cache-only、caching、または disabled です。デフォルト設定は innodb_only です。

カラム	説明
delete_policy	delete 操作のキャッシュポリシー。有効な値は、innodb_only、cache-only、caching、または disabled です。デフォルト設定は innodb_only です。
flush_policy	flush 操作のキャッシュポリシー。有効な値は、innodb_only、cache-only、caching、または disabled です。デフォルト設定は innodb_only です。

config_options テーブル

config_options テーブルは、SQL を使用して実行時に変更できる memcached 関連の設定を格納します。現時点でサポートされる構成オプションは、separator および table_map_delimiter です。

表 14.13 config_options カラム

カラム	説明

Name	memcached 関連の構成オプションの名前。現時点では、次の構成オプションが config_options テーブル経由でサポートされます。
	• separator: 複数の value_columns が定義されている場合に、1 つの長い文字列の値を別々の値に分離するために使用します。デフォルトでは、separator は | 文字です。たとえば、値カラムとして col1, col2 を定義し、区切り文字として | を定義した場合、memcached で次のコマンドを発行すると、値を col1 および col2 にそれぞれ挿入できます。
	`set keyx 10 0 19`
	`valuecolx|valuecoly`
	valuecol1x は col1 に格納され、valuecoly は col2 に格納されます。
	• table_map_delimiter: 特定のテーブル内のキーにアクセスするために、キー名に @@ 表記を使用するときの、スキーマ名とテーブル名を区切る文字。たとえば、@@t1.some_key と @@t2.some_key は同じキー値を持っていますが、異なるテーブルに格納されます。

Value	memcached 関連の構成オプションに割り当てられた値。

複数カラムのマッピング

- プラグインの初期化中に、InnoDB memcached が、containers テーブル内に定義されている情報を使用して構成されるとき、value_columns から構文解析されるマップされた各カラムは、マップされたテーブルに照らして検証されます。複数カラムがマップされる場合、カラムがそれぞれ存在し、正しいタイプであるかどうかを確認するための検査があります。
- 実行時に、memcached 挿入操作で、マップされたカラムの数よりも多くの区切り文字が値に存在する場合、マップされた値の数のみが取得されます。たとえば、マップされたカラムが 6 つで、区切られた値が 7 つ提供された場合、最初の 6 つの区切られた値のみが取得されます。7 番目の区切られた値は無視されます。
- マップされたカラムより区切られた値の方が少ない場合、入力値のないカラムは NULL に設定されます。未記入カラムを NULL に指定できない場合、挿入は失敗します。
- マップされた値より多くのカラムがテーブルにある場合、余分なカラムは出力結果に影響を及ぼしません。

テーブルの例

innodb_memcached_config.sql 構成スクリプトは、例として test データベース内にテーブル demo_test を作成します。また、追加のテーブルを作成することなく InnoDB memcached プラグインをただちに動作させることもできます。

container テーブルのエントリは、上記のいずれの目的で、どのカラムが使用されるかを定義します。

```
mysql> select * from innodb_memcache.containers;
+------+-----------+-----------+-------------+---------------+-------+------------+--------------------+------------------------+
| name | db_schema | db_table  | key_columns | value_columns | flags | cas_column | expire_time_column | unique_idx_name_on_key |
+------+-----------+-----------+-------------+---------------+-------+------------+--------------------+------------------------+
| aaa  | test      | demo_test | c1          | c2            | c3    | c4         | c5                 | PRIMARY                |
+------+-----------+-----------+-------------+---------------+-------+------------+--------------------+------------------------+
1 row in set (0.00 sec)

mysql> desc test.demo_test;
+-------+---------------------+------+-----+---------+-------+
| Field | Type                | Null | Key | Default | Extra |
+-------+---------------------+------+-----+---------+-------+
| c1    | varchar(32)         | NO   | PRI |         |       |
| c2    | varchar(1024)       | YES  |     | NULL    |       |
| c3    | int(11)             | YES  |     | NULL    |       |
| c4    | bigint(20) unsigned | YES  |     | NULL    |       |
| c5    | int(11)             | YES  |     | NULL    |       |
+-------+---------------------+------+-----+---------+-------+
5 rows in set (0.01 sec)
```

テーブル ID がキー名の @@ 表記によってリクエストされない場合は、次のようになります。

- ある行の name 値が default の場合、対応するテーブルが memcached プラグインによって使用されます。したがって、innodb_memcache.containers の最初のエントリを demo_test テーブルより後ろに移動させた場合、name 値に default を使用します。
- default の innodb_memcache.containers.name 値が存在しない場合、アルファベット順で先頭の name 値を持つ行が使用されます。

14.18.8 InnoDB memcached プラグインのトラブルシューティング

次のリストは、InnoDB memcached プラグイン使用時に発生する潜在的な問題と、解決策または回避策があればそれを示しています。

- MySQL エラーログにこのエラーが表示された場合、サーバーが起動に失敗することがあります。

  ```
  failed to set rlimit for open files. Try running as root or requesting
  smaller maxconns value.
  ```

  エラーメッセージは実際には memcached デーモンから出されたものです。1 つの解決策は、開くファイルの数について OS での制限を引き上げることです。コマンドはオペレーティングシステムによって異なります。たとえば、いくつかのオペレーティングシステム上で制限を確認して増加させるコマンドを、次に示します。

  ```
  # Linux
  $ ulimit -n
  1024
  ulimit -n 4096
  $ ulimit -n
  4096

  # OS X Lion (10.6)
  $ ulimit -n
  256
  ulimit -n 4096
  $ ulimit -n
  4096
  ```

  別の解決策は、-c オプションを使用して、memcached デーモンで使用可能な同時接続数を削減することで、デフォルトは 1024 です。MySQL 構成ファイル内の MySQL オプション daemon_memcached_option を使用して、この memcached オプションをエンコードします。

  ```
  [mysqld]
  ...
  loose-daemon_memcached_option='-c 64'
  ```

- memcached デーモンが InnoDB テーブルにデータを格納したり、またはデータを取得したりできない場合に問題をトラブルシューティングするには、MySQL 構成オプション daemon_memcached_option を使用し

て、memcached オプション -vvv を指定します。MySQL エラーログを調べて、memcached 操作に関係するデバッグ出力がないか検査します。

- memcached 項目値を保持するように指定されたカラムのデータ型が間違っている場合 (たとえば文字列型の代わりに数値型を指定するなど)、キー/値のペアを格納しようとすると、特定のエラーコードまたはメッセージを出さずに失敗します。
- daemon_memcached プラグインで、MySQL Server の開始に関して何らかの問題が発生する場合、MySQL 構成ファイル内の [mysqld] グループの下に次の行を追加して、トラブルシューティング中にプラグインを無効にします。

```
daemon_memcached=OFF
```

たとえば、必要なデータベースおよびテーブルをセットアップするための innodb_memcached_config.sql 構成スクリプトを実行する前に install plugin コマンドを実行した場合、サーバーがクラッシュして開始できないことがあります。または、innodb_memcache.containers テーブルに正しくないエントリをセットアップした場合、サーバーは開始できないことがあります。

MySQL インスタンス用の memcached プラグインを永続的にオフにするには、次のコマンドを発行します。

```
mysql> uninstall plugin daemon_memcached;
```

- 同じマシン上で MySQL の複数インスタンスを実行し、それぞれの memcached デーモンプラグインを有効にした場合、daemon_memcached_option 構成オプションを使用して、一意の memcached ポートを指定してください。
- 期待するテーブルを SQL ステートメントで検出できなかったり、テーブル内にデータがなかったりする場合でも、memcached API 呼び出しが引き続き動作し、期待するデータを取得できる場合もあります。これが発生するのは、innodb_memcache.containers テーブル内にエントリをセットアップしなかったか、GET または SET リクエストにキー @@table_id を指定して発行することによってそのテーブルに切り換えなかったか、innodb_memcache.containers 内の既存のエントリに変更を加えたあとで MySQL Server を再起動しなかった場合です。すべてのデータを単一カラム内に格納する test.demo_test テーブルをデーモンが使用中であっても、自由形式のストレージメカニズムが柔軟なため、col1|col2|col3 のようなマルチカラム値を格納または取得するためのリクエストは、通常は引き続き機能します。
- ユーザー独自の InnoDB テーブルを InnoDB memcached と一緒に使用するように定義し、テーブル内のカラムが NOT NULL に定義されている場合、InnoDB テーブル用のディスクリプタを memcached containers テーブル (innodb_memcached.containers) に挿入するとき、NOT NULL カラムに値を指定します。マップされたカラムよりも、ディスクリプタの INSERT ステートメントに含まれる区切られた値の方が少ない場合、入力値のないカラムは NULL に設定されます。NULL 値を NOT NULL カラムに挿入しようとすると、INSERT は失敗しますが、このことは InnoDB memcached プラグインを再初期化して変更内容を containers テーブルに適用したあとではじめて明らかになります。
- innodb_memcached.containers テーブルの cas_column および expire_time_column を NULL に設定した場合、memcached プラグインをロードしようとすると次のエラーが返されます。

```
InnoDB_Memcached: column 6 in the entry for config table 'containers' in
database 'innodb_memcache' has an invalid NULL value.
```

プラグイン memcached は cas_column および expire_time_column カラムでの NULL の使用を拒否します。これらのカラムを使用しない場合は、カラムの値を 0 に設定してください。

- memcached キーと値の長さが増加するにつれ、異なるポイントでサイズと長さの制限が発生します。
  - キーのサイズが 250 バイトを超える場合、memcached 操作はエラーを返します。これは memcached 内での現在の固定制限値です。
  - 値のサイズが 768 バイトを超えるか、3072 バイトを超えるか、innodb_page_size で指定されたサイズの 1/2 を超えた場合、InnoDB に関する制限が発生することがあります。これらの制限は、値カラムにインデックスを作成し、そのカラムで SQL からレポート生成クエリーを実行しようとするときに主に適用されます。詳細は、セクション14.6.7「InnoDB テーブル上の制限」を参照してください。
  - キーと値を組み合わせた最大サイズは 1M バイトです。
- 異なるバージョンを持つ MySQL Server 間で構成ファイルを共有している場合、memcached プラグインの最新の構成オプションを使用すると、古い MySQL バージョンで起動エラーが発生する可能性

があります。互換性の問題を回避するには、これらのオプション名の loose 形式を使用し、たとえば daemon_memcached_option='-c 64' でなく loose-daemon_memcached_option='-c 64' と指定します。

- 文字セットの設定を検証するための制約もチェックもありません。memcached は、キーおよび値をバイト形式で格納および取得するため、文字セットの違いは区別されません。ただし、memcached クライアントと MySQL テーブルで、同じ文字セットを使用する必要があります。

14.19 InnoDB のトラブルシューティング

InnoDB の問題のトラブルシューティングには、次の一般的なガイドラインが適用されます。

- 操作が失敗したか、またはバグが疑われる場合は、MySQL サーバーのエラーログを調べてください (セクション5.2.2「エラーログ」を参照してください)。
- その失敗がデッドロックに関連している場合は、InnoDB の各デッドロックに関する詳細が MySQL サーバーのエラーログに出力されるように、innodb_print_all_deadlocks オプションが有効になった状態で実行します。
- InnoDB データディクショナリに関する問題には、CREATE TABLE ステートメントの失敗 (孤立したテーブルファイル)、.InnoDB ファイルを開くことができない、および 「指定されたパスを見つけられませんでした」 エラーがあります。これらの種類の問題やエラーについては、セクション14.19.3「InnoDB データディクショナリの操作のトラブルシューティング」を参照してください。
- トラブルシューティング時は通常、mysqld_safe 経由、または Windows サービスとしてではなく、コマンドプロンプトから MySQL サーバーを実行することが最善です。それにより、mysqld がコンソールに出力する内容を確認できるため、何が発生しているかをより的確に把握できます。Windows では、出力先がコンソールウィンドウになるように、--console オプションを付けて mysqld を起動します。
- 問題に関する情報を取得するには、InnoDB モニターを有効にします (セクション14.15「InnoDB モニター」を参照してください)。その問題がパフォーマンスに関するものか、またはサーバーがハングアップしているように見える場合は、InnoDB の内部状態に関する情報を出力するために、標準モニターを有効にするようにしてください。問題がロックに関するものである場合は、ロックモニターを有効にします。問題がテーブルの作成中、またはその他のデータディクショナリの操作中のものである場合は、InnoDB 内部データディクショナリの内容を出力するために、テーブルモニターを有効にします。テーブルスペース情報を表示するには、テーブルスペースモニターを有効にします。

  InnoDB は、次の条件の下で InnoDB 標準モニターの出力を一時的に有効にします。

  - 長いセマフォー待機
  - InnoDB がバッファープール内に空きブロックを見つけることができない
  - ロックヒープまたはアダプティブハッシュインデックスによってバッファープールの 67% を超える領域が占有されている
- テーブルが破損していると思われる場合は、そのテーブルに対して CHECK TABLE を実行します。

14.19.1 InnoDB の I/O に関する問題のトラブルシューティング

InnoDB の I/O に関する問題のトラブルシューティング手順は、その問題がいつ、つまり MySQL サーバーの起動中か、あるいは通常の動作中にファイルシステムレベルの問題で DML または DDL ステートメントが失敗したときのどちらで発生したかによって異なります。

初期化の問題

InnoDB がそのテーブルスペースまたはログファイルを初期化しようとしたときに問題が発生した場合は、InnoDB によって作成されたすべてのファイル、つまりすべての ibdata ファイルおよびすべての ib_logfile ファイルを削除します。すでにいくつかの InnoDB テーブルを作成している場合は、これらのテーブルの対応する .frm ファイルと、複数のテーブルスペースを使用している場合はすべての .ibd ファイルも MySQL データベースディレクトリから削除します。次に、再度 InnoDB データベースを作成してみてください。もっとも簡単なトラブルシューティングとして、何が発生しているかがわかるように、コマンドプロンプトから MySQL サーバーを起動してください。

実行時の問題

InnoDB がファイル操作中にオペレーティングシステムのエラーを出力する場合、通常、この問題には次のいずれかの解決方法があります。

- InnoDB データファイルディレクトリと InnoDB ログディレクトリが存在することを確認します。

- mysqld に、これらのディレクトリ内にファイルを作成するためのアクセス権があることを確認します。
- 指定したオプションで起動できるように、mysqld が正しい my.cnf または my.ini オプションファイルを読み取れることを確認します。
- ディスクがいっぱいでなく、かつどのディスク割り当て制限も超えていないことを確認します。
- サブディレクトリとデータファイルに指定した名前が衝突していないことを確認します。
- innodb_data_home_dir および innodb_data_file_path 値の構文を再確認します。特に、innodb_data_file_path オプション内の MAX 値はすべて強い制限値であるため、その制限を超えると致命的エラーが発生します。

14.19.2 InnoDB のリカバリの強制的な実行

データベースページの破損を調査するために、SELECT ... INTO OUTFILE を使用して、データベースからテーブルをダンプできます。通常は、この方法で取得されたデータのほとんどが完全な状態にあります。重大な破損では、SELECT * FROM tbl_name ステートメントまたは InnoDB のバックグラウンド操作がクラッシュまたは表明したり、場合によっては InnoDB のロールフォワードリカバリがクラッシュしたりすることもあります。このような場合は、テーブルをダンプできるように、innodb_force_recovery オプションを使用して、バックグラウンド操作が実行されないようにして InnoDB ストレージエンジンを強制的に起動させることができます。たとえば、サーバーを再起動する前に、オプションファイルの [mysqld] セクションに次の行を追加できます。

```
[mysqld]
innodb_force_recovery = 1
```

> **警告**
>
> innodb_force_recovery を 0 を超える値に設定するのは、緊急の状況で InnoDB を起動し、テーブルをダンプできるようにする場合だけにしてください。それを行う前に、データベースの再作成が必要になった場合に備えて、データベースのバックアップコピーがあることを確認してください。4 以上の値を指定すると、データファイルが永続的に破損する場合があります。本番サーバーインスタンス上で 4 以上の innodb_force_recovery 設定を使用するのは、使用するデータベースの個別の物理コピーでその設定を正常にテストしたあとだけにしてください。InnoDB のリカバリを強制的に実行する場合は、常に innodb_force_recovery=1 から始め、必要がある場合にのみこの値を 1 ずつ増やすようにしてください。

innodb_force_recovery は、デフォルトでは 0 です (リカバリが強制的に実行されない通常の起動)。innodb_force_recovery の許可される 0 以外の値は 1 から 6 までです。大きい方の値には、小さい方の値の機能が含まれています。たとえば、3 の値には、値 1 と 2 のすべての機能が含まれています。

3 以下の innodb_force_recovery 値を使用してテーブルをダンプできる場合は、破損した個々のページ上の一部のデータしか失われないため、比較的安全です。4 以上の値は、データファイルが永続的に破損する場合があるため、危険であるとみなされます。6 の値は、データベースページが廃止された状態のままになり、それによって B ツリーやその他のデータベース構造にさらに多くの破損が導入される可能性があるため、きわめて危険であるとみなされます。

安全策として、innodb_force_recovery が 0 より大きい場合、InnoDB は INSERT、UPDATE、または DELETE 操作を回避します。MySQL 5.6.15 の時点では、4 以上の innodb_force_recovery 設定は InnoDB を読み取り専用モードにします。

- 1 (SRV_FORCE_IGNORE_CORRUPT)

  破損したページを検出した場合でも、サーバーが動作できるようにします。SELECT * FROM tbl_name での破損したインデックスレコードおよびページの飛び越しを試行します。これが、テーブルのダンプに役立ちます。

- 2 (SRV_FORCE_NO_BACKGROUND)

  マスタースレッドや、すべてのパージスレッドが実行されないようにします。パージ操作中にクラッシュが発生しそうになった場合は、このリカバリの値によって回避されます。

- 3 (SRV_FORCE_NO_TRX_UNDO)

  クラッシュリカバリのあとにトランザクションロールバックを実行しません。

- 4 (SRV_FORCE_NO_IBUF_MERGE)

  挿入バッファーのマージ操作を回避します。その操作によってクラッシュが発生しそうになった場合は、それが回避されます。テーブル統計を計算しません。この値を指定すると、データファイルが永続的に破損する場

合があります。この値を使用したあと、すべてのセカンダリインデックスを削除して再作成するように準備してください。MySQL 5.6.15 の時点では、InnoDB を読み取り専用に設定します。

- 5 (SRV_FORCE_NO_UNDO_LOG_SCAN)

  データベースを起動するときに、Undo ログを参照しません。InnoDB は、未完了のトランザクションでさえコミット済みとして処理します。この値を指定すると、データファイルが永続的に破損する場合があります。MySQL 5.6.15 の時点では、InnoDB を読み取り専用に設定します。

- 6 (SRV_FORCE_NO_LOG_REDO)

  リカバリに関連した Redo ログのロールフォワードを実行しません。この値を指定すると、データファイルが永続的に破損する場合があります。データベースページを廃止された状態のままにし、それによって B ツリーやその他のデータベース構造にさらに多くの破損が発生する可能性があります。MySQL 5.6.15 の時点では、InnoDB を読み取り専用に設定します。

テーブルからの SELECT を実行してテーブルをダンプしたり、3 以下の innodb_force_recovery 値を使用してテーブルの DROP または CREATE を実行したりすることができます。ロールバック時に特定のテーブルでクラッシュが発生することがわかっている場合は、そのテーブルを削除できます。失敗した大量のインポートまたは ALTER TABLE によってロールバックの暴走が発生する場合は、mysqld プロセスを強制終了し、innodb_force_recovery を 3 に設定してロールバックなしでデータベースを起動したあと、ロールバックの暴走の原因になっているテーブルの DROP を実行することができます。

テーブルデータ内の破損のためにテーブルの内容全体をダンプできない場合は、ORDER BY primary_key DESC 句を含むクエリーで、破損した部分のあとにあるテーブルの部分をダンプできる可能性があります。

InnoDB を起動するために innodb_force_recovery を大きな値にする必要がある場合は、複雑なクエリー (WHERE、ORDER BY、またはその他の句を含むクエリー) を失敗させることがある破損したデータ構造が存在する可能性があります。この場合は、基本的な SELECT * FROM t クエリーしか実行できない可能性があります。

14.19.3 InnoDB データディクショナリの操作のトラブルシューティング

テーブル定義に関する情報は、.frm ファイルと InnoDB データディクショナリの両方に格納されます。.frm ファイルをあちこちに移動したり、データディクショナリの操作の最中にサーバーがクラッシュしたりすると、これらの情報のソースに整合性がなくなることがあります。

データディクショナリの破損や一貫性の問題によって InnoDB を起動できない場合は、手動のリカバリに関する情報について、セクション14.19.2「InnoDB のリカバリの強制的な実行」を参照してください。

CREATE TABLE での問題

同期がとれていないデータディクショナリの 1 つの現象として、CREATE TABLE ステートメントが失敗することが挙げられます。これが発生した場合は、サーバーのエラーログを調べてください。そのログに、InnoDB 内部データディクショナリの内部にテーブルがすでに存在することが示されている場合は、InnoDB テーブルスペースファイルの内部に、対応する .frm ファイルのない孤立したテーブルがあります。そのエラーメッセージは次のようになります。

```
InnoDB: Error: table test/parent already exists in InnoDB internal
InnoDB: data dictionary. Have you deleted the .frm file
InnoDB: and not used DROP TABLE? Have you used DROP DATABASE
InnoDB: for InnoDB tables in MySQL version <= 3.23.43?
InnoDB: See the Restrictions section of the InnoDB manual.
InnoDB: You can drop the orphaned table inside InnoDB by
InnoDB: creating an InnoDB table with the same name in another
InnoDB: database and moving the .frm file to the current database.
InnoDB: Then MySQL thinks the table exists, and DROP TABLE will
InnoDB: succeed.
```

この孤立したテーブルは、エラーメッセージに示されている手順に従うことによって削除できます。それでも DROP TABLE を正常に使用できない場合、その問題の原因は mysql クライアントでの名前補完である可能性があります。この問題を回避するには、--skip-auto-rehash オプションを使用して mysql クライアントを起動し、DROP TABLE を再試行します。(名前補完がオンになっていると、mysql はテーブル名のリストを構築しようとしますが、今説明したような問題が存在した場合は失敗します。)

テーブルを開くときの問題

同期がとれていないデータディクショナリの別の現象として、MySQL から .InnoDB ファイルを開くことができないというエラーが出力されます。

```
ERROR 1016: Can't open file: 'child2.InnoDB'. (errno: 1)
```

エラーログに、次のようなメッセージが見つかります。

```
InnoDB: Cannot find table test/child2 from the internal data dictionary
InnoDB: of InnoDB though the .frm file for the table exists. Maybe you
InnoDB: have deleted and recreated InnoDB data files but have forgotten
InnoDB: to delete the corresponding .frm files of InnoDB tables?
```

これは、InnoDB の内部に対応するテーブルのない孤立した .frm ファイルが存在することを示します。この孤立した .frm ファイルは、手動で削除することによって削除できます。

孤立した中間テーブル

ALTER TABLE 操作の最中に MySQL がクラッシュした場合は、孤立した中間テーブルが残ったままになることがあります。中間テーブル名は 「#sql-」 で始まります。データディレクトリ内には #sql-*.ibd ファイルが見つかるほか、付随する #sql-*.frm ファイルも存在する可能性があります。中間テーブルはまた、テーブルモニターの出力にも表示され、InnoDB INFORMATION_SCHEMA テーブルでも参照されます。

孤立した中間テーブルを削除するには、#sql-*.ibd ファイルで定義されているテーブルスキーマに一致するテーブル形式ファイル (.frm ファイル) が必要です (カラムとインデックスが同じである必要があります)。クラッシュが ALTER TABLE 操作中のいつ発生したかに応じて、孤立した #sql-*.ibd ファイルには ALTER の前または ALTER のあとのスキーマ定義が存在する可能性があり、付随する #sql-*.frm ファイル (存在する場合) 内のデータも一致する場合と一致しない場合があります。

孤立した中間テーブルを削除するには、次の手順を実行します。

1. #sql-*.ibd ファイルに ALTER の前または ALTER のあとのどちらのスキーマ定義が存在するかを判定します。中間テーブルのカラムとインデックスは、テーブルモニターを使用して、または InnoDB INFORMATION_SCHEMA テーブルをクエリーすることによって表示できます。INNODB_SYS_TABLES は、中間テーブルの TABLE_ID を提供します。これを使用すると、INNODB_SYS_COLUMNS および INNODB_SYS_INDEXES からカラムとインデックスの情報を取得できます。

2. #sql-*.ibd ファイルに ALTER の前または ALTER のあとのどちらのスキーマ定義が存在するかを判定したら、一致する #sql-*.frm ファイルを別のデータベースディレクトリ内に作成します。たとえば、中間テーブルに ALTER のあとのスキーマ定義が存在する場合は、変更されたスキーマ定義に一致する .frm ファイルを作成します。

   ```
   mysql> CREATE TABLE tmp LIKE employees.salaries; ALTER TABLE tmp DROP COLUMN to_date;
   Query OK, 0 rows affected (0.02 sec)

   Query OK, 0 rows affected (0.06 sec)
   Records: 0  Duplicates: 0  Warnings: 0
   ```

3. .frm ファイルを孤立したテーブルが存在するデータベースディレクトリにコピーし、その名前を #sql-*.ibd ファイルの名前に一致するように変更します。

   ```
   shell> cp tmp.frm employees/#sql-ib87.frm
   ```

4. テーブルの名前にプリフィクス #mysql50# を付け、テーブル名を逆引用符で囲んだ DROP TABLE ステートメントを発行することによって、中間テーブルを削除します。例:

   ```
   mysql> DROP TABLE `#mysql50##sql-ib87`;
   Query OK, 0 rows affected (0.01 sec)
   ```

   #mysql50# のプリフィクスは、MySQL 5.1 で導入された file name safe encoding を無視するよう MySQL に指示します。テーブル名を逆引用符で囲むことは、「#」などの特殊文字を含むテーブル名に対して SQL ステートメントを実行するために必要です。

5. 残りの #sql-*.frm ファイルが存在する場合は、それを削除します。MySQL が「不明なテーブル」エラーをレポートしますが、これは無視できます。

   ```
   mysql> DROP TABLE `#mysql50##sql-36ab_2`;
   ERROR 1051 (42S02): Unknown table 'employees.#mysql50##sql-36ab_2'
   ```

テーブルスペースが見つからない問題

innodb_file_per_table が有効になっていると、.frm または .ibd ファイル (あるいはその両方) が見つからない場合に、次のメッセージが生成されることがあります。

```
InnoDB: in InnoDB data dictionary has tablespace id N,
```

```
InnoDB: but tablespace with that id or name does not exist. Have
InnoDB: you deleted or moved .ibd files?
InnoDB: This may also be a table created with CREATE TEMPORARY TABLE
InnoDB: whose .ibd and .frm files MySQL automatically removed, but the
InnoDB: table still exists in the InnoDB internal data dictionary.
```

これが発生した場合は、問題を解決するために次の手順を試してください。

1. 一致する .frm ファイルをどこかほかのデータベースディレクトリ内に作成し、それを孤立したテーブルが存在するデータベースディレクトリにコピーします。
2. 元のテーブルに対して DROP TABLE を発行します。それにより、テーブルが正常に削除され、InnoDB によって、.ibd ファイルが見つからなかったことを示す警告がエラーログに出力されるはずです。

14.19.4 InnoDB のエラー処理

InnoDB でのエラー処理は、SQL 標準で指定されているものと必ずしも同じではありません。この標準によると、SQL ステートメント中にエラーが発生した場合は必ず、そのステートメントのロールバックを実行するべきです。InnoDB は場合によって、ステートメントの一部のみ、またはトランザクション全体をロールバックします。次の各項目は、InnoDB がエラー処理をどのように実行するかについて説明しています。

- テーブルスペース内のファイル領域が不足した場合は、MySQL の Table is full エラーが発生し、InnoDB は SQL ステートメントをロールバックします。
- トランザクションデッドロックが発生すると、InnoDB はトランザクション全体をロールバックします。これが発生した場合は、トランザクション全体を再試行します。

  ロック待機のタイムアウトが発生すると、InnoDB は、ロックの待機中にタイムアウトが発生した 1 つのステートメントのみをロールバックします。(トランザクション全体がロールバックされるようにするには、--innodb_rollback_on_timeout オプションを使用してサーバーを起動します。)現在の動作を使用している場合はそのステートメントを、--innodb_rollback_on_timeout を使用している場合はトランザクション全体を再試行します。

  デッドロックとロック待機のタイムアウトはどちらもビジー状態のサーバーでは通常のことであり、アプリケーションはそれらが発生する可能性を認識し、発生した場合は再試行によって処理する必要があります。トランザクション中の最初のデータ変更からコミットまでの間に行う作業をできるだけ少なくして、ロックが可能性のある最短の時間、可能性のある最少の行数に対して保持されるようにすることにより、それらが発生する可能性を少なくすることができます。場合によっては、異なるトランザクション間での作業の分割が実際的で、かつ役立つことがあります。

  デッドロックまたはロック待機のタイムアウトのためにトランザクションロールバックが発生すると、そのトランザクション内のステートメントの効果が取り消されます。ただし、トランザクション開始ステートメントが START TRANSACTION または BEGIN ステートメントであった場合、そのステートメントはロールバックによって取り消されません。それ以上の SQL ステートメントは、COMMIT、ROLLBACK、または暗黙的なコミットを発生させる何らかの SQL ステートメントが現れるまで、そのトランザクションの一部になります。
- ステートメントで IGNORE オプションを指定していない場合、重複キーエラーは SQL ステートメントをロールバックします。
- row too long error は、SQL ステートメントをロールバックします。
- その他のエラーはほとんど (InnoDB ストレージエンジンレベルの上にある) コードの MySQL レイヤーによって検出され、対応する SQL ステートメントをロールバックします。1 つの SQL ステートメントのロールバックでは、ロックは解放されません。

暗黙的なロールバック中や、明示的な ROLLBACK SQL ステートメントの実行中に、SHOW PROCESSLIST は、関連する接続の State カラムに Rolling back を表示します。

14.19.5 InnoDB のエラーコード

次に示すのは、発生する可能性のある InnoDB 固有の一般的なエラーのリスト (ただし、すべてが含まれているわけではありません)、および各エラーが発生する理由と問題の解決方法に関する情報です。

- 1005 (ER_CANT_CREATE_TABLE)

  テーブルを作成できません。エラーメッセージがエラー 150 を示している場合は、外部キー制約が正しく形成されていなかったためにテーブルの作成が失敗しました。エラーメッセージがエラー -1 を示している場合は、

テーブルに内部の InnoDB テーブルの名前に一致したカラム名が含まれているためにテーブルの作成が失敗した可能性があります。

- 1016 (ER_CANT_OPEN_FILE)

  InnoDB テーブルの .frm ファイルが存在するにもかかわらず、このテーブルが InnoDB データファイルに見つかりません。セクション14.19.3「InnoDB データディクショナリの操作のトラブルシューティング」を参照してください。

- 1114 (ER_RECORD_FILE_FULL)

  InnoDB でテーブルスペース内の空き領域が不足しています。新しいデータファイルを追加するために、テーブルスペースを再構成してください。

- 1205 (ER_LOCK_WAIT_TIMEOUT)

  ロック待機のタイムアウトの期限が切れました。トランザクション全体ではなく、待機時間の長すぎたステートメントがロールバックされました。innodb_lock_wait_timeout 構成オプションの値は、SQL ステートメントがほかのトランザクションの完了をより長い時間待機するようにする場合は増やし、また長時間実行されるトランザクションが多すぎるためにロックの問題が発生し、さらにビジー状態のシステム上の並列性が低下している場合は減らすことができます。

- 1206 (ER_LOCK_TABLE_FULL)

  ロックの総数が、InnoDB でロックの管理専用に使用されているメモリーの量を超えています。このエラーを回避するには、innodb_buffer_pool_size の値を増やします。個々のアプリケーション内では、大きな操作をより小さな操作に分割することが回避方法になる場合があります。たとえば、このエラーが大きな INSERT で発生する場合は、複数のより小さな INSERT 操作を実行します。

- 1213 (ER_LOCK_DEADLOCK)

  トランザクションでデッドロックが発生し、アプリケーションで修正措置をとることができるように自動的にロールバックされました。このエラーからリカバリするには、このトランザクション内のすべての操作を再度実行します。デッドロックは、ロックに対する要求がトランザクション間で整合性のない順序で到着した場合に発生します。ロールバックされたトランザクションがそのすべてのロックを解放したため、ほかのトランザクションは要求したすべてのロックを取得できるようになりました。そのため、ロールバックされたトランザクションを再実行すると、そのトランザクションがほかのトランザクションの完了を待機しなければならないことがありますが、通常デッドロックは再発しません。デッドロックが頻繁に発生する場合は、ロック操作のシーケンス (LOCK TABLES、SELECT ... FOR UPDATE など) を、問題が発生する異なるトランザクションまたはアプリケーション間で整合性があるようにしてください。詳細は、セクション14.2.11「デッドロックの対処方法」を参照してください。

- 1216 (ER_NO_REFERENCED_ROW)

  行を追加しようとしていますが、親の行がないため、外部キー制約に違反します。最初に親の行を追加してください。

- 1217 (ER_ROW_IS_REFERENCED)

  子を持つ親の行を削除しようとしているため、外部キー制約に違反します。最初に子を削除してください。

- ERROR 1553 (HY000): インデックス 'fooIdx' をドロップできません。外部キー制約で必要とされています

  このエラーメッセージは、特定の参照制約を適用できる最後のインデックスを削除しようとしたときにレポートされます。

  DML ステートメントの最適なパフォーマンスを得るために、InnoDB では、親テーブルでの UPDATE および DELETE 操作で、対応する行が子テーブル内に存在するかどうかを容易にチェックできるように、インデックスは外部キーカラムに存在する必要があります。CREATE TABLE、CREATE INDEX、および ALTER TABLE ステートメントの副作用として、MySQL は、このようなインデックスを必要に応じて自動的に作成または削除します。

  インデックスを削除すると、InnoDB は、そのインデックスが外部キー制約のチェックに使用されていないかどうかをチェックします。同じ制約を適用するために使用できる別のインデックスが存在する場合は、引き続きインデックスを削除できます。InnoDB は、特定の参照制約を適用できる最後のインデックスを削除できないようにしています。

14.19.6 オペレーティングシステムのエラーコード

オペレーティングシステムのエラー番号に対するエラーメッセージを出力するには、次のいずれかのオプションを使用できます。perror プログラムは、MySQL 配布に付属しています。

```
$ perror 123
$ perl -MPOSIX -le 'print strerror 123'
$ python -c 'import os; print os.strerror(123)'
```

- Linux システムのエラーコード

  次の表は、Linux システムのエラーコードの部分的なリストを示しています。

番号	マクロ	説明
1	EPERM	操作が許可されていません
2	ENOENT	そのようなファイルまたはディレクトリはありません
3	ESRCH	そのようなプロセスはありません
4	EINTR	システムコールが中断されました
5	EIO	I/O エラーです
6	ENXIO	そのようなデバイスまたはアドレスはありません
7	E2BIG	引数リストが長すぎます
8	ENOEXEC	exec フォーマットエラー
9	EBADF	不正なファイル番号です
10	ECHILD	子プロセスがありません
11	EAGAIN	再試行してください
12	ENOMEM	メモリー不足です
13	EACCES	アクセス権が拒否されました
14	EFAULT	不正なアドレスです
15	ENOTBLK	ブロック型デバイスが必要です
16	EBUSY	デバイスまたはリソースがビジー状態です
17	EEXIST	ファイルが存在します
18	EXDEV	デバイスにまたがるリンクです
19	ENODEV	そのようなデバイスはありません
20	ENOTDIR	ディレクトリではありません
21	EISDIR	ディレクトリです
22	EINVAL	無効な引数です
23	ENFILE	ファイルテーブルがオーバーフローしました
24	EMFILE	開かれたファイルが多すぎます
25	ENOTTY	デバイスへの ioctl が正しくありません
26	ETXTBSY	テキストファイルがビジー状態です
27	EFBIG	ファイルが大きすぎます
28	ENOSPC	デバイスに領域が残されていません
29	ESPIPE	ファイルディスクリプタでシークが許可されていません
30	EROFS	読み取り専用ファイルシステムです
31	EMLINK	リンクが多すぎます

- Windows システムのエラーコード

  次の表は、Windows システムのいくつかの一般的なエラーコードのリストを示しています。完全なリストについては、Microsoft の Web サイトを参照してください。

番号	マクロ	説明
1	ERROR_INVALID_FUNCTION	ファンクションが間違っています。

番号	マクロ	説明
2	ERROR_FILE_NOT_FOUND	指定されたファイルが見つかりません。
3	ERROR_PATH_NOT_FOUND	指定されたパスを見つけられませんでした。
4	ERROR_TOO_MANY_OPEN_FILES	ファイルを開くことができません。
5	ERROR_ACCESS_DENIED	アクセスは拒否されました。
6	ERROR_INVALID_HANDLE	ハンドルが無効です。
7	ERROR_ARENA_TRASHED	記憶域制御ブロックが壊れています。
8	ERROR_NOT_ENOUGH_MEMORY	このコマンドを実行するための十分な記憶域がありません。
9	ERROR_INVALID_BLOCK	記憶域制御ブロックのアドレスが無効です。
10	ERROR_BAD_ENVIRONMENT	環境が間違っています。
11	ERROR_BAD_FORMAT	間違ったフォーマットのプログラムを読み込もうとしました。
12	ERROR_INVALID_ACCESS	アクセス コードが無効です。
13	ERROR_INVALID_DATA	データが無効です。
14	ERROR_OUTOFMEMORY	この操作を完了するための十分な記憶域がありません。
15	ERROR_INVALID_DRIVE	指定されたドライブが見つかりません。
16	ERROR_CURRENT_DIRECTORY	ディレクトリを削除できません。
17	ERROR_NOT_SAME_DEVICE	ファイルを別のディスク ドライブに移動できません。
18	ERROR_NO_MORE_FILES	これ以上ファイルはありません。
19	ERROR_WRITE_PROTECT	このメディアは書き込み禁止になっています。
20	ERROR_BAD_UNIT	指定されたデバイスが見つかりません。
21	ERROR_NOT_READY	デバイスの準備ができていません。
22	ERROR_BAD_COMMAND	デバイスがコマンドを認識できません。
23	ERROR_CRC	データ エラー (巡回冗長検査 (CRC) エラー) です。
24	ERROR_BAD_LENGTH	プログラムはコマンドを発行しましたが、コマンドの長さが間違っています。
25	ERROR_SEEK	指定されたディスクの領域またはトラックが見つかりません。
26	ERROR_NOT_DOS_DISK	指定されたディスクまたはフロッピー ディスクにアクセスできません。
27	ERROR_SECTOR_NOT_FOUND	要求されたセクターが見つかりません。
28	ERROR_OUT_OF_PAPER	プリンターが用紙切れです。
29	ERROR_WRITE_FAULT	指定されたデバイスに書き込めません。
30	ERROR_READ_FAULT	指定されたデバイスから読み取れません。
31	ERROR_GEN_FAILURE	システムに接続されたデバイスが機能していません。
32	ERROR_SHARING_VIOLATION	プロセスはファイルにアクセスできません。別のプロセスが使用中です。
33	ERROR_LOCK_VIOLATION	プロセスはファイルにアクセスできません。別のプロセスがファイルの一部をロックしています。
34	ERROR_WRONG_DISK	間違ったフロッピー ディスクがドライブに挿入されています。%2 (ボリューム シリアル番号: %3) をドライブ %1 に挿入してください。
36	ERROR_SHARING_BUFFER_EXCEEDED	開かれている共有ファイルが多すぎます。
38	ERROR_HANDLE_EOF	ファイルの終わりに達しました。
39	ERROR_HANDLE_DISK_FULL	ディスクがいっぱいです。
87	ERROR_INVALID_PARAMETER	パラメーターが正しくありません。
112	ERROR_DISK_FULL	ディスクがいっぱいです。
123	ERROR_INVALID_NAME	ファイル名、ディレクトリ名、またはボリューム ラベルの構文が正しくありません。
1450	ERROR_NO_SYSTEM_RESOURCES	システム リソースが不足しているため、要求されたサービスを完了できません。

第 15 章 代替ストレージエンジン

目次

15.1 ストレージエンジンの設定 1763
15.2 MyISAM ストレージエンジン 1763
15.2.1 MyISAM 起動オプション 1766
15.2.2 キーに必要な容量 1767
15.2.3 MyISAM テーブルのストレージフォーマット 1767
15.2.4 MyISAM テーブルの問題点 1769
15.3 MEMORY ストレージエンジン 1771
15.4 CSV ストレージエンジン 1774
15.4.1 CSV テーブルの修復と確認 1775
15.4.2 CSV の制限 1776
15.5 ARCHIVE ストレージエンジン 1776
15.6 BLACKHOLE ストレージエンジン 1778
15.7 MERGE ストレージエンジン 1780
15.7.1 MERGE テーブルの長所と短所 1782
15.7.2 MERGE テーブルの問題点 1783
15.8 FEDERATED ストレージエンジン 1784
15.8.1 FEDERATED ストレージエンジンの概要 1784
15.8.2 FEDERATED テーブルの作成方法 1785
15.8.3 FEDERATED ストレージエンジンの注記とヒント 1787
15.8.4 FEDERATED ストレージエンジンのリソース 1789
15.9 EXAMPLE ストレージエンジン 1789
15.10 ほかのストレージエンジン 1789
15.11 MySQL ストレージエンジンアーキテクチャーの概要 1789
15.11.1 プラガブルストレージエンジンのアーキテクチャー 1790
15.11.2 共通データベースサーバーレイヤー 1790

ストレージエンジンは、さまざまなテーブル型に対する SQL 操作を処理する MySQL コンポーネントです。InnoDB はデフォルトでもっとも汎用のストレージエンジンであり、Oracle は、特別なユースケースを除くテーブルについては、このエンジンの使用を推奨します。(デフォルトでは、MySQL 5.6 の CREATE TABLE ステートメントは InnoDB テーブルを作成します。)

MySQL Server は、ストレージエンジンが、動作中の MySQL サーバーにロードされたり、MySQL サーバーからアンロードされたりできる、プラガブルストレージエンジンアーキテクチャーを採用しています。

ご使用のサーバーがサポートするストレージエンジンを調べるには、SHOW ENGINES ステートメントを使用します。サポートカラムの値は、エンジンを使用できるかどうかを示します。YES、NO、または DEFAULT の値は、エンジンが「使用可能」、「使用可能でない」、または「デフォルトのストレージエンジンとして使用可能であり、現在設定されている」を表しています。

```
mysql> SHOW ENGINES\G
*************************** 1. row ***************************
      Engine: PERFORMANCE_SCHEMA
     Support: YES
     Comment: Performance Schema
Transactions: NO
          XA: NO
  Savepoints: NO
*************************** 2. row ***************************
      Engine: InnoDB
     Support: DEFAULT
     Comment: Supports transactions, row-level locking, and foreign keys
Transactions: YES
          XA: YES
  Savepoints: YES
*************************** 3. row ***************************
      Engine: MRG_MYISAM
     Support: YES
     Comment: Collection of identical MyISAM tables
Transactions: NO
          XA: NO
```

```
  Savepoints: NO
*************************** 4. row ***************************
      Engine: BLACKHOLE
     Support: YES
     Comment: /dev/null storage engine (anything you write to it disappears)
Transactions: NO
          XA: NO
  Savepoints: NO
*************************** 5. row ***************************
      Engine: MyISAM
     Support: YES
     Comment: MyISAM storage engine
Transactions: NO
          XA: NO
  Savepoints: NO
...
```

この章では、特別な目的の MySQL ストレージエンジンのユースケースについて説明します。第14章「InnoDB ストレージエンジン」および 第18章「MySQL Cluster NDB 7.3 および MySQL Cluster NDB 7.4」で説明するデフォルトの InnoDB ストレージエンジンまたは NDB ストレージエンジンについては説明しません。上級ユーザーのために、この章ではプラガブルストレージエンジンアーキテクチャーについても説明します (セクション 15.11「MySQL ストレージエンジンアーキテクチャーの概要」を参照してください)。

商用 MySQL Server バイナリで提供されるストレージエンジンのサポートについては、MySQL Web サイトの MySQL Enterprise Server 5.6 を参照してください。使用可能なストレージエンジンは、使用中の Enterprise Server のエディションによって異なる場合があります。

MySQL ストレージエンジンに関するよくある質問の回答については、セクションA.2「MySQL 5.6 FAQ: ストレージエンジン」を参照してください。

MySQL 5.6 がサポートするストレージエンジン

- InnoDB: MySQL 5.6 のデフォルトのストレージエンジン。InnoDB はトランザクションセーフな (ACID に準拠した) MySQL 用のストレージエンジンであり、ユーザーデータを保護するためのコミット、ロールバック、およびクラッシュリカバリ機能を備えています。InnoDB の行レベルロック (より粒度の粗いロックへのエスカレーションは行わない) と Oracle スタイルの一貫性非ロック読み取りにより、マルチユーザーの並列性とパフォーマンスが向上します。InnoDB では、主キーに基づいた一般的なクエリーの入出力を低減するため、クラスタ化されたインデックス内にユーザーデータが格納されます。InnoDB ではデータの整合性を維持できるように、FOREIGN KEY 参照整合性制約もサポートされています。InnoDB の詳細については、第14章「InnoDB ストレージエンジン」を参照してください。

- MyISAM: これらのテーブルのフットプリントは小さくなります。テーブルレベルのロック では、読み取り/書き込みの作業負荷でのパフォーマンスが抑えられるため、Web およびデータウェアハウス構成の読み取り専用または読み取りが大半の作業負荷の場合に使用されるのが一般的です。

- メモリー: すべてのデータを RAM に格納します (重要でないデータの短時間での検索が必要な環境で高速にアクセスするため)。このエンジンは以前は HEAP エンジンとして知られていました。このユースケースは減少しています。バッファープールのメモリー領域を持つ InnoDB は、ほとんどのデータまたはすべてのデータをメモリーに保持する汎用的で永続的な方法を提供し、NDBCLUSTER は大規模な分散データセットでキー値の高速な検索ができます。

- CSV: このテーブルは、カンマ区切り値を持つ実際のテキストファイルです。CSV テーブルにより、CSV フォーマットでデータをインポートしたりダンプしたりして、同じフォーマットを読み込んだり書き込んだりするスクリプトおよびアプリケーションとデータを交換できます。CSV テーブルはインデックス化されないため、通常の操作時はデータを InnoDB テーブルに保持し、インポートまたはエクスポートの段階でのみ CSV テーブルを使用するのが一般的です。

- アーカイブ: これらのインデックス化されていないコンパクトなテーブルは、ほとんど参照されない大量の履歴情報、アーカイブされた情報、またはセキュリティー監査情報を格納したり、検索したりするためのテーブルです。

- Blackhole: Blackhole ストレージエンジンはデータを受け付けますが、Unix /dev/null デバイスと同じように、格納しません。クエリーは常に空のセットを返します。これらのテーブルは、DML ステートメントはスレーブサーバーに送られますが、マスターサーバーはデータの独自のコピーを保持しないレプリケーション構成で使用できます。

- NDB (NDBCLUSTER としても知られています) - クラスタ化されたこのデータベースエンジンは、稼働時間と可用性の程度ができるだけ高くなくてはいけないアプリケーションに特に適しています。

> **注記**
>
> NDB ストレージエンジンは、標準の MySQL 5.6 リリースではサポートされていません。現在サポートされている MySQL Cluster リリースには、MySQL 5.1 がベースの MySQL Cluster NDB 7.1、MySQL 5.5 がベースの MySQL Cluster NDB 7.2、MySQL 5.6 がベースの MySQL Cluster NDB 7.3 が含まれています。MySQL Server がベースである間、これらのリリースには NDB のサポートも含まれます。MySQL 5.6 もベースにした現在開発中の MySQL Cluster NDB 7.4 が、Developer Milestone リリースでも使用できるようになりました。

- マージ: MySQL DBA または開発者は、一連のまったく同じ MyISAM テーブルを論理的にグループ分けして、それらを 1 つのオブジェクトとして参照します。データウェアハウスなどの VLDB 環境に適しています。
- Federated: 多くの物理サーバーから 1 つの論理サーバーを作成するために別々の MySQL サーバーをリンクする機能を提供します。分散またはデータマート環境に非常に適しています。
- 例: このエンジンは、新しいストレージエンジンの書き込みを開始する方法を示す MySQL ソースコードの例として機能します。これは、主に開発者が対象です。ストレージエンジンは何もしない 「stub」 です。このエンジンでテーブルを作成できますが、それらにデータを格納したり、それらからデータを取り出したりすることはできません。

サーバー全体またはスキーマ全体に同じストレージエンジンを使用するという制限はありません。いずれのテーブルにもストレージエンジンを指定できます。たとえばアプリケーションでは、InnoDB テーブルを使用している場合がほとんどであり、データをスプレッドシートにエクスポートするための CSV テーブルを 1 つ、テンポラリワークスペース用に MEMORY テーブルをいくつか持っています。

ストレージエンジンの選択

MySQL が提供するさまざまなストレージエンジンは、異なるユースケースで使用されることを想定して設計されています。次の表は、MySQL が提供するいくつかのストレージエンジンの概要について示しています。

表 15.1 ストレージエンジンの機能サマリー

機能	MyISAM	メモリー	InnoDB	アーカイブ	NDB
B ツリーインデックス	はい	はい	はい	いいえ	いいえ
MVCC	いいえ	いいえ	はい	いいえ	いいえ
T ツリーインデックス	いいえ	いいえ	いいえ	いいえ	はい
インデックスキャッシュ	はい	N/A	はい	いいえ	はい
クエリーキャッシュのサポート	はい	はい	はい	はい	はい
クラスタデータベースのサポート	いいえ	いいえ	いいえ	いいえ	はい
クラスタ化されたインデックス	いいえ	いいえ	はい	いいえ	いいえ
ストレージの制限	256T バイト	RAM	64T バイト	なし	384E バイト
データキャッシュ	いいえ	N/A	はい	いいえ	はい

機能	MyISAM	メモリー	InnoDB	アーカイブ	NDB
データディクショナリ向け更新統計	はい	はい	はい	はい	はい
トランザクション	いいえ	いいえ	はい	いいえ	はい
ハッシュインデックス	いいえ	はい	いいえ (note 1)	いいえ	はい
バックアップ/ポイントインタイムリカバリ (note 2)	はい	はい	はい	はい	はい
レプリケーションのサポート (note 2)	はい	はい	はい	はい	はい
ロック粒度	テーブル	テーブル	行	テーブル	行
全文検索インデックス	はい	いいえ	はい (note 3)	いいえ	いいえ
圧縮データ	はい (note 4)	いいえ	はい (note 5)	はい	いいえ
地理空間インデックスのサポート	はい	いいえ	はい (note 6)	いいえ	いいえ
地理空間データ型のサポート	はい	いいえ	はい	はい	はい
外部キーのサポート	いいえ	いいえ	はい	いいえ	いいえ
暗号化データ (note 7)	はい	はい	はい	はい	はい

Notes:

1. InnoDB は、アダプティブハッシュインデックス機能に対して、内部的にハッシュインデックスを利用します。

2. ストレージエンジン内ではなくサーバー内で実装されています。

3. InnoDB の FULLTEXT インデックスサポートは MySQL 5.6.4 以降で使用できます。

4. 圧縮された MyISAM テーブルがサポートされているのは、圧縮行フォーマットを使用している場合だけです。MyISAM で圧縮行フォーマットを使用するテーブルは、読み取り専用です。

5. 圧縮された InnoDB テーブルは InnoDB Barracuda ファイルフォーマットを必要とします。

6. InnoDB の地理空間インデックスサポートは MySQL 5.7.5 以降で使用できます。

7. ストレージエンジン内ではなくサーバー内で (暗号化関数を使って) 実装されています。

15.1 ストレージエンジンの設定

新しいテーブルを作成するときに、ENGINE テーブルオプションを CREATE TABLE ステートメントに加えることによって、どのストレージエンジンを利用するかを指定できます。

```
-- ENGINE=INNODB not needed unless you have set a different
-- default storage engine.
CREATE TABLE t1 (i INT) ENGINE = INNODB;
-- Simple table definitions can be switched from one to another.
CREATE TABLE t2 (i INT) ENGINE = CSV;
CREATE TABLE t3 (i INT) ENGINE = MEMORY;
```

ENGINE オプションを省略した場合、デフォルトのストレージエンジンが使用されます。デフォルトのエンジンは MySQL 5.6 の InnoDB です。デフォルトのエンジンを指定するには、--default-storage-engine サーバースタートアップオプションを使用するか、my.cnf 構成ファイルにある default-storage-engine オプションを設定するかします。

現在のセッションにデフォルトのストレージエンジンを設定するには、default_storage_engine 変数を設定します。

```
SET default_storage_engine=NDBCLUSTER;
```

MySQL 5.6.3 以降では、CREATE TEMPORARY TABLE で作成された TEMPORARY テーブルのストレージエンジンは、スタートアップ時または実行時のいずれかに default_tmp_storage_engine を設定することで、永続的なテーブルのエンジンから個別に設定できます。MySQL 5.6.3 より前は、default_storage_engine で永続および TEMPORARY の両方のテーブルのエンジンを設定します。

テーブルを別のストレージエンジンに変換するには、新しいエンジンを指定する ALTER TABLE ステートメントを使用します。

```
ALTER TABLE t ENGINE = InnoDB;
```

セクション13.1.17「CREATE TABLE 構文」および セクション13.1.7「ALTER TABLE 構文」を参照してください。

コンパイルされていないストレージエンジン、またはコンパイルされているが無効化されたストレージエンジンを使用する場合、MySQL はその代わりに、デフォルトのストレージエンジンを使用してテーブルを作成します。たとえばレプリケーションのセットアップで、マスターサーバーは、最大限の安全性を得るために InnoDB テーブルを使用し、スレーブサーバーは持続性と一貫性を犠牲にして速度を得るために、代替ストレージエンジンを使用する場合があります。

デフォルトでは、CREATE TABLE または ALTER TABLE がデフォルトのストレージエンジンを使用できない場合は、常に警告が生成されます。目的のエンジンが使用できない場合に、混乱を起こす意図しない動作をしないようにするには、NO_ENGINE_SUBSTITUTION SQL モードを有効にします。目的のエンジンが使用できない場合、この設定によって、警告の代わりにエラーが起こり、テーブルが作成されたり変更されたりしません。セクション5.1.7「サーバー SQL モード」を参照してください。

新しいテーブルの場合、MySQL はテーブルとカラムの定義を保持するために .frm ファイルを必ず作成します。テーブルのインデックスとデータは、ストレージエンジンによっても異なりますが、1 つまたは複数の別のファイルに格納してもかまいません。サーバーがストレージエンジンレベルの上位に .frm ファイルを作成します。個々のストレージエンジンは、それらが管理するテーブルに必要なファイルをさらに作成します。テーブル名に特殊文字が含まれている場合は、セクション9.2.3「識別子とファイル名のマッピング」で説明されているように、その文字のエンコードされたバージョンがテーブルファイルの名前に含まれます。

15.2 MyISAM ストレージエンジン

MyISAM は古い (そしてすでに使用できない) ISAM ストレージエンジンに基づいていますが、多くの役に立つ拡張機能を持っています。

表 15.2 MyISAM ストレージエンジンの機能

機能	Support
B ツリーインデックス	はい
MVCC	いいえ

機能	Support
T ツリーインデックス	いいえ
インデックスキャッシュ	はい
クエリーキャッシュのサポート	はい
クラスタデータベースのサポート	いいえ
クラスタ化されたインデックス	いいえ
ストレージの制限	256T バイト
データキャッシュ	いいえ
データディクショナリ向け更新統計	はい
トランザクション	いいえ
ハッシュインデックス	いいえ
バックアップ/ポイントインタイムリカバリ (ストレージエンジン内ではなくサーバー内で実装されています。)	はい
レプリケーションのサポート (ストレージエンジン内ではなくサーバー内で実装されています。)	はい
ロック粒度	テーブル
全文検索インデックス	はい
圧縮データ	はい (圧縮された MyISAM テーブルがサポートされているのは、圧縮行フォーマットを使用している場合だけです。MyISAM で圧縮行フォーマットを使用するテーブルは、読み取り専用です。)
地理空間インデックスのサポート	はい
地理空間データ型のサポート	はい
外部キーのサポート	いいえ
暗号化データ (ストレージエンジン内ではなくサーバー内で (暗号化関数を使って) 実装されています。)	はい

各 MyISAM テーブルはディスク上に 3 つのファイルとして格納されます。そのファイル名はテーブル名で始まり、ファイルタイプを示す拡張子が付きます。.frm ファイルはテーブルフォーマットを格納します。データファイルには .MYD (MYData) 拡張子が付きます。インデックスファイルには .MYI (MYIndex) 拡張子が付きます。

MyISAM テーブルが必要であることを明示的に指定するには、ENGINE テーブルオプションで指定します。

```
CREATE TABLE t (i INT) ENGINE = MYISAM;
```

MySQL 5.6 では通常、InnoDB がデフォルトエンジンであるため、ENGINE を使用して MyISAM ストレージエンジンを指定する必要があります。

mysqlcheck クライアントか myisamchk ユーティリティーで MyISAM テーブルをチェックしたり修正したりできます。容量を節約するために myisampackを使って MyISAM テーブルを圧縮することもできます。セクション4.5.3「mysqlcheck — テーブル保守プログラム」、セクション4.6.3「myisamchk — MyISAM テーブルメンテナンスユーティリティー」、セクション4.6.5「myisampack — 圧縮された読み取り専用の MyISAM テーブルの生成」を参照してください。

MyISAM テーブルには次のような特徴があります。

- すべてのデータ値は、下位バイトから順に格納されます。これにより、データマシンとオペレーティングシステムは依存しなくなります。バイナリポータビリティーのための唯一の要件は、2 の補数の符号付き整数と IEEE 浮動小数点フォーマットを使用することです。これらの要件は主流のマシンで幅広く使用されています。バイナリポータビリティーは、組み込みシステムには適用されない可能性があります。特別のプロセッサを使用している場合があるためです。

  下位バイトから順にデータを格納するため、大きな速度低下はありません。通常、テーブル行のバイトは整列しておらず、順番に未整列のバイトを読み込む処理は逆の順番に読み込む処理より時間がかかりません。また、カラム値をフェッチするサーバーのコードは、ほかのコードに比べて速度は重視されません。

- インデックスを効率良く圧縮ができるため、すべての数値キー値は上位バイトから順に格納されます。

- 大きなファイル (最大 63 ビットのファイル長) は、大きなファイルをサポートするファイルシステムとオペレーティングシステムでサポートされます。
- MyISAM テーブルの行数は、$(2^{32})^2$ (1.844E+19) の制限があります。
- 1 つの MyISAM テーブルの最大インデックス数は 64 です。

  1 つのインデックスの最大カラム数は 16 です。
- 最大キー長は 1000 バイトです。これは、ソースを変更して再コンパイルしても変えることができます。キーが 250 バイトより長いと、キーのブロックサイズはデフォルト値の 1024 バイトより大きい値が使用されます。
- ソートされた順番で行が挿入されたとき (AUTO_INCREMENT カラムを使用しているときと同様に)、上位のノードが 1 つのキーだけを含むように、インデックスツリーが分割されます。これにより、インデックスツリーの領域の利用率が向上します。
- テーブルごとに 1 つの AUTO_INCREMENT カラムの内部処理がサポートされます。MyISAM は INSERT 操作と UPDATE 操作でこのカラムを自動的に更新します。これにより、AUTO_INCREMENT カラムは速くなります (少なくとも 10 %)。シーケンスの一番上の値は、削除されると、再利用されません。(AUTO_INCREMENT カラムがマルチカラムインデックスの最後のカラムとして定義された場合、シーケンスの最上部から削除された値が再利用されます。)AUTO_INCREMENT 値は ALTER TABLE や myisamchk でリセットできます。
- 動的サイズの行は、削除を更新および挿入と併用すると、フラグメント化がかなり減少します。これは、削除された隣接ブロックを自動的に結合し、次のブロックが削除されたときにブロックを拡張することで行われます。
- MyISAM は同時挿入をサポートしています。テーブルのデータファイルの途中に空きブロックがなければ、ほかのスレッドがテーブルから読み取るのと同時に新しい行をそれに INSERT できます。行を削除した結果として、または動的長の行を現在の内容より多くのデータで更新した結果として、空きブロックが発生する可能性があります。すべての空きブロックが完全に使用されると (埋まると)、その後の挿入はふたたび並列になります。セクション8.10.3「同時挿入」を参照してください。
- データファイルとインデックスファイルを異なる物理デバイス上の異なるディレクトリに置き、DATA DIRECTORY および INDEX DIRECTORY テーブルオプションを CREATE TABLE に付けて速度を上げることができます。セクション13.1.17「CREATE TABLE 構文」を参照してください。
- BLOB と TEXT カラムはインデックスを付けることができます。
- インデックスを付けたカラムでは NULL 値が許可されます。これには、キー当たり 0 - 1 バイトが必要です。
- 文字カラムごとに異なる文字セットを持つことができます。セクション10.1「文字セットのサポート」を参照してください。
- MyISAM インデックスファイルの中に、テーブルが正しく閉じられたかどうかを表すフラグがあります。mysqld が --myisam-recover-options オプションで起動された場合、MyISAM テーブルはいつ開かれたかが自動的に確認され、正しく閉じられていなかった場合は修復されます。
- myisamchk は --update-state オプションを付けて実行したかどうかをテーブルにマークします。myisamchk --fast はこのマークがないテーブルだけを確認します。
- myisamchk --analyze はキー全体に対してするのと同様に、キーの一部分に対する統計データを格納します。
- myisampack は BLOB と VARCHAR カラムを圧縮できます。

MyISAM は次のような機能もサポートしています。

- 真の VARCHAR 型をサポートしています。VARCHAR カラムは 1 バイトか 2 バイトで格納される長さから始まります。
- VARCHAR カラムを持つテーブルの行の長さは固定でも動的でもかまいません。
- テーブル内の VARCHAR と CHAR カラムの長さの合計は、最大で 64K バイトになる場合があります。
- 任意の長さの UNIQUE 制約。

追加のリソース

- MyISAM ストレージエンジンに特化したフォーラムは https://forums.mysql.com/list.php?21で参照できます。

15.2.1 MyISAM 起動オプション

MyISAM テーブルの振る舞いを変えるために、次の mysqld オプションを使用できます。追加情報については セクション5.1.3「サーバーコマンドオプション」を参照してください。

表 15.3 MyISAM オプション/変数リファレンス

名前	コマンド行	オプションファイル	システム変数	ステータス変数	変数スコープ	動的
bulk_insert_buffer_size	はい	はい	はい		両方	はい
concurrent_insert	はい	はい	はい		グローバル	はい
delay_key_write	はい	はい	はい		グローバル	はい
have_rtree_keys			はい		グローバル	いいえ
key_buffer_size	はい	はい	はい		グローバル	はい
log-isam	はい	はい				
myisam-block-size	はい	はい				
myisam_data_pointer_size	はい	はい	はい		グローバル	はい
myisam_max_sort_file_size	はい	はい	はい		グローバル	はい
myisam_mmap_size	はい	はい	はい		グローバル	いいえ
myisam-recover-options	はい	はい				
myisam_recover_options			はい		グローバル	いいえ
myisam_repair_threads	はい	はい	はい		両方	はい
myisam_sort_buffer_size	はい	はい	はい		両方	はい
myisam_stats_method	はい	はい	はい		両方	はい
myisam_use_mmap	はい	はい	はい		グローバル	はい
skip-concurrent-insert	はい	はい				
- 変数: concurrent_insert						
tmp_table_size	はい	はい	はい		両方	はい

- --myisam-recover-options=モード

  クラッシュした MyISAM テーブルの自動リカバリにモードを設定します。

- --delay-key-write=ALL

  MyISAM テーブルへの書き込みの間にキーバッファーをフラッシュしないでください。

  > 注記
  >
  > これを行う場合、MyISAM テーブルの使用中に別のプログラムから (別の MySQL サーバーから、myisamchk を使用して、など)、このテーブルにアクセスしないでください。そのようにすると、インデックスが破損するおそれがあります。--external-locking を利用しても、このリスクは回避されません。

次のシステム変数は MyISAM テーブルの振る舞いに影響を与えます。追加情報については セクション5.1.4「サーバーシステム変数」を参照してください。

- bulk_insert_buffer_size

  大量挿入の最適化に使用されるツリーキャッシュのサイズです。

  > 注記
  >
  > これは、スレッド当たりの制限値です。

- myisam_max_sort_file_size

MyISAM インデックスを再作成するとき (REPAIR TABLE、ALTER TABLE、または LOAD DATA INFILE 中に)、MySQL が使用を許可されている一時ファイルの最大サイズ。ファイルサイズがこの値より大きい場合、さらに低速なキーキャッシュを代わりに使用してインデックスが作成されます。値はバイト単位で指定されます。

- myisam_sort_buffer_size

  テーブルのリカバリ時に使用されるバッファーのサイズを設定します。

自動リカバリが有効になるのは、mysqld を --myisam-recover-options オプションで起動した場合です。この場合、サーバーが MyISAM テーブルを開いたときに、テーブルにクラッシュのマークが付いているかどうかや、テーブルのオープンカウント変数が 0 でないかどうか、そして外部ロックが使用不可能な状態でサーバーを作動させているかどうかを確認します。これらの条件のいずれかが true である場合、次のことが起こります。

- サーバーは、テーブルにエラーがあるかどうかを確認します。
- サーバーがエラーを検出した場合、迅速なテーブル修復を行います (データファイルのソートは行いますが、再作成は行いません)。
- データファイルの中にエラーがあるために (たとえば、重複キーエラーなど) 修復が失敗した場合、サーバーは再試行して、今度はデータファイルを再作成します。
- それでも修復が失敗した場合、サーバーはもう一度古い修復オプション方式で試行します (ソートをせずに行ごとに書き込みます)。この方法は、どのタイプのエラーも修復できるはずであり、ディスク容量の要件は低くなっています。

このリカバリで、前に実行したステートメントからすべての行をリカバリできず、FORCE を --myisam-recover-options オプションの値に指定しなかった場合、自動修復はエラーメッセージをエラーログに書いて中止します。

```
Error: Couldn't repair table: test.g00pages
```

FORCE を指定すると、代わりにこのような警告が書かれます。

```
Warning: Found 344 of 354 rows when repairing ./test/g00pages
```

自動リカバリの値に BACKUP が含まれていると、リカバリプロセスはフォーム tbl_name-datetime.BAK の名前を持つファイルを作成します。これらのファイルを自動的にデータベースディレクトリからバックアップメディアに移動する cron スクリプトを持つことをお勧めします。

15.2.2 キーに必要な容量

MyISAM テーブルは B ツリーインデックスを使用します。インデックスファイルのサイズは、キーを (key_length+4)/0.67 と計算し、すべてのキーに対してその値を合計して概算できます。これは、すべてのキーがソート順に挿入され、テーブル内のキーが圧縮されていないときの最悪なケースです。

文字列のインデックスはスペース圧縮されています。最初のインデックス部が文字列の場合、プリフィクスも圧縮されています。文字列カラムに含まれる後続の空白が長い場合、またはそのカラムが VARCHAR カラムであるために、必ずしもその長さがフルに使用されることがない場合は、スペース圧縮によってインデックスファイルが最悪の数値よりも小さくなります。プリフィクスの圧縮は文字列から始まるキーで使用されます。多くの文字列が同一のプリフィクスで始まる場合、プリフィクスの圧縮が役に立ちます。

MyISAM テーブルでは、テーブルの作成時に PACK_KEYS=1 テーブルオプションを指定することで、数値のプリフィクスを圧縮することもできます。数値は上位バイトから順に格納されるため、同一のプリフィクスを持つ整数キーが多数あるときに役立ちます。

15.2.3 MyISAM テーブルのストレージフォーマット

MyISAM は 3 つの異なるストレージフォーマットをサポートします。使用するカラムの型によって、固定フォーマットと動的フォーマットの 2 つが自動的に選択されます。3 つ目は圧縮フォーマットで、myisampack ユーティリティーを使用した場合にのみ作成できます (セクション4.6.5「myisampack — 圧縮された読み取り専用の MyISAM テーブルの生成」を参照してください)。

BLOB または TEXT カラムを持たないテーブルに対して、CREATE TABLE または ALTER TABLE を使用する場合、ROW_FORMAT テーブルオプションで、テーブルフォーマットを強制的に FIXED または DYNAMIC にできます。

ROW_FORMAT についての情報は、セクション13.1.17「CREATE TABLE 構文」を参照してください。

圧縮された MyISAM テーブルを myisamchk --unpack を使用して解凍 (アンパック) できます。詳細については、セクション4.6.3「myisamchk — MyISAM テーブルメンテナンスユーティリティー」を参照してください。

15.2.3.1 静的 (固定長) テーブルの特長

MyISAM テーブルでは、静的フォーマットがデフォルトです。テーブルに可変長のカラムが含まれない場合に使用されます (VARCHAR、VARBINARY、BLOB、または TEXT)。各行は固定バイト数で格納されます。

3 つの MyISAM ストレージフォーマットの中で、静的フォーマットが一番シンプルで安全です (一番破損しにくい)。またこれは、ディスク上でデータファイルの中の行が容易に検出できるという理由で、オンディスクフォーマットの中でもっとも高速です。インデックスの中の行数に基づいて行を検索するには、行数に行長を掛けて行の位置を計算します。また、テーブルをスキャンするときに、ディスクの読み込み操作ごとに一定の行数を読み込むことが容易です。

MySQL サーバーが固定フォーマットの MyISAM ファイルに書き込んでいる最中にコンピュータがクラッシュした場合、その安全性が証明されます。この場合、myisamchk はそれぞれの行がどこで始まりどこで終わるかを簡単に判断できるため、通常、一部が書き込まれた行を除くすべての行を再利用できます。MyISAM テーブルインデックスは、データ行に基づいていつでも再構築できます。

> **注記**
>
> 固定長の行フォーマットは、BLOB または TEXT カラムがないテーブルでのみ使用できます。明示的な ROW_FORMAT 句を持つカラムでテーブルを作成すると、エラーや警告は発生しません。フォーマットの仕様は無視されます。

静的フォーマットのテーブルには、次の特徴があります。

- CHAR および VARCHAR カラムには、特定のカラム幅にスペースが埋め込まれます (しかし、カラムの型は変わりません)。BINARY および VARBINARY カラムには、カラム幅に 0x00 バイトが埋め込まれます。
- 非常に高速です。
- キャッシュが容易です。
- 行が固定位置にあるため、クラッシュしたあとも再構築が容易です。
- 大量の行を削除して、空きディスク容量をオペレーティングシステムに戻す場合を除いて、再編成の必要はありません。これを行うには、OPTIMIZE TABLE または myisamchk -r を使用します。
- 通常は、動的フォーマットテーブルよりも多くのディスク容量を必要とします。

15.2.3.2 動的テーブルの特徴

MyISAM テーブルが可変長カラムを含んでいる場合 (VARCHAR、VARBINARY、BLOB、または TEXT)、またはテーブルが ROW_FORMAT=DYNAMIC テーブルオプションで作成された場合は、動的ストレージフォーマットが使用されます。

動的フォーマットは、それぞれの行に行の長さを示すヘッダーがあるため、静的フォーマットよりも少し複雑です。更新の結果として行が長くなった場合、行がフラグメント化される可能性があります (非連続的な断片で格納されます)。

OPTIMIZE TABLE または myisamchk -r を使用して、テーブルをデフラグできます。可変長カラムも含んだテーブルで、固定長カラムに頻繁にアクセスしたり変更したりする場合、可変長カラムを他のテーブルに移動して、フラグメンテーションを回避する方法が良い場合があります。

動的フォーマットのテーブルには、次の特徴があります。

- 長さが 4 未満のカラムを除くすべての文字列カラムは動的です。
- それぞれの行の先頭には、どのカラムが空の文字列 (文字列カラムの場合) またはゼロ (数値カラムの場合) を含むかを示すビットマップが付いています。これには、NULL 値を含むカラムが含まれていません。後続スペースを削除したあとに文字列カラムの長さがゼロであったり、数値カラムの値がゼロであったりした場合、ビットマップの中でマークが付きますが、ディスクには保存されません。空ではない文字列は、長さバイトに文字列コンテンツを加えて保存されます。
- 通常、固定長テーブルに比べると、必要なディスク容量がかなり少なくなります。
- それぞれの行は、必要とする容量だけを使用します。ただし、行がさらに大きくなると、必要な数の断片に分割され、行のフラグメンテーションが起こることになります。たとえば、行の長さを延長する情報を使って行

を更新すると、その行はフラグメント化されます。このような場合、パフォーマンスを上げるために、ときどき OPTIMIZE TABLE または myisamchk -r を実行しなければいけないかもしれません。myisamchk -ei を使用して、テーブルの統計を取得します。

- 行がいくつもの断片にフラグメント化されている場合や、リンク (フラグメント) が失われている場合があるため、クラッシュ後の再構築は、静的フォーマットテーブルよりも難しくなります。
- 動的サイズの行の予想される行長は、次の式で計算されます。

```
3
+ (number of columns + 7) / 8
+ (number of char columns)
+ (packed size of numeric columns)
+ (length of strings)
+ (number of NULL columns + 7) / 8
```

それぞれのリンクには 6 バイトのペナルティーがあります。更新によって行が拡大される場合は、必ず動的行がリンクされます。新しいリンクはそれぞれ少なくとも 20 バイトであるため、おそらく次の拡張は同じリンクになります。そうでない場合、別のリンクが作成されます。myisamchk -ed を利用してリンク数を確認できます。OPTIMIZE TABLE または myisamchk -r を使用すると、すべてのリンクを削除できます。

15.2.3.3 圧縮テーブルの特徴

圧縮ストレージフォーマットは、myisampack ツールで生成される読み取り専用のフォーマットです。圧縮テーブルは myisamchk を使って解凍できます。

圧縮テーブルには次のような特徴があります。

- 圧縮テーブルに必要なディスク容量はごくわずかです。これによりディスクの使用量は最少になるため、低速のディスクを使用する場合に役立ちます (CD-ROM など)。
- それぞれの行は個々に圧縮されるため、アクセスのオーバーヘッドはごくわずかです。行のヘッダーに必要なバイト数は、テーブル中の一番大きい行によって異なりますが、1 - 3 バイトです。各カラムは別々に圧縮されます。カラムごとに異なる Huffman ツリーがあるのが一般的です。圧縮タイプのいくつかは次のとおりです。
  - サフィクススペース圧縮。
  - プリフィクススペース圧縮。
  - 値が 0 の数値は 1 ビットで格納されます。
  - 値の範囲が小さい整数カラムは、可能なかぎり小さな型を使って格納されます。たとえば、BIGINT カラム (8 バイト) のすべての値が -128 から 127 の範囲内にある場合は、このカラムを TINYINT カラム (1 バイト) として格納できます。
  - カラムの可能値が少ない場合は、データの型を ENUMに変換します。
  - カラムに、上記の圧縮型を組み合わせて使用してもかまいません。
- 固定長または動的長の行を使用できます。

> **注記**
>
> 圧縮テーブルは読み取り専用なので、テーブルの行を更新したり、行を追加したりはできませんが、DDL (データ定義言語) 操作は有効です。たとえば、DROP を使用してテーブルを削除しても、TRUNCATE TABLE を使用してテーブルを空にしてもかまいません。

15.2.4 MyISAM テーブルの問題点

MySQL がデータの格納に使用するファイルフォーマットは幅広い検査を受けていますが、データベーステーブルの破損を招きかねない状況は常に存在します。次に、これがどのようにして起こるのか、またどのように対処すればよいのかについて説明します。

15.2.4.1 MyISAM テーブルの破損

MyISAM のテーブルフォーマットは、きわめて信頼性の高いフォーマットです (SQL ステートメントが行うテーブルに対するすべての変更は、そのステートメントが戻る前に書き込まれます) が、それでも次の状況が発生した場合、テーブルが破損するおそれがあります。

- mysqld プロセスは、書き込みの最中に強制終了されます。
- コンピュータが予期せずシャットダウンされます (たとえば、コンピューターの電源が切られた場合など)。
- ハードウェア障害。
- サーバーが修正中のテーブルを、外部プログラム (myisamchkなど) を使用して同時に修正しています。
- MySQL または MyISAM コードのソフトウェアバグです。

テーブルが破損した場合の典型的な兆候は、次のとおりです。

- テーブルからデータを選択するときに、次のエラーが表示されます。

```
Incorrect key file for table: '...'. Try to repair it
```

- クエリーが、テーブル内で行を検出しない、または不完全な結果を返します。

MyISAM テーブルのヘルスを CHECK TABLE ステートメントを利用して確認でき、破損した MyISAM テーブルを REPAIR TABLE を利用して修復できます。mysqld が動作していない場合は、myisamchk コマンドを利用してテーブルを確認したり修復したりすることもできます。セクション13.7.2.2「CHECK TABLE 構文」、セクション13.7.2.5「REPAIR TABLE 構文」、および セクション4.6.3「myisamchk — MyISAM テーブルメンテナンスユーティリティー」 を参照してください。

テーブルが頻繁に破損する場合は、その原因を突き止めるようにしてください。もっとも重要なのは、サーバーのクラッシュによってテーブルが破損されたかどうかを確認することです。エラーログの最新の restarted mysqld メッセージを探すと、簡単に検証できます。このようなメッセージがある場合、テーブルの破損はサーバーのダウンによる可能性が高くなります。そうでなければ、破損は通常作業の最中に起きた可能性があります。これはバグです。問題点を明らかにする再現可能なテストケースを作成するべきです。セクションB.5.4.2「MySQL が繰り返しクラッシュする場合の対処方法」および セクション24.4「MySQL のデバッグおよび移植」を参照してください。

15.2.4.2 適切に閉じられなかったテーブルの問題

各 MyISAM インデックスファイル (.MYI ファイル) には、テーブルが適切に閉じられたかどうかをチェックするために使用できるカウンタがヘッダーの中にあります。CHECK TABLE または myisamchk から次のような警告が表示された場合、このカウンタの同期が取れていないことを示しています。

```
clients are using or haven't closed the table properly
```

この警告は、必ずしもテーブルが破損されたという意味ではありませんが、少なくともテーブルを確認したほうがよいでしょう。

カウンターは次のように機能します。

- MySQL でテーブルが最初に更新されるときに、インデックスファイルのヘッダー内にあるカウンタが増えます。
- その後の更新ではカウンタは変更されません。
- テーブルの最後のインスタンスが閉じられるとき (FLUSH TABLES 操作が行われたため、またはテーブルキャッシュの中に場所がないため) に、それまでにテーブルが更新されていると、カウンタの値が減少します。
- テーブルを修復するか、チェックして問題がなかった場合は、カウンタがゼロにリセットされます。
- テーブルを検査する可能性のあるほかのプロセスとの相互作用の問題を回避するため、カウンタがゼロである場合は、テーブルを閉じる際にカウンタの値は減りません。

つまり、カウンタが不正確になる可能性があるのは、次のような場合だけです。

- MyISAM テーブルのコピーが、最初に LOCK TABLES と FLUSH TABLES を発行しないで行われる。
- MySQL が更新されてから閉じられるまでの間にクラッシュした。(ただし、MySQL は各ステートメントで生じたすべての書き込みを常に発行するため、テーブルに問題がない可能性もあります)。
- mysqld と同時に使用した myisamchk --recover か myisamchk --update-state によって、テーブルが修正された。
- 別のサーバーによって使用されている最中に、複数の mysqld サーバーがテーブルを使用し、1 つのサーバーが REPAIR TABLE または CHECK TABLE をテーブルで実行した。このセットアップでは、ほかのサーバーから

警告を受ける可能性がありますが、CHECK TABLE の使用が安全です。しかし、あるサーバーがデータファイルを新しいファイルに置き換えた場合、別のサーバーには通知されないため、REPAIR TABLE は避けるべきです。

一般的に、複数のサーバー間でデータディレクトリを共有することは推奨されません。追加情報については、セクション5.3「1 つのマシン上での複数の MySQL インスタンスの実行」を参照してください。

15.3 MEMORY ストレージエンジン

MEMORY ストレージエンジン (従来は HEAP と呼ばれていました) は、メモリーに格納された内容で特定用途のテーブルを作成します。データは、クラッシュ、ハードウェア問題、または電源停止に弱いため、これらのテーブルは、一時的な作業領域またはほかのテーブルから抽出されたデータの読み取り専用キャッシュとして使用されるだけです。

表 15.4 MEMORY ストレージエンジンの機能

機能	Support
B ツリーインデックス	はい
MVCC	いいえ
T ツリーインデックス	いいえ
インデックスキャッシュ	N/A
クエリーキャッシュのサポート	はい
クラスタデータベースのサポート	いいえ
クラスタ化されたインデックス	いいえ
ストレージの制限	RAM
データキャッシュ	N/A
データディクショナリ向け更新統計	はい
トランザクション	いいえ
ハッシュインデックス	はい
バックアップ/ポイントインタイムリカバリ (ストレージエンジン内ではなくサーバー内で実装されています。)	はい
レプリケーションのサポート (ストレージエンジン内ではなくサーバー内で実装されています。)	はい
ロック粒度	テーブル
全文検索インデックス	いいえ
圧縮データ	いいえ
地理空間インデックスのサポート	いいえ
地理空間データ型のサポート	いいえ
外部キーのサポート	いいえ
暗号化データ (ストレージエンジン内ではなくサーバー内で (暗号化関数を使って) 実装されています。)	はい

MEMORY または MySQL Cluster を使用する場合　　重要で更新頻度の高い高可用性のデータに対して MEMORY ストレージエンジンを使用するアプリケーションを配備しようとする開発者は、MySQL Cluster がより良い選択かどうかを検討するはずです。MEMORY エンジンの典型的なユースケースには、次の特徴があります。

- セッション管理やキャッシングなどの一時的で重要でないデータに関連する操作。MySQL サーバーが停止または再起動したときに、MEMORY テーブルのデータは失われます。
- 高速アクセスおよび低待機時間のためのインメモリー保存。データボリュームはメモリー内に完全に収まり、オペレーティングシステムによる仮想メモリーページのスワップアウトはありません。
- 読み取り専用または読み取りが大半のデータのアクセスパターン (更新が制限されています)。

MySQL Cluster は、MEMORY エンジンと同じ機能をより高いパフォーマンスレベルで提供し、MEMORY で利用できない追加機能を提供します。

- クライアント間で競合の少ない行レベルロックとマルチスレッド操作。
- 書き込みを含むステートメント混在時の拡張性。
- データ持続性のためのディスクバックアップ式操作 (オプション)。
- 単一障害点がない、シェアードナッシングアーキテクチャーと複数ホスト操作。99.999% の可用性を実現できます。
- ノードをまたがる自動データ分散。アプリケーションの開発者はカスタムの共有またはパーティション化ソリューションを作る必要がありません。
- 可変長データ型 (MEMORY がサポートしない BLOB および TEXT を含みます) をサポートします。

MEMORY ストレージエンジンと MySQL Cluster の詳細な比較に関するホワイトペーパーについては、MySQL Cluster による Web サービスの拡張: MySQL Memory ストレージエンジンの代替を参照してください。このホワイトペーパーには、2 つの技術のパフォーマンス調査と、既存の MEMORY ユーザーが MySQL Cluster にどのように移行できるかについて説明するステップバイステップガイドが含まれています。

パフォーマンスの特徴

MEMORY のパフォーマンスは、更新処理時のシングルスレッド実行とテーブルロックオーバーヘッドが原因の競合によって抑制されます。このため、負荷が増えたときに拡張性が制限されます (特に、書き込みを含むステートメント混在時)。

MEMORY テーブルのインメモリー処理にかかわらず、それらは、汎用目的クエリーのために、または読み取り/書き込み負荷では、必ずしもビジーサーバーの InnoDB テーブルより高速である必要はありません。特に、更新実行に関与するテーブルロックは、複数セッションからの MEMORY テーブルの並列使用の速度を低下させる可能性があります。

MEMORY テーブルで実行されるクエリーの種類によっては、デフォルトのハッシュデータ構造 (一意キーで 1 つの値を検索する場合)、または汎用目的の B ツリーデータ構造 (等号、不等号、未満または「- を超える」などの範囲演算子などを含むすべての種類のクエリーの場合) のいずれかとしてインデックスを作成する場合があります。次のセクションでは、両方の種類のインデックスを作成するための構文について説明します。パフォーマンス面でよくある問題は、B ツリーインデックスがより効率的な作業負荷で、デフォルトのハッシュインデックスを使用していることです。

MEMORY テーブルの物理特性

MEMORY ストレージエンジンは、各テーブルと 1 つのディスクファイル (テーブルの定義を格納 (データではありません)) を関連付けます。ファイル名はテーブル名から始まり、.frm 拡張子が付きます。

MEMORY テーブルには次のような特徴があります。

- MEMORY テーブルの領域は小さなブロックに割り当てられます。テーブルは、挿入に 100% 動的ハッシュを使用します。オーバーフロー領域や余分なキー領域は必要ありません。フリーリスト用の余分な領域は必要ありません。削除された行はリンクリストに置かれ、新しいデータをテーブルに挿入するときに再利用されます。MEMORY テーブルでは、ハッシュテーブルで一般的に削除 + 挿入に関連付けられる問題も起こりません。
- MEMORY テーブルは固定長の行ストレージフォーマットを使用します。VARCHAR などの可変長型は、固定長を使用して格納されます。
- MEMORY テーブルは BLOB または TEXT カラムを含むことができません。
- MEMORY は AUTO_INCREMENT カラムのサポートを含みます。
- TEMPORARY MEMORY でないテーブルは、ほかの TEMPORARY でないテーブルと同様に、すべてのクライアントで共有されます。

MEMORY テーブルへの DDL 操作

MEMORY テーブルを作成するには、CREATE TABLE ステートメントで ENGINE=MEMORY 句を指定します。

```
CREATE TABLE t (i INT) ENGINE = MEMORY;
```

エンジンの名前が表すように、MEMORY テーブルはメモリーに格納されます。デフォルトではハッシュインデックスを使用するため、単一値の検索には非常に高速であり、一時テーブルの作成には非常に役立ちます。ただ

し、サーバーがシャットダウンすると、MEMORY テーブルに格納されたすべての行が失われます。テーブルの定義はディスク上の .frm ファイルに格納されているため、テーブル自体は引き続き存在しますが、サーバーが再起動したときにテーブルは空になります。

この例は、MEMORY テーブルをどのように作成、使用、および削除できるかを示しています。

```
mysql> CREATE TABLE test ENGINE=MEMORY
    ->     SELECT ip,SUM(downloads) AS down
    ->     FROM log_table GROUP BY ip;
mysql> SELECT COUNT(ip),AVG(down) FROM test;
mysql> DROP TABLE test;
```

MEMORY テーブルの最大サイズは max_heap_table_size システム変数によって制限されます (デフォルト値は 16M バイト)。MEMORY テーブルに異なるサイズ制限を適用するには、この変数値を変更します。CREATE TABLE、それに続く ALTER TABLE または TRUNCATE TABLE の実質的な値は、テーブルの有効期限に使用される値です。サーバーを再起動しても、既存の MEMORY テーブルの最大サイズがグローバルの max_heap_table_size 値に設定されます。各テーブルのサイズをこのセクションの後半で説明するように設定できます。

インデックス

MEMORY ストレージエンジンは HASH と BTREE の両方のインデックスをサポートしています。ここに示すように USING 句を追加することによりどちらであるかを指定できます。

```
CREATE TABLE lookup
  (id INT, INDEX USING HASH (id))
  ENGINE = MEMORY;
CREATE TABLE lookup
  (id INT, INDEX USING BTREE (id))
  ENGINE = MEMORY;
```

B ツリーとハッシュインデックスの一般的な特徴については、セクション8.3.1「MySQL のインデックスの使用の仕組み」を参照してください。

MEMORY テーブルでは、テーブル当たり最大で 64 個のインデックス、インデックス当たり 16 個のカラム、3072 バイトの最大キー長を持つことができます。

MEMORY テーブルのハッシュインデックスでキーの重複の程度が高いと (同じ値を含むインデックスエントリが多い)、キー値に影響を与えるテーブルへの更新処理とすべての削除処理の速度が大きく低下します。この低下の程度は重複の程度に比例します (または、インデックスカーディナリティーに反比例します)。BTREE インデックスを使用することで、この問題を回避できます。

MEMORY テーブルには、非一意キーを持つことができます。(これは、ハッシュインデックスの実装ではまれな特徴です。)

インデックスが付けられたカラムに NULL 値を含むことができます。

ユーザー作成の一時テーブル

MEMORY テーブルの内容はメモリーに格納されます。これは MEMORY テーブルが、クエリーの処理中にその場でサーバーが作成する内部一時テーブルと共有する特性です。ただし、2 つのタイプのテーブルには違いがあり、MEMORY テーブルはストレージ変換の影響を受けませんが、内部一時テーブルは次のような影響があります。

- 内部一時テーブルが大きくなりすぎると、セクション8.4.4「MySQL が内部一時テーブルを使用する仕組み」で説明したように、サーバーは自動的にオンディスクストレージに変換します。
- ユーザー作成の MEMORY テーブルは、決してディスクテーブルに変換されません。

データのロード

MySQL サーバーの起動時に MEMORY テーブルを移入するには、--init-file オプションを使用できます。たとえば、このファイルで INSERT INTO ... SELECT や LOAD DATA INFILE などのステートメントを実行することで、永続データソースからテーブルをロードできます。セクション5.1.3「サーバーコマンドオプション」およびセクション13.2.6「LOAD DATA INFILE 構文」を参照してください。

同時に別のセッションでアクセスされた MEMORY テーブルにデータをロードするため、MEMORY は INSERT DELAYED をサポートしています。セクション13.2.5.2「INSERT DELAYED 構文」を参照してください。

MEMORY テーブルとレプリケーション

サーバーの MEMORY テーブルは、シャットダウンされて再起動されたときに空になります。サーバーがレプリケーションマスターの場合は、そのスレーブはこれらのテーブルが空になったことを認識しないため、スレーブのテーブルからデータを選択した場合に内容が古いことがわかります。マスターとスレーブの MEMORY テーブルの同期を取るため、MEMORY テーブルが起動してからマスター側で最初に使用されたときに、スレーブ側でもテーブルを空にするため、DELETE ステートメントがマスターのバイナリログに書かれます。スレーブでは、マスターの再起動とテーブルの最初の使用までの間は、テーブルのデータが古いままです。スレーブへの直接クエリーが古いデータを戻す可能性があるこの期間を避けるには、--init-file オプションを使用して起動時にマスター上の MEMORY テーブルを移入してください。

メモリー使用量の管理

サーバーには、同時に使用されるすべての MEMORY テーブルを保持するための十分なメモリーが必要です。

MEMORY テーブルから各行を削除しても、メモリーは再利用されません。テーブル全体が削除された場合にのみ、メモリーが再利用されます。削除された行に以前に使用されたメモリーは同じテーブル内の新しい行に再利用されます。MEMORY テーブルの内容が必要でなくなったときに、それが使用していたすべてのメモリーを解放するには、DELETE または TRUNCATE TABLE を実行してすべての行を削除するか、DROP TABLE を使用してテーブルを完全に削除します。削除された行が使用していたメモリーを解放するには、ALTER TABLE ENGINE=MEMORY を使用してテーブルを強制的に再作成します。

MEMORY テーブルで 1 つの行に必要なメモリーは、次の式で計算されます。

```
SUM_OVER_ALL_BTREE_KEYS(max_length_of_key + sizeof(char*) * 4)
+ SUM_OVER_ALL_HASH_KEYS(sizeof(char*) * 2)
+ ALIGN(length_of_row+1, sizeof(char*))
```

ALIGN() は行の長さを char ポインタサイズのちょうど倍数にするための切り上げ係数を表します。sizeof(char*) は 32 ビットマシンでは 4、64 ビットマシンでは 8 です。

前に述べたように、max_heap_table_size システム変数は MEMORY テーブルの最大サイズの制限値を設定します。各テーブルの最大サイズを制御するには、各テーブルを作成する前に、この変数のセッション値を設定します。(すべてのクライアントが作成した MEMORY テーブルに、グローバル max_heap_table_size 値を使用するのでなければ、この値を変更しないでください。)次は、2 つの MEMORY テーブル (最大サイズがそれぞれ 1M バイトと 2M バイト) を作成する例です。

```
mysql> SET max_heap_table_size = 1024*1024;
Query OK, 0 rows affected (0.00 sec)

mysql> CREATE TABLE t1 (id INT, UNIQUE(id)) ENGINE = MEMORY;
Query OK, 0 rows affected (0.01 sec)

mysql> SET max_heap_table_size = 1024*1024*2;
Query OK, 0 rows affected (0.00 sec)

mysql> CREATE TABLE t2 (id INT, UNIQUE(id)) ENGINE = MEMORY;
Query OK, 0 rows affected (0.00 sec)
```

両方のテーブルは、サーバーが再起動した場合、サーバーのグローバル max_heap_table_size 値に戻ります。

MEMORY テーブルに対して CREATE TABLE ステートメントの MAX_ROWS テーブルオプションを指定して、テーブルに格納する予定の行数に関するヒントを提供することもできます。これによって max_heap_table_size 値 (引き続き最大テーブルサイズの制約として機能) を超えてテーブルが拡大できなくなります。MAX_ROWS を使用できるだけの最大限の柔軟性を得るには、少なくとも各 MEMORY テーブルが拡大できる値程度に max_heap_table_size を設定してください。

追加のリソース

MEMORY ストレージエンジンに特化したフォーラムは、https://forums.mysql.com/list.php?92で参照できます。

15.4 CSV ストレージエンジン

CSV ストレージエンジンは、カンマ区切り値形式を使用してデータをテキストファイルに保存します。

CSV ストレージエンジンは、常に MySQL サーバーにコンパイルされます。

CSV エンジンのソースを調べるには、MySQL ソース配布の storage/csv ディレクトリを検索します。

CSV テーブルを作成するときに、サーバーはデータベースディレクトリにテーブル形式ファイルを作成します。ファイルはテーブル名から始まり .frm 拡張子が付きます。ストレージエンジンはデータファイルも作成します。その名前はテーブル名で始まり .CSV 拡張子を持ちます。データファイルはプレーンテキストファイルです。データをテーブルに保存するとき、ストレージエンジンはデータファイルにカンマ区切り値形式で保存します。

```
mysql> CREATE TABLE test (i INT NOT NULL, c CHAR(10) NOT NULL)
    -> ENGINE = CSV;
Query OK, 0 rows affected (0.12 sec)

mysql> INSERT INTO test VALUES(1,'record one'),(2,'record two');
Query OK, 2 rows affected (0.00 sec)
Records: 2  Duplicates: 0  Warnings: 0

mysql> SELECT * FROM test;
+------+------------+
| i    | c          |
+------+------------+
|    1 | record one |
|    2 | record two |
+------+------------+
2 rows in set (0.00 sec)
```

CSV テーブルを作成すると、テーブルの状態とテーブルに存在する行数を格納する、対応するメタファイルが作成されます。このファイルの名前は CSM 拡張子のついたテーブル名と同じです。

前のステートメントの実行で作成されたデータベースディレクトリにある test.CSV ファイルを調べると、その内容は次のようであるはずです。

```
"1","record one"
"2","record two"
```

この形式は、Microsoft Excel や StarOffice Calc などのスプレッドシートアプリケーションで読み取ったり書き込んだりできます。

15.4.1 CSV テーブルの修復と確認

CSV ストレージエンジンは、CSV テーブルを検証したり、可能な場合は壊れた CSV テーブルを修復したりする CHECK および REPAIR ステートメントをサポートしています。

CHECK ステートメントを実行すると、正しいフィールドセパレータ、エスケープされたフィールド (引用符の対応または不足)、正しいフィールド数 (テーブル定義と比較)、および対応する CSV メタファイルの存在を探すことで、CSV ファイルの有効性がチェックされます。検出された最初の不正な行はエラーを報告します。有効なテーブルをチェックすると、次のような出力が生成されます。

```
mysql> check table csvtest;
+--------------+-------+----------+----------+
| Table        | Op    | Msg_type | Msg_text |
+--------------+-------+----------+----------+
| test.csvtest | check | status   | OK       |
+--------------+-------+----------+----------+
1 row in set (0.00 sec)
```

壊れたデータをチェックすると、障害が返ります。

```
mysql> check table csvtest;
+--------------+-------+----------+----------+
| Table        | Op    | Msg_type | Msg_text |
+--------------+-------+----------+----------+
| test.csvtest | check | error    | Corrupt  |
+--------------+-------+----------+----------+
1 row in set (0.01 sec)
```

チェックが失敗した場合、テーブルはクラッシュ (破損) とマークされます。テーブルが破損とマークされると、次に CHECK を実行したときや SELECT ステートメントを実行したときに自動的に修復されます。対応する破損ステータスや新しいステータスは、CHECKを実行したときに表示されます。

```
mysql> check table csvtest;
+--------------+-------+----------+---------------------------+
| Table        | Op    | Msg_type | Msg_text                  |
+--------------+-------+----------+---------------------------+
| test.csvtest | check | warning  | Table is marked as crashed |
| test.csvtest | check | status   | OK                        |
+--------------+-------+----------+---------------------------+
```

```
2 rows in set (0.08 sec)
```

テーブルを修正するために REPAIR を使用できます。これは既存の CSV データからできるだけ多くの有効な行をコピーしてから、既存の CSV ファイルをリカバリされた行で置き換えます。破損したデータ以降のすべての行は失われます。

```
mysql> repair table csvtest;
+-------------+--------+----------+----------+
| Table       | Op     | Msg_type | Msg_text |
+-------------+--------+----------+----------+
| test.csvtest | repair | status   | OK       |
+-------------+--------+----------+----------+
1 row in set (0.02 sec)
```

> **警告**
>
> 修復時、破損した最初の行までの CSV ファイルの行だけが新しいテーブルにコピーされます。破損した最初の行からテーブルの最後までのほかのすべての行は、有効な行であっても削除されます。

15.4.2 CSV の制限

CSV ストレージエンジンはインデックスをサポートしていません。

パーティション化は、CSV ストレージエンジンを使ったテーブルではサポートされていません。

CSV ストレージエンジンを使用して作成したすべてのテーブルには、すべてのカラムに NOT NULL 属性が必要です。ただし、下位互換性のため、以前の MySQL リリースで作成した、null許容カラムを持つテーブルを引き続き使用できます。(Bug #32050)

15.5 ARCHIVE ストレージエンジン

ARCHIVE ストレージエンジンは、非常に小さなフットプリントに大量のインデックス化されていないデータを格納する、特別な目的のテーブルを作成します。

表 15.5 ARCHIVE ストレージエンジンの機能

機能	Support
B ツリーインデックス	いいえ
MVCC	いいえ
T ツリーインデックス	いいえ
インデックスキャッシュ	いいえ
クエリーキャッシュのサポート	はい
クラスタデータベースのサポート	いいえ
クラスタ化されたインデックス	いいえ
ストレージの制限	なし
データキャッシュ	いいえ
データディクショナリ向け更新統計	はい
トランザクション	いいえ
ハッシュインデックス	いいえ
バックアップ/ポイントインタイムリカバリ (ストレージエンジン内ではなくサーバー内で実装されています。)	はい
レプリケーションのサポート (ストレージエンジン内ではなくサーバー内で実装されています。)	はい
ロック粒度	テーブル
全文検索インデックス	いいえ
圧縮データ	はい
地理空間インデックスのサポート	いいえ

機能	Support
地理空間データ型のサポート	はい
外部キーのサポート	いいえ
暗号化データ (ストレージエンジン内ではなくサーバー内で (暗号化関数を使って) 実装されています。)	はい

ARCHIVE ストレージエンジンは MySQL バイナリ配布に含まれています。ソースから MySQL を構築する場合にこのストレージエンジンを有効にするには、CMake を -DWITH_ARCHIVE_STORAGE_ENGINE オプションで呼び出します。

ARCHIVE エンジンのソースを調べるには、MySQL ソース配布の storage/archive ディレクトリを検索します。

ARCHIVE ストレージエンジンが SHOW ENGINES ステートメントで使用できるかどうかを確認できます。

ARCHIVE テーブルを作成するときに、サーバーはデータベースディレクトリにテーブル形式ファイルを作成します。ファイルはテーブル名から始まり .frm 拡張子が付きます。ストレージエンジンはほかのファイルを作成します。そのすべてのファイル名はテーブル名で始まります。データファイルの拡張子は .ARZ です。最適化操作中に .ARN ファイルが現れる場合があります。

ARCHIVE エンジンは INSERT と SELECT をサポートしていますが、DELETE、REPLACE、または UPDATE をサポートしていません。ORDER BY 操作、BLOB カラム、基本的に空間データ型を除くすべてのデータ型をサポートしています (セクション11.5.1「空間データ型」を参照してください)。ARCHIVE エンジンは低レベルロックを使用します。

ARCHIVE エンジンは AUTO_INCREMENT カラム属性をサポートしています。AUTO_INCREMENT カラムには、一意のインデックスまたは一意でないインデックスのどちらかを付けることができます。ほかのカラムにインデックスを作成しようとすると、エラーになります。ARCHIVE エンジンは、それぞれ、新しいテーブルの最初のシーケンス値を指定したり、既存テーブルのシーケンス値をリセットしたりする CREATE TABLE ステートメントの AUTO_INCREMENT テーブルオプションもサポートしています。

ARCHIVE は、現在の最大カラム値未満の値を AUTO_INCREMENT カラムに挿入する機能をサポートしていません。そのようにしようとすると、ER_DUP_KEY エラーになります。

ARCHIVE エンジンは BLOB カラムが要求されない場合はそれらを無視して、読み取り中にそれらを通り過ぎてスキャンします。

ストレージ: 行は挿入されるときに圧縮されます。ARCHIVE エンジンは zlib ロスレスデータ圧縮を使用します (http://www.zlib.net/を参照してください)。OPTIMIZE TABLE を使用してテーブルを解析したり、より小さいフォーマットにテーブルを圧縮したりできます (OPTIMIZE TABLE を利用する理由については、このセクションの後半を参照して下さい)。このエンジンは CHECK TABLEもサポートしています。使用される挿入のタイプはいくつかあります。

- INSERT ステートメントは行を圧縮バッファーに単純に入れ、バッファーは必要に応じてフラッシュします。バッファーへの挿入はロックで保護されています。SELECT は、入ってきた挿入が INSERT DELAYED (必要に応じてフラッシュ) だけだった場合を除いて、強制的にフラッシュを発生させます。セクション13.2.5.2「INSERT DELAYED 構文」を参照してください。
- 大量挿入は、ほかの挿入が同時に発生した場合を除いて (その場合は部分的に可視になります)、完了後にのみ可視になります。SELECT は、ロード中に通常の挿入が発生した場合を除いて、大量挿入をフラッシュすることはありません。

取り出し: 取り出しの際、要求によって行が圧縮解除され、行キャッシュはありません。SELECT 操作によって完全なテーブルスキャンが実行されます。SELECT が発生すると、現在使用できる行数を検出し、その行数を読み取ります。SELECT は一貫性読み取りとして実行されます。大量挿入または遅延挿入だけが使用された場合を除き、挿入時の多くの SELECT ステートメントが圧縮品質を低下させることがあります。圧縮品質を高めるために、OPTIMIZE TABLE または REPAIR TABLE を使用できます。SHOW TABLE STATUS によって報告される ARCHIVE テーブルの行数は常に正確です。セクション13.7.2.4「OPTIMIZE TABLE 構文」、セクション13.7.2.5「REPAIR TABLE 構文」、およびセクション13.7.5.37「SHOW TABLE STATUS 構文」を参照してください。

追加のリソース

- ARCHIVE ストレージエンジンに特化したフォーラムは https://forums.mysql.com/list.php?112 で参照できます。

15.6 BLACKHOLE ストレージエンジン

BLACKHOLE ストレージエンジンは、データを受け取るけれども破棄して格納しない「ブラックホール」として機能します。検索は、常に空の結果を返します。

```
mysql> CREATE TABLE test(i INT, c CHAR(10)) ENGINE = BLACKHOLE;
Query OK, 0 rows affected (0.03 sec)

mysql> INSERT INTO test VALUES(1,'record one'),(2,'record two');
Query OK, 2 rows affected (0.00 sec)
Records: 2  Duplicates: 0  Warnings: 0

mysql> SELECT * FROM test;
Empty set (0.00 sec)
```

ソースから MySQL を構築する場合に BLACKHOLE ストレージエンジンを有効にするには、CMake を -DWITH_BLACKHOLE_STORAGE_ENGINE オプションで呼び出します。

BLACKHOLE エンジンのソースを調べるには、MySQL ソース配布の sql ディレクトリを検索します。

BLACKHOLE テーブルを作成するときに、サーバーはデータベースディレクトリ上にテーブルフォーマットファイルを作成します。ファイルはテーブル名から始まり .frm 拡張子が付きます。テーブルに関連するファイルはほかにありません。

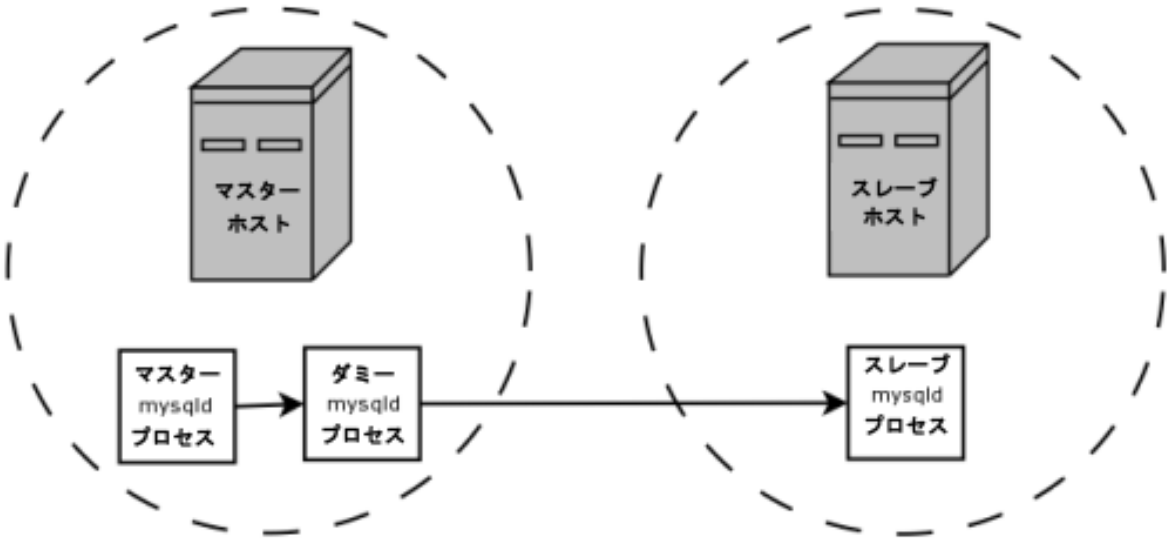
BLACKHOLE ストレージエンジンはすべての種類のインデックスをサポートしています。すなわち、テーブル定義にインデックス宣言を含めることができます。

BLACKHOLE ストレージエンジンが SHOW ENGINES ステートメントで使用できるかどうかを確認できます。

BLACKHOLE テーブルへの挿入はデータを格納しませんが、ステートメントベースバイナリロギングが有効になっている場合は、SQL ステートメントはログが記録されてスレーブサーバーに複製されます。これは、繰り返しまたはフィルタメカニズムとして役立つ場合があります。

> **注記**
>
> バイナリログに行ベースフォーマットを使用すると、更新と削除はスキップされ、ログも適用されません。このため、バイナリロギングフォーマットには ROW や MIXED ではなく STATEMENT を使用してください。

アプリケーションがスレーブ側フィルタリングルールを要求するけれども、すべてのバイナリログデータを最初にスレーブに転送するとトラフィックが多すぎるとします。そのような場合は、次に示すようにマスターホスト上に、デフォルトストレージエンジンが BLACKHOLE である 「dummy」スレーブプロセスをセットアップできます。



マスターはそのバイナリログに書き込みます。「dummy」 mysqld プロセスはスレーブとして機能し、replicate-do-* および replicate-ignore-* ルールの適切な組み合わせを適用し、それ自体のフィルタリングされた新しいバイ

ナリログを書き込みます。セクション17.1.4「レプリケーションおよびバイナリロギングのオプションと変数」を参照してください。フィルタリングされたこのログはスレーブに提供されます。

ダミープロセスは実際にはデータを保存しないため、レプリケーションマスターホスト上で追加 mysqld プロセスを実行することによる処理オーバーヘッドはほとんどありません。この種類のセットアップは、ほかのレプリケーションスレーブで繰り返すことができます。

BLACKHOLE テーブルの INSERT トリガーは期待どおりに機能します。しかし、実際には BLACKHOLE テーブルはデータを格納しないため、UPDATE および DELETE トリガーは有効ではありません。トリガー定義の FOR EACH ROW 句は、行がないために適用されません。

BLACKHOLE ストレージエンジンのその他の利用方法は、次のとおりです。

- ダンプファイル構文の検証。
- バイナリロギングのオーバーヘッドを測定 (バイナリロギングが有効である場合と有効でない場合のパフォーマンスを BLACKHOLE を利用して比較することで)。
- BLACKHOLE は本質的には 「no-op」ストレージエンジンであるため、ストレージエンジン自体には関係ないパフォーマンスボトルネックの検出に使用される場合があります。

コミットされたトランザクションはバイナリログに書き込まれ、ロールバックされたトランザクションは書き込まれないという意味で、BLACKHOLE エンジンはトランザクション対応です。

Blackhole エンジンと自動インクリメントカラム

Blackhole エンジンは no-op エンジンです。Blackhole を使用してテーブルで実行される操作は効果がありません。このことは、自動的に増分する主キーカラムの動作を考慮するときに念頭におくようにしてください。このエンジンはフィールド値を自動的に増分せず、自動インクリメントフィールドの状態を保持しません。これは、レプリケーションで重要な意味を持ちます。

次の 3 つの条件がすべて適用される次のレプリケーションシナリオを検討します。

1. マスターサーバーには、主キーである自動インクリメントフィールドを持つブラックホールテーブルがあります。
2. スレーブには同じテーブルが存在しますが、MyISAM エンジンを使用します。
3. INSERT ステートメント自身で自動インクリメント値を明示的に設定せずに、つまり SET INSERT_ID ステートメントを使用して、マスターのテーブルへの挿入が実行されます。

このシナリオでは、レプリケーションは主キーカラムでの重複エントリーエラーで失敗します。

ステートメントベースのレプリケーションでは、コンテキストイベントの INSERT_ID の値は常に同じになります。このため、主キーカラムで重複値を持つ行を挿入しようとすると、レプリケーションは失敗します。

行ベースのレプリケーションでは、エンジンが戻す行の値は、各挿入で常に同じです。このため、スレーブは主キーカラムで同じ値を使用する 2 つの挿入ログエントリーを再生しようとして、レプリケーションは失敗します。

カラムのフィルタリング

行ベースのレプリケーション (binlog_format=ROW) を使用すると、 セクション17.4.1.9「テーブル定義が異なるマスターとスレーブでのレプリケーション」セクションで説明されたように、最後のカラムがテーブルから失われたスレーブがサポートされます。

このフィルタリングはスレーブ側で機能します。すなわち、カラムはフィルタリングされる前にスレーブにコピーされます。スレーブにカラムをコピーすることが望ましくないケースが少なくとも 2 つあります。

1. データが機密の場合、スレーブサーバーはアクセスすべきではありません。
2. マスターが多くのスレーブを持つ場合、スレーブに送る前にフィルタリングすると、ネットワークトラフィックが削減される可能性があります。

マスターカラムのフィルタリングは、BLACKHOLE エンジンを使用して実現できます。これは、マスターテーブルフィルタリングの実現方法 (BLACKHOLE エンジンと --replicate-do-table または --replicate-ignore-table オプションを使用) に類似した方法で実行されます。

マスターのセットアップは次のとおりです。

```
CREATE TABLE t1 (public_col_1, ..., public_col_N,
                 secret_col_1, ..., secret_col_M) ENGINE=MyISAM;
```

信頼されたスレーブのセットアップは次のとおりです。

```
CREATE TABLE t1 (public_col_1, ..., public_col_N) ENGINE=BLACKHOLE;
```

信頼されないスレーブのセットアップは次のとおりです。

```
CREATE TABLE t1 (public_col_1, ..., public_col_N) ENGINE=MyISAM;
```

15.7 MERGE ストレージエンジン

MRG_MyISAM エンジンとしても知られている MERGE ストレージエンジンは、1 つのテーブルとして使用できる同一の MyISAM テーブルの集まりです。「同一の」というのは、すべてのテーブルが同一のカラムとインデックス情報を持つという意味です。カラムが異なる順番でリストされていたり、完全に同じカラムでなかったり、インデックスの順番が違っていたりすると MyISAM テーブルをマージできません。しかし、MyISAM テーブルのすべてまたはいずれかを myisampack で圧縮できます。セクション4.6.5「myisampack — 圧縮された読み取り専用の MyISAM テーブルの生成」を参照してください。AVG_ROW_LENGTH、MAX_ROWS、PACK_KEYS などのテーブルオプションの違いは問題ではありません。

MERGE テーブルに代わるものはパーティション化されたテーブルであり、1 つのテーブルのパーティションを別々のファイルに格納します。パーティション化によって、一部の操作がより効率的に実行でき、MyISAM ストレージエンジンに制限されません。詳細については、第19章「パーティション化」を参照してください。

MERGE テーブルを作成するときに、MySQL はディスク上に 2 つファイルを作成します。そのファイル名はテーブル名で始まり、ファイルタイプを示す拡張子が付きます。.frm ファイルはテーブルフォーマットを格納し、.MRG ファイルは 1 つのテーブルとして使用すべき基礎 MyISAM テーブル名を含みます。これらのテーブルは、MERGE テーブルと同じデータベースにある必要はありません。

MERGE テーブルでは、SELECT、DELETE、UPDATE、および INSERT を使用できます。MERGE テーブルにマッピングする MyISAM テーブルに対して SELECT、DELETE、および UPDATE 権限が必要です。

> 注記
>
> MERGE テーブルの利用は、次のセキュリティーに関する問題を引き起こします。ユーザーが MyISAM テーブル t に対するアクセス権限を持っていると、そのユーザーは t にアクセスできる MERGE テーブル m を作成できます。しかし、t に対するユーザーの権限があとで破棄された場合、ユーザーは m を介してアクセスすることで t にアクセスを続けることができます。

DROP TABLE を MERGE テーブルに使用すると、MERGE 指定だけが削除されます。基礎テーブルは影響を受けません。

MERGE テーブルを作成するには、どの MyISAM テーブルを使用するかを示す UNION=(list-of-tables) オプションを指定する必要があります。オプションとして、INSERT_METHOD オプションを指定して MERGE テーブルへの挿入方法を制御できます。FIRST または LAST の値を使用すると、それぞれ最初のまたは最後の基礎テーブルで挿入が実行されることになります。INSERT_METHOD オプションを指定しないか、または値 NO 付きでこのオプションを指定すると、MERGE テーブルへの挿入は許可されず、挿入の試みはエラーとなります。

次の例は、MERGE テーブルの作成方法を紹介しています。

```
mysql> CREATE TABLE t1 (
    ->    a INT NOT NULL AUTO_INCREMENT PRIMARY KEY,
    ->    message CHAR(20)) ENGINE=MyISAM;
mysql> CREATE TABLE t2 (
    ->    a INT NOT NULL AUTO_INCREMENT PRIMARY KEY,
    ->    message CHAR(20)) ENGINE=MyISAM;
mysql> INSERT INTO t1 (message) VALUES ('Testing'),('table'),('t1');
mysql> INSERT INTO t2 (message) VALUES ('Testing'),('table'),('t2');
mysql> CREATE TABLE total (
    ->    a INT NOT NULL AUTO_INCREMENT,
    ->    message CHAR(20), INDEX(a))
    ->    ENGINE=MERGE UNION=(t1,t2) INSERT_METHOD=LAST;
```

カラム a は PRIMARY KEY として基礎 MyISAM テーブルではインデックスが付けられていますが、MERGE テーブルではインデックスが付けられていません。MERGE テーブルは基礎テーブルセットに一意性を適用できな

いため、インデックスは設定されますが、PRIMARY KEY としては設定されません。(同様に、基礎テーブルで UNIQUE インデックスを持つカラムには、MERGE テーブルでインデックスが付けられますが、UNIQUE インデックスとしては付けられないはずです。)

MERGE テーブルを作成したあと、このテーブルを使用して、テーブルのグループにまとめて操作を行うクエリーを発行できます。

```
mysql> SELECT * FROM total;
+---+---------+
| a | message |
+---+---------+
| 1 | Testing |
| 2 | table   |
| 3 | t1      |
| 1 | Testing |
| 2 | table   |
| 3 | t2      |
+---+---------+
```

MERGE テーブルを MyISAM テーブルの別のコレクションに対して再マッピングするには、次のいずれかの方法を利用できます。

- MERGE テーブルを DROP して、再作成する。
- 基礎テーブルのリストを変更するために、ALTER TABLE tbl_name UNION=(...) を利用する。

  ALTER TABLE ... UNION=() (つまり、空の UNION 句) を使用してすべての基礎テーブルを削除することもできます。ただしこの場合、テーブルは実質的には空であり、新しい行を取得する基礎テーブルがないために挿入は失敗します。このようなテーブルは、新しい MERGE テーブルを CREATE TABLE ... LIKE で作成するためのテンプレートとして役立つ場合があります。

基礎テーブルの定義とインデックスは、MERGE テーブルの定義と厳密に一致する必要があります。一致がチェックされるのは、MERGE テーブルが作成されたときではなく、MERGE テーブルの一部のテーブルが開いたときです。いずれのテーブルも一致チェックに失敗した場合、テーブルのオープンをトリガーした操作は失敗します。すなわち、MERGE 内のテーブルの定義を変更すると、MERGE テーブルがアクセスされたときに失敗の原因となる可能性があります。それぞれのテーブルに適用される一致チェックは次のとおりです。

- 基礎テーブルと MERGE テーブルのカラム数は同じでなければいけません。
- 基礎テーブルと MERGE テーブルのカラムの順番は一致する必要があります。
- また、親の MERGE テーブル内の対応する各カラムの指定と基礎テーブルの指定を比較して、次のチェック内容を満たす必要があります。
  - 基礎テーブルと MERGE テーブルのカラムの型は一致する必要がある。
  - 基礎テーブルと MERGE テーブルのカラムの長さは一致する必要がある。
  - 基礎テーブルと MERGE テーブルのカラムは NULL でもかまわない。
- 基礎テーブルは、少なくとも MERGE テーブルと同じ数のインデックスを持つ必要がある。基礎テーブルのインデックスの数は MERGE テーブルより多くてもかまわないが、少なくすることはできない。

  > **注記**
  >
  > 同じカラムのインデックスは、MERGE テーブルと基礎 MyISAM テーブルの両方でまったく同じ順番でなければならないという、既知の問題が存在します。バグ #33653 を参照してください。

  各インデックスは次のチェック内容を満たす必要があります。

  - 基礎テーブルと MERGE テーブルのインデックスの型は一致する必要がある。
  - 基礎テーブルと MERGE テーブルのインデックス定義でのインデックス部の数 (すなわち、複合インデックス内に複数のカラム) は一致する必要があります。
  - 各インデックス部について。
    - インデックス部の長さは同じでなければいけない。
    - インデックス部の型は同じでなければいけない。

- インデックス部の言語は同じでなければいけない。
- インデックス部が NULL でかまわないかどうかをチェックする。

MERGE テーブルが基礎テーブルの問題のために、開いたり使用したりできない場合、CHECK TABLE は問題の原因となったテーブルに関する情報を表示します。

追加のリソース

- MERGE ストレージエンジンに特化したフォーラムは、https://forums.mysql.com/list.php?93で参照できます。

15.7.1 MERGE テーブルの長所と短所

MERGE テーブルは、次のような問題を解決するのに役立つことがあります。

- ログテーブルセットを簡単に管理する。たとえば、異なる月のデータを別々のテーブルに入力し、myisampack を利用してそれらの一部を圧縮してから、1 つのものとして利用するために MERGE テーブルを作成できます。
- スピードを上げる。大きな読み取り専用テーブルを同じ基準で分割し、個々のテーブルを異なるディスクに置くことができます。このように構成された MERGE テーブルは、1 つの大きなテーブルを使用するよりも、速度がかなり速くなる可能性があります。
- より効率的に検索を行う。検索する対象が正確にわかっている場合、あるクエリーで基礎テーブルの 1つだけを検索し、別のクエリーで MERGE テーブルを使用できます。重複するテーブルセットを使用する、多数の異なる MERGE テーブルを持つこともできます。
- より効率的な修復を行う。1 つの大きなテーブルを修復するよりも、MERGE テーブルにマッピングされた個々の小さいテーブルを修復する方が簡単です。
- 多くのテーブルを瞬時に 1 つのテーブルとしてマッピングする。MERGE テーブルは個々のテーブルのインデックスを利用するので、それ自体のインデックスを保守する必要はありません。その結果、MERGE テーブルコレクションは、作成や再マッピングを非常に速く行うことができます。(MERGE テーブルを作成するときは、インデックスが作成されない場合でも、インデックスの定義を指定する必要があります。)
- テーブルセットがあり、それらからオンデマンドで大きなテーブルを作成する場合は、代わりにそれらからオンデマンドで MERGE テーブルを作成できます。この方が、速度がかなり速くなり、多くのディスク容量が節約されます。
- オペレーティングシステムのファイルサイズ制限を超える。個々の MyISAM テーブルはこの制限に制約されますが、MyISAM テーブルのコレクションは制約されません。
- MyISAM テーブルにマッピングする MERGE テーブルを定義することで、その単一テーブルのエイリアスやシノニムを作成できます。これを行なっても、特に顕著なパフォーマンス面の影響はないはずです (個々の読み取りのためにいくつかの間接呼び出しや memcpy() 呼び出しがあるだけです)。

MERGE テーブルの短所は次のとおりです。

- MERGE テーブルに対して、同一の MyISAM テーブルしか使用できません。
- MyISAM 機能のいくつかは MERGE テーブルでは使用できません。たとえば、MERGE テーブルに FULLTEXT インデックスを作成できません。(基礎 MyISAM テーブルに FULLTEXT インデックスを作成できますが、MERGE テーブルを全文検索で検索できません。)
- MERGE テーブルが非一時的である場合、すべての基礎 MyISAM テーブルは非一時的である必要があります。MERGE テーブルが一時的である場合、MyISAM テーブルは一時的なテーブルと非一時的なテーブルが混在してもかまいません。
- MERGE テーブルは MyISAM テーブルより多くのファイルディスクリプタを使用します。10 個のクライアントが、10 個のテーブルにマッピングする MERGE テーブルを使用する場合、サーバーは (10 × 10) + 10 個のファイルディスクリプタを使用します。(10 個のクライアントに対してそれぞれ 10 個のデータファイルディスクリプタに加えて、クライアント間で共有される 10 個のインデックスファイルディスクリプタです。)
- インデックスの読み取りは低下します。インデックスを読み取るときに、MERGE ストレージエンジンはすべての基礎テーブルに読み取りを発行して、渡されたインデックス値に厳密に一致するかをチェックする必要があります。次のインデックス値を読み取るために、MERGE ストレージエンジンは読み取りバッファーを検索して次の値を探す必要があります。1 つのインデックスバッファーが使い果たされていた場合にのみ、スト

レージエンジンは次のインデックスブロックを読み取る必要があります。これで、MERGE インデックスは eq_ref 検索をかなり遅くしますが、ref 検索ではそれほど低下しません。eq_ref および ref の詳細情報については、セクション13.8.2「EXPLAIN 構文」を参照してください。

15.7.2 MERGE テーブルの問題点

MERGE テーブルの既知の問題点は次のとおりです。

- 5.1.23 バージョンより前の MySQL Server では、MyISAM の非一時的な子供のテーブルを持つ一時的なマージテーブルを作成できました。

  バージョン 5.1.23 からは、MERGE の子供は親のテーブルを介してロックされました。親が一時的であった場合、親がロックされなかったため、子供もロックされませんでした。MyISAM テーブルを同時に使用すると、テーブルが破損しました。

- ALTER TABLE を使用して MERGE テーブルを別のストレージエンジンに変えると、基礎テーブルへのマッピングが失われます。その代わり、変更されたテーブルに基礎 MyISAM テーブルの行がコピーされ、そのときに、指定されたストレージエンジンを使用します。

- MERGE テーブルの INSERT_METHOD テーブルオプションは、MERGE テーブルへの挿入にどの基礎 MyISAM テーブルを使用するかを示します。ただし、その MyISAM テーブルに AUTO_INCREMENT テーブルオプションを使用しても、1 つ以上の行が MyISAM テーブルに直接挿入されるまで、MERGE テーブルへの挿入は有効になりません。

- MERGE テーブルは、テーブル全体に一意制約を保持できません。INSERT を実行すると、データは最初または最後の MyISAM テーブル (INSERT_METHOD オプションで指定されます) に入ります。MySQL は一意のキー値が MyISAM テーブル内で一意のままであることを保証しますが、コレクション内のすべての基礎テーブルには保証しません。

- MERGE エンジンは基本テーブルセットに一意性を適用できないため、REPLACE は期待どおりに機能しません。次の 2 つの重要な事実があります。

  - REPLACE は、書き込もうとする基礎テーブルでのみ一意キー違反を検出できます (INSERT_METHOD オプションで指定されます)。これは MERGE テーブル自体の違反とは異なります。

  - REPLACE が一意キー違反を検出した場合、書き込んでいる基礎テーブルの対応する行だけを変更します。すなわち、INSERT_METHOD オプションで指定される最初または最後のテーブルです。

  INSERT ... ON DUPLICATE KEY UPDATE についても同様な考慮が適用されます。

- MERGE テーブルはパーティション化をサポートしていません。つまり、MERGE テーブルも、MERGE テーブルの基礎 MyISAM テーブルもパーティション化できません。

- 開いた MERGE テーブルにマッピングされたどのテーブルにも、ANALYZE TABLE、REPAIR TABLE、OPTIMIZE TABLE、ALTER TABLE、DROP TABLE、DELETE (WHERE 句なし)、または TRUNCATE TABLE を使用すべきではありません。そうする場合、MERGE テーブルは引き続き元のテーブルを参照しているため、予期しない結果となる可能性があります。この問題に対処するには、名前付きの操作を行う前に FLUSH TABLES ステートメントを発行することで、確実に MERGE テーブルが開いたままにならないようにします。

  予期しない結果には、MERGE テーブルに対する操作によってテーブルの破損が報告される可能性が含まれます。基礎 MyISAM テーブルで名前付き操作のあとにこれが発生した場合、破損メッセージは偽りです。これに対処するには、MyISAM テーブルを変更したあとで FLUSH TABLES ステートメントを発行します。

- MERGE ストレージエンジンのテーブルマッピングは MySQL の上位レイヤーから隠れているので、MERGE テーブルによって使用されているテーブルでの DROP TABLE は Windows では機能しません。Windows では開いているファイルの削除を許可しないため、最初にすべての MERGE テーブルをフラッシュするか (FLUSH TABLES を使用します)、テーブルを削除する前に MERGE テーブルを削除する必要があります。

- MyISAM テーブルと MERGE テーブルの定義は、テーブルにアクセスするときにチェックされます (たとえば、SELECT または INSERT ステートメントの一部として)。このチェックは、テーブルの定義と親の MERGE テーブルの定義が、カラムの順番、タイプ、サイズ、および関連するインデックスを比較することで一致することを保証します。テーブル間で違いがある場合、エラーが戻され、ステートメントは失敗します。テーブルが開いたときにこれらのチェックが行われるため、1 つのテーブルの定義に変更を加えると (カラムの変更、カラムの順序付け、エンジンの変更など)、ステートメントが失敗する原因になります。

- MERGE テーブルと、その基礎テーブルのインデックスの順番は同一でなければいけません。ALTER TABLE を使用して、MERGE テーブル内で使用されるテーブルに UNIQUE インデックスを追加し、次に ALTER

TABLE を使用して MERGE テーブル上に一意でないインデックスを追加した場合、基礎テーブル内に一意でないインデックスがすでに存在していると、それらのテーブルのインデックス順序は異なります。(これが発生するのは、重複キーをすばやく検出できるように ALTER TABLE が一意でないインデックスの前に UNIQUE インデックスを配置するためです。)その結果、このようなインデックスを持つテーブルに対するクエリーは予想外の結果をもたらす可能性があります。

- ERROR 1017 (HY000): Can't find file: 'tbl_name.MRG' (errno: 2)エラーメッセージが表示された場合、一般的に、いくつかの基礎テーブルが MyISAM ストレージエンジンを使用していないことを表しています。これらのテーブルがすべて MyISAM であることを確認してください。
- MERGE テーブルの行の最大値は 2^{64} です (~1.844E+19 で、MyISAM テーブルの場合と同じ)。複数の MyISAM テーブルを、この数よりも多くの行を含む単一の MERGE テーブルにマージできません。
- MERGE ストレージエンジンは、INSERT DELAYED ステートメントをサポートしていません。
- 親の MERGE テーブルを持つ、異なる行フォーマットの基礎 MyISAM テーブルを使用すると、現在失敗することが知られています。バグ #32364 を参照してください。
- LOCK TABLES が実施されているときは、非一時的な MERGE テーブルの結合リストを変更できません。次は動作しません。

```
CREATE TABLE m1 ... ENGINE=MRG_MYISAM ...;
LOCK TABLES t1 WRITE, t2 WRITE, m1 WRITE;
ALTER TABLE m1 ... UNION=(t1,t2) ...;
```

ただし、一時的な MERGE テーブルではこれを行うことができます。

- 一時的な MERGE としても、非一時的な MERGE テーブルとしても、CREATE ... SELECT で MERGE テーブルを作成できません。例:

```
CREATE TABLE m1 ... ENGINE=MRG_MYISAM ... SELECT ...;
```

これを試みると、tbl_name は BASE TABLE ではないというエラーとなります。

- あるケースでは、MERGE と基礎テーブル間で PACK_KEYS テーブルオプション値が異なると、基礎テーブルに CHAR または BINARY カラムが含まれている場合、予期しない結果になります。回避策として、ALTER TABLE を使用して、関係するすべてのテーブルの PACK_KEYS 値が同じであることを保証します。(Bug #50646)

15.8 FEDERATED ストレージエンジン

FEDERATED ストレージエンジンを使用すると、レプリケーションまたはクラスタの技術を使用しないで、リモートの MySQL データベースのデータにアクセスできます。ローカルの FEDERATED テーブルにクエリーを発行すると、リモート (連合) テーブルからデータを自動的に取得します。データはローカルテーブルに格納されません。

ソースから MySQL を構築する場合に FEDERATED ストレージエンジンを含めるには、CMake を -DWITH_FEDERATED_STORAGE_ENGINE オプションで呼び出します。

FEDERATED ストレージエンジンは、デフォルトでは動作中のサーバーで有効になっていません。FEDERATED を有効にするには、--federated オプションを使用して MySQL サーバーバイナリを起動する必要があります。

FEDERATED エンジンのソースを調べるには、MySQL ソース配布の storage/federated ディレクトリを検索します。

15.8.1 FEDERATED ストレージエンジンの概要

標準のストレージエンジン (MyISAM、CSV、InnoDB など) のいずれかを使用してテーブルを作成すると、そのテーブルはテーブルの定義と関連データで構成されます。FEDERATED テーブルを作成すると、テーブル定義は同じですが、データの物理ストレージはリモートサーバーで処理されます。

FEDERATED テーブルは 2 つの要素で構成されます。

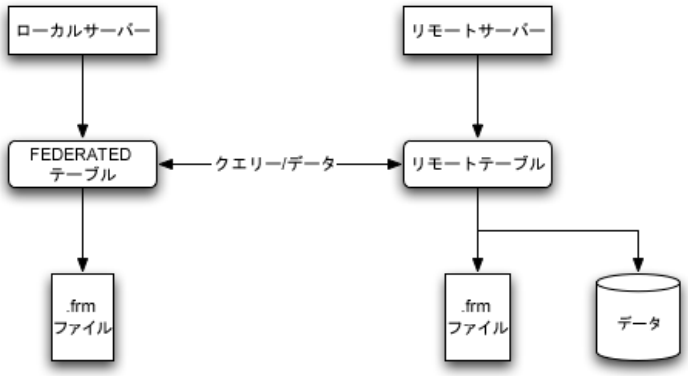
- データベーステーブルを持つ リモートサーバー。テーブル定義 (.frm ファイルに格納されます) と関連テーブルで構成されます。リモートテーブルのテーブルタイプは、MyISAM や InnoDB を含む、リモート mysqld サーバーがサポートするいずれのタイプであってもかまいません。
- データベーステーブルを持つ ローカルサーバー。テーブルの定義は、リモートサーバー上の対応するテーブルの定義に一致します。テーブル定義は .frm ファイルに格納されます。ただし、ローカルサーバーにデータファ

イルはありません。その代わり、テーブル定義にはリモートテーブルをポイントする接続文字列が含まれています。

ローカルサーバーで FEDERATED テーブルにクエリーやステートメントを実行すると、一般的にローカルデータファイルの情報を挿入、更新、または削除する操作は、実行するために代わりにリモートサーバーに送られ、そこでリモートサーバーのデータファイルを更新したり、リモートサーバーから一致する行を戻したりします。

FEDERATED テーブルのセットアップの基本的な構造は 図15.1「FEDERATED テーブルの構造」で示します。

図 15.1 FEDERATED テーブルの構造



FEDERATED テーブルを参照する SQL ステートメントをクライアントが発行する場合、ローカルサーバー (SQL ステートメントが実行される場所) とリモートサーバー (データが物理的に格納される場所) の間の情報の流れは次のとおりです。

1. ストレージエンジンは FEDERATED テーブルが持つ各カラムを調べて、リモートテーブルを参照する適当な SQL ステートメントを構築します。
2. ステートメントは MySQL クライアント API を使用してリモートサーバーに送られます。
3. リモートサーバーはステートメントを処理し、ローカルサーバーはステートメントが作成した結果 (影響を受けた行の数や結果セット) を取得します。
4. ステートメントが結果セットを作成する場合、各カラムは FEDERATED エンジンが求める内部ストレージエンジン形式に変換され、元のステートメントを発行したクライアントに結果を表示するために使用できます。

ローカルサーバーは、MySQL クライアントの C API 関数を使用してリモートサーバーと通信します。mysql_real_query() を呼び出して、ステートメントを送信します。結果セットを読み取るために、mysql_store_result() を使用し、mysql_fetch_row() を使用して 1 つずつ行をフェッチします。

15.8.2 FEDERATED テーブルの作成方法

FEDERATED テーブルを作成するときは、次の手順に従うようにしてください。

1. リモートサーバーにテーブルを作成します。または、SHOW CREATE TABLE ステートメントを使用するなどして、既存テーブルのテーブル定義のメモを取ります。
2. 同一のテーブル定義でローカルサーバーにデーブルを作成しますが、ローカルテーブルをリモートテーブルにリンクする接続情報を追加してください。

たとえば、リモートサーバーに次のテーブルを作成できます。

```
CREATE TABLE test_table (
    id     INT(20) NOT NULL AUTO_INCREMENT,
    name   VARCHAR(32) NOT NULL DEFAULT '',
    other  INT(20) NOT NULL DEFAULT '0',
    PRIMARY KEY  (id),
    INDEX name (name),
    INDEX other_key (other)
)
ENGINE=MyISAM
DEFAULT CHARSET=latin1;
```

リモートテーブルに連合したローカルテーブルを作成するには、2 つのオプションが使用できます。ローカルテーブルを作成し、CONNECTION を使用してリモートテーブルへの接続に使用される接続文字列 (サーバー名、ログイン、パスワードを含みます) を指定するか、CREATE SERVER ステートメントを使用してすでに作成された既存の接続を使用できます。

> **重要**
>
> ローカルテーブルを作成する場合、リモートテーブルに同一のフィールド定義を持つ必要があります。

> **注記**
>
> インデックスをホストのテーブルに追加することで、FEDERATED テーブルのパフォーマンスを向上できます。リモートサーバーに送られたクエリーには WHERE 句の内容が含まれており、それがリモートサーバーに送られてローカルに実行されるため、最適化が起こります。これにより、そうしないとローカル処理のためにサーバーからテーブル全体を要求することになるネットワークトラフィックが削減されます。

15.8.2.1 CONNECTION を使用した FEDERATED テーブルの作成

最初の方法を使用するには、CREATE TABLE ステートメントのエンジンタイプの後ろに CONNECTION 文字列を指定する必要があります。例:

```
CREATE TABLE federated_table (
    id     INT(20) NOT NULL AUTO_INCREMENT,
    name   VARCHAR(32) NOT NULL DEFAULT '',
    other  INT(20) NOT NULL DEFAULT '0',
    PRIMARY KEY  (id),
    INDEX name (name),
    INDEX other_key (other)
)
ENGINE=FEDERATED
DEFAULT CHARSET=latin1
CONNECTION='mysql://fed_user@remote_host:9306/federated/test_table';
```

> **注記**
>
> CONNECTION は MySQL の以前のバージョンで使われた COMMENT を置き換えるものです。

CONNECTION 文字列には、データを物理的に格納するために使用されるテーブルを含む、リモートサーバーへの接続に必要な情報が含まれています。接続文字列には、サーバー名、ログイン資格証明、ポート番号、およびデータベース/テーブル情報を指定します。この例では、リモートテーブルはサーバー remote_host 上にあり、ポート 9306 を使用します。名前とポート番号は、リモートテーブルとして使用するリモート MySQL サーバーのホスト名 (または IP アドレス) とポート番号に一致するべきです。

接続文字列の書式は次のとおりです。

```
scheme://user_name[:password]@host_name[:port_num]/db_name/tbl_name
```

ここでは:

- scheme: 認識された接続プロトコル。この時点では、mysql だけが scheme 値としてサポートされています。
- user_name: 接続のためのユーザー名。このユーザーは、リモートサーバー上に作成されている必要があり、リモートテーブルで必要なアクション (SELECT、INSERT、UPDATE など) を実行するのに適した権限を持つ必要があります。
- password: (オプション) user_name に対応するパスワード。
- host_name: リモートサーバーのホスト名または IP アドレス。
- port_num: (オプション) リモートサーバーのポート番号。デフォルトは 3306 です。
- db_name: リモートテーブルを保持するデータベースの名前。
- tbl_name: リモートテーブルの名前。ローカルテーブルとリモートテーブルの名前が一致する必要はありません。

接続文字列の例は次のとおりです。

```
CONNECTION='mysql://username:password@hostname:port/database/tablename'
CONNECTION='mysql://username@hostname/database/tablename'
CONNECTION='mysql://username:password@hostname/database/tablename'
```

15.8.2.2 CREATE SERVER を使用した FEDERATED テーブルの作成

多くの FEDERATED テーブルを同じサーバーに作成する場合、または FEDERATED テーブルの作成プロセスを単純化する必要がある場合、CREATE SERVER ステートメントを使用してサーバー接続パラメータを定義できます (CONNECTION 文字列の場合と同様)。

CREATE SERVER ステートメントの書式は次のとおりです。

```
CREATE SERVER
server_name
FOREIGN DATA WRAPPER wrapper_name
OPTIONS (option [, option] ...)
```

server_name は FEDERATED テーブルを作成するときに接続文字列で使用されます。

たとえば CONNECTION 文字列と同一のサーバー接続を作成するには、次のとおりです。

```
CONNECTION='mysql://fed_user@remote_host:9306/federated/test_table';
```

次のステートメントを使用することになります。

```
CREATE SERVER fedlink
FOREIGN DATA WRAPPER mysql
OPTIONS (USER 'fed_user', HOST 'remote_host', PORT 9306, DATABASE 'federated');
```

この接続を使用する FEDERATED テーブルを作成するには、CONNECTION キーワードも使用しますが、CREATE SERVER ステートメントで使用した名前を指定します。

```
CREATE TABLE test_table (
    id     INT(20) NOT NULL AUTO_INCREMENT,
    name   VARCHAR(32) NOT NULL DEFAULT '',
    other  INT(20) NOT NULL DEFAULT '0',
    PRIMARY KEY  (id),
    INDEX name (name),
    INDEX other_key (other)
)
ENGINE=FEDERATED
DEFAULT CHARSET=latin1
CONNECTION='fedlink/test_table';
```

この例の接続名には、接続の名前 (fedlink) とリンクするテーブルの名前 (test_table) が含まれます (区切りはスラッシュ)。テーブル名なしで接続名だけを指定した場合、代わりにローカルテーブルのテーブル名が使用されます。

CREATE SERVER の詳細情報については、セクション13.1.16「CREATE SERVER 構文」を参照してください。

CREATE SERVER ステートメントは、CONNECTION 文字列と同じ引数を受け入れます。CREATE SERVER ステートメントは mysql.servers テーブルの中の行を更新します。接続文字列のパラメータ間の通信、CREATE SERVER ステートメントのオプション、および mysql.servers テーブルのカラムに関する情報については、次の表を参照してください。参考までに、CONNECTION 文字列の書式は次のとおりです。

```
scheme://user_name[:password]@host_name[:port_num]/db_name/tbl_name
```

説明	CONNECTION 文字列	CREATE SERVER オプション	mysql.servers カラム
接続スキーム	scheme	wrapper_name	Wrapper
リモートユーザー	user_name	USER	Username
リモートパスワード	password	PASSWORD	Password
リモートホスト	host_name	HOST	Host
リモートポート	port_num	PORT	Port
リモートデータベース	db_name	DATABASE	Db

15.8.3 FEDERATED ストレージエンジンの注記とヒント

FEDERATED ストレージエンジンを使用するときは、次の点に注意することをお勧めします。

- FEDERATED テーブルをほかのスレーブに複製してもかまいませんが、スレーブサーバーが CONNECTION 文字列 (または mysql.servers テーブルの行) で定義されたユーザーとパスワードの組み合わせを使用してリモートサーバーに接続できることを確認する必要があります。

次の項目は、FEDERATED ストレージエンジンがサポートしている機能とサポートしていない機能を示します。

- リモートサーバーは MySQL サーバーでなくてはいけません。
- FEDERATED テーブルがポイントするリモートテーブルは、FEDERATED テーブルを介してそのテーブルにアクセスを試みる前に、存在している必要があります。
- ある FEDERATED テーブルがほかのテーブルをポイントすることは可能ですが、ループを作らないように注意する必要があります。
- FEDERATED テーブルは本質的にインデックスをサポートしていません。テーブルへのアクセスがリモートで処理されるため、リモートテーブルがインデックスをサポートします。同等の MyISAM やほかのテーブルからのインデックス定義がサポートされていない可能性があるため、FEDERATED テーブルを作成するときは注意を払うようにしてください。たとえば、VARCHAR、TEXT、または BLOB カラムにインデックスプリフィクスを付けた FEDERATED テーブルを作成すると、失敗します。MyISAM の次の定義は有効です。

  ```
  CREATE TABLE `T1`(`A` VARCHAR(100),UNIQUE KEY(`A`(30))) ENGINE=MYISAM;
  ```

  この例のキープリフィクスは FEDERATED エンジンと互換性がないため、同等のステートメントは失敗します。

  ```
  CREATE TABLE `T1`(`A` VARCHAR(100),UNIQUE KEY(`A`(30))) ENGINE=FEDERATED
    CONNECTION='MYSQL://127.0.0.1:3306/TEST/T1';
  ```

  可能であれば、これらのインデックスの問題を回避するため、リモートサーバーとローカルサーバーの両方にテーブルを作成する場合、カラムとインデックスの定義を分けるようにしてください。
- 内部的に、実装は SELECT、INSERT、UPDATE、および DELETE を使用しますが、HANDLER は使用しません。
- FEDERATED ストレージエンジンは、SELECT、INSERT、UPDATE、DELETE、TRUNCATE TABLE、およびインデックスをサポートしています。DROP TABLE を除いて、ALTER TABLE や、テーブルの構造に直接影響を与えるデータ定義言語ステートメントをサポートしていません。現在の実装は、プリペアドステートメントを使用しません。
- FEDERATED は INSERT ... ON DUPLICATE KEY UPDATE ステートメントを受け入れますが、重複キー違反が起こった場合、ステートメントはエラーで失敗します。
- 大量の挿入を実行する場合 (たとえば、INSERT INTO ... SELECT ... ステートメント) の FEDERATED テーブルのパフォーマンスは、選択された各行が FEDERATED テーブルで個々の INSERT ステートメントとして処理されるため、ほかのテーブルタイプに比べて低下します。
- トランザクションはサポートされていません。
- FEDERATED は、複数の行がバッチでリモートテーブルに送られるように大量挿入処理を実行します。これでパフォーマンスは向上し、リモートテーブルは改善を実行できます。また、リモートテーブルがトランザクション対応の場合、エラーが発生したときにリモートストレージエンジンはステートメントロールバックを適切に実行できます。この機能には次の制限があります。
  - 挿入のサイズは、サーバー間の最大パケットサイズを超えることはできません。挿入がこのサイズを超えた場合、複数のパケットに分割され、ロールバック問題が発生する可能性があります。
  - 大量挿入処理は INSERT ... ON DUPLICATE KEY UPDATE では起こりません。
- FEDERATED エンジンは、リモートテーブルが変わったかどうかを知る方法がありません。その理由は、このテーブルが、データベースシステム以外の何かによって決して書き込まれることのないデータファイルのように動作する必要があるためです。リモートデータベースに変更が加えられた場合に、ローカルテーブルのデータの完全性が損なわれる可能性があります。
- CONNECTION 文字列を使用する場合、パスワードに '@' 文字を使用できません。CREATE SERVER ステートメントを使用してサーバー接続を作成することで、この制限を回避できます。
- insert_id および timestamp オプションはデータプロバイダには伝達されません。
- FEDERATED テーブルに対して発行された DROP TABLE ステートメントは、ローカルテーブルだけを削除し、リモートテーブルは削除しません。

- FEDERATED テーブルはクエリーキャッシュでは機能しません。
- ユーザー定義のパーティション化は、FEDERATED テーブルではサポートされていません。

15.8.4 FEDERATED ストレージエンジンのリソース

次の追加リソースは、FEDERATED ストレージエンジンで利用できます。

- FEDERATED ストレージエンジンに特化したフォーラムは https://forums.mysql.com/list.php?105 で参照できます。

15.9 EXAMPLE ストレージエンジン

EXAMPLE ストレージエンジンは、何もしないスタブエンジンです。その目的は、新しいストレージエンジンの書き込みを開始する方法を示す MySQL ソースコードの例として機能することです。このため、主に開発者を対象としています。

ソースから MySQL を構築する場合に EXAMPLE ストレージエンジンを有効にするには、CMake を -DWITH_EXAMPLE_STORAGE_ENGINE オプションで呼び出します。

EXAMPLE エンジンのソースを調べるには、MySQL ソース配布の storage/example ディレクトリを検索します。

EXAMPLE テーブルを作成すると、サーバーはデータベースディレクトリ上にテーブル形式ファイルを作成します。ファイルはテーブル名から始まり .frm 拡張子が付きます。ほかのファイルは作成されません。データをテーブルに格納できません。検索は空の結果を返します。

```
mysql> CREATE TABLE test (i INT) ENGINE = EXAMPLE;
Query OK, 0 rows affected (0.78 sec)

mysql> INSERT INTO test VALUES(1),(2),(3);
ERROR 1031 (HY000): Table storage engine for 'test' doesn't »
                have this option

mysql> SELECT * FROM test;
Empty set (0.31 sec)
```

EXAMPLE ストレージエンジンはインデックスをサポートしていません。

15.10 ほかのストレージエンジン

ストレージエンジンは、カスタムストレージエンジンのインタフェースを使用したサードパーティーおよびコミュニティーメンバーの別のストレージエンジンを使用できる場合があります。

サードパーティーのエンジンは MySQL によってサポートされていません。これらのエンジンに関する詳細情報、ドキュメント、インストールガイド、バグレポート、ヘルプや支援については、そのエンジンの開発者に直接問い合わせてください。

プラガブルストレージエンジンアーキテクチャーで使用できるお客様のストレージエンジンの開発に関する詳細については、「MySQL Internals: Writing a Custom Storage Engine」を参照してください。

15.11 MySQL ストレージエンジンアーキテクチャーの概要

MySQL プラガブルストレージエンジンアーキテクチャーを採用すると、データベースの専門家は、特定のアプリケーションニーズに特化したストレージエンジンを選択でき、さらにアプリケーションの特定のコーディング要件を管理する必要が完全になくなります。MySQL サーバーのアーキテクチャーにより、アプリケーションプログラマと DBA はストレージレベルのすべての実装詳細から解放され、一貫した容易なアプリケーションモデルと API が得られます。したがって、異なるストレージエンジンの機能には違いがありますが、アプリケーションはその違いから解放されます。

プラガブルストレージエンジンのアーキテクチャーは、すべての基になるストレージエンジンに共通の管理およびサポートサービスの標準セットを提供します。ストレージエンジン自身は、物理サーバーレベルで保守される基になるデータに対してアクションを実際に実行するデータベースサーバーのコンポーネントです。

この効率的なモジュール形式のアーキテクチャーは、データウェアハウス、トランザクション処理、高可用性状況など、特別なアプリケーションニーズを特に対象にしたい人、またどれか 1 つのストレージエンジンに依存しないインタフェースとサービスのセットを利用するメリットを享受したい人にとって、大きなメリットが得られます。

アプリケーションプログラマと DBA は、コネクタ API およびストレージエンジンの上位にあるサービスレイヤーを介して MySQL データベースと対話します。アプリケーションの変更によって、基になるストレージエンジンの変更を求める要件が発生した場合、または新しいニーズをサポートするために 1 つ以上のストレージエンジンが追加された場合、これをうまく行うためにコーディングやプロセスを大幅に変更する必要はありません。MySQL サーバーのアーキテクチャーは、ストレージエンジンに適用される、一貫して使いやすい API を提供することで、ストレージエンジンの内在する複雑さからアプリケーションを解放します。

15.11.1 プラガブルストレージエンジンのアーキテクチャー

MySQL Server は、ストレージエンジンが、動作中の MySQL サーバーにロードされたり、MySQL サーバーからアンロードされたりできる、プラガブルストレージエンジンアーキテクチャーを採用しています。

ストレージエンジンのプラグイン

ストレージエンジンを使用する前に、INSTALL PLUGIN ステートメントを利用してストレージエンジンのプラグイン共用ライブラリを MySQL にロードする必要があります。たとえば、EXAMPLE エンジンのプラグインの名前が example で、共有ライブラリの名前が ha_example.so である場合、次のステートメントを使用してロードします。

```
mysql> INSTALL PLUGIN example SONAME 'ha_example.so';
```

プラガブルストレージエンジンをインストールするには、プラグインファイルは MySQL プラグインディレクトリにある必要があり、INSTALL PLUGIN ステートメントを発行するユーザーには、mysql.plugin テーブルの INSERT 権限が必要です。

共有ライブラリは MySQL サーバーのプラグインディレクトリの中にある必要があり、その場所は plugin_dirシステム変数によって指示されます。

ストレージエンジンのアンプラグ

ストレージエンジンをアンプラグするには、UNINSTALL PLUGIN ステートメントを利用します。

```
mysql> UNINSTALL PLUGIN example;
```

既存のテーブルが必要とするストレージエンジンをアンプラグした場合、それらのテーブルはアクセスできなくなりますが、引き続きディスク上 (適切な場所) に存在します。ストレージエンジンをアンプラグする前に、そのストレージエンジンを使用しているテーブルがないことを確認してください。

15.11.2 共通データベースサーバーレイヤー

MySQL プラガブルストレージエンジンは、データベースに対して実データの I/O 操作を行なったり、特定のアプリケーションニーズを対象とする機能セットを有効にしたり適用したりする役割を担う、MySQL データベースサーバーのコンポーネントです。特定のストレージエンジンを使用する主なメリットは、特定のアプリケーションに必要な機能だけが配布されるため、データベースのオーバーヘッドが小さくなり、データベースパフォーマンスが効率的になり向上します。これは、MySQL がこのように高パフォーマンスであると以前から知られてきた理由の 1 つであり、業界標準ベンチマークで独占的な地位を占める強力なデータベースに匹敵または対抗できる要因になっています。

技術的に見て、ストレージエンジンを支える独自のインフラストラクチャー要素は何でしょうか。機能を差別化している主な要素は次のとおりです。

- 並列性: いくつかのアプリケーションは、ほかのアプリケーションよりさらに粒度の細かいロック要件 (低レベルロックなど) を持ちます。適切なロック方式を選択すると、オーバーヘッドが低減されるため、全体のパフォーマンスが向上します。また、この分野はマルチバージョンの並列処理制御や「スナップショット」の読み込みのような機能もサポートします。
- トランザクションサポート: 必ずしもすべてのアプリケーションがトランザクションを必要としていませんが、必要な場合、ACID 準拠などの非常に明確な要件があります。
- 参照整合性: サーバーは DDL 定義の外部キーでリレーショナルデータベースの参照整合性を適用する必要があります。
- 物理ストレージ: これには、テーブルとインデックスの全体ページサイズやデータの物理ディスクへの格納に使用されるフォーマットのすべてが関係します。
- インデックスサポート: アプリケーションシナリオによっては、別のインデックス方式からメリットが得られる傾向があります。通常、各ストレージエンジンには独自のインデックス方式がありますが、ほぼすべてのエンジンに共通の方式 (B ツリーインデックスなど) もあります。

- メモリーキャッシュ: アプリケーションによってはあるメモリーキャッシュ方式の方がほかより応答が良いものがあり、あるメモリーキャッシュはすべてのストレージエンジンに共通だけれども (ユーザー接続または MySQL の高速クエリーキャッシュに使用されるものなど)、特定のストレージエンジンが動作するときにのみ独自に定義されるものもあります。
- パフォーマンスエイド: これには、並列操作用のマルチ I/O スレッド、スレッド並列処理、データベースチェックポイント、大量の挿入処理などが含まれます。
- その他のターゲット機能: これには、地理空間操作、データ操作のセキュリティー面の制限事項、およびその他の類似機能に対するサポートが含まれます。

プラガブルストレージエンジンの各インフラストラクチャーコンポーネントセットは、特定のアプリケーション向けの利点を提供できるように設計されています。反対に、あるコンポーネント機能セットを回避することは、不必要なオーバーヘッドを削減するのに役立ちます。特定アプリケーションの要件セットを理解して、適切な MySQL ストレージエンジンを選択することは、当然のことながらシステム全体の効率とパフォーマンスに大きな影響を与える可能性があります。

第 16 章 高可用性と拡張性

目次

16.1 Oracle VM Template for MySQL Enterprise 1795
16.2 DRBD/Pacemaker/Corosync/Oracle Linux を使用した MySQL の概要 1796
16.3 Windows フェイルオーバークラスタリングを使用した MySQL の概要 1798
16.4 Amazon EC2 インスタンスでの MySQL の使用 1800
16.4.1 EC2 AMI での MySQL のセットアップ 1800
16.4.2 EC2 インスタンスの制限 1801
16.4.3 EC2 を使用した MySQL データベースの配備 1802
16.5 ZFS レプリケーションの使用 1804
16.5.1 ZFS を使用したファイルシステムレプリケーション 1805
16.5.2 ZFS レプリケーションのための MySQL の構成 1806
16.5.3 ZFS での MySQL リカバリーの扱い 1806
16.6 MySQL と memcached の併用 1807
16.6.1 memcached のインストール 1808
16.6.2 memcached の使用 1809
16.6.3 memcached アプリケーションの開発 1825
16.6.4 memcached の統計の取得 1846
16.6.5 memcached の FAQ 1853

データは、今日の Web、モバイル、ソーシャル、エンタープライズ、およびクラウドアプリケーションの通貨です。データが常に利用できることを確保することは、いずれの組織でも一番の優先事項です。分単位のダウンタイムでも、収益と評価が大きく損なわれる可能性があります。

高可用性 (HA) を実現するための「万能な」方法はありません。固有のアプリケーション属性、ビジネス要件、運用性能、およびレガシーインフラストラクチャーはすべて、HA テクノロジーの選択に影響を与える可能性があります。そして、HA を実現する上でテクノロジーは 1 つの要素でしかありません。テクノロジー自身と同じくらい人とプロセスが重要です。

MySQL は可用性と拡張性を求める多くのアプリケーションに配備されています。可用性とは、ホストでの障害 (MySQL、オペレーティングシステム、ハードウェア、保守作業での障害を含む、対処しないとダウンタイムが発生する可能性がある) に対処し、必要に応じてリカバリさせる能力のことです。拡張性とは、データベース、およびアプリケーションクエリーの負荷の両方を複数の MySQL サーバーに分散させる能力のことです。

アプリケーションごとに運用および可用性要件が異なるため、MySQL は認定およびサポートされるソリューションを幅広く提供し、サービスレベル要件を満たす適切なレベルの高可用性 (HA) と拡張性を実現しています。このようなソリューションは、レプリケーションから、仮想化、地理的冗長性、99.999% の稼働時間を実現するマルチデータセンターソリューションまで用意しています。

アプリケーションのために最適な高可用性ソリューションを選択することは、主に次のことに依存します。

- 必要な可用性レベル。
- 配備されるアプリケーションのタイプ。
- 独自の環境で受け入れられるベストプラクティス。

MySQL がサポートするプライマリソリューションは次のとおりです。

- MySQL レプリケーション。詳細: 第17章「レプリケーション」。
- MySQL Cluster。詳細: 第18章「MySQL Cluster NDB 7.3 および MySQL Cluster NDB 7.4」。
- Oracle VM Template for MySQL。詳細: セクション16.1「Oracle VM Template for MySQL Enterprise」。
- DRBD、Corosync、Pacemaker を実装する MySQL。詳細: セクション16.2「DRBD/Pacemaker/Corosync/Oracle Linux を使用した MySQL の概要」。
- Windows Failover Clustering を実装する MySQL。詳細: セクション16.3「Windows フェイルオーバークラスタリングを使用した MySQL の概要」。
- Solaris Cluster を実装する MySQL。Solaris Cluster の理解を深めてください。

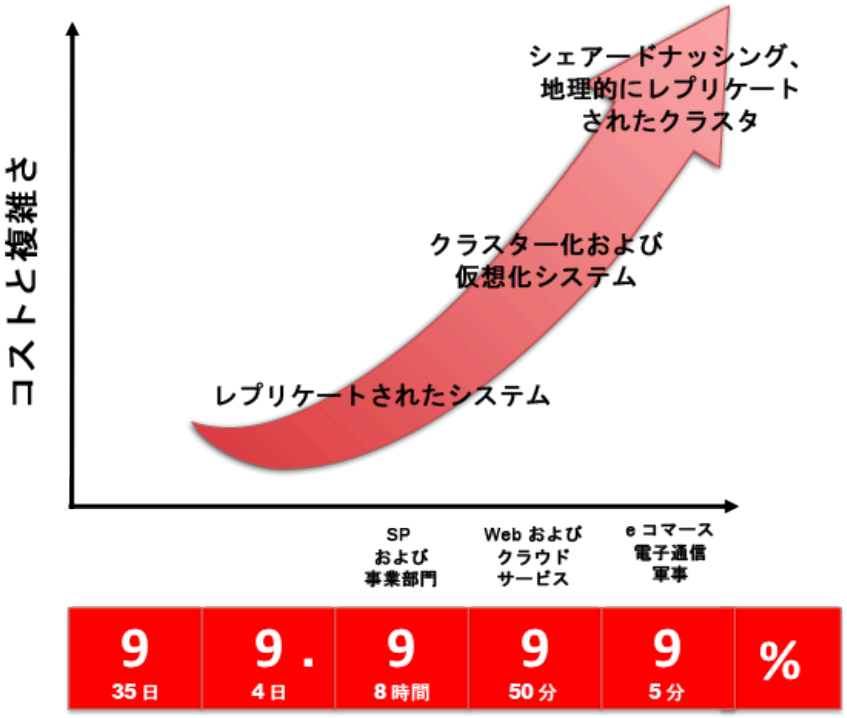
追加オプションをサードパーティーソリューションで利用できます。

高可用性データベースサービスの実現に使用される各アーキテクチャーは、それが提供する稼働時間のレベルによって区別されます。これらのアーキテクチャーは、次の 3 つの主なカテゴリにグループ分けできます。

- データレプリケーション。
- クラスター化および仮想化システム。
- シェアードナッシング、地理的にレプリケートされたクラスタ。

次の図で示すように、これらの各アーキテクチャーは段階的に稼働時間レベルが高くなるため、それらによってコストと複雑さのレベルが潜在的に大きくなる可能性があり、これらのバランスを取る必要があります。単に高可用性アーキテクチャーを配備するだけでは、実際に HA が実現されることの保証にはなりません。実際、シェアードナッシングクラスタの実装と保守が不十分であると、単純なデータレプリケーションソリューションよりも、可用性レベルがあっさりと低下する可能性があります。

図 16.1 トレードオフ: コストと複雑さ vs 可用性



次の表では、さまざまな MySQL ソリューションの HA および拡張性機能を比較しています。

表 16.1 MySQL HA ソリューションの機能比較

要件	MySQL レプリケーション	DRBD	Oracle VM テンプレート	MySQL Cluster
可用性				
プラットフォームサポート	MySQL Server がサポートするすべて (https://www.mysql.com/support/supportedplatforms/database.html)	Linux	Oracle Linux	MySQL Cluster がサポートするすべて (https://www.mysql.com/support/supportedplatforms/cluster.html)
IP フェイルオーバーの自動化	いいえ	はい	はい	コネクタおよび構成に依存
データベースフェイルオーバーの自動化	いいえ	はい	はい	はい

要件	MySQL レプリケーション	DRBD	Oracle VM テンプレート	MySQL Cluster
データ再同期の自動化	いいえ	はい	N/A - 共有ストレージ	はい
代表的なフェイルオーバー時間	ユーザー/スクリプトに依存	構成に依存、60 秒以上	構成に依存、60 秒以上	1 秒以下
同期レプリケーション	いいえ、非同期および準同期	はい	N/A - 共有ストレージ	はい
共有ストレージ	いいえ、分散型	いいえ、分散型	はい	いいえ、分散型
地理的冗長性のサポート	はい	はい、MySQL レプリケーション	はい、MySQL レプリケーション	はい、MySQL レプリケーション
更新スキーマオンライン	いいえ	いいえ	いいえ	はい
拡張性				
ノード数	1 つのマスター、複数のスレーブ	1 つのアクティブ (プライマリ) ノード、1 つのパッシブ (セカンダリ) ノード	1 つのアクティブ (プライマリ) ノード、1 つのパッシブ (セカンダリ) ノード	255
組み込みロードバランス	読み取り、MySQL レプリケーション経由	読み取り、MySQL レプリケーション経由	読み取り、MySQL レプリケーション経由およびフェイルオーバー中	はい、読み取りおよび書き込み
読み取りの多いワークロードをサポート対応	はい	はい	はい	はい
書き込みの多いワークロードをサポート対応	はい、アプリケーションレベルシャーディングで	はい、アプリケーションレベルシャーディング - 複数のアクティブ/パッシブペアで	はい、アプリケーションレベルシャーディング - 複数のアクティブ/パッシブペアで	はい、自動シャーディングで
スケールオンライン (ノードの追加、再パーティション化など)	いいえ	いいえ	いいえ	はい

16.1 Oracle VM Template for MySQL Enterprise

仮想化は、クラウドコンピューティングの基盤を提供しながらデータセンターの効率性と高可用性を実現できる重要な技術です。MySQL Enterprise Edition を Oracle Linux、Oracle VM テンプレートに統合することが、仮想化された MySQL インスタンスをプロビジョニングするためのもっとも高速かつ簡単で信頼性の高い方法であり、ユーザーは高可用性サービスに対する膨大の重要を満たすことができます。

Oracle VM テンプレートは迅速な配備が可能で、手動による構成作業をなくすことができます。Oracle Linux および Oracle VM で動作し、本番使用向けに認定された、インストール済みかつ事前構成済みの仮想化された MySQL 5.5 Enterprise Edition ソフトウェアイメージを提供します。MySQL ソフトウェアイメージは、広範な統合および品質保証テストを開発プロセスの一部として受けてきました。

迅速なプロビジョニングに加えて、MySQL ユーザーは、厳しい SLA (サービスレベル契約) の要求を以下と組み合わせて組織が実現するように設計された Oracle VM の統合された高可用性機能のメリットも受けることができます。

- 障害からの自動リカバリ。Oracle VM は、物理サーバー、VM、または MySQL データベースの停止後に、サーバープールの使用可能なサーバーで障害が起きたインスタンスを自動的に再起動します。
- ライブ移行。運用スタッフは、MySQL の動作中インスタンスを保守作業中にサーバープール内の代替ホストに移動できます。

Oracle VM Template for MySQL Enterprise の作成、配備、および使用に関する説明は、次から入手できます。

- Oracle VM Template for MySQL Enterprise のホワイトペーパー: http://www.mysql.com/why-mysql/white-papers/mysql_wp_oracle-vm-template-for-mee.php.

- テンプレートのダウンロードに添付される README ファイル。

Oracle VM Template for MySQL Enterprise をダウンロードするには、https://edelivery.oracle.com/oraclevmに移動して次の指示に従ってください。

- 登録情報 (氏名、会社名、電子メールアドレス、国名) を入力してダウンロードの承諾をクリックします。
- 「Select a Product Pack」プルダウンメニューから「Oracle VM Templates」を選択して、「Go」をクリックします。
- Oracle VM Template のリストから MySQL Enterprise を選択します。
- ファイルをダウンロードして解凍し、その後の指示について README を参照してください。

16.2 DRBD/Pacemaker/Corosync/Oracle Linux を使用した MySQL の概要

DRBD (Distributed Replication Block Device) は MySQL HA (高可用性) の主要ソリューションの 1 つです。Pacemaker および Corosync を合わせて使用すると、ユーザーは次のことを実現できます。

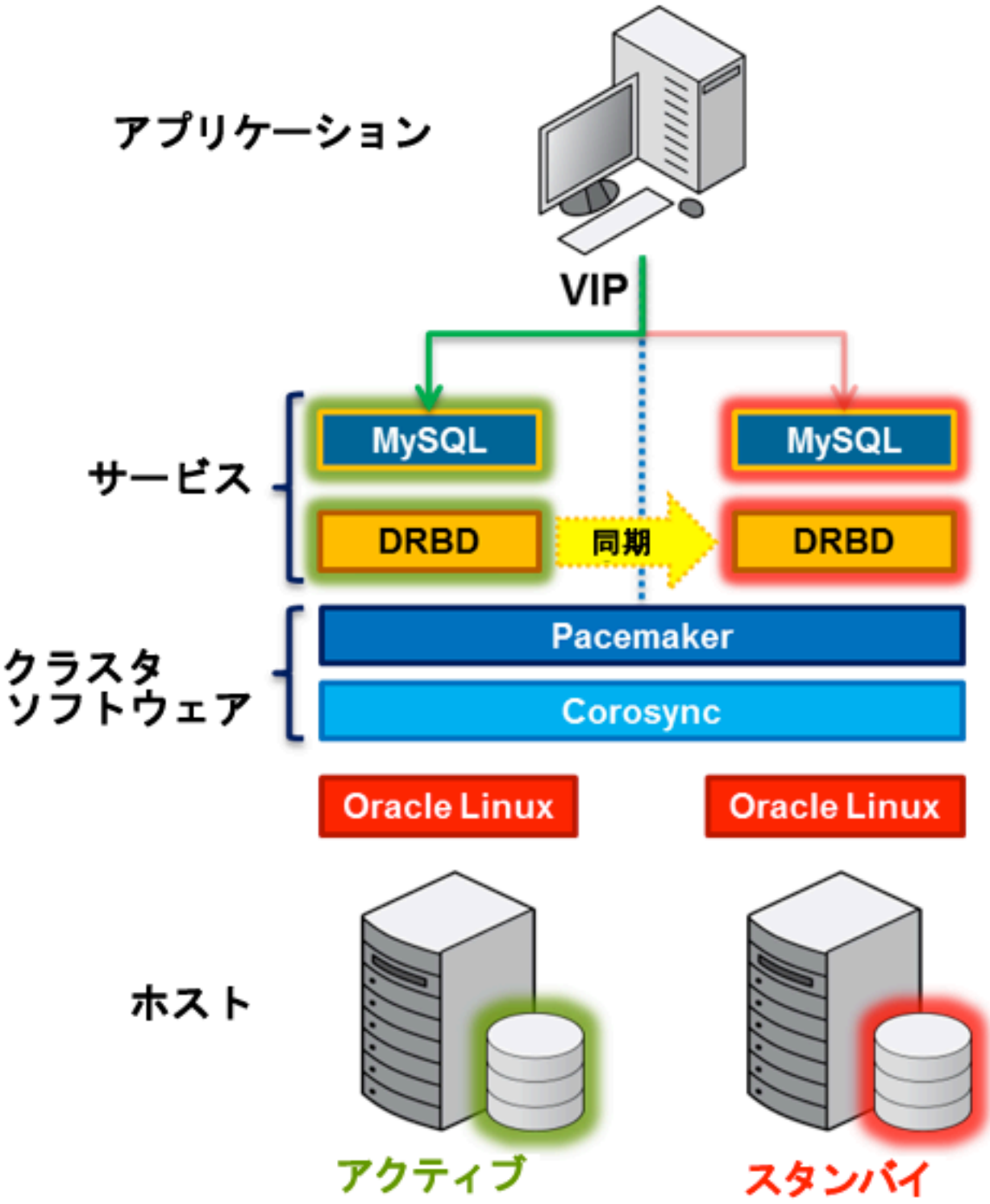
- 成熟した実証済みオープンソーステクノロジーのエンドツーエンド統合スタック、これを Oracle が完全にサポート (MySQL Enterprise Edition の一部として)。
- 自動フェイルオーバーおよびリカバリによるサービス継続性。
- 同期レプリケーションによるミラーリング、確定されたトランザクションを失うリスクなしにノード間フェイルオーバー。
- コモディティーハードウェアから HA クラスタを構築、共有ストレージは不要。

次の図では、MySQL サービスに高可用性レベルを実現するために使用できるスタックを示します。

もっとも低いレベル 2 では、物理的な冗長性を提供するために 2 つのホストが必要です。仮想環境を使用する場合、これらの 2 つのホストは別の物理マシン上に置くことをお勧めします。共有ストレージの必要がない点が重要な特徴です。いずれかの時点で、サービスは一方のホストではアクティブであり、もう一方のホストではスタンバイモードです。

Pacemaker および Corosync は、サービスと基盤となるホスト/オペレーティングシステムとの間に位置するクラスタリングレイヤーを提供するために結合されます。Pacemaker はサービスの起動と停止を担っており、それらが正確に 1 つのホスト上で動作することを保証することで、高可用性を実現してデータの破損を回避します。Corosync は、ノード間に Pacemaker がジョブを実行するための基盤となるメッセージングインフラストラクチャーを提供し、さらにクラスタ内のノードメンバーシップを処理し、変更を Pacemaker に通知します。

図 16.2 MySQL、DRBD、Pacemaker、および Corosync スタック



Pacemaker コアプロセスには、管理すべき特定サービスの知識は組み込まれていません。代わりに、サービス固有のアクションにラッパーを提供するエージェントを使用します。たとえば、このソリューションでは、仮想 IP アドレス、MySQL、および DRBD にエージェントを使用します。これらはすべて既存のエージェントであり、Pacemaker にパッケージされます。

この構成で Pacemaker が管理する必須のサービスは、DRBD、MySQL、およびアクティブな MySQL サービスに接続するためにアプリケーションが使用する仮想 IP アドレスです。

DRBD は、ブロック型デバイス (一般的に、スピニングディスクまたは半導体ディスク) のデータをアプリケーション、データベース、さらにファイルシステムに透過的に同期します。DRBD は、ext3 や ext4 などのジャーナリングファイルシステムを使用することを要求します。このソリューションのために、アクティブ-スタンバイモードで動作します。任意の時点で、DRBD が管理するディレクトリには、2 つのホストの正確に 1 つから読み書きのためにアクセスでき、もう一方には (読み取りであっても) アクセスできません。アクティブホストに行われた変更は、DRBD によって同期的にスタンバイホストに複製されます。

完全な MySQL および DRBD スタックをインストール、構成、プロビジョニング、およびテストするための詳細な説明については、次のガイドをダウンロードしてください。

- MySQL データベース。
- DRBD カーネルモジュールおよびユーザーランドユーティリティー。
- Pacemaker および Corosync クラスタメッセージングおよび管理プロセス。
- Oracle Linux オペレーティングシステム。

http://www.mysql.com/why-mysql/white-papers/mysql-high-availability-drbd-configuration-deployment-guide/からガイドをダウンロードしてください。

DRBD のサポート

MySQL 用の完全な DRBD スタックは Oracle によって認定されました。問題がオペレーティングシステム、DRBD、クラスタリングソフトウェア、または MySQL に関係するかどうかにかかわらず、スタック全体に対して 1 つの契約を提供する商用サポートは、MySQL Enterprise Edition および Oracle Linux Premier Support の両方の契約をしたユーザーが利用できます。

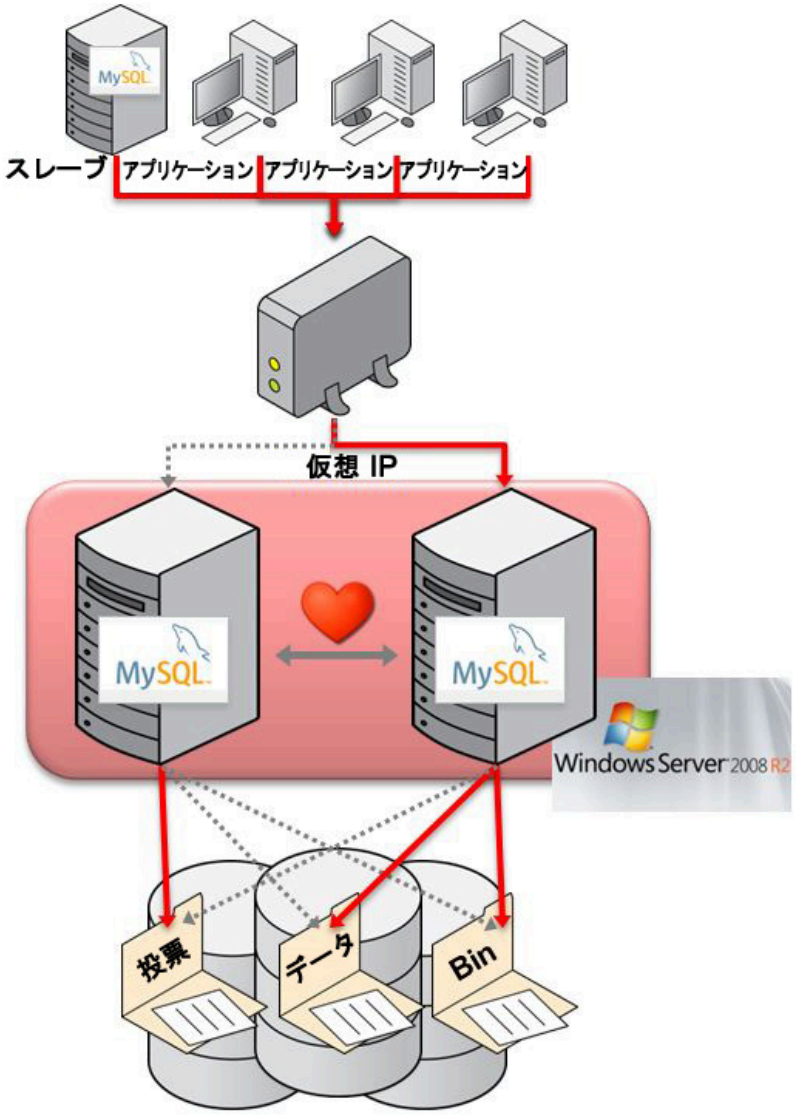
16.3 Windows フェイルオーバークラスタリングを使用した MySQL の概要

Microsoft Windows は、MySQL ユーザーコミュニティーの調査によると、常に MySQL の一番の開発環境としてランクされています。

MySQL Enterprise Edition は、Windows Server 2008 R2 フェイルオーバークラスタリング (WSFC) で認定され、サポートされているため、組織は Microsoft ネイティブの Windows クラスタリングサービスを使用して高いレベルの可用性を要求するビジネスクリティカルアプリケーションを安全に配備できます。

次の図は、高可用性サービスを提供するために MySQL と Windows Server フェイルオーバークラスタリングとの統合を示しています。

図 16.3 Windows Server フェイルオーバークラスタリングを使用した代表的な MySQL HA 構成



このアーキテクチャーでは、MySQL はアクティブ/パッシブ構成で配備されています。MySQL または基盤となるサーバーのいずれかで発生した障害は、自動的に検出され、MySQL インスタンスはパッシブノードで再起動されます。データベースにアクセスするアプリケーションや MySQL レプリケーションスレーブは、MySQL のリカバリが完了し、接続の受け付けを開始したあとに、同じ仮想 IP アドレスを使用して新しい MySQL に自動的に再接続できます。

Windows フェイルオーバークラスタリングを使用する MySQL には、クラスタ内に少なくとも 2 つのサーバーと共有ストレージ (たとえば、FC-AL SAN または iSCSI ディスク) が必要です。

MySQL バイナリとデータファイルは共有ストレージに格納され、Windows フェイルオーバークラスタリングは任意の時点で 1 つのクラスタノードだけがこれらのファイルにアクセスすることを保証します。

クライアントは、仮想 IP アドレス (VIP) を介して MySQL サービスに接続します。フェイルオーバーが発生した場合、多少の接続口スは生じますが、フェイルオーバーが発生したときにアクティブであったトランザクションの失敗に対処する以外に、フェイルオーバーの発生を意識する必要はありません。

Windows Server フェイルオーバークラスタリングを使用する MySQL の構成について詳しくは、http://www.mysql.com/why-mysql/white-papers/mysql_wp_windows_failover_clustering.phpに投稿されたホワイトペーパーを参照できます。

Windows Server フェイルオーバークラスタリングの背景情報と使用情報については、Microsoft Technet サイトの該当ページを参照してください。

- フェイルオーバークラスタリング

- Windows Server 2008 R2 のフェイルオーバークラスタリング

16.4 Amazon EC2 インスタンスでの MySQL の使用

Amazon Elastic Compute Cloud (EC2) サービスは、MySQL を含むさまざまな異なるアプリケーションとサービスを実行するために構築および配備できる仮想サーバーを提供します。EC2 サービスは Xen フレームワークをベースにしており、Amazon Machine Image (AMI) と呼ばれる仮想マシンの個々のインスタンスを実装する x86 Linux ベースプラットフォームをサポートしています。作成した AMI インスタンスへの完全な (ルート) アクセス権限を持つため、選択する任意の方法で構成したりインストールしたりできます。

EC2 を使用するには、使用する予定の構成とアプリケーションに基づく AMI を作成し、Amazon Simple Storage Service (S3) に AMI をアップロードします。S3 リソースから、EC2 環境でインスタンスとして実行する AMI の 1 つまたは複数のコピーを配備できます。EC2 環境は、インスタンスの管理と制御、および動作中のインスタンスに関するコンテキスト情報を提供します。

AMI、構成、およびアプリケーションを作成して管理できるため、選択した任意の環境を配備して作成できます。これには、基本的な MySQL サーバーに加えて、EC2 環境の利点を引き出すための拡張性に優れたレプリケーション、HA および拡張性シナリオ、さらに MySQL サービスとアプリケーションの要求の成長に合わせてインスタンスを追加配備できる機能が含まれています。

作業の配備と配布を支援するため、Amazon EC2 では小 (m1.small で特定)、大 (m1.large)、特大 (m1.xlarge) の異なる 3 つのインスタンスを利用できます。これらの異なるタイプは、EC2 Compute Unit (ECU) で測定された異なるレベルの計算能力を提供します。異なるインスタンス構成のサマリーを次の表に示しています。

EC2 属性	小	大	特大
プラットフォーム	32 ビット	64 ビット	64 ビット
CPU コア	1	2	4
ECU	1	4	8
RAM	1.7G バイト	7.5G バイト	15G バイト
ストレージ	150G バイト	840G バイト	1680G バイト
I/O パフォーマンス	中	高	高

EC2 環境で MySQL を配備して使用するための代表的なモデルは、データベースデータとアプリケーションを保持するために使用できる基本的な AMI を作成することです。データベースとアプリケーションの基本環境が作成されると、適切なインスタンスへの AMI の配備を選択できます。小から大または特大 EC2 インスタンスへ再配備できる AMI を所有することの柔軟性によって、アプリケーションまたはデータベーススタックを再構築しなくてもハードウェア環境を容易にアップグレードできます。

EC2 インストールで MySQL をセットアップおよびインストールする方法や、実行中のインスタンスにデータを移植および移行する方法など、EC2 で MySQL を始める際は、セクション16.4.1「EC2 AMI での MySQL のセットアップ」を参照してください。

レプリケーションのセットアップに関するガイドを含む、拡張可能な EC2 環境の作成方法に関するヒントとアドバイスについては、セクション16.4.3「EC2 を使用した MySQL データベースの配備」を参照してください。

16.4.1 EC2 AMI での MySQL のセットアップ

MySQL を実装する EC2 AMI をセットアップするには、Amazon が提供する事前構成済み AMI を使用する方法を含め、さまざまな方法があります。

Amazon が提供するデフォルト Getting Started AMI は Fedora Core 4 を使用し、yum を使用して MySQL をインストールできます。

```
shell> yum install mysql
```

これは、MySQL サーバーと、Perl DBI API の Perl DBD::mysql ドライバの両方をインストールします。

代わりに、標準インストール内で MySQL をインクルードする AMI の 1 つを使用することもできます。

さらに、MySQL Web サイトからダウンロードされた標準版 MySQL をインストールすることもできます。インストールのプロセスと手順は、Linux での MySQL のほかのインストールと同じです。第2章「MySQL のインストールと更新」を参照してください。

MySQL の標準構成では、データファイルはデフォルトの場所 /var/lib/mysql に置かれます。EC2 インスタンスのデフォルトデータディレクトリは /mnt です (大および特大のインスタンスでは、この構成を変更できます)。/etc/my.cnf を編集して、より大きなストレージ領域を指すように datadir オプションを設定する必要があります。

> **重要**
>
> EC2 インスタンスでメインの保管場所をはじめて使用するときには、初期化が必要です。初期化プロセスは、デバイスに最初に書き込んだときに自動的に開始されます。デバイスはすぐに使い始めることができますが、新しいデバイスの書き込みパフォーマンスは初期化プロセスが終了するまで初期の書き込みでかなり低下します。
>
> 新しいインスタンスをセットアップするときにこの問題を回避するには、MySQL データベースに入力する前に初期化プロセスを開始することをお勧めします。これを行う 1 つの方法は、dd を使用してファイルシステムに書き込むことです。
>
> ```
> root-shell> dd if=/dev/zero of=initialize bs=1024M count=50
> ```
>
> 上記では、ファイルシステムに 50G バイトを作成し、初期化プロセスを開始します。プロセスが終了したらファイルを削除します。
>
> 初期化プロセスは時間がかかる場合があります。小インスタンスで、初期化には 2 時間から 3 時間かかります。大および特大デバイスの場合、初期化にそれぞれ 10 時間または 20 時間かかる場合があります。

MySQL データファイルの正しい保管場所を構成することに加えて、配備のインスタンス構成を保存する前に、インスタンスに次のほかの設定をセットアップすることも検討してください。

- MySQL サーバー ID を設定します。これをレプリケーションに使用すると、ID 情報が正しく設定されます。
- バイナリロギングを有効にします。サーバーの起動および停止なしでレプリケーションを初期化できます。
- ストレージエンジンのキャッシュとメモリーのパラメータを設定します。EC2 環境で使用するストレージエンジンに制約または制限はありません。構成を選択してください (たぶん、配備する予定のインスタンスに適した MySQL で提供される標準構成の 1 つを使用して)。大および特大インスタンスには、キャッシュ専用の RAM が割り当てられます。アプリケーションスタックの一部としてサーバーに memcached をインストールすることを選択する場合は、MySQL と memcached の両方に十分なメモリーが存在している必要があります。

MySQL を実装する AMI と残りのアプリケーションスタックを構成したあとに、インスタンスを配備して再利用できるように AMI を保存してください。

アプリケーションスタックが AMI に構成されたあと、mysqldump を使用してデータベースのダンプを作成し、ダンプを EC2 インスタンスに転送してから、EC2 インスタンスデータベースに情報を再ロードすることで、MySQL データベースへのデータ入力を実行するようにしてください。

本番環境のアプリケーションでインスタンスを使用する前に、EC2 インスタンス環境の制限を確認してください。セクション16.4.2「EC2 インスタンスの制限」を参照してください。MySQL AMI を使い始める前に、配備に関する説明を参照してください。セクション16.4.3「EC2 を使用した MySQL データベースの配備」を参照してください。

16.4.2 EC2 インスタンスの制限

アプリケーションを配備する前に EC2 インスタンスの次の制限について確認してください。これらが Amazon EC2 環境内に配備できるかどうかに影響することはないはずですが、これらによってアプリケーションをサポートする環境をセットアップおよび構成する方法が変わる場合があります。

- インスタンス内に格納されたデータは永続的ではありません。インスタンスを作成してインスタンスにデータを移入すると、データはマシンの動作中のみ所定の場所にありますが、リブート後は失われます。インスタンスをシャットダウンすると、そこに含まれるデータは失われます。

  情報を失わないようにするには、mysqldump を使用して定期的にバックアップを取ってください。格納されているデータが重要である場合、障害に備えてデータの「ライブ」バックアップを保持するために、レプリケーションの使用を検討してください。バックアップを作成するときは、オフサイトにデータをコピーするときに適用される転送課金を避けるため、データを Amazon S3 サービスに書き込んでください。

- EC2 インスタンスは永続的ではありません。インスタンスが動作しているハードウェアで障害が発生した場合、そのインスタンスはシャットダウンされます。これにより、データまたはサービスの損失が生じる可能性があります。

  ただし、EBS を使用する場合、EBS ストレージボリュームを EC2 インスタンスに接続できるので、その EBS ボリュームは永続的になります。ディスクと同じように、EBS ボリュームにも障害が発生することがありますが、ボリュームのポイントインタイムスナップショットを作成できます。スナップショットは Amazon S3 に永続的であり、ボリュームに障害が発生した場合のデータのリストアに使用できます。

- EC2 インスタンスを非 EC2 環境に複製するときは、EC2 サービスに対する転送コストを確認してください。異なる EC2 インスタンス間のデータ転送は無料であるため、EC2 環境でレプリケーションを使用しても追加料金は発生しません。
- 一部の HA 機能は、直接サポートされていないか、またはその有用性を損なう可能性のある制限要因や問題点を抱えています。たとえば、DRBD または MySQL Cluster の使用が機能しない場合があります。デフォルトストレージ構成も冗長ではありません。ソフトウェアベースの RAID を使用して冗長性を改善できますが、これは追加のパフォーマンス負荷を伴います。

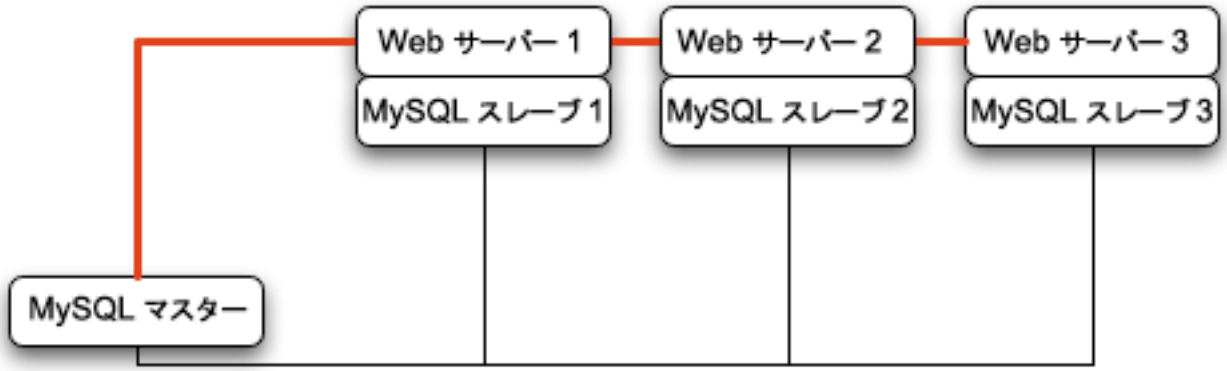
16.4.3 EC2 を使用した MySQL データベースの配備

EC2 インスタンスの稼働時間と可用性を保証することはできませんが、EC2 環境に MySQL を配備するときは、EC2 インスタンスに作業を簡単に配布できる方法を使用してください。これにはいくつかの方法があります。シャーディング技術 (複数のサーバーにアプリケーションを分割して、データセットとユーザーの特定のブロックを異なるサーバーに割り当てる) を使用することが、これを効果的に行う方法です。一般的なルールとして、インスタンスをより大きなマシンにアップグレードするよりも、より多くの EC2 インスタンスを作成してより多くのユーザーをサポートする方が簡単です。

EC2 アーキテクチャーは、EC2 インスタンスを長期間の高可用性ソリューションとしてではなく、一時的なキャッシュベースソリューションとして扱うときに最適に機能します。複数のマシンを使用することに加えて、memcached などのほかのサービスの利点を活用してアプリケーションに追加のキャッシュを提供することは MySQL サーバー上の負荷を低減し、書き込みに集中するのに役立ちます。EC2 内の大および特大インスタンスでは、使用可能な RAM がデータのために大きなメモリーキャッシュを提供できます。

独自のハードウェアで使用するほとんどのタイプのスケールアウトトポロジは EC2 環境内で使用および適用できます。ただし、すでに説明した制限とアドバイスを使用して、潜在的な障害によってデータが失われないようにしてください。また、各 EC2 インスタンスの相対能力が非常に低いため、シャーディングを使用するようにアプリケーションを変更したり、EC2 インスタンスを追加してアプリケーションのパフォーマンスを改善したりできるようにしてください。

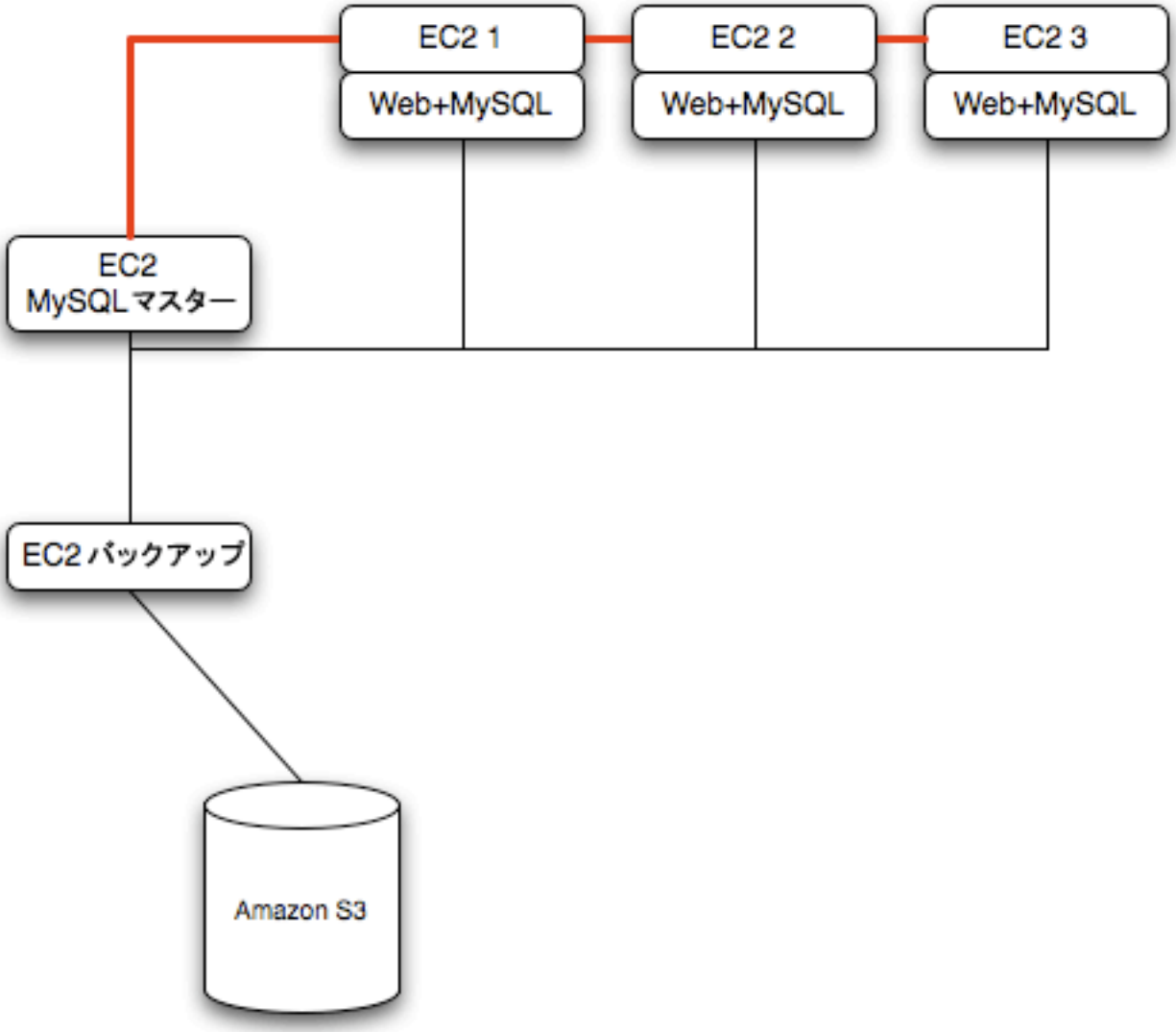
たとえば、次に示す典型的なスケールアウト環境を見てください。ここでは、単一マスターが 1 つ以上のスレーブ (この例では 3 つ) に複製され、Web サーバーが各レプリケーションスレーブ上で動作しています。



この構造は、マスター用に 1 つの EC2 インスタンス、Web および MySQL スレーブサーバーごとに 1 つのインスタンスを使用することで、EC2 環境に完全に再現できます。

> 注記
>
> EC2 環境内では、EC2 インスタンスが使用する内部 (プライベート) IP アドレスは一定です。インスタンス間で通信するときは、常にこれらの内部アドレスと名前を使用します。パブリック IP アドレスを使用するのは、外部と通信する場合 (たとえば、アプリケーションを公開する場合) だけです。

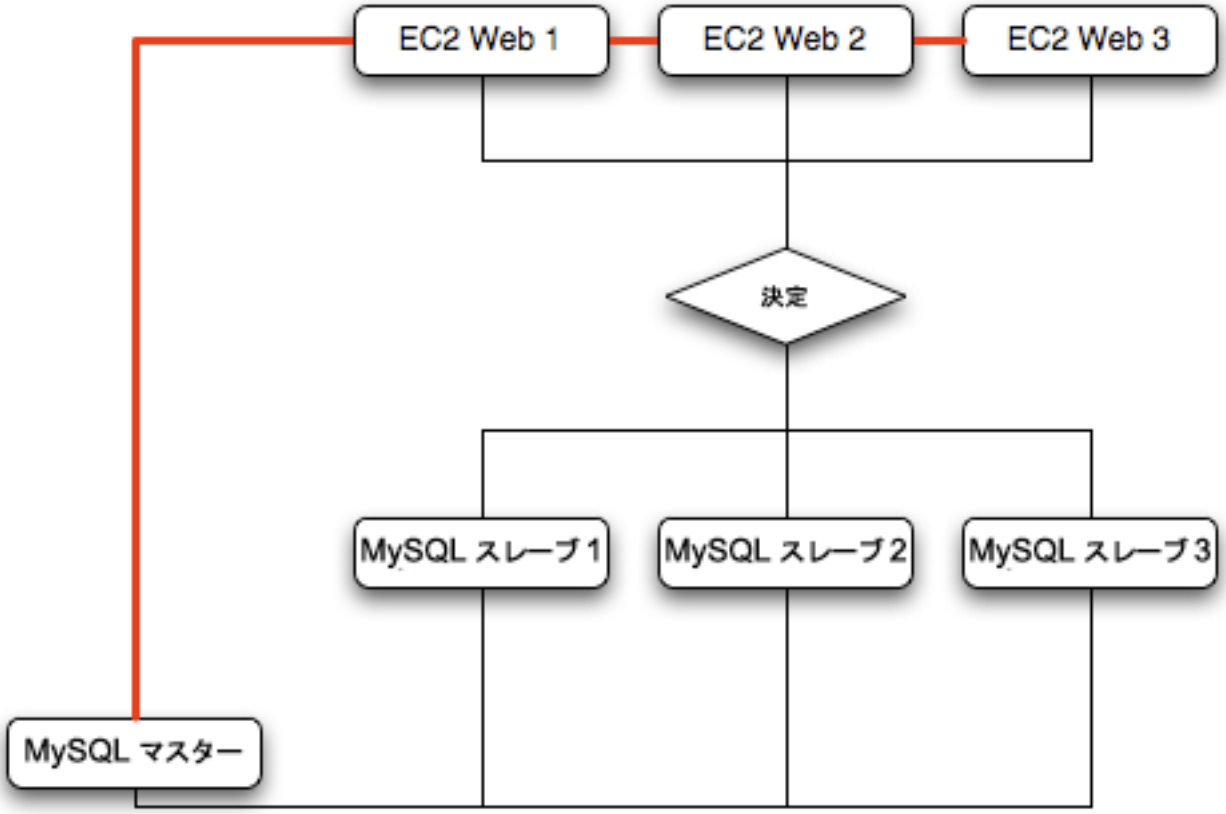
データベースの信頼性を確保するには、アクティブなバックアップおよびストレージの割り当てるレプリケーションスレーブを 1 つ以上 Amazon S3 施設に追加します。次のトポロジでこれの例を見ることができます。



EC2 インスタンスで memcached を使用すると、パフォーマンスが向上するはずです。大および特大インスタンスには、非常に多くの RAM があります。アプリケーションで memcached を使用するには、データベースから情報をロードするときに、まず項目がキャッシュ内に存在するかどうかを確認します。探しているデータがキャッシュ内に存在する場合は、それを使用します。そうでない場合は、データベースからデータを再ロードしてキャッシュに移入します。

シャーディング は、該当するグループに応じて一意のデータセットを提供するために、個々のマシンまたはマシングループに割り当てることでデータベース全体のデータを分割します。たとえば、名字が文字 A - D で終わるすべてのユーザーを単一サーバーに置きます。ユーザーがアプリケーションに接続して名字が既知の場合は、クエリーを該当する MySQL サーバーにリダイレクトできます。

EC2 でシャーディングを使用する場合、Web サーバーと MySQL サーバーを個別の EC2 インスタンスに分けてから、シャーディング決定ロジックをアプリケーションに適用します。データアクセスにどの MySQL サーバーを使用すべきかを知っていると、クエリーを該当するサーバーに分配してください。次の図でこれの例を見ることができます。



警告

シャーディングと EC2 を使用する場合、インスタンスの障害の潜在性がアプリケーションに影響しないようにしてください。特定のシャードに MySQL サーバーを提供する EC2 インスタンスが失敗した場合、そのシャード上のすべてのデータは使用できなくなります。

16.5 ZFS レプリケーションの使用

高可用性環境に対応するため、情報の即時コピーを現在アクティブなマシンとホットバックアップの両方に提供することは、HA ソリューションの重要な部分です。この問題には、第17章「レプリケーション」やセクション 16.2「DRBD/Pacemaker/Corosync/Oracle Linux を使用した MySQL の概要」を含めて多くのソリューションがあります。

ZFS ファイルシステムには、ファイルシステム内容のスナップショットを作成したり、スナップショットを別のマシンに転送したり、スナップショットを解凍してファイルシステムを再作成したりする機能が用意されています。スナップショットはいつでも作成でき、必要な数のスナップショットを作成できます。スナップショットを継続的に作成、転送、およびリストアすることで、DRBD に類似した方法で 1 つまたは複数のマシン間に同期を提供できます。

次の例では、/scratchpool にマウントされた、単一 ZFS プールで動作する単純な Solaris システムを示します。

```
Filesystem             size   used  avail capacity  Mounted on
/dev/dsk/c0d0s0        4.6G   3.7G   886M    82%    /
/devices                 0K     0K     0K     0%    /devices
ctfs                     0K     0K     0K     0%    /system/contract
proc                     0K     0K     0K     0%    /proc
mnttab                   0K     0K     0K     0%    /etc/mnttab
swap                   1.4G   892K   1.4G     1%    /etc/svc/volatile
objfs                    0K     0K     0K     0%    /system/object
/usr/lib/libc/libc_hwcap1.so.1
                       4.6G   3.7G   886M    82%    /lib/libc.so.1
fd                       0K     0K     0K     0%    /dev/fd
swap                   1.4G    40K   1.4G     1%    /tmp
swap                   1.4G    28K   1.4G     1%    /var/run
/dev/dsk/c0d0s7         26G   913M    25G     4%    /export/home
```

```
scratchpool           16G    24K    16G     1%    /scratchpool
```

MySQL データは /scratchpool 上のディレクトリに格納されます。いくつかの基本レプリケーション機能を説明するために、/scratchpool にはほかの項目も格納されています。

```
total 17
drwxr-xr-x  31 root     bin          50 Jul 21 07:32 DTT/
drwxr-xr-x   4 root     bin           5 Jul 21 07:32 SUNWmlib/
drwxr-xr-x  14 root     sys          16 Nov  5 09:56 SUNWspro/
drwxrwxrwx  19 1000     1000         40 Nov  6 19:16 emacs-22.1/
```

ファイルシステムのスナップショットを作成するには、プールおよびスナップショット名を指定して zfs snapshot を使用します。

```
root-shell> zfs snapshot scratchpool@snap1
```

すでに作成されたスナップショットの一覧を示すには:

```
root-shell> zfs list -t snapshot
NAME              USED  AVAIL  REFER  MOUNTPOINT
scratchpool@snap1     0      -  24.5K  -
scratchpool@snap2     0      -  24.5K  -
```

スナップショット自体はファイルシステムメタデータに格納され、それらを保持するために必要な容量は、スナップショットが作成される方法が理由で時間とともに変化します。スナップショットの初回の作成は、スナップショット全体を保持するために必要なデータとメタデータの全体コピーを作成する代わりに、ZFS はスナップショットが作成されたときのポイントインタイムのメタデータだけを記録するため、非常に高速です。

元のファイルシステムに加えられた変更が多くなるにつれ、古いブロックのレコードを保持するためにより多くの容量が必要になるため、スナップショットのサイズが増えます。多くのスナップショットを作成していて (たとえば 1 日に 1 つ)、週の早い時期に作成したスナップショットを削除する場合、新しい状態を構成する変更は、週を構成する 7 個のスナップショットに分散されるのではなく、より最近のスナップショットに含まれる必要があるため、新しい方のスナップショットのサイズが増える場合もあります。

スナップショットは、一般のファイルとしてではなくファイルシステムメタデータ内に存在するため、直接バックアップできません。スナップショットを別のファイルシステムやテープなどにコピーできる形式にするには、zfs send コマンドを使用してストリーム版スナップショットを作成します。

たとえば、スナップショットをファイルに書き出すには:

```
root-shell> zfs send scratchpool@snap1 >/backup/scratchpool-snap1
```

またはテープ:

```
root-shell> zfs send scratchpool@snap1 >/dev/rmt/0
```

zfs send を使用して、2 つのスナップショット間の増分変更を書き出すこともできます。

```
root-shell> zfs send scratchpool@snap1 scratchpool@snap2 >/backup/scratchpool-changes
```

スナップショットをリカバリするには、スナップショット情報を新しいファイルシステムまたは既存のものに適用する zfs recv を使用します。

16.5.1 ZFS を使用したファイルシステムレプリケーション

zfs send および zfs recv はストリームを使用してデータを交換するため、これらを使用して zfs send、ssh、および zfs recv を結合することで、あるシステムから別のシステムに情報を複製できます。

たとえば、scratchpool ファイルシステムのスナップショットを、新しいサーバー上の slavepool と呼ばれる新しいファイルシステムにコピーするには、次のコマンドを使用します。このシーケンスは、scratchpool のスナップショット、スレーブマシンへの転送 (ssh とログイン資格証明を使用)、および zfs recv を使用したスレーブでのスナップショットのリカバリを結合します。

```
root-shell> zfs send scratchpool@snap1 |ssh id@host pfexec zfs recv -F slavepool
```

パイプラインの最初の部分 zfs send scratchpool@snap1 はスナップショットをストリームします。ssh コマンドと、他方のサーバーでそれが実行するコマンド pfexec zfs recv -F slavepool は、ストリームされたスナップショットデータを受け取り、slavepool に書き込みます。この例では -F オプションを指定しました (スナップショットデータが強制的に適用されるため破壊的)。複製されたファイルシステムの最初のバージョンを作成しているため、これで問題ありません。

スレーブマシンでは、複製されたファイルシステムに正確に同じ内容が含まれています。

```
root-shell> ls -al /slavepool/
```

```
total 23
drwxr-xr-x   6 root     root           7 Nov  8 09:13 ./
drwxr-xr-x  29 root     root          34 Nov  9 07:06 ../
drwxr-xr-x  31 root     bin           50 Jul 21 07:32 DTT/
drwxr-xr-x   4 root     bin            5 Jul 21 07:32 SUNWmlib/
drwxr-xr-x  14 root     sys           16 Nov  5 09:56 SUNWspro/
drwxrwxrwx  19 1000     1000          40 Nov  6 19:16 emacs-22.1/
```

スナップショットが作成されたあとに、ファイルシステムを再度同期するには、新しいスナップショットを作成してから、zfs send の増分スナップショット機能を使用して 2 つのスナップショット間の変更をスレーブマシンに再度送ります。

```
root-shell> zfs send -i scratchpool@snapshot1 scratchpool@snapshot2 |ssh id@host pfexec zfs recv slavepool
```

この操作は、スレーブマシン上のファイルシステムがまったく変更されなかった場合にのみ成功します。変更された宛先ファイルシステムには増分変更を適用できません。上記の例では、ls コマンドがファイルまたはディレクトリの最終アクセス時間などのメタデータを変更することで問題を引き起こします。

スレーブファイルシステムでの変更を防ぐには、スレーブ上のファイルシステムを読み取り専用に設定します。

```
root-shell> zfs set readonly=on slavepool
```

readonly を設定することは、通常の方法ではスレーブ上のファイルシステムを変更できないことを意味します (ファイルシステムメタデータを含む)。一般的にメタデータを更新する操作 (ls のように) は、ファイルシステム状態の更新を試みることなく、サイレントにそれらの関数を実行します。

スレーブファイルシステムの本質は元のファイルシステムの静的コピーにすぎません。ただし、読み取り専用に構成されている場合でも、ファイルシステムはそれに適用されるスナップショットを持つことができます。読み取り専用に設定されているファイルシステムで、初期コピーを再実行します。

```
root-shell> zfs send scratchpool@snap1 |ssh id@host pfexec zfs recv -F slavepool
```

これで、元のファイルシステムに変更を行い、それらをスレーブに複製できます。

16.5.2 ZFS レプリケーションのための MySQL の構成

ソースファイルシステムで MySQL を構成するのは、複製する予定のファイルシステムでデータを作成する場合です。次の例の構成ファイルは /scratchpool/mysql-data をデータディレクトリとして使用するように更新されたので、テーブルを初期化できます。

```
root-shell> mysql_install_db --defaults-file=/etc/mysql/5.5/my.cnf --user=mysql
```

初期情報を同期するには、新しいスナップショットを実行してから、zfs send を使用して増分スナップショットをスレーブに送信します。

```
root-shell> zfs snapshot scratchpool@snap2
root-shell> zfs send -i scratchpool@snap1 scratchpool@snap2|ssh id@host pfexec zfs recv slavepool
```

slavepool の MySQL データディレクトリを調べることで、スレーブにデータがあることをダブルチェックします。

```
root-shell> ls -al /slavepool/mysql-data/
```

これで、MySQL を起動し、何らかのデータを作成し、zfs send/ zfs recv を使用して変更をスレーブに複製して変更を同期できます。

同期を実行する頻度は、アプリケーションと環境によって異なります。制限は、スナップショットを実行してからネットワークに変更を送信するために必要な速度です。

プロセスを自動化するには、スナップショットを実行し、送信し、操作を受信するスクリプトを作成し、cron を使用して特定の時間または間隔で変更を同期します。

16.5.3 ZFS での MySQL リカバリーの扱い

ZFS レプリケーションを使用してデータの定期コピーを提供するときは、元のシステムで障害が発生した場合に、手動または自動でテーブルをリカバリできることを確認してください。

障害が発生した場合、この順番に従ってください。

1. マスター上のスクリプトが起動されて動作中の場合は停止します。
2. スレーブファイルシステムを読み取り/書き込みに設定します。

```
root-shell> zfs set readonly=off slavepool
```

3. スレーブで mysqld を起動します。InnoDB を使用している場合、必要に応じて自動リカバリーを取得してテーブルのデータが正しいことを確認します (ここでは INSERT 途中のスナップショットから起動したことが示されています)。

```
InnoDB: The log sequence number in ibdata files does not match
InnoDB: the log sequence number in the ib_logfiles!
081109 15:59:59  InnoDB: Database was not shut down normally!
InnoDB: Starting crash recovery.
InnoDB: Reading tablespace information from the .ibd files...
InnoDB: Restoring possible half-written data pages from the doublewrite
InnoDB: buffer...
081109 16:00:03  InnoDB: Started; log sequence number 0 1142807951
081109 16:00:03 [Note] /slavepool/mysql-5.0.67-solaris10-i386/bin/mysqld: ready for connections.
Version: '5.0.67'  socket: '/tmp/mysql.sock'  port: 3306  MySQL Community Server (GPL)
```

InnoDB テーブルと定期的な同期スケジュールを使用して、重大なデータ損失のリスクを低減してください。MyISAM テーブルでは、REPAIR TABLE の実行が必要な場合があり、一部の情報を失った可能性もあります。

16.6 MySQL と memcached の併用

memcached は、使用可能な専用または予備の RAM にデータとオブジェクトを格納して、アプリケーションが解析のレイヤーやディスク I/O を介さずに高速にアクセスできるようにする、単純でスケーラビリティーの高いキーベースのキャッシュです。使用するには、1 台以上のホストで memcached コマンドを実行し、共有キャッシュを使用してオブジェクトを格納します。スレッドサポートの詳細は、セクション16.6.2「memcached の使用」を参照してください。

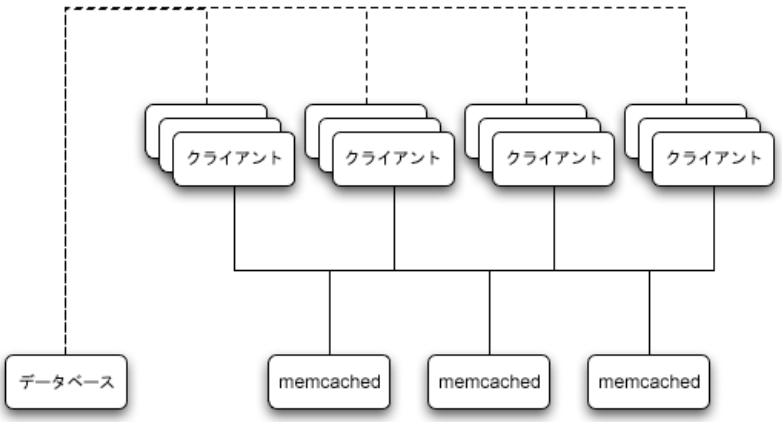
memcached を使用すると、次のような利点があります。

- すべての情報が RAM に格納されるため、ディスクから毎回情報をロードするよりもアクセス速度が向上します。
- キー/値ペアの「値」部分にデータ型の制約がないため、複雑な構造、ドキュメント、イメージ、またはそれらの組み合わせなどのデータをキャッシュできます。
- このインメモリーキャッシュを、一時的な情報を保持するため、またはデータベースにも格納される情報の読み取り専用キャッシュとして使用する場合、memcached サーバーの障害は重要ではありません。永続的なデータについては、データベースクエリーを使用する代替のルックアップ方法にフォールバックして、別のサーバー上の RAM にデータを再ロードできます。

一般的な使用環境では、memcached が提供するキャッシュから情報を読み取るようにアプリケーションを変更します。情報が memcached に存在しない場合は、MySQL データベースからデータがロードされ、キャッシュされたデータを利用して後続する同じオブジェクトの要求を処理できるように、キャッシュにデータが書き込まれます。

通常の配備レイアウトについては、図16.4「memcached アーキテクチャーの概要」を参照してください。

図 16.4 memcached アーキテクチャーの概要



この構造例では、どのクライアントもいずれかの memcached サーバーに接続して特定のキーを要求できます。各クライアントは、図に示したすべてのサーバーと通信するように構成されます。クライアントの内部で情報を格納するよう要求されると、データ参照に使用されるキーがハッシュ化され、このハッシュを使用していずれか

の memcached サーバーが選択されます。memcached サーバーの選択はサーバーに接続する前にクライアント上で行われるため、このプロセスは軽量のままです。

クライアントが同じキーを要求すると、同じアルゴリズムが再度使用されます。同じキーによって同じハッシュが生成され、同じ memcached サーバーがデータのソースとして選択されます。この方法を使用すると、キャッシュされたデータがすべての memcached サーバーに分散し、どのクライアントからもキャッシュされた情報にアクセスできるようになります。この結果、データベースからネイティブに情報を読み取る場合に比べてはるかに高速に情報 (特に複雑なデータや構造) を返すことができる分散型のメモリーベースのキャッシュが提供されます。

従来型の memcached サーバー内に保持されるデータはディスクには格納されず (RAM にのみ格納されます。つまり、データの永続性はありません)、RAM キャッシュは常にバッキングストア (MySQL データベース) から移入されます。memcached サーバーに障害が発生した場合は、常に MySQL データベースからデータをリカバリできます。

16.6.1 memcached のインストール

ソースコードから直接 memcached をビルドしてインストールできます。または、既存のオペレーティングシステムパッケージやインストールを使用できます。

バイナリ配布からの memcached のインストール

Red Hat または Fedora ホストに memcached をインストールするには、yum を使用します。

```
root-shell> yum install memcached
```

> **注記**
>
> CentOS では、別のソースから適切な RPM を入手できる場合があります。または、ソースの tarball を使用します。

Debian または Ubuntu ホストに memcached をインストールするには、apt-get を使用します。

```
root-shell> apt-get install memcached
```

Gentoo ホストに memcached をインストールするには、emerge を使用します。

```
root-shell> emerge install memcached
```

ソースからの memcached のビルド

ほかの Unix ベースのプラットフォーム (Solaris、AIX、HP-UX、および OS X を含む) とまだ説明されていない Linux 配布では、ソースからインストールする必要があります。Linux では、改良された epoll インタフェースを含む 2.6 ベースのカーネルが使用されているか確認してください。すべてのプラットフォームで、libevent 1.1 以降がインストールされていることを確認してください。libevent は、libevent の Web ページから入手できます。

memcached のソースは、memcached の Web サイトから入手できます。

memcached をビルドするには、次の手順に従います。

1. memcached ソースパッケージを抽出します。

   ```
   shell> gunzip -c memcached-1.2.5.tar.gz | tar xf -
   ```

2. memcached-1.2.5 ディレクトリに移動します。

   ```
   shell> cd memcached-1.2.5
   ```

3. configure を実行します

   ```
   shell> ./configure
   ```

   configure に指定できる追加のオプション:

   - --prefix

     別のインストールディレクトリを指定するには、--prefix オプションを使用します。

     ```
     shell> ./configure --prefix=/opt
     ```

     デフォルトでは /usr/local ディレクトリが使用されます。

- --with-libevent

  libevent がインストールされていて、configure でこのライブラリが見つからない場合は、--with-libevent オプションを使用してインストールされたライブラリの場所を指定します。

- --enable-64bit

  memcached の 64 ビットバージョン (大量の RAM を割り当てた単一インスタンスを使用できます) をビルドするには、--enable-64bit を使用します。

- --enable-threads

  memcached のマルチスレッドサポート (負荷が高いサーバーで応答時間が向上します) を有効にするには、--enable-threads を使用します。スレッドサポートを有効にするには、オペレーティングシステムの内部で POSIX スレッドがサポートされている必要があります。スレッドサポートの詳細は、セクション 16.6.2.7「memcached のスレッドサポート」を参照してください。

- --enable-dtrace

  memcached には、memcached インスタンスのモニターとベンチマークの実行に使用できる一連の DTrace スレッドが含まれています。詳細については、セクション16.6.2.5「memcached と DTrace の使用」を参照してください。

4. make を実行して memcached をビルドします。

```
shell> make
```

5. make install を実行して memcached をインストールします。

```
shell> make install
```

16.6.2 memcached の使用

memcached の使用を開始するには、1 台以上のサーバーで memcached サービスを起動します。memcached を実行すると、サーバーが設定され、メモリーが割り当てられ、クライアントからの接続の待機が開始されます。

> **注記**
>
> memcached を実行する際は、いずれかの特権 TCP/IP ポート (1024 以下) で待機する場合を除き、権限を持つユーザー (root) である必要はありません。ただし、setrlimit または同様の機能を使用してメモリー制限が設定されていないユーザーを使用する必要があります。

サーバーを起動するには、権限を持たない (つまり、root 以外の) ユーザーとして memcached を実行します。

```
shell> memcached
```

デフォルトでは、memcached は次の設定を使用します。

- 64M バイトのメモリー割り当て
- すべてのネットワークインタフェースでポート 11211 を使用して接続を待機します
- 最大で 1024 個の同時接続をサポートします

一般に、memcached の起動時に必要なオプションの完全な組み合わせを指定し、通常は memcached の初期化を処理する起動スクリプトを提供します。たとえば、次の行はキャッシュ用に最大 1024M バイトの RAM を割り当て、IP アドレス 192.168.0.110 のポート 11211 で待機し、バックグラウンドデーモンとして動作するように memcached を起動します。

```
shell> memcached -d -m 1024 -p 11211 -l 192.168.0.110
```

memcached がブート時に起動されるようにするには、init スクリプトと構成パラメータを確認してください。

memcached は次のオプションをサポートします。

- -u user

  root として memcached を起動する場合は、-u オプションを使用して memcached の実行用のユーザーを指定します。

```
shell> memcached -u memcache
```

- -m memory

  オブジェクトを格納するために memcached に割り当てるメモリーの量を設定します。デフォルトは 64M バイトです。

  キャッシュ用に割り当てるメモリーの量を増やすには、-m オプションを使用して割り当てる RAM の量 (メガバイト単位) を指定します。割り当てる RAM の量が多いほど、格納できるデータの量も多くなり、キャッシュの効率も向上します。

  > **警告**
  >
  > 使用可能な RAM より大きいメモリー割り当てを指定しないでください。指定した値が大きすぎると、memcached に割り当てられた RAM の一部に対して物理 RAM ではなくスワップ領域が使用されます。これにより、データが RAM に直接格納されず、ディスクにスワップされるため、値を格納および取得するときに遅延が発生する可能性があります。
  >
  > vmstat コマンドの出力を使用すると、次の free カラムに示されるように、空きメモリーを取得できます。

  ```
  shell> vmstat
  kthr      memory            page            disk          faults      cpu
  r b w   swap  free  re  mf pi po fr de sr s1 s2 -- --   in   sy   cs us sy id
  0 0 0 5170504 3450392 2  7  2  0  0  0  4  0  0  0  0  296   54  199  0  0 100
  ```

  たとえば、3G バイトの RAM を取得するには:

  ```
  shell> memcached -m 3072
  ```

  4G バイトの制限を超えるメモリーにアクセスするために PAE を使用する 32 ビット x86 システムでは、最大プロセスサイズを超える RAM を割り当てることはできません。これを回避するには、memcached の複数のインスタンスを実行し、それぞれ異なるポートで待機します。

  ```
  shell> memcached -m 1024 -p11211
  shell> memcached -m 1024 -p11212
  shell> memcached -m 1024 -p11213
  ```

  > **注記**
  >
  > すべてのシステム (特に 32 ビット) で、memcached アプリケーションとメモリー設定の両方に関して十分な余裕を確保してください。たとえば、4G バイトの RAM を搭載した専用の memcached ホストがある場合は、3500M バイトを超えるメモリーサイズを設定しないでください。これができなかった場合は、クラッシュや深刻なパフォーマンスの問題が発生する可能性があります。

- -l interface

  接続を待機するネットワークインタフェース/アドレスを指定します。デフォルトでは、使用可能なすべてのアドレス (INADDR_ANY) で待機します。

  ```
  shell> memcached -l 192.168.0.110
  ```

  IPv6 アドレスのサポートは memcached 1.2.5 で追加されました。

- -p port

  接続に使用する TCP ポートを指定します。デフォルトは 18080 です。

  ```
  shell> memcached -p 18080
  ```

- -U port

  接続に使用する UDP ポートを指定します。デフォルトは 11211 です。0 を指定すると UDP が無効になります。

  ```
  shell> memcached -U 18080
  ```

- -s socket

待機に使用する Unix ソケットを指定します。

クライアントと同じサーバーで memcached を実行する場合は、-s オプションを使用すると、ネットワークインタフェースが無効になり、ローカルの Unix ソケットを使用できます。

```
shell> memcached -s /tmp/memcached
```

Unix ソケットを使用すると、ネットワークサポートが自動的に無効になり、ネットワークポートが節約されます (Web サーバーやその他のプロセスでより多くのポートを使用できます)。

- -a mask

  Unix ソケットで使用するアクセスマスク (8 進数) を指定します。デフォルトは 0700 です。

- -c connections

  memcached サービスに対する同時接続の最大数を指定します。デフォルトは 1024 です。

  ```
  shell> memcached -c 2048
  ```

  このオプションは、(memcached サービスへの過負荷を避けるために) 接続数を減らすか、または接続数を増やして memcached サーバーを実行しているサーバーをより効果的に利用するために使用します。

- -t threads

  受信した要求を処理するときに使用するスレッドの数を指定します。

  デフォルトでは、memcached は 4 つの並列スレッドを使用するように構成されます。スレッドは、ロックシステムを使用して異なるスレッドによる同じ値の上書きや更新を防止することにより、キャッシュ内のデータの格納と取得のパフォーマンスを向上させます。スレッドの数を増減するには、-t オプションを使用します。

  ```
  shell> memcached -t 8
  ```

- -d

  memcached をデーモン (バックグラウンド) プロセスとして実行します。

  ```
  shell> memcached -d
  ```

- -r

  コアファイルの上限サイズを最大化します。これにより、障害が発生したときにメモリー領域全体が setrlimit で設定された上限までコアファイルとしてディスクにダンプされます。

- -M

  メモリーが使い果たされたときに、クライアントにエラーを返します。これは、キャッシュから古い項目を削除して新しい項目が入るようにする通常の動作に代わって適用されます。

- -k

  ページングされたすべてのメモリーをロックダウンします。これにより、新しい項目がキャッシュに格納されたときにメモリーの新しいスラブを割り当てるのではなく、使用前にメモリーが確保されます。

  > 注記
  >
  > ロックできるメモリーの量には、ユーザーレベルの制限があります。使用可能なメモリー量を超えて割り当てようとすると、失敗します。(-u user のユーザーではなく) デーモンを起動するときに使用したユーザーに対して制限を設定するには、シェル内で ulimit -S -l NUM_KB を使用します

- -v

  冗長モード。メインイベントループの実行中にエラーおよび警告を出力します。

- -vv

  高冗長モード。-v で出力される情報に加えて、個々のクライアントコマンドと応答も出力されます。

- -vvv

超冗長モード。-vv で出力される情報に加えて、内部の状態遷移も表示されます。

- -h

  ヘルプメッセージを出力して終了します。

- -i

  memcached と libevent のライセンスを出力します。

- -I mem

  memcached インスタンスに格納できるオブジェクトの最大サイズを指定します。このサイズは、単位の接尾辞 (キロバイトを表す k、メガバイトを表す m) をサポートします。たとえば、サポートされる最大オブジェクトサイズを 32M バイトに増やすには:

  ```
  shell> memcached -I 32m
  ```

  指定できる最大オブジェクトサイズは 128M バイト、デフォルトは 1M バイトです。

  このオプションは 1.4.2 で追加されました。

- -b

  バックログキューの制限を設定します。バックログキューによって、memcached による処理を待機できるネットワーク接続の数が構成されます。この制限を増やすと、クライアントが memcached インスタンスに接続できないエラーを受信する回数が減る可能性がありますが、サーバーのパフォーマンスは向上しません。デフォルトは 1024 です。

- -P pidfile

  memcached インスタンスのプロセス ID を file に保存します。

- -f

  チャンクサイズの増大係数を設定します。新しいメモリーチャンクを割り当てる場合、新しいチャンクの割り当てサイズはデフォルトのスラブサイズにこの係数を掛けて決定されます。

  大規模なテストを行わずにこのオプションの効果を確認するには、-vv コマンド行オプションを使用して、計算されたスラブサイズを表示します。詳細については、セクション16.6.2.8「memcached ログ」を参照してください。

- -n bytes

  キー + 値 + フラグ情報に割り当てられる最小領域。デフォルトは 48 バイトです。

- -L

  大規模メモリーページをサポートするシステムで、大規模メモリーページの使用を有効にします。大規模メモリーページを使用すると、memcached は 1 つの大きなチャンクに項目キャッシュを割り当てることができるようになり、メモリーへのアクセス時にミスの数が減ってパフォーマンスが向上する可能性があります。

- -C

  コンペアアンドスワップ (CAS) 操作の使用を無効にします。

  このオプションは memcached 1.3.x で追加されました。

- -D char

  デフォルトの文字をキープリフィクスと ID の区切りとして使用するように設定します。これは、プリフィクス単位の統計レポートで使用されます (セクション16.6.4「memcached の統計の取得」を参照してください)。デフォルトはコロン (:) です。このオプションを使用すると、統計収集が自動的にオンになります。使用しなかった場合は、stats detail on コマンドをサーバーに送信して統計収集を有効にできます。

  このオプションは memcached 1.3.x で追加されました。

- -R num

  イベントプロセスあたりの最大要求数を設定します。デフォルトは 20 です。

- -B protocol

  バインディングプロトコル (つまり、クライアント接続に対するデフォルトの memcached プロトコルサポート) を設定します。オプションは、ascii、binary、または auto です。自動 (auto) がデフォルトです。

  このオプションは、memcached 1.4.0 で追加されました。

16.6.2.1 memcached の配備

memcached を使用するときは、いくつかの異なる可能な配備戦略とトポロジを使用できます。使用する正確な戦略は、アプリケーションと環境によって異なります。システム内に memcached を配備するためのシステムを開発するときは、次の点に留意してください。

- memcached はキャッシュメカニズムに過ぎません。消失したり、別の場所からロードしたりできない情報の格納に使用しないでください。
- memcached プロトコルにはセキュリティーが組み込まれていません。最低でも、memcached を実行しているサーバーがネットワークの内部からのみアクセス可能であり、使用されるネットワークポートが (ファイアウォールなどを使用して) ブロックされることを確認してください。memcached サーバーに格納されている情報の機密性が高い場合は、その情報を memcached に格納する前に暗号化してください。
- memcached には、どのようなフェイルオーバーも用意されていません。これは、異なる memcached インスタンス間で通信が行われないためです。あるインスタンスに障害が発生した場合、アプリケーションはそのインスタンスをリストから削除し、データを再ロードして、別の memcached インスタンスにデータを書き込む必要があります。
- これらのタスクに異なる複数の物理マシンを使用すると、クライアントと memcached 間の待機時間に問題が発生する可能性があります。待機時間に問題が発生した場合は、memcached インスタンスをクライアントに移動してください。
- キーの長さは memcached サーバーによって決定されます。デフォルトの最大キーサイズは 250 バイトです。
- 特にクライアントが複数ある場合は、単一障害点を回避するため、少なくとも 2 つの memcached インスタンスを使用してください。できるかぎり多くの memcached ノードを作成するのが理想的です。memcached インスタンスをプールに追加したり、プールから削除したりすると、キー/値ペアのハッシュ化と分布に影響する可能性があります。問題を回避する方法については、セクション16.6.2.4「memcached のハッシュ化/分布タイプ」を参照してください。

16.6.2.2 名前空間の使用

memcached キャッシュは非常に単純かつ大規模なキー/値ストレージシステムであるため、データを異なるセクションに区分化する手段がありません。たとえば、MySQL データベースから返される一意の ID によって情報を格納する場合、2 つの異なるテーブルからデータを格納すると、同じ ID が両方のテーブルで有効になるため、問題が発生する可能性があります。

一部のインタフェースには、情報をキャッシュに格納するときに名前空間を作成するための自動メカニズムが用意されています。これらの名前空間は、実際には、値がキャッシュに格納されるたび、または値がキャッシュから取得されるたびに特定の ID の前に付けられる単なるプリフィクスです。

オブジェクトを記述するキーと、オブジェクトの格納時にキーの内部に指定する一意の識別子を使用することで、この同じ基本原則を実装できます。たとえば、ユーザーデータを格納するときは、ユーザーの ID に user: や user- のようなプリフィクスを付けます。

> **注記**
>
> 名前空間やプリフィクスを使用して制御されるのは、格納または取得されるキーだけです。memcached 内にはセキュリティーがないため、特定のクライアントのみが特定の名前空間を持つキーにアクセスできるようにする方法はありません。名前空間は、データを識別し、キー/値ペアの破損を防止する手段としてのみ有効です。

16.6.2.3 データ失効

memcached インスタンス内には、2 つのタイプのデータ失効があります。1 つ目のタイプは、新しいキー/値ペアが memcached インスタンスに格納された時点で適用されます。適切なスラブ内に値を格納する十分な領域がない場合は、既存のもっとも長い間使われていない (LRU) オブジェクトがキャッシュから削除 (消去) され、新しい項目用の領域が確保されます。

LRU アルゴリズムにより、アクティブに使用されなくなったオブジェクトや、データが古いかほとんど無価値である可能性がある長期間使用されていないオブジェクトが削除されます。ただし、memcached に割り当てられた

メモリーがキャッシュ内で定期的に使用される必須オブジェクトの数より小さいシステムでは、多くの項目がアクティブに使用されているにもかかわらず、失効してキャッシュから削除される可能性があります。消去 (失効するオブジェクト) のレベルをより適切に把握するには、統計メカニズムを使用します。詳細については、セクション16.6.4「memcached の統計の取得」を参照してください。

この消去動作を変更するには、memcached の起動時に -M コマンド行オプションを設定します。このオプションを使用すると、メモリーが使い果たされたときに、古いデータが自動的に消去されず、エラーが返されるようになります。

2 つ目のタイプの失効システムは、キャッシュにキー/値ペアが挿入されたとき、またはキャッシュから項目を削除するときに設定できる明示的なメカニズムです。失効時間を使用することで、キャッシュ内のデータが最新であり、アプリケーションのニーズと要件に合っていることを効果的に確認できます。

失効時間を明示的に設定する一般的なシナリオとして、ユーザーが Web サイトにアクセスしたときのセッションデータのキャッシュなどが考えられます。memcached は、設定された明示的な失効時間をオブジェクトが要求された現在の時間と比較する遅延失効メカニズムを使用します。失効していないオブジェクトだけが返されます。

キャッシュからオブジェクトを明示的に削除するときに失効時間を設定することもできます。この場合、失効時間は実際にはタイムアウトであり、特定のキーの値を設定しようとして拒否される期間を示します。

16.6.2.4 memcached のハッシュ化/分布タイプ

memcached クライアントインタフェースは、特定の memcached インスタンスのデータを設定または取得するときにどのホストを使用すべきかを決定するため、マルチサーバー構成で使用される異なる分布アルゴリズムをサポートします。値を取得または設定すると、指定されたキーからハッシュが作成され、構成済みサーバーのリストからホストを選択するために使用されます。ハッシュ化メカニズムは指定されたキーをハッシュのベースとして使用するため、設定操作と取得操作の両方で同じサーバーが選択されます。

このプロセスは次のように考えることができます。一連のサーバー (a、b、および c) があり、クライアントは格納または取得されるキーに基づく整数を返すハッシュ化アルゴリズムを使用します。生成された値を使用して、クライアントに構成されたサーバーリストからサーバーが選択されます。memcache クライアント内のもっとも標準的なクライアントハッシュ化では、構成された memcached サーバーの数で値を割る簡単なモジュラス計算が使用されます。このプロセスは、擬似コードで次のように要約できます。

```
@memcservers = ['a.memc','b.memc','c.memc'];
$value = hash($key);
$chosen = $value % length(@memcservers);
```

上記を値で置き換えると:

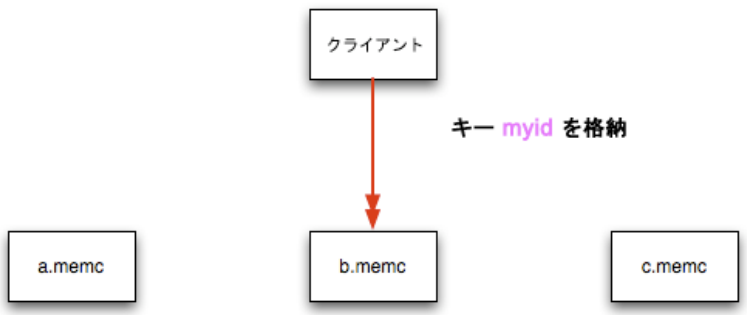
```
@memcservers = ['a.memc','b.memc','c.memc'];
$value = hash('myid');
$chosen = 7009 % 3;
```

上の例では、クライアントのハッシュ化アルゴリズムによってインデックス 1 (7009 % 3 = 1) のサーバーが選択され、そのサーバーでキーと値が格納または取得されます。

> **注記**
>
> この選択とハッシュ化のプロセスは、使用している memcached クライアントによって自動的に処理されます。ユーザーは使用する memcached サーバーのリストを提供するだけで済みます。

次の図16.5「memcached のハッシュ選択」は、これをグラフィカルに表現したものです。

図 16.5 memcached のハッシュ選択



同じハッシュと選択のプロセスは、memcached クライアント内の指定されたキーの操作で実行されます。

この方法を使用することで、いくつかのメリットが得られます。

- 接続するサーバーのハッシュ化と選択がクライアントの内部で完全に処理されます。これにより、接続する正しいマシンを決定するためにネットワーク通信を行う必要がなくなります。
- memcached サーバーの決定がクライアントの内部で完全に行われるため、実行される操作 (設定、取得、増分など) に関係なくサーバーを自動的に選択できます。
- この決定はクライアント内で処理されるため、ハッシュ化アルゴリズムは特定のキーに対して同じ値を返します。サーバー環境の違いによって値が影響を受けたり、リセットされたりすることはありません。
- 選択は非常に高速です。キー値に対するハッシュ化アルゴリズムは高速であり、その結果、使用可能なマシンの単純な配列からサーバーが選択されます。
- クライアント側のハッシュ化を使用することで、各 memcached サーバー間のデータの分布が簡略化されます。ハッシュ化アルゴリズムによって返される値の自然な分布では、使用可能なサーバー間でキーが自動的に分散されます。

クライアント内で構成されているサーバーリストが同じままであれば、格納されている同じキーによって同じ値が返されるため、同じサーバーが選択されます。

ただし、同じハッシュ化メカニズムを使用しないと、同じデータが別のインタフェースによって別のサーバーに記録され、memcached の領域が無駄になるだけでなく、情報の相違につながる可能性があります。

> 注記
>
> マルチインタフェース互換のハッシュ化メカニズムを使用する 1 つの方法は、libmemcached ライブラリと関連インタフェースを使用することです。異なる言語 (C、Ruby、Perl、および Python を含む) のインタフェースが同じクライアントライブラリインタフェースを使用するため、ID から常に同じハッシュコードが生成されます。

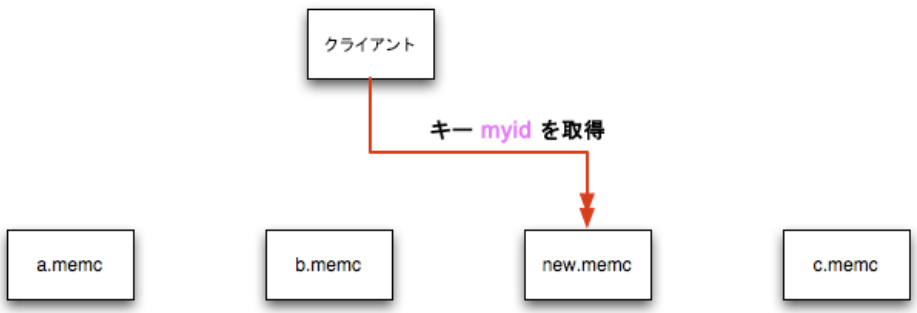
クライアント側でサーバーを選択する場合の問題は、memcached サーバーを使用する各クライアントでサーバーリスト (それらの順番を含む) の整合性を維持し、それらのサーバーを使用可能にしておく必要があることです。次のときに、キーに対して操作を実行しようとする場合:

- 使用可能なインスタンスのリストに新しい memcached インスタンスを追加した
- 使用可能なインスタンスのリストから memcached インスタンスを削除した
- memcached インスタンスの順序が変更された

特定のキーに対してハッシュ化アルゴリズムが使用されるときにサーバーリストが異なる場合は、ハッシュ計算によってリストから別のサーバーが選択される可能性があります。

次の例の new.memc のように、サーバーリストに新しい memcached インスタンスが追加された場合は、同じキー (myid) を使用する GET 操作によってキャッシュミスが発生します。これは、このキーから同じ値が算出され、サーバーの配列から同じインデックスが選択されるが、インデックス 2 はデータが本来格納されているサーバー c.memc ではなく新しいサーバーを指すようになったためです。このため、別の memcached インスタンスのキャッシュにそのキーが存在するにもかかわらず、キャッシュミスが発生します。

図 16.6 新しい memcached インスタンスを含む memcached のハッシュ選択



これは、c.memc と new.memc の両方にキー myid の情報が含まれることを意味しますが、このキーに対して各サーバーに格納される情報は各インスタンスで異なる可能性があります。より重要な問題は、新しいサーバーによってキーの分布が変わるにつれて、データ取得時のキャッシュミスの数が大幅に増加することです。さらに、この結果として memcached インスタンスにキャッシュされているデータを再構築する必要があるため、データベースの読み取り回数が増加します。

クライアントに構成されているサーバーリストをアクティブに管理し、構成済みの memcached インスタンスが使用可能として識別されたときに各インスタンスを追加および削除する場合も、同じ結果になる可能性があります。たとえば、ある memcached インスタンスが接続できなくなったことをクライアントが検出したときにそのインスタンスを削除すると、ここに示したようにサーバーの選択が失敗する可能性があります。

これによって重大な問題が発生してキャッシュが無効になることを防ぐため、サーバーの選択に使用するハッシュ化アルゴリズムを選択できます。ハッシュ化アルゴリズムには、整合とモジュラの 2 つの一般的なタイプがあります。

整合ハッシュ化アルゴリズムでは、構成済みサーバーのリストが変更された場合でも、同じキーをサーバーリストに適用すると、同じサーバーを使用してキーが格納または取得されます。これは、構成リストのサーバーを追加および削除しながら、特定のキーに対して常に同じサーバーを使用できることを意味します。使用可能な整合ハッシュ化アルゴリズムには、Ketama と Wheel の 2 つのタイプがあります。どちらのタイプも libmemcached によってサポートされ、PHP および Java 用の実装が入手可能です。

整合ハッシュ化アルゴリズムにはいくつかの制限があります。既存の構成済みサーバーのリストにサーバーを追加すると、通常の分布の一部として新しいサーバーにキーが分配されます。リストからサーバーを削除すると、そのキーはリスト内の別のサーバーに再割り当てされるため、キャッシュに情報を再度移入する必要があります。また、整合ハッシュ化アルゴリズムでは、複数のクライアント間でサーバーを一貫して選択する必要があるにもかかわらず、各クライアントに異なるサーバーリストが含まれているという問題は解決されません。整合性が適用されるのは、1 つのクライアントの内部だけです。

モジュラハッシュ化アルゴリズムでは、クライアントはハッシュを計算して構成済みサーバーリストからサーバーを選択することにより、サーバーを選択します。サーバーリストが変わると、モジュラハッシュ化アルゴリズムを使用して選択されるサーバーも変わります。この結果、前述の動作が発生します。サーバーリストの変更は、データの取得時に別のサーバーが選択されることを意味し、キャッシュに情報が再移入されるため、キャッシュミスとデータベース負荷の増大につながります。

各クライアントで memcached インスタンスを 1 つだけ使用する場合や、クライアントに構成された memcached サーバーのリストが一度も変更されていない場合は、ハッシュ化アルゴリズムの選択による大きな効果もないため、重要ではありません。

サーバーを定期的に変更する場合や、多数のクライアントで共有される共通のサーバーセットを使用する場合は、整合ハッシュ化アルゴリズムを使用することで、確実にキャッシュデータの重複を防ぎ、データを均等に分配しやすくなります。

16.6.2.5 memcached と DTrace の使用

memcached には、サーバー動作のモニターに使用できるいくつかの異なる DTrace プローブが含まれています。含まれているプローブは、キー/値ペアの追加、更新、または削除時の個々の接続、スラブ割り当て、およびハッシュテーブルの変更をモニターできます。

DTrace および DTrace スクリプトの作成の詳細は、『DTrace ユーザーガイド』をお読みください。

memcached 1.2.6 で追加された DTrace プローブのサポートには、アプリケーションをモニターしやすくするいくつかの DTrace プローブが含まれています。DTrace は、Solaris 10、OpenSolaris、OS X 10.5、および FreeBSD でサポートされます。memcached で DTrace プローブを有効にするには、ソースからビルドし、--enable-dtrace オプションを使用します。詳細については、セクション16.6.1「memcached のインストール」を参照してください。

memcached でサポートされるプローブは次のとおりです。

- conn-allocate(connid)

  接続プールから接続オブジェクトが割り当てられたときに起動します。

  - connid: 接続 ID。

- conn-release(connid)

  接続オブジェクトが接続プールに解放されたときに起動します。

引数:

- connid: 接続 ID。

- conn-create(ptr)

  新しい接続オブジェクトが作成される (つまり、接続プールに空き接続オブジェクトがない) ときに起動します。

  引数:

  - ptr: 接続オブジェクトへのポインタ

- conn-destroy(ptr)

  接続オブジェクトが破壊されるときに起動します。

  引数:

  - ptr: 接続オブジェクトへのポインタ。

- conn-dispatch(connid, threadid)

  メインまたは接続管理スレッドからワーカースレッドに接続がディスパッチされるときに起動します。

  引数:

  - connid: 接続 ID。
  - threadid: スレッド ID。

- slabs-allocate(size, slabclass, slabsize, ptr)

  スラブアロケータからメモリーを割り当てます。

  引数:

  - size: 要求されたサイズ。
  - slabclass: 割り当てはこのクラスで行われます。
  - slabsize: このクラスの各項目のサイズ。
  - ptr: 割り当てたメモリーへのポインタ。

- slabs-allocate-failed(size, slabclass)

  メモリーの割り当てに失敗しました (メモリー不足)。

  引数:

  - size: 要求されたサイズ。
  - slabclass: 要求を満たすことができなかったクラス。

- slabs-slabclass-allocate(slabclass)

  スラブクラスに追加の領域が必要なときに起動します。

  引数:

  - slabclass: 追加のメモリーを必要とするクラス。

- slabs-slabclass-allocate-failed(slabclass)

  メモリーの割り当てに失敗しました (メモリー不足)。

  引数:

  - slabclass: 追加のメモリーを取得できなかったクラス。

- slabs-free(size, slabclass, ptr)

  メモリーを解放します。

  引数:

  - size: 解放するメモリーの量 (バイト単位)。
  - slabclass: メモリーが属しているクラス。
  - ptr: 解放するメモリーへのポインタ。

- assoc-find(key, depth)

  ハッシュテーブルで指定されたキーが検索されたときに起動します。これら 2 つの要素によって、ハッシュ関数がどの程度適切に機能しているかを把握できます。トラバーサルは、関数があまり最適ではなく、CPU の処理能力が浪費されていることを示します。

  引数:

  - key: 検索されたキー。
  - depth: ハッシュテーブルのリストの深さ。

- assoc-insert(key, nokeys)

  新しい項目が挿入されたときに起動します。

  引数:

  - key: 挿入されたキー。
  - nokeys: 現在格納されているキーの総数 (挿入が要求されたキーを含む)。

- assoc-delete(key, nokeys)

  新しい項目が削除されたときに起動します。

  引数:

  - key: 削除されたキー。
  - nokeys: 現在格納されているキーの総数 (削除が要求されたキーを除く)。

- item-link(key, size)

  キャッシュ内で項目がリンクされるときに起動します。

  引数:

  - key: 項目のキー。
  - size: データのサイズ。

- item-unlink(key, size)

  項目が削除されるときに起動します。

  引数:

  - key: 項目のキー。
  - size: データのサイズ。

- item-remove(key, size)

  項目の参照カウントが減ったときに起動します。

  引数:

  - key: 項目のキー。

  - size: データのサイズ。
- item-update(key, size)

  「最終参照」時間が更新されたときに起動します。

  引数:

  - key: 項目のキー。
  - size: データのサイズ。
- item-replace(oldkey, oldsize, newkey, newsize)

  項目が別の項目に置き換えられるときに起動します。

  引数:

  - oldkey: 置き換えられる項目のキー。
  - oldsize: 古い項目のサイズ。
  - newkey: 新しい項目のキー。
  - newsize: 新しい項目のサイズ。
- process-command-start(connid, request, size)

  コマンドの処理が開始したときに起動します。

  引数:

  - connid: 接続 ID。
  - request: 受信された要求。
  - size: 要求のサイズ。
- process-command-end(connid, response, size)

  コマンドの処理が終了したときに起動します。

  引数:

  - connid: 接続 ID。
  - response: クライアントに返信される応答。
  - size: 応答のサイズ。
- command-get(connid, key, size)

  get コマンドに対して起動します。

  引数:

  - connid: 接続 ID。
  - key: 要求されたキー。
  - size: このキーのデータのサイズ (見つからない場合は -1)。

- command-gets(connid, key, size, casid)

  gets コマンドに対して起動します。

  引数:

  - connid: 接続 ID。
  - key: 要求されたキー。
  - size: このキーのデータのサイズ (見つからない場合は -1)。
  - casid: 項目の CAS ID。
- command-add(connid, key, size)

  add コマンドに対して起動します。

  引数:

  - connid: 接続 ID。
  - key: 要求されたキー。
  - size: このキーのデータの新しいサイズ (見つからない場合は -1)。
- command-set(connid, key, size)

  set コマンドに対して起動します。

  引数:

  - connid: 接続 ID。
  - key: 要求されたキー。
  - size: このキーのデータの新しいサイズ (見つからない場合は -1)。
- command-replace(connid, key, size)

  replace コマンドに対して起動します。

  引数:

  - connid: 接続 ID。
  - key: 要求されたキー。
  - size: このキーのデータの新しいサイズ (見つからない場合は -1)。
- command-prepend(connid, key, size)

  prepend コマンドに対して起動します。

  引数:

  - connid: 接続 ID。
  - key: 要求されたキー。
  - size: このキーのデータの新しいサイズ (見つからない場合は -1)。
- command-append(connid, key, size)

  append コマンドに対して起動します。

  引数:

  - connid: 接続 ID。
  - key: 要求されたキー。

- size: このキーのデータの新しいサイズ (見つからない場合は -1)。

- command-cas(connid, key, size, casid)

  cas コマンドに対して起動します。

  引数:

  - connid: 接続 ID。
  - key: 要求されたキー。
  - size: このキーのデータのサイズ (見つからない場合は -1)。
  - casid: 要求された CAS ID。

- command-incr(connid, key, val)

  incr コマンドに対して起動します。

  引数:

  - connid: 接続 ID。
  - key: 要求されたキー。
  - val: 新しい値。

- command-decr(connid, key, val)

  decr コマンドに対して起動します。

  引数:

  - connid: 接続 ID。
  - key: 要求されたキー。
  - val: 新しい値。

- command-delete(connid, key, exptime)

  delete コマンドに対して起動します。

  引数:

  - connid: 接続 ID。
  - key: 要求されたキー。
  - exptime: 失効時間。

16.6.2.6 memcached 内でのメモリー割り当て

memcached を最初に起動したときは、構成したメモリーが自動的に割り当てられません。代わりに、memcached はキャッシュへの情報保存が開始されたあとで、はじめて物理メモリーの割り当てと確保を開始します。

キャッシュにデータを格納し始めたときに、memcached は項目上のデータのメモリーを項目単位では割り当てません。代わりに、スラブ割り当てを使用して、メモリーの使用量を最適化し、キャッシュの情報が失効したときのメモリーの断片化を防止します。

スラブ割り当てを使用すると、メモリーが 1M バイトのブロック単位で確保されます。スラブは、同じサイズのいくつかのブロックに分割されます。キャッシュに値を格納しようとすると、memcached はキャッシュに追加される値のサイズをチェックし、その項目に適したサイズの割り当てを含むスラブを特定します。項目のサイズに合うスラブがすでに存在する場合は、そのスラブ内のブロックに項目が書き込まれます。

新しい項目が既存のどのブロックのサイズよりも大きい場合は、新しいスラブが作成され、適切なサイズのブロックに分割されます。適切なブロックサイズを持つスラブがすでに存在するが、空きブロックがない場合は、

新しいスラブが作成されます。既存の項目を、そのキーに対する既存のブロック割り当てより大きいデータで更新すると、そのキーは適切なスラブに再割り当てされます。

たとえば、最小ブロックのデフォルトのサイズは 88 バイト (値の 40 バイトと、キーおよびフラグデータに対するデフォルトの 48 バイト) です。キャッシュに最初に格納する項目のサイズが 40 バイト未満である場合は、88 バイトのブロックサイズを持つスラブが作成され、値が格納されます。

格納予定のデータサイズがこの値より大きい場合は、ブロックサイズをチャンクサイズ係数の単位で増加させ、値を保持できる大きさのブロックサイズが特定されます。ブロックサイズは常にスケール係数の関数であり、チャンクサイズにちょうど分割できるブロックサイズに丸められます。

この構造の例については、図16.7「memcached でのメモリー割り当て」を参照してください。

図 16.7 memcached でのメモリー割り当て

この結果、memcached に割り当てられたメモリーの範囲内に複数のページが割り当てられます。各ページは、(デフォルトで) 1M バイトのサイズを持ち、キー/値ペアを格納するのに必要なチャンクサイズに従って異なる数のチャンクに分割されます。各インスタンスには複数のページが割り当てられ、特定のサイズのチャンクを必要とする新しい項目を作成する必要があるときは、常にページが作成されます。スラブは複数のページで構成される場合があり、スラブ内の各ページには同じ数のチャンクが含まれています。

新しいスラブのチャンクサイズは、ベースとなるチャンクサイズとチャンクサイズ増大係数の組み合わせによって決まります。たとえば、初期のチャンクサイズが 104 バイトで、デフォルトのチャンクサイズ増大係数 (1.25) が使用される場合、次に割り当てられるチャンクサイズは 104*1.25 にもっとも適合する 2 の累乗 (136 バイト) になります。

このようにしてページを割り当てることで、メモリーの断片化が回避されます。ただし、格納するオブジェクトの分布によっては、項目のサイズが大幅に異なる場合に、スラブとチャンクが効果的に分布しなくなる可能性があります。たとえば、各チャンクサイズ内の項目数が比較的少ない場合は、割り当てられた各ページにごく少数のチャンクしか存在しないため、多くのメモリーが無駄になる可能性があります。

-f コマンド行オプションを使用して増大係数を調整すると、この影響を軽減できます。このオプションは、割り当てられたチャンクとスラブをより効果的に利用できるように、適用される増大係数を調整します。現在のスラブ割り当ての統計を確認する方法については、セクション16.6.4.2「memcached のスラブ統計」を参照してください。

オペレーティングシステムでサポートされる場合は、-L コマンド行オプションを指定して memcached を起動することもできます。このオプションは、起動中に大規模メモリーページを使用してすべてのメモリーを事前に割り当てます。これにより、CPU メモリーキャッシュ内のミスの数が減ってパフォーマンスが向上する可能性があります。

16.6.2.7 memcached のスレッドサポート

memcached をソースからビルドするときに内部のスレッド実装を有効にすると、memcached は libevent システムに加えて複数のスレッドを使用して要求を処理します。

有効にすると、スレッド実装は次のように動作します。

- スレッド化は、コード内の関数をラップして、同じグローバル構造の同時更新から保護する基本的な機能を提供することによって処理されます。
- 各スレッドは、独自の libevent インスタンスを使用してパフォーマンスを向上させます。
- TCP/IP 接続は、TCP/IP ソケットで待機する単独のスレッドによって処理されます。その後、各接続は単純なラウンドロビン方式でいずれかのアクティブスレッドに分配されます。その後、接続が開いている間は、各接続はこのスレッド内でのみ動作します。

- UDP 接続については、すべてのスレッドが 1 つの UDP ソケットで受信要求を待機します。現在別の要求を処理していないスレッドは、受信パケットを無視します。残りの (ビジーではない) スレッドのいずれかが要求を読み取り、応答を送信します。この実装では、要求を処理する可能性があるスレッドがスリープから復帰するため、CPU 負荷が増加する可能性があります。

スレッドを使用すると、ハッシュテーブルを更新する要求を個々のスレッド間で分散できるため、複数の CPU コアが使用可能なサーバーではパフォーマンスが向上する可能性があります。使用されるロックメカニズムのオーバーヘッドを最小限に抑えるには、さまざまなスレッド値を試して、特定のワークロード内の要求数とタイプに基づく最適なパフォーマンスを実現してください。

16.6.2.8 memcached ログ

-v、-vv、または -vvv オプションを使用して冗長モードを有効にすると、memcached が出力する情報に実行中の操作の詳細が追加されます。

冗長オプションを指定しないと、通常は memcached の正常な動作中に出力が生成されません。

- -v 使用時の出力

  もっとも低い冗長レベルでは、次が表示されます。

  - エラーおよび警告
  - 一時的なエラー
  - プロトコルおよびソケットのエラー (使用可能な接続の使い果たしなど)
  - 登録された個々のクライアント接続 (使用されているソケットディスクリプタ番号とプロトコルを含む)。

    例:

    ```
    32: Client using the ascii protocol
    33: Client using the ascii protocol
    ```

    ソケットディスクリプタはクライアントが接続されている間だけ有効になります。永続的ではない接続は表示されない場合があります。

  このレベルで出力されるエラーメッセージの例を次に示します。

  ```
  <%d send buffer was %d, now %d
  Can't listen for events on fd %d
  Can't read from libevent pipe
  Catastrophic: event fd doesn't match conn fd!
  Couldn't build response
  Couldn't realloc input buffer
  Couldn't update event
  Failed to build UDP headers
  Failed to read, and not due to blocking
  Too many open connections
  Unexpected state %d
  ```

- -vv 使用時の出力

  2 番目の冗長レベルを使用すると、プロトコルの動作、更新されたキー、チャンクとネットワークの動作および詳細に関するより詳しい情報が提供されます。

  この冗長レベルで memcached を起動すると、最初に個々のスラブクラスのサイズ、チャンクサイズ、およびスラブあたりのエントリ数が表示されます。これらは、スラブが割り当てられたことを示すのではなく、データが追加されたときに作成されるスラブを示します。情報を送信するために使用される待機キューおよびバッファーに関する情報も表示されます。デフォルトのメモリーおよび増大係数を含む TCP/IP ベースシステムに関して生成された出力の例を次に示します。

  ```
  shell> memcached -vv
  slab class   1: chunk size     80 perslab 13107
  slab class   2: chunk size    104 perslab 10082
  slab class   3: chunk size    136 perslab  7710
  slab class   4: chunk size    176 perslab  5957
  slab class   5: chunk size    224 perslab  4681
  slab class   6: chunk size    280 perslab  3744
  slab class   7: chunk size    352 perslab  2978
  slab class   8: chunk size    440 perslab  2383
  slab class   9: chunk size    552 perslab  1899
  slab class  10: chunk size    696 perslab  1506
  ```

```
slab class  11: chunk size    872 perslab  1202
slab class  12: chunk size   1096 perslab   956
slab class  13: chunk size   1376 perslab   762
slab class  14: chunk size   1720 perslab   609
slab class  15: chunk size   2152 perslab   487
slab class  16: chunk size   2696 perslab   388
slab class  17: chunk size   3376 perslab   310
slab class  18: chunk size   4224 perslab   248
slab class  19: chunk size   5280 perslab   198
slab class  20: chunk size   6600 perslab   158
slab class  21: chunk size   8256 perslab   127
slab class  22: chunk size  10320 perslab   101
slab class  23: chunk size  12904 perslab    81
slab class  24: chunk size  16136 perslab    64
slab class  25: chunk size  20176 perslab    51
slab class  26: chunk size  25224 perslab    41
slab class  27: chunk size  31536 perslab    33
slab class  28: chunk size  39424 perslab    26
slab class  29: chunk size  49280 perslab    21
slab class  30: chunk size  61600 perslab    17
slab class  31: chunk size  77000 perslab    13
slab class  32: chunk size  96256 perslab    10
slab class  33: chunk size 120320 perslab     8
slab class  34: chunk size 150400 perslab     6
slab class  35: chunk size 188000 perslab     5
slab class  36: chunk size 235000 perslab     4
slab class  37: chunk size 293752 perslab     3
slab class  38: chunk size 367192 perslab     2
slab class  39: chunk size 458992 perslab     2
<26 server listening (auto-negotiate)
<29 server listening (auto-negotiate)
<30 send buffer was 57344, now 2097152
<31 send buffer was 57344, now 2097152
<30 server listening (udp)
<30 server listening (udp)
<31 server listening (udp)
<30 server listening (udp)
<30 server listening (udp)
<31 server listening (udp)
<31 server listening (udp)
<31 server listening (udp)
```

この冗長レベルを使用すると、異なるメモリー割り当てを含むスラブに対して使用される増大係数の効果を効果的にチェックできます。また、その結果を使用して、キャッシュに格納するデータに合わせて増大係数を調整できます。たとえば、増大係数を 4 に設定した (各スラブのサイズが 4 倍になった) 場合:

```
shell> memcached -f 4 -m 1g -vv
slab class   1: chunk size     80 perslab 13107
slab class   2: chunk size    320 perslab  3276
slab class   3: chunk size   1280 perslab   819
slab class   4: chunk size   5120 perslab   204
slab class   5: chunk size  20480 perslab    51
slab class   6: chunk size  81920 perslab    12
slab class   7: chunk size 327680 perslab     3
...
```

この冗長レベルでは、キャッシュの使用中に、キーおよびその他の情報の格納とリカバリに関する詳細情報も出力されます。一般的な設定/取得および増分/減分操作中の出力の例を次に示します。

```
32: Client using the ascii protocol
<32 set my_key 0 0 10
>32 STORED
<32 set object_key 1 0 36
>32 STORED
<32 get my_key
>32 sending key my_key
>32 END
<32 get object_key
>32 sending key object_key
>32 END
<32 set key 0 0 6
>32 STORED
<32 incr key 1
>32 789544
<32 decr key 1
>32 789543
<32 incr key 2
>32 789545
<32 set my_key 0 0 10
```

```
>32 STORED
<32 set object_key 1 0 36
>32 STORED
<32 get my_key
>32 sending key my_key
>32 END
<32 get object_key
>32 sending key object_key1 1 36

>32 END
<32 set key 0 0 6
>32 STORED
<32 incr key 1
>32 789544
<32 decr key 1
>32 789543
<32 incr key 2
>32 789545
```

クライアント通信中は、各行の最初の文字は情報の流れの方向を示しています。< はクライアントから memcached サーバーへの通信、> はクライアントへの通信をそれぞれ示します。数字は、この接続のソケットディスクリプタを表す数値です。

- -vvv 使用時の出力

  この冗長レベルには、クライアントとの間でコンテンツの読み取りおよび書き込みを行なっている間の、イベントライブラリのさまざまな接続状態の遷移が含まれます。これは、クライアント通信の問題を診断して識別するために使用します。たとえば、この情報を使用して、クライアント操作の読み取り中または操作が返されて完了する前に memcached がクライアントに情報を返すのに長い時間がかかっているかどうかを判定できます。設定操作の一般的なシーケンスの例を次に示します。

```
<32 new auto-negotiating client connection
32: going from conn_new_cmd to conn_waiting
32: going from conn_waiting to conn_read
32: going from conn_read to conn_parse_cmd
32: Client using the ascii protocol
<32 set my_key 0 0 10
32: going from conn_parse_cmd to conn_nread
> NOT FOUND my_key
>32 STORED
32: going from conn_nread to conn_write
32: going from conn_write to conn_new_cmd
32: going from conn_new_cmd to conn_waiting
32: going from conn_waiting to conn_read
32: going from conn_read to conn_closing
<32 connection closed.
```

memcached のすべての冗長レベルは、問題のデバッグ中または調査中に使用するために設計されています。生成される情報 (特に -vvv を使用したとき) の量は、(特に負荷の高いサーバーでは) 重要です。エラー情報を (特にディスクに) 書き出すと、memcached を使用することで達成したパフォーマンス向上の一部が打ち消される可能性があります。そのため、本番環境や配備環境での使用は推奨されません。

16.6.3 memcached アプリケーションの開発

いくつかの言語インタフェースを使用すると、アプリケーションから memcached サーバーを使用して情報を格納および取得できます。memcached アプリケーションは、Perl、PHP、Python、Ruby、C、Java などの広く使用されている言語で記述できます。

memcached サーバーに格納されたデータは 1 つの文字列 (キー) で参照され、キャッシュへの格納時とキャッシュからの取得時にはそのキーが参照として使用されます。したがって、このキャッシュは大規模な連想配列またはハッシュテーブルのように動作します。キャッシュに格納された情報を構造化またはその他の方法で整理できません。複数テーブルや複合キー値などのデータベースの概念をエミュレートするには、キーとして使用する文字列に追加の情報をエンコードする必要があります。たとえば、特定の緯度と経度に対応する住所を格納または検索するには、これら 2 つの数値を 1 つのカンマ区切り文字列に変換し、それをキーとして使用します。

16.6.3.1 memcached の基本操作

memcached のインタフェースは、キャッシュ内の情報を格納および取得するために次のメソッドをサポートします。これらは、言語固有のしくみが異なる可能性がありますが、異なるすべての API 間で一貫しています。

- get(key): キャッシュから情報を取得します。指定されたキーが存在する場合は、そのキーに関連付けられた値を返します。指定されたキーが存在しない場合は、NULL、nil、undefined、または対応する言語のもっとも近い等価要素を返します。

- set(key, value [, expiry]): キャッシュ内のキーに関連付けられた項目を指定された値に設定します。これは、キーがすでに存在する場合は既存の項目を更新し、キーが存在しない場合は新しいキー/値ペアを追加します。失効時間が指定されている場合は、失効時間に達したときにその項目が失効します (そして削除されます)。この時間は秒単位で指定され、値が 30 日未満 (30*24*60*60) の場合は相対時間とみなされ、この値より大きい場合は絶対時間 (エポック) とみなされます。
- add(key, value [, expiry]): 指定されたキーがまだ存在しない場合に、そのキーとそれに関連する値をキャッシュに追加します。
- replace(key, value [, expiry]): 指定された key が存在する場合にのみ、そのキーに関連付けられた項目を置き換えます。新しい値は value パラメータで指定されます。
- delete(key [, time]): key とそれに関連する項目をキャッシュから削除します。time を指定すると、指定した key による別の項目の追加が指定した期間だけブロックされます。
- incr(key , value): key に関連付けられた項目を指定された value だけ増分します。
- decr(key , value): key に関連付けられた項目を指定された value だけ減分します。
- flush_all: キャッシュ内の現在の値をすべて無効化 (または失効化) します。これらは技術的にはまだ存在します (削除されません) が、次にアクセスしようとしたときに警告が表示されず破壊されます。

どの実装でも、これらの関数のほとんどまたはすべてが対応するネイティブ言語インタフェースによって複製されています。

可能な場合は、データベースの 1 つのカラム値をキャッシュするのではなく、memcached を使用してすべての項目を格納してください。たとえば、オブジェクト (請求書、ユーザー履歴、またはブログ投稿) に関するレコードを表示するときは、データベースから関連するエントリデータをロードし、それをアプリケーションが通常必要とする内部構造にまとめます。この完成されたオブジェクトをキャッシュに保存します。

複雑なデータ構造は直接格納できません。ほとんどのインタフェースは、データを自動的に直列化 (つまり、元のポインタとネストを再構築できるテキスト形式に変換) します。Perl では Storable、PHP では serialize、Python では cPickle (または Pickle)、Java では Serializable インタフェースがそれぞれ使用されます。ほとんどの場合、使用される直列化インタフェースはカスタマイズ可能です。memcached インスタンスに格納されたデータを異なる言語インタフェース間で共有するには、JSON (Javascript Object Notation) などの共通の直列化ソリューションの使用を検討してください。

16.6.3.2 MySQL キャッシュレイヤーとしての memcached の使用

memcached を使用して MySQL のデータをキャッシュする場合、アプリケーションはデータベースからデータを取得し、適切なキー/値ペアをキャッシュにロードする必要があります。その後の検索は、キャッシュから直接実行できます。

MySQL にはクエリーされたデータ用に独自のインメモリーキャッシュメカニズム (InnoDB のバッファープールや MySQL クエリーキャッシュなど) が用意されているので、個々のカラム値や行のキャッシュへのロード以外の機能も使用できます。結合クエリーによって複数のテーブルから取得された値や複数の行から集められた結果セットなどの複合値のキャッシュにより適しています。

> **注意**
>
> memcached インスタンス内の情報へのアクセスや情報の更新に必要なセキュリティーはないため、キャッシュ内の情報は機密扱い以外のデータに限定してください。マシンにアクセスできるすべてのユーザーが、情報を読み取り、表示し、場合によっては更新できます。データをセキュアな状態に保つには、情報をキャッシュする前に暗号化します。サーバーに接続できるユーザーを制限するには、ネットワークアクセスを無効にするか、IPTables または同様の手法を使用して memcached ポートへのアクセスを選択したホストのセットに制限します。

memcached は、キャッシュが元の設計に組み込まれていなかった場合でも、既存のアプリケーションに導入できます。多くの言語および環境では、アプリケーションの変更はほんの数行です。最初にデータをロードするときにキャッシュからの読み取りを試行し、情報がキャッシュされていなかった場合は古い方法にフォールバックし、データを読み取ったらその情報でキャッシュを更新します。

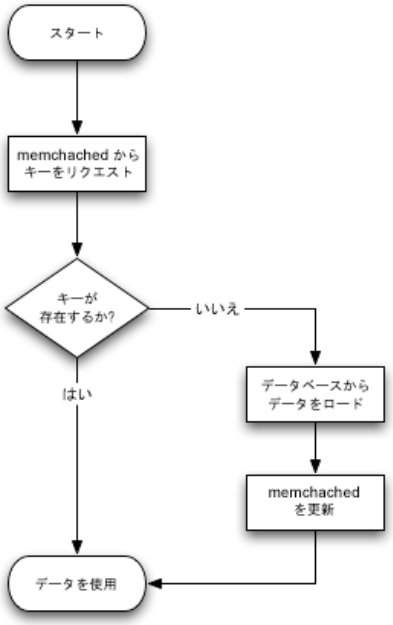
どの言語でも、MySQL のキャッシュソリューションとして memcached を使用する一般的な手順は次のとおりです。

1. キャッシュから項目を要求します。

2. 項目が存在する場合は、その項目データを使用します。
3. 項目が存在しない場合は、MySQL からデータをロードし、その値をキャッシュに格納します。つまり、その値はキャッシュからそれを要求する次のクライアントで使用可能になります。

この手順のフローチャートについては、図16.8「一般的な memcached アプリケーションのフローチャート」を参照してください。

図 16.8 一般的な memcached アプリケーションのフローチャート



memcached アプリケーションへのデータベースのベストプラクティスの適用

MySQL のデータをキャッシュするもっとも直接的な方法は、2 カラムのテーブルを使用して、1 つ目のカラムを主キーにすることです。memcached キーの一意性要件のため、データベーススキーマで主キーと一意制約が適切に使用されていることを確認してください。

複数のカラム値を結合して 1 つの memcached 項目値にする場合は、値をそのコンポーネントに簡単に解析できるようなデータ型を選択してください (たとえば、数値間に区切り文字を使用するなど)。

memcached の検索にもっとも簡単にマップできるクエリーは、WHERE 句が 1 つで、= または IN 演算子を使用するクエリーです。複雑な WHERE 句や、<、>、BETWEEN、LIKE などの演算子を使用するものの場合、memcached ではキーや関連する値を簡単または効率的にスキャンまたはフィルタする方法がないため、通常はこれらの操作をベースとなるデータベースに対する SQL クエリーとして実行します。

16.6.3.3 C および C++ での libmemcached の使用

libmemcached ライブラリは、memcached に対する C および C++ インタフェースを提供し、いくつかの異なる追加 API 実装 (Perl、Python、および Ruby を含む) の基盤にもなります。libmemcached のコア関数を理解することは、こうしたほかのインタフェースを使用するときにも役立ちます。

C ライブラリは、memcached のもっとも包括的なインタフェースライブラリであり、libmemcached ライブラリをベースとしないインタフェースでは常時公開されていない関数と操作システムを提供します。

さまざまな関数を、その基本操作に応じて分類できます。コア API へのインタフェースとなる関数に加えて、いくつかのユーティリティー関数によって拡張機能 (データの末尾への追加や先頭への追加など) が提供されます。

libmemcached をビルドしてインストールするには、libmemcached パッケージをダウンロードし、configure を実行してから、ビルドおよびインストールします。

```
shell> tar xjf libmemcached-0.21.tar.gz
shell> cd libmemcached-0.21
shell> ./configure
shell> make
```

```
shell> make install
```

多くの Linux オペレーティングシステムでは、通常の yum、apt-get、または同様のコマンドを使用すると、対応する libmemcached パッケージをインストールできます。

このライブラリを使用するアプリケーションをビルドするには、最初にサーバーリストを設定します。memcached_st メイン構造体の内部に構成されたサーバーを直接操作したり、サーバーリストを別個に移入したりして、そのリストを memcached_st 構造体に追加します。次の例では、後者の方法を使用しています。サーバーリストを設定したあとは、関数を呼び出してデータを格納または取得できます。事前設定値を localhost に設定する簡単なアプリケーションをここに示します。

```
#include <stdio.h>
#include <string.h>
#include <unistd.h>
#include <libmemcached/memcached.h>

int main(int argc, char *argv[])
{
  memcached_server_st *servers = NULL;
  memcached_st *memc;
  memcached_return rc;
  char *key= "keystring";
  char *value= "keyvalue";

  memcached_server_st *memcached_servers_parse (char *server_strings);
  memc= memcached_create(NULL);

  servers= memcached_server_list_append(servers, "localhost", 11211, &rc);
  rc= memcached_server_push(memc, servers);

  if (rc == MEMCACHED_SUCCESS)
    fprintf(stderr,"Added server successfully\n");
  else
    fprintf(stderr,"Couldn't add server: %s\n",memcached_strerror(memc, rc));

  rc= memcached_set(memc, key, strlen(key), value, strlen(value), (time_t)0, (uint32_t)0);

  if (rc == MEMCACHED_SUCCESS)
    fprintf(stderr,"Key stored successfully\n");
  else
    fprintf(stderr,"Couldn't store key: %s\n",memcached_strerror(memc, rc));

  return 0;
}
```

操作が成功したかどうかをテストするには、特定の関数の戻り値 (または移入された結果コード) を使用します。操作が成功した場合、この値は常に MEMCACHED_SUCCESS に設定されます。エラーが発生した場合は、memcached_strerror() 関数を使用して、結果コードを出力可能な文字列に変換します。

アプリケーションをビルドするには、memcached ライブラリを指定します。

```
shell> gcc -o memc_basic memc_basic.c -lmemcached
```

memcached サーバーを起動したあとで上記のサンプルアプリケーションを実行すると、成功メッセージが返されます。

```
shell> memc_basic
Added server successfully
Key stored successfully
```

libmemcached のベース関数

libmemcached のベース関数を使用すると、memcached サーバーとのインタフェースとして使用される memcached_st メイン構造体を作成、破壊、およびクローニングできます。主な関数の定義を次に示します。

```
memcached_st *memcached_create (memcached_st *ptr);
```

ほかの libmemcached API 関数で使用する新しい memcached_st 構造体を作成します。既存の静的な memcached_st 構造体を指定することも、NULL を指定して新しい構造体を割り当てることもできます。作成された構造体へのポインタを返します。失敗時は NULL を返します。

```
void memcached_free (memcached_st *ptr);
```

以前に作成された memcached_st 構造体に割り当てられていた構造体とメモリーを解放します。

```
memcached_st *memcached_clone(memcached_st *clone, memcached_st *source);
```

指定された source から既存の memcached 構造体をクローニングし、その構造体に定義されているデフォルトとサーバーリストをコピーします。

libmemcached のサーバー関数

libmemcached API は、memcached_server_st 構造体に格納されているサーバーリストを使用して、残りの関数で使用されるサーバーリストとして機能します。memcached を使用するには、最初にサーバーリストを作成し、次にそのサーバーリストを有効な libmemcached オブジェクトに適用します。

このサーバーリストとアクティブな libmemcached オブジェクト内のサーバーリストは別個に操作できるため、アクティブな libmemcached インタフェースの実行中にサーバーリストを更新および管理できます。

memcached_st 構造体内のサーバーリストを操作する関数は、次のとおりです。

```
memcached_return
  memcached_server_add (memcached_st *ptr,
                        char *hostname,
                        unsigned int port);
```

指定された hostname および port を使用して、ptr で指定された memcached_st 構造体にサーバーを追加します。

```
memcached_return
  memcached_server_add_unix_socket (memcached_st *ptr,
                                    char *socket);
```

memcached_st 構造体に構成されたサーバーリストに Unix ソケットを追加します。

```
unsigned int memcached_server_count (memcached_st *ptr);
```

memcached_st 構造体に含まれている構成済みサーバー数のカウントを返します。

```
memcached_server_st *
  memcached_server_list (memcached_st *ptr);
```

memcached_st 構造体に含まれる定義済みホストの配列を返します。

```
memcached_return
  memcached_server_push (memcached_st *ptr,
                         memcached_server_st *list);
```

現在の memcached_st 構造体に構成されているサーバーリストに対して既存のサーバーリストをプッシュします。これにより、既存のリストの末尾にサーバーが追加されますが、重複はチェックされません。

memcached_server_st 構造体を使用して memcached サーバーのリストを作成し、それらを memcached_st 構造体に個別に適用できます。

```
memcached_server_st *
  memcached_server_list_append (memcached_server_st *ptr,
                                char *hostname,
                                unsigned int port,
                                memcached_return *error);
```

hostname および port を使用して、ptr のサーバーリストにサーバーを追加します。結果コードは error 引数によって処理されます。この引数は既存の memcached_return 変数を指します。この関数は返されたリストへのポインタを返します。

```
unsigned int memcached_server_list_count (memcached_server_st *ptr);
```

サーバーリスト内のサーバー数を返します。

```
void memcached_server_list_free (memcached_server_st *ptr);
```

サーバーリストに関連付けられたメモリーを解放します。

```
memcached_server_st *memcached_servers_parse (char *server_strings);
```

サーバーリストを含む文字列を解析します。個々のサーバーはカンマ、空白、またはその両方で区切られ、個々のサーバーは server[:port] という形式になっています。戻り値はサーバーリスト構造体です。

libmemcached の設定関数

libmemcached 内の設定関連関数は、memcached プロトコルでサポートされるコア関数と同じ機能を備えています。各種関数の完全な定義は、すべてのベース関数 (add、replace、prepend、append) と同じです。たとえば、memcached_set() の関数定義は次のとおりです。

```
memcached_return
  memcached_set (memcached_st *ptr,
                 const char *key,
                 size_t key_length,
                 const char *value,
                 size_t value_length,
                 time_t expiration,
                 uint32_t flags);
```

ptr は memcached_st 構造体です。key と key_length はキーの名前と長さを定義し、value と value_length は対応する値と長さを定義します。失効およびオプションフラグを設定することもできます。詳細については、libmemcached の動作の制御を参照してください。

この表に、libmemcached の残りの設定関連関数と、memcached プロトコルでサポートされる同等のコア関数の概要を示します。

libmemcached の関数	同等のコア関数
memcached_set(memc, key, key_length, value, value_length, expiration, flags)	汎用の set() 操作。
memcached_add(memc, key, key_length, value, value_length, expiration, flags)	汎用の add() 関数。
memcached_replace(memc, key, key_length, value, value_length, expiration, flags)	汎用の replace()。
memcached_prepend(memc, key, key_length, value, value_length, expiration, flags)	指定された value を指定された key の現在の値の前に追加します。
memcached_append(memc, key, key_length, value, value_length, expiration, flags)	指定された value を指定された key の現在の値のあとに追加します。
memcached_cas(memc, key, key_length, value, value_length, expiration, flags, cas)	サーバー内の対応する cas 値が同じ場合は、特定のキーのデータを上書きします。
memcached_set_by_key(memc, master_key, master_key_length, key, key_length, value, value_length, expiration, flags)	汎用の set() とほぼ同じですが、個々のサーバーの識別に使用できる追加のマスターキーオプションを備えています。
memcached_add_by_key(memc, master_key, master_key_length, key, key_length, value, value_length, expiration, flags)	汎用の add() とほぼ同じですが、個々のサーバーの識別に使用できる追加のマスターキーオプションを備えています。
memcached_replace_by_key(memc, master_key, master_key_length, key, key_length, value, value_length, expiration, flags)	汎用の replace() とほぼ同じですが、個々のサーバーの識別に使用できる追加のマスターキーオプションを備えています。
memcached_prepend_by_key(memc, master_key, master_key_length, key, key_length, value, value_length, expiration, flags)	memcached_prepend() とほぼ同じですが、個々のサーバーの識別に使用できる追加のマスターキーオプションを備えています。
memcached_append_by_key(memc, master_key, master_key_length, key, key_length, value, value_length, expiration, flags)	memcached_append() とほぼ同じですが、個々のサーバーの識別に使用できる追加のマスターキーオプションを備えています。
memcached_cas_by_key(memc, master_key, master_key_length, key, key_length, value, value_length, expiration, flags)	memcached_cas() とほぼ同じですが、個々のサーバーの識別に使用できる追加のマスターキーオプションを備えています。

by_key メソッドには、サーバー選択のハッシュ化段階で使用および適用されるマスターキーを定義する 2 つの引数が追加されています。これは、次の定義で確認できます。

```
memcached_return
  memcached_set_by_key(memcached_st *ptr,
                       const char *master_key,
                       size_t master_key_length,
                       const char *key,
                       size_t key_length,
```

```
                    const char *value,
                    size_t value_length,
                    time_t expiration,
                    uint32_t flags);
```

すべての関数が MEMCACHED_SUCCESS 定数と比較できる memcached_return 型の値を返します。

libmemcached の取得関数

libmemcached の関数は、1 つの項目への直接アクセスと、多数のキーを同時にフェッチするときの応答が非常に高速な複数キーの要求メカニズムを提供します。

汎用の get() と同等である主要な get スタイルの関数は、memcached_get() です。この関数は、指定されたキーに関連付けられた値を指す文字列ポインタを返します。

```
char *memcached_get (memcached_st *ptr,
                     const char *key, size_t key_length,
                     size_t *value_length,
                     uint32_t *flags,
                     memcached_return *error);
```

複数キーの get である memcached_mget() も使用可能です。複数キーの取得操作を使用すると、memcached_get() を個々に呼び出してキー値を取得するよりも、1 ブロックの操作を非常に高速に実行できます。複数キー取得を開始するには、memcached_mget() を呼び出します。

```
memcached_return
   memcached_mget (memcached_st *ptr,
                   char **keys, size_t *key_length,
                   unsigned int number_of_keys);
```

戻り値は操作の成功です。keys パラメータはキーを含む文字列の配列、key_length は対応する各キーの長さを含む配列です。number_of_keys は配列に設定したキーの数です。

個々の値をフェッチするには、memcached_fetch() を使用して対応する各値を取得します。

```
char *memcached_fetch (memcached_st *ptr,
                       const char *key, size_t *key_length,
                       size_t *value_length,
                       uint32_t *flags,
                       memcached_return *error);
```

この関数はキー値を返し、key、key_length、および value_length パラメータに対応するキーと長さの情報を移入します。この関数は、返す値がなくなったときに NULL を返します。キーデータの移入と情報の戻りを含む完全な例をここに示します。

```
#include <stdio.h>
#include <sstring.h>
#include <unistd.h>
#include <libmemcached/memcached.h>

int main(int argc, char *argv[])
{
  memcached_server_st *servers = NULL;
  memcached_st *memc;
  memcached_return rc;
  char *keys[]= {"huey", "dewey", "louie"};
  size_t key_length[3];
  char *values[]= {"red", "blue", "green"};
  size_t value_length[3];
  unsigned int x;
  uint32_t flags;

  char return_key[MEMCACHED_MAX_KEY];
  size_t return_key_length;
  char *return_value;
  size_t return_value_length;

  memc= memcached_create(NULL);

  servers= memcached_server_list_append(servers, "localhost", 11211, &rc);
  rc= memcached_server_push(memc, servers);

  if (rc == MEMCACHED_SUCCESS)
    fprintf(stderr,"Added server successfully\n");
  else
    fprintf(stderr,"Couldn't add server: %s\n",memcached_strerror(memc, rc));
```

```
 for(x= 0; x < 3; x++)
  {
    key_length[x] = strlen(keys[x]);
    value_length[x] = strlen(values[x]);

    rc= memcached_set(memc, keys[x], key_length[x], values[x],
                      value_length[x], (time_t)0, (uint32_t)0);
    if (rc == MEMCACHED_SUCCESS)
      fprintf(stderr,"Key %s stored successfully\n",keys[x]);
    else
      fprintf(stderr,"Couldn't store key: %s\n",memcached_strerror(memc, rc));
  }

 rc= memcached_mget(memc, keys, key_length, 3);

 if (rc == MEMCACHED_SUCCESS)
  {
    while ((return_value= memcached_fetch(memc, return_key, &return_key_length,
                                          &return_value_length, &flags, &rc)) != NULL)
      {
        if (rc == MEMCACHED_SUCCESS)
          {
            fprintf(stderr,"Key %s returned %s\n",return_key, return_value);
          }
      }
  }

 return 0;
}
```

上記のアプリケーションを実行すると、次の出力が生成されます。

```
shell> memc_multi_fetch
Added server successfully
Key huey stored successfully
Key dewey stored successfully
Key louie stored successfully
Key huey returned red
Key dewey returned blue
Key louie returned green
```

libmemcached の動作の制御

libmemcached の動作は、1 つ以上の動作フラグを設定することで変更できます。これらは、グローバルに設定するか、個々の関数の呼び出し中に適用できます。サーバーの選択時に使用されるハッシュ化メカニズムなど、追加の設定を受け入れる動作もあります。

グローバルな動作を設定するには:

```
memcached_return
  memcached_behavior_set (memcached_st *ptr,
                          memcached_behavior flag,
                          uint64_t data);
```

現在の動作設定を取得するには:

```
uint64_t
  memcached_behavior_get (memcached_st *ptr,
                          memcached_behavior flag);
```

libmemcached の動作フラグを次の表に示します。

動作	説明
MEMCACHED_BEHAVIOR_NO_BLOCK	libmemcached で非同期 I/O を使用します。
MEMCACHED_BEHAVIOR_TCP_NODELAY	ネットワークソケットの NODELAY をオンにします。
MEMCACHED_BEHAVIOR_HASH	値がない場合は、MD5 を使用してキー用のデフォルトのハッシュ化アルゴリズムを設定します。ほかの有効な値として、MEMCACHED_HASH_DEFAULT、MEMCACHED_HASH_MD5、MEMCACH および MEMCACHED_HASH_FNV1A_32 があります。
MEMCACHED_BEHAVIOR_DISTRIBUTION	特定の値を格納するために使用されるサーバーの選択方法を変更します。デフォルトの方法は MEMCACHED_DISTRIBUTION_MODULA です。整合ハッシュ化を有効にするに

動作	説明
	は、MEMCACHED_DISTRIBUTION_CONSISTENT を設定します。MEMCACHED_DISTRIBUTION_CONSISTENT は、値 MEMCACHED_DISTRIBUTION_CONSISTENT_KETAMA のエイリアスです。
MEMCACHED_BEHAVIOR_CACHE_LOOKUPS	DNS サービスに対して行われたルックアップをキャッシュします。個々のホストに IP アドレスではなく名前を使用している場合は、これによってパフォーマンスが向上することがあります。
MEMCACHED_BEHAVIOR_SUPPORT_CAS	CAS 操作をサポートします。パフォーマンスが低下するため、デフォルトではこれは無効になっています。
MEMCACHED_BEHAVIOR_KETAMA	デフォルトの分布を MEMCACHED_DISTRIBUTION_CONSISTENT_KETAMA に設定し、ハッシュを MEMCACHED_HASH_MD5 に設定します。
MEMCACHED_BEHAVIOR_POLL_TIMEOUT	poll() で使用されるタイムアウト値を変更します。タイムアウト値には signed int ポインタを指定します。
MEMCACHED_BEHAVIOR_BUFFER_REQUEST	I/O 要求を送信せずにバッファリングします。このデータは、取得操作または接続のクローズによってフラッシュされます。
MEMCACHED_BEHAVIOR_VERIFY_KEY	指定されたキーが有効かどうかを libmemcached が強制的に検証します。
MEMCACHED_BEHAVIOR_SORT_HOSTS	設定すると、memcached_st 構造体の構成済みホストのリストに追加されたホストが、ソートされた順序でホストリストに配置されます。これにより、整合ハッシュ化が有効になっている場合は、その動作が破棄されます。
MEMCACHED_BEHAVIOR_CONNECT_TIMEOUT	非ブロックモードでは、これによってソケット接続中のタイムアウトの値が変更されます。

libmemcached のコマンド行ユーティリティー

libmemcached には、主要な C ライブラリインタフェースに加えて、memcached アプリケーションの操作やデバッグに役立ついくつかのコマンド行ユーティリティーも含まれています。

すべてのコマンド行ツールはいくつかの引数を受け入れますが、その中でもっとも重要なものは、情報を返すときに接続するサーバーのリストを指定する servers です。

主要なツールは次のとおりです。

- memcat: 指定された各 ID の値をコマンド行に表示します。

```
shell> memcat --servers=localhost hwkey
Hello world
```

- memcp: ファイル名をキーとして使用して、ファイルの内容をキャッシュにコピーします。

```
shell> echo "Hello World" > hwkey
shell> memcp --servers=localhost hwkey
shell> memcat --servers=localhost hwkey
Hello world
```

- memrm: キャッシュから項目を削除します。

```
shell> memcat --servers=localhost hwkey
Hello world
shell> memrm --servers=localhost hwkey
shell> memcat --servers=localhost hwkey
```

- memslap: 取得/設定および複数クライアントの操作をシミュレートして、1 つ以上の memcached サーバーに対する負荷をテストします。たとえば、取得操作を実行する 100 台のクライアントの負荷をシミュレートできます。

```
shell> memslap --servers=localhost --concurrency=100 --flush --test=get
memslap --servers=localhost --concurrency=100 --flush --test=get Threads connecting to servers 100
 Took 13.571 seconds to read data
```

- memflush: memcached キャッシュの内容をフラッシュ (空に) します。

```
shell> memflush --servers=localhost
```

16.6.3.4 Perl での MySQL と memcached の使用

Cache::Memcached モジュールは、Memcache プロトコルへのネイティブインタフェースを提供し、memcached に用意されているコア関数をサポートします。このモジュールは、オペレーティングシステムのパッケージ管理システムまたは CPAN を使用してインストールします。

```
root-shell> perl -MCPAN -e 'install Cache::Memcached'
```

Perl から Cache::Memcached モジュールを介して memcached を使用するには、最初に接続用のサーバーリストとその他のパラメータを定義する新しい Cache::Memcached オブジェクトを作成します。唯一の引数は、キャッシュインタフェース用のオプションを含むハッシュです。たとえば、3 台の memcached サーバーを使用する新しいインスタンスを作成するには:

```
use Cache::Memcached;

my $cache = new Cache::Memcached {
    'servers' => [
        '192.168.0.100:11211',
        '192.168.0.101:11211',
        '192.168.0.102:11211',
 ],
};
```

> **注記**
>
> 複数のサーバーとともに Cache::Memcached インタフェースを使用すると、この API はグループ内のすべてのサーバーに対して特定の操作を自動的に実行します。たとえば、Cache::Memcached を介して統計情報を取得すると、ホスト単位のデータを含むハッシュとともに、グループ内のすべてのサーバーに関して一般化された統計が返されます。

キャッシュオブジェクトインスタンスの作成時にインスタンスに対して追加のプロパティーを設定するには、オプションのハッシュの一部としてそのオプションを指定します。または、インスタンスに対して対応するメソッドを使用することもできます。

- servers または set_servers() メソッド: 使用されるサーバーのリストを指定します。このサーバーリストは、各要素がアドレスとポート番号の (コロンで区切られた) 組み合わせであるサーバー配列への参照です。Unix ソケットによるローカル接続を指定することもできます (たとえば、/tmp/sock/memcached)。(ハッシュ化時にどの程度の頻度でそのサーバーを使用すべきかを示す) 重み付きでサーバーを指定するには、memcached サーバーインスタンスと重み値を含む配列参照を指定します。数値が大きいほど、優先度が高くなります。
- compress_threshold または set_compress_threshold() メソッド: 値を圧縮するときのしきい値を指定します。指定した数値より大きい値は、格納時および取得時に (zlib を使用して) 自動的に圧縮されます。
- no_rehash または set_norehash() メソッド: 最初に選択したサーバーが使用できなかった場合に、新しいサーバーの検索を無効にします。
- readonly または set_readonly() メソッド: memcached サーバーへの書き込みを無効にします。

Cache::Memcached オブジェクトインスタンスを構成したあとは、set() および get() メソッドを使用すると memcached サーバーの情報を格納および取得できます。キャッシュに格納されているオブジェクトは、Storable モジュールを使用して自動的に直列化および直列化解除されます。

Cache::Memcached インタフェースは、データを格納/取得するために次のメソッドをサポートします。これらは、表に示すように汎用のメソッドと関連しています。

Cache::Memcached の関数	同等の汎用メソッド
get()	汎用の get()。
get_multi(keys)	1 つのクエリーのみを使用して memcache から複数の keys を取得します。キー/値ペアのハッシュ参照を返します。
set()	汎用の set()。
add()	汎用の add()。
replace()	汎用の replace()。
delete()	汎用の delete()。

Cache::Memcached の関数	同等の汎用メソッド
incr()	汎用の incr()。
decr()	汎用の decr()。

Perl および Cache::Memcached モジュールで memcached を使用する完全な例を次に示します。

```
#!/usr/bin/perl

use Cache::Memcached;
use DBI;
use Data::Dumper;

# Configure the memcached server

my $cache = new Cache::Memcached {
   'servers' => [
                 'localhost:11211',
                 ],
   };

# Get the film name from the command line
# memcached keys must not contain spaces, so create
# a key name by replacing spaces with underscores

my $filmname = shift or die "Must specify the film name\n";
my $filmkey = $filmname;
$filmkey =~ s/ /_/;

# Load the data from the cache

my $filmdata = $cache->get($filmkey);

# If the data wasn't in the cache, then we load it from the database

if (!defined($filmdata))
{
   $filmdata = load_filmdata($filmname);

   if (defined($filmdata))
   {

# Set the data into the cache, using the key

 if ($cache->set($filmkey,$filmdata))
       {
          print STDERR "Film data loaded from database and cached\n";
       }
       else
       {
          print STDERR "Couldn't store to cache\n";
 }
   }
   else
   {
     die "Couldn't find $filmname\n";
   }
}
else
{
   print STDERR "Film data loaded from Memcached\n";
}

sub load_filmdata
{
   my ($filmname) = @_;

   my $dsn = "DBI:mysql:database=sakila;host=localhost;port=3306";

   $dbh = DBI->connect($dsn, 'sakila','password');

   my ($filmbase) = $dbh->selectrow_hashref(sprintf('select * from film where title = %s',
                                          $dbh->quote($filmname)));

   if (!defined($filmname))
   {
     return (undef);
   }
```

```
   $filmbase->{stars} =
 $dbh->selectall_arrayref(sprintf('select concat(first_name," ",last_name) ' .
                                  'from film_actor left join (actor) ' .
                                  'on (film_actor.actor_id = actor.actor_id) ' .
                                  ' where film_id=%s',
                                  $dbh->quote($filmbase->{film_id})));

   return($filmbase);
}
```

この例では、Sakila データベースを使用して、データベースから映画のデータを取得し、映画と俳優の複合レコードを memcached に書き込みます。ある映画を要求したときに、それが存在しなかった場合は、この結果が得られます。

```
shell> memcached-sakila.pl "ROCK INSTINCT"
Film data loaded from database and cached
```

キャッシュにすでに追加されている映画にアクセスするとき:

```
shell> memcached-sakila.pl "ROCK INSTINCT"
Film data loaded from Memcached
```

16.6.3.5 Python での MySQL と memcached の使用

Python の memcache モジュールは、memcached サーバーへのインタフェースであり、純粋に Python で (つまり、いずれかの C API を使用せずに) 記述されています。Python Memcached からコピーをダウンロードしてインストールできます。

インストールするには、パッケージをダウンロードして、Python インストーラを実行します。

```
python setup.py install
running install
running bdist_egg
running egg_info
creating python_memcached.egg-info
...
removing 'build/bdist.linux-x86_64/egg' (and everything under it)
Processing python_memcached-1.43-py2.4.egg
creating /usr/lib64/python2.4/site-packages/python_memcached-1.43-py2.4.egg
Extracting python_memcached-1.43-py2.4.egg to /usr/lib64/python2.4/site-packages
Adding python-memcached 1.43 to easy-install.pth file

Installed /usr/lib64/python2.4/site-packages/python_memcached-1.43-py2.4.egg
Processing dependencies for python-memcached==1.43
Finished processing dependencies for python-memcached==1.43
```

インストールが完了すると、memcache モジュールは memcached サーバーに対するクラスベースのインタフェースを提供します。Python のデータ構造体を memcached の項目として格納すると、それらは Python の cPickle または pickle モジュールを使用して自動的に直列化 (文字列値に変換) されます。

新しい memcache インタフェースを作成するには、memcache モジュールをインポートし、memcache.Client クラスの新しいインスタンスを作成します。たとえば、memcached デーモンが localhost 上でデフォルトポートを使用して実行されている場合:

```
import memcache
memc = memcache.Client(['127.0.0.1:11211'])
```

1 つ目の引数は、使用する各 memcached インスタンスのサーバーとポート番号を含む文字列の配列です。デバッグを有効にするには、オプションの debug パラメータを 1 に設定します。

デフォルトでは、項目を複数のサーバー間で分配するために使用されるハッシュ化メカニズムは crc32 です。使用する関数を変更するには、memcache.serverHashFunction の値を、代わりに使用する関数に設定します。例:

```
from zlib import adler32
memcache.serverHashFunction = adler32
```

memcache インスタンス内で使用するサーバーを定義すると、コア関数によって汎用のインタフェース仕様と同じ機能が提供されます。次の表に、サポートされる関数のサマリーを示します。

Python の memcache 関数	同等の汎用関数
get()	汎用の get()。

Python の memcache 関数	同等の汎用関数
get_multi(keys)	指定された keys の配列から複数の値を取得します。キー/値ペアのハッシュ参照を返します。
set()	汎用の set()。
set_multi(dict [, expiry [, key_prefix]])	指定された dict から複数のキー/値ペアを設定します。
add()	汎用の add()。
replace()	汎用の replace()。
prepend(key, value [, expiry])	指定された value を既存の key の値の前に追加します。
append(key, value [, expiry[])	指定された value を既存の key の値のあとに追加します。
delete()	汎用の delete()。
delete_multi(keys [, expiry [, key_prefix]])	keys 配列内の各文字列と一致するハッシュからすべてのキーを削除します。
incr()	汎用の incr()。
decr()	汎用の decr()。

> 注記
>
> Python の memcache モジュール内では、すべての *_multi() 関数でオプションの key_prefix パラメータをサポートします。この文字列は、指定すると、すべてのキールックアップに対するプリフィクスとして使用されます。たとえば、次を呼び出す場合:
>
> ```
> memc.get_multi(['a','b'], key_prefix='users:')
> ```
>
> この関数は、サーバーからキー users:a および users:b を取得します。

MySQL から生データをロードして、memcache インスタンスの情報を格納および取得する例をここに示します。

```
import sys
import MySQLdb
import memcache

memc = memcache.Client(['127.0.0.1:11211'], debug=1);

try:
    conn = MySQLdb.connect (host = "localhost",
                            user = "sakila",
                            passwd = "password",
                            db = "sakila")
except MySQLdb.Error, e:
     print "Error %d: %s" % (e.args[0], e.args[1])
     sys.exit (1)

popularfilms = memc.get('top5films')

if not popularfilms:
    cursor = conn.cursor()
    cursor.execute('select film_id,title from film order by rental_rate desc limit 5')
    rows = cursor.fetchall()
    memc.set('top5films',rows,60)
    print "Updated memcached with MySQL data"
else:
    print "Loaded data from memcached"
    for row in popularfilms:
        print "%s, %s" % (row[0], row[1])
```

はじめて実行すると、データが MySQL データベースからロードされ、memcached サーバーに格納されます。

```
shell> python memc_python.py
Updated memcached with MySQL data
```

データは cPickle/pickle を使用して自動的に直列化されるため、データを memcached からロードしたときは、そのオブジェクトを直接使用できます。上の例では、memcached に格納された情報が Python DB のカーソルから取得した行の形式になっています。(60 秒の失効時間内に) 情報にアクセスすると、データが memcached からロードされ、ダンプされます。

```
shell> python memc_python.py
```

```
Loaded data from memcached
2, ACE GOLDFINGER
7, AIRPLANE SIERRA
8, AIRPORT POLLOCK
10, ALADDIN CALENDAR
13, ALI FOREVER
```

直列化と直列化解除は自動的に行われます。Python データの直列化はほかのインタフェースや言語と互換性がない可能性があるため、初期化時に使用される直列化モジュールを変更できます。たとえば、ある言語で記述されたスクリプトを使用して複雑なデータ構造を格納し、別の言語で記述されたスクリプトでそれらにアクセスするときは、JSON フォーマットを使用できます。

16.6.3.6 PHP での MySQL と memcached の使用

PHP は、PECL 拡張によって Memcache 関数のサポートを提供します。PHP の memcache 拡張を有効にするには、PHP をソースからビルドするときに configure に --enable-memcache オプションを指定してビルドします。

Red Hat ベースのサーバーをインストールする場合は、php-pecl-memcache RPM をインストールできます。

```
root-shell> yum --install php-pecl-memcache
```

Debian ベースの配布では、php-memcache パッケージを使用します。

グローバル実行時構成オプションを設定するには、php.ini ファイル内に構成オプションの値を指定します。次の表に、個々のグローバル実行時構成オプションの名前、デフォルト値、および説明を示します。

構成オプション	デフォルト	説明
memcache.allow_failover	1	最初に選択したサーバーに障害が発生した場合にリスト内の別のサーバーをクエリーするかどうか指定します。
memcache.max_failover_attempts	20	失敗を返す前に試行するサーバーの数を指定します。
memcache.chunk_size	8192	memcached サーバーとのデータ交換に使用するネットワークチャンクのサイズを定義します。
memcache.default_port	11211	memcached サーバーとの通信時に使用するデフォルトポートを定義します。
memcache.hash_strategy	standard	使用するハッシュ方式を指定します。consistent に設定すると、キーをほかのサーバーに再マップしないで、サーバーをプールに追加したりプールから削除したりできるようになります。standard に設定すると、格納時に別のサーバーを使用する可能性がある古い (モジュラ) 方式が使用されます。
memcache.hash_function	crc32	キーをサーバーにマップするときに使用する関数を指定します。crc32 にすると、標準の CRC32 ハッシュが使用されます。fnv にすると、FNV-1a ハッシュ化アルゴリズムが使用されます。

memcached サーバーへの接続を作成するには、新しい Memcache オブジェクトを作成してから、接続オプションを指定します。例:

```
<?php

$cache = new Memcache;
$cache->connect('localhost',11211);
?>
```

これにより、指定したサーバーへの接続がただちに開きます。

複数の memcached サーバーを使用するには、addServer() を使用して memcache オブジェクトにサーバーを追加する必要があります。

```
bool Memcache::addServer ( string $host [, int $port [, bool $persistent
            [, int $weight [, int $timeout [, int $retry_interval
            [, bool $status [, callback $failure_callback
```

```
                ]]]]]]] )
```

php-memcache モジュール内のサーバー管理メカニズムは、このインタフェースの重要な部分であり、memcached インスタンスへのメインインタフェースとハッシュ化メカニズムによるインスタンスの選択方法を制御します。

2 つの memcached インスタンスへの単純な接続を作成するには:

```
<?php

$cache = new Memcache;
$cache->addServer('192.168.0.100',11211);
$cache->addServer('192.168.0.101',11211);
?>
```

このシナリオでは、インスタンス接続は明示的に開かれず、値を格納または取得しようとしたときにのみ開かれます。memcached インスタンスへの永続的な接続を有効にするには、$persistent 引数を true に設定します。これはデフォルト設定であり、これによって接続が開いたままになります。

別のインスタンスへのキー分布を制御するには、memcache.hash_strategy グローバル設定を使用します。これは、選択に使用されるハッシュ化メカニズムを設定します。各サーバーに別の重みを追加することもできます。この重みは、インスタンスエントリがインスタンスリストに現れる回数を効果的に増やすことによって、ほかのインスタンスよりもそのインスタンスが選択される可能性を増やします。この重みを設定するには、$weight 引数を 1 より大きい値に設定します。

情報を設定および取得する関数は、このテーブルに示すように、memcached に用意されている汎用関数インタフェースと同等です。

PECL の memcache 関数	汎用関数
get()	汎用の get()。
set()	汎用の set()。
add()	汎用の add()。
replace()	汎用の replace()。
delete()	汎用の delete()。
increment()	汎用の incr()。
decrement()	汎用の decr()。

PECL の memcache インタフェースの完全な例を次に示します。このコードは、ユーザーが映画名を指定したときに、Sakila データベースから映画のデータをロードします。memcached インスタンスに格納されているデータは mysqli の結果行として記録され、この情報は API によって自動的に直列化されます。

```
<?php

$memc = new Memcache;
$memc->addServer('localhost','11211');

if(empty($_POST['film'])) {
?>
  <html xmlns="http://www.w3.org/1999/xhtml" xml:lang="en" lang="en">
    <head>
      <meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
      <title>Simple Memcache Lookup</title>
    </head>
    <body>
      <form method="post">
        <p><b>Film</b>: <input type="text" size="20" name="film"></p>
        <input type="submit">
      </form>
      <hr/>
<?php

} else {

    echo "Loading data...\n";

    $film   = htmlspecialchars($_POST['film'], ENT_QUOTES, 'UTF-8');
    $mfilms = $memc->get($film);

    if ($mfilms) {
```

```
    printf("<p>Film data for %s loaded from memcache</p>", $mfilms['title']);

    foreach (array_keys($mfilms) as $key) {
      printf("<p><b>%s</b>: %s</p>", $key, $mfilms[$key]);
    }

  } else {

    $mysqli = mysqli('localhost','sakila','password','sakila');

    if (mysqli_connect_error()) {
      sprintf("Database error: (%d) %s", mysqli_connect_errno(), mysqli_connect_error());
      exit;
    }

    $sql = sprintf('SELECT * FROM film WHERE title="%s"', $mysqli->real_escape_string($film));

    $result = $mysqli->query($sql);

    if (!$result) {
      sprintf("Database error: (%d) %s", $mysqli->errno, $mysqli->error);
      exit;
    }

    $row = $result->fetch_assoc();

    $memc->set($row['title'], $row);

    printf("<p>Loaded (%s) from MySQL</p>", htmlspecialchars($row['title'], ENT_QUOTES, 'UTF-8');
  }
}
?>
 </body>
</html>
```

PHP では、PHP と関連する Apache インスタンスの実行が継続するかぎり、memcached インスタンスへの接続は開いたまま維持されます。実行中のインスタンスでサーバーをリストに追加したり、リストから削除したりすると (たとえば、追加のサーバーが指定された別のスクリプトを起動したときなど)、接続は共有されますが、スクリプト内で明示的に構成されたインスタンスのみが選択されます。

スクリプト内でサーバーリストに変更を加えることによって問題が発生しないようにするには、必ず整合ハッシュ化メカニズムを使用してください。

16.6.3.7 Ruby での MySQL と memcached の使用

Ruby には、memcached へのインタフェースとなる異なるモジュールがあります。Ruby-MemCache クライアントライブラリは、libmemcached などの外部ライブラリを必要としない memcached へのネイティブインタフェースを提供します。インストーラパッケージは、http://www.deveiate.org/projects/RMemCache から入手できます。

インストールするには、パッケージを抽出して install.rb を実行します。

```
shell> install.rb
```

RubyGems がある場合は、Ruby-MemCache gem をインストールできます。

```
shell> gem install Ruby-MemCache
Bulk updating Gem source index for: http://gems.rubyforge.org
Install required dependency io-reactor? [Yn]  y
Successfully installed Ruby-MemCache-0.0.1
Successfully installed io-reactor-0.05
Installing ri documentation for io-reactor-0.05...
Installing RDoc documentation for io-reactor-0.05...
```

Ruby の内部から memcached インスタンスを使用するには、MemCache オブジェクトの新しいインスタンスを作成します。

```
require 'memcache'
memc = MemCache::new '192.168.0.100:11211'
```

ハッシュ化時にサーバーが選択される可能性を増やすため、各サーバーに重みを追加するには、サーバーのホスト名/ポート文字列の末尾に重みカウントを追加します。

```
require 'memcache'
```

```
memc = MemCache::new '192.168.0.100:11211:3'
```

既存のリストにサーバーを追加するには、それらを MemCache オブジェクトの末尾に直接追加します。

```
memc += ["192.168.0.101:11211"]
```

データをキャッシュに設定するには、標準の Ruby ハッシュオブジェクトとまったく同じように機能する新しいキャッシュオブジェクト内で、キーに値を割り当てます。

```
memc["key"] = "value"
```

または、値を取得するには:

```
print memc["key"]
```

より明示的なアクションとして、次の表に要約したように、memcached の主要な API 関数を模倣するメソッドインタフェースを使用できます。

Ruby の MemCache メソッド	同等の memcached API 関数
get()	汎用の get()。
get_hash(keys)	複数の keys の値を取得し、キーとその値のハッシュとして情報を返します。
set()	汎用の set()。
set_many(pairs)	ハッシュ pairs 内のキーおよび値の値を設定します。
add()	汎用の add()。
replace()	汎用の replace()。
delete()	汎用の delete()。
incr()	汎用の incr()。
decr()	汎用の decr()。

16.6.3.8 Java での MySQL と memcached の使用

Java に含まれる com.danga.MemCached クラスは、memcached インスタンスへのネイティブインタフェースを提供します。このクライアントは、https://github.com/gwhalin/Memcached-Java-Client/downloads から入手できます。この Java クラスは libmemcached と互換性があるハッシュを使用するため、同じ memcached インスタンスにアクセスする Java と libmemcached のアプリケーションを混在および調和できます。Java とその他のインタフェース間の直列化には互換性がありません。これが問題になる場合は、JSON または同様の非バイナリ直列化フォーマットを使用してください。

ほとんどのシステムでは、パッケージをダウンロードして jar を直接使用できます。

com.danga.MemCached インタフェースを使用するには、MemCachedClient インスタンスを作成し、SockIOPool を構成してサーバーリストを構成します。このプールを指定することで、サーバーリスト、重み、および接続パラメータを設定し、クライアントと構成した memcached インスタンス間の接続を最適化します。

一般に、1 つのクラス内で memcached インタフェースを一度構成すれば、あとはこのインタフェースを残りのアプリケーション全体で使用できます。

たとえば、基本的なインタフェースを作成するには、最初に MemCachedClient とベースの SockIOPool 設定を構成します。

```
public class MyClass {

   protected static MemCachedClient mcc = new MemCachedClient();

   static {

      String[] servers =
         {
            "localhost:11211",
         };

      Integer[] weights = { 1 };
```

```
SockIOPool pool = SockIOPool.getInstance();

pool.setServers( servers );
pool.setWeights( weights );
```

上の例では、使用する memcached インスタンスの配列を作成することによってサーバーリストが構成されています。次に、各サーバーに対する個別の重みを構成します。

接続に関する残りのプロパティーはオプションですが、プールのパラメータを設定することによって接続の数値 (初期接続数、最小接続数、最大接続数、およびアイドルタイムアウト) を設定できます。

```
pool.setInitConn( 5 );
pool.setMinConn( 5 );
pool.setMaxConn( 250 );
pool.setMaxIdle( 1000 * 60 * 60 * 6
```

パラメータを構成したあとは、接続プールを初期化します。

```
pool.initialize();
```

これで、プールと memcached インスタンスへの接続を使用できます。

特定のキーの格納時に使用されるサーバーの選択に使用するハッシュ化アルゴリズムを設定するには、pool.setHashingAlg() を使用します。

```
pool.setHashingAlg( SockIOPool.NEW_COMPAT_HASH );
```

有効な値は、基本的なモジュラハッシュ化アルゴリズムでもある NEW_COMPAT_HASH、OLD_COMPAT_HASH、および NATIVE_HASH です。整合ハッシュ化アルゴリズムの場合は、CONSISTENT_HASH を使用します。これらの定数は、libmemcached 内の対応するハッシュ設定と同等です。

次の表に、Java の com.danga.MemCached メソッドと memcached インタフェース仕様に含まれる同等の汎用メソッドの概要を示します。

Java の com.danga.MemCached メソッド	同等の汎用メソッド
get()	汎用の get()。
getMulti(keys)	複数の keys の値を取得し、キーに java.lang.String、対応する値に java.lang.Object をそれぞれ使用して、ハッシュマップとして情報を返します。
set()	汎用の set()。
add()	汎用の add()。
replace()	汎用の replace()。
delete()	汎用の delete()。
incr()	汎用の incr()。
decr()	汎用の decr()。

16.6.3.9 memcached の TCP テキストプロトコルの使用

memcached サーバーとの通信は、TCP または UDP プロトコルを介して実現されます。TCP プロトコルを使用すると、単純なテキストベースのインタフェースを使用して情報を交換できます。

memcached と通信するときは、サーバー用に構成されたポートを使用するとサーバーに接続できます。サーバーとの接続を開くときに、認証やログインは必要ありません。接続した直後から、サーバーへのコマンド送信を開始できます。完了したときは、特別な切断コマンドを送信せずに接続を終了できます。クライアントは、待機時間を短縮し、パフォーマンスを向上させるため、接続を開いたままにすることが推奨されます。

データは 2 つの形式で memcached サーバーに送信されます。

- サーバーにコマンドを送信し、サーバーから応答を受信するために使用されるテキスト行。
- 特定のキーに対する値の情報を送受信するために使用される非構造化データ。データは受信したときと同じフォーマットでクライアントに返されます。

テキスト行 (コマンドと応答) および非構造化データは、どちらも常に文字列 \r\n で終了します。格納されるデータにこのシーケンスが含まれている可能性があるため、(非構造化データが送信される前にクライアントから返される) データの長さを使用してデータの末尾が特定されるはずです。

サーバーに対するコマンドは、操作に応じて構造化されています。

- ストレージコマンド: set、add、replace、append、prepend、cas

  サーバーに対するストレージコマンドは、次の形式を取ります。

  ```
  command key [flags] [exptime] length [noreply]
  ```

  または、コンペアアンドスワップ (CAS) を使用する場合:

  ```
  cas key [flags] [exptime] length [casunique] [noreply]
  ```

  ここでは:

  - command: コマンド名。
    - set: キーに対する値を格納します
    - add: キーがまだ存在しない場合に、そのキーに対してこの値を格納します
    - replace: キーがすでに存在する場合に、そのキーに対してこのキーを格納します
    - append: 指定されたキー値の末尾に指定された値を追加します。flags および exptime 引数は使用しないでください。
    - prepend: 指定されたキーの指定された値の末尾にキャッシュ内の現在の値を追加します。flags および exptime 引数は使用しないでください。
    - cas: 指定された casunique が一致する場合にのみ、指定されたキーを指定された値に設定します。これは、自分が情報を最後にフェッチして以降どのユーザーもその情報を更新していない場合に、その情報を変更するのと実質的に同等です。
  - key: キー。すべてのデータは特定のキーを使用して格納されます。キーに制御文字および空白文字を含めることはできません。キーの最大サイズは 250 文字です。
  - flags: 操作のフラグ (整数)。memcached のフラグは透過的です。memcached サーバーはフラグの内容を無視します。これらは、クライアントが情報のタイプを示すために使用します。memcached 1.2.0 以前では、この値は 16 ビット整数値です。memcached 1.2.1 以降では、この値は 32 ビット整数です。
  - exptime: 失効時間 (失効がない場合は 0)。
  - length: 指定された値ブロックの長さ (バイト単位、終端の \r\n 文字を除く)。
  - casunique: 既存エントリの一意の 64 ビット値。これは、既存の値と比較するために使用されます。cas 更新を発行するときに、gets コマンドから返された値を使用します。
  - noreply: コマンドに応答しないようにサーバーに指示します。

  たとえば、値 abcdef をキー xyzkey に格納するには、次を使用します。

  ```
  set xyzkey 0 0 6\r\nabcdef\r\n
  ```

  サーバーからの戻り値は、ステータスまたはエラー情報を示す 1 行です。詳細については、表 16.3「memcached プロトコルの応答」を参照してください。

- 取得コマンド: get、gets

  取得コマンドは次の形式を取ります。

  ```
  get key1 [key2 .... keyn]
  gets key1 [key2 ... keyn]
  ```

  要求する各キーを空白文字で区切って、複数のキーをコマンドに指定できます。

  サーバーは、次の形式の情報行で応答します。

  ```
  VALUE key flags bytes [casunique]
  ```

ここでは:

- key: キー名。
- flags: 値を格納したときに memcached サーバーに提供されたフラグ整数の値。
- bytes: 格納されている値のサイズ (終端の \r\n 文字シーケンスを除く)。
- casunique: 項目を識別する一意の 64 ビット整数。

情報行の直後に値のデータブロックが続きます。例:

```
get xyzkey\r\n
VALUE xyzkey 0 6\r\n
abcdef\r\n
```

複数のキーを要求した場合は、見つかったキーごとに情報行とデータブロックが返されます。要求されたキーがキャッシュに存在しない場合、情報は返されません。

- 削除コマンド: delete

削除コマンドは次の形式を取ります。

```
delete key [time] [noreply]
```

ここでは:

- key: キー名。
- time: クライアントがサーバーにこのキーの add または replace コマンドを拒否するように求める秒単位の時間 (または特定の Unix 時間)。この期間中は、すべての add、replace、get、および gets コマンドが失敗します。set 操作は成功します。この期間が過ぎると、そのキーは永続的に削除され、すべてのコマンドが許可されます。

  指定しなかった場合、この値は 0 (ただちに削除) とみなされます。
- noreply: コマンドに応答しないようにサーバーに指示します。

このコマンドに対する応答は、キーが正常に削除されたことを示す DELETED か、指定されたキーが見つからなかったことを示す NOT_FOUND のいずれかです。

- 増分/減分: incr、decr

増分および減分コマンドは、個別の取得/設定シーケンスを実行せずにサーバー内のキーの値を変更します。これらの操作は、現在格納されている値が 64 ビット整数であることを前提としています。格納されている値が 64 ビット整数でない場合は、増分または減分操作を適用する前にその値が 0 とみなされます。

増分および減分コマンドは次の形式を取ります。

```
incr key value [noreply]
decr key value [noreply]
```

ここでは:

- key: キー名。
- value: 増分または減分値として使用される整数。
- noreply: コマンドに応答しないようにサーバーに指示します。

応答は次のとおりです。

- NOT_FOUND: 指定されたキーが見つかりませんでした。
- value: 指定されたキーに関連付けられた新しい値。

値は符号なしとみなされます。decr 操作では、値が 0 未満に減分されることはありません。incr 操作では、64 ビットの最大値で値が折り返します。

- 統計コマンド: stats

stats コマンドは、memcached インスタンスの現在のステータスとそこに格納されているデータに関する詳細な統計情報を提供します。

統計コマンドは次の形式を取ります。

```
STAT [name] [value]
```

ここでは:

- name: 返される統計の名前 (オプション)。指定しなかった場合は、一般統計が返されます。
- value: 特定の統計操作を実行するときに使用される特定の値。

戻り値は、次のようにフォーマットされた統計データのリストです。

```
STAT name value
```

統計は END という 1 行で終了します。

詳細については、セクション16.6.4「memcached の統計の取得」を参照してください。

参考のため、サポートされる各種のコマンドとその形式のリストを次に示します。

表 16.2 memcached コマンドのリファレンス

コマンド	コマンド形式
set	set key flags exptime length、set key flags exptime length noreply
add	add key flags exptime length、add key flags exptime length noreply
replace	replace key flags exptime length、replace key flags exptime length noreply
append	append key length、append key length noreply
prepend	prepend key length、prepend key length noreply
cas	cas key flags exptime length casunique、cas key flags exptime length casunique noreply
get	get key1 [key2 ... keyn]
gets	
delete	delete key、delete key noreply、delete key expiry、delete key expiry noreply
incr	incr key、incr key noreply、incr key value、incr key value noreply
decr	decr key、decr key noreply、decr key value、decr key value noreply
stat	stat、stat name、stat name value

サーバーにコマンドを送信すると、サーバーからの応答は次の表に示すいずれかの設定になります。サーバーからの応答値はすべて \r\n で終了します。

表 16.3 memcached プロトコルの応答

文字列	説明
STORED	値が正常に格納されました。
NOT_STORED	値が格納されませんでしたが、エラーが原因ではありません。値を追加または (存在する場合に) 更新するコマンド (add や replace など)、または項目がすでに削除するように設定されているときのコマンドに対応します。
EXISTS	cas コマンドを使用したときに、格納しようとしている項目がすでに存在し、最後のチェック以降に変更されています。
NOT_FOUND	格納、更新、または削除しようとしている項目が存在しないか、すでに削除されています。
ERROR	存在しないコマンド名を送信しました。
CLIENT_ERROR errorstring	入力行にエラーがありました。詳細は errorstring に含まれています。
SERVER_ERROR errorstring	サーバーにエラーがあったため、情報を返すことができません。極端な場合には、このエラーの発生後にサーバーがクライアントから切断されることがあります。

文字列	説明
VALUE keys flags length	要求されたキーが見つかり、格納されている key、flags、およびデータブロックが指定された length で返されます。
DELETED	要求されたキーがサーバーから削除されました。
STAT name value	統計データの行。
END	統計データの終わり。

16.6.4 memcached の統計の取得

memcached システムには、キャッシュに格納されるデータに関する情報、キャッシュヒット率、および個々の項目を格納するために使用されるスラブ割り当てを介したメモリーの使用状況と情報の分布に関する詳細情報を収集する統計システムが組み込まれています。統計は、主要な統計を提供する基本レベルと、memcached サーバーの特定の分野に関するより具体的な統計の両方が提供されます。

これらの情報は、キャッシュとメモリーの使用状況を正しいレベルで取得し、スラブ割り当てと構成プロパティーが最適なレベルで設定されていることを確認するのに役立ちます。

統計インタフェースは標準の memcached プロトコルを介して使用できるため、telnet を使用して memcached に接続してレポートにアクセスできます。付属の memcached-tool には、セクション16.6.4.2「memcached のスラブ統計」およびセクション16.6.4.1「memcached の一般統計」の情報を取得するためのサポートが含まれています。詳細については、セクション16.6.4.6「memcached-tool の使用」を参照してください。

また、ほとんどの言語 API インタフェースに、サーバーから統計を取得するための関数が用意されています。

たとえば、telnet を使用して基本統計を取得するには:

```
shell> telnet localhost 11211
Trying ::1...
Connected to localhost.
Escape character is '^]'.
stats
STAT pid 23599
STAT uptime 675
STAT time 1211439587
STAT version 1.2.5
STAT pointer_size 32
STAT rusage_user 1.404992
STAT rusage_system 4.694685
STAT curr_items 32
STAT total_items 56361
STAT bytes 2642
STAT curr_connections 53
STAT total_connections 438
STAT connection_structures 55
STAT cmd_get 113482
STAT cmd_set 80519
STAT get_hits 78926
STAT get_misses 34556
STAT evictions 0
STAT bytes_read 6379783
STAT bytes_written 4860179
STAT limit_maxbytes 67108864
STAT threads 1
END
```

Perl と Cache::Memcached モジュールを使用するときは、stats() 関数によって、接続オブジェクト内に現在構成されているサーバーに関する情報と memcached サーバーの全体的な統計が返されます。

たとえば、次の Perl スクリプトは統計を取得し、返されたハッシュ参照をダンプします。

```
use Cache::Memcached;
use Data::Dumper;

my $memc = new Cache::Memcached;
$memc->set_servers(\@ARGV);

print Dumper($memc->stats());
```

前の Telnet の例で使用したのと同じ memcached に対して実行すると、ホスト単位の統計と全体的な統計を含むハッシュ参照が得られます。

```
$VAR1 = {
  'hosts' => {
       'localhost:11211' => {
                'misc' => {
                     'bytes' => '2421',
                     'curr_connections' => '3',
                     'connection_structures' => '56',
                     'pointer_size' => '32',
                     'time' => '1211440166',
                     'total_items' => '410956',
                     'cmd_set' => '588167',
                     'bytes_written' => '35715151',
                     'evictions' => '0',
                     'curr_items' => '31',
                     'pid' => '23599',
                     'limit_maxbytes' => '67108864',
                     'uptime' => '1254',
                     'rusage_user' => '9.857805',
                     'cmd_get' => '838451',
                     'rusage_system' => '34.096988',
                     'version' => '1.2.5',
                     'get_hits' => '581511',
                     'bytes_read' => '46665716',
                     'threads' => '1',
                     'total_connections' => '3104',
                     'get_misses' => '256940'
                   },
                'sizes' => {
                     '128' => '16',
                     '64' => '15'
                   }
              }
     },
  'self' => {},
  'total' => {
       'cmd_get' => 838451,
       'bytes' => 2421,
       'get_hits' => 581511,
       'connection_structures' => 56,
       'bytes_read' => 46665716,
       'total_items' => 410956,
       'total_connections' => 3104,
       'cmd_set' => 588167,
       'bytes_written' => 35715151,
       'curr_items' => 31,
       'get_misses' => 256940
     }
    };
```

統計はいくつかの異なるセクションに分かれており、stats コマンドにタイプを追加して要求できます。個々の統計出力については、次のセクションで詳しく説明します。

- 一般統計については、セクション16.6.4.1「memcached の一般統計」を参照してください。
- スラブ統計 (slabs) については、セクション16.6.4.2「memcached のスラブ統計」を参照してください。
- 項目統計 (items) については、セクション16.6.4.3「memcached の項目統計」を参照してください。
- サイズ統計 (sizes) については、セクション16.6.4.4「memcached のサイズ統計」を参照してください。
- 詳細ステータス (detail) については、セクション16.6.4.5「memcached の詳細統計」を参照してください。

16.6.4.1 memcached の一般統計

一般統計の出力には、memcached インスタンスのパフォーマンスと使用の概要が示されます。コマンドによって返される統計とそれらの意味を、次の表に示します。

各統計値の値の型を定義するために、次の用語が使用されます。

- 32u: 32 ビット符号なし整数
- 64u: 64 ビット符号なし整数
- 32u:32u: コロンで区切られた 2 つの 32 ビット符号なし整数
- String: 文字列

統計	データ型	説明	バージョン
pid	32u	memcached インスタンスのプロセス ID。	
uptime	32u	この memcached インスタンスの稼働時間 (秒単位)。	
time	32u	現在の時間 (エポックとして)。	
version	string	このインスタンスのバージョン文字列。	
pointer_size	string	ビット単位で指定されたこのホストのポインタのサイズ (32 または 64)。	
rusage_user	32u:32u	このインスタンスの合計ユーザー時間 (秒:ミリ秒)。	
rusage_system	32u:32u	このインスタンスの合計システム時間 (秒:ミリ秒)。	
curr_items	32u	このインスタンスによって格納された項目の現在数。	
total_items	32u	このインスタンスの存続期間中に格納された項目の合計数。	
bytes	64u	このサーバーが項目を格納するために使用した現在のバイト数。	
curr_connections	32u	現在開いている接続の数。	
total_connections	32u	サーバーが実行を開始してから開かれた接続の合計数。	
connection_structures	32u	サーバーが割り当てた接続構造の数。	
cmd_get	64u	取得要求 (get 操作) の合計数。	
cmd_set	64u	ストレージ要求 (set 操作) の合計数。	
get_hits	64u	要求され、存在することがわかったキーの数。	
get_misses	64u	要求され、見つからなかった項目の数。	
delete_hits	64u	削除され、存在することがわかったキーの数。	1.3.x
delete_misses	64u	削除され、見つからなかった項目の数。	1.3.x
incr_hits	64u	増分され、存在することがわかったキーの数。	1.3.x
incr_misses	64u	増分され、見つからなかった項目の数。	1.3.x
decr_hits	64u	減分され、存在することがわかったキーの数。	1.3.x
decr_misses	64u	減分され、見つからなかった項目の数。	1.3.x
cas_hits	64u	コンペアアンドスワップが行われ、存在することがわかったキーの数。	1.3.x
cas_misses	64u	コンペアアンドスワップが行われ、見つからなかった項目の数。	1.3.x
cas_badvalue	64u	コンペアアンドスワップが行われたが、比較対象の (元の) 値が指定された値と一致しなかったキーの数。	1.3.x
evictions	64u	新しい項目用のメモリーを解放するためにキャッシュから削除された有効な項目数。	
bytes_read	64u	このサーバーがネットワークから読み取った合計バイト数。	
bytes_written	64u	このサーバーがネットワークに送信した合計バイト数。	
limit_maxbytes	32u	このサーバーで格納への使用が許可されたバイト数。	
threads	32u	要求されたワーカースレッドの数。	
conn_yields	64u	接続の生成数 (-R オプションに関連します)。	1.4.0

ここに示した統計の中でもっとも有用なものは、キャッシュヒット、キャッシュミス、および削除の数です。

get_misses の数が大きい場合は、キャッシュへの情報移入がまだ完了していないことを示している可能性があります。この数は、時間の経過とともにキャッシュの get_hits の数に比べて減少します。ただし、長時間の実行後にキャッシュミスの数がキャッシュヒットより多い場合は、キャッシュのサイズが小さすぎるため、メモリーの合計サイズまたは memcached インスタンスの数を増やしてヒット率を向上させる必要があることを示す可能性があります。

キャッシュの evictions の数が (特に格納されている項目の数に比べて) 多い場合は、キャッシュが小さすぎて、定期的にキャッシュし続ける必要がある情報量を保持できないことを示します。キャッシュ内に項目を保持する代

わりに、新しい項目が入るように項目が削除されるため、キャッシュ内の項目の回転率は常に高くなりますが、キャッシュの効率は下がります。

16.6.4.2 memcached のスラブ統計

slabs 統計を取得するには、stats slabs コマンドまたは同等の API を使用します。

スラブ統計には、キャッシュ内に情報を格納するために作成され、割り当てられたスラブに関する情報が示されます。個々のスラブクラスに関する情報とスラブの全体的な統計の両方を取得します。

```
STAT 1:chunk_size 104
STAT 1:chunks_per_page 10082
STAT 1:total_pages 1
STAT 1:total_chunks 10082
STAT 1:used_chunks 10081
STAT 1:free_chunks 1
STAT 1:free_chunks_end 10079
STAT 9:chunk_size 696
STAT 9:chunks_per_page 1506
STAT 9:total_pages 63
STAT 9:total_chunks 94878
STAT 9:used_chunks 94878
STAT 9:free_chunks 0
STAT 9:free_chunks_end 0
STAT active_slabs 2
STAT total_malloced 67083616
END
```

各スラブクラスの個々の統計には、プリフィクスとしてスラブ ID が付いています。割り当てられた各スラブには、サイズのもっとも小さいものから大きいものまで順に、一意の ID が付けられます。このプリフィクス番号は、指定された増大係数から計算されたチャンクに基づくスラブクラス番号を示しています。したがって、この例では 1 が最初のチャンクサイズであり、9 が 9 番目に割り当てられたチャンクサイズです。

各チャンクサイズに対して返されるパラメータと各パラメータの説明を、次の表に示します。

統計	説明	バージョン
chunk_size	このスラブクラス内の各チャンクに割り当てられたスペース。	
chunks_per_page	このスラブクラスの 1 つのページに含まれるチャンクの数。	
total_pages	このスラブクラスに割り当てられたページの数。	
total_chunks	このスラブクラスに割り当てられたチャンクの数。	
used_chunks	項目に割り当てられたチャンクの数。	
free_chunks	項目にまだ割り当てられていないチャンクの数。	
free_chunks_end	最後に割り当てられたページの最後にある空きチャンクの数。	
get_hits	このチャンクの取得ヒットの数	1.3.x
cmd_set	このチャンクに対する設定コマンドの数	1.3.x
delete_hits	このチャンクの削除ヒットの数	1.3.x
incr_hits	このチャンクの増分ヒットの数	1.3.x
decr_hits	このチャンクの減分ヒットの数	1.3.x
cas_hits	このチャンクの CAS ヒットの数	1.3.x
cas_badval	このチャンクで既存の値が一致しなかった CAS ヒットの数	1.3.x
mem_requested	このチャンク内の要求メモリーの実際のメモリー量	1.4.1

次に示す追加の統計は、チャンク単位ではなく、サーバー全体の情報を対象としています。

統計	説明	バージョン
active_slabs	割り当てられたスラブクラスの合計数。	
total_malloced	スラブのページに割り当てられたメモリーの合計量。	

スラブ統計のキー値は、chunk_size と、対応する total_chunks および used_chunks パラメータです。これらは、システム内のチャンクのサイズ使用状況の目安になります。1 つのキー/値ペアが適切なサイズのチャンクに配置されます。

これらの統計から、サイズとチャンクの割り当ておよび分布を把握できます。多くの項目をいくつかの大幅に異なるサイズで格納する場合は、チャンクとメモリーの浪費を防ぐため、チャンクサイズの増大係数をより大きなステップで増加するように調整してください。増大係数が不適切であることを示すわかりやすい兆候は、スラブクラスの数が多いのに、各スラブ内で実際に使用されているチャンクの数が比較的少ないことです。増大係数を増やすと、作成されるスラブクラスの数が減るため、割り当てられたページをより有効に利用できます。

16.6.4.3 memcached の項目統計

items 統計を取得するには、stats items コマンドまたは同等の API を使用します。

items 統計には、特定のスラブクラス内に割り当てられた個々の項目に関する情報が示されます。

```
STAT items:2:number 1
STAT items:2:age 452
STAT items:2:evicted 0
STAT items:2:evicted_nonzero 0
STAT items:2:evicted_time 2
STAT items:2:outofmemory 0
STAT items:2:tailrepairs 0
...
STAT items:27:number 1
STAT items:27:age 452
STAT items:27:evicted 0
STAT items:27:evicted_nonzero 0
STAT items:27:evicted_time 2
STAT items:27:outofmemory 0
STAT items:27:tailrepairs 0
```

各統計に付けられたプリフィクス番号は、stats slabs 統計によって返される対応するチャンクサイズに関連しています。結果として表示されるのは、各スラブサイズ内の各チャンクに格納された項目数と、それらの存続期間、削除カウント、およびメモリー不足カウントです。この統計のサマリーを次の表に示します。

統計	説明	
number	このスラブクラスに現在格納されている項目の数。	
age	このスラブクラス内でもっとも古い項目の存続期間 (秒単位)。	
evicted	新しいエントリを入れるために削除された項目の数。	
evicted_time	エントリが最後に削除された時間	
evicted_nonzero	0 以外のエントリが最後に削除された時間	1.4.0
outofmemory	このスラブクラスの、メモリー不足エラーが発生した項目の数 (-M コマンド行オプションが有効な場合のみの値)。	
tailrepairs	特定の ID のエントリを修復する必要が生じた回数	

項目レベルの統計は、特定のスラブ内に格納されている項目の数、およびそれらの更新と再利用のレベルの特定に使用できます。これは、ほかと比べて削除の回数が非常に多いスラブクラスがあるかどうかを識別するのに役立ちます。

16.6.4.4 memcached のサイズ統計

サイズ統計を取得するには、stats sizes コマンドまたは同等の API を使用します。

サイズ統計には、キャッシュ内の各サイズの項目のサイズと数に関する情報が示されます。これらの情報は 2 つのカラムで返されます。1 つ目のカラムは項目のサイズ (もっとも近い 32 バイト境界に丸められます) で、2 つ目のカラムはキャッシュに含まれるそのサイズの項目の数です。

```
96 35
128 38
160 807
192 804
224 410
256 222
288 83
320 39
352 53
384 33
416 64
```

```
448 51
480 30
512 54
544 39
576 10065
```

> **注意**
>
> この統計を実行すると、キャッシュから各項目が読み取られ、そのサイズが計算されるたびにキャッシュが口ックアップします。大規模なキャッシュでは、これに時間がかかり、プロセスが完了するまで設定または取得操作を実行できない場合があります。

項目サイズの統計は、格納しているオブジェクトのサイズを特定する場合にのみ役立ちます。実際のメモリー割り当てはチャンクサイズとページサイズにのみ関連しているため、これらの情報は綿密なデバッグまたは診断セッションでのみ役立ちます。

16.6.4.5 memcached の詳細統計

memcached 1.3.x 以降では、キャッシュに格納された個々のキーに対する取得、設定、および削除操作に関する詳細統計を有効にして取得し、それらの試行によって特定のキーがヒットした (見つかった) かどうかを特定できます。これらの操作は、詳細統計分析がオンになっている間だけ記録されます。

詳細統計を有効にするには、memcached サーバーに stats detail on コマンドを送信する必要があります。

```
$ telnet localhost 11211
Trying 127.0.0.1...
Connected to tiger.
Escape character is '^]'.
stats detail on
OK
```

個々の統計は、キー (サーバーに現在格納されていないキーを含む) に対する get、set、および del 操作のたびに記録されます。たとえば、キー abckey の値を取得しようとして存在しなかった場合、詳細統計が有効になっている間は、指定されたキーが現在格納されていなくても、そのキーに対して行われた get 操作が記録されます。hits (つまり、サーバーに存在するキーに対する get または del 操作の数) もカウントされます。

詳細統計をオフにするには、memcached サーバーに対して stats detail off コマンドを送信します。

```
$ telnet localhost 11211
Trying 127.0.0.1...
Connected to tiger.
Escape character is '^]'.
stats detail on
OK
```

プロセス中に記録された詳細統計を取得するには、memcached サーバーに対して stats detail dump コマンドを送信します。

```
stats detail dump
PREFIX hykkey get 0 hit 0 set 1 del 0
PREFIX xyzkey get 0 hit 0 set 1 del 0
PREFIX yukkey get 1 hit 0 set 0 del 0
PREFIX abckey get 3 hit 3 set 1 del 0
END
```

詳細統計の情報を使用して hit と get または del のカウントを比較すると、memcached クライアントがサーバーに存在しないキーを数多く使用しているかどうかを特定できます。これらの情報はキーごとに記録されるため、失敗や操作が特定のキーに集中しているかどうかも特定できます。

16.6.4.6 memcached-tool の使用

memcached-tool は、memcached ソースディレクトリの scripts ディレクトリ内にあります。このツールを使用すると、memcached インスタンスから一部のレポートおよび統計に簡単にアクセスできます。

このコマンドの基本形式は次のとおりです。

```
shell> ./memcached-tool hostname:port [command]
```

デフォルトの出力では、スラブの割り当てと使用状況のリストが生成されます。例:

```
shell> memcached-tool localhost:11211 display
 #  Item_Size  Max_age  Pages  Count  Full?  Evicted Evict_Time OOM
 1     80B       93s      1     20     no       0       0    0
```

```
 2    104B      93s      1     16     no       0       0    0
 3    136B     1335s     1     28     no       0       0    0
 4    176B     1335s     1     24     no       0       0    0
 5    224B     1335s     1     32     no       0       0    0
 6    280B     1335s     1     34     no       0       0    0
 7    352B     1335s     1     36     no       0       0    0
 8    440B     1335s     1     46     no       0       0    0
 9    552B     1335s     1     58     no       0       0    0
10    696B     1335s     1     66     no       0       0    0
11    872B     1335s     1     89     no       0       0    0
12    1.1K     1335s     1    112     no       0       0    0
13    1.3K     1335s     1    145     no       0       0    0
14    1.7K     1335s     1    123     no       0       0    0
15    2.1K     1335s     1    198     no       0       0    0
16    2.6K     1335s     1    199     no       0       0    0
17    3.3K     1335s     1    229     no       0       0    0
18    4.1K     1335s     1    248    yes      36       2    0
19    5.2K     1335s     2    328     no       0       0    0
20    6.4K     1335s     2    316    yes     387       1    0
21    8.1K     1335s     3    381    yes     492       1    0
22   10.1K     1335s     3    303    yes     598       2    0
23   12.6K     1335s     5    405    yes     605       1    0
24   15.8K     1335s     6    384    yes     766       2    0
25   19.7K     1335s     7    357    yes     908     170    0
26   24.6K     1336s     7    287    yes    1012       1    0
27   30.8K     1336s     7    231    yes    1193     169    0
28   38.5K     1336s     4    104    yes    1323     169    0
29   48.1K     1336s     1     21    yes    1287       1    0
30   60.2K     1336s     1     17    yes    1093     169    0
31   75.2K     1337s     1     13    yes     713     168    0
32   94.0K     1337s     1     10    yes     278     168    0
33  117.5K     1336s     1      3     no       0       0    0
```

この出力は、command に display を指定した場合と同じです。

```
shell> memcached-tool localhost:11211 display
  #  Item_Size  Max_age   Pages   Count   Full?  Evicted Evict_Time OOM
  1     80B       93s      1      20      no       0       0    0
  2     104B      93s      1      16      no       0       0    0
...
```

この出力には、slabs 統計の出力を要約したものが表示されます。出力に表示されるカラムを次に示します。

- #: スラブ番号
- Item_Size: スラブのサイズ
- Max_age: スラブ内のもっとも古い項目の存続期間
- Pages: スラブに割り当てられたページの数
- Count: このスラブ内の項目の数
- Full?: スラブが完全に移入されているかどうか
- Evicted: このスラブから削除されたオブジェクトの数
- Evict_Time: 最後の削除から経過した時間 (秒単位)
- OOM: メモリー不足エラーが発生した項目の数

stats コマンドを使用してサーバーの一般統計のダンプを取得することもできます。

```
shell> memcached-tool localhost:11211 stats
#localhost:11211   Field       Value
        accepting_conns           1
                  bytes         162
            bytes_read         485
         bytes_written        6820
             cas_badval           0
              cas_hits           0
            cas_misses           0
             cmd_flush           0
               cmd_get           4
               cmd_set           2
           conn_yields           0
  connection_structures          11
```

```
     curr_connections         10
          curr_items          2
           decr_hits          0
         decr_misses          1
         delete_hits          0
       delete_misses          0
          evictions           0
           get_hits           4
         get_misses           0
          incr_hits           0
        incr_misses           2
     limit_maxbytes     67108864
listen_disabled_num           0
                pid       12981
      pointer_size          32
     rusage_system     0.013911
       rusage_user     0.011876
            threads           4
               time  1255518565
  total_connections          20
        total_items           2
             uptime         880
            version       1.4.2
```

16.6.5 memcached の FAQ

16.6.5.1 memcached は Windows 環境で実行できますか。 1853
16.6.5.2 memcached に格納できるオブジェクトの最大サイズはどれくらいですか。それは構成可能ですか。 1853
16.6.5.3 memcached はデータベースへの書き込みの多いアプリケーションよりデータベースの読み取りの多いアプリケーションでより効果的であるというのは本当ですか。 1854
16.6.5.4 永続的な接続を使用しないことのオーバーヘッドはありますか。永続的な接続が常に推奨される場合、そのデメリットは何ですか (たとえば、ロックの問題)。 1854
16.6.5.5 memcached クライアントによって操作されている memcached サーバーのいずれかがクラッシュした場合はどうなりますか。 1854
16.6.5.6 memcached サーバーの推奨されるハードウェア構成はどのようなものですか。 1854
16.6.5.7 memcached は、テキストの読み取り/書き込みよりもビデオおよび音声に効果がありますか。 1854
16.6.5.8 memcached は ASPX で動作しますか。 1854
16.6.5.9 memcache の接続を確立するにはどのくらいコストがかかりますか。それらの接続はプールするべきですか。 1854
16.6.5.10 memcached サーバーが停止した場合、データはどのように処理されますか。 1854
16.6.5.11 MySQL データベースの自動インクリメントカラムは、複数の memcached インスタンスでどのように調整されますか。 1855
16.6.5.12 圧縮を使用できますか。 1855
16.6.5.13 異なるタイプの memcached を同じサーバーで別々のノードとして実装し、同じサーバーで決定性のあるものと決定性のないものを持つことはできますか。 1855
16.6.5.14 パフォーマンスが向上することを確認するため、および memcached への構成変更の影響を判断するために、実装をテストするベストプラクティスはどのようなものですか。開始時の構成を単純にすることを推奨しますか。 1855

16.6.5.1 memcached は Windows 環境で実行できますか。

いいえ。現在、memcached は Unix/Linux プラットフォームでのみ使用できます。使用可能な非公式の移植版については、http://www.codeplex.com/memcachedproviders を参照してください。

16.6.5.2 memcached に格納できるオブジェクトの最大サイズはどれくらいですか。それは構成可能ですか。

デフォルトの最大オブジェクトサイズは 1M バイトです。memcached 1.4.2 以降では、-I コマンド行オプションを使用してオブジェクトの最大サイズを変更できます。

これより前のバージョンの場合、このサイズを増やすには memcached を再コンパイルする必要があります。ソース内の slabs.c ファイルの POWER_BLOCK の値を変更できます。

memcached 1.4.2 以降では、-I コマンド行オプションを使用して、サポートされる最大のオブジェクトサイズを構成できます。たとえば、最大のオブジェクトサイズを 5M バイトに増やすには、次のように実行します。

```
$ memcached -I 5m
```

オブジェクトが最大のオブジェクトサイズより大きい場合は、それを手動で分割する必要があります。memcached は非常に簡単であり、キーとデータを渡すと、それを RAM にキャッシュすることが試み

られます。デフォルトの最大サイズを超えて格納しようとすると、速度を保つために値が切り捨てられます。

16.6.5.3 memcached はデータベースへの書き込みの多いアプリケーションよりデータベースの読み取りの多いアプリケーションでより効果的であるというのは本当ですか。

はい。memcached は、データベースへの書き込みに関与することはなく、データベースからすでに読み取ったデータを RAM にキャッシュする方法です。

16.6.5.4 永続的な接続を使用しないことのオーバーヘッドはありますか。永続的な接続が常に推奨される場合、そのデメリットは何ですか (たとえば、ロックの問題)。

memcached と通信するときに永続的な接続を使用しない場合、毎回接続を開くときに待機時間が少し長くなります。その影響は、MySQL に永続的ではない接続を使用する場合と同等です。

一般に、memcached 内でロックが使用されることは非常に少ないため、永続的な接続でロックまたはその他の問題が発生する可能性はごくわずかです。問題がある場合は、要求がタイムアウトして結果が返されないため、アプリケーションが MySQL からふたたびロードする必要があります。

16.6.5.5 memcached クライアントによって操作されている memcached サーバーのいずれかがクラッシュした場合はどうなりますか。

これは自動的に処理されません。クライアントがサーバーから応答を取得できない場合、MySQL データベースからデータをロードするためのフォールバックメカニズムをコーディングしてください。

すべてのクライアント API は、memcached インスタンスを実行中に追加および削除する機能を提供しています。アプリケーションで memcached サーバーが応答しなくなったことが認識された場合は、そのサーバーをサーバーのリストから削除すると、キーがリスト内の別の memcached サーバーに自動的に再配分されます。すべてのサーバーでキャッシュの内容を維持することが重要である場合は、一貫性のあるハッシュアルゴリズムをサポートする API を使用してください。詳細は、セクション 16.6.2.4「memcached のハッシュ化/分布タイプ」を参照してください。

16.6.5.6 memcached サーバーの推奨されるハードウェア構成はどのようなものですか。

memcached の処理オーバーヘッドはごくわずかです。必要なのは余剰の物理 RAM 容量のみです。memcached サーバーには専用のマシンは必要ありません。余剰の RAM 容量がある Web サーバー、アプリケーションサーバー、またはデータベースサーバーがある場合は、それらを memcached に使用します。

専用の memcached サーバーを構築および配備する場合は、比較的性能の低い CPU、大量の RAM、および 1 つ以上のギガビット Ethernet インタフェースを使用します。

16.6.5.7 memcached は、テキストの読み取り/書き込みよりもビデオおよび音声に効果がありますか。

memcached はすべての種類のデータに対して同様に動作します。memcached を実行した場合、格納される値はデータのストリームです。ただし、memcached に格納できるオブジェクトの最大サイズは 1M バイトですが、memcached 1.4.2 以降では -I オプションを使用することによって、またはバージョン 1.4.2 より前はソースを変更することによって、それより大きく構成できます。音声およびビデオのコンテンツに memcached を使用することを計画している場合は、最大のオブジェクトサイズを増やすことがほとんどです。また、memcached は読み取りのために情報をキャッシュするソリューションです。キャッシュの情報を更新するときを除いて、書き込みに使用しないでください。

16.6.5.8 memcached は ASPX で動作しますか。

多数の言語および環境のための移植版およびインタフェースがあります。ASPX はベースとなる言語 (C#、VisualBasic など) に依存しています。ASP.NET を使用している場合は、C# の memcached ライブラリがあります。詳細は、https://sourceforge.net/projects/memcacheddotnet/ を参照してください。

16.6.5.9 memcache の接続を確立するにはどのくらいコストがかかりますか。それらの接続はプールするべきですか。

リクエストの送信および結果の取得を開始する前に、セキュリティー、認証、またはその他のハンドシェイクは行われないため、接続のオープンには比較的コストはかかりません。ほとんどの API は、待機時間を減らすために、memcached インスタンスへの永続的な接続をサポートしています。接続プールは、使用している API によって異なりますが、TCP/IP を介して直接通信している場合は、接続プールのパフォーマンスが若干よくなります。

16.6.5.10 memcached サーバーが停止した場合、データはどのように処理されますか。

その動作はアプリケーションによってまったく異なります。ほとんどのアプリケーションは、データベースからデータをロードすることにフォールバックします (memcached の情報を更新するときのように)。複数の memcached サーバーを使用している場合は、停止したサーバーをリストから削除して、パフォーマンスに影響しないようにできます。そうしないと、クライアントはロードしようとしているキーと対応している memcached サーバーと通信することを試みます。

16.6.5.13. MySQL データベースの自動インクリメントカラムは、複数の memcached インスタンスでどのように調整されますか。

調整されません。MySQL と memcached に関連はありません (アプリケーションにそのような関連を作成した場合 (または、memcached およびデータベース定義に MySQL UDF を使用している場合) を除きます)。

memcached の複数インスタンスに自動インクリメントキーに基づいて情報を格納している場合、情報はいずれかの memcached インスタンスに格納されるのみです。クライアントは、キー値を使用して、情報が格納されている memcached インスタンスを判別します。同じ情報は、キャッシュメモリーが無駄になるため、すべてのインスタンスに格納されません。

16.6.5.14. 圧縮を使用できますか。

はい。ほとんどのクライアント API は何らかの圧縮をサポートしており、格納中に値が圧縮に適しているかどうかを判断するしきい値を指定できるものもあります。

16.6.5.15. 異なるタイプの memcached を同じサーバーで別々のノードとして実装し、同じサーバーで決定性のあるものと決定性のないものを持つことはできますか。

はい。単一のサーバー上で memcached の複数のインスタンスを実行し、使用するサーバーのリストをクライアントの構成で選択できます。

16.6.5.16. パフォーマンスが向上することを確認するため、および memcached への構成変更の影響を判断するために、実装をテストするベストプラクティスはどのようなものですか。開始時の構成を単純にすることを推奨しますか。

パフォーマンスをテストする最適な方法は、memcached インスタンスを起動する方法です。最初に、データが使用または表示される直前にデータを memcached に格納するようにアプリケーションを変更します。API によってデータが直列化されるため、コードを 1 行変更すればよいだけです。そして、通常は MySQL から情報をロードするプロセスの開始を、memcached にデータをリクエストするコードに変更します。memcached からデータをロードできない場合は、デフォルトで MySQL プロセスからロードされます。

変更する必要があるのはおそらく数行のコードのみです。利点を最大に活用するには、memcached を MySQL テーブルの個別の行の単純なキャッシュとして使用するのではなく、オブジェクト全体 (たとえば、Web ページ、ブログの投稿、ディスカッションスレッドなどのすべてのコンポーネント) をキャッシュしてください。

memcached では、開始時の構成を単純にすること、または長期間にわたってそれを維持することは簡単です。基本的な構造を作成して実行したら、多くの場合、使用中に変更することは、アプリケーションによって使用されるサーバーのリストにサーバーを追加することのみです。memcached サーバーを管理する必要はなく、複雑な構成はありません。リストにサーバーを追加したら、クライアント API および memcached サーバーに判断を任せます。

第 17 章 レプリケーション

目次

17.1 レプリケーション構成 1858
17.1.1 レプリケーションのセットアップ方法 1859
17.1.2 レプリケーション形式 1867
17.1.3 グローバルトランザクション識別子を使用したレプリケーション 1873
17.1.4 レプリケーションおよびバイナリロギングのオプションと変数 1879
17.1.5 一般的なレプリケーション管理タスク 1949
17.2 レプリケーションの実装 1951
17.2.1 レプリケーション実装の詳細 1952
17.2.2 レプリケーションリレーおよびステータスログ 1953
17.2.3 サーバーがレプリケーションフィルタリングルールをどのように評価するか 1958
17.3 レプリケーションソリューション 1964
17.3.1 バックアップ用にレプリケーションを使用する 1964
17.3.2 異なるマスターおよびスレーブストレージエンジンでレプリケーションを使用する 1968
17.3.3 スケールアウトのためにレプリケーションを使用する 1969
17.3.4 異なるデータベースを異なるスレーブに複製する 1970
17.3.5 レプリケーションパフォーマンスを改善する 1971
17.3.6 フェイルオーバー中にマスターを切り替える 1971
17.3.7 SSL を使用してレプリケーションをセットアップする 1973
17.3.8 準同期レプリケーション 1975
17.3.9 遅延レプリケーション 1978
17.4 レプリケーションの注釈とヒント 1979
17.4.1 レプリケーションの機能と問題 1979
17.4.2 MySQL バージョン間のレプリケーション互換性 2001
17.4.3 レプリケーションセットアップをアップグレードする 2002
17.4.4 レプリケーションのトラブルシューティング 2003
17.4.5 レプリケーションバグまたは問題を報告する方法 2004

レプリケーションによって、1 つの MySQL データベースサーバー (マスター) のデータを、1 つまたは複数の MySQL データベースサーバー (スレーブ) に複製できます。レプリケーションはデフォルトで非同期であるため、スレーブはマスターから更新を受け取るために永続的に接続されている必要はありません。これは、長距離間接続でも、さらにダイアルアップサービスのような一時的または断続的接続上でも更新が可能であることを意味しています。構成に応じて、すべてのデータベース、選択したデータベース、さらにデータベース内の選択したテーブルを複製できます。

MySQL Replication をはじめて使用する人がよくする質問の回答については、セクションA.13「MySQL 5.6 FAQ: レプリケーション」を参照してください。

MySQL のレプリケーションの長所は次のとおりです。

- スケールアウトソリューション - パフォーマンスの向上のために複数のスレーブに負荷を分散します。この環境では、すべての書き込みと更新をマスターサーバーで実行する必要があります。ただし、読み取りの場合は、1 つ以上のスレーブで実行してもかまいません。このモデルでは、書き込みのパフォーマンスを向上させながら (マスターが更新専用であるため)、スレーブ数が増加しても読み取り速度を劇的に速めることができます。
- データセキュリティー - データはスレーブに複製され、スレーブはレプリケーションプロセスを一時停止できるため、対応するマスターデータを壊すことなくスレーブでバックアップサービスを実行できます。
- 分析 - マスターでライブデータを作成しながら、スレーブで情報の分析を実行できるため、マスターのパフォーマンスに影響しません。
- 長距離データ配布 - 支店でメインデータのコピーを使用して作業する場合に、レプリケーションを使用してデータのローカルコピーを作成してそれらを使用できます (マスターへの永続的なアクセスは不要)。

MySQL のレプリケーションの特徴は、一方向の非同期レプリケーションをサポートしていることであり、1 つのサーバーがマスターとして機能し、1 つまたは複数のサーバーがスレーブとして機能します。これは、MySQL クラスタの特徴である同期レプリケーションとは対照的です (第18章「MySQL Cluster NDB 7.3 および MySQL Cluster NDB 7.4」を参照してください)。MySQL 5.6 では、組み込みの非同期レプリケーションに加えて、準同期レプリケーションへのインタフェースがサポートされています。準同期レプリケーションでは、マスター側

で実行されたコミットは、トランザクションを実行したセッションに戻る前に、少なくとも 1 つのスレーブがトランザクションのイベントを受け取ってログに記録したことを通知するまでブロックされます。セクション17.3.8「準同期レプリケーション」を参照してください。MySQL 5.6 は、スレーブサーバーが、少なくとも指定した時間だけ意図的にマスターより遅れるような遅延レプリケーションもサポートしています。セクション17.3.9「遅延レプリケーション」を参照してください。

2 つのサーバー間でレプリケーションをセットアップするために使用できるソリューションはいくつかありますが、最善の使用方法はデータの存在や使用しているエンジンタイプによって異なります。利用可能なオプションの詳細については、セクション17.1.1「レプリケーションのセットアップ方法」を参照してください。

レプリケーション形式の主要なタイプは 2 つあり、1 つは、SQL ステートメント全体を複製する Statement Based Replication (SBR: ステートメントベースレプリケーション)、もう 1 つは変更があった行だけを複製する Row Based Replication (RBR: 行ベースレプリケーション) です。3 つ目の Mixed Based Replication (MBR: ミックスベースレプリケーション) を使用することもできます。さまざまなレプリケーション形式の詳細については、セクション17.1.2「レプリケーション形式」を参照してください。MySQL 5.6 では、ステートメントベースの形式がデフォルトです。

MySQL 5.6.5 以降では、グローバルトランザクション ID (GTID) に基づくトランザクションレプリケーションをサポートしています。このタイプのレプリケーションを使用するときは、ログファイルまたはこれらのファイル内の位置と直接連携する必要がないため、共通する多くのレプリケーションタスクが大幅に単純化されます。GTID を使用するレプリケーションは完全にトランザクション対応であるため、マスターで確定されたすべてのトランザクションがスレーブでも適用されるかぎり、マスターとスレーブとの一貫性は保証されます。GTID および GTID ベースのレプリケーションの詳細については、セクション17.1.3「グローバルトランザクション識別子を使用したレプリケーション」を参照してください。

レプリケーションは、いくつかのオプションと変数によって制御されます。これらは、レプリケーション、タイムアウト、データベース、およびデータベースとテーブルに適用できるフィルターのコア操作を制御します。利用可能なオプションの詳細については、セクション17.1.4「レプリケーションおよびバイナリロギングのオプションと変数」を参照してください。

レプリケーションを使用することで、いくつかの異なる問題を解決できます (パフォーマンスの問題、異なるデータベースのバックアップのサポート、より大きなソリューションの一部としてシステム障害を軽減する、など)。これらの問題の対処方法については、セクション17.3「レプリケーションソリューション」を参照してください。

レプリケーション機能の詳細、バージョン互換性、アップグレード、および問題とその解決 (FAQ を含む) など、さまざまなデータ型とステートメントがレプリケーション中にどのように処理されるかに関する説明とヒントについては、セクション17.4「レプリケーションの注釈とヒント」を参照してください。

レプリケーションの実装、レプリケーションの仕組み、バイナリログのプロセスと内容、バックアップスレッド、およびステートメントの記録とレプリケーションの方法を決めるために使用されるルールの詳細については、セクション17.2「レプリケーションの実装」を参照してください。

17.1 レプリケーション構成

MySQL 内のサーバー間のレプリケーションはバイナリロギングのメカニズムに基づいています。マスター (データベースの変更元) として動作する MySQL インスタンスは、更新および変更を「イベント」としてバイナリログに書き込みます。バイナリログ内の情報は、記録されているデータベース変更に応じて異なるロギング形式で格納されます。スレーブは、マスターからバイナリログを読み取って、スレーブのローカルデータベースにあるバイナリログのイベントを実行するように構成されます。

> **重要**
>
> 特定のイベントのログのみを記録するようにマスターを構成することはできません。

マスターはこのシナリオでは「「ダム」」です。いったんバイナリロギングが有効になると、すべてのステートメントはバイナリログに記録されます。各スレーブはバイナリログの内容全体のコピーを受け取ります。バイナリログのどのステートメントを実行するべきかを決めるのはスレーブの役割で、特定のイベントのログのみを記録するようにマスターを構成することはできません。特に指定しない場合は、マスターバイナリログのすべてのイベントがスレーブで実行されます。必要な場合、特定のデータベースまたはテーブルに適用するイベントだけを処理するようにスレーブを構成できます。

各スレーブはバイナリログ座標 (マスターから読み取って処理したファイル名とファイル内での位置) のレコードを保持します。これは、複数のスレーブがマスターに接続して同じバイナリログの別の部分を実行できることを意味します。スレーブはこのプロセスを制御するため、マスターの操作に影響を与えることなく、サーバーから個々のスレーブに接続したり接続解除したりできます。また、各スレーブはバイナリログ内の現在の位置を記録しているため、スレーブを接続解除し、再接続してから処理を再開できます。

マスターと各スレーブは一意 ID を使用して構成する必要があります (server-id オプションを使用)。また、各スレーブは、マスターホスト名、ログファイル名、およびそのファイル内の位置に関する情報を使用して構成する必要があります。これらの詳細は、スレーブで CHANGE MASTER TO ステートメントを使用して MySQL セッション内から制御できます。詳細はスレーブのマスター情報リポジトリ (ファイルまたはテーブル) に格納されます (セクション17.2.2「レプリケーションリレーおよびステータスログ」を参照してください)。

このセクションでは、レプリケーション環境に必要なセットアップと構成について説明します。新しいレプリケーション環境を作成するための手順を追った説明を含みます。このセクションの主な内容は次のとおりです。

- レプリケーション用に 2 つ以上のサーバーをセットアップするためのガイドについては、セクション17.1.1「レプリケーションのセットアップ方法」で、システムの構成について扱い、マスターとスレーブ間でデータをコピーする方法について説明します。
- バイナリログ内のイベントはいくつかの形式で記録されます。これらは、ステートメントベースレプリケーション (SBR) または行ベースレプリケーション (RBR) と呼ばれます。3 つ目のタイプ、混合形式レプリケーション (MIXED) は、SBR または RBR レプリケーションを自動的に使用し、必要に応じて SBR と RBR の両方の形式の利点を活用します。さまざまな形式については、セクション17.1.2「レプリケーション形式」を参照してください。
- レプリケーションに適用するさまざまな構成のオプションと変数に関する詳細は、セクション17.1.4「レプリケーションおよびバイナリロギングのオプションと変数」を参照してください。
- レプリケーションプロセスが開始されると、管理または監視はほとんど必要ありません。ただし、実行することが望ましい一般的なタスクに関するアドバイスについては、セクション17.1.5「一般的なレプリケーション管理タスク」を参照してください。

17.1.1 レプリケーションのセットアップ方法

このセクションでは、MySQL サーバーの完全レプリケーションのセットアップ方法について説明します。レプリケーションのセットアップにはいくつかの異なる方法があり、使用する方法はレプリケーションをどのようにセットアップするか、およびマスターデータベース内にすでにデータがあるかどうかによって異なります。

すべてのレプリケーションセットアップに共通する汎用タスクがいくつかあります。

- マスターでは、バイナリロギングを有効にして、一意サーバー ID を構成する必要があります。これには、サーバーの再起動が必要となる場合があります。セクション17.1.1.1「レプリケーションマスター構成の設定」を参照してください。
- マスターに接続する各スレーブでは、一意サーバー ID を構成する必要があります。これには、サーバーの再起動が必要となる場合があります。セクション17.1.1.2「レプリケーションスレーブ構成の設定」を参照してください。
- 必要に応じて、スレーブ用に別のユーザーを作成し、レプリケーションのためにバイナリログを読み取る際のマスターでの認証に使用します。セクション17.1.1.3「レプリケーション用ユーザーの作成」を参照してください。
- データスナップショットを作成する前、またはレプリケーションプロセスを開始する前に、マスター上のバイナリログの位置を記録してください。この情報はスレーブを構成するときに必要になり、これによりスレーブはバイナリログ内のどこからイベントの実行を開始するかがわかります。セクション17.1.1.4「レプリケーションマスターバイナリログ座標の取得」を参照してください。
- マスター上にすでにデータがあり、それを使用してスレーブと同期する場合は、データスナップショットを作成する必要があります。データベーススナップショットを作成するには、データベースのサイズとファイルの位置によって、さまざまな方法があります。 スナップショットは、mysqldump (セクション17.1.1.5「mysqldump を使用したデータスナップショットの作成」を参照してください) を使用するかデータファイルを直接コピーすることで (セクション17.1.1.6「ローデータファイルを使用したデータスナップショットの作成」を参照してください) 作成します。
- ホスト名、ログイン資格証明、バイナリログファイルの名前と位置など、マスターに接続するための設定を使用してスレーブを構成します。セクション17.1.1.10「スレーブでのマスター構成の設定」を参照してください。

基本オプションを構成したあとは、次のシナリオを選択します。

- データを含まないマスターとスレーブのフレッシュインストールでレプリケーションをセットアップするには、セクション17.1.1.7「新しいマスターとスレーブを使用したレプリケーションのセットアップ」を参照してください。

- 既存の MySQL サーバーのデータを使用して新しいマスターのレプリケーションをセットアップするには、セクション17.1.1.8「既存のデータによるレプリケーションのセットアップ」を参照してください。
- 既存のレプリケーション環境にレプリケーションのスレーブを追加するには、セクション17.1.1.9「既存のレプリケーション環境への追加スレーブの導入」を参照してください。

MySQL レプリケーションサーバーを管理する前に、この章全体を読み、セクション13.4.1「マスターサーバーを制御するための SQL ステートメント」とセクション13.4.2「スレーブサーバーを制御するための SQL ステートメント」で説明したすべてのステートメントを試みてください。また、セクション17.1.4「レプリケーションおよびバイナリロギングのオプションと変数」で説明されたレプリケーションの起動オプションについても習得してください。

> 注記
>
> セットアッププロセスのあるステップでは、SUPER 権限が必要です。この権限がないと、レプリケーションを有効にできない可能性があります。

17.1.1.1 レプリケーションマスター構成の設定

レプリケーションマスターでは、バイナリロギングを有効にして、一意サーバー ID を作成する必要があります。これがまだ行われていなかった場合、サーバーの再起動が必要です。

バイナリログがマスターからスレーブへ変更を複製するための基盤であるため、バイナリロギングがマスターで有効である必要があります。バイナリロギングが log-bin オプションを使用して有効化されていないと、レプリケーションはできません。

レプリケーショングループ内の各サーバーは、一意サーバー ID を使用して構成する必要があります。この ID はグループ内の個々のサーバーを特定するために使用され、1 から (2^{32}) - 1 の間の正の整数でなければいけません。番号をどのように編成および選択するかは任意です。

バイナリログとサーバー ID のオプションを構成するには、MySQL サーバーをシャットダウンして、my.cnf または my.ini ファイルを編集します。構成ファイルの [mysqld] セクションに、log-bin および server-id オプションを追加します。これらのオプションはすでに存在しているが、コメントアウトされている場合、そのオプションのコメントを解除して必要に応じて変更します。たとえば、mysql-bin のログファイル名プリフィクスを使用してバイナリロギングを有効にし、ID が 1 のサーバーを構成するには、次の行を使用します。

```
[mysqld]
log-bin=mysql-bin
server-id=1
```

変更したあとに、サーバーを再起動します。

> 注記
>
> server-id を省略すると (または明示的にデフォルト値の 0 に設定すると)、マスターはスレーブからのどのような接続も拒否します。

> 注記
>
> トランザクション対応 InnoDB を使用するレプリケーションセットアップで持続性と一貫性をできるだけ高くするため、my.cnf ファイルで innodb_flush_log_at_trx_commit=1 および sync_binlog=1 を使用してください。

> 注記
>
> skip-networking オプションがレプリケーションマスターで有効になっていないことを確認します。ネットワーキングが無効化されていると、スレーブはマスターと通信できず、レプリケーションは失敗します。

17.1.1.2 レプリケーションスレーブ構成の設定

レプリケーションスレーブでは、一意サーバー ID を作成する必要があります。これがまだ行われていなかった場合、この部分のスレーブセットアップにはサーバーの再起動が必要です。

スレーブサーバー ID がまだ設定されていない場合、または現在の値がマスターサーバーに選択した値と競合する場合は、スレーブサーバーをシャットダウンし、構成ファイルの [mysqld] セクションを編集して、一意サーバー ID を指定します。例:

```
[mysqld]
server-id=2
```

変更したあとに、サーバーを再起動します。

複数のスレーブをセットアップしている場合、それぞれのスレーブでは、一意 server-id 値がマスターのものおよびほかのすべてのスレーブのものと異なる必要があります。

> 注記
>
> server-id を省略すると (または明示的にデフォルト値の 0 に設定すると)、スレーブはマスターへの接続を拒否します。

レプリケーションをセットアップするために、スレーブでバイナリロギングを有効にする必要はありません。ただし、スレーブでバイナリロギングを有効にすると、データバックアップとクラッシュリカバリにスレーブのバイナリログを使用でき、より複雑なレプリケーショントポロジの一部としてスレーブを使用することもできます。たとえばこの場合、このスレーブはほかのスレーブへのマスターとして動作します。

17.1.1.3 レプリケーション用ユーザーの作成

それぞれのスレーブは MySQL のユーザー名とパスワードを使用してマスターに接続するため、スレーブが接続に使用できるユーザーアカウントがマスター上になければいけません。この操作には、REPLICATION SLAVE 権限が付与されているすべてのアカウントを使用できます。スレーブごとに異なるアカウントを作成したり、各スレーブに同じアカウントを使用してマスターに接続したりすることを選択できます。

レプリケーション用に特別にアカウントを作成する必要はありませんが、レプリケーションのユーザー名とパスワードはマスターの情報リポジトリファイルまたはテーブルにプレーンテキストで格納されていることを確認してください (セクション17.2.2.2「スレーブステータスログ」を参照してください)。このため、ほかのアカウントのセキュリティーを損なう可能性を最小限に抑えるため、レプリケーションプロセスにのみ権限を持つ別のアカウントを作成することをお勧めします。

新しいアカウントを作成するには、CREATE USER を使用します。レプリケーションに必要な権限をこのアカウントに付与するには、GRANT ステートメントを使用します。レプリケーションの目的にだけアカウントを作成する場合、そのアカウントには REPLICATION SLAVE 権限だけが必要です。たとえば、mydomain.com ドメイン内のホストからレプリケーションのために接続できる新しいユーザー repl をセットアップするには、マスター上で次のステートメントを発行します。

```
mysql> CREATE USER 'repl'@'%.mydomain.com' IDENTIFIED BY 'slavepass';
mysql> GRANT REPLICATION SLAVE ON *.* TO 'repl'@'%.mydomain.com';
```

ユーザーアカウントを操作するためのステートメントの詳細については、セクション13.7.1「アカウント管理ステートメント」を参照してください。

17.1.1.4 レプリケーションマスターバイナリログ座標の取得

マスターバイナリログ内のマスターの現在座標を注目してください。この情報は、現在のポイントでレプリケーションプロセスを起動するようにスレーブを構成するために必要です。

レプリケーションプロセスを起動する前に、スレーブ上で同期する既存のデータがマスター上にある場合、マスターがステートメントの実行を継続するのを許可する前に、マスターでのステートメントの処理を停止してから、現在のバイナリログ座標を取得してそのデータをダンプする必要があります。ステートメントの実行を停止しない場合は、使用するデータダンプとマスターステータス情報が一致せず、スレーブ上でデータベースが一貫性を失ったり破損したりします。

マスターバイナリログ座標を取得するには、次の手順に従ってください。

1. コマンド行クライアントでマスターに接続することでマスター上でセッションを起動し、FLUSH TABLES WITH READ LOCK ステートメントを実行することですべてのテーブルをフラッシュして書き込みステートメントをブロックします。

   ```
   mysql> FLUSH TABLES WITH READ LOCK;
   ```

   InnoDB テーブルの場合、FLUSH TABLES WITH READ LOCK は COMMIT 操作もブロックします。

   > 警告
   >
   > 読み取りロックを有効のままにするため、FLUSH TABLES ステートメントを発行したクライアントを実行中のままにしてください。クライアントを終了すると、ロックは解除されます。

2. マスターの別のセッションで、SHOW MASTER STATUS ステートメントを使用して現在のバイナリログファイルの名前と位置を調べます。

```
mysql > SHOW MASTER STATUS;
+------------------+----------+--------------+------------------+
| File             | Position | Binlog_Do_DB | Binlog_Ignore_DB |
+------------------+----------+--------------+------------------+
| mysql-bin.000003 | 73       | test         | manual,mysql     |
+------------------+----------+--------------+------------------+
```

File カラムはログファイルの名前を表し、Position はファイル内の位置を表します。この例では、バイナリログファイルは mysql-bin.000003 で、位置は 73 です。これらの値を記録します。これらは、あとでスレーブをセットアップするときに必要です。それらは、スレーブがマスターからの新しい更新の処理を始めるべきレプリケーション座標を表します。

バイナリロギングを有効にしないでマスターがすでに動作していた場合、SHOW MASTER STATUS または mysqldump --master-data が表示するログファイルの名前と位置の値は空になります。この場合、あとでスレーブのログファイルと位置を指定するときに使用する必要がある値は、空の文字列 ('') と 4 です。

これで、スレーブがレプリケーションを開始する適切な位置でバイナリログから読み取りを開始するために必要な情報を入手しました。

レプリケーションを開始する前にスレーブと同期する必要がある既存のデータがある場合は、ロックが適切に維持されるようにクライアントを実行したままにしてから、セクション17.1.1.5「mysqldump を使用したデータスナップショットの作成」またはセクション17.1.1.6「ローデータファイルを使用したデータスナップショットの作成」に進んでください。ここでの意図は、スレーブにコピーされたデータがマスターと同期された状態であるように、これ以上の変更を避けることです。

まったく新しいマスターとスレーブのレプリケーショングループをセットアップしている場合、最初のセッションを終了して読み取りロックを解除できます。

17.1.1.5 mysqldump を使用したデータスナップショットの作成

既存のマスターデータベースでデータのスナップショットを作成する 1 つの方法は、mysqldump ツールを使用して、複製するすべてのデータベースのダンプを作成することです。データダンプが完了したら、レプリケーションプロセスを開始する前に、このデータをスレーブにインポートします。

ここで示した例では、dbdump.db という名前のファイルにすべてのデータベースをダンプし、レプリケーションプロセスを開始するためにスレーブ側で必要な CHANGE MASTER TO ステートメントを自動的に加える --master-data オプションを追加しています。

```
shell> mysqldump --all-databases --master-data > dbdump.db
```

--master-data を使用しない場合は、個々のセッションですべてのテーブルを手動でロックしてから (FLUSH TABLES WITH READ LOCK を使用)、mysqldump を実行し、2 番目のセッションから終了するか UNLOCK TABLES を実行するかしてロックを解除する必要があります。また、SHOW MASTER STATUS を使用してスナップショットに一致するバイナリログ位置情報を取得し、これを使用してスレーブ起動時に対応する CHANGE MASTER TO ステートメントを発行する必要があります。

ダンプに含めるデータベースを選択するときに、各スレーブでレプリケーションプロセスに含めないデータベースを除外する必要があることを覚えておいてください。

データをインポートするには、ダンプファイルをスレーブにコピーするか、スレーブにリモートで接続したときにマスターからファイルにアクセスします。

17.1.1.6 ローデータファイルを使用したデータスナップショットの作成

データベースが大きい場合、ローデータファイルをコピーする方が、mysqldump を使用して各スレーブでファイルをインポートするよりも、効率が高くなる場合があります。この方法では、INSERT ステートメントが繰り返されたときのインデックス更新のオーバーヘッドが省かれます。

複雑なキャッシュまたはロギングアルゴリズムを持つストレージエンジンのテーブルでこの方法を使用すると、完全な「ポイントインタイム」スナップショットを作成するための追加手順が必要となります。グローバル読み取りロックを獲得した場合、最初のコピーコマンドはキャッシュ情報とロギング更新が除外する場合があります。ストレージエンジンがこれにどのように反応するかは、そのクラッシュリカバリ能力によります。

マスターとスレーブで ft_stopword_file、ft_min_word_len、または ft_max_word_len の値が異なり、全文インデックスを持つテーブルをコピーする場合も、この方法が確実に機能するとはかぎりません。

InnoDB テーブルを使用する場合、MySQL Enterprise Backup コンポーネントから mysqlbackup コマンドを使用して、一貫性のあるスナップショットを作成できます。このコマンドは、あとでスレーブで使用するために、スナップショットに対応するログ名とオフセットを記録します。MySQL Enterprise Backup は MySQL Enterprise サ

ブスクリプションの一部として同梱される製品です。詳細は、セクション25.2「MySQL Enterprise Backup」を参照してください。

または、コールドバックアップ技術を使用して InnoDB テーブルの信頼できるバイナリスナップショットを取得します。MySQL Server のスローシャットダウン を実行したあとにすべてのデータファイルをコピーします。

MyISAM テーブルのローデータスナップショットを作成するには、MySQL データファイルが単一ファイルシステム上に存在するという条件で、cp、copy などの標準コピーツール、scp、rsync などのリモートコピーツール、zip、tar などのアーカイブツール、dump などのファイルシステムスナップショットツールを使用できます。特定のデータベースだけを複製する場合、それらのテーブルに関係するファイルだけをコピーします。(InnoDB の場合、すべてのデータベースのすべてのテーブルは、innodb_file_per_table オプションを有効にしている場合を除いて、system tablespace ファイルに格納されます。)

次のファイルは明確にアーカイブから除外することをお勧めします。

- mysql データベースに関連するファイル。
- (使用する場合) マスター情報リポジトリファイル (セクション17.2.2「レプリケーションリレーおよびステータスログ」を参照してください)。
- マスターのバイナリログファイル。
- リレーログファイル。

ローデータスナップショットでもっとも一貫性のある結果を得るには、次のように、プロセス中にマスターサーバーをシャットダウンします。

1. 読み取りロックを獲得してマスターのステータスを取得します。セクション17.1.1.4「レプリケーションマスターバイナリログ座標の取得」を参照してください。
2. 個別のセッションで、マスターサーバーをシャットダウンします。

   ```
   shell> mysqladmin shutdown
   ```

3. MySQL データファイルのコピーを作成します。次の例では、これを行うための一般的な方法を示します。この中の 1 つだけを選択する必要があります。

   ```
   shell> tar cf /tmp/db.tar ./data
   shell> zip -r /tmp/db.zip ./data
   shell> rsync --recursive ./data /tmp/dbdata
   ```

4. マスターサーバーを再起動します。

InnoDB テーブルを使用しない場合、次の手順で説明するとおり、サーバーをシャットダウンしないでマスターからシステムのスナップショットを取得できます。

1. 読み取りロックを獲得してマスターのステータスを取得します。セクション17.1.1.4「レプリケーションマスターバイナリログ座標の取得」を参照してください。
2. MySQL データファイルのコピーを作成します。次の例では、これを行うための一般的な方法を示します。この中の 1 つだけを選択する必要があります。

   ```
   shell> tar cf /tmp/db.tar ./data
   shell> zip -r /tmp/db.zip ./data
   shell> rsync --recursive ./data /tmp/dbdata
   ```

3. 読み取りロックを獲得したクライアントでは、ロックを解除します。

   ```
   mysql> UNLOCK TABLES;
   ```

データベースのアーカイブまたはコピーを作成したあと、スレーブレプリケーションプロセスを開始する前に、各スレーブにファイルをコピーします。

17.1.1.7 新しいマスターとスレーブを使用したレプリケーションのセットアップ

レプリケーションをセットアップするもっとも簡単で容易な方法は、新しいマスターおよびスレーブサーバーを使用することです。

新しいサーバーをセットアップしているけれども、レプリケーション構成にロードする、別のサーバーからのデータベースの既存のダンプがある場合、この方法も使用できます。データを新しいマスターにロードすることで、データはスレーブに自動的に複製されます。

新しいマスターとスレーブの間でレプリケーションをセットアップするには、次のようにします。

1. 必要な構成プロパティーで MySQL マスターを構成します。セクション17.1.1.1「レプリケーションマスター構成の設定」を参照してください。
2. MySQL マスターをセットアップします。
3. ユーザーをセットアップします。セクション17.1.1.3「レプリケーション用ユーザーの作成」を参照してください。
4. マスターステータス情報を取得します。セクション17.1.1.4「レプリケーションマスターバイナリログ座標の取得」を参照してください。
5. マスターで、読み取りロックを解除します。

```
mysql> UNLOCK TABLES;
```

6. スレーブで、MySQL 構成を編集します。セクション17.1.1.2「レプリケーションスレーブ構成の設定」を参照してください。
7. MySQL スレーブを起動します。
8. CHANGE MASTER TO ステートメントを実行して、マスターレプリケーションサーバー構成を設定します。セクション17.1.1.10「スレーブでのマスター構成の設定」を参照してください。

各スレーブでスレーブセットアップ手順を実行します。

新しいサーバー構成にはロードまたは交換するデータがないため、情報をコピーしたりインポートしたりする必要はありません。

別の既存データベースサーバーのデータを使用して新しいレプリケーション環境をセットアップする場合、そのサーバーから生成されたダンプファイルを新しいマスターで実行する必要があります。データベース更新は自動的にスレーブへ伝達されます。

```
shell> mysql -h master < fulldb.dump
```

17.1.1.8 既存のデータによるレプリケーションのセットアップ

既存データでレプリケーションをセットアップする場合、レプリケーションサービスを開始する前に、マスターからのデータをスレーブに取得するための最善の方法を決める必要があります。

既存のデータでレプリケーションをセットアップするための基本プロセスは次のとおりです。

1. MySQL マスターが動作している状態で、レプリケーション中にマスターに接続するときにスレーブが使用するユーザーを作成します。セクション17.1.1.3「レプリケーション用ユーザーの作成」を参照してください。
2. まだ server-id を構成していなくて、マスターサーバーでバイナリロギングを有効にしていない場合は、これらのオプションを構成するためにサーバーをシャットダウンする必要があります。セクション17.1.1.1「レプリケーションマスター構成の設定」を参照してください。

   マスターサーバーをシャットダウンする必要がある場合は、そのデータベースのスナップショットを作成するための良い機会です。マスターを停止し、構成を更新し、スナップショットを作成する前に、マスターのステータスを取得してください (セクション17.1.1.4「レプリケーションマスターバイナリログ座標の取得」を参照してください)。ローデータファイルを使用してスナップショットを作成する方法については、セクション17.1.1.6「ローデータファイルを使用したデータスナップショットの作成」を参照してください。
3. マスターサーバーがすでに正しく構成されている場合、そのステータスを取得してから (セクション17.1.1.4「レプリケーションマスターバイナリログ座標の取得」を参照してください) mysqldump を使用してスナップショットを作成するか (セクション17.1.1.5「mysqldump を使用したデータスナップショットの作成」を参照してください)、セクション17.1.1.6「ローデータファイルを使用したデータスナップショットの作成」のガイドを使用してライブサーバーのロースナップショットを作成します。
4. スレーブの構成を更新します。セクション17.1.1.2「レプリケーションスレーブ構成の設定」を参照してください。
5. データのスナップショットをマスター上にどのように作成したかによって、次の手順は異なります。

   mysqldump を使用した場合:

   a. レプリケーションが起動しないように、--skip-slave-start オプションを使用してスレーブを起動します。

   b. ダンプファイルをインポートします。

```
shell> mysql < fulldb.dump
```

ローデータファイルでスナップショットを作成した場合:

a. スレーブデータディレクトリにデータファイルを抽出します。例:

```
shell> tar xvf dbdump.tar
```

スレーブサーバーがファイルにアクセスして変更できるように、それらへの許可と所有権を設定する必要がある場合があります。

b. レプリケーションが起動しないように、--skip-slave-start オプションを使用してスレーブを起動します。

6. マスターからのレプリケーション座標を使用してスレーブを構成します。これは、バイナリログファイルと、ファイル内でレプリケーションを開始する必要がある位置をスレーブに伝えます。また、マスターのログイン資格証明とホスト名を使用してスレーブを構成します。必要な CHANGE MASTER TO ステートメントの詳細は、セクション17.1.1.10「スレーブでのマスター構成の設定」を参照してください。

7. スレーブスレッドを起動します。

```
mysql> START SLAVE;
```

この手順を行なったあと、スレーブはマスターに接続し、スナップショット作成後に発生した更新を反映するはずです。

マスターに server-id オプションを設定するのを忘れた場合、スレーブはそれに接続できません。

スレーブに server-id オプションを設定するのを忘れた場合、スレーブのエラーログに次にようなエラーを取得します。

```
Warning: You should set server-id to a non-0 value if master_host
is set; we will force server id to 2, but this MySQL server will
not act as a slave.
```

何らかの理由で複製できない場合も、スレーブのエラーログにエラーメッセージを取得します。

スレーブはそのマスター情報リポジトリに格納された情報を使用して、マスターのバイナリログがどの程度処理されたかを追跡します。リポジトリはファイルまたはテーブルの形式で、--master-info-repository の値セットによって決定されます。スレーブが --master-info-repository=FILE で動作している場合、データディレクトリの中に、名前が master.info と relay-log.info の 2 つのファイルが見つかります。代わりに --master-info-repository=TABLE である場合、この情報は mysql データベースのテーブル master_slave_info に格納されます。いずれの場合も、何を実行しているかが正確にわかり、その影響を十分に理解している場合を除いて、ファイルまたはテーブルを削除したり編集したりしないでください。その場合でも、CHANGE MASTER TO ステートメントを使用してレプリケーションパラメータを変更することをお勧めします。スレーブはステートメントで指定した値を使用して、ステータスファイルを自動的に更新できます。詳細については、セクション17.2.2「レプリケーションリレーおよびステータスログ」を参照してください。

> **注記**
>
> マスター情報リポジトリの内容は、コマンド行または my.cnf で指定されたいくつかのサーバーオプションをオーバーライドします。詳細については、セクション17.1.4「レプリケーションおよびバイナリロギングのオプションと変数」を参照してください。

複数のスレーブでも、マスターの単一スナップショットで十分です。追加のスレーブをセットアップする場合は、同じマスタースナップショットを使用して、先述の手順のスレーブ部分に従ってください。

17.1.1.9 既存のレプリケーション環境への追加スレーブの導入

既存のレプリケーション構成に別のスレーブを追加する場合は、マスターを停止することなく実行できます。代わりに、既存のスレーブのコピーを作成することで新しいスレーブをセットアップしますが、新しいスレーブは別の server-id 値で構成します。

既存のスレーブを複製するには、次のようにします。

1. 既存のスレーブをシャットダウンします。

```
shell> mysqladmin shutdown
```

2. 既存のスレーブから新しいスレーブにデータディレクトリをコピーします。これを行うには、tar または WinZip を使用してアーカイブを作成するか、cp、rsync などのツールを使用して直接コピーを実行します。ログファイルとリレーログファイルを確実にコピーしてください。

新しいレプリケーションスレーブを追加するときに直面するよくある問題は、次のような一連の警告またはエラーメッセージで新しいスレーブが失敗することです。

```
071118 16:44:10 [Warning] Neither --relay-log nor --relay-log-index were used; so
replication may break when this MySQL server acts as a slave and has his hostname
changed!! Please use '--relay-log=new_slave_hostname-relay-bin' to avoid this problem.
071118 16:44:10 [ERROR] Failed to open the relay log './old_slave_hostname-relay-bin.003525'
(relay_log_pos 22940879)
071118 16:44:10 [ERROR] Could not find target log during relay log initialization
071118 16:44:10 [ERROR] Failed to initialize the master info structure
```

これは、--relay-log オプションが指定されていない場合、リレーログファイルにそれらのファイル名の一部としてホスト名が含まれているためです。(これは、--relay-log-index オプションが使用されていない場合のリレーログインデックスファイルにも当てはまります。これらのオプションの詳細については、セクション17.1.4「レプリケーションおよびバイナリロギングのオプションと変数」を参照してください。

この問題を回避するには、新しいスレーブの --relay-log に対して、既存のスレーブで使用されたものと同じ値を使用します。(このオプションが既存のスレーブで明示的に設定されなかった場合は、existing_slave_hostname-relay-bin を使用します。)これができない場合、既存のスレーブのリレーログインデックスファイルを新しいスレーブにコピーし、新しいスレーブの --relay-log-index オプションを既存のスレーブで使用されていたものに設定します。(このオプションが既存のスレーブで明示的に設定されなかった場合は、existing_slave_hostname-relay-bin.indexを使用します。)また、すでに新しいスレーブを起動しようとして (このセクションの残りの手順に従ったあとに)、前に説明したものと同じようなエラーが発生した場合は、次の手順を実行します。

a. まだそのようにしていなかった場合は、新しいスレーブで STOP SLAVE を発行します。

   すでに既存のスレーブを再度起動した場合は、既存のスレーブでも STOP SLAVE を発行します。

b. 既存のスレーブのリレーログインデックスファイルの内容を新しいスレーブのリレーログインデックスファイルにコピーして、確実にファイルの既存の内容を上書きします。

c. このセクションの残りの手順に進みます。

3. マスター情報およびリレーログ情報リポジトリを既存のスレーブから新しいスレーブにコピーします (セクション17.2.2「レプリケーションリレーおよびステータスログ」を参照してください)。これらは、マスターのバイナリログとスレーブのリレーログの現在のログ座標を保持しています。

4. 既存のスレーブを起動します。

5. 新しいスレーブで構成を編集し、マスターまたは既存のスレーブのいずれかで使用されていない一意 server-id を新しいスレーブに割り当てます。

6. 新しいスレーブを起動します。スレーブは、そのマスター情報リポジトリ内の情報を使用して、レプリケーションプロセスを開始します。

17.1.1.10 スレーブでのマスター構成の設定

レプリケーションのためにマスターと通信するようにスレーブを設定するには、必要な接続情報をスレーブに伝える必要があります。これを行うには、次のステートメントをスレーブで実行して、オプション値をシステムに関係する実際の値と置き換えます。

```
mysql> CHANGE MASTER TO
    ->     MASTER_HOST='master_host_name',
    ->     MASTER_USER='replication_user_name',
    ->     MASTER_PASSWORD='replication_password',
    ->     MASTER_LOG_FILE='recorded_log_file_name',
    ->     MASTER_LOG_POS=recorded_log_position;
```

> **注記**
>
> レプリケーションでは、Unix ソケットファイルを使用できません。また、TCP/IP を使用してマスター MySQL サーバーに接続できる必要があります。

CHANGE MASTER TO ステートメントには、ほかのオプションもあります。たとえば、SSL を使用してセキュアなレプリケーションをセットアップできます。オプションの完全なリスト、および文字列値オプションに許可された最大長に関する情報については、セクション13.4.2.1「CHANGE MASTER TO 構文」を参照してください。

17.1.2 レプリケーション形式

レプリケーションが機能するのは、バイナリログに書き込まれたイベントがマスターから読み取られ、スレーブで処理されるためです。イベントは、イベントタイプに従ってさまざまな形式でバイナリログに記録されます。使用されるさまざまなレプリケーション形式は、イベントがマスターのバイナリログに記録されたときに使用されたバイナリロギング形式に対応しています。バイナリロギング形式とレプリケーションで使用される用語との関係は次のとおりです。

- ステートメントベースのバイナリロギングを使用する場合、マスターは SQL ステートメントをバイナリログに書き込みます。マスターからスレーブへのレプリケーションは、スレーブで SQL ステートメントを実行することによって機能します。これはステートメントベースのレプリケーション (SBR と短縮されることが多い) と呼ばれ、標準 MySQL ステートメントベースのバイナリロギング形式に対応します。MySQL バージョン 5.1.4 以前のレプリケーション機能では、この形式だけが使用されました。
- 行ベースのロギングを使用する場合、マスターは、各テーブル行がどのように変更されたかを表すイベントをバイナリログに書き込みます。マスターからスレーブへのレプリケーションは、テーブル行が変更されたことを表すイベントをスレーブにコピーすることによって機能します。これは行ベースのレプリケーション (RBRと短縮されることが多い) と呼ばれます。行ベースのレプリケーションでは、マスターは、各テーブル行がどのように変更されたかを表すイベントをバイナリログに書き込みます。
- 変更ログが記録されるときにステートメントベースと行ベースのどちらが適しているかによって、これらのロギングの組み合わせを使用するように MySQL を構成することもできます。これは混合形式のロギングと呼ばれます。混合形式のロギングを使用する場合、デフォルトでステートメントベースのログが使用されます。ステートメントに応じて、また使用されるストレージエンジンに応じて、ログは特定のケースで行ベースに自動的に切りかえられます。混合形式を使用するレプリケーションは多くの場合、混合ベースのレプリケーションまたは混合形式のレプリケーションと呼ばれます。詳細については、セクション5.2.4.3「混合形式のバイナリロギング形式」を参照してください。

MySQL 5.6 では、ステートメントベースの形式がデフォルトです。

MySQL Cluster　すべての MySQL Cluster NDB 7.3 および MySQL Cluster NDB 7.4 リリースのデフォルトバイナリロギング形式は 混合 です。MySQL Cluster Replication は常に行ベースのレプリケーションを使用し、NDB ストレージエンジンはステートメントベースのレプリケーションを使用できません。詳細については、セクション18.6.2「MySQL Cluster レプリケーションの一般要件」を参照してください。

混合形式を使用する場合、バイナリロギング形式は、使用されているストレージエンジンと実行されているステートメントによってある程度決定されます。混合形式のロギング、および異なるロギング形式のサポートを管理するルールの詳細は、セクション5.2.4.3「混合形式のバイナリロギング形式」を参照してください。

実行中 MySQL サーバーのロギング形式は、binlog_format サーバーシステム変数を設定することで制御されます。この変数はセッションまたはグローバルスコープで設定できます。新しい設定が有効になるタイミングと方法を管理するルールは、ほかの MySQL サーバーシステム変数と同じです。現在のセッションの変数を設定すると、そのセッションが終わるまで持続し、変更はほかのセッションに可視でなく、変数をグローバルに設定した場合、有効にするにはサーバーの再起動が必要です。詳細については、セクション13.7.4「SET 構文」を参照してください。

実行時にバイナリロギング形式を変更できない、つまりそのようにするとレプリケーションが失敗する状況があります。セクション5.2.4.2「バイナリログ形式の設定」を参照してください。

グローバルまたはセッション binlog_format 値を設定するには、SUPER 権限が必要です。

ステートメントベースと行ベースのレプリケーション形式には、異なる問題と制限があります。関連するメリットとデメリットの比較は、セクション17.1.2.1「ステートメントベースおよび行ベースレプリケーションのメリットとデメリット」を参照してください。

ステートメントベースのレプリケーションでは、ストアドルーチンまたはトリガーの複製で問題が発生する場合があります。代わりに行ベースのレプリケーションを使用することで、これらの問題を回避できます。詳細については、セクション20.7「ストアドプログラムのバイナリロギング」を参照してください。

17.1.2.1 ステートメントベースおよび行ベースレプリケーションのメリットとデメリット

それぞれのバイナリロギングの形式にメリットとデメリットがあります。ほとんどのユーザーにとって、データの完全性とパフォーマンスの最善の組み合わせが得られるのは、混合レプリケーション形式であるはずです。ただし、特定のタスクを実行するときにステートメントベースまたは行ベースレプリケーション形式固有の機能を利用する場合、関連するメリットとデメリットのサマリーを記述したこのセクションの情報を使用して、どちらが二ーズに最適であるかを決めることができます。

- ステートメントベースレプリケーションのメリット

- ステートメントベースレプリケーションのデメリット
- 行ベースレプリケーションのメリット
- 行ベースレプリケーションのデメリット

ステートメントベースレプリケーションのメリット

- バージョン 3.23 以来、MySQL に存在する実証済みのテクノロジーです。
- ログファイルに書き込まれるデータが少ないです。更新または削除が多くの行に影響を与える場合、これによってログファイルに必要なストレージ容量がかなり少なくなります。つまり、バックアップの取得とリストアをより短時間で達成できます。
- ログファイルには変更があったすべてのステートメントが含まれるため、データベースの監査に使用できます。

ステートメントベースレプリケーションのデメリット

- SBR にとって安全でないステートメント
  データを変更するすべてのステートメント (INSERT DELETE、UPDATE、REPLACE ステートメントなど) を、ステートメントベースレプリケーションを使用して複製できるわけではありません。ステートメントベースレプリケーションを使用する場合、非決定的動作は複製が困難です。そのような DML (データ変更言語) ステートメントの例には次が含まれます。
  - 非決定的な UDF またはストアドプログラムに依存するステートメント。そのような UDF またはストアドプログラムによって返される値は、それに提供されるパラメータ以外の要因に依存するため。(ただし、行ベースレプリケーションは、UDF またはストアドプログラムによって返される値を単純に置き換えるため、テーブル行およびデータに対するその影響はマスターとスレーブの両方で同じです。)詳細は、セクション 17.4.1.11「呼び出される機能のレプリケーション」を参照してください。
  - ORDER BY なしで LIMIT 句を使用する DELETE および UPDATE ステートメントは非決定的です。セクション17.4.1.16「レプリケーションと LIMIT」を参照してください。
  - 次のいずれかの関数を使用するステートメントは、ステートメントベースレプリケーションでは適切に複製できません。
    - LOAD_FILE()
    - UUID()、UUID_SHORT()
    - USER()
    - FOUND_ROWS()
    - SYSDATE() (マスターとスレーブの両方が --sysdate-is-now オプションで起動される場合を除きます)
    - GET_LOCK()
    - IS_FREE_LOCK()
    - IS_USED_LOCK()
    - MASTER_POS_WAIT()
    - RAND()
    - RELEASE_LOCK()
    - SLEEP()
    - VERSION()

    ただし、NOW() などを含めてほかのすべての関数はステートメントベースレプリケーションで正しく複製されます。

    詳細については、セクション17.4.1.15「レプリケーションとシステム関数」を参照してください。

  ステートメントベースレプリケーションで正しく複製できないステートメントは、ここに示すもののような警告でログが記録されます。

```
[Warning] Statement is not safe to log in statement format.
```

このような場合、類似の警告がクライアントにも発行されます。クライアントは SHOW WARNINGS を使用してそれを表示できます。

- INSERT ... SELECT は、行ベースレプリケーションよりも多くの行レベルロックが必要です。
- WHERE 句でインデックスが使用されていないためにテーブルスキャンを必要とする UPDATE ステートメントは、行ベースレプリケーションの場合より多くの行をロックする必要があります。
- InnoDB の場合: AUTO_INCREMENT を使用する INSERT ステートメントは、競合しないほかの INSERT ステートメントをブロックします。
- 複雑なステートメントの場合、行が更新または挿入される前に、スレーブでステートメントを評価して実行する必要があります。行ベースレプリケーションの場合、スレーブはステートメント全体を実行するのではなく、影響を受ける行だけを変更する必要があります。
- スレーブでの評価でエラーがあった場合、特に複雑なステートメントを実行するときに、ステートメントベースレプリケーションでは、影響を受ける行全体でエラーのマージンが時間とともに徐々に増える場合があります。セクション17.4.1.26「レプリケーション中のスレーブエラー」を参照してください。
- ストアドファンクションは、呼び出し元のステートメントと同じ NOW() 値で実行します。ただし、これはストアドプロシージャーには当てはまりません。
- 決定的な UDF はスレーブで適用される必要があります。
- テーブル定義は、マスターとスレーブで (ほぼ) 同じでなければいけません。詳細については、セクション17.4.1.9「テーブル定義が異なるマスターとスレーブでのレプリケーション」を参照してください。

行ベースレプリケーションのメリット

- すべての変更を複製できます。これはもっとも安全な形式のレプリケーションです。

  mysql データベースは複製されません。代わりに、mysql データベースはノード固有データベースとして見られます。行ベースレプリケーションは、このデータベース内のテーブルでサポートされません。代わりに、GRANT、REVOKE、およびトリガー、ストアドルーチン (ストアドプロシージャーを含む)、およびビューの操作など、通常この情報を更新するステートメントは、ステートメントベースレプリケーションでスレーブにすべて複製されます。

  CREATE TABLE ... SELECT などのステートメントの場合、CREATE ステートメントはテーブル定義から生成されてステートメントベース形式を使用して複製される一方、行挿入は行ベース形式を使用して複製されます。

- このテクノロジーはほかのほとんどのデータベース管理システムと同じです。ほかのシステムについての知識は MySQL で通用します。
- マスターで必要な行ロックは少ないため、次の種類のステートメントでは並列性が高くなります。
  - INSERT ... SELECT
  - AUTO_INCREMENT 付きの INSERT ステートメント
  - キーを使用しないまたは検査された行のほとんどを変更しない WHERE 句付きの UPDATE または DELETE ステートメント。
- INSERT、UPDATE、または DELETE ステートメントの場合、スレーブで必要な行ロックが少ないです。

行ベースレプリケーションのデメリット

- RBR では、ログに書き込む必要があるデータが増える可能性があります。ステートメントベースレプリケーションでは、DML ステートメント (UPDATE、DELETE ステートメントなど) を複製するためにステートメントだけをバイナリログに書き込みます。一方、行ベースレプリケーションでは変更されたすべての行をバイナリログに書き込みます。ステートメントが多くの行を変更する場合、行ベースレプリケーションは非常に多くのデータをバイナリログに書き込む可能性があります。このことはロールバックされるステートメントにも当てはまります。すなわち、バックアップの取得とリストアにも、より多くの時間がかかる可能性があります。また、データを書き込むためにバイナリログがロックされる時間が長くなるため、並列性の問題が発生する場合があります。

- 大きな BLOB 値を生成する決定的 UDF の場合は、ステートメントベースレプリケーションより行ベースレプリケーションの方が複製に時間がかかります。これは、データを生成するステートメントではなく、BLOB カラム値がログに書き込まれるためです。
- ログを検査してもどのステートメントが実行されたかを確認できず、マスターからどのステートメントが到着して実行されたかをスレーブで確認することもできません。

  ただし、オプション --base64-output=DECODE-ROWS および --verbose を付けて mysqlbinlog を使用すると、何のデータが変更されたかがわかります。
- MyISAM ストレージエンジンを使用するテーブルの場合、INSERT ステートメントを行ベースイベントとしてバイナリログに適用するときの方が、ステートメントとして適用するときよりも、スレーブでより強いロックが必要です。これは、MyISAM テーブルでの同時挿入が、行ベースレプリケーションを使用するときにサポートされないことを意味します。

17.1.2.2 行ベースロギングおよびレプリケーションの使用

MySQL はステートメントベースロギング (SBL)、行ベースロギング (RBL)、または混合形式ロギングを使用します。使用されるバイナリログの種類は、ロギングのサイズと効率に影響します。このため、行ベースレプリケーション (RBR) またはステートメントベースレプリケーション (SBR) の選択は、アプリケーションと環境に依存します。 このセクションでは、行ベース形式ログを使用するときの既知の問題について説明し、これをレプリケーションで使用する際のベストプラクティスをいくつか紹介します。

詳細については、セクション17.1.2「レプリケーション形式」およびセクション17.1.2.1「ステートメントベースおよび行ベースレプリケーションのメリットとデメリット」を参照してください。

MySQL Cluster レプリケーションに固有の問題について詳しくは (行ベースレプリケーションに依存します)、セクション18.6.3「MySQL Cluster レプリケーションの既知の問題」を参照してください。

- 一時テーブルの行ベースロギング　セクション17.4.1.22「レプリケーションと一時テーブル」で説明したように、行ベース形式を使用する場合、一時テーブルは複製されません。混合形式ロギングを使用する場合、一時テーブルを使用する「安全な」ステートメントは、ステートメントベース形式を使用してログが記録されます。詳細については、セクション17.1.2.1「ステートメントベースおよび行ベースレプリケーションのメリットとデメリット」を参照してください。

  行ベース形式を使用する場合、一時テーブルは複製されません (必要がないため)。また、一時テーブルはそれらを作成したスレッドからのみ読み取れるため、ステートメントベース形式を使用する場合でも、それらを複製することから得られるメリットはまずありません。

  MySQL 5.6 では、一時テーブルが作成された場合でも、ステートメントベースから行ベースにバイナリロギングモードを切り替えることができます。ただし、行ベース形式の使用中は、MySQL サーバーは所定の一時テーブルが作成されたときに有効であったロギングモードを特定できません。このため、このようなケースのサーバーは、所定のクライアントセッションが終了するときに、そこにまだ存在する各一時テーブルの DROP TEMPORARY TABLE IF EXISTS ステートメントのログを記録します。これは、一部のケースで不必要な DROP TEMPORARY TABLE ステートメントのログが記録される可能性があることを意味しますが、このステートメントは無害であり、テーブルが存在しない場合でも IF NOT EXISTS オプションがあることでエラーになりません。

  MySQL 5.6.6 以前では、行ベースロギングを使用するときに --disable-gtid-unsafe-statements オプションを使用すると、一時テーブルを使用する非トランザクション DML ステートメントがエラーで失敗していました (バイナリログに書き込まれないという事実にかかわらず)。MySQL 5.6.7 以降は、ステートメントが影響を与える非トランザクションテーブルが一時テーブルであるかぎり、binlog_format=ROW を使用するときにこのようなステートメントが許可されます (Bug #14272672)。
- 非トランザクションテーブルの RBL と同期　多くの行が影響を受ける場合、変更のセットは複数のイベントに分割されます。ステートメントがコミットすると、これらのイベントのすべてがバイナリログに書き込まれます。スレーブで実行中は、使用されるすべてのテーブルにテーブルロックが適用され、行はバッチモードで適用されます。(これは、スレーブによるテーブルコピーに使用されるエンジンに応じて効果的であったりなかったりします。)
- 待機時間およびバイナリログサイズ　RBL は各行の変更をバイナリログに書き込むため、そのサイズは急激に増える場合があります。このため、マスターでの変更に対応する変更をクライアントで行うために必要な時間が大きく増える可能性があります。アプリケーションでこのような遅延が発生する可能性を意識してください。
- バイナリログの読み取り　mysqlbinlog は、BINLOG ステートメントを使用してバイナリログ内の行ベースイベントを表示します(セクション13.7.6.1「BINLOG 構文」を参照してください)。このステートメントは base 64 でエンコードされた文字列 (その意味は明白ではありません) としてイベントを表示します。--base64-

output=DECODE-ROWS および --verbose オプションを付けて呼び出されたときは、mysqlbinlog はバイナリログの内容を人間が読める形式にします。バイナリログイベントが行ベース形式で書き込まれ、それらを読み取ったりレプリケーションまたはデータベース障害からリカバリしたりしたい場合は、このコマンドでバイナリログの内容を読み取ることができます。詳細については、セクション4.6.8.2「mysqlbinlog 行イベントの表示」を参照してください。

- バイナリログ実行エラーと slave_exec_mode　slave_exec_mode が IDEMPOTENT の場合は、元の行が見つからないために RBL から変更を適用できないときでも、エラーをトリガーしたり、レプリケーションが失敗したりしません。これは、更新がスレーブに適用されない可能性があるため、マスターとスレーブが同期されなくなったことを意味します。slave_exec_mode が IDEMPOTENT のときの、RBR での待機時間問題と非トランザクションテーブルの使用によって、マスターとスレーブとの相違がさらに広がる可能性があります。slave_exec_mode の詳細については、セクション5.1.4「サーバーシステム変数」を参照してください。

  > **注記**
  >
  > slave_exec_mode=IDEMPOTENT は一般的に、MySQL Cluster (IDEMPOTENT がデフォルト値) で循環レプリケーションまたはマルチマスターレプリケーションの場合にのみ役立ちます。

  ほかのシナリオの場合、slave_exec_mode を STRICT に設定することで通常は十分で、これがデフォルト値です。

  > **注記**
  >
  > 以前は MySQL Cluster を使用するときのデフォルト値は slave_exec_mode=IDEMPOTENT でしたが、MySQL Cluster NDB 7.3 以降ではこれは本当ではありません。

- バイナリログチェックサムがない　RBL はチェックサムを使用しないため、バイナリログを処理するときにネットワーク、ディスク、およびその他のエラーが特定されない場合があります。ネットワーク破損なしでデータを確実に転送するには、レプリケーション接続に SSL を使用します。CHANGE MASTER TO ステートメントには、SSL を使用するレプリケーションを有効にするオプションがあります。SSL を使用する MySQL のセットアップに関する一般的な情報については、セクション13.4.2.1「CHANGE MASTER TO 構文」を参照してください。

- サーバー ID に基づくフィルタリングはサポートされない　MySQL 5.6 では、CHANGE MASTER TO ステートメントで IGNORE_SERVER_IDS オプションを使用することで、サーバー ID に基づいてフィルタできます。このオプションは、ステートメントベースおよび行ベースロギング形式で使用できます。一部のスレーブで変更を除外するための別の方法は、UPDATE および DELETE ステートメントで、関係 @@server_id <> id_value 句を含む WHERE 句を使用することです。たとえば、WHERE @@server_id <> 1。ただし、これは行ベースロギングでは正しく動作しません。ステートメントフィルタリングに server_id システム変数を使用するには、ステートメントベースロギングを使用します。

- データベースレベルレプリケーションオプション　--replicate-do-db、--replicate-ignore-db、および --replicate-rewrite-db オプションの効果は、行ベースまたはステートメントベースのどちらのロギングが使用されるかに応じてかなり異なります。このため、データベースレベルオプションは避け、代わりに --replicate-do-table や --replicate-ignore-table などのテーブルレベルオプションを使用することをお勧めします。これらのオプションと、レプリケーション形式がそれらの動作にどのように影響するかについて詳しくは、セクション17.1.4「レプリケーションおよびバイナリロギングのオプションと変数」を参照してください。

- RBL、非トランザクションテーブル、および停止したスレーブ　行ベースロギングを使用するときに、スレーブサーバーが停止していて、スレーブスレッドが非トランザクションテーブルを更新している場合、スレーブデータベースが矛盾状態に到達する可能性があります。このため、行ベース形式を使用して複製されたすべてのテーブルに、InnoDB などのトランザクションストレージエンジンを使用することをお勧めします。スレーブ MySQL サーバーをシャットダウンする前に STOP SLAVE または STOP SLAVE SQL_THREAD を使用することは、問題発生の回避に役立ち、使用するロギング形式やストレージエンジンにかかわらず常に推奨されます。

17.1.2.3 バイナリロギングでの安全および安全でないステートメントの判断

MySQL レプリケーションでのステートメントの「安全」とは、ステートメントベース形式を使用してステートメントとその結果を正しく複製できるかどうかのことです。これがステートメントに当てはまる場合、ステートメントは安全と言い、そうでない場合は安全でないと言います。

一般的に、ステートメントが決定的である場合は安全であり、そうでない場合は安全ではありません。ただし、特定の非決定的関数は「安全でない」と見なされません (このセクションの後半の安全でないと見なされない非決定的関数。を参照してください)。また、浮動小数点数学関数 (ハードウェア依存) からの結果を使用するステート

メントは、常に安全でないと見なされます (セクション17.4.1.12「レプリケーションと浮動小数点値」を参照してください)。

安全および安全でないステートメントの処理　ステートメントは、ステートメントが安全と見なされるかどうかに応じて、およびバイナリロギング形式 (すなわち、binlog_format の現在の値) に基づいて異なる方法で処理されます。

- 行ベースロギングを使用する場合、安全および安全でないステートメントの扱いに違いはありません。
- 混合形式ロギングを使用する場合、安全でないとフラグされたステートメントは行ベース形式を使用してログが記録され、安全と見なされたステートメントはステートメントベース形式を使用してログが記録されます。
- ステートメントベースロギングを使用する場合、安全でないとフラグされたステートメントはこの結果に警告を生成します。安全なステートメントは通常どおりにログが記録されます。

安全でないとフラグされた各ステートメントは警告を生成します。以前は、このようなステートメントが大量にマスターで実行された場合、エラーログファイルが非常に大きくなる可能性がありました。これを回避するため、MySQL は次のように動作する警告抑止メカニズム (MySQL 5.6.7 で導入) を提供します。最近の 50 の ER_BINLOG_UNSAFE_STATEMENT 警告が 50 秒間隔で 50 回超生成された場合、警告抑止が有効になります。有効になっているときは、これによってこのような警告がエラーログに書き込まれることはありません。代わりに、このタイプの警告が 50 個生成されるたびに、「最後の警告が N 回、最近の S 秒間に繰り返されました」との注記がエラーログに書き込まれます。50 個の最近のこのような警告が 50 秒以内に発行されるかぎり、これが継続します。頻度がこのしきい値を下回ると、再度通常どおりに警告ログが記録されます。警告抑止は、ステートメントベースロギングでステートメントの安全がどのように判断されるか、および警告がクライアントにどのように送信されるかに影響しません。MySQL クライアントは引き続きこのようなステートメントごとに 1 つの警告を受け取ります。

詳細については、セクション17.1.2「レプリケーション形式」を参照してください。

安全でないと見なされるステートメント
次の特徴を持つステートメントは安全でないと見なされます。

- スレーブで異なる値を返す可能性があるシステム関数を含むステートメント。　これらの関数には FOUND_ROWS()、GET_LOCK()、IS_FREE_LOCK()、IS_USED_LOCK()、LOAD_FILE()、MASTER_POS_WAIT()、PASSW が含まれます。

  安全でないと見なされない非決定的関数。　これらの関数は決定的ではありませんが、ロギングおよびレプリケーション目的の場合は安全として処理されます: CONNECTION_ID()、CURDATE()、CURRENT_DATE()、CURRENT_TIME()、CURRENT_TIMESTAMP()、CURTIME()、LA および UTC_TIMESTAMP()。

  詳細については、セクション17.4.1.15「レプリケーションとシステム関数」を参照してください。

- システム変数への参照　ほとんどのシステム変数は、ステートメントベース形式で正しく複製されません。セクション17.4.1.34「レプリケーションと変数」を参照してください。例外については、セクション5.2.4.3「混合形式のバイナリロギング形式」を参照してください。
- UDF　UDF が何をするかは制御できないため、それが安全でないステートメントを実行していると推定する必要があります。
- トリガーまたはストアドプログラムは AUTO_INCREMENT カラムを持つテーブルを更新する。　行が更新される順番がマスターとスレーブで異なる可能性があるため、これは安全ではありません。

  また、複合主キーを持つテーブルに、この複合キーの先頭カラムでない AUTO_INCREMENT カラムが含まれるテーブルに INSERT することは、安全ではありません。

  詳細については、セクション17.4.1.1「レプリケーションと AUTO_INCREMENT」を参照してください。

- INSERT DELAYED ステートメント　このステートメントは、この行の挿入が並列実行されるステートメントによって差し込まれる可能性があるため、安全でないと見なされます。
- 複数の主キーまたは一意キーを持つテーブルでの INSERT ... ON DUPLICATE KEY UPDATE ステートメント　複数の主キーまたは一意キーを持つテーブルに対して実行されるときのこのステートメントは、安全でないと見なされます (ストレージエンジンがキーをチェックする順番に影響されやすいですが、これは決定的でなく、さらに MySQL Server が更新する行の選択がこれに依存するためです)。

  複数の一意キーまたは主キーを持つテーブルに対する INSERT ... ON DUPLICATE KEY UPDATE ステートメントは、MySQL 5.6.6 以降のステートメントベースレプリケーションでは安全でないとマークされます。(Bug #11765650、Bug #58637)

- LIMIT を使用する更新　行の取得順序が指定されていないため、安全でないと見なされます。セクション17.4.1.16「レプリケーションと LIMIT」を参照してください。
- ログテーブルへのアクセスまたは参照　システムログテーブルの内容は、マスターとスレーブで異なる可能性があります。
- トランザクション操作後の非トランザクション操作　トランザクション内で、トランザクション読み取りまたは書き込み後に非トランザクション読み取りまたは書き込みを実行することを許可することは、安全でないと見なされます。

  詳細については、セクション17.4.1.31「レプリケーションとトランザクション」を参照してください。
- セルフロギングテーブルへのアクセスまたは参照　セルフロギングテーブルへのすべての読み取りと書き込みは、安全でないと見なされます。トランザクション内で、セルフロギングテーブルへの読み取りまたは書き込みに続くステートメントも、安全でないと見なされます。
- LOAD DATA INFILE ステートメント　MySQL 5.6 以降では、LOAD DATA INFILE は安全でないと見なされ、ステートメントベースモードで警告が生成され、混合形式ロギング使用時に行ベース形式に切り替わる原因となります。セクション17.4.1.17「レプリケーションと LOAD DATA INFILE」を参照してください。

追加情報については セクション17.4.1「レプリケーションの機能と問題」を参照してください。

17.1.3 グローバルトランザクション識別子を使用したレプリケーション

このセクションでは、MySQL 5.6.5 で導入された、グローバルトランザクション識別子 (GTID) を使用したトランザクションベースレプリケーションについて説明します。GTID を使用することで、各トランザクションが発生元サーバーでコミットされて、スレーブによって適用されたあとに、それを特定して追跡できます。これは、GTID を使用することで、新しいスレーブを起動するとき、または新しいマスターにフェイルオーバーするときに、ログファイルまたはそれらのファイル内での位置を参照する必要がなくなり、これらのタスクが大幅に簡易化されることを意味します。GTID ベースレプリケーションが完全にトランザクションベースであるため、マスターとスレーブが一貫しているかどうかを判断することが容易です。マスターでコミットされたすべてのトランザクションがスレーブでもコミットされているかぎり、両者の一貫性は保証されます。ステートメントベースまたは行ベースレプリケーションを GTID に基づいて使用できます (セクション17.1.2「レプリケーション形式」を参照してください)。ただし、最善の結果を得るには、行ベース形式を使用することをお勧めします。

このセクションでは、次のトピックについて説明します。

- GTID がどのように定義および作成され、それらが MySQL Server 内でどのように表現されるか (セクション17.1.3.1「GTID の概念」を参照してください)。
- GTID ベースレプリケーションをセットアップおよび起動するための一般的な手順 (セクション17.1.3.2「GTID を使用したレプリケーションのセットアップ」を参照してください)。
- GTID を使用するときに新しいレプリケーションサーバーをプロビジョニングするために推奨される方法 (セクション17.1.3.3「フェイルオーバーおよびスケールアウトでの GTID の使用」を参照してください)。
- GTID ベースレプリケーションを使用するときに留意すべき制約と制限 (セクション17.1.3.4「GTID ベースレプリケーションの制約」を参照してください)。

GTID ベースレプリケーションに関係する MySQL Server オプションおよび変数については、セクション17.1.4.5「グローバルトランザクション ID のオプションと変数」を参照してください。GTID と一緒に使用するために MySQL 5.6 がサポートする SQL 関数については、セクション12.16「グローバルトランザクション ID とともに使用される関数」も参照してください。

> **注記**
>
> GTID は、MySQL Cluster が使用する NDB ストレージエンジンと互換性がなく、サポートされません。MySQL Cluster で GTID を有効にすると、NDB の問題が発生し、MySQL Cluster レプリケーションも失敗する可能性が非常に高くなります。

17.1.3.1 GTID の概念

グローバルトランザクション識別子 (GTID) は、発生元のサーバー (マスター) で作成され、そこでコミットされた各トランザクションに関連付けられる一意識別子です。この識別子は、サーバーはもちろん、特定のレプリケーションセットアップ内のすべてのサーバーに一意です。すべてのトランザクションとすべての GTID との間には 1 対 1 マッピングがあります。

GTID は座標のペアとして表現され、次に示すように、コロン文字 (:) で区切られます。

```
GTID = source_id:transaction_id
```

source_id は発生元サーバーを識別します。通常、サーバーの server_uuid はこの目的のために使用されます。transaction_id は、このサーバーでコミットされた順番によって決められるシーケンス番号です。たとえば、コミットされる最初のトランザクションには、その transaction_id として 1 が割り当てられ、同じ発生元サーバーでコミットされる 10 番目のトランザクションには transaction_id として 10 が割り当てられます。トランザクションに、GTID のシーケンス番号として 0 を割り当てることはできません。たとえば、UUID が 3E11FA47-71CA-11E1-9E33-C80AA9429562 の発生元サーバーでコミットされた 23 番目のトランザクションの GTID は次のとおりです。

```
3E11FA47-71CA-11E1-9E33-C80AA9429562:23
```

この形式は、SHOW SLAVE STATUS などのステートメントの出力やバイナリログで GTID を表すために使用されます。それらは、mysqlbinlog --base64-output=DECODE-ROWS でログファイルを表示するとき、または SHOW BINLOG EVENTS からの出力でも見ることができます。

GTID は SHOW MASTER STATUS や SHOW SLAVE STATUS などのステートメントの出力に書き込まれると、同じサーバーから発生する GTID シーケンスは次のように単一表現にまとめられることがあります。

```
3E11FA47-71CA-11E1-9E33-C80AA9429562:1-5
```

この例は、server_uuid が 3E11FA47-71CA-11E1-9E33-C80AA9429562 の MySQL Server で発生する 1 番目から 5 番目までのトランザクションを表します。

MySQL 5.6.6 以降では、この形式は START SLAVE のオプション SQL_BEFORE_GTIDS および SQL_AFTER_GTIDS で必要な引数を指定するためにも使用されます。

GTID セット

GTID セットとは、次のように表現されるグローバルトランザクション識別子のセットのことです。

```
gtid_set:
    uuid_set [, uuid_set] ...
    | ''

uuid_set:
    uuid:interval[:interval]...

uuid:
    hhhhhhhh-hhhh-hhhh-hhhh-hhhhhhhhhhhh

h:
    [0-9|A-F]

interval:
    n[-n]

    (n >= 1)
```

GTID セットは MySQL Server でいくつかの方法で使用されます。たとえば、gtid_executed と gtid_purged のシステム変数によって格納される値は、GTID セットとして表現されます。また、関数 GTID_SUBSET() および GTID_SUBTRACT() には、入力として GTID セットが必要です。

GTID はマスターとスレーブとの間で常に保持されます。これは、バイナリログを検査することで、スレーブに適用されたトランザクションのソースをいつでも特定できることを意味します。また、ある GTID のトランザクションがあるサーバーでコミットされると、同じ GTID のそれ以降のトランザクションはそのサーバーで無視されます。このように、マスターでコミットされるトランザクションはスレーブで一度だけ適用できるため、一貫性の保証に役立ちます。

GTID が使用されるときは、マスター上でのファイルの名前やそのファイル内での位置などの非ローカルデータがスレーブに必要なくなります。マスターとの同期に必要なすべてのデータは、レプリケーションデータストリームから直接取得されます。データベース管理者または開発者からは、GTID は、ファイルオフセットペアを完全に置き換えるものです (以前はマスターとスレーブ間のデータフローを開始、停止、または再開ポイントを決定するために必要でした)。これは、レプリケーションに GTID を使用しているときは、所定のマスターから複製するようにスレーブに指示するために使用する MASTER_LOG_FILE または MASTER_LOG_POS オプションを CHANGE MASTER TO ステートメントに含める必要がありません (含めたくありません)。これらのオプションの代わりに必要なのは、MySQL 5.6.5 で導入された MASTER_AUTO_POSITION オプションを有効にすることだけです。GTID ベースレプリケーションを使用してマスターとスレーブを構成して起動するために必要な正確な手順については、セクション17.1.3.2「GTID を使用したレプリケーションのセットアップ」を参照してください。

GTID の生成とライフサイクルは次の手順で構成されます。

1. トランザクションがマスター上で実行され、コミットされます。

   このトランザクションには、マスターの UUID と、このサーバーでまだ使用されていない最小のゼロでないトランザクションシーケンス番号を使用する GTID が割り当てられます。GTID はマスターのバイナリログに書き込まれます (ログ内のトランザクション自体の直前)。

2. バイナリログがスレーブに転送され、スレーブのリレーログに格納されたあと (このプロセスのために確立されたメカニズムを使用します。詳細については、セクション17.2「レプリケーションの実装」を参照してください)、スレーブは GTID を読み取り、その gtid_next システム変数値をこの GTID として設定します。これは、次のトランザクションのログはこの GTID を使用して記録される必要があることをスレーブに伝えます。

   スレーブはセッションコンテキストに gtid_next を設定することに注目することが重要です。

3. スレーブは、自身のバイナリログにトランザクションログを記録するためにこの GTID が確実にまだ使用されていないことをチェックします。この GTID が使用されていない場合にのみ、スレーブは GTID を書き込み、トランザクションを適用します (そして、トランザクションをバイナリログに書き込みます)。トランザクション自体を処理する前に、まずトランザクションの GTID を読み取ってチェックすることで、スレーブは、この GTID を持つ前のトランザクションがスレーブにまだ適用されていないこと、さらにほかのセッションがまだこの GTID を読み取っておらず、関連付けられたトランザクションをまだコミットしていないことを保証します。つまり、複数のクライアントが並列に同じトランザクションを適用することは許可されません。

4. gtid_next が空でないため、スレーブはこのトランザクションの GTID を生成しようとせず、代わりにこの変数に格納された GTID (すなわち、マスターから取得した GTID) をバイナリログ内のトランザクションの直前に書き込みます。

17.1.3.2 GTID を使用したレプリケーションのセットアップ

このセクションでは、MySQL 5.6 で GTID ベースレプリケーションを構成および起動するためのプロセスについて説明します。これは、レプリケーションマスターをはじめて起動しているか、停止できることを前提とする「コールドスタート」手順です。動作中マスターから GTID を使用したレプリケーションスレーブのプロビジョニングについては、セクション17.1.3.3「フェイルオーバーおよびスケールアウトでの GTID の使用」を参照してください。

最大限単純な GTID レプリケーショントポロジー (1 つのマスターと 1 つのスレーブで構成) を起動するこのプロセスの主要手順は、次のとおりです。

1. レプリケーションがすでに動作している場合、両方のサーバーを読み取り専用にすることでそれらを同期します。

2. 両方のサーバーを停止します。

3. GTID、バイナリロギング、およびスレーブ更新ロギングを有効にし、GTID ベースレプリケーションに安全でないステートメントを無効にした状態で、両方のサーバーを再起動します。また、サーバーは読み取り専用モードで起動し、スレーブ SQL および I/O スレッドがスレーブで起動されないようにしてください。

   説明したサーバーを起動するために必要な mysqld オプションについては、このセクションの後半の例で説明します。

4. マスターをレプリケーションデータソースとして使用し、自動ポジショニングを使用するように、スレーブに指示してから、スレーブを起動します。

   この手順の実施に必要な SQL ステートメントは、このセクションの後半の例で説明します。

5. 更新を再度受け取ることができるように、両方のサーバーの読み取り専用モードを無効にします。

次の例では、2 つのサーバーが MySQL の 「クラシック」ファイルベースレプリケーションプロトコルを使用してマスターおよびスレーブとしてすでに動作しています。

後続のほとんどの手順では、SUPER 権限を持つ MySQL root アカウントまたは別の MySQL ユーザーアカウントを使用する必要があります。mysqladmin shutdown には、SUPER 権限または SHUTDOWN 権限が必要です。

手順 1: サーバーを同期します　　サーバーを読み取り専用にします。これを行うには、次のステートメントを両方のサーバーで実行することで、read_only システム変数を有効にします。

```
mysql> SET @@GLOBAL.read_only = ON;
```

すると、スレーブはマスターに追い付くことができます。続行する前にスレーブがすべての更新を処理したことを確認することが非常に重要です。

手順 2: 両方のサーバーを停止します。　ここで示すように、mysqladmin を使用して各サーバーを停止します。ここで、username はサーバーをシャットダウンするのに十分な権限を持つ MySQL ユーザーのユーザー名です。

```
shell> mysqladmin -uusername -p shutdown
```

次に、プロンプトにこのユーザーのパスワードを指定します。

手順 3: GTID を有効にした状態で両方のサーバーを再起動します　グローバルトランザクション識別子でバイナリログギングを有効にするには、GTID モード、バイナリログギング、およびスレーブ更新ログギングを有効にし、GTID ベースレプリケーションに安全でないステートメントを無効にした状態で、各サーバーを起動する必要があります。また、読み取り専用モードで両方のサーバーを起動することで不要または予期しない更新がどちらかのサーバーで実行されないようにしてください。これは、両方のサーバーは (少なくとも) 次の mysqld_safe 呼び出しで示すオプションで起動する必要があることを意味します。

```
shell> mysqld_safe --gtid_mode=ON --log-bin --log-slave-updates --enforce-gtid-consistency &
```

> **注記**
>
> MySQL 5.6.9 より前では、--enforce-gtid-consistency の名前は --disable-gtid-unsafe-statements でした。

また、ここで示す例で指定したほかのサーバーオプションと --skip-slave-start オプションでスレーブを起動してください。

> **注記**
>
> --gtid-mode はブールではなく、列挙です。このオプションを設定するときは、値 ON または OFF の一方だけを使用します。0 や 1 などの数値を使用すると、予期しない結果になる場合があります。

--gtid-mode および --enforce-gtid-consistency サーバーオプションの詳細については、セクション17.1.4.5「グローバルトランザクション ID のオプションと変数」を参照してください。

構成に応じて、mysqld_safe またはほかの mysqld 起動スクリプトに追加オプションを指定します。

手順 4: マスターを使用するようにスレーブに指示します　マスターをレプリケーションデータソースとして使用し、ファイルベースポジショニングではなく GTID ベース自動ポジショニングを使用するように、スレーブに指示します。トランザクションが GTID で識別されることをスレーブに伝える MASTER_AUTO_POSITION オプションを使用して、スレーブで CHANGE MASTER TO ステートメントを実行します。

マスターのホスト名とポート番号、およびマスターへの接続にスレーブが使用できるレプリケーションユーザーアカウントのユーザー名とパスワードの適切な値を指定する必要がある場合もあります。これらが手順 1 より前にすでに設定されていて、変更する必要がない場合、対応するオプションをここで示すステートメントから安全に省略できます。

```
mysql> CHANGE MASTER TO
    >     MASTER_HOST = host,
    >     MASTER_PORT = port,
    >     MASTER_USER = user,
    >     MASTER_PASSWORD = password,
    >     MASTER_AUTO_POSITION = 1;
```

MASTER_LOG_FILE オプションと MASTER_LOG_POS オプションは、MASTER_AUTO_POSITION が 1 に設定されているときに使用できない場合があります。そのように試みると、CHANGE MASTER TO ステートメントはエラーで失敗します。(GTID ベースレプリケーションからファイルと位置に基づくレプリケーションに戻る必要がある場合、CHANGE MASTER TO ステートメントで MASTER_AUTO_POSITION = 0 とともにこれらのオプションの 1 つまたは両方を使用する必要があります。)

CHANGE MASTER TO ステートメントが成功したことを前提として、このようにスレーブを起動できます。

```
mysql> START SLAVE;
```

手順 5: 読み取り専用モードを無効にします　次のステートメントを実行することで、マスターは更新の受け付けを再度開始できます。

```
mysql> SET @@GLOBAL.read_only = OFF;
```

これで、GTID ベースレプリケーションは動作中になり、マスターで前のようにアクティビティーを開始 (または再開) できます。セクション17.1.3.3「フェイルオーバーおよびスケールアウトでの GTID の使用」では、GTID 使用時の新しいスレーブの作成について説明します。

17.1.3.3 フェイルオーバーおよびスケールアウトでの GTID の使用

MySQL 5.6.9 でグローバルトランザクション識別子 (GTID) に基づいて MySQL Replication を使用するとき、さらにあとで新しいスレーブ (スケールアウトに使用でき、必要に応じてフェイルオーバー時にマスターに昇格される) をプロビジョニングするために、いくつかの技術があります。このセクションでは、次に示す 4 つの技術について説明します。

- 単純なレプリケーション
- データとトランザクションをスレーブにコピーする
- 空のトランザクションの注入
- gtid_purged によるトランザクションの除外

グローバルトランザクション識別子は、特にレプリケーションデータフローおよびフェイルオーバーアクティビティーの一般管理を簡易化するために、MySQL Replication に追加されました。各識別子は、全体でトランザクションを構成するバイナリログイベントセットを一意に識別します。GTID はデータベースに変更を適用する際に重要な役割を果たします。サーバーは、以前に処理済みと認識している識別子のトランザクションを自動的にスキップします。この動作は、自動レプリケーションポジショニングおよび正確なフェイルオーバーのために重要です。

トランザクションを構成する識別子とイベントセットとの間のマッピングは、バイナリログで取得されます。このことは、別の既存のサーバーからのデータで新しいサーバーをプロビジョニングする際に、いくつかの課題を提起します。新しいサーバーで識別子セットを再生するには、古いサーバーから新しいサーバーに識別子をコピーし、識別子と実際のイベントとの間の関係を保持する必要があります。これは、フェイルオーバーまたはスイッチオーバーで新しいマスターになる候補としてすぐに使用可能なスレーブをリストアするために必要です。

単純なレプリケーション　これは、新しいサーバーですべての識別子とトランザクションを再生するためのもっとも簡単な方法です。単純に、新しいサーバーをすべての実行履歴を持つマスターのスレーブにして、両方のサーバーでグローバルトランザクション識別子を有効にします。詳細については、セクション17.1.3.2「GTID を使用したレプリケーションのセットアップ」を参照してください。

レプリケーションが起動されたあと、新しいサーバーはマスターからバイナリログ全体をコピーすることで、すべての GTID に関するすべての情報を取得します。

この方法は単純で効果的ですが、スレーブはマスターからバイナリログを読み取る必要があります。新しいスレーブがマスターに追い付くために比較的長い時間かかることがあり、この方法は迅速なフェイルオーバーまたはバックアップからのリストアには適していません。このセクションでは、バイナリログファイルを新しいサーバーにコピーすることで、マスターからすべての実行履歴をフェッチするのを回避する方法について説明します。

データとトランザクションをスレーブにコピーする　再生を トランザクション履歴全体で行うには時間がかかることがあり、新しいレプリケーションスレーブをセットアップするときに大きなボトルネックになります。この要件を解消するために、マスターに含まれているデータセットのスナップショット、バイナリログ、およびグローバルトランザクション情報がスレーブにインポートされます。バイナリログが再生されたあとにレプリケーションを開始できるため、スレーブは残りのトランザクションに追い付くことができます。

この方法にはいくつかの種類があり、ここで示すように、データダンプとバイナリログからのトランザクションがスレーブに転送される方法に違いがあります。

データセット	トランザクション履歴
• mysql クライアントを使用して、mysqldump で作成されたダンプファイルをインポートします。--master-data オプションを使用してバイナリロギング情報を取り込み、AUTO (デフォルト) または ON の --set-gtid-purged (MySQL 5.6.9 以降で使用可能) を使用して実行済みトランザクションに関する情報を取り込みます。スレーブでダンプをインポートするときは、--gtid-mode=ON にしてください。(Bug #14832472) • スレーブを停止し、マスターのデータディレクトリの内容をスレーブのデータディレクトリにコピーしてから、スレーブを再起動します。	gtid_mode が ON でない場合、GTID モードを有効にした状態でサーバーを再起動します。 • mysqlbinlog を --read-from-remote-server および --read-from-remote-master オプションで使用してバイナリログをインポートします。 • マスターのバイナリログファイルをスレーブにコピーします。mysqlbinlog --read-from-remote-server --raw を使用して、スレーブからコピーを作成できます。これらは、次のいずれかの方法でスレーブに読み取ることができます。 • スレーブの binlog.index ファイルを更新して、コピーされたログファイルをポイントします。次に、mysql クライアントで CHANGE MASTER TO

データセット	トランザクション履歴
	ステートメントを実行して最初のログファイルをポイントし、START SLAVE を実行してそれらを読み取ります。 • mysqlbinlog > file (--raw オプションなし) を使用して、mysql クライアントが処理できる SQL ファイルにバイナリログファイルをエクスポートします。

セクション4.6.8.3「バイナリログファイルのバックアップのための mysqlbinlog の使用」も参照してください。

この方法には、新しいサーバーをほぼ直後に使用できるというメリットがあります。ただし、スナップショットまたはダンプファイルが再生されている間にコミットされたトランザクションだけは、既存のマスターから取得される必要があります。これは、そのスレーブはすぐには使用できないことを意味しますが、スレーブがこれら少量の残りのトランザクションに追い付くために比較的短い時間しか必要ないはずです。

前もってターゲットサーバーにバイナリログをコピーすることは、トランザクション実行履歴全体をリアルタイムにマスターから読み取るよりも、通常は速いです。ただし、サイズやその他の考慮事項により、必要なときにこれらのファイルをターゲットに移動することが常に実現できるとはかぎらない場合があります。このセクションで説明した新しいスレーブをプロビジョニングするための残りの 2 つの方法は、別の手段を使用してトランザクションに関する情報を新しいスレーブに転送します。

空のトランザクションの注入　マスターのグローバル gtid_executed 変数には、マスターで実行されるすべてのトランザクションのセットが含まれます。新しいサーバーをプロビジョニングするためにスナップショットを作成するときにバイナリログをコピーする代わりに、スナップショットが作成されたサーバーで gtid_executed の内容に注目できます。新しいサーバーをレプリケーションチェーンに追加する前に、マスターの gtid_executed に含まれるトランザクション識別子ごとに、新しいサーバーで次のように単純に空のトランザクションをコミットします。

```
SET GTID_NEXT='aaa-bbb-ccc-ddd:N';

BEGIN;
COMMIT;

SET GTID_NEXT='AUTOMATIC';
```

すべてのトランザクション識別子が空のトランザクションを使用してこのように回復されたあとに、ここで示すようにスレーブのバイナリログをフラッシュしてパージする必要があります。ここで、N は現在のバイナリログファイル名のゼロでないサフィクスです。

```
FLUSH LOGS;
PURGE BINARY LOGS TO 'master-bin.00000N';
```

あとでマスターに昇格されたときにこのサーバーが不適切なトランザクションでレプリケーションストリームがあふれるのを回避するために、これを行うことをお勧めします。(FLUSH LOGS ステートメントは強制的に新しいバイナリログファイルを作成します。PURGE BINARY LOGS は空のトランザクションをパージしますが、その識別子を保持します。)

この方法は、本質的にはスナップショットであるけれども、あとでマスターになれるサーバーを作成します (そのバイナリログ履歴がレプリケーションストリームのそれに一致したとき、つまりマスターに追い付いたとき)。この結果は、残りのプロビジョニング方法を使用して得られる結果に実質的に似ています (次のいくつかの段落で説明します)。

gtid_purged によるトランザクションの除外　マスターのグローバル gtid_purged 変数には、マスターのバイナリログからパージされたすべてのトランザクションのセットが含まれます。前に説明した方法と同様に (空のトランザクションの注入を参照してください)、スナップショットが作成されたサーバーで (バイナリログを新しいサーバーにコピーする代わりに) gtid_executed の値を記録できます。前の方法とは異なり、空のトランザクションをコミットする必要はありません (PURGE BINARY LOGS を発行する必要はありません)。代わりに、バックアップまたはスナップショットが作成されたサーバーの gtid_executed の値に基づいて、スレーブで直接 gtid_purged を設定できます。

> **注記**
>
> MySQL 5.6.9 より前は、gtid_purged を設定できませんでした。(Bug #14797808)

空のトランザクションを使用する方法と同様に、この方法は、機能的にはスナップショットであるけれども、あとでマスターになれるサーバーを作成します (そのバイナリログ履歴がレプリケーションマスターまたはグループのそれに一致したとき)。

17.1.3.4 GTID ベースレプリケーションの制約

GTID ベースレプリケーションはトランザクションに依存しているため、そうでなければ MySQL で使用できるいくつかの機能が、それを使用するときにサポートされません。このセクションでは、GTID ベースレプリケーションの制約と制限についての情報を提供します。

非トランザクションストレージエンジンに関係する更新　GTID を使用する場合、MyISAM などの非トランザクションストレージエンジンを使用するテーブルへの更新は、InnoDB などのトランザクションストレージエンジンを使用するテーブルへの更新と同じステートメントまたはトランザクションで実行できません。

この制約は、非トランザクションストレージエンジンを使用するテーブルへの更新とトランザクションストレージエンジンを使用するテーブルへの更新が、同じトランザクション内に混在していると、複数の GTID が同じトランザクションに割り当てられる可能性があるためです。

このような問題は、マスターとスレーブが異なるストレージエンジンをそれぞれのバージョンの同じテーブルに使用するときにも発生する可能性があります (一方のストレージエンジンはトランザクション、他方はそうではない)。

今挙げたいずれの場合も、トランザクションと GTID との間の 1 対 1 対応が壊れていて、GTID ベースレプリケーションは正しく機能できません。

CREATE TABLE ... SELECT ステートメント　CREATE TABLE ... SELECT はステートメントベースレプリケーションには安全ではありません。行ベースレプリケーションを使用する場合、このステートメントのログは実際には 2 つの異なるイベントとして記録されます。1 つはテーブルを作成するためのもの、他方はソーステーブルから作成されたばかりの新しいテーブルに行を挿入するためのものです。このステートメントがトランザクション内で実行されると、同じトランザクション識別子を受け取るこれらの 2 つのイベントにいくつかのケースが可能であり、これは挿入を含むトランザクションがスレーブによってスキップされることを意味します。このため、GTID ベースレプリケーションを使用する場合は、CREATE TABLE ... SELECT はサポートされません。

一時テーブル　CREATE TEMPORARY TABLE および DROP TEMPORARY TABLE ステートメントは、GTID を使用するとき (つまり、サーバーが --enforce-gtid-consistency オプションで起動されたとき) に、トランザクション内でサポートされません。GTID を有効にした状態でこれらのステートメントを使用することは可能です (ただし、トランザクションの外側のみ、および autocommit=1 のときのみ)。

サポートされないステートメントの実行の回避　GTID ベースレプリケーションが失敗するステートメントの実行を回避するために、GTID を有効にするときは、すべてのサーバーが --enforce-gtid-consistency オプションで起動される必要があります。これにより、このセクションですでに説明したタイプのステートメントはエラーで失敗します。

GTID を有効にするときに必要なほかの起動オプションについては、セクション17.1.3.2「GTID を使用したレプリケーションのセットアップ」を参照してください。

GTID を使用するときに、sql_slave_skip_counter はサポートされません。トランザクションをスキップする必要がある場合は、代わりにマスターの gtid_executed 変数を使用してください。詳細については、空のトランザクションの注入を参照してください。

GTID モードと mysqldump　MySQL 5.6.9 以降では、ターゲットサーバーのバイナリログに GTID がないこと場合に、mysqldump を使用して作成されたダンプを GTID モードが有効な状態で動作する MySQL Server にインポートできます。

MySQL 5.6.9 より前では、mysqldump はグローバルトランザクション ID を記録せず、GTID をリストアするためにバイナリログと mysqlbinlog を使用する必要がありました。(Bug #14797808、Bug #14832472)

GTID モードと mysql_upgrade　MySQL 5.6.7 より前では、mysql_upgrade が --write-binlog=OFF で実行されないかぎり、mysql_upgrade は --gtid-mode=ON で動作する MySQL Server に接続できませんでした。(そうでない場合は、mysqld は、mysql_upgrade を実行する前に --gtid-mode=OFF で再起動されてから、あとで --gtid_mode=ON で再起動される必要がありました。)これは MySQL 5.6.7 以降では問題ではなく、mysql_upgrade はデフォルトで --write-binlog=OFF で動作します。(Bug #13833710) ただし、そのようにすることは推奨されません。mysql_upgrade が MyISAM ストレージエンジン (非トランザクション) を使用するシステムテーブルに変更を加えることができるためです。

17.1.4 レプリケーションおよびバイナリロギングのオプションと変数

以降のセクションでは、mysqld オプション、およびレプリケーションで使用されてバイナリログを制御するためのサーバー変数の情報について説明します。レプリケーションマスターとレプリケーションスレーブで使用されるオプションと変数のうち、バイナリロギングに関係するオプションおよび変数であるものは、別途説明します。これらのオプションと変数に関する基本情報するクイックリファレンス表のセットも含まれています。

特に重要なものは --server-id オプションです。

コマンド行形式	--server-id=#
システム変数	server_id
スコープ	グローバル
動的	はい
型	数値
デフォルト	0
最小値	0
最大値	4294967295

このオプションはマスターとスレーブのどちらのレプリケーションサーバーに共通で、レプリケーションで使用され、マスターおよびスレーブサーバーがそれら自身を一意に識別できます。詳細については、セクション17.1.4.2「レプリケーションマスターのオプションと変数」,およびセクション17.1.4.3「レプリケーションスレーブのオプションと変数」を参照してください。

マスターおよび各スレーブでは、--server-id オプションを使用して、範囲が 1 から 2^{32} − 1 の一意レプリケーション ID を確立する必要があります。「一意」とは、各 ID が、ほかのレプリケーションマスターまたはスレーブで使用されるほかのあらゆる ID と異なっている必要があるということです。たとえば、server-id=3。

--server-id を省略すると、デフォルト ID は 0 になり、この場合、マスターはすべてのスレーブからの接続を拒否し、スレーブはマスターへの接続を拒否します。MySQL 5.6 では、サーバー ID が明示的に 0 に設定されていても、デフォルトの使用が許可されていても、サーバーは server_id システム変数を 1 に設定します。これは MySQL 5.7 で修正された既知の問題です。

詳細については、セクション17.1.1.2「レプリケーションスレーブ構成の設定」を参照してください。

server_uuid

MySQL 5.6 以降、サーバーはユーザーが指定する --server-id に加えて真の UUID を生成します。これは、グローバルな読み取り専用変数 server_uuid として使用できます。

システム変数	server_uuid
スコープ	グローバル
動的	いいえ
型	文字列

起動時、MySQL サーバーは次のように自動的に UUID を取得します。

1. ファイル data_dir/auto.cnf に書かれている UUID を読み取って使用しようとします (ここで、data_dir はサーバーのデータディレクトリ)。
2. data_dir/auto.cnf が見つからない場合、新しい UUID を生成してこのファイルに保存します (必要に応じてファイルを作成します)。

auto.cnf ファイルは、my.cnf または my.ini ファイルに使用されるものに類似した形式です。MySQL 5.6 では、auto.cnf に単一 server_uuid 設定と値を含む、単一 [auto] セクションのみがあります。ファイルの内容は次に示すものに似ています。

```
[auto]
server_uuid=8a94f357-aab4-11df-86ab-c80aa9429562
```

> **重要**
>
> auto.cnf ファイルは自動的に生成されます。このファイルを書き込んだり修正したりしようとしないでください。

MySQL 5.6 以降では、MySQL レプリケーションを使用するときに、マスターとスレーブは互いの UUID がわかります。スレーブの UUID の値は SHOW SLAVE HOSTS の出力で確認できます。START SLAVE が実行されたあとは (前ではありません)、マスターの UUID の値はスレーブでは SHOW SLAVE STATUS の出力で確認できます。

> **注記**
>
> STOP SLAVE または RESET SLAVE ステートメントを発行しても、スレーブで使用されるマスターの UUID はリセットされません。

MySQL 5.6.5 以降では、サーバーの server_uuid は、そのサーバーで発生するトランザクションの GTID でも使用されます。詳細は、セクション17.1.3「グローバルトランザクション識別子を使用したレプリケーション」を参照してください。

起動時に、スレーブ I/O スレッドは、そのマスターの UUID がそれ自身と等しい場合、--replicate-same-server-id オプションが設定されている場合を除き、エラーを生成して中断します。また次のどちらかが true の場合、スレーブ I/O スレッドは警告を生成します。

- 予期された server_uuid を持つマスターが存在しない。
- CHANGE MASTER TO ステートメントがこれまで実行されなかったのみ、マスターの server_uuid が変化した。

> **注記**
>
> MySQL 5.6 で server_uuid システム変数が追加されても、このセクションですでに説明したように、MySQL レプリケーションの準備と実行の一部として MySQL サーバーごとに一意の --server-id を設定する必要があることは変わりません。

17.1.4.1 レプリケーション、バイナリロギングオプション、および変数のリファレンス

次の表は、レプリケーションおよびバイナリログに適用できる、MySQL コマンド行オプションとシステム変数に関する基本情報の一覧です。

表 17.1 MySQL 5.6 でのレプリケーションのオプションおよび変数のサマリー

オプションまたは変数名		
コマンド行	システム変数	ステータス変数
オプションファイル	スコープ	動的
メモ		
Com_change_master		
いいえ	いいえ	はい
いいえ	両方	いいえ
説明: CHANGE MASTER ステートメントの数		
Com_show_master_status		
いいえ	いいえ	はい
いいえ	両方	いいえ
説明: SHOW MASTER STATUS ステートメントの数		
Com_show_new_master		
いいえ	いいえ	はい
いいえ	両方	いいえ
説明: SHOW NEW MASTER ステートメントの数		
Com_show_slave_hosts		
いいえ	いいえ	はい
いいえ	両方	いいえ
説明: SHOW SLAVE HOSTS ステートメントの数		
Com_show_slave_status		
いいえ	いいえ	はい
いいえ	両方	いいえ
説明: SHOW SLAVE STATUS ステートメントの数		
Com_slave_start		
いいえ	いいえ	はい
いいえ	両方	いいえ
説明: START SLAVE ステートメントの数		
Com_slave_stop		

オプションまたは変数名		
コマンド行	システム変数	ステータス変数
オプションファイル	スコープ	動的
メモ		
いいえ	いいえ	はい
いいえ	両方	いいえ
説明: STOP SLAVE ステートメントの数 Rpl_semi_sync_master_clients		
いいえ	いいえ	はい
いいえ	グローバル	いいえ
説明: 準同期スレーブの数 Rpl_semi_sync_master_net_avg_wait_time		
いいえ	いいえ	はい
いいえ	グローバル	いいえ
説明: マスターがスレーブ応答を待機した平均時間 Rpl_semi_sync_master_net_wait_time		
いいえ	いいえ	はい
いいえ	グローバル	いいえ
説明: マスターがスレーブ応答を待機した合計時間 Rpl_semi_sync_master_net_waits		
いいえ	いいえ	はい
いいえ	グローバル	いいえ
説明: マスターがスレーブ応答を待機した合計回数 Rpl_semi_sync_master_no_times		
いいえ	いいえ	はい
いいえ	グローバル	いいえ
説明: マスターが準同期レプリケーションをオフにした回数 Rpl_semi_sync_master_no_tx		
いいえ	いいえ	はい
いいえ	グローバル	いいえ
説明: 正常に肯定応答されなかったコミットの数 Rpl_semi_sync_master_status		
いいえ	いいえ	はい
いいえ	グローバル	いいえ
説明: 準同期レプリケーションがマスター上で動作しているかどうか Rpl_semi_sync_master_timefunc_failures		
いいえ	いいえ	はい
いいえ	グローバル	いいえ
説明: 時間関数を呼び出すときにマスターが失敗した回数 Rpl_semi_sync_master_tx_avg_wait_time		
いいえ	いいえ	はい
いいえ	グローバル	いいえ
説明: マスターが各トランザクションを待機した平均時間 Rpl_semi_sync_master_tx_wait_time		
いいえ	いいえ	はい

オプションまたは変数名		
コマンド行	システム変数	ステータス変数
オプションファイル	スコープ	動的
メモ		
いいえ	グローバル	いいえ
説明: マスターがトランザクションを待機した合計時間 Rpl_semi_sync_master_tx_waits		
いいえ	いいえ	はい
いいえ	グローバル	いいえ
説明: マスターがトランザクションを待機した合計回数 Rpl_semi_sync_master_wait_pos_backtraverse		
いいえ	いいえ	はい
いいえ	グローバル	いいえ
説明: 以前待機したイベントよりもバイナリ座標が低いイベントをマスターが待機した合計回数 Rpl_semi_sync_master_wait_sessions		
いいえ	いいえ	はい
いいえ	グローバル	いいえ
説明: 現在スレーブ応答を待機しているセッションの数 Rpl_semi_sync_master_yes_tx		
いいえ	いいえ	はい
いいえ	グローバル	いいえ
説明: 正常に肯定応答されたコミットの数 Rpl_semi_sync_slave_status		
いいえ	いいえ	はい
いいえ	グローバル	いいえ
説明: 準同期レプリケーションがスレーブ上で動作しているかどうか slave_exec_mode		
はい	はい	いいえ
はい	グローバル	はい
説明: スレーブスレッドを IDEMPOTENT モード (重大なおよびほかのいくつかのエラーが抑止されます) と STRICT モードの間で切り替えることを許可します。IDEMPOTENT が常に使用される MySQL Cluster を除いて、STRICT モードがデフォルトです。 Slave_open_temp_tables		
いいえ	いいえ	はい
いいえ	グローバル	いいえ
説明: スレーブ SQL スレッドが現在開いている一時テーブルの数 Slave_retried_transactions		
いいえ	いいえ	はい
いいえ	グローバル	いいえ
説明: 起動以降、レプリケーションスレーブ SQL スレッドがトランザクションを再試行した合計回数 Slave_running		
いいえ	いいえ	はい
いいえ	グローバル	いいえ
説明: このサーバーのレプリケーションスレーブとしての状態 (スレーブ I/O スレッドステータス) abort-slave-event-count		
はい	いいえ	いいえ

オプションまたは変数名		
コマンド行	システム変数	ステータス変数
オプションファイル	スコープ	動的
メモ		
はい		いいえ
説明: mysql-test によってレプリケーションのデバッグとテストのために使用されるオプション binlog_gtid_recovery_simplified		
はい	はい	いいえ
はい	グローバル	いいえ
説明: GTID リカバリ中にバイナリログがどのように反復されるかを制御します disable-gtid-unsafe-statements		
はい	はい	いいえ
はい	グローバル	いいえ
説明: 廃止: MySQL 5.6.9 では、--enforce-gtid-consistency によって置き換えられます。 disable_gtid_unsafe_statements		
はい	はい	いいえ
はい	グローバル	いいえ
説明: 廃止: MySQL 5.6.9 では、enforce_gtid_consistency によって置き換えられます。 disconnect-slave-event-count		
はい	いいえ	いいえ
はい		いいえ
説明: mysql-test によってレプリケーションのデバッグとテストのために使用されるオプション enforce-gtid-consistency		
はい	はい	いいえ
はい	グローバル	いいえ
説明: トランザクションセーフな方法でログを記録できないステートメントの実行を回避します。これらには、トランザクション内の CREATE TEMPORARY TABLE、CREATE TABLE ... SELECT、トランザクションおよび非トランザクションの両方のテーブルを更新するトランザクションやステートメントなどがあります。 enforce_gtid_consistency		
はい	はい	いいえ
はい	グローバル	いいえ
説明: true のときは、トランザクションセーフな方法でログを記録できないため、ステートメントの実行は許可されません。このようなステートメントには、トランザクション内の CREATE TEMPORARY TABLE、CREATE TABLE ... SELECT、トランザクションおよび非トランザクションの両方のテーブルを更新するトランザクションやステートメントなどがあります。読み取り専用。サーバーの起動時に --disable-gtid-unsafe-statements を使用して設定します。 gtid_done		
いいえ	はい	いいえ
いいえ	両方	いいえ
説明: 廃止: MySQL 5.6.9 では、gtid_executed に置き換えられました。 gtid_executed		
いいえ	はい	いいえ
いいえ	両方	いいえ
説明: グローバル: バイナリログ（グローバル）または現在のトランザクション（セッション）内のすべての GTID。読み取り専用 gtid_lost		
いいえ	はい	いいえ

オプションまたは変数名		
コマンド行	システム変数	ステータス変数
オプションファイル	スコープ	動的
メモ		
いいえ	グローバル	いいえ
説明: 廃止: MySQL 5.6.9 では、gtid_purged に置き換えられました。		
gtid-mode		
はい	はい	いいえ
はい	グローバル	いいえ
説明: ON は GTID を有効にし、OFF は無効にします。UPGRADE_STEP_1 と UPGRADE_STEP_2 は現在サポートされていません。		
gtid_mode		
いいえ	はい	いいえ
いいえ	グローバル	いいえ
説明: GTID が有効かどうかを示します。読み取り専用		
gtid_next		
いいえ	はい	いいえ
いいえ	セッション	はい
説明: 次に実行するステートメントの GTID を指定します。詳細は、ドキュメントを参照してください。		
gtid_owned		
いいえ	はい	いいえ
いいえ	両方	いいえ
説明: 所有者（グローバル）のスレッド ID に加えて、このクライアント (セッション) またはすべてのクライアントによって所有される GTID セットです。読み取り専用		
gtid_purged		
いいえ	はい	いいえ
いいえ	グローバル	はい
説明: バイナリログからパージされたすべてのグループのセット。		
init_slave		
はい	はい	いいえ
はい	グローバル	はい
説明: スレーブがマスターに接続するときに実行されるステートメント		
log-slave-updates		
はい	はい	いいえ
はい	グローバル	いいえ
説明: このオプションは、スレーブ SQL スレッドが実行した更新のログを自身のバイナリログに記録するようにスレーブに指示します。		
log_slave_updates		
はい	はい	いいえ
はい	グローバル	いいえ
説明: スレーブがその SQL スレッドによって実行された更新のログを自身のバイナリログに記録するかどうかを指示します。読み取り専用。--log-slave-updates サーバーオプションを使用して設定します。		
master-info-file		
はい	いいえ	いいえ
はい		いいえ
説明: マスターを記憶するファイルの場所と名前、および I/O レプリケーションスレッドがマスターのバイナリログ内でどこにあるか		

オプションまたは変数名		
コマンド行	システム変数	ステータス変数
オプションファイル	スコープ	動的
メモ		
master-info-repository		
はい	いいえ	いいえ
はい		いいえ
説明: マスターのバイナリログ内のマスターステータス情報とレプリケーション I/O スレッド位置をファイルまたはテーブルに書き込むかどうか。		
master_info_repository		
はい	はい	いいえ
はい	グローバル	はい
説明: マスターのバイナリログ内のマスターステータス情報とレプリケーション I/O スレッド位置をファイルまたはテーブルに書き込むかどうか。		
master-retry-count		
はい	いいえ	いいえ
はい		いいえ
説明: スレーブがマスターへの接続を試行する回数 (このあとは停止)		
relay-log		
はい	はい	いいえ
はい	グローバル	いいえ
説明: リレーログに使用する場所とベース名		
relay_log_basename		
いいえ	はい	いいえ
いいえ	グローバル	いいえ
説明: リレーログへの完全パス（ファイル名を含む）		
relay-log-index		
はい	はい	いいえ
はい	グローバル	いいえ
説明: 最後のリレーログのリストを保持するファイルに使用する場所と名前		
relay-log-info-file		
はい	いいえ	いいえ
はい		いいえ
説明: SQL レプリケーションスレッドがリレーログ内のどこにあるかを記憶するファイルの場所と名前		
relay_log_info_file		
はい	はい	いいえ
はい	グローバル	いいえ
説明: スレーブがリレーログの情報を記録するファイルの名前。		
relay-log-info-repository		
はい	いいえ	いいえ
はい		いいえ
説明: レプリケーション SQL スレッドのリレーログ内での場所をファイルまたはテーブルに書き込むかどうか。		
relay_log_info_repository		
いいえ	はい	いいえ
いいえ	グローバル	はい

オプションまたは変数名		
コマンド行	システム変数	ステータス変数
オプションファイル	スコープ	動的
メモ		
説明: レプリケーション SQL スレッドのリレーログ内での場所をファイルまたはテーブルに書き込むかどうか。 relay_log_index		
はい	はい	いいえ
はい	グローバル	いいえ
説明: リレーログインデックスファイルの名前。 relay_log_purge		
はい	はい	いいえ
はい	グローバル	はい
説明: リレーログがパージされるかどうかを決定します relay-log-recovery		
はい	いいえ	いいえ
はい		いいえ
説明: 起動時のマスターからのリレーログファイル自動リカバリを有効にします relay_log_recovery		
はい	はい	いいえ
はい	グローバル	はい
説明: 起動時のマスターからのリレーログファイル自動リカバリが有効であるかどうか。クラッシュセーフスレーブにするには有効にする必要があります。 relay_log_space_limit		
はい	はい	いいえ
はい	グローバル	いいえ
説明: すべてのリレーログに使用する最大領域 replicate-do-db		
はい	いいえ	いいえ
はい		いいえ
説明: 指定されたデータベースにレプリケーションを制限するようにスレーブ SQL スレッドに指示します replicate-do-table		
はい	いいえ	いいえ
はい		いいえ
説明: 指定されたテーブルにレプリケーションを制限するようにスレーブ SQL スレッドに指示します replicate-ignore-db		
はい	いいえ	いいえ
はい		いいえ
説明: 指定されたデータベースに複製しないようにスレーブ SQL スレッドに指示します replicate-ignore-table		
はい	いいえ	いいえ
はい		いいえ
説明: 指定されたテーブルに複製しないようにスレーブ SQL スレッドに指示します replicate-rewrite-db		
はい	いいえ	いいえ
はい		いいえ

オプションまたは変数名		
コマンド行	システム変数	ステータス変数
オプションファイル	スコープ	動的
メモ		
説明: 元の名前とは異なる名前のデータベースに更新します replicate-same-server-id		
はい	いいえ	いいえ
はい		いいえ
説明: レプリケーションで、1 に設定されている場合は、同じサーバー ID を持つイベントをスキップしません replicate-wild-do-table		
はい	いいえ	いいえ
はい		いいえ
説明: 指定されたワイルドカードパターンに一致するテーブルにレプリケーションを制限するようにスレーブスレッドに指示します replicate-wild-ignore-table		
はい	いいえ	いいえ
はい		いいえ
説明: 指定されたワイルドカードパターンに一致するテーブルに複製しないようにスレーブスレッドに指示します report-host		
はい	はい	いいえ
はい	グローバル	いいえ
説明: スレーブ登録中にマスターに報告するスレーブのホスト名または IP report-password		
はい	はい	いいえ
はい	グローバル	いいえ
説明: スレーブサーバーがマスターに報告すべき任意パスワード。MySQL レプリケーションユーザーアカウント用のパスワードと同じではないもの report-port		
はい	はい	いいえ
はい	グローバル	いいえ
説明: スレーブ登録中にマスターに報告される、スレーブに接続するためのポート report-user		
はい	はい	いいえ
はい	グローバル	いいえ
説明: スレーブサーバーがマスターに報告すべき任意ユーザー名。MySQL レプリケーションユーザーアカウントで使用される名前と同じではないもの。 rpl_semi_sync_master_enabled		
いいえ	はい	いいえ
いいえ	グローバル	はい
説明: 準同期レプリケーションがマスターで有効であるかどうか rpl_semi_sync_master_timeout		
いいえ	はい	いいえ
いいえ	グローバル	はい
説明: スレーブ肯定応答を待機するミリ秒 rpl_semi_sync_master_trace_level		
いいえ	はい	いいえ

オプションまたは変数名		
コマンド行	システム変数	ステータス変数
オプションファイル	スコープ	動的
メモ		
いいえ	グローバル	はい
説明: マスターでの準同期レプリケーションデバッグトレースレベル rpl_semi_sync_master_wait_no_slave		
いいえ	はい	いいえ
いいえ	グローバル	はい
説明: マスターがタイムアウトを待機するかどうか (スレーブなしでも可) rpl_semi_sync_slave_enabled		
いいえ	はい	いいえ
いいえ	グローバル	はい
説明: 準同期レプリケーションがスレーブで有効かどうか rpl_semi_sync_slave_trace_level		
いいえ	はい	いいえ
いいえ	グローバル	はい
説明: スレーブでの準同期レプリケーションデバッグトレースレベル rpl_stop_slave_timeout		
はい	はい	いいえ
はい	グローバル	はい
説明: STOP SLAVE が待機する秒数を設定します (このあとはタイムアウト)。 server_uuid		
いいえ	はい	いいえ
いいえ	グローバル	いいえ
説明: サーバーのグローバル一意ID、サーバー起動時に自動的に生成 （ 再生成 ） show-slave-auth-info		
はい	いいえ	いいえ
はい		いいえ
説明: このマスターでの SHOW SLAVE HOSTS でユーザー名とパスワードを表示します simplified_binlog_gtid_recovery		
はい	はい	いいえ
はい	グローバル	いいえ
説明: GTID リカバリ中にバイナリログがどのように反復されるかを制御します skip-slave-start		
はい	いいえ	いいえ
はい		いいえ
説明: 設定されている場合は、スレーブは自動起動されません slave-load-tmpdir		
はい	はい	いいえ
はい	グローバル	いいえ
説明: スレーブが LOAD DATA INFILE ステートメントを複製するときにその一次ファイルを置くべき場所 slave-skip-errors		
はい	はい	いいえ
はい	グローバル	いいえ

オプションまたは変数名		
コマンド行	システム変数	ステータス変数
オプションファイル	スコープ	動的
メモ		
説明: 提供されたリストからクエリーがエラーを返すときでもレプリケーションを継続するようにスレーブスレッドに指示します slave-checkpoint-group		
はい	いいえ	いいえ
はい		いいえ
説明: 進捗状況ステータスを更新するためにチェックポイント操作が呼び出される前に、マルチスレッドスレーブによって処理される最大トランザクション数。 slave_checkpoint_group		
はい	はい	いいえ
はい	グローバル	はい
説明: 進捗状況ステータスを更新するためにチェックポイント操作が呼び出される前に、マルチスレッドスレーブによって処理される最大トランザクション数。 slave-checkpoint-period		
はい	いいえ	いいえ
はい		いいえ
説明: このミリ秒後に、マルチスレッドスレーブの進捗状況ステータスを更新し、リレーログ情報をディスクにフラッシュします。 slave_checkpoint_period		
はい	はい	いいえ
はい	グローバル	はい
説明: このミリ秒後に、マルチスレッドスレーブの進捗状況ステータスを更新し、リレーログ情報をディスクにフラッシュします。 slave_compressed_protocol		
はい	はい	いいえ
はい	グローバル	はい
説明: マスター/スレーブプロトコルで圧縮を使用します slave-max-allowed-packet		
はい	いいえ	いいえ
はい		いいえ
説明: レプリケーションマスターからスレーブに送信できるパケットの最大サイズ（バイト単位）。max_allowed_packet をオーバーライドします。 slave_max_allowed_packet		
いいえ	はい	いいえ
いいえ	グローバル	はい
説明: レプリケーションマスターからスレーブに送信できるパケットの最大サイズ（バイト単位）。max_allowed_packet をオーバーライドします。 slave_net_timeout		
はい	はい	いいえ
はい	グローバル	はい
説明: マスター/スレーブ接続から後続のデータを待機する秒数 (このあとは読み取りを中止) slave_parallel_workers		
いいえ	はい	いいえ
いいえ	グローバル	はい

オプションまたは変数名		
コマンド行	システム変数	ステータス変数
オプションファイル	スコープ	動的
メモ		
説明: イベントを並列実行するためのワーカースレッドの数。0 (デフォルト) に設定すると、スレーブマルチスレッドが無効になります。 slave-parallel-workers		
はい	いいえ	いいえ
はい		いいえ
説明: イベントを並列実行するためのワーカースレッドの数。0 (デフォルト) に設定すると、スレーブマルチスレッドが無効になります。 slave-pending-jobs-size-max		
はい	いいえ	いいえ
いいえ		いいえ
説明: まだ適用されていないイベントを保持するスレーブワーカーキューの最大サイズ。 slave_pending_jobs_size_max		
いいえ	はい	いいえ
いいえ	グローバル	はい
説明: まだ適用されていないイベントを保持するスレーブワーカーキューの最大サイズ。 slave-rows-search-algorithms		
はい	いいえ	いいえ
はい		いいえ
説明: スレーブ更新バッチに使用される検索アルゴリズムを決定します。リスト INDEX_SEARCH、TABLE_SCAN、HASH_SCAN からの任意の 2 つまたは 3 つ。デフォルトは TABLE_SCAN、INDEX_SCAN。 slave_rows_search_algorithms		
いいえ	はい	いいえ
いいえ	グローバル	はい
説明: スレーブ更新バッチに使用される検索アルゴリズムを決定します。リスト INDEX_SEARCH、TABLE_SCAN、HASH_SCAN からの任意の 2 つまたは 3 つ。デフォルトは TABLE_SCAN、INDEX_SCAN。 slave_transaction_retries		
はい	はい	いいえ
はい	グローバル	はい
説明: スレーブ SQL スレッドがデッドロックまたはロック待機タイムアウト経過によって失敗した場合にトランザクションを再試行する回数 (このあとは中止して停止) slave_type_conversions		
はい	はい	いいえ
はい	グローバル	いいえ
説明: レプリケーションスレーブでの型変換モードを制御します。値は、リスト ALL_LOSSY、ALL_NON_LOSSY からのゼロ個以上の要素のリストです。空の文字列に設定すると、マスターとスレーブ間の型変換を禁止します。 sql_slave_skip_counter		
いいえ	はい	いいえ
いいえ	グローバル	はい
説明: スレーブサーバーがスキップすべき、マスターからのイベントの数。GTID レプリケーションと互換性はありません。 sync_binlog		

オプションまたは変数名		
コマンド行	システム変数	ステータス変数
オプションファイル	スコープ	動的
メモ		
はい	はい	いいえ
はい	グローバル	はい
説明: # 番目のイベントごとに同期的にバイナリログをディスクにフラッシュします		
sync_master_info		
はい	はい	いいえ
はい	グローバル	はい
説明: # 番目のイベントごとに master.info をディスクに同期します。		
sync_relay_log		
はい	はい	いいえ
はい	グローバル	はい
説明: # 番目のイベントごとにリレーログをディスクに同期します。		
sync_relay_log_info		
はい	はい	いいえ
はい	グローバル	はい
説明: # 番目のイベントごとに relay.info ファイルをディスクに同期します。		

セクション17.1.4.2「レプリケーションマスターのオプションと変数」は、レプリケーションマスターサーバーに関係するオプションと変数に関する詳細情報を提供します。レプリケーションスレーブに関係するオプションと変数に関する詳細情報については、セクション17.1.4.3「レプリケーションスレーブのオプションと変数」を参照してください。

表 17.2 MySQL 5.6 でのバイナリロギングのオプションおよび変数のサマリー

オプションまたは変数名		
コマンド行	システム変数	ステータス変数
オプションファイル	スコープ	動的
メモ		
Binlog_cache_disk_use		
いいえ	いいえ	はい
いいえ	グローバル	いいえ
説明: バイナリログキャッシュの代わりに一時ファイルを使用したトランザクションの数		
binlog_row_image		
はい	はい	いいえ
はい	両方	はい
説明: 行変更のログを記録するときに full または minimal イメージを使用します。許容される値は full、minimal、および noblob です。		
binlog_rows_query_log_events		
いいえ	はい	いいえ
いいえ	両方	はい
説明: TRUE のとき、行ベースロギングモードで行クエリーログイベントのロギングを有効にします。デフォルトは FALSE です。5.6.2 より前のレプリケーションスレーブまたはほかのリーダーのログを生成するときは有効にしないでください。		
Binlog_stmt_cache_disk_use		
いいえ	いいえ	はい
いいえ	グローバル	いいえ

オプションまたは変数名		
コマンド行	システム変数	ステータス変数
オプションファイル	スコープ	動的
メモ		
説明: バイナリログステートメントキャッシュの代わりに一時ファイルを使用した非トランザクションステートメントの数 Binlog_cache_use		
いいえ	いいえ	はい
いいえ	グローバル	いいえ
説明: 一時バイナリログキャッシュを使用したトランザクションの数 Binlog_stmt_cache_use		
いいえ	いいえ	はい
いいえ	グローバル	いいえ
説明: 一時バイナリログステートメントキャッシュを使用したステートメントの数 Com_show_binlog_events		
いいえ	いいえ	はい
いいえ	両方	いいえ
説明: SHOW BINLOG EVENTS ステートメントの数 Com_show_binlogs		
いいえ	いいえ	はい
いいえ	両方	いいえ
SHOW BINLOGS ステートメントの数 binlog-checksum		
はい	いいえ	いいえ
はい		いいえ
説明: バイナリログチェックサムを有効化/無効化します binlog_checksum		
いいえ	はい	いいえ
いいえ	グローバル	はい
説明: バイナリログチェックサムを有効化/無効化します binlog-do-db		
はい	いいえ	いいえ
はい		いいえ
説明: バイナリロギングを特定のデータベースに制限します binlog_error_action		
はい	はい	いいえ
はい	両方	はい
説明: サーバーがバイナリログに書き込めないときに何が起きるかを制御します。2 つのいずれかの値を想定できます: IGNORE_ERROR (デフォルト) は、サーバーはエラーログを記録しますが、更新処理を継続することを意味します。ABORT_SERVER の場合は、バイナリログを書き込めないときにサーバーがシャットダウンします。 binlog-ignore-db		
はい	いいえ	いいえ
はい		いいえ
説明: 指定されたデータベースへの更新をバイナリログにログを記録すべきでないことをマスターに指示します binlogging_impossible_mode		
はい	はい	いいえ

オプションまたは変数名		
コマンド行	システム変数	ステータス変数
オプションファイル	スコープ	動的
メモ		
はい	両方	はい
説明: 非推奨であり、将来のバージョンで削除されます。代わりに、名前が変更された binlog_error_action を使用します。		
binlog-row-event-max-size		
はい	いいえ	いいえ
はい		いいえ
説明: バイナリログ最大イベントサイズ		
binlog_cache_size		
はい	はい	いいえ
はい	グローバル	はい
説明: トランザクション中にバイナリログのために SQL ステートメントを保持するキャッシュのサイズ		
binlog_max_flush_queue_time		
いいえ	はい	いいえ
いいえ	グローバル	はい
説明: バイナリログにフラッシュする前にどのくらいトランザクションを読み取るか		
binlog_order_commits		
いいえ	はい	いいえ
いいえ	グローバル	はい
説明: バイナリログへの書き込みと同じ順序でコミットするかどうか		
binlog_stmt_cache_size		
はい	はい	いいえ
はい	グローバル	はい
説明: トランザクション中にバイナリログのために非トランザクションステートメントを保持するキャッシュのサイズ		
binlog_format		
はい	はい	いいえ
はい	両方	はい
説明: バイナリログの形式を指定します		
binlog-rows-query-log-events		
はい	いいえ	いいえ
はい		いいえ
説明: 行ベースロギングを使用するときに行クエリーログイベントのロギングを有効にします。デフォルトで無効になっています。5.6.2 より前のスレーブ/リーダーにログを生成するときは有効にしないでください。		
binlog_direct_non_transactional_updates		
はい	はい	いいえ
はい	両方	はい
説明: 非トランザクションエンジンにステートメント形式を使用する更新が直接バイナリログに書き込まれます。使用する前にドキュメントを参照してください。		
log_bin_basename		
いいえ	はい	いいえ
いいえ	グローバル	いいえ
説明: バイナリログへの完全パス（ファイル名を含む）		
log-bin-use-v1-row-events		

オプションまたは変数名		
コマンド行	システム変数	ステータス変数
オプションファイル	スコープ	動的
メモ		
はい	はい	いいえ
はい	グローバル	いいえ
説明: バージョン 1 バイナリログ行イベントを使用します log_bin_use_v1_row_events		
はい	はい	いいえ
はい	グローバル	いいえ
説明: サーバーがバージョン 1 バイナリログ行イベントを使用しているかどうか master-verify-checksum		
はい	いいえ	いいえ
はい		いいえ
説明: マスターがバイナリログから読み取るときにチェックサムを検査します。 master_verify_checksum		
いいえ	はい	いいえ
いいえ	グローバル	はい
説明: マスターがバイナリログからチェックサムを読み取ります。 max-binlog-dump-events		
はい	いいえ	いいえ
はい		いいえ
説明: mysql-test によってレプリケーションのデバッグとテストのために使用されるオプション max_binlog_cache_size		
はい	はい	いいえ
はい	グローバル	はい
説明: マルチステートメントトランザクションをキャッシュするために使用される合計サイズを制限するために使用できます max_binlog_size		
はい	はい	いいえ
はい	グローバル	はい
説明: サイズがこの値を超えたときにバイナリログが自動的にローテーションします。 max_binlog_stmt_cache_size		
はい	はい	いいえ
はい	グローバル	はい
説明: トランザクション中にすべての非トランザクションステートメントをキャッシュするために使用される合計サイズを制限するために使用できます slave-sql-verify-checksum		
はい	いいえ	いいえ
はい		いいえ
説明: スレーブがリレーログから読み取るときにチェックサムを検査します slave_sql_verify_checksum		
いいえ	はい	いいえ
いいえ	グローバル	はい
説明: スレーブがリレーログから読み取るときにチェックサムを検査します。 sporadic-binlog-dump-fail		

オプションまたは変数名		
コマンド行	システム変数	ステータス変数
オプションファイル	スコープ	動的
メモ		
はい	いいえ	いいえ
はい		いいえ
説明: mysql-test によってレプリケーションのデバッグとテストのために使用されるオプション		

セクション17.1.4.4「バイナリログのオプションと変数」は、バイナリロギングに関係するオプションと変数に関する詳細情報を提供します。バイナリログに関するその他の一般情報については、セクション5.2.4「バイナリログ」を参照してください。

sql_log_bin および sql_log_off 変数については、セクション5.1.4「サーバーシステム変数」を参照してください。

mysqld で使用されるすべてのコマンド行オプション、システム変数、およびステータス変数を示す表については、セクション5.1.1「サーバーオプションおよび変数リファレンス」を参照してください。

17.1.4.2 レプリケーションマスターのオプションと変数

このセクションでは、レプリケーションマスターサーバーで使用できるサーバーオプションとシステム変数について説明します。オプションはコマンド行またはオプションファイルで指定できます。システム変数値はSET を使用して指定できます。

マスターと各スレーブでは、server-id オプションを使用して一意レプリケーション ID を確立する必要があります。サーバーごとに、一意の正の整数を 1 から $2^{32} - 1$ の範囲から選択してください。各 ID は、ほかのレプリケーションのマスターまたはスレーブが使用するほかのあらゆる ID と異なっている必要があります。例: server-id=3。

バイナリロギングを制御するためにマスターで使用されるオプションについては、セクション17.1.4.4「バイナリログのオプションと変数」を参照してください。

レプリケーションマスターで使用されるシステム変数

次のシステム変数は、レプリケーションマスターを制御するために使用されます。

- auto_increment_increment

システム変数	auto_increment_increment
スコープ	グローバル、セッション
動的	はい
型	数値
デフォルト	1
最小値	1
最大値	65535

auto_increment_increment および auto_increment_offset は、マスターからマスターへのレプリケーションで使用するためのもので、AUTO_INCREMENT カラムの操作を制御するために使用できます。両方の変数はグローバル値とセッション値を持ち、各値は 1 から 65,535 (1 と 65,535 を含みます) の間の整数値を取ることができます。これらの 2 つの変数のいずれかの値を 0 に設定すると、代わりにその値は 1 に設定されます。これらの 2 つの変数のいずれかの値を 65,535 より大きな整数または 0 より小さい整数に設定しようとすると、代わりにその値は 65,535 に設定されます。auto_increment_increment または auto_increment_offset の値を整数でない値に設定しようとすると、エラーが発生し、変数の実際の値は変化しません。

> **注記**
>
> auto_increment_increment は NDB テーブルで使用する場合にもサポートされます。

これら 2 つの変数は、次のように AUTO_INCREMENT カラムの動作に影響します。

- auto_increment_increment は、連続するカラム値の間隔を制御します。例:

```
mysql> SHOW VARIABLES LIKE 'auto_inc%';
+--------------------------+-------+
```

```
| Variable_name            | Value |
+--------------------------+-------+
| auto_increment_increment | 1     |
| auto_increment_offset    | 1     |
+--------------------------+-------+
2 rows in set (0.00 sec)

mysql> CREATE TABLE autoinc1
    -> (col INT NOT NULL AUTO_INCREMENT PRIMARY KEY);
  Query OK, 0 rows affected (0.04 sec)

mysql> SET @@auto_increment_increment=10;
Query OK, 0 rows affected (0.00 sec)

mysql> SHOW VARIABLES LIKE 'auto_inc%';
+--------------------------+-------+
| Variable_name            | Value |
+--------------------------+-------+
| auto_increment_increment | 10    |
| auto_increment_offset    | 1     |
+--------------------------+-------+
2 rows in set (0.01 sec)

mysql> INSERT INTO autoinc1 VALUES (NULL), (NULL), (NULL), (NULL);
Query OK, 4 rows affected (0.00 sec)
Records: 4  Duplicates: 0  Warnings: 0

mysql> SELECT col FROM autoinc1;
+-----+
| col |
+-----+
|   1 |
|  11 |
|  21 |
|  31 |
+-----+
4 rows in set (0.00 sec)
```

- auto_increment_offset は AUTO_INCREMENT カラム値の開始点を指定します。次のことは、auto_increment_increment の記述で示した例のように、同じセッション中にこれらのステートメントが実行されるものと仮定しています。

```
mysql> SET @@auto_increment_offset=5;
Query OK, 0 rows affected (0.00 sec)

mysql> SHOW VARIABLES LIKE 'auto_inc%';
+--------------------------+-------+
| Variable_name            | Value |
+--------------------------+-------+
| auto_increment_increment | 10    |
| auto_increment_offset    | 5     |
+--------------------------+-------+
2 rows in set (0.00 sec)

mysql> CREATE TABLE autoinc2
    -> (col INT NOT NULL AUTO_INCREMENT PRIMARY KEY);
Query OK, 0 rows affected (0.06 sec)

mysql> INSERT INTO autoinc2 VALUES (NULL), (NULL), (NULL), (NULL);
Query OK, 4 rows affected (0.00 sec)
Records: 4  Duplicates: 0  Warnings: 0

mysql> SELECT col FROM autoinc2;
+-----+
| col |
+-----+
|   5 |
|  15 |
|  25 |
|  35 |
+-----+
4 rows in set (0.02 sec)
```

  auto_increment_offset の値が auto_increment_increment の値よりも大きいと、auto_increment_offset の値は無視されます。

これらの変数のいずれかが変更されてから、AUTO_INCREMENT カラムを含むテーブルに新しい行が挿入される場合、結果は反直感的に見える場合があります。AUTO_INCREMENT 値のシリーズがカラムにすでに存在す

る値に関係なく計算され、挿入される次の値が AUTO_INCREMENT カラムに存在する最大値よりも大きなシリーズ内最小値であるためです。シリーズは次のように計算されます。

auto_increment_offset + N × auto_increment_increment

ここで、N はシリーズ内正の整数値 [1, 2, 3, ...] です。例:

```
mysql> SHOW VARIABLES LIKE 'auto_inc%';
+--------------------------+-------+
| Variable_name            | Value |
+--------------------------+-------+
| auto_increment_increment | 10    |
| auto_increment_offset    | 5     |
+--------------------------+-------+
2 rows in set (0.00 sec)

mysql> SELECT col FROM autoinc1;
+-----+
| col |
+-----+
|   1 |
|  11 |
|  21 |
|  31 |
+-----+
4 rows in set (0.00 sec)

mysql> INSERT INTO autoinc1 VALUES (NULL), (NULL), (NULL), (NULL);
Query OK, 4 rows affected (0.00 sec)
Records: 4  Duplicates: 0  Warnings: 0

mysql> SELECT col FROM autoinc1;
+-----+
| col |
+-----+
|   1 |
|  11 |
|  21 |
|  31 |
|  35 |
|  45 |
|  55 |
|  65 |
+-----+
8 rows in set (0.00 sec)
```

auto_increment_increment と auto_increment_offset に示される値は、シリーズ 5 + N × 10、つまり [5, 15, 25, 35, 45, ...] を生成します。INSERT より前に col カラムに存在する最高値は 31、AUTO_INCREMENT シリーズ内で次に使用できる値は 35 なので、col に挿入される値はそのポイントで始まり、結果は SELECT クエリーで示されるようになります。

これらの 2 つの変数の影響を単一テーブルに制限することはできません。これらの変数は MySQL サーバーのすべてのテーブルのすべての AUTO_INCREMENT カラムの動作を制御します。どちらかの変数のグローバル値が設定されると、グローバル値が変更されるか、セッション値の設定によってオーバーライドされるまで、または mysqld が再起動されるまでその効果は持続します。ローカル値が設定されると、新しい値は、セッションの期間に現在のユーザーが新しい行を挿入したすべてのテーブルの AUTO_INCREMENT カラムに影響します (そのセッション中にそれらの値が変更される場合を除く)。

auto_increment_increment のデフォルト値は 1 です。セクション17.4.1.1「レプリケーションと AUTO_INCREMENT」を参照してください。

- auto_increment_offset

システム変数	auto_increment_offset
スコープ	グローバル、セッション
動的	はい
型	数値
デフォルト	1
最小値	1

最大値	65535

この変数のデフォルト値は 1 です。詳細については、auto_increment_increment の説明を参照してください。

> **注記**
>
> auto_increment_offset は NDB テーブルで使用する場合にもサポートされます。

17.1.4.3 レプリケーションスレーブのオプションと変数

レプリケーションスレーブの起動オプション

スレーブステータスログをテーブルに記録するためのオプション

廃止されたレプリケーションスレーブオプション

レプリケーションスレーブで使用されるシステム変数

オプションはコマンド行またはオプションファイルで指定します。多くのオプションは、サーバーの動作中に CHANGE MASTER TO ステートメントを使用して設定できます。システム変数値は SET を使用して指定します。

サーバー ID　マスターと各スレーブでは、server-id オプションを使用して、範囲が 1 から $2^{32} - 1$ の一意レプリケーション ID を確立する必要があります。「一意」とは、各 ID が、ほかのレプリケーションマスターまたはスレーブで使用されるほかのあらゆる ID とは異なっている必要があるということです。my.cnf ファイルの例:

```
[mysqld]
server-id=3
```

レプリケーションスレーブの起動オプション

このセクションでは、レプリケーションスレーブサーバーを制御するための起動オプションについて説明します。これらのオプションの多くは、サーバーの動作中に CHANGE MASTER TO ステートメントを使用して設定できます。--replicate-* などのその他のオプションは、スレーブサーバーが起動するときにのみ設定できます。レプリケーションに関連するシステム変数はこのセクションの後半で説明します。

- --abort-slave-event-count

コマンド行形式	--abort-slave-event-count=#
型	数値
デフォルト	0
最小値	0

このオプションが 0 (デフォルト) 以外の正の整数値に設定されると、次のようにレプリケーションの動作に影響します。スレーブ SQL スレッドが起動したあと、値ログイベントの実行が許可され、スレーブ SQL スレッドはマスターからのネットワーク接続が切断されたかのようにそれ以上イベントを受け取りません。スレーブのスレッドの実行は継続し、SHOW SLAVE STATUS からの出力には Slave_IO_Running および Slave_SQL_Running カラムの両方に Yes が表示されますが、それ以降のイベントはリレーログから読み取られません。

このオプションは、レプリケーションのテストとデバッグのために、MySQL テストスイートで内部的に使用されます。本番環境設定で使用することを想定していません。

- --disconnect-slave-event-count

コマンド行形式	--disconnect-slave-event-count=#
型	数値
デフォルト	0

このオプションは、レプリケーションのテストとデバッグのために、MySQL テストスイートで内部的に使用されます。

- --log-slave-updates

コマンド行形式	--log-slave-updates

システム変数	log_slave_updates
スコープ	グローバル
動的	いいえ
型	ブール
デフォルト	OFF

通常、スレーブはマスターサーバーから受け取った更新をそれ自身のバイナリログに書き込みません。このオプションによって、スレーブはその SQL スレッドによって実行される更新をそれ自身のバイナリログに書き込みます。このオプションを有効にするには、--log-bin オプションを使用してスレーブを起動してバイナリロギングを有効にする必要もあります。MySQL 5.5 より前では、--log-slave-updates オプションを使用しても、--log-bin オプションでサーバーを起動することなくサーバーは起動せず、エラーで失敗します。MySQL 5.6 では、警告が生成されるだけです。(Bug #44663) --log-slave-updates はレプリケーションサーバーをチェーンするときに使用されます。たとえば、このようにレプリケーションサーバーをセットアップするとします。

```
A -> B -> C
```

ここでは、A はスレーブ B のマスターとして機能し、B はスレーブ C のマスターとして機能します。これが機能するには、B はマスターかつスレーブである必要があります。バイナリロギングを有効にするために A と B の両方を --log-bin で起動し、A から受け取った更新が B によってそのバイナリログに記録されるように B を --log-slave-updates オプションで起動する必要があります。

- --log-slow-slave-statements

コマンド行形式	--log-slow-slave-statements (≤ 5.6.10)
削除	5.6.11
型	ブール
デフォルト	OFF

スロークエリーログが有効化されている場合、このオプションはスレーブでの実行に long_query_time 秒を超える時間がかかったクエリーのロギングを有効にします。

このコマンド行オプションは MySQL 5.6.11 で削除され、log_slow_slave_statements システム変数によって置き換えられました。システム変数はオプションと同じ方法でコマンド行またはオプションファイルに設定できるため、サーバー起動時に何らかの変更を行う必要はありませんが、システム変数は実行時に値を検査または設定することも可能です。

- --log-warnings[=level]

コマンド行形式	--log-warnings[=#]
システム変数	log_warnings
スコープ (≥ 5.6.4)	グローバル
スコープ (≤ 5.6.3)	グローバル、セッション
動的	はい
型	数値
デフォルト	1
最小値	0
最大値 (64 ビットプラットフォーム)	18446744073709551615
最大値 (32 ビットプラットフォーム)	4294967295

サーバーは、それが実行していることに関するメッセージをより多くエラーログに記録します。レプリケーションに関して、サーバーは、それがネットワークまたは接続障害後の再接続に成功したことの警告を生成し、各スレーブスレッドがどのように起動したかに関する情報を提供します。このオプションはデフォルトでは (1) が有効です。無効にするには、--log-warnings=0 を使用します。値が 1 より大きい場合、中止された接続

がエラーログに書き込まれ、新しい接続の試行についてのアクセス拒否エラーが書き込まれます。セクション B.5.2.11「通信エラーおよび中止された接続」を参照してください。

> **注記**
>
> このオプションの影響はレプリケーションに制限されません。サーバーアクティビティー全体を対象として警告を生成します。

- --master-info-file=file_name

コマンド行形式	--master-info-file=file_name
型	ファイル名
デフォルト	master.info

マスターの情報をスレーブが記録するファイルに使用する名前。データディレクトリ内のデフォルト名は master.info です。このファイルの形式については、セクション17.2.2.2「スレーブステータスログ」を参照してください。

- --master-retry-count=count

コマンド行形式	--master-retry-count=#
非推奨	5.6.1
型	数値
デフォルト	86400
最小値	0
最大値 (64 ビットプラットフォーム)	18446744073709551615
最大値 (32 ビットプラットフォーム)	4294967295

スレーブマスターに接続を試みる回数 (これを超えると中断)。再接続は、CHANGE MASTER TO ステートメントの MASTER_CONNECT_RETRY オプションによって設定された間隔で (デフォルトは 60) で試行されます。再接続は、--slave-net-timeout オプションに応じてスレーブによるデータ読み取りがタイムアウトしたときにトリガーされます。デフォルト値は 86400 です。値 0 は「永続」を意味し、スレーブは永久に接続を試みます。

このオプションは MySQL 5.6.1 以降で非推奨となり、将来の MySQL リリースで削除される予定です。代わりに、CHANGE MASTER TO ステートメントの MASTER_RETRY_COUNT オプションを使用するように、アプリケーションを更新してください。

- --max-relay-log-size=size

コマンド行形式	--max-relay-log-size=#
システム変数	max_relay_log_size
スコープ	グローバル
動的	はい
型	数値
デフォルト	0
最小値	0
最大値	1073741824

このサイズで、サーバーはリレーログファイルを自動的にローテーションします。この値がゼロでない場合は、サイズがこの値を超えたときにリレーログは自動的にローテーションされます。この値がゼロ (デフォルト) の場合、リレーログローテーションが発生するサイズは max_binlog_size の値によって決められます。詳細については、セクション17.2.2.1「スレーブリレーログ」を参照してください。

- --read-only

コマンド行形式	--read-only
システム変数	read_only

スコープ	グローバル
動的	はい
型	ブール
デフォルト	false

スレーブがスレーブスレッドまたは SUPER 権限を持つユーザー以外からの更新を許可しなくなります。スレーブサーバーで、スレーブがクライアントからは受け付けず、確実にそのマスターサーバーからのみ更新を受け付けるようにするには、これが役立つ場合があります。この変数は TEMPORARY テーブルには適用されません。

- --relay-log=file_name

コマンド行形式	--relay-log=file_name
システム変数	relay_log
スコープ	グローバル
動的	いいえ
型	ファイル名

リレーログのベース名。デフォルトベース名は host_name-relay-bin です。別のディレクトリを指定するために先頭に絶対パス名を付けたベース名が指定されないかぎり、サーバーはデータディレクトリにファイルを書き込みます。サーバーは、ベース名に数値サフィクスを追加することで、順番にリレーログファイルを作成します。

MySQL がサーバーオプションを解析する方法が原因で、このオプションを指定する場合は、値を指定する必要があります。デフォルトベース名はオプションが実際に指定されない場合にのみ使用されます。値を指定しないで --relay-log オプションを使用する場合、予期しない動作になる場合があります。この動作は、使用されるほかのオプション、それらが指定される順序、およびそれらがコマンド行でまたはオプションファイルのどちらで指定されたかに依存します。MySQL がサーバーオプションをどのように処理するかについて詳しくは、セクション4.2.3「プログラムオプションの指定」を参照してください。

このオプションを指定した場合、指定された値はリレーログインデックスファイルのベース名としても使用されます。--relay-log-index オプションを使用して別のリレーログインデックスファイルを指定することで、この動作をオーバーライドできます。

MySQL 5.6.5 以降は、サーバーがインデックスファイルからエントリを読み取るときに、エントリに相対パスが含まれるかどうかをチェックします。その場合、パスの相対部は --relay-log オプションを使用して設定された絶対パスに置き換わります。絶対パスは変わりません。このような場合、使用される新しいパスを有効にするために、インデックスを手動で編集する必要があります。MySQL 5.6.5 より前は、バイナリログまたはリレーログファイルの位置を変更するときは手動介入が必要でした。(Bug #11745230、Bug #12133)

次のタスクを実行するときに、--relay-log オプションが役立つ場合があります。

- 名前がホスト名に依存しないリレーログを作成する。
- リレーログが非常に大きくなる傾向があり、max_relay_log_size を小さくしたくないため、リレーログをデータディレクトリ以外の領域に置く必要がある場合。
- ディスク間のロードバランシングを使用して速度を上げるため。

MySQL 5.6.2 以降では、リレーログファイル名 (およびパス) を relay_log_basename システム変数から取得できます。

- --relay-log-index=file_name

コマンド行形式	--relay-log-index=file_name
システム変数	relay_log_index
スコープ	グローバル
動的	いいえ

型	ファイル名

リレーログインデックスファイルに使用する名前。データディレクトリ内のデフォルト名は host_name-relay-bin.index です。ここで、host_name はスレーブサーバーの名前です。

MySQL がサーバーオプションを解析する方法が原因で、このオプションを指定する場合は、値を指定する必要があります。デフォルトベース名はオプションが実際に指定されない場合にのみ使用されます。値を指定しないで --relay-log-index オプションを使用する場合、予期しない動作になる場合があります。この動作は、使用されるほかのオプション、それらが指定される順序、およびオプションがコマンド行またはオプションファイルのどちらで指定されたに依存します。MySQL がサーバーオプションをどのように処理するかについて詳しくは、セクション4.2.3「プログラムオプションの指定」を参照してください。

このオプションを指定した場合、指定される値はリレーログのベース名としても使用されます。--relay-log オプションを使用して別のリレーログファイルベース名を指定することで、この動作をオーバーライドできます。

- --relay-log-info-file=file_name

コマンド行形式	--relay-log-info-file=file_name
型	ファイル名
デフォルト	relay-log.info

スレーブがリレーログの情報を記録するファイルに使用する名前。データディレクトリ内のデフォルト名は relay-log.info です。このファイルの形式については、セクション17.2.2.2「スレーブステータスログ」を参照してください。

- --relay-log-purge={0|1}

コマンド行形式	--relay-log-purge
システム変数	relay_log_purge
スコープ	グローバル
動的	はい
型	ブール
デフォルト	TRUE

リレーログファイルが不要になるとすぐに自動的にパージすることを無効または有効にします。デフォルト値は 1 (有効)です。これは SET GLOBAL relay_log_purge = N で動的に変更できるグローバル変数です。--relay-log-recovery オプションを使用するときにリレーログのパージを無効にすると、データー貫性が危険にさらされるため、クラッシュセーフではありません。

- --relay-log-recovery={0|1}

コマンド行形式	--relay-log-recovery
型	ブール
デフォルト	FALSE

サーバー起動直後のリレーログ自動リカバリを有効にします。リカバリプロセスでは、新しいリレーログファイルを作成し、SQL スレッド位置をこの新しいリレーログに初期化し、I/O スレッドを SQL スレッド位置に初期化します。その後、マスターからのリレーログ読み取りが続行されます。これは、破損した可能性のあるリレーログが処理されないことを保証するために、レプリケーションスレーブでクラッシュ後に使用してください。デフォルト値は 0 (無効)です。

クラッシュセーフなスレーブを提供するには、このオプションを有効にし (1 に設定)、--relay-log-info-repository を TABLE に設定し、relay-log-purge を有効にする必要があります。relay-log-purge が無効なときに --relay-log-recovery オプションを有効にすることで、パージされなかったファイルからリレーログを読み取り、データ矛盾が発生し、クラッシュセーフでなくなるリスクが抑止されます。詳細は、クラッシュセーフレプリケーションを参照してください。

MySQL 5.6.6 より前で、このオプションがマルチスレッドスレーブで有効の場合、スレーブはエラーで失敗し、そのスレーブで CHANGE MASTER TO を実行できません。MySQL 5.6.6 以降では、START SLAVE UNTIL SQL_AFTER_MTS_GAPS を使用してリレーログ内のギャップを処理できます。このステートメントを実行したあと、CHANGE MASTER TO を使用してこのスレーブを新しいマスターにフェイルオーバーできます。(Bug #13893363)

- --relay-log-space-limit=size

コマンド行形式	--relay-log-space-limit=#
システム変数	relay_log_space_limit
スコープ	グローバル
動的	いいえ
型	数値
デフォルト	0
最小値	0
最大値 (64 ビットプラットフォーム)	18446744073709551615
最大値 (32 ビットプラットフォーム)	4294967295

このオプションは、スレーブのすべてのリレーログの合計サイズ (バイト単位) の上限を設定します。値 0 は「制限なし」を表します。これは、ディスク領域が制限されているスレーブサーバーホストに役立ちます。制限に達した場合、SQL スレッドが追い付いて未使用のリレーログの一部を削除するまで、I/O スレッドはマスターサーバーからのバイナリログイベントの読み取りを停止します。この制限は絶対ではありません。SQL スレッドがリレーログを削除する前により多くのイベントを必要とする場合があります。その場合、SQL スレッドが一部のリレーログを削除できるようになるまで I/O スレッドは制限を超えます。そうしないとデッドロックになるためです。--relay-log-space-limit を --max-relay-log-size (または --max-relay-log-size が 0 の場合は --max-binlog-size) の値の 2 倍未満に設定しないでください。その場合、I/O スレッドが空き領域を待機する可能性があります。--relay-log-space-limit を超えたけれども、SQL スレッドはパージするリレーログを持たず、I/O スレッドを満たすことができないためです。この場合、I/O スレッドは強制的に --relay-log-space-limit を一時的に無視します。

- --replicate-do-db=db_name

コマンド行形式	--replicate-do-db=name
型	文字列

このオプションの影響は、ステートメントベースまたは行ベースのどちらのレプリケーションを使用中かによって異なります。

ステートメントベースのレプリケーション　デフォルトデータベース (つまり、USE で選択されたもの) が db_name であるステートメントにレプリケーションを限定するように、スレーブ SQL スレッドに指示します。複数のデータベースを指定するには、このオプションを複数回 (データベースごとに 1 回) 使用します。ただし、このようにすると別のデータベースは選択される (またはデータベースが選択されない) けれども、UPDATE some_db.some_table SET foo='bar' などのクロスデータベースステートメントを複製しません。

> **警告**
>
> 複数のデータベースを指定するには、このオプションの複数インスタンスを使用する必要があります。データベース名にカンマを含めることができるため、カンマで区切られたリストを指定すると、リストは単一データベースの名前として扱われます。

ステートメントベースレプリケーションを使用するときに期待するとおりに機能しない例: スレーブが --replicate-do-db=sales で起動され、マスターで次のステートメントを発行する場合、UPDATE ステートメントは複製されません。

```
USE prices;
UPDATE sales.january SET amount=amount+1000;
```

この「デフォルトデータベースだけをチェックする」動作の主な理由は、ステートメントだけから複製すべきかどうかを知るのが難しいためです (たとえば、複数のデータベースをまたがって動作する複数テーブルDELETE ステートメントまたは UPDATE ステートメントを使用する場合)。また、必要がない場合、すべてのデータベースではなくデフォルトデータベースだけをチェックする方が早いです。

行ベースのレプリケーション　データベース db_name にレプリケーションを制限するように、スレーブ SQL スレッドに指示します。db_name に属するテーブルだけが変更されます。現在のデータベースはこれに影響し

ません。スレーブが --replicate-do-db=sales で起動され、行ベースレプリケーションが有効である場合、次のステートメントがマスターで実行されます。

```
USE prices;
UPDATE sales.february SET amount=amount+100;
```

スレーブ上の sales データベース内の february テーブルが UPDATE ステートメントに従って変更されます。これは USE ステートメントが発行されたかどうかに関係なく発生します。ただし、行ベースレプリケーションと --replicate-do-db=sales の使用時に、次のステートメントをマスターで発行してもスレーブに影響はありません。

```
USE prices;
UPDATE prices.march SET amount=amount-25;
```

ステートメント USE prices が USE sales に変更された場合でも、UPDATE ステートメントの結果は複製されません。

--replicate-do-db が行ベースレプリケーションとステートメントベースレプリケーションでどのように扱われるかについてもう 1 つ重要な違いは、複数のデータベースを参照するステートメントで発生します。スレーブが --replicate-do-db=db1 で起動され、次のステートメントがマスターで実行されるものとします。

```
USE db1;
UPDATE db1.table1 SET col1 = 10, db2.table2 SET col2 = 20;
```

ステートメントベースレプリケーションを使用する場合、両方のテーブルがスレーブで更新されます。しかし、行ベースレプリケーションを使用するときは、table1 だけがスレーブで影響を受けます。table2 は別のデータベース内にあるため、スレーブ上の table2 は UPDATE によって変更されません。ここで、USE db1 ステートメントの代わりに、USE db4 ステートメントが使用されたものとします。

```
USE db4;
UPDATE db1.table1 SET col1 = 10, db2.table2 SET col2 = 20;
```

このケースで、ステートメントベースレプリケーションを使用するときは、UPDATE ステートメントの影響はスレーブではありません。しかし、行ベースレプリケーションを使用する場合は、UPDATE はスレーブ上の table1 を変更しますが、table2 は変更しません。つまり、--replicate-do-db によって指定されたデータベース内のテーブルのみが変更され、デフォルトデータベースを選択してもこの動作に影響しません。

クロスデータベース更新を機能させる必要がある場合は、代わりに --replicate-wild-do-table=db_name.% を使用してください。セクション17.2.3「サーバーがレプリケーションフィルタリングルールをどのように評価するか」を参照してください。

> **注記**
>
> このオプションは、--binlog-do-db がバイナリロギングに影響するのと同じ方法でレプリケーションに影響し、--replicate-do-db がレプリケーション動作にどのように影響するかに対してレプリケーション形式がどのように影響するかは、--binlog-do-db 動作に対して口ギング形式がどのように影響するかと同じです。
>
> このオプションは、BEGIN、COMMIT、または ROLLBACK ステートメントに影響しません。

- --replicate-ignore-db=db_name

コマンド行形式	--replicate-ignore-db=name

型	文字列

--replicate-do-db と同様に、このオプションの影響はステートメントベースまたは行ベースのどちらのレプリケーションが使用されるかによって異なります。

ステートメントベースのレプリケーション　デフォルトデータベース (つまり、USE で選択されたもの) が db_name であるステートメントを複製しないようにスレーブ SQL スレッドに指示します。

行ベースのレプリケーション　データベース db_name 内のテーブルを更新しないようにスレーブ SQL スレッドに指示します。デフォルトデータベースは影響しません。

ステートメントベースレプリケーションを使用する場合、次の例は予期したとおりに機能しません。スレーブが --replicate-ignore-db=sales で起動され、次のステートメントをマスターで発行するものとします。

```
USE prices;
UPDATE sales.january SET amount=amount+1000;
```

このような場合 UPDATE ステートメントは複製されます。--replicate-ignore-db が (USE ステートメントで指定された) デフォルトデータベースにのみ適用されるためです。sales データベースがステートメントで明示的に指定されたため、ステートメントはフィルタされませんでした。しかし、行ベースレプリケーションを使用するときは、UPDATE ステートメントの影響はスレーブに伝達されず、sales.january テーブルのスレーブのコピーは変更されません。この例では、sales データベースのマスターのコピー内のテーブルに加えられたすべての変更は、--replicate-ignore-db=sales によってスレーブで無視されます。

無視するデータベースを複数指定するには、このオプションを複数回 (データベースごとに 1 回) 使用します。データベース名にカンマを含めることができるため、カンマで区切られたリストを指定すると、リストは単一データベースの名前として扱われます。

クロスデータベース更新を使用していて、これらの更新を複製したくない場合は、このオプションを使用しないでください。セクション17.2.3「サーバーがレプリケーションフィルタリングルールをどのように評価するか」を参照してください。

クロスデータベース更新を機能させる必要がある場合、代わりに --replicate-wild-ignore-table=db_name.% を使用してください。セクション17.2.3「サーバーがレプリケーションフィルタリングルールをどのように評価するか」を参照してください。

> **注記**
>
> このオプションは、--binlog-ignore-db がバイナリロギングに影響するのと同じ方法でレプリケーションに影響し、--replicate-ignore-db がレプリケーション動作にどのように影響するかに対してレプリケーション形式がどのように影響するかは、--binlog-ignore-db 動作に対して口ギング形式がどのように影響するかと同じです。
>
> このオプションは、BEGIN、COMMIT、または ROLLBACK ステートメントに影響しません。

- --replicate-do-table=db_name.tbl_name

コマンド行形式	--replicate-do-table=name
型	文字列

レプリケーションを特定のテーブルに制限するようにスレーブ SQL スレッドに指示することで、レプリケーションフィルタを作成します。複数のテーブルを指定するには、このオプションを複数回 (テーブルごとに 1 回) 使用します。--replicate-do-db とは対照的に、これはクロスデータベース更新とデフォルトデータベース更新の両方に機能します。セクション17.2.3「サーバーがレプリケーションフィルタリングルールをどのように評価するか」を参照してください。

このオプションは、テーブルに適用されるステートメントにのみ影響します。ストアドルーチンなど、ほかのデータベースオブジェクトにのみ適用されるステートメントには影響しません。ストアドルーチンに作用するステートメントをフィルタするには、1 つまたは複数の --replicate-*-db オプションを使用します。

- --replicate-ignore-table=db_name.tbl_name

コマンド行形式	--replicate-ignore-table=name

型	文字列

指定されたテーブルを更新するステートメントを複製しないように (ほかのテーブルが同じステートメントによって更新される可能性があったとしても) スレーブ SQL スレッドに指示することで、レプリケーションフィルタを作成します。無視するテーブルを複数指定するには、このオプションを複数回 (テーブルごとに 1 回) 使用します。--replicate-ignore-db とは対照的に、これはクロスデータベース更新に機能します。セクション 17.2.3「サーバーがレプリケーションフィルタリングルールをどのように評価するか」を参照してください。

このオプションは、テーブルに適用されるステートメントにのみ影響します。ストアドルーチンなど、ほかのデータベースオブジェクトにのみ適用されるステートメントには影響しません。ストアドルーチンに作用するステートメントをフィルタするには、1 つまたは複数の --replicate-*-db オプションを使用します。

- --replicate-rewrite-db=from_name->to_name

コマンド行形式	--replicate-rewrite-db=old_name->new_name
型	文字列

デフォルトデータベース (つまり、USE で選択されたもの) を to_name に変換するレプリケーションフィルタを作成するように (それがマスター上の from_name であった場合)、スレーブに指示します。(CREATE DATABASE、DROP DATABASE、ALTER DATABASE などのステートメントではなく) テーブルに関係するステートメントだけが影響されます (from_name がマスター上のデフォルトデータベースの場合のみ)。複数の書き換えを指定するには、複数回このオプションを使用します。サーバーは、一致する from_name 値で最初のものを使用します。データベース名変換は、--replicate-* ルールがテストされる前に行われます。

このオプションを使用するときにテーブル名がデータベース名で修飾されるステートメントは、--replicate-do-table などのテーブルレベルレプリケーションフィルタリングオプションで機能しません。名前が a のデータベースがマスター上にあり、名前が b のものがスレーブ上にあり、それぞれにテーブル t が含まれ、--replicate-rewrite-db='a->b' でマスターを起動したものとします。あとで、DELETE FROM a.t を実行します。この場合、関連のあるフィルタリングルールは、次に示した理由で機能しません。

1. テーブル t がスレーブのデータベース b 内にあるため、--replicate-do-table=a.t は機能しません。
2. --replicate-do-table=b.t は、元のステートメントを照合しないため無視されます。
3. --replicate-do-table=*.t も、--replicate-do-table=a.t と同等に処理されるため機能しません。

同様に、--replication-rewrite-db オプションはクロスデータベース更新では機能しません。

このオプションをコマンド行で使用するときに、「>」 がコマンドインタープリターに固有である場合は、オプション値を引用符で囲みます。例:

```
shell> mysqld --replicate-rewrite-db="olddb->newdb"
```

> **注記**
>
> MySQL 5.6.7 より前では、マルチスレッドスレーブはこのオプションを正しく処理しませんでした。(Bug #14232958)

- --replicate-same-server-id

コマンド行形式	--replicate-same-server-id
型	ブール
デフォルト	FALSE

スレーブサーバーで使用されるべきです。循環レプリケーションの原因となる無限ループを避けるため、通常はデフォルト設定の 0 を使用してください。1 に設定すると、スレーブはそれ自身のサーバー ID を持つイベントをスキップしません。通常は、これはまれな構成でのみ役立ちます。--log-slave-updates が使用されている場合、1 に設定できません。デフォルトでは、バイナリログイベントのサーバー ID がスレーブのものである場合、スレーブ I/O スレッドはリレーログにそれらを書き込みません (この最適化はディスク使用量の節約に役立ちます)。--replicate-same-server-id を使用する場合、このオプションでスレーブを確実に起動してから、スレーブ SQL スレッドで実行したいスレーブ独自のイベントをスレーブで読み取ってください。

- --replicate-wild-do-table=db_name.tbl_name

コマンド行形式	--replicate-wild-do-table=name

型	文字列

更新されるテーブルが指定されたデータベースおよびテーブル名パターンに一致するステートメントにレプリケーションを制限するようにスレーブスレッドに指示することで、レプリケーションフィルタを作成します。パターンには 「%」 および 「_」 ワイルドカード文字を含めることができます。これらは LIKE パターンマッチング演算子と同じ意味を持ちます。複数のテーブルを指定するには、このオプションを複数回 (テーブルごとに 1 回) 使用します。これはクロスデータベース更新に役立ちます。セクション17.2.3「サーバーがレプリケーションフィルタリングルールをどのように評価するか」を参照してください。

このオプションはテーブル、ビュー、およびトリガーに適用されます。ストアドプロシージャーと関数、またはイベントには適用されません。後者のオブジェクトで作用するステートメントをフィルタするには、1 つまたは複数の --replicate-*-db オプションを使用します。

例: --replicate-wild-do-table=foo%.bar% は、データベース名が foo で始まり、テーブル名が bar で始まるテーブルを使用する更新のみを複製します。

テーブル名パターンが % の場合、それは任意のテーブル名に一致し、このオプションはデータベースレベルステートメント (CREATE DATABASE、DROP DATABASE、および ALTER DATABASE) にも適用されます。たとえば、--replicate-wild-do-table=foo%.% を使用する場合に、データベース名がパターン foo% に一致する場合はデータベースレベルステートメントが複製されます。

リテラルワイルドカード文字をデータベースまたはテーブル名パターンに含めるには、バックスラッシュでそれらをエスケープします。たとえば、名前が my_own%db のデータベースのすべてのテーブルを複製するけれども、my1ownAABCdb データベースからのテーブルは複製しないときは、「_」 および 「%」 文字を --replicate-wild-do-table=my_own\%db のように エスケープしてください。このオプションをコマンド行で使用する場合、コマンドインタープリターによっては、バックスラッシュを二重にしたりオプション値を引用符で囲んだりする必要があります。たとえば、bash シェルでは、--replicate-wild-do-table=my_own\\%db と入力する必要があります。

- --replicate-wild-ignore-table=db_name.tbl_name

コマンド行形式	--replicate-wild-ignore-table=name
型	文字列

何らかのテーブルが指定されたワイルドカードパターンに一致するステートメントをスレーブスレッドが複製するのを抑止する、レプリケーションフィルタを作成します。無視するテーブルを複数指定するには、このオプションを複数回 (テーブルごとに 1 回) 使用します。これはクロスデータベース更新に役立ちます。セクション17.2.3「サーバーがレプリケーションフィルタリングルールをどのように評価するか」を参照してください。

例: --replicate-wild-ignore-table=foo%.bar% は、データベース名が foo で始まりテーブル名が bar で始まるテーブルを使用する更新を複製しません。

照合の仕組みについては、--replicate-wild-do-table オプションの説明を参照してください。オプション値にリテラルワイルドカード文字を含めるためのルールは、--replicate-wild-ignore-table 場合と同じです。

- --report-host=host_name

コマンド行形式	--report-host=host_name
システム変数	report_host
スコープ	グローバル
動的	いいえ
型	文字列

スレーブ登録中にマスターに報告されるスレーブのホスト名または IP アドレス。この値は、マスターサーバーでの SHOW SLAVE HOSTS の出力に出現します。マスターにスレーブ自身を登録しない場合は、値を設定しないままにします。

> 注記
>
> スレーブ接続後にマスターが TCP/IP ソケットからスレーブの IP アドレスを読み取るだけでは十分ではありません。NAT およびその他のルーティングの問題により、その IP はマスターまたはほかのホストからスレーブへの接続に有効でない可能性があります。

- --report-password=password

コマンド行形式	--report-password=name
システム変数	report_password
スコープ	グローバル
動的	いいえ
型	文字列

スレーブ登録中にマスターに報告されるスレーブのアカウントパスワード。この値は、--show-slave-auth-info オプションが指定された場合にマスターサーバーでの SHOW SLAVE HOSTS の出力に出現します。

このオプションの名前は別の方法で暗示されることもありますが、--report-password は MySQL ユーザー権限システムに接続されないため、MySQL レプリケーションユーザーアカウントのパスワードとは必ずしも同じとはかぎりません（または同じである可能性は高くありません）。

- --report-port=slave_port_num

コマンド行形式	--report-port=#
システム変数	report_port
スコープ	グローバル
動的	いいえ
型	数値
デフォルト (≥ 5.6.5)	[slave_port]
デフォルト (≤ 5.6.4)	0
最小値	0
最大値	65535

スレーブに接続するための TCP/IP ポート番号で、スレーブ登録中にマスターに報告されます。スレーブが非デフォルトポートで待機している場合、またはマスターまたはほかのクライアントからスレーブへの特別なトンネルがある場合にのみ、これを設定します。確実でない場合は、このオプションを使用しないでください。

MySQL 5.6.5 より前は、このオプションのデフォルト値は 3306 でした。MySQL 5.6.5 以降では、表示される値はスレーブで実際に使用されるポート番号です (Bug #13333431)。この変更は SHOW SLAVE HOSTS によって表示されるデフォルト値にも影響します。

- --report-user=user_name

コマンド行形式	--report-user=name
システム変数	report_user
スコープ	グローバル
動的	いいえ
型	文字列

スレーブ登録中にマスターに報告されるスレーブのアカウントユーザー名。この値は、--show-slave-auth-info オプションが指定された場合にマスターサーバーでの SHOW SLAVE HOSTS の出力に出現します。

このオプションの名前は別の方法で暗示されることもありますが、--report-user は MySQL ユーザー権限システムに接続されないため、MySQL レプリケーションユーザーアカウントの名前とは必ずしも同じとはかぎりません（または同じである可能性は高くありません）。

- --show-slave-auth-info

コマンド行形式	--show-slave-auth-info
型	ブール
デフォルト	FALSE

--report-user および --report-password オプションで起動されたスレーブのマスターサーバーでの、SHOW SLAVE HOSTS の出力にスレーブユーザー名とパスワードを表示します。

- --slave-checkpoint-group=#

コマンド行形式	--slave-checkpoint-group=#
導入	5.6.3
型	数値
デフォルト	512
最小値	32
最大値	524280
ブロックサイズ	8

SHOW SLAVE STATUS によって表示されるマルチスレッドスレーブステータスを更新するためにチェックポイント操作が呼び出される前に、スレーブが処理できる最大トランザクション数を設定します。このオプションを設定しても、マルチスレッドが有効でないスレーブには影響しません。

このオプションは、--slave-checkpoint-period オプションとの組み合わせで、どちらかの制限を超えたときに、チェックポイントが実行され、トランザクションの数と最後のチェックポイントから経過した時間の両方を追跡するカウンタがリセットされる、という方法で機能します。

このオプションの最小許容値は 32 です (サーバーが -DWITH_DEBUG を使用して構築された場合を除く、その場合の最小値は 1)。効果的な値は常に 8 の倍数です。そのような倍数でない値に設定することもできますが、サーバーは値を格納する前に次に小さい 8 の倍数に丸めます。(例外: このような丸めはデバッグサーバーでは実行されません。)サーバーの構築方法にかかわらず、デフォルト値は 512 であり、最大許容値は 524280 です。

--slave-checkpoint-group は MySQL 5.6.3 で追加されました。

- --slave-checkpoint-period=#

コマンド行形式	--slave-checkpoint-period=#
導入	5.6.3
型	数値
デフォルト	300
最小値	1
最大値	4G
Unit	millisecond

SHOW SLAVE STATUS によって表示されるマルチスレッドのスレーブステータスを更新するためにチェックポイント操作が呼び出される前に、経過できる最大時間 (ミリ秒単位) を設定します。このオプションを設定しても、マルチスレッドが有効でないスレーブには影響しません。

このオプションは、--slave-checkpoint-group オプションとの組み合わせで、どちらかの制限を超えたときに、チェックポイントが実行され、トランザクションの数と最後のチェックポイントから経過した時間の両方を追跡するカウンタがリセットされる、という方法で機能します。

このオプションの最小許容値は 1 です (サーバーが -DWITH_DEBUG を使用して構築された場合を除く、その場合の最小値は 0)。サーバーの構築方法にかかわらず、デフォルト値は 300 であり、最大可能値は 4294967296 (4G バイト) です。

--slave-checkpoint-period は MySQL 5.6.3 で追加されました。

- --slave-parallel-workers

コマンド行形式	--slave-parallel-workers=#
導入	5.6.3
型	数値
デフォルト	0
最小値	0

最大値	1024

レプリケーションイベント（トランザクション）を並列に実行するためのスレーブワーカースレッドの数を設定します。この変数を 0 (デフォルト) に設定すると、並列実行が無効になります。最大値は 1024 です。

並列実行が有効のときは、スレーブ SQL スレッドはスレーブワーカースレッドのコーディネーターとして機能し、トランザクションはそれらにデータベースごとに分散されます。これは、スレーブ上のワーカースレッドは、ほかのデータベースへの更新が完了するのを待たずに、所定のデータベースに基づいてトランザクションを次々に処理できることを意味します。スレーブでのマルチスレッドの現在の実装は、データがデータベースごとに分割されていて、所定のデータベース内の更新が正しく機能するようにマスターの場合と同様に相対順序で行われることを前提としています。ただし、2 つのデータベース間でトランザクションを調整する必要はありません。

異なるデータベースでのトランザクションは、スレーブではマスターとは異なる順序が実行されることがあるという事実のため、最後に実行されたトランザクションをチェックしても、マスターからの以前のすべてのトランザクションがスレーブ上で実行されたことが保証されません。これは、マルチスレッド化したスレーブを使用する場合にロギングとリカバリに影響します。スレーブ上でマルチスレッドを使用したときにバイナリロギング情報をどのように解釈すればよいかについては、セクション13.7.5.35「SHOW SLAVE STATUS 構文」を参照してください。また、START SLAVE UNTIL がマルチスレッドスレーブでサポートされないことを意味します。

マルチスレッドが有効のときは、slave_transaction_retries は 0 に等しいとして処理され、変更できません。(現在のところ、トランザクションの再試行はマルチスレッドスレーブではサポートされません。)

MySQL 5.6.7 以降では、異なるデータベース内のテーブル間に外部キー関係を適用すると、マルチスレッドスレーブが並列モードではなくシーケンシャルを使用することになり、これがパフォーマンスに悪影響を与える可能性があることも認識するようにしてください。(Bug #14092635)

このオプションは MySQL 5.6.3 で追加されました。

> **注記**
>
> このオプション (または対応する slave_parallel_workers システム変数) に設定される値は、MySQL 5.6.3 で正しく処理されるとはかぎりません。この問題は MySQL 5.6.4 で修正されました (Bug #13334470)。

- --slave-pending-jobs-size-max=#

コマンド行形式	--slave-pending-jobs-size-max=#
導入	5.6.3
型	数値
デフォルト	16M
最小値	1024
最大値	18EB
Unit	bytes
ブロックサイズ	1024

マルチスレッドスレーブの場合、このオプションは、まだ適用されていないイベントを保持するスレーブワーカーキューに使用可能な最大メモリー量 (バイト単位) を設定します。このオプションを設定しても、マルチスレッドが有効でないスレーブには影響しません。

このオプションの可能な最小値は 1024 で、デフォルトは 16M バイトです。可能な最大値は 18446744073709551615 (16E バイト) です。1024 の正確な倍数でない値は、保存される前に次に大きい 1024 の倍数に丸められます。

> **重要**
>
> このオプションの値はマスターの max_allowed_packet の値以上である必要があります。そうでない場合は、マスターから到着する処理すべきイベントが残っているときにスレーブワーカーキューがいっぱいになる場合があります。

このオプションは MySQL 5.6.3 で追加されました。

- --skip-slave-start

コマンド行形式	--skip-slave-start
型	ブール
デフォルト	FALSE

サーバーの起動時にスレーブスレッドを起動しないように、スレーブサーバーに指示します。スレッドをあとで起動するには、START SLAVE ステートメントを使用します。

- --slave_compressed_protocol={0|1}

コマンド行形式	--slave-compressed-protocol
システム変数	slave_compressed_protocol
スコープ	グローバル
動的	はい
型	ブール
デフォルト	OFF

このオプションが 1 に設定されている場合は、スレーブとマスターの両方がサポートしている場合は、マスター/スレーブプロトコルに圧縮を使用します。デフォルトは 0 (圧縮なし) です。

- --slave-load-tmpdir=file_name

コマンド行形式	--slave-load-tmpdir=path
システム変数	slave_load_tmpdir
スコープ	グローバル
動的	いいえ
型	ディレクトリ名
デフォルト	/tmp

スレーブが一時ファイルを作成するディレクトリの名前。このオプションはデフォルトでは、tmpdir システム変数の値と同じです。スレーブ SQL スレッドは、LOAD DATA INFILE ステートメントを複製するときに、リレーログからロードされるファイルを一時ファイルに抽出してから、それらをテーブルにロードします。マスターでロードされるファイルが非常に大きい場合は、スレーブの一時ファイルも非常に大きくなります。このため、このオプションを使用して、利用可能な多くの空き領域を持つファイルシステム内にあるディレクトリに一時ファイルを置くように、スレーブに指示することをお勧めします。この場合、リレーログも非常に大きいため、--relay-log オプションを使用してそのファイルシステムにリレーログも置くことをお勧めします。

このオプションで指定するディレクトリは (メモリーベースファイルシステムではなく) ディスクベースファイルシステムに配置してください。LOAD DATA INFILE の複製に使用される一時ファイルがマシン再起動後も存続する必要があるためです。このディレクトリは、システム起動プロセス中にオペレーティングシステムによってクリアされるものではいけません。

- slave-max-allowed-packet=bytes

コマンド行形式	--slave-max-allowed-packet=#
導入	5.6.6
型	数値
デフォルト	1073741824
最小値	1024
最大値	1073741824

MySQL 5.6.6 以降では、このオプションがスレーブ SQL スレッドおよび I/O スレッドの最大パケットサイズ（バイト単位）を設定するので、行ベースレプリケーションを使用する大きな更新が max_allowed_packet を超えていることが原因でレプリケーションが失敗することはありません。(Bug #12400221、Bug #60926)

対応するサーバー変数 slave_max_allowed_packet は常に、1024 の正の整数の倍数である値を持ちます。このような倍数でない値に設定しても、値は次に大きい 1024 の倍数に自動的に丸められます。(たとえば、--slave-

max-allowed-packet=10000 でサーバーを起動する場合、使用される値は 9216 です。値として 0 に設定すると、1024 が使用されます。)このような場合、切り捨ての警告が発行されます。

最大 (およびデフォルト) 値は 1073741824 (1G バイト) で、最小は 1024 です。

- --slave-net-timeout=seconds

コマンド行形式	--slave-net-timeout=#
システム変数	slave_net_timeout
スコープ	グローバル
動的	はい
型	数値
デフォルト	3600
最小値	1

マスターからの後続のデータを待機する秒数 (これ以降は、スレーブは接続が切断されていると見なし、読み取りを中止し、再接続を試行)。最初の再試行はタイムアウトの直後に発生します。再試行の間隔は CHANGE MASTER TO ステートメントの MASTER_CONNECT_RETRY オプションで制御され、再接続の試行回数は --master-retry-count オプションによって制限されます。デフォルトは 3600 秒 (1 時間) です。

- slave-rows-search-algorithms=list

コマンド行形式	--slave-rows-search-algorithms=list
導入	5.6.6
型	セット
デフォルト	TABLE_SCAN,INDEX_SCAN
有効な値	TABLE_SCAN,INDEX_SCAN INDEX_SCAN,HASH_SCAN TABLE_SCAN,HASH_SCAN TABLE_SCAN,INDEX_SCAN,HASH_SCAN (INDEX_SCAN,HASH_SCAN と同等)

slave_allow_batching を使用して行ベースロギングとレプリケーションのために大量の行を準備するときに、このオプションは行の一致がどのように検索されるか、つまり、主キー、一意キー、または何らかのほかのキーを使用する検索、またはキーをまったく使用しない検索にハッシュを使用するかどうかを制御します。このオプションは、リスト INDEX_SCAN、TABLE_SCAN、HASH_SCAN から 2 つの値 (または 3 つも可) のカンマ区切りリストを取ります。リストを引用符で囲む必要はありませんが、引用符を使用するかどうかにかかわらず、リストに空白文字を含めてはいけません。可能な組み合わせ (リスト) とその結果を次の表に示します。

使用されるインデックス/オプション値	INDEX_SCAN,HASH_SCAN または INDEX_SCAN,TABLE_SCAN,HASH_SCAN	INDEX_SCAN,TABLE_SCAN	TABLE_SCAN,HASH_SCAN
主キーまたは一意キー	インデックススキャン	インデックススキャン	インデックスに基づくハッシュスキャン
(ほかの) キー	インデックスに基づくハッシュスキャン	インデックススキャン	インデックスに基づくハッシュスキャン
インデックスなし	ハッシュスキャン	テーブルスキャン	ハッシュスキャン

リスト内でアルゴリズムが指定される順序は、SELECT または SHOW VARIABLES ステートメントで表示される順序と違いはありません (直前に示した表で使用されるものと同じ)。デフォルト値は TABLE_SCAN,INDEX_SCAN で、これはインデックスを使用できるすべての検索はそれらを使用し、インデックスなしの検索はテーブルスキャンを使用することを意味します。

INDEX_SCAN,TABLE_SCAN,HASH_SCAN を指定することは、INDEX_SCAN,HASH_SCAN を指定する場合と同じ結果になります。主キーまたは一意キーを使用しない検索にハッシュを使用するには、このオ

プションを INDEX_SCAN,HASH_SCAN に設定します。すべての検索にハッシュを強制的に使用するには、TABLE_SCAN,HASH_SCAN に設定します。

このオプションは MySQL 5.6.6 で追加されました。

- --slave-skip-errors=[err_code1,err_code2,...|all|ddl_exist_errors]

コマンド行形式	--slave-skip-errors=name
システム変数	slave_skip_errors
スコープ	グローバル
動的	いいえ
型	文字列
デフォルト	OFF
有効な値	OFF [list of error codes] all ddl_exist_errors
有効な値	OFF [list of error codes] all ddl_exist_errors
有効な値	OFF [list of error codes] all

レプリケーションは通常、スレーブでエラーが発生したときに停止します。これは、データ内の不一致を手動で解決する機会です。このオプションは、ステートメントがオプション値にリストされたエラーを返すときに、レプリケーションを継続するようにスレーブ SQL スレッドに指示します。

このオプションは、エラーが発生している理由を完全に理解しないかぎり使用しないでください。レプリケーションセットアップとクライアントプログラムにバグがなく、MySQL 自体にバグがない場合は、レプリケーションを停止するエラーは発生しないはずです。このオプションを無計画に使用すると、スレーブとマスターの同期が絶望的に取れなくなり、これがなぜ発生したかの見当がつかなくなります。

エラーコードについては、スレーブエラーログおよび SHOW SLAVE STATUS の出力のエラーメッセージによって提供される数値を使用してください。付録B「エラー、エラーコード、および一般的な問題」 にはサーバーエラーコードがリストされています。

非推奨値 all を使用して、スレーブにすべてのエラーメッセージを無視させて、何が発生しても継続させることもできます（ただし、使用すべきではありません）。言うまでもなく、all を使用した場合、データの完全性に関して保証はありません。スレーブのデータがマスター上のものとかなり違っていても不満を言わないで下さい（バグレポートを出さないでください）。以上のことを警告しました。

MySQL 5.6 と MySQL Cluster NDB 7.3 以降では、追加の簡略版 ddl_exist_errors がサポートされ、これはエラーコードリスト 1007,1008,1050,1051,1054,1060,1061,1068,1094,1146 に相当します。

例:

```
--slave-skip-errors=1062,1053
--slave-skip-errors=all
--slave-skip-errors=ddl_exist_errors
```

- --slave-sql-verify-checksum={0|1}

コマンド行形式	--slave-sql-verify-checksum=value
導入	5.6.2

型	ブール
デフォルト	0
有効な値	0 1

このオプションが有効のときは、スレーブはリレーログから読み取られたチェックサムを調べて、不一致があった場合にスレーブはエラーで停止します。デフォルトで無効になっています。

このオプションは MySQL 5.6.2 で追加されました。

スレーブステータスログをテーブルに記録するためのオプション

MySQL 5.6 以降では、レプリケーションスレーブステータス情報をファイルではなくテーブルに記録できます。マスター情報ログおよびリレーログ情報ログの書き込みは、ここで示す、MySQL 5.6.2 で追加された 2 つのサーバーオプションを使用して個別に構成できます。

- --master-info-repository={FILE|TABLE}

コマンド行形式	--master-info-repository=FILE|TABLE
導入	5.6.2
型	文字列
デフォルト	FILE
有効な値	FILE TABLE

このオプションにより、サーバーはそのマスター情報ログをファイルまたはテーブルに書き込みます。ファイル名はデフォルトで master.info になります。ファイルの名前は --master-info-file サーバーオプションを使用して変更できます。

このオプションのデフォルト値は FILE です。TABLE を使用する場合、ログは mysql データベースの slave_master_info テーブルに書き込まれます。

--master-info-repository オプションは MySQL 5.6.2 で追加されました。

- --relay-log-info-repository={FILE|TABLE}

コマンド行形式	--relay-log-info-repository=FILE|TABLE
導入	5.6.2
型	文字列
デフォルト	FILE
有効な値	FILE TABLE

このオプションにより、サーバーはそのリレーログ情報のログをファイルまたはテーブルに記録します。ファイル名はデフォルトで relay-log.info になります。ファイルの名前は --relay-log-info-file サーバーオプションを使用して変更できます。

このオプションのデフォルト値は FILE です。TABLE を使用する場合、ログは mysql データベースの slave_relay_log_info テーブルに書き込まれます。

レプリケーションをクラッシュセーフにするために、このオプションは TABLE に設定する必要があります。さらに、--relay-log-recovery オプションを有効にする必要があります。詳細は、クラッシュセーフレプリケーションを参照してください。

--relay-log-info-repository オプションは MySQL 5.6.2 で追加されました。

情報ログテーブルとその内容は、所定の MySQL Server にローカルと見なされます。MySQL 5.6.9 以降では、それらは複製されず、それらへの変更はバイナリログに書き込まれません。(Bug #14741537)

詳細については、セクション17.2.2「レプリケーションリレーおよびステータスログ」を参照してください。

廃止されたレプリケーションスレーブオプション

次のオプションは MySQL 5.5 で削除されました。MySQL 5.6 でこれらのオプションを使用して mysqld を起動しようとすると、サーバーは次で停止します: 不明な変数エラー.これらのオプションにすでに関連付けられているレプリケーションパラメータを設定するには、CHANGE MASTER TO ... ステートメントを使用する必要があります (セクション13.4.2.1「CHANGE MASTER TO 構文」を参照してください)。

影響するオプションをこのリストに示します。

- --master-host
- --master-user
- --master-password
- --master-port
- --master-connect-retry
- --master-ssl
- --master-ssl-ca
- --master-ssl-capath
- --master-ssl-cert
- --master-ssl-cipher
- --master-ssl-key

レプリケーションスレーブで使用されるシステム変数

次の一覧で、レプリケーションスレーブサーバーを制御するためのシステム変数について説明します。これらはサーバー起動時に設定でき、それらの一部は SET を使用して実行時に変更できます。レプリケーションスレーブで使用されるサーバーオプションは、このセクションですでにリストされています。

- init_slave

コマンド行形式	--init-slave=name
システム変数	init_slave
スコープ	グローバル
動的	はい
型	文字列

この変数は init_connect に似ていますが、SQL スレッドが起動するたびにスレーブサーバーによって実行される文字列です。文字列の形式は init_connect 変数の場合と同じです。

> **注記**
>
> SQL スレッドは、クライアントに確認応答を送信してから init_slave を実行します。このため、START SLAVE が戻されるときに init_slave が実行されていることは保証されません。詳細については、セクション13.4.2.5「START SLAVE 構文」を参照してください。

- log_slow_slave_statements

導入	5.6.11
システム変数	log_slow_slave_statements
スコープ	グローバル
動的	はい
型	ブール
デフォルト	OFF

スロークエリーログが有効のときは、この変数はスレーブで実行するために long_query_time 秒を超える時間がかかったクエリーのロギングを有効にします。この変数は MySQL 5.6.11 で追加されました。

- master_info_repository

コマンド行形式	--master-info-repository=FILE|TABLE
導入	5.6.2
システム変数	master_info_repository
スコープ	グローバル
動的	はい
型	文字列
デフォルト	FILE
有効な値	FILE TABLE

この変数の設定によって、スレーブがマスターステータスおよび接続情報のログを FILE (master.info) または TABLE (mysql.slave_master_info) のどちらに記録するかが決まります。

この変数の設定は、sync_master_info システム変数の設定による影響にも直接影響します。詳細については、変数の説明を参照してください。

この変数は MySQL 5.6.2 で追加されました。

- relay_log

コマンド行形式	--relay-log=file_name
システム変数	relay_log
スコープ	グローバル
動的	いいえ
型	ファイル名

リレーログファイルの名前。

- relay_log_basename

導入	5.6.2
システム変数	relay_log_basename
スコープ	グローバル
動的	いいえ
型	ファイル名
デフォルト	datadir + '/' + hostname + '-relay-bin'

リレーログファイルの名前と完全パスを保持します。

relay_log_basename システム変数は MySQL 5.6.2 で追加されました。

- relay_log_index

コマンド行形式	--relay-log-index
システム変数	relay_log_index
スコープ	グローバル
動的	いいえ
型	ファイル名
デフォルト	*host_name*-relay-bin.index

リレーログのインデックスファイルの名前。データディレクトリ内のデフォルト名は host_name-relay-bin.index です。ここで、host_name はスレーブサーバーの名前です。

- relay_log_info_file

コマンド行形式	--relay-log-info-file=file_name
システム変数	relay_log_info_file
スコープ	グローバル
動的	いいえ
型	ファイル名
デフォルト	relay-log.info

スレーブがリレーログの情報を記録するファイルの名前。データディレクトリ内のデフォルト名は relay-log.info です。

- relay_log_info_repository

導入	5.6.2
システム変数	relay_log_info_repository
スコープ	グローバル
動的	はい
型	文字列
デフォルト	FILE
有効な値	FILE TABLE

この変数によって、スレーブのリレーログ内での位置が FILE (relay-log.info) または TABLE (mysql.slave_relay_log_info) のどちらに書き込まれるかが決まります。

この変数の設定は、sync_relay_log_info システム変数の設定による影響にも直接影響します。詳細については、変数の説明を参照してください。

この変数は MySQL 5.6.2 で追加されました。

- relay_log_recovery

コマンド行形式	--relay-log-recovery
システム変数	relay_log_recovery
スコープ	グローバル
動的 (≥ 5.6.6)	いいえ
動的 (≤ 5.6.5)	はい
型	ブール
デフォルト	FALSE

サーバー起動直後のリレーログ自動リカバリを有効にします。リカバリプロセスでは、新しいリレーログファイルを作成し、SQL スレッド位置をこの新しいリレーログに初期化し、I/O スレッドを SQL スレッド位置に初期化します。その後、マスターからのリレーログ読み取りが続行されます。MySQL 5.6.5 以前では、このグローバル変数を動的に変更できました。MySQL 5.6.6 以降は読み取り専用です。(Bug #13840948) MySQL Server バージョンにかかわらず、--relay-log-recovery オプションでスレーブを起動することで、その値を変更できます。これは、レプリケーションスレーブでクラッシュが発生したあとに、破損した可能性のあるリレーログが処理されないことを保証するために使用するべきであり、クラッシュセーフなスレーブを保証するために使用する必要があります。デフォルト値は 0 (無効)です。

relay_log_recovery が有効であり、スレーブがマルチスレッドモードで動作中に発生したエラーが原因で停止したときでも、ログにギャップがある場合には CHANGE MASTER TO を実行できません。MySQL 5.6.6 以降では、START SLAVE UNTIL SQL_AFTER_MTS_GAPS を使用して確実にすべてのギャップが処理されてから、単一スレッドモードに戻ったり CHANGE MASTER TO ステートメントを実行したりするようにしてください。

- rpl_stop_slave_timeout

コマンド行形式	--rpl-stop-slave-timeout=seconds
導入	5.6.13
システム変数	rpl_stop_slave_timeout
スコープ	グローバル
動的	はい
型	整数
デフォルト	31536000
最小値	2
最大値	31536000

MySQL 5.6.13 以降では、この変数を設定することで、タイムアウトまでに STOP SLAVE が待機する時間（秒単位）を制御できます。これは、STOP SLAVE ステートメントと、スレーブへのさまざまなクライアント接続を使用するほかのスレーブ SQL ステートメントとの間のデッドロックを回避するために使用できます。rpl_stop_slave_timeout の最大値およびデフォルト値は 31536000 秒 (1 年) です。最小は 2 秒です。

- slave_checkpoint_group

コマンド行形式	--slave-checkpoint-group=#
導入	5.6.3
システム変数	slave_checkpoint_group=#
スコープ	グローバル
動的	はい
型	数値
デフォルト	512
最小値	32
最大値	524280
ブロックサイズ	8

SHOW SLAVE STATUS によって表示されるマルチスレッドスレーブステータスを更新するためにチェックポイント操作が呼び出される前に、スレーブが処理できる最大トランザクション数を設定します。この変数を設定しても、マルチスレッドが有効ではないスレーブには影響しません。

この変数は、slave_checkpoint_period システム変数との組み合わせで、どちらかの制限を超えたときに、チェックポイントが実行され、トランザクションの数と最後のチェックポイントから経過した時間の両方を追跡するカウンタがリセットされる、という方法で機能します。

この変数の最小許容値は 32 です (サーバーが -DWITH_DEBUG を使用して構築された場合を除く、この場合の最小値は 1)。効果的な値は常に 8 の倍数です。そのような倍数でない値に設定することもできますが、サーバーは値を格納する前に次に小さい 8 の倍数に丸めます。(例外: このような丸めはデバッグサーバーでは実行されません。)サーバーの構築方法にかかわらず、デフォルト値は 512 であり、最大許容値は 524280 です。

slave_checkpoint_group は MySQL 5.6.3 で追加されました。

- slave_checkpoint_period

コマンド行形式	--slave-checkpoint-period=#
導入	5.6.3
システム変数	slave_checkpoint_period=#
スコープ	グローバル
動的	はい
型	数値
デフォルト	300
最小値	1

最大値	4G
Unit	millisecond

SHOW SLAVE STATUS によって表示されるマルチスレッドのスレーブステータスを更新するためにチェックポイント操作が呼び出される前に、経過できる最大時間（ミリ秒単位）を設定します。この変数を設定しても、マルチスレッドが有効ではないスレーブには影響しません。

この変数は、slave_checkpoint_group システム変数との組み合わせで、どちらかの制限を超えたときに、チェックポイントが実行され、トランザクションの数と最後のチェックポイントから経過した時間の両方を追跡するカウンタがリセットされる、という方法で機能します。

この変数の最小許容値は 1 です (サーバーが -DWITH_DEBUG を使用して構築された場合を除く、この場合の最小値は 0)。サーバーの構築方法にかかわらず、デフォルト値は 300 であり、最大可能値は 4294967296 (4G バイト) です。

slave_checkpoint_period は MySQL 5.6.3 で追加されました。

- slave_compressed_protocol

コマンド行形式	--slave-compressed-protocol
システム変数	slave_compressed_protocol
スコープ	グローバル
動的	はい
型	ブール
デフォルト	OFF

スレーブ/マスタープロトコルの圧縮を使用するかどうか (スレーブとマスターの両方がサポートしている場合)。

- slave_exec_mode

コマンド行形式	--slave-exec-mode=mode
システム変数	slave_exec_mode
スコープ	グローバル
動的	はい
型	列挙
デフォルト	IDEMPOTENT (NDB) STRICT (Other)
有効な値	IDEMPOTENT STRICT

IDEMPOTENT または STRICT モードがレプリケーション競合解決とエラーチェックで使用されるかどうかを制御します。IDEMPOTENT モードでは、キーが重複しているエラーとキーが見つからないエラーが抑制されます。

このモードは、MySQL Cluster レプリケーションのマルチマスターレプリケーション、循環レプリケーション、およびその他の特別なレプリケーションシナリオに必要です。(詳しくは、セクション18.6.10「MySQL Cluster レプリケーション: マルチマスターと循環レプリケーション」およびセクション18.6.11「MySQL Cluster レプリケーションの競合解決」を参照してください。)MySQL Cluster で提供される mysqld は、slave_exec_mode に明示的に設定される値を無視し、これを常に IDEMPOTENT として扱います。

MySQL Server 5.6 では、STRICT モードがデフォルト値です。これを変更しないでください。現在のところ、IDEMPOTENT モードは NDB でのみサポートされます。

- slave_load_tmpdir

コマンド行形式	--slave-load-tmpdir=path
システム変数	slave_load_tmpdir

スコープ	グローバル
動的	いいえ
型	ディレクトリ名
デフォルト	/tmp

LOAD DATA INFILE ステートメントを複製するためにスレーブが一時ファイルを作成するディレクトリの名前。

- slave_max_allowed_packet

導入	5.6.6
システム変数	slave_max_allowed_packet
スコープ	グローバル
動的	はい
型	数値
デフォルト	1073741824
最小値	1024
最大値	1073741824

MySQL 5.6.6 以降では、この変数はスレーブ SQL スレッドおよび I/O スレッドの最大パケットサイズを設定するので、更新が max_allowed_packet を超えたことが原因で、行ベースレプリケーションを使用する大きな更新がレプリケーションに失敗することはありません。

このグローバル変数は常に、1024 の正の整数の倍数である値を持ちます。そうでない何らかの値に設定しても、値は次に大きい 1024 の倍数に自動的に丸められて、格納または使用されます。slave_max_allowed_packet を 0 に設定すると、1024 が使用されます。(このような場合、切り捨ての警告が発行されます。)デフォルトおよび最大値は 1073741824 (1G バイト) で、最小は 1024 です。

slave_max_allowed_packet は --slave-max-allowed-packet オプションを使用して、起動時に設定することもできます。

- slave_net_timeout

コマンド行形式	--slave-net-timeout=#
システム変数	slave_net_timeout
スコープ	グローバル
動的	はい
型	数値
デフォルト	3600
最小値	1

マスター/スレーブ接続から後続のデータを待機する秒数 (これ以降は、読み取りを中止)。

- slave_parallel_workers

コマンド行形式	--slave-parallel-workers=#
導入	5.6.3
システム変数	slave_parallel_workers
スコープ	グローバル
動的	はい
型	数値
デフォルト	0
最小値	0

最大値	1024

レプリケーションイベント（トランザクション）を並列に実行するためのスレーブワーカースレッドの数を設定します。この変数を 0 (デフォルト) に設定すると、並列実行が無効になります。最大値は 1024 です。

並列実行が有効のときは、スレーブ SQL スレッドはスレーブワーカースレッドのコーディネーターとして機能し、トランザクションはそれらにデータベースごとに分散されます。これは、スレーブ上のワーカースレッドは、ほかのデータベースへの更新が完了するのを待たずに、所定のデータベースに基づいてトランザクションを次々に処理できることを意味します。スレーブでのマルチスレッドの現在の実装は、データがデータベースごとに分割されていて、所定のデータベース内の更新が正しく機能するようにマスターの場合と同様に相対順序で行われることを前提としています。ただし、2 つのデータベース間でトランザクションを調整する必要はありません。

異なるデータベースでのトランザクションは、スレーブではマスターとは異なる順序が実行されることがあるという事実のため、最後に実行されたトランザクションをチェックしても、マスターからの以前のすべてのトランザクションがスレーブ上で実行されたことが保証されません。これは、マルチスレッド化したスレーブを使用する場合にロギングとリカバリに影響します。スレーブ上でマルチスレッドを使用したときにバイナリロギング情報をどのように解釈すればよいかについては、セクション13.7.5.35「SHOW SLAVE STATUS 構文」を参照してください。また、START SLAVE UNTIL がマルチスレッドスレーブでサポートされないことを意味します。

マルチスレッドが有効のときは、slave_transaction_retries は 0 に等しいとして処理され、変更できません。(現在のところ、トランザクションの再試行はマルチスレッドスレーブではサポートされません。)

この変数は MySQL 5.6.3 で追加されました。

> **注記**
>
> この変数 (または対応する --slave-parallel-workers オプション) の値セットは MySQL 5.6.3 で正しく処理されるとはかぎりません。この問題は MySQL 5.6.4 で修正されました (Bug #13334470)。

- slave_pending_jobs_size_max

導入	5.6.3
システム変数	slave_pending_jobs_size_max
スコープ	グローバル
動的	はい
型	数値
デフォルト	0
最小値	1024
最大値	18EB
Unit	bytes
ブロックサイズ	1024

マルチスレッドスレーブの場合、この変数は、まだ適用されていないイベントを保持するスレーブワーカーキューに使用可能な最大メモリー量 (バイト単位) を設定します。この変数を設定しても、マルチスレッドが有効ではないスレーブには影響しません。

この変数の可能な最小値は 1024 で、デフォルトは 16M バイトです。可能な最大値は 18446744073709551615 (16E バイト) です。1024 の正確な倍数でない値は、保存される前に次に大きい 1024 の倍数に丸められます。

> **重要**
>
> この変数の値はマスターの max_allowed_packet の値以上である必要があります。そうでない場合は、マスターから到着する処理すべきイベントが残っているときにスレーブワーカーキューがいっぱいになる場合があります。

slave_pending_jobs_size_max は MySQL 5.6.3 で追加されました。

- slave_rows_search_algorithms

導入	5.6.6
システム変数	slave_rows_search_algorithms=list
スコープ	グローバル
動的	はい
型	セット
デフォルト	TABLE_SCAN,INDEX_SCAN
有効な値	TABLE_SCAN,INDEX_SCAN INDEX_SCAN,HASH_SCAN TABLE_SCAN,HASH_SCAN TABLE_SCAN,INDEX_SCAN,HASH_SCAN (INDEX_SCAN,HASH_SCAN と同等)

slave_allow_batching を使用して行ベースロギングとレプリケーションのために大量の行を準備するときに、slave_rows_search_algorithms 変数は行の一致がどのように検索されるか、つまり、主キー、一意キー、または何らかのほかのキーを使用する検索、またはキーをまったく使用しない検索にハッシュを使用するかどうかを制御します。このオプションは、リスト INDEX_SCAN、TABLE_SCAN、HASH_SCAN から少なくとも 2 つの値のカンマ区切りリストを取ります。値は文字列を想定するため、値は引用符で囲む必要があります。また、値に空白文字を含めてはいけません。可能な組み合わせ (リスト) とその結果を次の表に示します。

使用されるインデックス/オプション値	INDEX_SCAN,HASH_SCAN または INDEX_SCAN,TABLE_SCAN,HASH_SCAN	INDEX_SCAN,TABLE_SCAN	TABLE_SCAN,HASH_SCAN
主キーまたは一意キー	インデックススキャン	インデックススキャン	インデックスハッシュ
(ほかの) キー	インデックスハッシュ	インデックススキャン	インデックスハッシュ
インデックスなし	テーブルハッシュ	テーブルスキャン	テーブルハッシュ

リスト内でアルゴリズムが指定される順序は、ここで示すように、SELECT または SHOW VARIABLES ステートメントで表示される順序と違いはありません。

```
mysql> SET GLOBAL slave_rows_search_algorithms = "INDEX_SCAN,TABLE_SCAN";
Query OK, 0 rows affected (0.00 sec)

mysql> SHOW VARIABLES LIKE '%algorithms%';
+-------------------------------+-----------------------+
| Variable_name                 | Value                 |
+-------------------------------+-----------------------+
| slave_rows_search_algorithms | TABLE_SCAN,INDEX_SCAN |
+-------------------------------+-----------------------+
1 row in set (0.00 sec)

mysql> SET GLOBAL slave_rows_search_algorithms = "TABLE_SCAN,INDEX_SCAN";
Query OK, 0 rows affected (0.00 sec)

mysql> SHOW VARIABLES LIKE '%algorithms%';
+-------------------------------+-----------------------+
| Variable_name                 | Value                 |
+-------------------------------+-----------------------+
| slave_rows_search_algorithms | TABLE_SCAN,INDEX_SCAN |
+-------------------------------+-----------------------+
1 row in set (0.00 sec)
```

デフォルト値は TABLE_SCAN,INDEX_SCAN で、これはインデックスを使用できるすべての検索はそれらを使用し、インデックスなしの検索はテーブルスキャンを使用することを意味します。

INDEX_SCAN,TABLE_SCAN,HASH_SCAN を指定することは、INDEX_SCAN,HASH_SCAN を指定する場合と同じ結果になります。主キーまたは一意キーを使用しない検索にハッシュを使用するには、この変数を INDEX_SCAN,HASH_SCAN に設定します。すべての検索にハッシュを強制的に使用するには、TABLE_SCAN,HASH_SCAN に設定します。

この変数は MySQL 5.6.6 で追加されました。

- slave_skip_errors

コマンド行形式	--slave-skip-errors=name
システム変数	slave_skip_errors
スコープ	グローバル
動的	いいえ
型	文字列
デフォルト	OFF
有効な値	OFF [list of error codes] all ddl_exist_errors
有効な値	OFF [list of error codes] all ddl_exist_errors
有効な値	OFF [list of error codes] all

レプリケーションは通常、スレーブでエラーが発生したときに停止します。これは、データ内の不一致を手動で解決する機会です。この変数は、ステートメントが変数値にリストされたエラーを返すときに、レプリケーションを継続するようにスレーブ SQL スレッドに指示します。

- slave_sql_verify_checksum

導入	5.6.2
システム変数	slave_sql_verify_checksum
スコープ	グローバル
動的	はい
型	ブール
デフォルト	1
有効な値	0 1

スレーブ SQL スレッドはリレーログから読み取られたチェックサムを使用してデータを検証します。不一致があった場合、スレーブはエラーで停止します。

> **注記**
>
> スレーブ I/O スレッドは常に、ネットワークからイベントを受け取るときに可能であればチェックサムを読み取ります。

slave_sql_verify_checksum は MySQL 5.6.2 で追加されました。

- slave_transaction_retries

コマンド行形式	--slave-transaction-retries=#
システム変数	slave_transaction_retries
スコープ	グローバル

動的	はい
型	数値
デフォルト	10
最小値	0
最大値 (64 ビットプラットフォーム)	18446744073709551615
最大値 (32 ビットプラットフォーム)	4294967295

InnoDB デッドロック、またはトランザクションの実行時間が InnoDB の innodb_lock_wait_timeout、または NDB の TransactionDeadlockDetectionTimeout または TransactionInactiveTimeout を超えたために、レプリケーションスレーブ SQL スレッドがトランザクションの実行に失敗した場合、自動的に slave_transaction_retries 回再試行してからエラーで停止します。デフォルト値は 10 です。

マルチスレッドスレーブを使用するときは、トランザクションを再試行できません。つまり、slave_parallel_workers が 0 より大きい場合は、slave_transaction_retries が 0 に等しいとして扱われ、変更できません。

- slave_type_conversions

コマンド行形式	--slave-type-conversions=set
システム変数	slave_type_conversions
スコープ	グローバル
動的	いいえ
型	セット
デフォルト	
有効な値 (≥ 5.6.13)	ALL_LOSSY ALL_NON_LOSSY ALL_SIGNED ALL_UNSIGNED
有効な値 (≤ 5.6.12)	ALL_LOSSY ALL_NON_LOSSY

行ベースレプリケーションを使用するときにスレーブで有効なタイプ変換モードを制御します。MySQL 5.6.13 以降では、その値はリスト ALL_LOSSY、ALL_NON_LOSSY、ALL_SIGNED、ALL_UNSIGNED からのゼロ個以上の要素のカンマ区切りセットです。この変数を空の文字列に設定すると、マスターとスレーブの間のタイプ変換が禁止されます。変更を有効にするには、スレーブの再起動が必要です。

ALL_SIGNED および ALL_UNSIGNED は MySQL 5.6.13 で追加されました (Bug#15831300)。行ベースレプリケーションで属性の昇格と降格に適用できるタイプ変換モードの詳細については、行ベースレプリケーション: 属性の昇格と降格を参照してください。

- sql_slave_skip_counter

システム変数	sql_slave_skip_counter
スコープ	グローバル
動的	はい
型	数値

マスターからのイベントのうち、スレーブサーバーがスキップすべき数。

このオプションは GTID ベースレプリケーションと互換性がなく、--gtid-mode=ON のときにゼロ以外の値に設定してはいけません。MySQL 5.6.10 以降では、そのようにすることは明確に禁止されています。(Bug #15833516) GTID を採用するときにトランザクションをスキップする必要がある場合は、代わりにマスターか

ら gtid_executed を使用してください。これを行う方法については、空のトランザクションの注入を参照してください。

> **重要**
>
> この変数を設定することで指定される数のイベントをスキップして、スレーブがイベントグループの途中で開始する場合、スレーブは次のイベントグループの開始を見つけるまでスキップを続け、そのポイントから開始します。詳細については、セクション13.4.2.4「SET GLOBAL sql_slave_skip_counter 構文」を参照してください。

- sync_master_info

コマンド行形式	--sync-master-info=#
システム変数	sync_master_info
スコープ	グローバル
動的	はい
型	数値
デフォルト (64 ビットプラットフォーム, ≥ 5.6.6)	10000
デフォルト (64 ビットプラットフォーム, ≤ 5.6.5)	0
デフォルト (32 ビットプラットフォーム, ≥ 5.6.6)	10000
デフォルト (32 ビットプラットフォーム, ≤ 5.6.5)	0
最小値	0
最大値 (64 ビットプラットフォーム)	18446744073709551615
最大値 (32 ビットプラットフォーム)	4294967295

レプリケーションスレーブでのこの変数の影響は、次の段落で説明するように、スレーブの master_info_repository が FILE または TABLE のどちらに設定されているかによって異なります。

master_info_repository = FILE　sync_master_info の値が 0 より大きい場合は、スレーブはすべての sync_master_info イベントのあとにその master.info ファイルをディスクに同期します (fdatasync() を使用)。0 の場合は、MySQL サーバーは master.info ファイルからディスクへの同期を実行しません。代わりに、サーバーはオペレーティングシステムに依存して、ほかのファイルと同様にその内容を定期的にフラッシュします。

master_info_repository = TABLE　sync_master_info の値が 0 より大きい場合は、スレーブはすべての sync_master_info イベントのあとにそのマスター情報リポジトリテーブルを更新します。0 の場合は、テーブルは更新されません。

sync_master_info のデフォルト値は、MySQL 5.6.6 以降では 10000、その前は 0 です。

- sync_relay_log

コマンド行形式	--sync-relay-log=#
システム変数	sync_relay_log
スコープ	グローバル
動的	はい
型	数値
デフォルト (64 ビットプラットフォーム, ≥ 5.6.6)	10000
デフォルト (64 ビットプラットフォーム, ≤ 5.6.5)	0

デフォルト (32 ビットプラットフォーム, ≥ 5.6.6)	10000
デフォルト (32 ビットプラットフォーム, ≤ 5.6.5)	0
最小値	0
最大値 (64 ビットプラットフォーム)	18446744073709551615
最大値 (32 ビットプラットフォーム)	4294967295

この変数の値が 0 より大きい場合は、すべての sync_relay_log イベントがリレーログに書き込まれたあとに、MySQL サーバーはそのリレーログをディスクに同期します (fdatasync() を使用)。

sync_relay_log を 0 に設定すると、ディスクへの同期は実行されません。この場合、サーバーはオペレーティングシステムに依存してほかのファイルに関してリレーログの内容をときどきフラッシュします。

MySQL 5.6.6 より前は、0 がこの変数のデフォルトでした。MySQL 5.6. 以降では、デフォルトは 10000 です。

値 1 が一番安全な選択です (クラッシュの場合にリレーログから失われるイベントが最大で 1 つです)。しかし、一番遅い選択でもあります (ディスクにバッテリ付きキャッシュがある場合を除きます。その場合は同期が非常に速くなります)。

- sync_relay_log_info

コマンド行形式	--sync-relay-log-info=#
システム変数	sync_relay_log_info
スコープ	グローバル
動的	はい
型	数値
デフォルト (64 ビットプラットフォーム, ≥ 5.6.6)	10000
デフォルト (64 ビットプラットフォーム, ≤ 5.6.5)	0
デフォルト (32 ビットプラットフォーム, ≥ 5.6.6)	10000
デフォルト (32 ビットプラットフォーム, ≤ 5.6.5)	0
最小値	0
最大値 (64 ビットプラットフォーム)	18446744073709551615
最大値 (32 ビットプラットフォーム)	4294967295

スレーブでのこの変数の影響は、サーバーの relay_log_info_repository 設定 (FILE または TABLE) 、これが TABLE の場合は、さらにリレーログ情報テーブルで使用されるストレージエンジンがトランザクション対応か (InnoDB など) そうでないか (MyISAM) に依存します。これらの要因が、sync_relay_log_info 値がゼロおよびゼロより大きい場合にサーバーの動作にどのように影響するかを次の表で示します。

sync_relay_log_info	relay_log_info_repository		
	FILE	TABLE	
		トランザクション対応	トランザクション対応でない
N > 0	スレーブは各 N トランザクション後にその relay-log.info ファイルをディスクに同期します (fdatasync() を使用)。	テーブルは各トランザクション後に更新されます。(N は事実上無視されます。)	テーブルは各 N イベント後に更新されます。

sync_relay_log_info	relay_log_info_repository		
	FILE	TABLE	
		トランザクション対応	トランザクション対応でない
0	MySQL サーバーは relay-log.info ファイルからディスクへの同期を実行しません。代わりに、サーバーはオペレーティングシステムに依存してほかのファイルと同様にその内容を定期的にフラッシュします。		テーブルは更新されません。

sync_relay_log_info のデフォルト値は、MySQL 5.6.6 以降では 10000、それより前は 0 です。

17.1.4.4 バイナリログのオプションと変数

バイナリロギングで使用する起動オプション

バイナリロギングで使用されるシステム変数

このセクションで説明する mysqld オプションおよびシステム変数を使用して、バイナリログの操作に影響を与えたり、バイナリログにどのステートメントが書き込まれたかを制御したりできます。バイナリログの追加情報については、セクション5.2.4「バイナリログ」を参照してください。MySQL サーバーのオプションとシステム変数の使用に関する追加情報については、セクション5.1.3「サーバーコマンドオプション」およびセクション5.1.4「サーバーシステム変数」を参照してください。

バイナリロギングで使用する起動オプション

次のリストでは、バイナリログを有効化したり構成したりするための起動オプションについて説明します。バイナリロギングで使用するシステム変数については、このセクションの後半で説明します。

- --binlog-row-event-max-size=N

コマンド行形式	--binlog-row-event-max-size=#
型	数値
デフォルト (64 ビットプラットフォーム, ≥ 5.6.6)	8192
デフォルト (64 ビットプラットフォーム, ≤ 5.6.5)	1024
デフォルト (32 ビットプラットフォーム, ≥ 5.6.6)	8192
デフォルト (32 ビットプラットフォーム, ≤ 5.6.5)	1024
最小値	256
最大値 (64 ビットプラットフォーム)	18446744073709551615
最大値 (32 ビットプラットフォーム)	4294967295

行ベースのバイナリログイベントの最大サイズをバイト単位で指定します。可能であれば、行はこのサイズより小さいイベントにグループ化されます。値は 256 の倍数であるべきです。デフォルトは、MySQL 5.6.6 以降では 8192、それより前では 1024 です。セクション17.1.2「レプリケーション形式」を参照してください。

- --log-bin[=base_name]

コマンド行形式	--log-bin
システム変数	log_bin
スコープ	グローバル
動的	いいえ

型	ファイル名

バイナリロギングを有効化します。サーバーはデータを変更するすべてのステートメントのログをバイナリログに記録し、これはバックアップとレプリケーションに使用されます。セクション5.2.4「バイナリログ」を参照してください。

オプション値 (指定された場合) はログシーケンスのベース名です。サーバーは、ベース名に数字サフィクスを追加することで、バイナリログファイルを次々に作成します。ベース名を指定することをお勧めします (その理由は セクションB.5.8「MySQL の既知の問題」を参照してください)。そうでない場合、MySQL はベース名として host_name-bin を使用します。

MySQL 5.6.5 以降のサーバーは、インデックスファイルからエントリを読み取るときに、エントリに相対パスが含まれるかどうかをチェックし、さらにそうである場合は、パスの相対部が --log-bin オプションを使用して設定された絶対パスに置き換えられます。絶対パスは変わりません。このような場合、使用される新しいパスを有効にするために、インデックスを手動で編集する必要があります。MySQL 5.6.5 より前は、バイナリログまたはリレーログファイルの位置を変更するときは、手動介入が必要でした。(Bug #11745230、Bug #12133)

このオプションを設定することで、log_bin システム変数は ON (または 1) に設定されます (ベース名にではなく)。MySQL 5.6.2 以降では、バイナリログファイル名 (パス付き) は log_bin_basename システム変数として使用できます。

- --log-bin-index[=file_name]

コマンド行形式	--log-bin-index=file_name
型	ファイル名

バイナリログファイル名のインデックスファイル。セクション5.2.4「バイナリログ」を参照してください。ファイル名を省略した場合、および --log-bin でこれを指定しなかった場合、MySQL はファイル名として host_name-bin.index を使用します。

- --log-bin-trust-function-creators[={0|1}]

コマンド行形式	--log-bin-trust-function-creators
システム変数	log_bin_trust_function_creators
スコープ	グローバル
動的	はい
型	ブール
デフォルト	FALSE

このオプションは対応する log_bin_trust_function_creators システム変数を設定します。引数が指定されない場合、このオプションは変数を 1 に設定します。log_bin_trust_function_creators は、ストアドファンクションおよびトリガー作成に対して MySQL がどのように制限を適用するかに影響します。セクション20.7「ストアドプログラムのバイナリロギング」を参照してください。

- --log-bin-use-v1-row-events[={0|1}]

コマンド行形式	--log-bin-use-v1-row-events[={0|1}]
導入	5.6.6
システム変数	log_bin_use_v1_row_events
スコープ	グローバル
動的	いいえ
型	ブール
デフォルト	0

バージョン 2 バイナリログ行イベントを MySQL 5.6.6 以降で使用できます。ただし、バージョン 2 イベントは前の MySQL Server リリースで読み取れません。このオプションを 1 に設定することで、mysqld はバージョン 1 ロギングイベント（これが、以前のリリースで使用されるバイナリログイベントの唯一のバージョン）を使

用してバイナリログを書き込み、古いスレーブで読み取れるバイナリログを生成します。--log-bin-use-v1-row-events を 0 (デフォルト) に設定することで、mysqld はバージョン 2 バイナリログイベントを使用します。

このオプションに使用される値は、読み取り専用 log_bin_use_v1_row_events システム変数から取得できます。

--log-bin-use-v1-row-events は主に、NDB$EPOCH_TRANS() (バージョン 2 バイナリログ行イベントが必要) を競合検出関数として使用してレプリケーション競合検出および解決をセットアップするときに役立ちます。したがって、このオプションと --ndb-log-transaction-id は互換性がありません。

詳細は、セクション18.6.11「MySQL Cluster レプリケーションの競合解決」を参照してください。

- --log-short-format

コマンド行形式	--log-short-format
型	ブール
デフォルト	FALSE

バイナリログおよびスロークエリーログがアクティブ化されている場合、これらのログに記録する情報を少なくします。

ステートメント選択オプション　次のリスト内のオプションは、どのステートメントがバイナリログに書き込まれ、レプリケーションマスターサーバーによってそのスレーブに送られるかを制御します。マスターから受け取ったステートメントのどれを実行または無視すべきかを制御する、スレーブサーバーのオプションもあります。詳細については、セクション17.1.4.3「レプリケーションスレーブのオプションと変数」を参照してください。

- --binlog-do-db=db_name

コマンド行形式	--binlog-do-db=name
型	文字列

このオプションは、--replicate-do-db がレプリケーションに影響するのと同様にバイナリロギングに影響します。

このオプションの影響は、ステートメントベースまたは行ベースロギング形式のどちらが使用されるかに依存します (--replicate-do-db の影響がステートメントベースまたは行ベースレプリケーションのどちらが使用されたかに依存すると同じ)。指定されたステートメントのログを記録するために使用される形式が、binlog_format の値で示される形式と必ずしも同じではないことに留意してください。たとえば、CREATE TABLE や ALTER TABLE などの DDL ステートメントは、有効になっている ロギング形式にかかわらず、常にステートメントとしてログが記録されるため、--binlog-do-db の次のステートメントベースルールはステートメントのログが記録されるかどうかの判断に常に適用されます。

ステートメントベースのロギング　デフォルトデータベース (つまり、USE で選択されたもの) が db_name であるステートメントだけが、バイナリログに書き込まれます。複数のデータベースを指定するには、このオプションを複数回 （ データベースごとに 1 回 ） 使用します。ただし、このようにしても、別のデータベースが選択されているとき （ またはデータベースが選択されていないとき ） に、UPDATE some_db.some_table SET foo='bar' などのクロスデータベースステートメントのログは記録されません。

> **警告**
>
> 複数のデータベースを指定するには、このオプションの複数インスタンスを使用する必要があります。データベース名にカンマを含めることができるため、カンマ区切りリストを指定した場合は、リストは単一データベースの名前として扱われます。

ステートメントベースロギングを使用するときに想定される、機能しない例: サーバーが --binlog-do-db=sales で起動され、次のステートメントを発行する場合、UPDATE ステートメントのログは記録されません。

```
USE prices;
UPDATE sales.january SET amount=amount+1000;
```

この「デフォルトデータベースをチェックするだけ」動作の主な理由は、ステートメントだけから複製すべきかどうかを知ることが難しいことです (たとえば、複数のデータベースをまたがって動作する複数テーブル DELETE ステートメントまたは複数テーブル UPDATE ステートメントを使用する場合)。また、必要がない場合、すべてのデータベースではなくデフォルトデータベースだけをチェックする方が早いです。

もう 1 つのケースは自明ではないかもしれませんが、オプションを設定するときに指定されなかったけれども所定のデータベースが複製されます。サーバーが --binlog-do-db=sales で起動される場合、--binlog-do-db の設定時に prices が含まれなかったけれども、次の UPDATE ステートメントのログが記録されます。

```
USE sales;
UPDATE prices.discounts SET percentage = percentage + 10;
```

UPDATE ステートメントが発行されたときに sales がデフォルトデータベースであるため、UPDATE のログが記録されます。

行ベースのロギング　ロギングはデータベース db_name に制限されます。db_name に属するテーブルへの変更だけがログに記録されます。デフォルトデータベースはこれに影響しません。サーバーが --binlog-do-db=sales で起動され、行ベースロギングが有効であると想定すると、次のステートメントが実行されます。

```
USE prices;
UPDATE sales.february SET amount=amount+100;
```

sales データベース内の february テーブルへの変更が、UPDATE ステートメントに従ってログに記録されます。これは USE ステートメントが発行されたかどうかにかかわらず発生します。ただし、行ベースロギング形式および --binlog-do-db=sales を使用するときは、次の UPDATE によって行われた変更のログは記録されません。

```
USE prices;
UPDATE prices.march SET amount=amount-25;
```

USE prices ステートメントが USE sales に変更された場合でも、UPDATE ステートメントの結果は依然としてバイナリログに書き込まれません。

--binlog-do-db 処理でステートメントベースロギングと行ベースロギング間のもう 1 つの重要な違いは、複数のデータベースを参照するステートメントに関して発生します。サーバーが --binlog-do-db=db1 で起動され、次のステートメントが実行されると想定します。

```
USE db1;
UPDATE db1.table1 SET col1 = 10, db2.table2 SET col2 = 20;
```

ステートメントベースロギングを使用している場合、両方のテーブルへの更新がバイナリログに書き込まれます。一方、行ベース形式を使用するときは、table1 への変更だけがログに記録されます。table2 は別のデータベース内にあり、UPDATE によって変更されません。ここで、USE db1 ステートメントの代わりに、USE db4 ステートメントが使用されたものとします。

```
USE db4;
UPDATE db1.table1 SET col1 = 10, db2.table2 SET col2 = 20;
```

この場合、ステートメントベースロギングを使用するときに UPDATE ステートメントはバイナリログに書き込まれません。一方、行ベースロギングを使用するときは、table1 への変更のログは記録されますが、table2 へはそうなりません。つまり、--binlog-do-db によって指定されたデータベース内のテーブルへの変更のみがログに記録され、デフォルトデータベースの選択はこの動作に影響しません。

- --binlog-ignore-db=db_name

コマンド行形式	--binlog-ignore-db=name
型	文字列

このオプションは、--replicate-ignore-db がレプリケーションに影響するように、バイナリロギングに影響します。

このオプションの影響は、ステートメントベースまたは行ベースロギング形式のどちらが使用されるかに依存します (--replicate-ignore-db の影響がステートメントベースまたは行ベースレプリケーションのどちらが使用されたかに依存すると同じ)。指定されたステートメントのログを記録するために使用される形式が、binlog_format の値で示される形式と必ずしも同じではないことに留意してください。たとえば、CREATE TABLE や ALTER TABLE などの DDL ステートメントは、有効になっている ロギング形式にかかわらず、常に

ステートメントとしてログが記録されるため、--binlog-ignore-db の次のステートメントベースルールはステートメントのログが記録されるかどうかの判断に常に適用されます。

ステートメントベースのロギング　デフォルトデータベース (つまり、USE で選択されたもの) が db_name であるステートメントのログを記録しないようにサーバーに指示します。

MySQL 5.6.12 より前は、このオプションにより、デフォルトデータベースが指定されなかった場合 (つまり、SELECT DATABASE() が NULL を返したとき) は、完全修飾テーブル名を含むステートメントのログを記録しませんでした。MySQL 5.6.12 以降では、デフォルトデータベースがないときは、--binlog-ignore-db オプションは適用されず、常にこのようなステートメントのログが記録されます。(Bug #11829838、Bug #60188)

行ベース形式　データベース db_name 内のテーブルへの更新のログを記録しないようにサーバーに指示します。現在のデータベースは影響しません。

ステートメントベースロギングを使用するとき、次の例は予期するとおりに機能しません。サーバーが --binlog-ignore-db=sales で起動され、次のステートメントを発行すると想定します。

```
USE prices;
UPDATE sales.january SET amount=amount+1000;
```

--binlog-ignore-db がデフォルトデータベース (USE ステートメントで決定) にのみ適用されるため、このような場合は UPDATE ステートメントのログが記録されます。sales データベースがステートメントで明示的に指定されたため、ステートメントはフィルタされませんでした。一方、行ベースロギングを使用するときは、UPDATE ステートメントの結果はバイナリログに書き込まれず、これは sales.january テーブルへの変更がログに記録されないことを意味します。この例では、--binlog-ignore-db=sales によって、マスターの sales データベースのコピー内のテーブルに加えられたすべての変更がバイナリロギングのために無視されます。

無視するデータベースを複数指定するには、このオプションを複数回 (データベースごとに 1 回) 使用します。データベース名にカンマを含めることができるため、カンマ区切りリストを指定した場合は、リストは単一データベースの名前として扱われます。

クロスデータベース更新を使用していて、それらの更新ログを記録したくない場合は、このオプションを使用しないでください。

チェックサムオプション　MySQL 5.6.2 以降では、MySQL はバイナリログチェックサムの読み取りと書き込みをサポートします。これらは、ここで示す 2 つのオプションを使用して有効化されます。

- --binlog-checksum={NONE|CRC32}

コマンド行形式	--binlog-checksum=type
導入	5.6.2
型	文字列
デフォルト (≥ 5.6.6)	CRC32
デフォルト (≤ 5.6.5)	NONE
有効な値	NONE CRC32

このオプションを有効にすることで、マスターはバイナリログに書き込まれるイベントのチェックサムを書き込みます。NONE に設定すると無効になり、そうしない場合は、アルゴリズムの名前を使用してチェックサムが生成されます。現在のところ、CRC32 チェックサムだけがサポートされます。MySQL 5.6.6 以降では、CRC32 がデフォルトです。

このオプションは MySQL 5.6.2 で追加されました。

- --master-verify-checksum={0|1}

コマンド行形式	--master-verify-checksum=name
導入	5.6.2
型	ブール
デフォルト	OFF

このオプションを有効にすることで、マスターはチェックサムを使用してバイナリログからのイベントを検証し、不一致の場合はエラーで停止します。デフォルトで無効になっています。

このオプションは MySQL 5.6.2 で追加されました。

スレーブ（リレーから）ログによるチェックサム読み取りを制御するには、--slave-sql-verify-checksum オプションを使用します。

テストおよびデバッグのオプション　次のバイナリログオプションは、レプリケーションテストおよびデバッグで使用されます。これらは通常操作での使用を意図していません。

- --max-binlog-dump-events=N

コマンド行形式	--max-binlog-dump-events=#
型	数値
デフォルト	0

このオプションは、レプリケーションのテストとデバッグのために、MySQL テストスイートで内部的に使用されます。

- --sporadic-binlog-dump-fail

コマンド行形式	--sporadic-binlog-dump-fail
型	ブール
デフォルト	FALSE

このオプションは、レプリケーションのテストとデバッグのために、MySQL テストスイートで内部的に使用されます。

- --binlog-rows-query-log-events

コマンド行形式	--binlog-rows-query-log-events
導入	5.6.2
型	ブール
デフォルト	FALSE

このオプションは MySQL 5.6.2 で追加され、binlog_rows_query_log_events を有効にします。MySQL 5.6.1 以前のスレーブサーバーまたは mysqlbinlog のバージョンのためにログを生成するときは、OFF (デフォルト) に設定する必要があります。

バイナリロギングで使用されるシステム変数

次の一覧で、バイナリロギングを制御するためのシステム変数について説明します。これらはサーバー起動時に設定でき、それらの一部は SET を使用して実行時に変更できます。バイナリロギングを制御するために使用されるサーバーオプションは、このセクションですでにリストされています。sql_log_bin および sql_log_off 変数については、セクション5.1.4「サーバーシステム変数」を参照してください。

- binlog_cache_size

コマンド行形式	--binlog-cache-size=#
システム変数	binlog_cache_size
スコープ	グローバル
動的	はい
型	数値
デフォルト	32768
最小値	4096
最大値 (64 ビットプラットフォーム)	18446744073709551615
最大値 (32 ビットプラットフォーム)	4294967295

トランザクション中にバイナリログへの変更を保持するキャッシュのサイズ。サーバーがトランザクションストレージエンジンをサポートし、サーバーのバイナリログが有効 (--log-bin オプション) になっている場合は、バイナリログキャッシュがクライアントごとに割り当てられます。大きなトランザクションをよく使用する場合、パフォーマンスを向上するためにこのキャッシュサイズを増やすことができます。Binlog_cache_use および Binlog_cache_disk_use ステータス変数は、この変数のサイズを調整するために役立つことがあります。セクション5.2.4「バイナリログ」を参照してください。

binlog_cache_size はトランザクションキャッシュのみのサイズを設定します。ステートメントキャッシュのサイズは binlog_stmt_cache_size システム変数によって管理されます。

- binlog_checksum

導入	5.6.2
システム変数	binlog_checksum
スコープ	グローバル
動的	はい
型	文字列
デフォルト (≥ 5.6.6)	CRC32
デフォルト (≤ 5.6.5)	NONE
有効な値	NONE CRC32

この変数が有効のときは、マスターはバイナリログに各イベントのチェックサムを書き込みます。binlog_checksum は値 NONE (無効) および CRC32 をサポートします。デフォルトは MySQL 5.6.6 以降ではCRC32、それより前は NONE です。

binlog_checksum が無効 (値 NONE) のときは、サーバーは各イベントのイベント長 (チェックサムではなく) を書き込んでチェックすることで、すべてそろったイベントのみをバイナリログに書き込んでいることを検証します。

この変数の値を変更すると、バイナリログがローテーションします。チェックサムは常にバイナリログファイル全体に (その一部だけにではなく) 書き込まれます。

この変数は MySQL 5.6.2 で追加されました。

MySQL 5.6.6 以降では、マスター上のこの変数をスレーブで認識されない値に設定すると、スレーブは独自の binlog_checksum 値を NONE に設定し、レプリケーションをエラーで停止します。(Bug #13553750、Bug #61096) 古いスレーブとの下位互換性が懸念される場合は、明示的に値を NONE に設定することをお勧めします。

- binlog_direct_non_transactional_updates

コマンド行形式	--binlog-direct-non-transactional-updates[=value]
システム変数	binlog_direct_non_transactional_updates
スコープ	グローバル、セッション
動的	はい
型	ブール
デフォルト	OFF

トランザクションテーブルおよび非トランザクションテーブルの両方への更新がトランザクションに含まれるときは、並列問題によってスレーブが不整合になる可能性があります。MySQL は、非トランザクションステートメントをトランザクションキャッシュに書き込むことで (コミット時にフラッシュされる)、これらのステートメント間の因果関係を維持しようとします。ただし、トランザクションに代わって非トランザクション

テーブルに行われた変更がほかの接続にただちに可視になるときに、問題が起こります (これらの変更がただちにバイナリログに書き込まれない場合があるため)。

binlog_direct_non_transactional_updates 変数は、この問題に対する 1 つの可能な回避策を提供します。デフォルトでは、この変数は無効です。binlog_direct_non_transactional_updates を有効にすることで、非トランザクションテーブルへの更新が、トランザクションキャッシュではなく直接バイナリログに書き込まれます。

binlog_direct_non_transactional_updates は、ステートメントベースバイナリロギング形式を使用して複製されるステートメントにのみ機能します。つまり、binlog_format の値が STATEMENT のとき、または binlog_format が MIXED で、与えられたステートメントがステートメントベース形式を使用して複製されているときにのみ機能します。バイナリログ形式が ROW のとき、または binlog_format が MIXED に設定され、与えられたステートメントが行ベース形式で複製されるときは、この変数は効果がありません。

> **重要**
>
> この変数を有効にする前に、トランザクションおよび非トランザクションテーブルの間に依存関係がないことを確認する必要があります。このような依存関係の例は、ステートメント INSERT INTO myisam_table SELECT * FROM innodb_table です。そうでない場合、このようなステートメントによって、スレーブとマスターの間に相違が発生する可能性があります。

MySQL 5.6 では、バイナリログ形式が ROW または MIXED のとき、この変数は効果がありません。(Bug #51291)

- binlog_error_action

コマンド行形式	--binlog-error-action[=value]
導入	5.6.22
システム変数	binlog_error_action
スコープ	グローバル、セッション
動的	はい
型	列挙
デフォルト	IGNORE_ERROR
有効な値	IGNORE_ERROR ABORT_SERVER

サーバーがバイナリログに書き込みめないとき (マスターのログが不整合になり、レプリケーションスレーブが同期を失う可能性がある) に何が起きるかを制御します。以前のリリースでは、名前 binlogging_impossible_mode を使用していました。

MySQL 5.6 では、binlog_error_action のデフォルトは IGNORE_ERROR で、サーバーがエラーのログを記録し、ロギングを停止してから、更新の実行を継続することを意味します。これは、古いバージョンの MySQL Server との下位互換性を提供するためです。この変数を ABORT_SERVER に設定すると、サーバーがバイナリログに書き込めないときはロギングを停止し、シャットダウンします。これが推奨される設定です (特に複雑なレプリケーション環境で)。

- binlog_format

コマンド行形式	--binlog-format=format
システム変数	binlog_format
スコープ	グローバル、セッション
動的	はい
型	列挙
デフォルト (≥ 5.6.10-ndb-7.3.1)	MIXED
デフォルト	STATEMENT
有効な値	ROW STATEMENT

	MIXED

この変数はバイナリロギング形式を設定し、STATEMENT、ROW、MIXED のいずれかが可能です。セクション17.1.2「レプリケーション形式」を参照してください。binlog_format は、起動時に --binlog-format オプションで、または実行時に binlog_format 変数で設定されます。

> **注記**
>
> 実行時にロギング形式を変更できますが、レプリケーションの進行中に変更することは推奨されていません。これは一つには、スレーブがマスターの binlog_format 設定に従わず、所定の MySQL Server が独自のロギング形式のみを変更できるという事実によります。

MySQL 5.6 では、デフォルト形式は STATEMENT です。例外: MySQL Cluster NDB 7.3 以降では、デフォルトは MIXED です。ステートメントベースレプリケーションは MySQL Cluster ではサポートされていません。

グローバルまたはセッション binlog_format 値を設定するには、SUPER 権限が必要です。

この変数への変更がいつ有効になり、影響はどのくらい長く続くかを管理するルールは、ほかの MySQL サーバーシステム変数の場合と同じです。詳細については、セクション13.7.4「SET 構文」を参照してください。

MIXED が指定されている場合、行ベースレプリケーションだけが適切な結果になることが保証されている場合を除き、ステートメントベースレプリケーションが使用されます。たとえば、これはユーザー定義関数 (UDF) または UUID() 関数がステートメントに含まれているときに発生します。このルールの例外は、MIXED がストアドファンクションおよびトリガーのために常にステートメントベースレプリケーションを使用することです。

レプリケーション形式を実行時に切り替えることができない例外もあります。

- ストアドファンクションまたはトリガー内から。
- セッションが現在行ベースレプリケーションモードで、開いている一時テーブルを持つ場合。
- トランザクション内から。

これらの場合に形式を切りかえようとするとエラーになります。

バイナリログ形式は次のサーバーオプションの動作に影響を与えます。

- --replicate-do-db
- --replicate-ignore-db
- --binlog-do-db
- --binlog-ignore-db

これらの影響の詳細は、個々のオプションの説明に記載されています。

- binlog_gtid_recovery_simplified

コマンド行形式	--binlog-gtid-recovery-simplified
導入	5.6.23
システム変数	binlog_gtid_recovery_simplified
スコープ	グローバル
動的	いいえ
型	ブール
デフォルト	FALSE

MySQL バージョン 5.6.21 では、この変数は simplified_binlog_gtid_recovery として追加され、MySQL バージョン 5.6.23 では、その名前が binlog_gtid_recovery_simplified に変わりました。

デフォルトでは、MySQL はクラッシュからリカバリするときに、バイナリログファイルを反復して一番古いファイルから始めて GTID イベントを検索するため、大量のバイナリログファイルがある場合はこれ

に時間がかかることがあります。このオプションを有効にすることで、代わりに一番新しいバイナリログファイルから GTID イベントが検索されます。Previous_gtids_log_event および Gtid_log_event が検出されると、gtid_executed と gtid_purged はリカバリ中に通常どおりこれらのイベントを使用して初期化されます。GTID イベントが検出されない場合は、2 番目のスキャンが一番古いバイナリログファイルから GTID イベントを検索します。これらのファイルのどちらにも GTID イベントが含まれない場合は、これ以上バイナリログファイルは検索されず、gtid_executed と gtid_purged は空の文字列に設定されます。

- binlogging_impossible_mode

コマンド行形式	--binlogging-impossible-mode[=value]
導入	5.6.20
非推奨	5.6.22
システム変数	binlogging_impossible_mode
スコープ	グローバル、セッション
動的	はい
型	列挙
デフォルト	IGNORE_ERROR
有効な値	IGNORE_ERROR ABORT_SERVER

このオプションは非推奨で、今後の MySQL リリースで削除される予定です。名前が変更された binlog_error_action を使用して、サーバーがバイナリログに書き込めないときに何が起きるかを制御してください。

- binlog_max_flush_queue_time

導入	5.6.6
システム変数	binlog_max_flush_queue_time
スコープ	グローバル
動的	はい
型	数値
デフォルト	0
最小値	0
最大値	100000

グループコミットを進める前にフラッシュキューからのトランザクション読み取り (および、sync_binlog が 0 より大きい場合はログとディスクの同期) をどのくらい続けるか (マイクロ秒単位)。値が 0 (デフォルト) の場合、タイムアウトはなく、キューが空になるまでサーバーは新しいトランザクションの読み取りを続けます。

通常、binlog_max_flush_queue_time を 0 に設定されたままでかまいません。サーバーが大量の接続 (たとえば、100 以上)、および低遅延要件の多くの短いトランザクションを処理する場合、0 より大きな値を設定して、ディスクへのフラッシュを強制的により頻繁にすることが役立つ場合があります。

この変数は MySQL 5.6.6 で追加されました。

- binlog_order_commits

導入	5.6.6
システム変数	binlog_order_commits
スコープ	グローバル
動的	はい
型	ブール

デフォルト	ON

この変数が有効の場合は (デフォルト)、トランザクションはバイナリログに書き込まれる順序と同じ順序でコミットされます。無効の場合は、トランザクションは並列にコミットされる場合があります。場合によっては、この変数を無効にすることでパフォーマンスが向上することがあります。

この変数は MySQL 5.6.6 で追加されました。

- binlog_row_image

コマンド行形式	--binlog-row-image=image_type
導入	5.6.2
システム変数	binlog_row_image=image_type
スコープ	グローバル、セッション
動的	はい
型	列挙
デフォルト	full
有効な値	full (すべてのカラムをログに記録) minimal (変更されたカラムおよび、行を特定するために必要とされたカラムのみをログに記録) noblob (不要な BLOB カラムおよび TEXT カラム以外のすべてのカラムをログに記録)

MySQL 行ベースレプリケーションでは、各行変更イベントに 2 つのイメージ、つまり「前」イメージ (更新される行を検索するときにこれらのカラムが照合される) と「後」イメージ (変更を含む) が含まれます。通常、MySQL は前イメージと後イメージの両方のためにすべての行 (つまり、すべてのカラム) のログを記録します。ただし、両方のイメージにすべてのカラムが厳密に含まれている必要はなく、多くの場合、実際に必要なカラムのログのみを記録することでディスク、メモリー、およびネットワーク使用量を節約できます。

> 注記
>
> 行を削除するときは、削除後に伝達する変更後の値がないため、前イメージのみのログが記録されます。行を挿入するときは、照合される既存の行がないため、後イメージのみのログが記録されます。行を更新するときのみ、前イメージと後イメージの両方が必要で、両方がバイナリログに書き込まれます。

前イメージについては、行を一意に識別するために必要な最小カラムセットのログが記録されることのみが必要です。行を含むテーブルに主キーがある場合、主キーカラムだけがバイナリログに書き込まれます。そうではなく、テーブルに一意キーがあり、そのカラムのすべてが NOT NULL の場合は、一意キー内のカラムのログのみを記録する必要があります。(テーブルに NULL カラムなしの主キーまたは一意キーがない場合、すべてのカラムが前イメージで使用され、それらのログが記録される必要があります。)後イメージでは、実際に変更されたカラムのログのみを記録する必要があります。

MySQL 5.6 ではサーバーに、binlog_row_image システム変数を使用して full または minimal 行のログを記録させることができます。この変数は実際には、次のリストで示す 3 つの可能な値の 1 つを取ることができます。

- full: 前イメージと後イメージの両方にすべてのカラムのログを記録します。
- minimal: 変更する行を識別するために必要なカラムのみのログを前イメージに記録し、実際に変更されるカラムのみのログを後イメージに記録します。

- noblob: すべてのカラム (full と同じ) のログを記録しますが、行の識別に必要がない、または変更されなかった BLOB および TEXT カラムは除きます。

  > **注記**
  >
  > この変数は MySQL Cluster ではサポートされません。設定しても NDB テーブルのロギングには影響しません。(Bug #16316828)

デフォルト値は full です。MySQL 5.5 以前は、前イメージと後イメージの両方に常に full 行イメージが使用されます。MySQL 5.6 以降のマスターから以前のバージョンの MySQL を実行するからスレーブに複製する必要がある場合、マスターは常にこの値を使用してください。

minimal または noblob を使用するときは、ソーステーブルと宛先テーブルの両方について次の条件が true の場合にのみ、所定のテーブルに対して削除と更新が正しく機能することが保証されます。

- すべてのカラムが同じ順番で存在する必要があります。それぞれのカラムがもう一方のテーブル内の対応するものと同じデータ型を使用する必要があります。
- これらのテーブルの主キー定義が同じである必要があります。

(つまり、これらのテーブルは同じである必要がありますが、テーブルの主キーの一部でないインデックスがある場合にはそれらは除きます。)

これらの条件が満たされない場合は、宛先テーブル内の主キーカラム値が、削除または更新のための一意一致を提供するために不十分であることが判明する場合があります。この場合、警告またはエラーは発行されません。マスターとスレーブはサイレントに相違し、一貫性がなくなります。

バイナリロギング形式が STATEMENT のときは、この変数を設定しても効果はありません。binlog_format が MIXED のときは、binlog_row_image の設定は行ベース形式を使用してログが記録される変更に適用されますが、この設定はステートメントとしてログが記録される変更には影響しません。

グローバルまたはセッションレベルのいずれかで binlog_row_image を設定しても、暗黙的にコミットされません。これは、トランザクションの進行中にトランザクションに影響を与えずにこの変数を変更できることを意味します。

- binlog_rows_query_log_events

導入	5.6.2
システム変数	binlog_rows_query_log_events
スコープ	グローバル、セッション
動的	はい
型	ブール
デフォルト	FALSE

binlog_rows_query_log_events システム変数は行ベースロギングにのみ影響します。有効のときは、MySQL 5.6.2 以降のサーバーは行クエリーログイベントなどの情報ログイベントをそのバイナリログに書き込みます。この情報は、デバッグおよび関連する目的 (マスター上で発行された元のクエリーを行更新から再構成できないときにそのクエリーを取得する、など) のために使用できます。

これらのイベントは通常、バイナリログを読み取る MySQL 5.6.2 以降のプログラムで無視されるため、レプリケーションまたはバックアップからのリストア時に問題は発生しません。これは、MySQL 5.6.1 以前の mysqld または mysqlbinlog では true ではありません。ログを読み取る古いバージョンのプログラムが情報ログイベントに遭遇すると、それは失敗し、その時点で読み取りを停止します。MySQL 5.6.1 以前の配布からのスレーブレプリケーション MySQL サーバーおよびその他のリーダー (たとえば、mysqlbinlog) がバイナリログを読み取れるようにするには、ログ記録中は binlog_rows_query_log_events を無効にする必要があります。

- binlog_stmt_cache_size

コマンド行形式	--binlog-stmt-cache-size=#
導入	5.6.1
システム変数	binlog_stmt_cache_size
スコープ	グローバル

動的	はい
型	数値
デフォルト	32768
最小値	4096
最大値 (64 ビットプラットフォーム)	18446744073709551615
最大値 (32 ビットプラットフォーム)	4294967295

この変数は、トランザクション中に発行される非トランザクションステートメントを保持するバイナリログログ用キャッシュのサイズを決定します。サーバーがトランザクションストレージエンジンをサポートし、かつサーバーでバイナリログが有効（--log-bin オプション）になっている場合は、個別のバイナリログトランザクションおよびステートメントキャッシュが各クライアントに割り当てられます。トランザクション中に大きな非トランザクションステートメントを頻繁に使用する場合は、このキャッシュサイズを増やしてパフォーマンスを向上させることができます。Binlog_stmt_cache_use および Binlog_stmt_cache_disk_use ステータス変数は、この変数のサイズを調整する場合に役立つ場合があります。セクション5.2.4「バイナリログ」を参照してください。

binlog_cache_size システム変数はトランザクションキャッシュのサイズを設定します。

- log_bin

システム変数	log_bin
スコープ	グローバル
動的	いいえ

バイナリログが有効かどうか。--log-bin オプションが使用されている場合、この変数の値は ON です。そうでない場合、OFF です。この変数は、バイナリロギングのステータス (有効または無効) についてのみ報告します。--log-bin が設定されている値を実際に報告するわけではありません。

セクション5.2.4「バイナリログ」を参照してください。

- log_bin_basename

導入	5.6.2
システム変数	log_bin_basename
スコープ	グローバル
動的	いいえ
型	ファイル名
デフォルト	datadir + '/' + hostname + '-bin'

バイナリログファイルの名前と完全パスを保持します。log_bin システム変数とは異なり、log_bin_basename は --log-bin サーバーオプションで設定される名前を反映します。

log_bin_basename システム変数は MySQL 5.6.2 で追加されました。

- log_bin_index

導入	5.6.4
システム変数	log_bin_index
スコープ	グローバル
動的	いいえ
型	ファイル名

バイナリログファイル名のインデックスファイル。

log_bin_index システム変数は MySQL 5.6.4 で追加されました。

- log_bin_use_v1_row_events

コマンド行形式	--log-bin-use-v1-row-events[={0|1}]
導入	5.6.6
システム変数	log_bin_use_v1_row_events
スコープ	グローバル
動的	いいえ
型	ブール
デフォルト	0

MySQL 5.6.6 以降で使用可能なバージョン 2 バイナリロギングが使用されているかどうかを示します。値 1 は、サーバーがバージョン 1 ロギングイベント（MySQL 5.6.5 および以前の MySQL Server リリースで使用されるバイナリログイベントの唯一のバージョン）を使用してバイナリログを書き出していて、古いスレーブで読み取れるバイナリログを生成していることを示します。値 0 は、バージョン 2 バイナリログイベントが使用されていることを示します。

この変数は読み取り専用です。バージョン 1 および バージョン 2 バイナリイベントバイナリロギングを切りかえるには、mysqld を --log-bin-use-v1-row-events オプションで再起動する必要があります。

MySQL Cluster レプリケーションのアップグレードを実行するとき以外では、--log-bin-use-v1-events は主に、NDB$EPOCH_TRANS() (バージョン 2 バイナリ行イベントロギングが必要) を使用してレプリケーション競合検出および解決をセットアップときに役立ちます。したがって、このオプションと --ndb-log-transaction-id は互換性がありません。

> **注記**
>
> MySQL Cluster NDB 7.3 以降はデフォルトでバージョン 2 バイナリログ行イベントを使用します。アップグレードまたはダウングレードを計画するとき、および MySQL Cluster Replication を使用してセットアップする場合は、このことに留意してください。

詳細は、セクション18.6.11「MySQL Cluster レプリケーションの競合解決」を参照してください。

- log_slave_updates

コマンド行形式	--log-slave-updates
システム変数	log_slave_updates
スコープ	グローバル
動的	いいえ
型	ブール
デフォルト	FALSE

スレーブサーバーがマスターサーバーから受け取った更新のログをスレーブ独自のバイナリログに記録する必要があるかどうか。この変数を有効にするには、スレーブでバイナリロギングを有効にする必要があります。セクション17.1.4「レプリケーションおよびバイナリロギングのオプションと変数」を参照してください。

- master_verify_checksum

導入	5.6.2
システム変数	master_verify_checksum
スコープ	グローバル
動的	はい
型	ブール
デフォルト	OFF

この変数を有効にすることで、マスターはバイナリログから読み取るときにチェックサムを検査します。master_verify_checksum はデフォルトでは無効になっています。その場合は、マスターはバイナリログか

らのイベント長を使用してイベントを検証するため、すべてそろったイベントだけがバイナリログから読み取られます。

この変数は MySQL 5.6.2 で追加されました。

- max_binlog_cache_size

コマンド行形式	--max-binlog-cache-size=#
システム変数	max_binlog_cache_size
スコープ	グローバル
動的	はい
型	数値
デフォルト	18446744073709551615
最小値	4096
最大値	18446744073709551615

トランザクション内の非トランザクションステートメントがこのバイト数より多くのメモリーを必要とする場合、サーバーはMulti-statement transaction required more than 'max_binlog_cache_size' bytes of storageエラーを生成します。最小値は 4096 です。可能な最大値は 16E バイト (エクサバイト) です。推奨される最大値は 4G バイトです。これは、MySQL が現在 4G バイトより大きなバイナリログ位置で作業できないという事実によります。

> **注記**
>
> MySQL 5.6.7 より前では、64 ビット Windows プラットフォームは、この変数に格納された値を 4G に切り捨てました (これより大きい値に設定されたとしても) (Bug #13961678)。

max_binlog_cache_size はトランザクションキャッシュのみのサイズを設定します。ステートメントキャッシュの上限値は max_binlog_stmt_cache_size システム変数によって管理されます。

MySQL 5.6 では、max_binlog_cache_size のセッションの可視性は binlog_cache_size システム変数のものと一致します。つまり、この値の変更は、値が変更されたあとに起動された新しいセッションにのみ影響します。

- max_binlog_size

コマンド行形式	--max-binlog-size=#
システム変数	max_binlog_size
スコープ	グローバル
動的	はい
型	数値
デフォルト	1073741824
最小値	4096
最大値	1073741824

バイナリログへの書き込みによって、現在のログファイルサイズがこの変数の値を超えた場合、サーバーはバイナリログをローテーションします (現在のファイルを閉じて、新しいものを開きます)。最小値は 4096 バイトです。最大値およびデフォルト値は 1G バイトです。

トランザクションはバイナリログにひとまとまりで書き込まれ、複数のバイナリログ間に分割されることはありません。このため、大きなトランザクションの場合、max_binlog_size より大きいバイナリログファイルが見られることがあります。

max_relay_log_size が 0 の場合、max_binlog_size の値がリレーログにも適用されます。

- max_binlog_stmt_cache_size

コマンド行形式	--max-binlog-stmt-cache-size=#
導入	5.6.1

システム変数	max_binlog_stmt_cache_size
スコープ	グローバル
動的	はい
型	数値
デフォルト	18446744073709547520
最小値	4096
最大値	18446744073709547520

トランザクション内の非トランザクションステートメントがこのバイト数より多くのメモリーを必要とする場合、サーバーはエラーを生成します。最小値は 4096 です。最大値およびデフォルト値は、32 ビットプラットフォームでは 4G バイト、64 ビットプラットフォームでは 16E バイト (エクサバイト) です。

> **注記**
>
> MySQL 5.6.7 より前では、64 ビット Windows プラットフォームは、この変数に格納された値を 4G に切り捨てました (これより大きい値に設定されたとしても) (Bug #13961678)。

max_binlog_stmt_cache_size はステートメントキャッシュのみのサイズを設定します。トランザクションキャッシュの上限値は max_binlog_cache_size システム変数によって排他的に管理されます。

- sync_binlog

コマンド行形式	--sync-binlog=#
システム変数	sync_binlog
スコープ	グローバル
動的	はい
型	数値
デフォルト	0
最小値	0
最大値	4294967295

この変数の値が 0 より大きい場合は、sync_binlog コミットグループがバイナリログに書き込まれたあとに、MySQL サーバーはそのバイナリログをディスクに同期します (fdatasync() を使用)。sync_binlog のデフォルト値は 0 で、これはディスクに同期しません。この場合、サーバーはオペレーティングシステムに依存して、ほかのファイルに関してバイナリログの内容をときどきフラッシュします。値 1 が一番安全な選択です (クラッシュの場合にバイナリログから失われるコミットグループが最大で 1 つです)。しかし、一番遅い選択でもあります (ディスクにバッテリ付きキャッシュがある場合を除きます。その場合は同期が非常に速くなります)。

17.1.4.5 グローバルトランザクション ID のオプションと変数

GTID レプリケーションで使用される起動オプション

GTID レプリケーションで使用されるシステム変数

このセクションで説明した MySQL Server オプションおよびシステム変数は、MySQL 5.6.5 で導入されたグローバルトランザクション識別子 (GTID) のモニターと制御に使用されます。

> **注記**
>
> これらのオプションと変数の多くは、MySQL 5.6.9 で名前が変わりました。詳細については、このセクションの説明を参照してください。

追加情報については セクション17.1.3「グローバルトランザクション識別子を使用したレプリケーション」を参照してください。

GTID レプリケーションで使用される起動オプション

次のサーバー起動オプションは GTID ベースレプリケーションで使用されます。

- --disable-gtid-unsafe-statements

コマンド行形式	--disable-gtid-unsafe-statements[=value]
導入	5.6.5
削除	5.6.9
システム変数	disable_gtid_unsafe_statements
スコープ	グローバル
動的	いいえ
型	ブール
デフォルト	false

廃止: MySQL 5.6.9 で --enforce-gtid-consistency に置き換わりました。(Bug #14775984)

- --enforce-gtid-consistency

コマンド行形式	--enforce-gtid-consistency[=value]
導入	5.6.9
システム変数	enforce_gtid_consistency
スコープ	グローバル
動的	いいえ
型	ブール
デフォルト	false

このオプションが有効のときは、トランザクションセーフな方法でログを記録できるステートメントのみの実行を許可することで、GTID 一貫性を適用します。--enforce-gtid-consistency を有効にしてから --gtid-mode を ON に設定する必要があります。そうしないで GTID モードを有効にすると、エラーで失敗します。システムが GTID を使用する準備ができているかどうかをテストするために、--gtid-mode を使用する前にこのオプションを使用できます (使用することをお勧めします)。

--enforce-gtid-consistency が有効のときは、トランザクションセーフなステートメントのみのログを記録できるため、次に示す操作はこのオプションで使用できないことになります。

- CREATE TABLE ... SELECT ステートメント
- トランザクション内の CREATE TEMPORARY TABLE ステートメント
- トランザクションおよび非トランザクションテーブルの両方を更新するトランザクションまたはステートメント。

MySQL 5.6.9 より前では、このオプションの名前は --disable-gtid-unsafe-statements でした。(Bug #14775984)

MySQL 5.6.7 より前では、このオプションを使用することで、一時テーブルでの非トランザクション DML が失敗しました (行ベースバイナリロギングを使用するときに一時テーブルへの変更のログが記録されない)。MySQL 5.6.7 以降では、影響されるすべてのテーブルが一時テーブルであるかぎり、--disable-gtid-unsafe-statements (MySQL 5.6.9 以降では --enforce-gtid-consistency) の状態で非トランザクション DML ステートメントが一時テーブルで許可されます (Bug #14272672)。

MySQL 5.6.7 より前では、--write-binlog が明示的に無効の状態で mysql_upgrade が実行されていないかぎり、mysql_upgrade をこのオプションが有効の状態で動作する MySQL Server で使用できませんでした。(Bug #13833710、Bug #14221043) MySQL 5.6.7 以降では、--gtid-mode=ON の状態のサーバー上で mysql_upgrade を実行できますが、推奨されていません。MySQL システムテーブルが、非トランザクションである MyISAM ストレージエンジンを使用するためです。

MySQL 5.6.8 以前では、--enforce-gtid-consistency が使用されるとき (このオプションは --disable-gtid-unsafe-statements と呼ばれていました) に、非トランザクションテーブルに影響するステートメントを使用できませんでした。MySQL 5.6.9 以降では、このオプションは非トランザクションテーブルを更新する単一ステートメントを許可します。これは主に、mysql_install_db や mysql_upgrade などのプログラムで使用されることを意図しています。(Bug #14722659)

- --gtid-mode

コマンド行形式	--gtid-mode=MODE
導入	5.6.5
システム変数	gtid_mode
スコープ	グローバル
動的	いいえ
型	列挙
デフォルト	OFF
有効な値	OFF UPGRADE_STEP_1 UPGRADE_STEP_2 ON

このオプションは GTID が有効であるかどうかを指定します。--gtid-mode=ON でサーバーを起動するには、サーバーも --log-bin、--log-slave-updates、および --enforce-gtid-consistency オプションで起動される必要があります。

このオプションを、バイナリログまたはリレーログ内に GTID があるときに OFF に、または実行すべき無名トランザクションが残っているときに ON に設定すると、エラーになります。

> **重要**
>
> このオプションはブール値を採用しません。つまり、値は列挙されます。このオプションを設定するときに数値の使用を試みないでください。予期しない結果になることがあります。値 UPGRADE_STEP_1 および UPGRADE_STEP_2 は今後の使用のために予約されていますが、現在のところ本番環境ではサポートされていません。これら 2 つの値のいずれかを --gtid-mode で使用する場合、サーバーは起動を拒否します。

MySQL 5.6.7 より前では、--write-binlog が明示的に無効の状態で mysql_upgrade が実行されていないかぎり、mysql_upgrade をこのオプションが有効の状態で動作する MySQL Server で使用できませんでした。(Bug #13833710、Bug #14221043) MySQL 5.6.7 以降では、--gtid-mode=ON の状態のサーバー上で mysql_upgrade を実行できますが、推奨されていません。非トランザクションである MyISAM ストレージエンジンを使用する MySQL システムテーブルに変更を加える可能性があるためです。

MySQL 5.6.10 より前では、--gtid-mode が ON に設定されているときに sql_slave_skip_counter 変数のグローバル値を 1 に設定しても、効果はありませんでした。(Bug #15833516) MySQL 5.6.9 以前のバージョンでの回避策は、CHANGE MASTER TO ... MASTER_LOG_FILE = ... MASTER_LOG_POS = ... を使用して (必要に応じてこのステートメントに MASTER_AUTO_POSITION = 0 オプションを含めて)、スレーブの位置をリセットすることです。

GTID レプリケーションで使用されるシステム変数

次のシステム変数は GTID ベースレプリケーションで使用されます。

- disable_gtid_unsafe_statements

コマンド行形式	--disable-gtid-unsafe-statements[=value]
導入	5.6.5
削除	5.6.9
システム変数	disable_gtid_unsafe_statements
スコープ	グローバル
動的	いいえ
型	ブール
デフォルト	false

廃止: MySQL 5.6.9 で enforce_gtid_consistency に置き換わりました。(Bug #14775984)

- gtid_done

導入	5.6.5
削除	5.6.9
システム変数	gtid_done
スコープ	グローバル、セッション
動的	いいえ
型	文字列
Unit	set of GTIDs

廃止: MySQL 5.6.9 で gtid_executed に置き換わりました。(Bug #14775984)

- enforce_gtid_consistency

コマンド行形式	--enforce-gtid-consistency[=value]
導入	5.6.9
システム変数	enforce_gtid_consistency
スコープ	グローバル
動的	いいえ
型	ブール
デフォルト	false

この変数が true のときは、トランザクションセーフな方法でログを記録できるステートメントのみの実行を許可することで、サーバーは GTID 一貫性を適用します。サーバーを --gtid-mode=ON で起動する前に、GTID 一貫性を有効にする (--enforce-gtid-consistency を使用することで) 必要があります。そうしないで GTID モードを有効にすると、エラーで失敗します。システムが GTID を使用する準備ができているかどうかをテストするために、--gtid-mode を使用する前に GTID 一貫性を有効にできます (お勧めします)。

enforce_gtid_consistency が true のときは、トランザクションセーフなステートメントのみのログを記録できるため、次に示す操作はこの場合には使用できないことになります。

- CREATE TABLE ... SELECT ステートメント
- トランザクション内の CREATE TEMPORARY TABLE ステートメント
- トランザクションおよび非トランザクションテーブルの両方を更新するトランザクションまたはステートメント。

この変数は読み取り専用です。これを設定するには、MySQL Server の起動時にコマンド行またはオプションファイルで --enforce-gtid-consistency オプションを使用してください。

MySQL 5.6.9 より前では、この変数の名前は disable_gtid_unsafe_statements でした。(Bug #14775984)

- gtid_executed

導入	5.6.9
システム変数	gtid_executed
スコープ	グローバル、セッション
動的	いいえ
型	文字列

Unit	set of GTIDs

この変数は、グローバルスコープで使用されるとき、バイナリログにログが記録されるすべてのトランザクションセットの表現を含みます。これは、SHOW MASTER STATUS および SHOW SLAVE STATUS の出力内の Executed_Gtid_Set カラムの値と同じです。

この変数は、セッションスコープで使用されるとき、現在のセッションのキャッシュに書き込まれるすべてのトランザクションセットの表現を含みます。

任意の時点のバイナリログで見つかるトランザクションのセットは、GTID_SUBTRACT(@@GLOBAL.gtid_executed, @@GLOBAL.gtid_purged) に等しいです (つまり、バイナリログ内で、まだパージされていないすべてのトランザクション)。

サーバーが起動すると、@@GLOBAL.gtid_executed は次の 2 つのセットの和集合に初期化されます。

- 一番新しいバイナリログの Previous_gtids_log_event にリストされる GTID
- 一番新しいバイナリログ内のすべての Gtid_log_event で見つかる GTID

これ以降、トランザクションが実行されるときに GTID がそのセットに追加されます。

RESET MASTER を発行することで、この変数のグローバル値 (ただし、セッション値ではない) は空の文字列にリセットされます。GTID は、セットが RESET MASTER によってクリアされるときを除いてセットから削除されません。セットは、サーバーがシャットダウンされてすべてのバイナリログが削除される場合にもクリアされます。

MySQL 5.6.9 より前では、この変数は gtid_done と呼ばれていました。

- gtid_lost

導入	5.6.5
削除	5.6.9
システム変数	gtid_lost
スコープ	グローバル
動的	いいえ
型	文字列
Unit	set of GTIDs

廃止: MySQL 5.6.9 で gtid_purged に置き換わりました。(Bug #14775984)

- gtid_mode

導入	5.6.5
システム変数	gtid_mode
スコープ	グローバル
動的	いいえ
型	列挙
デフォルト	OFF
有効な値	OFF UPGRADE_STEP_1 UPGRADE_STEP_2 ON

GTID が有効かどうかを示します。読み取り専用。--gtid-mode を使用して設定します。

- gtid_next

システム変数	gtid_next
スコープ	セッション
動的	はい
型	列挙
デフォルト	AUTOMATIC
有効な値	AUTOMATIC ANONYMOUS UUID:NUMBER

この変数は、次の GTID が取得されたかどうか、およびどのように取得されたかを指定するために使用されます。gtid_next は次の値のいずれかを取ることができます。

- AUTOMATIC: 自動的に生成される次のグローバルトランザクション ID を使用します。
- ANONYMOUS: トランザクションはグローバル識別子を持たず、ファイルと位置のみで識別されます。
- UUID:NUMBER 形式のグローバルトランザクション ID。

この変数を設定するには、SUPER 権限が必要です。gtid_mode が OFF の場合、この変数を設定しても効果はありません。

MySQL 5.6.20 より前は、GTID は有効でも gtid_next が AUTOMATIC でない場合、DROP TABLE は、非一時テーブルと一時テーブル、またはトランザクションストレージエンジンを使用する一時テーブルと非トランザクションストレージエンジンを使用する一時テーブルの組み合わせで使用されたときに、正しく機能しません。MySQL 5.6.20 以降では、DROP TABLE または DROP TEMPORARY TABLE は、テーブルのこれらの組み合わせのいずれかで使用されると明示的なエラーで失敗します。(Bug #17620053)

MySQL 5.6.11 だけですが、gtid_next が AUTOMATIC 以外の値に設定されているとき、ステートメント CHANGE MASTER TO、START SLAVE、STOP SLAVE、REPAIR TABLE、OPTIMIZE TABLE、ANALYZE TABLE、CHECK TABLE、CREATE SERVER、ALTER SERVER、DROP SERVER、CACHE INDEX、LOAD INDEX INTO CACHE、FLUSH、または RESET を実行できません。このような場合、ステートメントはエラーで失敗します。このようなステートメントは、MySQL 5.6.12 以降では拒否されません。(Bug #16062608、Bug #16715809、Bug #69045)

- gtid_owned

導入	5.6.5
システム変数	gtid_owned
スコープ	グローバル、セッション
動的	いいえ
型	文字列
Unit	set of GTIDs

この読み取り専用変数は、内容がそのスコープに依存するリストを保持します。セッションスコープで使用されるときは、リストはこのクライアントが所有するすべての GTID を保持します。グローバルスコープで使用されるときは、すべての GTID とその所有者のリストを保持します。

- gtid_purged

導入	5.6.9
システム変数	gtid_purged
スコープ	グローバル
動的	はい
型	文字列

Unit	set of GTIDs

バイナリログからパージされたすべてのトランザクションのセット。

サーバーが起動するときに、gtid_purged のグローバル値は、一番古いバイナリログの Previous_gtid_log_event に含まれる GTID のセットに初期化されます。バイナリログがパージされると、一番古いものになったバイナリログから @@GLOBAL.gtid_purged が再度読み取られます。

RESET MASTER を発行することで、この変数の値は空の文字列にリセットされます。

MySQL 5.6.9 より前では、この変数は gtid_lost と呼ばれ、読み取り専用でした。MySQL 5.6.9 以降では、この変数の値を更新できます (ただし、すでにリストされたものに GTID を追加することでのみ、および gtid_executed が設定解除されたとき、つまり新しいサーバー上でのみ)。(Bug #14797808)

- simplified_binlog_gtid_recovery

コマンド行形式	--simplified-binlog-gtid-recovery
導入	5.6.21
非推奨	5.6.23
システム変数	simplified_binlog_gtid_recovery
スコープ	グローバル
動的	いいえ
型	ブール
デフォルト	FALSE

このオプションは非推奨で、今後の MySQL リリースで削除される予定です。名前が変更された binlog_gtid_recovery_simplified を使用して、クラッシュ後に MySQL がバイナリログファイルをどのように反復するかを制御します。

17.1.5 一般的なレプリケーション管理タスク

レプリケーションが開始されると、定期的な管理をあまり行わなくても実行されるはずです。レプリケーション環境に応じて、各スレーブのレプリケーションステータスを定期的に (毎日、またはさらに頻繁に) 確認することをお勧めします。

17.1.5.1 レプリケーションステータスの確認

レプリケーションプロセスを管理するときにもっとも一般的なタスクは、レプリケーションが実行中であること、およびスレーブとマスターとの間でエラーがないことを確認することです。このための主要なステートメントは SHOW SLAVE STATUS で、各スレーブで実行する必要があります。

```
mysql> SHOW SLAVE STATUS\G
*************************** 1. row ***************************
               Slave_IO_State: Waiting for master to send event
                  Master_Host: master1
                  Master_User: root
                  Master_Port: 3306
                Connect_Retry: 60
              Master_Log_File: mysql-bin.000004
          Read_Master_Log_Pos: 931
               Relay_Log_File: slave1-relay-bin.000056
                Relay_Log_Pos: 950
        Relay_Master_Log_File: mysql-bin.000004
             Slave_IO_Running: Yes
            Slave_SQL_Running: Yes
              Replicate_Do_DB:
          Replicate_Ignore_DB:
           Replicate_Do_Table:
       Replicate_Ignore_Table:
      Replicate_Wild_Do_Table:
  Replicate_Wild_Ignore_Table:
                   Last_Errno: 0
                   Last_Error:
                 Skip_Counter: 0
          Exec_Master_Log_Pos: 931
              Relay_Log_Space: 1365
              Until_Condition: None
               Until_Log_File:
```

```
           Until_Log_Pos: 0
        Master_SSL_Allowed: No
        Master_SSL_CA_File:
        Master_SSL_CA_Path:
           Master_SSL_Cert:
         Master_SSL_Cipher:
            Master_SSL_Key:
     Seconds_Behind_Master: 0
Master_SSL_Verify_Server_Cert: No
             Last_IO_Errno: 0
             Last_IO_Error:
            Last_SQL_Errno: 0
            Last_SQL_Error:
 Replicate_Ignore_Server_Ids: 0
```

ステータスレポートの中で調査すべき主要フィールドは、次のとおりです。

- Slave_IO_State: スレーブの現在のステータス。詳しくは、セクション8.12.5.6「レプリケーションスレーブの I/O スレッド状態」およびセクション8.12.5.7「レプリケーションスレーブ SQL スレッドの状態」を参照してください。
- Slave_IO_Running: マスターのバイナリログを読み取るための I/O スレッドが実行中かどうか。レプリケーションをまだ起動していないか STOP SLAVE で明示的に停止した場合を除き、通常はこれを Yes にすることをお勧めします。
- Slave_SQL_Running: リレーログでイベントを実行するための SQL スレッドが実行中かどうか。I/O スレッドと同様、これは通常は Yes にすることをお勧めします。
- Last_IO_Error、Last_SQL_Error: リレーログを処理するときに I/O および SQL スレッドによって登録された最後のエラー。理想的には、これらはエラーがないことを示すブランクであるべきです。
- Seconds_Behind_Master: スレーブ SQL スレッドがマスターバイナリログの処理より何秒遅れているか。高い (または増加する) 数値は、スレーブがマスターからのイベントを適時に処理できないことを示している可能性があります。

  Seconds_Behind_Master が値 0 の場合は通常、スレーブがマスターに追い付いたことを意味すると解釈できますが、これが厳密には正しくない場合がいくつかあります。たとえば、マスターとスレーブの間のネットワーク接続が切断されたけれども、スレーブ I/O スレッドがまだこれを通知されていない場合、つまり slave_net_timeout がまだ経過していない場合に、これが発生する可能性があります。

  Seconds_Behind_Master の過渡値が状況を正確に反映しない場合もあります。スレーブ SQL スレッドが I/O に追い付くと、Seconds_Behind_Master は 0 を表示しますが、スレーブ I/O スレッドがまだ新しいイベントをキューに入れているときは、Seconds_Behind_Master は SQL スレッドが新しいイベントの実行を終了するまで大きな値を示す場合があります。これは特に、イベントのタイムスタンプが古い場合に発生する可能性があります。そのような場合、比較的短い期間に数回 SHOW SLAVE STATUS 実行すると、この値が 0 と比較的大きな値との間を前後に繰り返し変化するのが見える可能性があります。

フィールドのいくつかのペアは、スレーブがマスターバイナリログからイベントを読み取り、リレーログでそれらを処理するときの進捗状況に関する情報を提供します。

- (Master_Log_file、Read_Master_Log_Pos): スレーブ I/O スレッドがマスターバイナリログからどのくらい離れてイベントを読み取ったかを示す、ログ内の座標。
- (Relay_Master_Log_File、Exec_Master_Log_Pos): スレーブ SQL スレッドがマスターバイナリログから受け取ったイベントをどのくらい離れて実行したかを示す、ログ内の座標。
- (Relay_Log_File、Relay_Log_Pos): スレーブ SQL スレッドがスレーブリレーログをどのくらい離れて実行したかを示す、リレーログ内の座標。これらは前述の座標に対応しますが、マスターバイナリログ座標ではなく、スレーブリレーログ座標で表現されます。

マスターでは、接続されたスレーブのステータスを SHOW PROCESSLIST を使用して確認することで、実行中プロセスのリストを確認できます。スレーブ接続では、Command フィールドに Binlog Dump があります。

```
mysql> SHOW PROCESSLIST \G;
*************************** 4. row ***************************
     Id: 10
   User: root
   Host: slave1:58371
     db: NULL
Command: Binlog Dump
   Time: 777
  State: Has sent all binlog to slave; waiting for binlog to be updated
   Info: NULL
```

レプリケーションプロセスを実行するのはスレーブであるため、このレポートで入手できる情報はほとんどありません。

--report-host オプションで起動されて、マスターに接続されているスレーブの場合は、マスターでの SHOW SLAVE HOSTS ステートメントはスレーブに関する基本情報を示します。出力には、スレーブサーバーの ID、--report-host オプションの値、接続中のポート、マスター ID が含まれます。

```
mysql> SHOW SLAVE HOSTS;
+-----------+--------+------+------------------+-----------+
| Server_id | Host   | Port | Rpl_recovery_rank | Master_id |
+-----------+--------+------+------------------+-----------+
|        10 | slave1 | 3306 |                0 |         1 |
+-----------+--------+------+------------------+-----------+
1 row in set (0.00 sec)
```

17.1.5.2 スレーブでレプリケーションを一時停止する

STOP SLAVE および START SLAVE ステートメントを使用して、スレーブでステートメントのレプリケーションを停止したり開始したりできます。

マスターからバイナリログの処理を停止するには、STOP SLAVE を使用します。

```
mysql> STOP SLAVE;
```

レプリケーションが停止されると、スレーブ I/O スレッドはマスターバイナリログからイベントを読み取りそれらをリレーログに書き込むのを停止し、SQL スレッドはリレーログからイベントの読み取りそれらを実行するのを停止します。スレッドタイプを指定することで、I/O または SQL スレッドを個別に一時停止できます。

```
mysql> STOP SLAVE IO_THREAD;
mysql> STOP SLAVE SQL_THREAD;
```

実行を再開するには、START SLAVE ステートメントを使用します。

```
mysql> START SLAVE;
```

特定のスレッドを開始するには、スレッドタイプを指定します。

```
mysql> START SLAVE IO_THREAD;
mysql> START SLAVE SQL_THREAD;
```

マスターからイベントを処理することでのみ更新を実行するスレーブの場合、SQL スレッドのみを停止することが、バックアップまたはほかのタスクを実行する場合に役立つ可能性があります。I/O スレッドは引き続きマスターからイベントを読み取りますが、それらは実行されません。これにより、SQL スレッドを再起動するときに、スレーブが追い付くことが簡単になります。

I/O スレッドだけを停止することで、SQL スレッドはリレーログが終了したポイントまでリレーログ内のイベントを実行できます。マスターからすでに受け取ったイベントに追い付くために実行を停止したいとき、スレーブで管理を実行したいけれども特定のポイントまですべての更新を処理したことを確認したいときに、これが役立つ場合があります。この方法は、マスターで管理を実行中に、スレーブでイベント受け取りを一時停止するときにも使用できます。I/O スレッドは停止するけれども SQL スレッドの実行を許可することで、レプリケーションが再開したときに実行される大量のイベントバックログを確実になくすのに役立ちます。

17.2 レプリケーションの実装

レプリケーションは、データベースへのあらゆる変更 (更新、削除など) をバイナリログで追跡するマスターサーバーに基づきます。バイナリログは、サーバーが起動した瞬間からデータベースの構造または内容 (データ) を変更するあらゆるイベントが書き込まれた記録として機能します。SELECT ステートメントは通常、データベースの構造および内容を変更しないため記録されません。

マスターに接続する各スレーブはバイナリログのコピーを要求します。つまり、マスターがスレーブにデータをプッシュのではなく、スレーブがマスターからデータをプルします。スレーブはまた、それが受け取るバイナリログからイベントを実行します。これによって、元の変更がマスターで実行されたように繰り返されることになります。マスターで最初に実行された変更に応じて、テーブルが作成されたり、それらの構造が変更されたり、データが挿入、削除、更新されたりします。

各スレーブは独立しているので、マスターのバイナリログからの変更の再生が、マスターに接続された各スレーブで単独に発生します。さらに、各スレーブはバイナリログのコピーをマスターから要求することでのみ受け取るため、スレーブはそれ自身のペースでデータベースのコピーを読み取ったり更新したりでき、マスター側またはスレーブ側で最新のデータベースステータスを更新できる能力に影響を与えずにレプリケーションプロセスを自由に開始したり停止したりできます。

レプリケーション実装の仕様に関する詳細は、セクション17.2.1「レプリケーション実装の詳細」を参照してください。

マスターとスレーブは、レプリケーションプロセスに関するステータスをモニターできるように、定期的にそれらを報告します。すべてのレプリケーション関連ステータスの説明については、セクション8.12.5「スレッド情報の検査」を参照してください。

マスターバイナリログは、スレーブ上のローカルリレーログに書き込まれてから処理されます。スレーブは、マスターのバイナリログとローカルリレーログと一緒に、現在の位置に関する情報も記録します。セクション17.2.2「レプリケーションリレーおよびステータスログ」を参照してください。

データベース変更は、さまざまな構成オプションおよびイベント評価を制御する変数に従って適用されるルールセットに応じて、スレーブ上でフィルタされます。これらのルールがどのように適用されるかについての詳細は、セクション17.2.3「サーバーがレプリケーションフィルタリングルールをどのように評価するか」を参照してください。

17.2.1 レプリケーション実装の詳細

MySQL レプリケーション機能は、3 つのスレッド (マスターサーバー上で 1 つ、スレーブ上で 2 つ) を使用して実装されます。

- Binlog ダンプスレッド　マスターは、スレーブが接続したときにバイナリログ内容をスレーブに送信するスレッドを作成します。このスレッドは、マスター上の SHOW PROCESSLIST の出力内で Binlog Dump スレッドとして識別できます。

  バイナリログダンプスレッドは、スレーブに送信される各イベントを読み取るために、マスターのバイナリログでロックを獲得します。イベントが読み取られるとすぐにロックは解除されます (イベントがスレーブに送信される前でも)。

- スレーブ I/O スレッド　START SLAVE ステートメントがスレーブサーバー上で発行されると、スレーブは I/O スレッドを作成し、これがマスターに接続されてそのバイナリログに記録された更新を送信することを要求します。

  スレーブ I/O スレッドは、マスターの Binlog Dump スレッドが送信する更新を読み取って (前の項目を参照)、それらをスレーブのリレーログで構成されるローカルファイルにコピーします。

  このスレッドの状態は、SHOW SLAVE STATUS の出力では Slave_IO_running として、SHOW STATUS の出力では Slave_running として表示されます。

- スレーブ SQL スレッド　スレーブは、スレーブ I/O スレッドによって書き込まれるリレーログを読み取り、それらに含まれるイベントを実行する SQL スレッドを作成します。

マスター/スレーブ接続ごとに 3 つのスレーブがあることをすでに説明しました。複数のスレーブを持つマスターは、現在接続されているスレーブごとに 1 つのバイナリログダンプスレッドを作成し、スレーブごとに独自の I/O および SQL スレッドがあります。

スレーブは 2 つのスレッドを使用して、マスターからの更新を読み取ることとそれらを実行することを、独立タスクに分類します。このようにしても、ステートメント実行が遅い場合でも、ステートメントを読み取るタスクが遅くなることはありません。たとえば、スレーブサーバーがしばらくの間実行されなかった場合、その I/O スレッドはスレーブ起動時にマスターからすべてのバイナリログ内容をすばやくフェッチできます (SQL スレッドがかなり遅れていても)。SQL スレッドがすべてのフェッチ済みステートメントの実行を完了する前にスレーブが停止した場合でも、I/O スレッドは少なくともすべてのものをフェッチしているため、ステートメントの安全なコピーがスレーブのリレーログにローカルに保存されていて、次にスレーブが起動するときに実行できる状態です。これによって、マスターサーバーはそのバイナリログをすぐにパージできます (スレーブがそれらの内容をフェッチするのをそれ以上待機する必要がないため)。

SHOW PROCESSLIST ステートメントは、レプリケーションに関してマスターおよびスレーブ上で何が起きているかを伝える情報を提供します。マスター状態については、セクション8.12.5.5「レプリケーションマスタースレッドの状態」を参照してください。スレーブ状態については、セクション8.12.5.6「レプリケーションスレーブの I/O スレッド状態」およびセクション8.12.5.7「レプリケーションスレーブ SQL スレッドの状態」を参照してください。

次の例では、3 つのスレッドが SHOW PROCESSLIST からの出力でどのように表示されるかを示します。

マスターサーバーでは、SHOW PROCESSLIST からの出力は次のようになります。

```
mysql> SHOW PROCESSLIST\G
*************************** 1. row ***************************
```

```
    Id: 2
  User: root
  Host: localhost:32931
    db: NULL
Command: Binlog Dump
  Time: 94
 State: Has sent all binlog to slave; waiting for binlog to
        be updated
  Info: NULL
```

ここで、スレッド 2 は接続されたスレーブにサービスを提供する Binlog Dump レプリケーションスレッドです。State 情報は、すべての未処理更新がスレーブに送信され、マスターは後続の更新の発生を待機していることを示しています。マスターサーバー上で Binlog Dump スレッドが見られない場合、これは、レプリケーションが実行中でない、つまり現在スレーブが接続されていないことを意味します。

スレーブサーバーでは、SHOW PROCESSLIST からの出力は次のようになります。

```
mysql> SHOW PROCESSLIST\G
*************************** 1. row ***************************
    Id: 10
  User: system user
  Host:
    db: NULL
Command: Connect
  Time: 11
 State: Waiting for master to send event
  Info: NULL
*************************** 2. row ***************************
    Id: 11
  User: system user
  Host:
    db: NULL
Command: Connect
  Time: 11
 State: Has read all relay log; waiting for the slave I/O
        thread to update it
  Info: NULL
```

State 情報は、スレッド 10 がマスターサーバーと通信している I/O スレッド、スレッド 11 がリレーログに格納された更新を処理している SQL スレッドであることを示しています。SHOW PROCESSLIST が実行された時点で、両方のスレッドはアイドルで、後続の更新を待機中でした。

Time カラムの値は、マスターと比較してスレーブがどのくらい遅れているかを示すことができます。セクション A.13「MySQL 5.6 FAQ: レプリケーション」を参照してください。マスター側で Binlog Dump スレッドの活動がない状態で十分な時間が経過した場合、マスターはスレーブがもう接続されていないと判断します。ほかのクライアント接続に関して、これのタイムアウトは net_write_timeout および net_retry_count の値によって異なります。これらの詳細については、セクション5.1.4「サーバーシステム変数」を参照してください。

SHOW SLAVE STATUS ステートメントは、スレーブサーバー上のレプリケーション処理に関する追加情報を提供します。セクション17.1.5.1「レプリケーションステータスの確認」を参照してください。

17.2.2 レプリケーションリレーおよびステータスログ

レプリケーション中にスレーブサーバーは、マスターからスレーブにリレーされるバイナリログイベントを保持し、現在のステータスとリレーログ内の位置に関する情報を記録するいくつかのログを作成します。処理に使用される口グには、ここで示すように 3 つのタイプがあります。

- リレーログは、スレーブ I/O スレッドによってマスターのバイナリログから読み取られ、書き込まれるイベントから構成されます。リレーログ内のイベントは、SQL スレッドの一部としてスレーブ上で実行されます。
- マスター情報ログ には、スレーブのマスターへの接続に関するステータスと現在の構成情報が含まれます。このログは、マスターホスト名、ログイン資格証明、およびスレーブがマスターのバイナリログからどのくらい離れて読み取ったかを示す座標に関する情報を保持します。

  MySQL 5.6 より前では、このログは常にファイル (master.info) でしたが、MySQL 5.6 以降では、スレーブを --master-info-repository=TABLE で起動することで、このログをファイルにではなく mysql.slave_master_info テーブルに書き込むことができます。
- リレーログ情報 は、スレーブのリレーログ内の実行ポイントに関するステータス情報を保持します。

  MySQL 5.6 より前では、このログは常にファイル (relay-log.info) でしたが、MySQL 5.6 以降では、スレーブを --relay-log-info-repository=TABLE で起動することで、このログをファイルにではなく mysql.slave_relay_log_info テーブルに書き込むことができます。

MySQL 5.6.7 より前では、slave_master_info および slave_relay_log_info テーブルの Master_id カラムは、マスターのサーバー ID ではなくスレーブのサーバー ID を示していました。(Bug #12334346)

クラッシュセーフレプリケーション　ステータスおよびリレー情報のログのためにテーブルを使用するときにレプリケーションがクラッシュセーフであるためには、これらのテーブルは InnoDB などのトランザクションストレージエンジンを使用する必要があります。MySQL 5.6.6 以降では、これらのテーブルは InnoDB を使用して作成されます。(Bug #13538891)

このため、スレーブでのクラッシュ安全性を保証するには、--relay-log-recovery が有効な状態でスレーブを実行し、さらに --relay-log-info-repository を TABLE に設定する必要があります。

MySQL 5.6.6 より前では、mysqld がレプリケーションロギングテーブルを初期化できなかった場合、スレーブは起動を拒否しました。MySQL 5.6.6 以降では、これが発生すると警告が出されますが、スレーブは起動を続けることが許可されます。(Bug #13971348) この状況が発生する可能性が高いのは、スレーブロギングテーブルをサポートしないバージョンの MySQL からサポートされるバージョンにアップグレードするときです。

MySQL 5.6.5 以前では、slave_master_info および slave_relay_log_info テーブルがデフォルトで MyISAM を使用していて、これはレプリケーションを開始する前にここで示すように ALTER TABLE ... ENGINE=InnoDB を発行することで、これらのテーブルが使用するストレージエンジンを変更する必要がありました。

```
ALTER TABLE mysql.slave_master_info ENGINE=InnoDB;
ALTER TABLE mysql.slave_relay_log_info ENGINE=InnoDB;
```

ALTER TABLE ステートメントは、mysql データベースで適切な権限を持つ MySQL root またはほかのユーザーアカウントで実行される必要があります。レプリケーションの実行中にこれを実行しようとしてはいけません。MySQL 5.6.3 以降では、レプリケーションの進行中にこれらのいずれかのテーブルで ALTER TABLE を実行しようとすることは許可されません。MySQL 5.6.4 以降では、これらのテーブルのいずれかまたは両方で書き込みロックを必要とするステートメントを実行することは、レプリケーションの進行中は許可されませんが、読み取りのみを実行するステートメントはいつでも許可されます。

> **重要**
>
> slave_master_info または slave_relay_log_info テーブルで手動で行を更新または挿入しようとしないでください。そのようにすることは、未定義の動作になる可能性があり、サポートされていません。

MySQL 5.6.4 より前では、mysqldump はレプリケーションログテーブルをダンプしませんでした (それらが名前で指定されて --master-data オプションが使用された場合を除く)。

17.2.2.1 スレーブリレーログ

リレーログは、バイナリログと同様に、データベース変更を記述するイベントを含む番号付きファイルのセットと、使用されたすべてのリレーログファイルの名前を含むインデックスファイルとで構成されます。

用語「リレーログファイル」は一般的に、データベースイベントを含む個々の番号付きファイルを示します。用語「リレーログ」は、番号付きリレーログファイルとインデックスファイルのセットの総称です。

リレーログファイルはバイナリログファイルと同じ形式で、mysqlbinlog を使用して読み取ることができます (セクション4.6.8「mysqlbinlog — バイナリログファイルを処理するためのユーティリティー」を参照)。

デフォルトでは、データディレクトリ内でのリレーログファイル名の形式は host_name-relay-bin.nnnnnn です。ここで、host_name はスレーブサーバーホストの名前、nnnnnn はシーケンス番号です。連続するリレーログファイルは、000001 で始まる連続シーケンス番号を使用して作成されます。スレーブはインデックスファイルを使用して現在使用中のリレーログファイルを追跡します。データディレクトリ内でのデフォルトのリレーログインデックスファイル名は、host_name-relay-bin.index です。

デフォルトのリレーログファイルおよびリレーログインデックスファイル名はそれぞれ、--relay-log および --relay-log-index サーバーオプションでオーバーライドできます (セクション17.1.4「レプリケーションおよびバイナリロギングのオプションと変数」を参照)。

スレーブがデフォルトのホストベースリレーログファイル名を使用する場合、レプリケーションセットアップ後にスレーブのホスト名を変更すると、レプリケーションが次のエラーで失敗する可能性があります: Failed to open the relay log および Could not find target log during relay log initialization。これは既知の問題です (Bug #2122 を参照してください)。今後スレーブのホスト名が変更される見込みがある場合は (たとえば、DHCP を使用してそのホスト名を変更できるように、スレーブでネットワークがセットアップされている場合)、スレーブを最初にセットアップするときに --relay-log および --relay-log-index オプションを使用してリレーログファイル名を明示的に指定することで、この問題を完全に回避できます。これにより、サーバーホスト名の変更との間に名前の依存関係がなくなります。

レプリケーションがすでに開始されたあとに問題が発生した場合、これを回避する 1 つの方法は、スレーブサーバーを停止し、古いリレーログインデックスファイルの内容を新しいものの前に付加してからスレーブを再起動することです。Unix システムでは、ここで示すようにこれを実行できます。

```
shell> cat new_relay_log_name.index >> old_relay_log_name.index
shell> mv old_relay_log_name.index new_relay_log_name.index
```

スレーブサーバーは次の条件で新しいリレーログファイルを作成します。

- I/O スレッドが起動するたび。
- ログがフラッシュされたとき。たとえば、FLUSH LOGS または mysqladmin flush-logs で。
- 現在のリレーログファイルのサイズが「大きくなりすぎた」とき。次のように判断します。
  - max_relay_log_size (最大リレーログファイルサイズ) の値が 0 より大きい場合。
  - max_relay_log_size の値が 0 の場合、max_binlog_size が最大リレーログファイルサイズを判断します。

SQL スレッドは、各リレーログファイル内のすべてのイベントの実行し、それが不要になるとすぐに、ファイルを自動的に削除します。SQL スレッドがこの処理を担当するため、リレーログを削除するための明示的なメカニズムはありません。ただし、FLUSH LOGS はリレーログをローテーションしますが、これは SQL スレッドがいつそれらを削除するかに影響します。

17.2.2.2 スレーブステータスログ

レプリケーションスレーブサーバーは 2 つのログを作成します。デフォルトでは、これらのログは master.info および relay-log.info という名前のファイルで、データディレクトリ内に作成されます。これらのファイルの名前と場所はそれぞれ、--master-info-file および --relay-log-info-file オプションを使用して変更できます。MySQL 5.6 以降では、適切なオプションでサーバーを起動することで、これらのどちらかまたは両方のログを mysql データベース内のテーブルに書き込むこともできます: --master-info-repository を使用すると、マスター情報ログが mysql.slave_master_info テーブルに書き込まれ、--relay-log-info-repository を使用すると、リレーログ情報ログが mysql.slave_relay_log_info テーブルに書き込まれます。セクション17.1.4「レプリケーションおよびバイナリロギングのオプションと変数」を参照してください。

2 つのステータスログには、SHOW SLAVE STATUS ステートメントの出力で示されるような情報が含まれます (セクション13.4.2「スレーブサーバーを制御するための SQL ステートメント」を参照)。ステータスログはディスクに格納されるため、スレーブサーバーのシャットダウン後も存在します。スレーブは、次に起動するときに 2 つのログを読み取って、マスターからのバイナリログ読み取りがそれ自身のリレーログの処理がどのくらい進んだかを判断します。

マスター情報ログファイルまたはテーブルは、マスターに接続するためのパスワードを含んでいるため、保護することをお勧めします。セクション6.1.2.3「パスワードおよびロギング」を参照してください。

スレーブ I/O スレッドはマスター情報ログを更新します。次の表は、master.info ファイル内の行、mysql.slave_master_info テーブル内のカラム、および SHOW SLAVE STATUS で表示されるカラムの間の対応を示しています。

master.info ファイル内の行	slave_master_info テーブルのカラム	SHOW SLAVE STATUS のカラム	説明
1	Number_of_lines	[None]	ファイル内の行、またはテーブル内のカラムの数
2	Master_log_name	Master_Log_File	マスターから現在読み取られているマスターバイナリログの名前
3	Master_log_pos	Read_Master_Log_Pos	マスターから読み取られたマスターのバイナリログ内の現在の位置です
4	Host	Master_Host	マスターのホスト名
5	User_name	Master_User	マスターへの接続に使用されるユーザー名

master.info ファイル内の行	slave_master_info テーブルのカラム	SHOW SLAVE STATUS のカラム	説明
6	User_password	パスワード (SHOW SLAVE STATUS では表示されない)	マスターへの接続に使用されるパスワード
7	Port	Master_Port	マスターへの接続に使用されるネットワークポート
8	Connect_retry	Connect_Retry	マスターへの再接続を試みる前にスレーブが待機する期間（秒単位）
9	Enabled_ssl	Master_SSL_Allowed	サーバーが SSL 接続をサポートするかどうかを示す
10	Ssl_ca	Master_SSL_CA_File	認証局 (CA) 証明書に使用されるファイル
11	Ssl_capath	Master_SSL_CA_Path	認証局 (CA) 証明書へのパス
12	Ssl_cert	Master_SSL_Cert	SSL 証明書ファイルの名前
13	Ssl_cipher	Master_SSL_Cipher	SSL 接続のハンドシェイクで使用される可能な暗号のリスト
14	Ssl_key	Master_SSL_Key	SSL キーファイルの名前
15	Ssl_verify_server_cert	Master_SSL_Verify_Server_Cert	サーバー証明書を検証するかどうか
16	Heartbeat	[None]	レプリケーションハートビートの間隔（秒単位）
17	Bind	Master_Bind	マスターへの接続にスレーブのどのネットワークインタフェースを使用すべきか
18	Ignored_server_ids	Replicate_Ignore_Server_Ids	無視するサーバー ID のリスト。Ignored_server_ids の場合、サーバー ID のリストの前に、無視するサーバー ID の合計数が付きます。
19	Uuid	Master_UUID	マスターの一意 ID
20	Retry_count	Master_Retry_Count	許容される再接続試行の最大数
21	Ssl_crl	[None]	SSL 証明書失効リストファイルへのパス (MySQL バージョン 5.6.3 で追加)
22	Ssl_crl_path	[None]	SSL 証明書失効リストファイルを

master.info ファイル内の行	slave_master_info テーブルのカラム	SHOW SLAVE STATUS のカラム	説明
			含むディレクトリへのパス (MySQL バージョン 5.6.3 で追加)
23	Enabled_auto_position	Auto_position	自動ポジショニングが使用中かどうか (MySQL バージョン 5.6.5 で追加)

> **注記**
>
> MySQL 5.6.3 より前では、Ssl_verify_server_cert カラムの名前は Ssl_verify_servert_cert でした。(Bug #12407446、Bug #60988)

スレーブ SQL スレッドはリレーログ情報ログを更新します。MySQL 5.6 では、relay-log.info ファイルに行数とレプリケーション遅延値が含まれます。次の表は、relay-log.info ファイル内の行、mysql.slave_relay_log_info テーブル内のカラム、および SHOW SLAVE STATUS で表示されるカラムの間の対応を示します。

relay-log.info 内の行	slave_relay_log_info テーブルのカラム	SHOW SLAVE STATUS のカラム	説明
1	Number_of_lines	[None]	ファイル内の行またはテーブル内のカラムの数
2	Relay_log_name	Relay_Log_File	現在のリレーログファイルの名前
3	Relay_log_pos	Relay_Log_Pos	リレーログファイル内の現在の位置。この位置までのイベントがスレーブデータベースで実行された
4	Master_log_name	Relay_Master_Log_File	リレーログファイル内のイベントが読み取られたマスターバイナリログファイルの名前
5	Master_log_pos	Exec_Master_Log_Pos	すでに実行されたイベントの、マスターのバイナリログファイル内での対応位置
5	Sql_delay	SQL_Delay	スレーブがマスターより遅れる必要がある秒数
6	Number_of_workers	[None]	レプリケーションイベント（トランザクション）を並列に実行するためのスレーブワーカースレッドの数（MySQL バージョン 5.6.7 で追加）
7	Id	[None]	スレーブがマスターより遅れる必要がある秒数（MySQL バージョン 5.6.7 で追加）

MySQL 5.6 より前では、relay-log.info ファイルに行数または遅延値が含まれません (さらに、slave_relay_log_info テーブルを使用できません)。

行	Status カラム	説明
1	Relay_Log_File	現在のリレーログファイルの名前
2	Relay_Log_Pos	リレーログファイル内の現在の位置。この位置までのイベントがスレーブデータベースで実行された
3	Relay_Master_Log_File	リレーログファイル内のイベントが読み取られたマスターバイナリログファイルの名前
4	Exec_Master_Log_Pos	すでに実行されたイベントの、マスターのバイナリログファイル内での対応位置

> 注記
>
> スレーブサーバーを MySQL 5.6 より古いバージョンにダウングレードした場合、古いサーバーは relay-log.info ファイルを正しく読み取りません。これに対処するには、テキストエディタでファイルを変更し、行数を含む最初の行を削除してください。

relay-log.info ファイルの内容と SHOW SLAVE STATUS ステートメントで表示される状態は、relay-log.info ファイルがディスクにフラッシュされていない場合には一致しない場合があります。理想を言えば、オフライン (つまり、mysqld が動作していない) であるスレーブ上でのみ relay-log.info を表示することをお勧めします。動作中のシステムでは、SHOW SLAVE STATUS を使用するか、ステータスログをテーブルに書き込んでいる場合は slave_master_info および slave_relay_log_info テーブルを照会できます。

スレーブのデータをバックアップするときは、リレーログファイルに加えてこれらの 2 つのステータスログもバックアップすることをお勧めします。ステータスログは、スレーブからデータをリストアしたあとにレプリケーションを再開するために必要です。リレーログを失ったけれども、まだリレーログ情報ログがある場合には、それを調べて、マスターバイナリログで SQL スレッドがどのくらい離れて実行されたかを判断できます。それから MASTER_LOG_FILE および MASTER_LOG_POS オプションで CHANGE MASTER TO を使用することで、そのポイントからバイナリログを再度読み取るようにスレーブに指示できます。もちろん、これにはバイナリログがまだマスター上に存在している必要があります。

17.2.3 サーバーがレプリケーションフィルタリングルールをどのように評価するか

マスターサーバーがそのバイナリログにステートメントを書き込まない場合は、ステートメントは複製されません。サーバーがステートメントのログを記録すると、ステートメントはすべてのスレーブに送信され、各スレーブはそれを実行するか無視するかを判断します。

マスターでは、--binlog-do-db および --binlog-ignore-db オプションを使用してバイナリロギングを制御することで、どのデータベースが変更ログを記録するかを制御できます。これらのオプションを評価するときにサーバーが使用するルールの詳細は、セクション17.2.3.1「データベースレベルレプリケーションオプションおよびバイナリロギングオプションの評価」を参照してください。複製するデータベースとテーブルを制御するためにこれらのオプションを使用しないでください。代わりに、スレーブ上でフィルタリングを使用してスレーブで実行されるイベントを制御してください。

スレーブ側では、マスターから受け取るステートメントを実行するか無視するかの判断は、スレーブの起動時に使用される --replicate-* オプションに従って行われます。セクション17.1.4「レプリケーションおよびバイナリロギングのオプションと変数」を参照してください。

--replicate-* オプションがないときは (もっとも単純なケース)、スレーブはマスターから受け取るすべてのステートメントを実行します。そうでない場合は、結果は指定された特定のオプションに依存します。

データベースレベルオプション (--replicate-do-db、--replicate-ignore-db) が最初にチェックされます。このプロセスの説明については、セクション17.2.3.1「データベースレベルレプリケーションオプションおよびバイナリロギングオプションの評価」を参照してください。データベースレベルオプションが使用されない場合は、オプションチェックはテーブルレベルオプション (使用されている場合がある) のチェックに進みます (これらの説明については、セクション17.2.3.2「テーブルレベルレプリケーションオプションの評価」を参照してください)。1 つ以上のデータベースレベルオプションが使用されているけれども、一致するものがない場合は、ステートメントは複製されません。

データベースにのみ影響するステートメントの場合 (つまり、CREATE DATABASE、DROP DATABASE、および ALTER DATABASE)、データベースレベルオプションは --replicate-wild-do-table オプションより常に優先されます。つまり、このようなステートメントの場合、適用するデータベースレベルオプションがない場合にかぎり、--replicate-wild-do-table オプションがチェックされます。これは、MySQL の以前のバージョンの動作からの変更です。スレーブが --replicate-do-db=dbx --replicate-wild-do-table=db%.t1 で起動された場合には、ステートメント CREATE DATABASE dbx は複製されませんでした。(Bug #46110)

オプションセットに何が影響するかの判断を簡素化するために、「do」 と 「ignore」 オプション、またはワイルドカードと非ワイルドカードオプションを混在させるのを避けることをお勧めします。

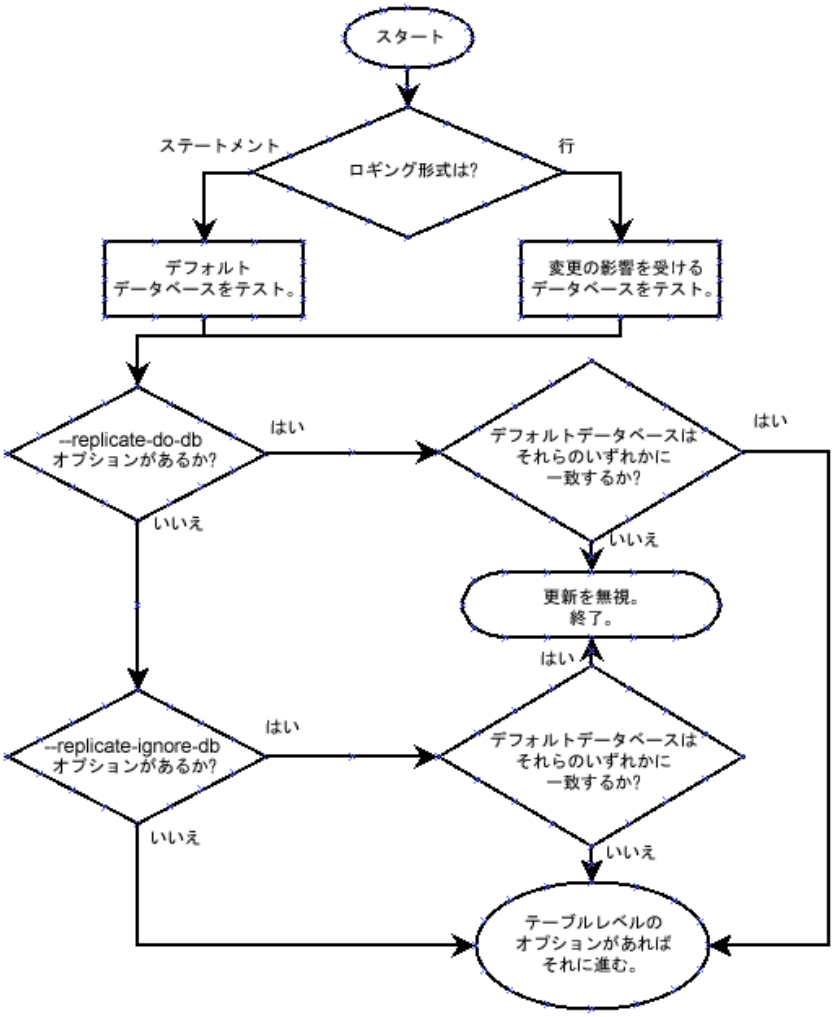
--replicate-rewrite-db オプションが指定された場合、それらは --replicate-* フィルタリングルールがテストされる前に適用されます。

> **注記**
>
> MySQL 5.6 では、すべてのレプリケーションフィルタリングオプションは、lower_case_table_names システム変数の影響を含めて、MySQL サーバー内のほかの場所のデータベースおよびテーブルの名前に適用されるものと同じ、大文字と小文字の区別に関するルールに従います。
>
> これは以前のバージョンの MySQL からの変更点です。(Bug #51639)

17.2.3.1 データベースレベルレプリケーションオプションおよびバイナリロギングオプションの評価

レプリケーションオプションを評価するときに、スレーブは適用される --replicate-do-db または --replicate-ignore-db オプションがあるかどうかを検査することで始めます。--binlog-do-db または --binlog-ignore-db を使用するときは、プロセスは似ていますが、オプションはマスター上で検査されます。

ステートメントレベルレプリケーションでは、デフォルトデータベースの一致が検査されます。行ベースレプリケーションでは、データを変更すべきデータベースが検査されるデータベースです。バイナリロギング形式にかかわらず、データベースレベルオプションの検査は次の図で示すように進みます。



関係する手順の一覧は次のとおりです。

1. --replicate-do-db オプションはありますか ?

- はい　それらのいずれかがデータベースに一致しますか？
  - はい　ステートメントを実行して終了します。
  - いいえ　ステートメントを無視して終了します。
- いいえ　手順 2 に進みます。

2. --replicate-ignore-db オプションはありますか？
   - はい　それらのいずれかがデータベースに一致しますか？
     - はい　ステートメントを無視して終了します。
     - いいえ　手順 3 に進みます。
   - いいえ　手順 3 に進みます。
3. テーブルレベルレプリケーションオプションがある場合、それらの検査に進みます。これらのオプションの検査方法の説明については、セクション17.2.3.2「テーブルレベルレプリケーションオプションの評価」を参照してください。

> **重要**
>
> この段階でまだ許可されているステートメントは、実際にはまだ実行されていません。ステートメントはすべてのテーブルレベルオプション（ある場合）が検査されるまで実行されず、そのプロセスの結果がステートメントの実行を許可します。

バイナリロギングの場合、関連する手順の一覧は次のとおりです。

1. --binlog-do-db または --binlog-ignore-db オプションはありますか？
   - はい　手順 2 に進みます。
   - いいえ　ステートメントのログを記録して終了します。
2. デフォルトデータベースはありますか (データベースが USE で選択されていますか)？
   - はい　手順 3 に進みます。
   - いいえ　ステートメントを無視して終了します。
3. デフォルトデータベースがあります。--binlog-do-db オプションはありますか？
   - はい　それらのいずれかがデータベースに一致しますか？
     - はい　ステートメントのログを記録して終了します。
     - いいえ　ステートメントを無視して終了します。
   - いいえ　手順 4 に進みます。
4. --binlog-ignore-db オプションのいずれかがデータベースに一致しますか？
   - はい　ステートメントを無視して終了します。
   - いいえ　ステートメントのログを記録して終了します。

> **重要**
>
> ステートメントベースロギングの場合、CREATE DATABASE、ALTER DATABASE、および DROP DATABASE ステートメントに適用されるルールにだけ例外が作成されています。これらの場合には、更新のログを記録または無視するかを判断するときに、作成、変更、またはドロップされるデータベースがデフォルトデータベースを置き換えます。

--binlog-do-db は「ほかのデータベースを無視する」ことを意味する場合があります。たとえば、ステートメントベースロギングを使用するときに、--binlog-do-db=sales だけで動作するサーバーは、デフォルトデータベースが

sales ではないバイナリログステートメントに書き込みません。同じオプションで行ベースロギングを使用するときは、サーバーは sales 内のデータを変更する更新のみのログを記録します。

17.2.3.2 テーブルレベルレプリケーションオプションの評価

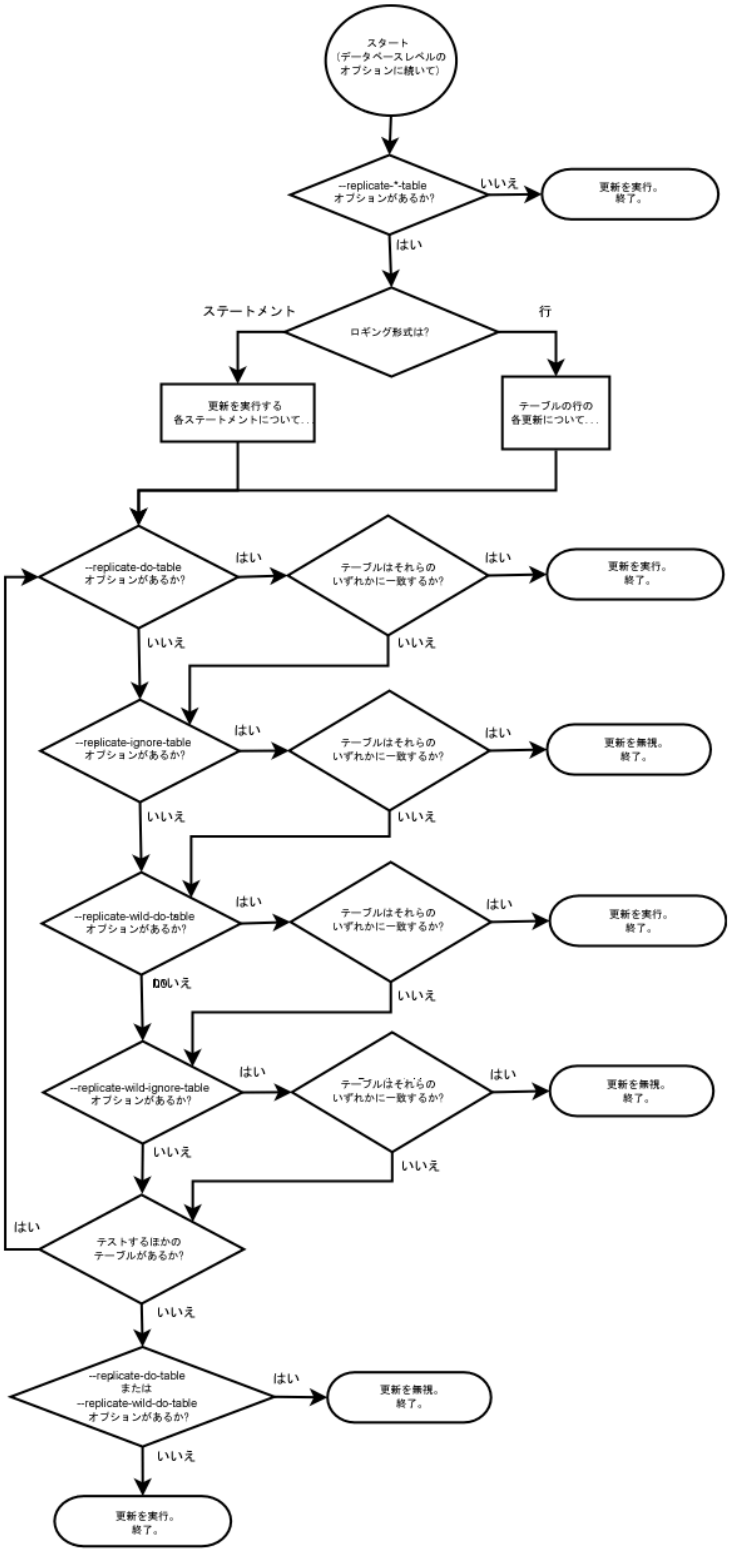
スレーブは、次の 2 つの条件のいずれかが true の場合にのみ、テーブルオプションを検査して評価します。

- 一致するデータベースオプションが見つからなかった。
- 1 つ以上のデータベースオプションが見つかり、前のセクションで説明したルール (セクション17.2.3.1「データベースレベルレプリケーションオプションおよびバイナリロギングオプションの評価」を参照) に従って、「実行」条件に達していると評価された。

最初に、予備条件として、スレーブはステートメントベースレプリケーションが有効であるかどうかを検査します。そうである場合、かつステートメントがストアドファンクション内にある場合には、スレーブはステートメントを実行して終了します。行ベースレプリケーションが有効である場合、スレーブはステートメントがマスター上のストアドファンクション内にあるかどうかがわからないため、この条件は適用されません。

> **注記**
>
> ステートメントベースレプリケーションの場合、レプリケーションイベントがステートメントを表現します (あるイベントを構成するすべての変更が単一 SQL ステートメントに関連付けられています)。行ベースレプリケーションの場合、各イベントが単一テーブル行内の変更を表現します (このため、UPDATE mytable SET mycol = 1 などの単一ステートメントが多くの行ベースイベントを生成する場合があります)。イベントの観点から見ると、テーブルオプションを検査するプロセスは行ベースおよびステートメントベースレプリケーションの両方について同じです。

このポイントに到達して、テーブルオプションがない場合には、スレーブは単純にすべてのイベントを実行します。--replicate-do-table または --replicate-wild-do-table オプションがある場合は、それが実行すべきイベントの場合、イベントはこれらのいずれかに一致する必要があります。そうでない場合、無視されます。--replicate-ignore-table または --replicate-wild-ignore-table オプションがある場合、これらのオプションのいずれかに一致するものを除いてすべてのイベントが実行されます。このプロセスを次の図に示します。



次の手順で、この評価の詳細について説明します。

1. テーブルオプションがありますか？

   - はい　　手順 2 に進みます。
   - いいえ　　イベントを実行して終了します。

2. --replicate-do-table オプションはありますか？

- はい　テーブルはそれらのいずれかに一致しますか？
  - はい　イベントを実行して終了します。
  - いいえ　手順 3 に進みます。
- いいえ　手順 3 に進みます。

3. --replicate-ignore-table オプションはありますか？
   - はい　テーブルはそれらのいずれかに一致しますか？
     - はい　イベントを無視して終了します。
     - いいえ　手順 4 に進みます。
   - いいえ　手順 4 に進みます。
4. --replicate-wild-do-table オプションはありますか？
   - はい　テーブルはそれらのいずれかに一致しますか？
     - はい　イベントを実行して終了します。
     - いいえ　手順 5 に進みます。
   - いいえ　手順 5 に進みます。
5. --replicate-wild-ignore-table オプションはありますか？
   - はい　テーブルはそれらのいずれかに一致しますか？
     - はい　イベントを無視して終了します。
     - いいえ　手順 6 に進みます。
   - いいえ　手順 6 に進みます。
6. --replicate-do-table または --replicate-wild-do-table オプションはありますか？
   - はい　イベントを無視して終了します。
   - いいえ　イベントを実行して終了します。

17.2.3.3 レプリケーションルールの適用

このセクションでは、レプリケーションフィルタリングオプションのさまざまな組み合わせの追加説明と例を示します。

次の表に、レプリケーションフィルタルールのいくつかの典型的な組み合わせをいくつか示します。

条件 (オプションのタイプ)	結果
--replicate-* オプションが全然ない	スレーブはマスターから受け取るすべてのイベントを実行します。
--replicate-*-db オプション、しかしテーブルオプションがない	スレーブは、データベースオプションを使用してイベントを受け入れるか無視します。テーブル制約がないため、これらのオプションで許可されるすべてのイベントを実行します。
--replicate-*-table オプション、しかしデータベースオプションがない	データベース条件がないため、データベース検査段階ですべてのイベントが受け入れられます。スレーブは、テーブルオプションにのみ基づいてイベントを実行または無視します。
データベースおよびテーブルオプションの組み合わせ	スレーブは、データベースオプションを使用してイベントを受け入れるか無視します。それから、テーブルオプションに従ってこれらのオプションで許可されるすべてのイベントを評価します。これは、直観に反しているように見えたり、ステートメントベースまたは行ベースレプリケーションのどちらを使用するかによっ

条件 (オプションのタイプ)	結果
	て異なる結果となったりする場合があります。例のテキストを参照してください。

より複雑な例を示します。ステートメントベースおよび行ベースの両方の設定の結果を調べます。

データベース db1 内に mytbl1、データベース db2 内に mytbl2 という 2つのテーブルがマスター上にあり、スレーブが次のオプションで動作していると想定します（ほかのレプリケーションフィルタリングオプションはない）。

```
replicate-ignore-db = db1
replicate-do-table  = db2.tbl2
```

ここで、次のステートメントをマスター上で実行します。

```
USE db1;
INSERT INTO db2.tbl2 VALUES (1);
```

スレーブでの結果はバイナリログ形式によって大きく異なり、どちらかのケースで当初の予測に一致しない場合があります。

ステートメントベースのレプリケーション　USE ステートメントによって db1 がデフォルトデータベースになります。このため、--replicate-ignore-db オプションが一致し、INSERT ステートメントが無視されます。テーブルオプションは検査されません。

行ベースのレプリケーション　デフォルトデータベースは、行ベースレプリケーションを使用するときにスレーブがデータベースオプションをどのように読み取るかに影響しません。その結果、USE ステートメントは --replicate-ignore-db オプションがどのように処理されるかに違いをもたらしません。このオプションで指定されるデータベースは INSERT ステートメントがデータを変更するデータベースに一致しないため、スレーブはテーブルオプションの検査に進みます。--replicate-do-table で指定されるテーブルは、更新すべきテーブルに一致し、行が挿入されます。

17.3 レプリケーションソリューション

レプリケーションは、多くの異なる環境でさまざまな目的に使用できます。このセクションでは、特定のソリューションタイプにレプリケーションを使用する場合の一般的な注意点とアドバイスを提供します。

セットアップに関する注意点、バックアップ手順、バックアップするファイルなど、バックアップ環境でレプリケーションを使用する場合の情報については、セクション17.3.1「バックアップ用にレプリケーションを使用する」を参照してください。

マスターとスレーブで異なるストレージエンジンを使用している場合のアドバイスやヒントは、セクション17.3.2「異なるマスターおよびスレーブストレージエンジンでレプリケーションを使用する」を参照してください。

スケールアウトソリューションとしてレプリケーションを使用するには、ソリューションを使用するアプリケーションのロジックとオペレーションで若干の変更が必要になります。セクション17.3.3「スケールアウトのためにレプリケーションを使用する」を参照してください。

パフォーマンスやデータ分散のため、データベースごとに異なるレプリケーションスレーブに複製することをお勧めします。セクション17.3.4「異なるデータベースを異なるスレーブに複製する」を参照してください。

レプリケーションスレーブの数が増えるにつれ、マスターでの負荷が増えてパフォーマンスの低下につながる可能性があります (バイナリログを各スレーブに複製する必要があるため)。単一セカンダリサーバーをレプリケーションマスターとして使用する方法など、レプリケーションパフォーマンスを向上するためのヒントについては、セクション17.3.5「レプリケーションパフォーマンスを改善する」を参照してください。

非常時のフェイルオーバーソリューションの一環としてマスターを切り替えたりスレーブをマスターに変換したりする際のガイドについては、セクション17.3.6「フェイルオーバー中にマスターを切り替える」を参照してください。

レプリケーション通信を安全に行うため、SSL を使用して通信チャネルを暗号化できます。手順を追った説明については、セクション17.3.7「SSL を使用してレプリケーションをセットアップする」を参照してください。

17.3.1 バックアップ用にレプリケーションを使用する

レプリケーションをバックアップソリューションとして使用するには、データをマスターからスレーブに複製してから、データスレーブをバックアップします。スレーブはマスターの実行動作に影響を与えずに一時停止した

りシャットダウンしたりできるため、「ライブ」データの効果的なスナップショットを作成できます (ほかの方法ではマスターをシャットダウンする必要がある)。

データベースをどのようにバックアップするかは、そのサイズ、およびデータだけまたはデータとレプリケーションスレーブ状態 (障害時にスレーブを再構築できるように) をバックアップするのかによって異なります。つまり、2 つの選択肢があります。

- マスター上のデータをバックアップするためのソリューションとしてレプリケーションを使用し、データベースのサイズがあまり大きくない場合は、mysqldump ツールをお勧めします。セクション17.3.1.1「mysqldump を使用してスレーブをバックアップする」を参照してください。
- より大きなデータベースの場合は、mysqldump は実用的または効率的でなく、代わりにローデータファイルをバックアップできます。ローデータファイルオプションを使用することは、スレーブ障害時にスレーブを再作成できるように、バイナリおよびリレーログをバックアップできることも意味します。詳細については、セクション17.3.1.2「スレーブからローデータをバックアップする」を参照してください。

もう一つのバックアップ方法は読み取り専用状態のサーバーを置くことで、これはマスターまたはスレーブサーバーに使用できます。バックアップは読み取り専用サーバーに対して実行され、これが通常の読み取り/書き込み操作ステータスに戻されます。セクション17.3.1.3「マスターまたはスレーブを読み取り専用にしてバックアップする」を参照してください。

17.3.1.1 mysqldump を使用してスレーブをバックアップする

mysqldump を使用してデータベースのコピーを作成することで、MySQL Server の別のインスタンスに情報をインポートできる形式でデータベース内のすべてのデータを取得できます (セクション4.5.4「mysqldump — データベースバックアッププログラム」を参照してください)。情報の形式が SQL ステートメントであるため、緊急時にデータにアクセスする必要がある場合にファイルを実行中サーバーに配布して適用することも簡単にできます。ただし、データセットのサイズが非常に大きい場合は、mysqldump が実用的でない場合があります。

mysqldump を使用するときは、一貫性のあるデータセットがダンプに含まれるように、スレーブ上でレプリケーションを停止してからダンププロセスを起動してください。

1. スレーブでの要求処理を停止します。mysqladmin を使用してスレーブでのレプリケーションを完全に停止できます。

   ```
   shell> mysqladmin stop-slave
   ```

   イベント実行を一時停止するために、スレーブ SQL スレッドだけを停止することもできます。

   ```
   shell> mysql -e 'STOP SLAVE SQL_THREAD;'
   ```

   これにより、スレーブは引き続きマスターのバイナリログからデータ変更イベントを受け取り、I/O スレッドを使用してそれらをリレーログに格納できますが、スレーブはこれらのイベントを実行したりそのデータを変更したりできません。ビジーなレプリケーション環境では、バックアップ中に I/O スレッドの実行を許可することで、スレーブ SQL スレッド再起動時に追い付くプロセスの速度が向上する場合があります。

2. mysqldump を実行してデータベースをダンプします。すべてのデータベースをダンプしたり、ダンプするデータベースを選択したりできます。たとえば、すべてのデータベースをダンプするには:

   ```
   shell> mysqldump --all-databases > fulldb.dump
   ```

3. ダンプが完了したら、スレーブの動作を再開します。

   ```
   shell> mysqladmin start-slave
   ```

前の例では、ログイン資格証明 (ユーザー名、パスワード) をコマンドに追加したり、毎日自動的に実行できるプロセスをスクリプトにバンドルしたりすることをお勧めします。

この方法を使用する場合、スレーブレプリケーションプロセスをモニターして、バックアップの実行にかかる時間がマスターからのイベントにスレーブが追い付く能力に影響しないようにしてください。セクション17.1.5.1「レプリケーションステータスの確認」を参照してください。スレーブが追い付けない場合は、別のスレーブを追加してバックアッププロセスを分散することをお勧めします。このシナリオの構成方法の例は、セクション17.3.4「異なるデータベースを異なるスレーブに複製する」を参照してください。

17.3.1.2 スレーブからローデータをバックアップする

コピーされるファイルの完全性を保証するには、MySQL レプリケーションスレーブ上のローデータファイルのバックアップが、スレーブサーバーがシャットダウンしている間に実行されるべきです。MySQL サーバーが

まだ実行中の場合は、バックグラウンドタスクがデータベースファイルをまだ更新中の可能性があります (特に、InnoDB などのストレージエンジンをバックグラウンドプロセスで使用するもの)。InnoDB の場合、これらの問題はクラッシュリカバリ中に解決されるはずですが、バックアッププロセス中にマスターの実行に影響を与えずにスレーブサーバーをシャットダウンできるため、この機能を利用することは意味があります。

サーバーをシャットダウンしてファイルをバックアップするには：

1. スレーブ MySQL サーバーをシャットダウンします。

```
shell> mysqladmin shutdown
```

2. データファイルをコピーします。cp、tar、WinZip など、コピーまたはアーカイブに適したユーティリティーを使用できます。たとえば、データディレクトリが現在のディレクトリの下にある場合、ディレクトリ全体を次のようにアーカイブできます。

```
shell> tar cf /tmp/dbbackup.tar ./data
```

3. MySQL サーバーを再起動します。Unix の場合:

```
shell> mysqld_safe &
```

Windows の場合:

```
C:\> "C:\Program Files\MySQL\MySQL Server 5.6\bin\mysqld"
```

通常は、スレーブ MySQL サーバーのデータディレクトリ全体をバックアップしてください。データをリストアしてそれらをスレーブとして運用できるようにしたい場合 (たとえば、スレーブの障害時)、スレーブのデータに加えて、スレーブステータスファイル、マスター情報およびリレーログ情報リポジトリ、およびリレーログファイルもバックアップしてください。これらのファイルは、スレーブのデータをリストアしたあとにレプリケーションを再開するために必要です。

リレーログを失ったけれども relay-log.info ファイルがある場合は、これを調べることで SQL スレッドがマスターバイナリログ内でどのくらい実行されたかを判断できます。それから MASTER_LOG_FILE および MASTER_LOG_POS オプションで CHANGE MASTER TO を使用することで、そのポイントからバイナリログを再度読み取るようにスレーブに指示できます。これには、マスターサーバー上にバイナリログが残っている必要があります。

スレーブが LOAD DATA INFILE ステートメントをレプリケートする場合、スレーブがこのために使用するディレクトリ内に存在する SQL_LOAD-* ファイルもバックアップしてください。スレーブは、中断した LOAD DATA INFILE 操作のレプリケーションを再開するためにこれらのファイルを必要とします。このディレクトリの場所は --slave-load-tmpdir オプションの値です。そのオプションでサーバーを起動しなかった場合、ディレクトリの場所は tmpdir システム変数の値になります。

17.3.1.3 マスターまたはスレーブを読み取り専用にしてバックアップする

グローバル読み取りロックを獲得してから read_only システム変数を操作して、バックアップするサーバーの状態を読み取り専用に変更することで、レプリケーションセットアップでマスターまたはスレーブサーバーをバックアップできます。

1. サーバーを読み取り専用にします (検索のみが処理され、更新はブロックされます)。
2. バックアップを実行します。
3. サーバーを通常の読み取り/書き込み状態に戻します。

> **注記**
>
> このセクションの手順では、バックアップするサーバーを、サーバーからデータを取得するバックアップ方式 (mysqldump など) に安全な状態に変換しています (セクション 4.5.4「mysqldump — データベースバックアッププログラム」を参照してください)。バイナリバックアップを作成するために、ファイルを直接コピーする方法でこれらの手順の使用を試みてはいけません (サーバーが変更後データをまだメモリー内にキャッシュしていてディスクにフラッシュしていない可能性があるため)。

後続の手順では、マスターサーバーおよびスレーブサーバーに対してこれを行う方法を説明します。ここで説明する両方のシナリオでは、次のレプリケーションセットアップを想定します。

- マスターサーバー M1

- マスターとして M1 を持つスレーブサーバー S1
- M1 に接続されたクライアント C1
- S1 に接続されたクライアント C2

どちらのシナリオでも、グローバル読み取りロックを獲得して read_only 変数を操作するステートメントは、バックアップするサーバーで実行され、そのサーバーのスレーブには伝達されません。

シナリオ 1: 読み取り専用マスターでのバックアップ

これらのステートメントをマスター M1 で実行して、これを読み取り専用状態にします。

```
mysql> FLUSH TABLES WITH READ LOCK;
mysql> SET GLOBAL read_only = ON;
```

M1 が読み取り専用状態のときは、次の属性が true になります。

- C1 によって M1 に送られる更新要求は、サーバーが読み取り専用モードであるためブロックされます。
- C1 によって M1 に送られるクエリー結果要求は成功します。
- M1 でバックアップを作成することは安全です。
- S1 でバックアップを作成することは安全ではありません。このサーバーはまだ実行中で、バイナリログを処理中であったり、クライアント C2 から着信する要求を更新したりする可能性があります。

M1 が読み取り専用のときに、バックアップを実行してください。たとえば、mysqldump を使用できます。

M1 でのバックアップ操作が完了したあとに、これらのステートメントを実行することで M1 を通常の動作状態に戻します。

```
mysql> SET GLOBAL read_only = OFF;
mysql> UNLOCK TABLES;
```

M1 でバックアップを実行することは安全ですが (バックアップに関するかぎり)、パフォーマンスには最適ではありません (M1 のクライアントが更新の実行をブロックされるため)。

この方法は、レプリケーションセットアップのマスターサーバーのバックアップに適用されますが、非レプリケーションセットアップの単一サーバーに使用することもできます。

シナリオ 2: 読み取り専用スレーブでのバックアップ

これらのステートメントをスレーブ S1 で実行して、これを読み取り専用状態にします。

```
mysql> FLUSH TABLES WITH READ LOCK;
mysql> SET GLOBAL read_only = ON;
```

S1 が読み取り専用状態のときは、次の属性が true になります。

- マスター M1 が動作を継続しているため、マスター上でバックアップを作成することは安全ではありません。
- スレーブ S1 が停止しているため、スレーブ S1 上でバックアップを作成することは安全です。

これらの属性は一般的なバックアップシナリオの基礎を提供します。あるスレーブが少しの間バックアップの実行でビジーであっても問題ではありません。ネットワーク全体に影響せず、システムはバックアップ中でも動作しているためです。特に、クライアントは引き続きマスターサーバー上で更新を実行でき、依然としてスレーブでのバックアップアクティビティーによる影響を受けません。

S1 が読み取り専用のときに、バックアップを実行してください。たとえば、mysqldump を使用できます。

S1 でのバックアップ操作が完了したあとに、これらのステートメントを実行することで S1 を通常の動作状態に戻します。

```
mysql> SET GLOBAL read_only = OFF;
mysql> UNLOCK TABLES;
```

スレーブは、通常動作に戻ったあとにマスターのバイナリログからの未処理の更新に追い付くことで、ふたたびマスターに同期されます。

17.3.2 異なるマスターおよびスレーブストレージエンジンでレプリケーションを使用する

レプリケーションプロセスでは、マスター上のソーステーブルとスレーブ上の複製先テーブルが異なるエンジンタイプを使用しているかどうかは重要ではありません。実際、default_storage_engine および storage_engine システム変数は複製されません。

これは、異なるレプリケーションシナリオに異なるエンジンタイプを利用できるという点で、レプリケーションプロセスにいくつかの利点を提供します。たとえば、典型的なスケールアウトシナリオでは (セクション 17.3.3「スケールアウトのためにレプリケーションを使用する」を参照してください)、マスター上ではトランザクション機能を利用するために InnoDB テーブルを使用し、スレーブ上ではデータが読み取られるだけでトランザクションサポートが必要ないので MyISAM を使用します。データロギング環境でレプリケーションを使用するときは、スレーブ上で Archive ストレージエンジンを使用することをお勧めします。

マスターおよびスレーブ上で異なるエンジンを構成するかどうかは、最初のレプリケーションプロセスをどのようにセットアップするかに依存します。

- mysqldump を使用してマスター上のデータベーススナップショットを作成した場合は、ダンプファイルテキストを編集して各テーブルで使用されるエンジンタイプを変更できます。

  mysqldump の別の使用方法は、スレーブ上で使用したくないエンジンタイプを無効にしてから、このダンプを使用してスレーブ上にデータを構築することです。たとえば、FEDERATED エンジンを無効にするために、--skip-federated オプションをスレーブ上に追加できます。作成するテーブル用に特定のエンジンが存在しない場合、MySQL はデフォルトエンジンタイプ (通常は MyISAM) を使用します。(これには、NO_ENGINE_SUBSTITUTION SQL モードが有効でないことが必要です。)この方法でほかのエンジンを無効にする場合は、必要なエンジンだけをサポートする特別なバイナリを構築してスレーブで使用することを検討することをお勧めします。

- ローデータファイル (バイナリバックアップ) を使用してスレーブをセットアップしている場合、最初のテーブル形式を変更できなくなります。代わりに、スレーブが起動したあとに ALTER TABLE を使用してテーブルタイプを変更してください。

- 新しいマスター/スレーブレプリケーションセットアップで、マスター上に現在テーブルがない場合は、新しいテーブルを作成するときにエンジンタイプを指定することは避けてください。

レプリケーションソリューションをすでに実行していて、既存のテーブルを別のエンジンタイプに変更する場合は、これらの手順に従ってください。

1. スレーブがレプリケーション更新を実行するのを停止します。

   ```
   mysql> STOP SLAVE;
   ```

   これにより、中断なしにエンジンタイプを変更できるようになります。

2. 変更するテーブルごとに ALTER TABLE ... ENGINE='engine_type' を実行します。

3. スレーブレプリケーションプロセスを再開します。

   ```
   mysql> START SLAVE;
   ```

default_storage_engine 変数は複製されませんが、エンジン指定を含む CREATE TABLE および ALTER TABLE ステートメントはスレーブに正しく複製されます。たとえば、CSV テーブルがある場合はこれを実行します。

```
mysql> ALTER TABLE csvtable Engine='MyISAM';
```

上記のステートメントはスレーブに複製され、スレーブ上のエンジンタイプが MyISAM に変換されます (以前にスレーブ上のエンジンタイプを CSV 以外のエンジンに変更した場合でも)。マスターとスレーブでのエンジンの違いを保持する場合は、新しいテーブルを作成するときに default_storage_engine 変数を慎重に使用してください。たとえば、次の代わりに:

```
mysql> CREATE TABLE tablea (columna int) Engine=MyISAM;
```

この形式を使用してください。

```
mysql> SET default_storage_engine=MyISAM;
mysql> CREATE TABLE tablea (columna int);
```

複製されるときに、default_storage_engine 変数は無視され、CREATE TABLE ステートメントはスレーブ上のデフォルトエンジンを使用してスレーブ上で実行されます。

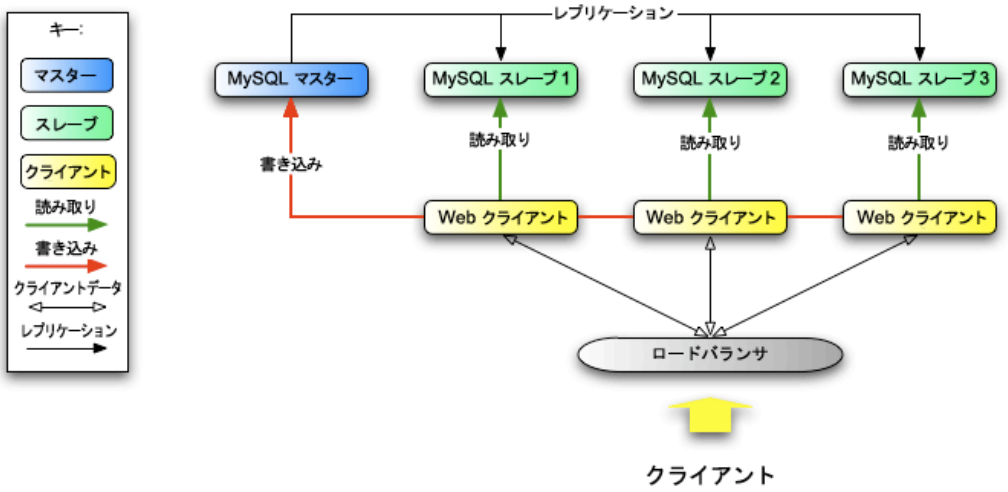
17.3.3 スケールアウトのためにレプリケーションを使用する

レプリケーションをスケールアウトソリューションとして、つまり、いくつかの合理的な制限内でデータベースクエリーの負荷を複数のデータベースサーバーに分割するために使用できます

レプリケーションが 1 つのマスターから 1 つ以上のスレーブに分散するように機能するため、スケールアウトにレプリケーションを使用することは、読み取りが非常に多く、書き込み/更新が少ない環境で最適に機能します。ほとんどの Web サイトはこのカテゴリ (ユーザーが Web サイトをブラウズしたり、記事や投稿を読んだり、製品を見たりする) に該当します。更新は、セッション管理中、購入するとき、またはフォーラムにコメント/メッセージを追加するときにのみ発生します。

この状況でレプリケーションを使用すると、読み取りをレプリケーションスレーブに分散しながら、Web サーバーは書き込みが必要なときにレプリケーションマスターとやり取りできます。このシナリオのためのサンプルレプリケーションレイアウトは、図17.1「スケールアウト中のパフォーマンスを向上するためにレプリケーションを使用する」で見ることができます。

図 17.1 スケールアウト中のパフォーマンスを向上するためにレプリケーションを使用する



データベースにアクセスするコードの一部が適切に抽象化/モジュール化されている場合は、それを複製されたセットアップで動作するように変換することはとても効率的かつ簡単であるはずです。データベースアクセスの実装を、すべての書き込みをマスターに送信し、読み取りをマスターまたはスレーブに送信するように変更してください。コードがこのレベルの抽象を備えていない場合、複製されたシステムのセットアップは整理するための機会および動機となります。まずは、次の関数を実装するラッパーライブラリまたはモジュールを作成してください。

- safe_writer_connect()
- safe_reader_connect()
- safe_reader_statement()
- safe_writer_statement()

各関数名の safe_ は、その関数がすべてのエラー条件の処理を引き受けることを意味します。関数に別の名前を使用できます。重要なことは、読み取りのための接続、書き込みのための接続、読み取りの実行、および書き込みの実行に対して、統一されたインタフェースを持つことです。

次に、ラッパーライブラリを使用するようにクライアントコードを変換してください。これは、最初は苦しくて怖い工程かもしれませんが、長い目でみるとやるだけの価値があります。説明した方法を使用するすべてのアプリケーションは、マスター/スレーブ構成を利用できます (複数のスレーブを使用するものであっても)。こうしたコードは非常に保守しやすく、トラブルシューティングオプションを追加するのも手間がかかりません。たとえば、1 つか 2 つの関数を変更するだけで、各ステートメントにどのくらいの時間がかかったかや、発行されたステートメントの中でどのステートメントがエラーになったかのログを記録できます。

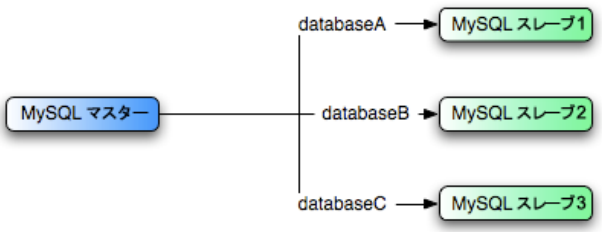
たくさんのコードを書いた経験がある場合は、MySQL 標準配布に付属の replace ユーティリティーを使用することで変換タスクを自動化したり、独自の変換スクリプトを書いたりすることをお勧めします。理想的には、コードが一貫性のあるプログラミングスタイル規則を使用するべきです。そうでない場合は、一貫性のあるスタイ

ルを使用するために、とにかく書き換えたり、少なくとも詳しく調べて手動で整理したりすることをお勧めします。

17.3.4 異なるデータベースを異なるスレーブに複製する

マスターが 1 つで、異なるデータベースを異なるスレーブに複製したい場合があります。たとえば、データ分析時の負荷を分散するために、異なる売上データを異なる部門に分散したい場合です。このレイアウトの例を図 17.2「データベースを別個のレプリケーションスレーブに複製するためにレプリケーションを使用する」に示します。

図 17.2 データベースを別個のレプリケーションスレーブに複製するためにレプリケーションを使用する



この分散は、マスターとスレーブを通常どおり構成してから、各スレーブ上で --replicate-wild-do-table 構成オプションを使用して各スレーブが処理するバイナリログステートメントを制限することで、実現できます。

> **重要**
>
> ステートメントベースレプリケーションを使用するときに、この目的のために --replicate-do-db を使用してはいけません。ステートメントベースレプリケーションでは、このオプションの効果が、現在選択されているデータベースによって異なるためです。このことは、混合形式のレプリケーションにも当てはまります。一部の更新をステートメントベース形式を使用して複製できるためです。
>
> しかし、行ベースレプリケーションだけを使用している場合には、この目的のために --replicate-do-db を使用しても安全なはずです。この場合には、現在選択されているデータベースがオプションの動作に影響しないためです。

たとえば、図17.2「データベースを別個のレプリケーションスレーブに複製するためにレプリケーションを使用する」 に示すような分散をサポートするには、START SLAVE を実行する前に、各レプリケーションスレーブを次のように構成してください。

- レプリケーションスレーブ 1 は --replicate-wild-do-table=databaseA.% を使用するべきです。
- レプリケーションスレーブ 2 は --replicate-wild-do-table=databaseB.% を使用するべきです。
- レプリケーションスレーブ 3 は --replicate-wild-do-table=databaseC.% を使用するべきです。

この構成の各スレーブはマスターからバイナリログ全体を受け取りますが、そのバイナリログから、そのスレーブで有効な --replicate-wild-do-table オプションによって含まれるデータベースとテーブルに適用されるイベントだけを実行します。

レプリケーションが開始する前にスレーブに同期する必要があるデータがある場合、いくつかの選択肢があります。

- すべてのデータを各スレーブに同期し、保持したくないデータベースまたはテーブル、あるいはその両方を削除します。
- mysqldump を使用してデータベースごとに別々のダンプファイルを作成し、各スレーブに該当するダンプファイルをロードします。
- ローデータファイルダンプを使用して、各スレーブに必要な固有のファイルとデータベースのみを含めます。

> **注記**
>
> これは、innodb_file_per_table を使用しないかぎり、InnoDB データベースでは機能しません。

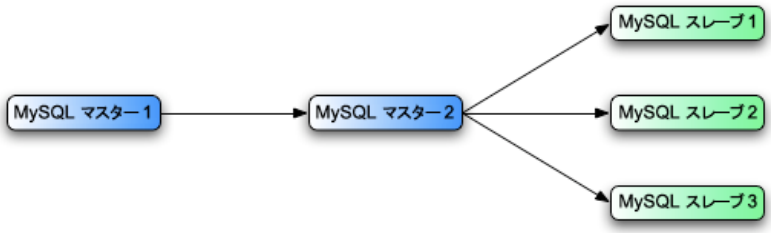
17.3.5 レプリケーションパフォーマンスを改善する

マスターに接続するスレーブの数が増えるにつれ、各スレーブがマスターへのクライアント接続を使用するため、最小限ですが負荷も増えます。また、各スレーブはマスターバイナリログのフルコピーを受け取る必要があるため、マスター上のネットワーク負荷も増加し、ボトルネックになる場合があります。

1 つのマスターに接続される非常に多くのスレーブを使用していて、そのマスターも要求の処理にビジーな場合には（たとえば、スケールアウトソリューションの一部として）、レプリケーションプロセスのパフォーマンスを改善することをお勧めします。

レプリケーションプロセスのパフォーマンスを改善する方法の 1 つは、より深いレプリケーション構造を作成して、マスターは 1 つのスレーブにのみ複製し、残りのスレーブは個々のレプリケーション要件のためにこのプライマリスレーブに接続します。この構造のサンプルを図17.3「パフォーマンスを改善するために追加レプリケーションホストを使用する」に示します。

図 17.3 パフォーマンスを改善するために追加レプリケーションホストを使用する



これが機能するには、MySQL インスタンスを次のように構成する必要があります。

- マスター 1 はプライマリマスターで、すべての変更と更新がこのデータベースに書き込まれます。バイナリロギングはこのマシンで有効にすべきです。
- マスター 2 はマスター 1 のスレーブで、レプリケーション構造内の残りのスレーブにレプリケーション機能を提供します。マスター 2 はマスター 1 への接続を許可される唯一のマシンです。マスター 2 のバイナリロギングも有効になっていて、--log-slave-updates オプションによってマスター 1 からのレプリケーション命令がマスター 2 のバイナリログにも書き込まれるため、それらを真のスレーブに複製できます。
- スレーブ 1、スレーブ 2、およびスレーブ 3 はマスター 2 のスレーブとして機能し、マスター 2 からの情報 (実際にはマスター 1 でログが記録された更新で構成される) を複製します。

上記のソリューションは、クライアント負荷とプライマリマスターでのネットワークインタフェース負荷を低減するため、直接のデータベースソリューションとして使用されるときにプライマリマスターの全体的なパフォーマンスを改善するはずです。

スレーブがマスター上のレプリケーションプロセスに追い付くことに問題がある場合には、利用できるオプションがいくつかあります。

- 可能であれば、リレーログとデータファイルを異なる物理ドライブに置きます。これを行うには、--relay-log オプションを使用してリレーログの場所を指定します。
- スレーブがマスターよりかなり遅い場合は、異なるデータベースを異なるスレーブに複製する責任を分割することをお勧めします。セクション17.3.4「異なるデータベースを異なるスレーブに複製する」を参照してください。
- マスターがトランザクションを使用するため、スレーブ上でのトランザクションサポートに関心がない場合は、スレーブ上で MyISAM または別の非トランザクションエンジンを使用してください。セクション17.3.2「異なるマスターおよびスレーブストレージエンジンでレプリケーションを使用する」を参照してください。
- スレーブがマスターとして動作せず、障害時に確実にマスターを回復できる潜在的ソリューションが実装されている場合は、--log-slave-updates をオフに切り替えることができます。これにより、「ダム」スレーブが実行したイベントのログが自身のバイナリログに記録されなくなります。

17.3.6 フェイルオーバー中にマスターを切り替える

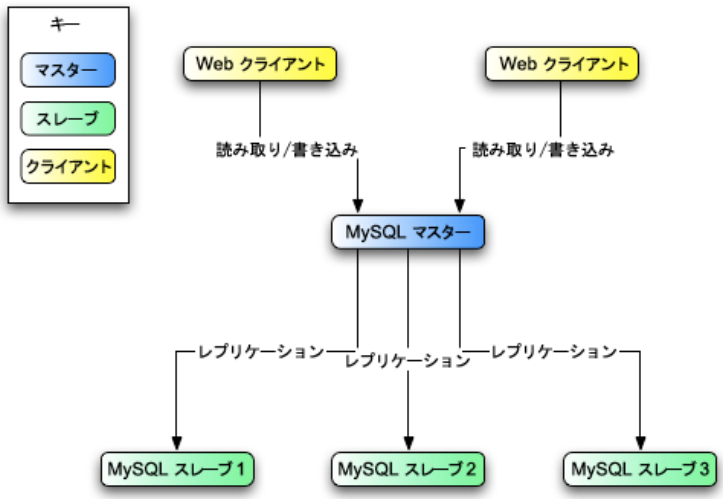
マスターと 1 つ以上のスレーブをセットアップする必要があります。それから、起動しているかどうかを確認するためにマスターをモニターするアプリケーションまたはスクリプトを書いたり、障害時に別のマスターに変更

するようにスレーブおよびアプリケーションに指示したりする必要があります。このセクションでは、この方法でフェイルオーバーをセットアップするときに遭遇するいくつかの問題について説明します。

CHANGE MASTER TO ステートメントを使用して、新しいマスターに切り替えるようにスレーブに指示できます。スレーブは、マスター上のデータベースがスレーブ上のものと互換性があるかどうかを確認しません。単に、新しいマスターのバイナリログ内の指定された座標からイベントを読み取って実行するだけです。フェイルオーバーの状況では、グループ内のすべてのサーバーが同じバイナリログファイルから同じイベントを実行するのが一般的であるため、イベントのソースを変更しても、変更を加えるときに注意することで、データベースの構造または完全性に影響を与えないはずです。

スレーブは、--log-bin オプション付き、--log-slave-updates なしで実行するべきです。この方法では、スレーブはスレーブ mysqld を再起動せずにマスターになる準備が整っています。図17.4「レプリケーションを使用する冗長性、初期構造」で示す構造を想定してください。

図 17.4 レプリケーションを使用する冗長性、初期構造



この図では、MySQL Master はマスターデータベースを保持し、MySQL Slave ホストはレプリケーションスレーブで、Web Client マシンはデータベース読み取りおよび書き込みを発行しています。読み取りだけを発行する（そして、一般的にスレーブに接続されている）Web クライアントは示されていません。障害時に新しいサーバーに切り替える必要がないためです。読み取り/書き込みスケールアウトレプリケーション構造の詳細例については、セクション17.3.3「スケールアウトのためにレプリケーションを使用する」を参照してください。

各 MySQL Slave (Slave 1、Slave 2、Slave 3) は、--log-bin 付き、--log-slave-updates なしで動作しているスレーブです。--log-slave-updates が指定されないかぎり、スレーブがマスターから受け取る更新のログはバイナリログに記録されないため、各スレーブ上のバイナリログは最初は空です。何らかの原因により MySQL Master が使用できなくなった場合、スレーブの 1 つを選んで新しいマスターにできます。たとえば、Slave 1 を選択した場合、すべての Web Clients を Slave 1 (そのバイナリログに更新を書き込む) にリダイレクトされるはずです。すると、Slave 2 と Slave 3 は Slave 1 から複製されるはずです。

--log-slave-updates なしでスレーブを実行する理由は、スレーブの 1 つを新しいマスターにしたために、スレーブが更新を 2 回受け取らないようにすることです。Slave 1 は、その --log-slave-updates が有効である場合は、Master から受け取る更新を自身のバイナリログに書き込みます。これは、Slave 2 のマスターが Master から Slave 1 に変更されると、Slave 1 がすでに Master から受け取っていた更新を受け取る可能性があることを意味します。

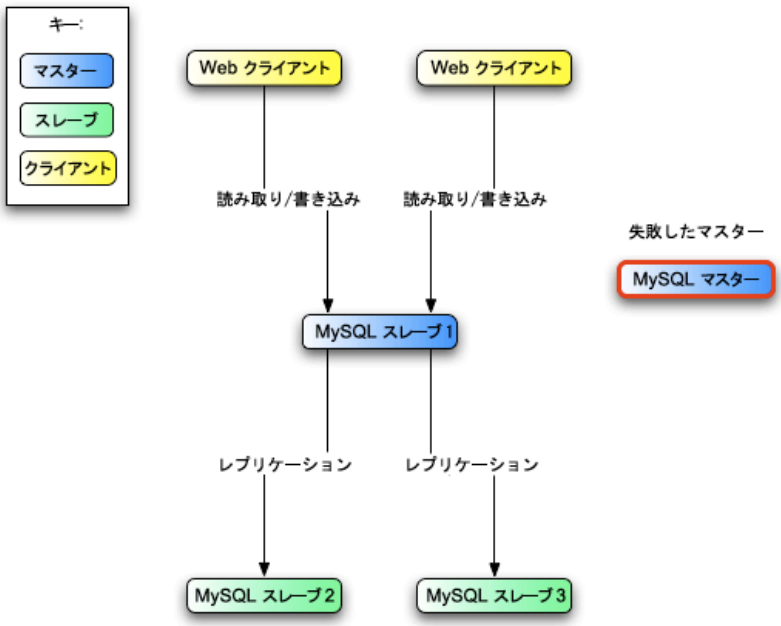
すべてのスレーブがそれぞれのリレーログ内のステートメントを処理したことを確認してください。各スレーブで STOP SLAVE IO_THREAD を発行して、Has read all relay log を見るまで SHOW PROCESSLIST の出力をチェックしてください。これがすべてのスレーブで true のときは、それらを新しいセットアップに再構成できます。昇格してマスターになるスレーブ Slave 1 では、STOP SLAVE と RESET MASTER を発行してください。

ほかのスレーブ Slave 2 と Slave 3 では、STOP SLAVE および CHANGE MASTER TO MASTER_HOST='Slave1' を使用します (ここで、'Slave1' は Slave 1 の実際のホスト名を表します)。CHANGE MASTER TO を使用するには、Slave 2 または Slave 3 から Slave 1 に接続する方法に関するすべての情報 (ユーザー、パスワード、ポート) を追加してください。ここで CHANGE MASTER TO ステートメントを発行するときは、Slave 1 バイナリログファイルの名前、または読み取るログ位置を指定する必要はありません (最初のバイナリログファイルおよび位置 4 がデフォルトです)。最後に、Slave 2 と Slave 3 で START SLAVE を実行します。

新しいレプリケーションセットアップが実行されたあと、各 Web Client がそれぞれのステートメントを Slave 1 に送るように指示する必要があります。この時点から、Web Client から Slave 1 に送られるすべての更新ステートメントが Slave 1 のバイナリログに書き込まれます (Master の停止以降に Slave 1 に送られるすべての更新ステートメントが含まれます)。

結果のサーバー構造を図17.5「レプリケーションを使用する冗長性、マスター障害後」に示します。

図 17.5 レプリケーションを使用する冗長性、マスター障害後



Master がふたたび利用できる状態になったら、それを Slave 1 のスレーブにするべきです。これを実行するには、先ほど Slave 2 と Slave 3 で発行したものと同じ CHANGE MASTER TO ステートメントを Master で発行します。すると Master が Slave 1 のスレーブになり、オフラインであったときに受け取らなかった Web Client 書き込みを受け取ります。

Master をふたたびマスターにするには (たとえば、これがもっとも強力なマシンであるため)、Slave 1 が利用できない状態で Master が新しいマスターになるように先ほどの手順を使用します。この手順では、Master の Slave 1、Slave 2、および Slave 3 を作成する前に、Master で RESET MASTER を実行するのを忘れないでください。これを実行しない場合は、スレーブが Web Client アプリケーションから、Master が利用できなくなった時点より前の古い書き込みを受け取る可能性があります。

スレーブ間の同期がないため (それらが同じマスターを共有していても)、一部のスレーブがほかのものよりかなり進んでしまう可能性があることを承知してください。これは場合によっては、前の例で説明した手順が期待どおりに機能しない可能性があることを意味します。ただし実際には、すべてのスレーブ上のリレーログは互いにかなり近いはずです。

アプリケーションがマスターの場所を常に知っているための 1 つの方法は、マスターの動的 DNS エントリを持つことです。bind で nsupdate を使用することで DNS を動的に更新できます。

17.3.7 SSL を使用してレプリケーションをセットアップする

レプリケーション中に必要なバイナリログの転送を暗号化するために SSL を使用するには、マスターとスレーブの両方が SSL ネットワーク接続をサポートする必要があります。どちらかのホストが SSL 接続をサポートしない場合は (SSL 用にコンパイルまたは構成されていなかったため)、SSL 接続ベースのレプリケーションは実現できません。

SSL 接続を使用するレプリケーションをセットアップすることは、SSL を使用するサーバーとクライアントをセットアップすることに似ています。マスターで使用できる適切なセキュリティー証明書、および各スレーブで同様の証明書 (同じ認証局からの) を取得 (または作成) する必要があります。

サーバーとクライアントを SSL 接続用にセットアップする際の詳細は、セクション6.3.10.2「SSL を使用するための MySQL の構成」を参照してください。

マスター上で SSL を有効にするには、適切な証明書を作成または取得してから、マスターの my.cnf ファイルの [mysqld] セクション内でマスターの構成に次の構成オプションを追加する必要があります。

```
[mysqld]
ssl-ca=cacert.pem
ssl-cert=server-cert.pem
ssl-key=server-key.pem
```

証明書へのパスは相対でも絶対でもかまいません。この目的のためには完全パスを常に使用することをお勧めします。

オプションは次のとおりです。

- ssl-ca は認証局（ CA ）証明書を識別します。
- ssl-cert はサーバー公開鍵を識別します。これをクライアントに送信し、それに含まれる CA 証明書と照合して認証できます。
- ssl-key はサーバー秘密鍵を識別します。

スレーブでは、SSL 情報を設定するために 2 つのオプションを利用できます。スレーブの my.cnf ファイルの [client] セクションにスレーブ証明書を追加するか、または CHANGE MASTER TO ステートメントを使用して SSL 情報を明示的に指定できます。

- オプションファイルを使用してスレーブ証明書を追加するには、スレーブの my.cnf ファイルの [client] セクションに次の行を追加します。

  ```
  [client]
  ssl-ca=cacert.pem
  ssl-cert=client-cert.pem
  ssl-key=client-key.pem
  ```

  スレーブがマスターに接続しないように --skip-slave-start オプションを使用して、スレーブサーバーを再起動します。SSL 接続を有効にする MASTER_SSL オプションを使用して、マスター構成を指定する CHANGE MASTER TO を使用します。

  ```
  mysql> CHANGE MASTER TO
      -> MASTER_HOST='master_hostname',
      -> MASTER_USER='replicate',
      -> MASTER_PASSWORD='password',
      -> MASTER_SSL=1;
  ```

- CHANGE MASTER TO ステートメントを使用して SSL 証明書のオプション指定するには、SSL オプションを付加します。

  ```
  mysql> CHANGE MASTER TO
      -> MASTER_HOST='master_hostname',
      -> MASTER_USER='replicate',
      -> MASTER_PASSWORD='password',
      -> MASTER_SSL=1,
      -> MASTER_SSL_CA = 'ca_file_name',
      -> MASTER_SSL_CAPATH = 'ca_directory_name',
      -> MASTER_SSL_CERT = 'cert_file_name',
      -> MASTER_SSL_KEY = 'key_file_name';
  ```

マスター情報が更新されたあとに、スレーブレプリケーションプロセスを起動します。

```
mysql> START SLAVE;
```

SHOW SLAVE STATUS ステートメントを使用して、SSL 接続が正常に確立されたことを確認できます。

CHANGE MASTER TO ステートメントの詳細については、セクション13.4.2.1「CHANGE MASTER TO 構文」を参照してください。

レプリケーション中に SSL 接続の使用を適用する場合は、REPLICATION SLAVE 権限のユーザーを作成して、そのユーザーに REQUIRE SSL オプションを使用します。例:

```
mysql> CREATE USER 'repl'@'%.mydomain.com' IDENTIFIED BY 'slavepass';
mysql> GRANT REPLICATION SLAVE ON *.*
    -> TO 'repl'@'%.mydomain.com' REQUIRE SSL;
```

アカウントがすでに存在する場合は、このステートメントでそれに REQUIRE SSL を追加できます。

```
mysql> GRANT USAGE ON *.*
```

```
    -> TO 'repl'@'%.mydomain.com' REQUIRE SSL;
```

17.3.8 準同期レプリケーション

MySQL 5.6 は、非同期レプリケーションを内蔵していますが、さらにプラグインによって実装される準同期レプリケーションへのインタフェースをサポートします。このセクションでは、準同期レプリケーションの概要とその動作について説明します。後続のセクションでは、準同期レプリケーションへの管理インタフェース、およびこれをインストール、構成、およびモニターする方法について説明します。

MySQL レプリケーションはデフォルトで非同期です。マスターはイベントをそのバイナリログに書き込みますが、スレーブがそれらを取得して処理したかどうかまたはその日時を認識しません。非同期レプリケーションでは、マスターがクラッシュしても、それがコミットしたトランザクションがスレーブに転送されていない場合があります。つまり、このケースでマスターからスレーブへのフェイルオーバーが発生すると、マスターに関係するトランザクションがないサーバーにフェイルオーバーされる場合があります。

準同期レプリケーションは非同期レプリケーションの代替として使用できます。

- スレーブは、マスターに接続されるときに準同期対応かどうかを伝えます。
- 準同期レプリケーションがマスター側で有効であり、準同期スレーブが少なくとも 1 つある場合、マスター上でトランザクションコミットを実行するスレッドは、コミット後にブロックされ、少なくとも 1 つの準同期スレーブがそのトランザクションのすべてのイベントを受け取ったことを通知するか、タイムアウトが発生するまで、待機します。
- スレーブは、イベントがそのリレーログに書き込まれてディスクにフラッシュされたあとにのみ、トランザクションのイベントを受け取ったことを通知します。
- スレーブからのトランザクション受け取り通知がない状態でタイムアウトが発生した場合、マスターは非同期レプリケーションに戻ります。少なくとも 1 つの準同期スレーブが追い付いたときに、マスターは準同期レプリケーションに戻ります。
- 準同期レプリケーションはマスターとスレーブの両方で有効になっている必要があります。準同期レプリケーションがマスターで無効になっている場合、またはマスター上で有効でもスレーブ上でそうなっていない場合は、マスターは非同期レプリケーションを使用します。

マスターは、ブロック中 (コミットを実行したあとにスレーブからの通知を待機中) はトランザクションを実行したセッションに戻りません。ブロックが終了すると、マスターはそのセッションに戻り、ほかのステートメントの実行に進めます。この時点で、トランザクションはマスター側でコミット済みで、そのイベントを受け取ったことが少なくとも 1 つのスレーブから通知されています。

ブロックはバイナリログに書き込まれるロールバック後にも発生し、これは非トランザクションテーブルを変更するトランザクションがロールバックされるときに発生します。ロールバックされるトランザクションは、トランザクションテーブルに影響しない場合を含めて、ログが記録されます。非トランザクションテーブルへの変更はロールバックできず、スレーブに送信される必要があるためです。

トランザクションコンテキストで発生しないステートメントの場合 (つまり、トランザクションが START TRANSACTION または SET autocommit = 0 で起動されなかったとき)、自動コミットが有効になっていて、各ステートメントは暗黙的にコミットされます。準同期レプリケーションでは、明示的なトランザクションコミットと同様に、マスターはこのような各ステートメントをコミットしたあとにブロックされます。

「準同期レプリケーション」の「準」の意味を理解するには、非同期および完全同期レプリケーションと比較してください。

- 非同期レプリケーションでは、マスターはイベントをそのバイナリログに書き込み、スレーブは準備ができたときにそれらを要求します。イベントがスレーブに必ず到達することは保証されていません。
- 完全同期レプリケーションでは、マスターがトランザクションをコミットすると、マスターがトランザクションを実行したセッションに戻る前に、すべてのスレーブもトランザクションをコミットします。これの欠点は、トランザクションの完了が大きく遅れる場合があることです。
- 準同期レプリケーションは、非同期および完全同期レプリケーションの中間です。マスターはコミット後、少なくとも 1 つのスレーブがイベントを受け取ってログを記録するまで待機します。すべてのスレーブが受け取りを通知するのを待機せず、スレーブ側でイベントが完全に実行されてコミットされたことではなく、受け取りのみを要求します。

準同期レプリケーションでは、非同期レプリケーションに比べて、データの完全性が向上します。コミットが成功したことが返されると、データが少なくとも 2 つの場所 (マスター上と少なくとも 1 つのスレーブ上) に存

在することがわかります。マスターはコミットしたけれども、マスターがスレーブからの通知を待機中にクラッシュが発生した場合は、トランザクションがスレーブに到達できなかった可能性があります。

準同期レプリケーションは、バイナリログイベントをマスターからスレーブに送信できる速度を制限することで、ビジーセッションに速度制限を設定することもできます。あるユーザーがとてもビジーのときは、これによって速度が遅くなり、一部の配備状況で役立ちます。

準同期レプリケーションは、スレーブを待機する必要性によってコミットが遅くなるため、パフォーマンスに若干影響します。これは、データ完全性の向上とのトレードオフです。速度低下の量は、スレーブにコミットを送信して、スレーブからの受け取りの通知を待機するための、TCP/IP ラウンドトリップ時間以上です。これは、準同期レプリケーションは高速ネットワーク上で通信する近いサーバーに最適で、低速ネットワークで通信する遠いサーバーに最悪であることを意味します。

17.3.8.1 準同期レプリケーション管理インタフェース

準同期レプリケーションへの管理インタフェースにはいくつかのコンポーネントがあります。

- 準同期機能を実装する 2 つのプラグイン。マスター側に 1 つのプラグイン、スレーブ側に 1 つあります。
- プラグインの動作を制御するシステム変数。例:
  - rpl_semi_sync_master_enabled

    準同期レプリケーションがマスター上で有効かどうかを制御します。プラグインを有効または無効にするには、この変数をそれぞれ 1 または 0 に設定します。デフォルトは 0 （ オフ ） です。
  - rpl_semi_sync_master_timeout

    タイムアウトが発生して非同期レプリケーションに戻すまでに、スレーブからの肯定応答のコミットをマスターが待機する時間を制御する、ミリ秒単位の値。デフォルト値は 10000 (10 秒) です。
  - rpl_semi_sync_slave_enabled

    rpl_semi_sync_master_enabled に似ていますが、スレーブプラグインを制御します。

  すべての rpl_semi_sync_xxx システム変数はセクション5.1.4「サーバーシステム変数」で説明されています。
- 準同期レプリケーションモニタリングを有効にするステータス変数。例:
  - Rpl_semi_sync_master_clients

    準同期スレーブの数。
  - Rpl_semi_sync_master_status

    準同期レプリケーションがマスター上で現在動作中であるかどうか。プラグインが有効になっていてコミット通知が発生していない場合、値は 1 です。プラグインが有効になっていないか、コミット通知タイムアウトによりマスターが非同期レプリケーションに戻った場合、これは 0 です。
  - Rpl_semi_sync_master_no_tx

    スレーブによって正しく認証されなかったコミット数。
  - Rpl_semi_sync_master_yes_tx

    スレーブによって正しく認証されたコミットの数。
  - Rpl_semi_sync_slave_status

    準同期レプリケーションがスレーブ上で現在動作中かどうか。プラグインが有効になっていて、スレーブ I/O スレッドが実行中の場合は 1、それ以外の場合は 0 です。

  すべての Rpl_semi_sync_xxx ステータス変数はセクション5.1.6「サーバーステータス変数」で説明されています。

システムおよびステータス変数は、該当するマスターまたはスレーブプラグインが INSTALL PLUGIN でインストールされた場合にのみ利用できます。

17.3.8.2 準同期レプリケーションのインストールと構成

準同期レプリケーションはプラグインを使用して実装されるため、プラグインがサーバーにインストールされて利用できる状態である必要があります。プラグインがインストールされたあと、それに関連付けられたシステム変数によって制御します。これらのシステム変数は、関連付けられたプラグインがインストールされるまで利用できません。

準同期レプリケーションを使用するには、次の要件を満たす必要があります。

- MySQL 5.5 以降がインストールされている必要があります。
- プラグインをインストールする機能には、動的ローディングをサポートする MySQL サーバーが必要です。これを検証するために、have_dynamic_loading システム変数の値が YES であることを確認してください。バイナリ配布は動的ローディングをサポートしているはずです。
- レプリケーションがすでに機能している必要があります。マスター/スレーブ関係の作成については、セクション17.1.1「レプリケーションのセットアップ方法」を参照してください。

準同期レプリケーションをセットアップするには、次の指示を使用してください。ここで示す INSTALL PLUGIN、SET GLOBAL、STOP SLAVE、および START SLAVE ステートメントには、SUPER 権限が必要です。

準同期レプリケーションプラグインは MySQL 配布に含まれています。

コンポーネント配布を解凍します (マスター側とスレーブ側のファイルが含まれます)。

コンポーネントファイルを該当するサーバーのプラグインディレクトリにインストールします。semisync_master* ファイルをマスターサーバーのプラグインディレクトリにインストールします。semisync_slave* ファイルを各スレーブサーバーのプラグインディレクトリにインストールします。プラグインディレクトリの位置は、サーバーの plugin_dir システム変数の値として利用できます。

プラグインをロードするには、準同期にするマスターと各スレーブで INSTALL PLUGIN ステートメントを使用します。

マスターで:

```
mysql> INSTALL PLUGIN rpl_semi_sync_master SONAME 'semisync_master.so';
```

各スレーブで:

```
mysql> INSTALL PLUGIN rpl_semi_sync_slave SONAME 'semisync_slave.so';
```

上述のコマンドは、プラグインファイル名サフィクス .so を使用します。システムで異なるサフィクスを適用してもかまいません。プラグインファイル名に自信がない場合は、サーバーのプラグインディレクトリでプラグインを探してください。

プラグインをインストールしようとして、Linux でここで示すようなエラーが発生する場合は、libimf をインストールする必要があります。

```
mysql> INSTALL PLUGIN rpl_semi_sync_master SONAME 'semisync_master.so';
ERROR 1126 (HY000): Can't open shared library
'/usr/local/mysql/lib/plugin/semisync_master.so' (errno: 22 libimf.so: cannot open
shared object file: No such file or directory)
```

libimf は https://dev.mysql.com/downloads/os-linux.htmlから取得できます。

どのプラグインがインストールされているかを確認するには、SHOW PLUGINS ステートメントを使用するか、INFORMATION_SCHEMA.PLUGINS テーブルを照会してください。

準同期レプリケーションプラグインがインストールされたあとは、デフォルトで無効になっています。準同期レプリケーションを有効にするには、プラグインがマスター側とスレーブ側の両方で有効にする必要があります。一方の側だけが有効の場合には、レプリケーションは非同期になります。

インストールされたプラグインが有効かどうかを制御するには、該当するシステム変数を設定します。これらの変数は実行時に SET GLOBAL を使用して、またはコマンド行またはオプションファイルでサーバー起動時に設定できます。

実行時に、これらのマスター側システム変数を利用できます。

```
mysql> SET GLOBAL rpl_semi_sync_master_enabled = {0|1};
```

```
mysql> SET GLOBAL rpl_semi_sync_master_timeout = N;
```

スレーブ側で、このシステム変数を利用できます。

```
mysql> SET GLOBAL rpl_semi_sync_slave_enabled = {0|1};
```

rpl_semi_sync_master_enabled または rpl_semi_sync_slave_enabled の場合、準同期レプリケーションを有効にするには値を 1、無効にするには 0 にすべきです。デフォルトでは、これらの変数は 0 に設定されています。

rpl_semi_sync_master_timeout の場合、値 N はミリ秒で指定されます。デフォルト値は 10000 (10 秒) です。

実行時にスレーブ上で準同期レプリケーションを有効にする場合、スレーブ I/O スレッドを起動して (すでに実行中の場合はまず停止してから)、スレーブをマスターに接続し、準同期スレーブとして登録する必要があります。

```
mysql> STOP SLAVE IO_THREAD; START SLAVE IO_THREAD;
```

I/O スレッドがすでに実行中で再起動しない場合は、スレーブは非同期レプリケーションを使用し続けます。

サーバー起動時に、準同期レプリケーションを制御する変数をコマンド行オプションとしてまたはオプションファイルに設定できます。オプションファイルにリストされる設定は、サーバーが起動するたびに有効になります。たとえば、マスター側およびスレーブ側の my.cnf ファイルに次のように変数を設定できます。

マスターで:

```
[mysqld]
rpl_semi_sync_master_enabled=1
rpl_semi_sync_master_timeout=1000 # 1 second
```

各スレーブで:

```
[mysqld]
rpl_semi_sync_slave_enabled=1
```

17.3.8.3 準同期レプリケーションモニタリング

準同期レプリケーション機能用のプラグインはいくつのシステム変数とステータス変数を公開しており、その構成および運用状態を判断するためにそれらを調べることができます。

システム変数は準同期レプリケーションがどのように構成されているかを反映します。これらの値を確認するには、SHOW VARIABLES を使用します。

```
mysql> SHOW VARIABLES LIKE 'rpl_semi_sync%';
```

ステータス変数によって、準同期レプリケーションの動作をモニターできます。これらの値を確認するには、SHOW STATUS を使用します。

```
mysql> SHOW STATUS LIKE 'Rpl_semi_sync%';
```

コミット-ブロックがタイムアウトになるかスレーブが追い付いたことにより、マスターが非同期レプリケーションと準同期レプリケーションを切り替えるときは、Rpl_semi_sync_master_status ステータス変数を該当する値に設定します。マスター上で準同期レプリケーションから非同期レプリケーションに自動的にフォールバックすることは、rpl_semi_sync_master_enabled システム変数がマスター側で 1 である可能性があることを意味します (準同期レプリケーションがその時点で実際に動作していなくても)。Rpl_semi_sync_master_status ステータス変数をモニターすることで、マスターが現在非同期または準同期レプリケーションのどちらを使用しているかを判断できます。

多数の準同期スレーブがどのように接続されているかを確認するには、Rpl_semi_sync_master_clients をチェックします。

スレーブから成功または不成功と通知されたコミットの数は、Rpl_semi_sync_master_yes_tx および Rpl_semi_sync_master_no_tx 変数で示されます。

スレーブ側の Rpl_semi_sync_slave_status は、準同期レプリケーションが現在動作しているかどうかを示します。

17.3.9 遅延レプリケーション

MySQL 5.6 は、スレーブサーバーがマスターから意図的に少なくとも指定した時間遅れる、遅延レプリケーションをサポートします。デフォルトの遅延は 0 秒です。CHANGE MASTER TO の MASTER_DELAY オプションを使用して、遅延を N 秒に設定してください。

```
CHANGE MASTER TO MASTER_DELAY = N;
```

マスターから受信するイベントは、マスター上での実行より少なくとも N 秒後になるまで実行されません。例外は、形式記述イベントまたはログファイルローテーションイベントには遅延がなく、これらは SQL スレッドの内部状態にのみ影響します。

遅延レプリケーションはいくつかの目的に使用できます。

- マスター上でのユーザーの誤りから保護するため。DBA は遅延スレーブを障害の直前までロールバックできます。
- 遅延があるときにシステムがどのように動作するかをテストするため。たとえば、アプリケーションで、スレーブでの大きな負荷が原因で遅延が発生する場合があります。しかし、この負荷レベルを生成するのが難しい場合があります。遅延レプリケーションは、負荷をシミュレートしなくても遅延をシミュレートできます。遅延をしているスレーブに関連する条件をデバッグするために使用することもできます。
- バックアップをリロードすることなく、データベースが以前にどのようであったかを調べるため。たとえば、遅延が 1 週間で、最後の数日間の開発の前にデータベースがどのようであったかを DBA が確認する必要がある場合には、遅延スレーブを検査できます。

START SLAVE および STOP SLAVE はすぐに適用され、遅延を無視します。RESET SLAVE は遅延を 0 にリセットします。

SHOW SLAVE STATUS には、遅延に関する情報を提供する次のような 3 つのフィールドがあります。

- SQL_Delay: スレーブがマスターより遅延する必要がある秒数を示す、負でない整数。
- SQL_Remaining_Delay: Slave_SQL_Running_State が Waiting until MASTER_DELAY seconds after master executed event のときに、このフィールドには遅延の残り秒数を示す整数が含まれます。ほかのときは、このフィールドは NULL です。
- Slave_SQL_Running_State: SQL スレッドの状態を示す文字列 (Slave_IO_State に似ています)。値は、SHOW PROCESSLIST で表示される、SQL スレッドの State 値と同じです。

スレーブ SQL スレッドがイベント実行前に遅延が経過するのを待機しているときは、SHOW PROCESSLIST はその State 値を Waiting until MASTER_DELAY seconds after master executed event として表示します。

relay-log.info ファイルに遅延値が含まれるようになり、ファイル形式が変更されました。セクション17.2.2.2「スレーブステータスログ」を参照してください。たとえば、ファイルの最初の行がファイル内の行数を示すようになりました。スレーブサーバーを MySQL 5.6 より前のバージョンにダウングレードすると、古いサーバーはファイルを正しく読み取りません。これに対処するには、テキストエディタでファイルを変更し、行数を含む最初の行を削除します。

17.4 レプリケーションの注釈とヒント

17.4.1 レプリケーションの機能と問題

以降のセクションでは、MySQL レプリケーションでサポートされていることとされていないことに関する情報、および特定のステートメントの複製時に発生する可能性がある固有の問題と状況に関する情報を提供します。

ステートメントベースレプリケーションは、SQL レベルでのマスターとスレーブ間の互換性に依存します。つまり、SBR が成功するには、使用する SQL 機能がマスターおよびスレーブサーバーの両方でサポートされる必要があります。たとえば、マスターサーバーで MySQL 5.6 (以降) でのみ利用できる機能を使用する場合、MySQL 5.5 (以前) を使用するスレーブに複製できません。

このような非互換性は、本番前リリースの MySQL を使用するときのリリースシリーズ内で発生することもあります。たとえば、SLEEP() 関数は MySQL 5.0.12 以降で使用できます。この関数をマスターで使用する場合、MySQL 5.0.11 以前を使用するスレーブに複製できません。

このため、本番設定のステートメントベースレプリケーションには、GA (Generally Available) リリースの SQL を使用してください。あるリリースシリーズが GA リリースステータスに到達すると、オラクルはそのシリーズ内で新しい SQL ステートメントを導入したりそれらの機能を変更したりしないためです。

MySQL 5.6 と 1 つ前の MySQL リリースシリーズとの間でステートメントベースレプリケーションを使用することを計画している場合は、そのシリーズのレプリケーション特性に関する情報について、以前のリリースシリーズに対応するエディションの MySQL リファレンスマニュアルを参照することもお勧めします。

MySQL のステートメントベースレプリケーションでは、ストアドルーチンまたはトリガーの複製で問題が発生する場合があります。これらの問題は、代わりに MySQL の行ベースのレプリケーションを使用することで回避できます。問題の詳細な一覧は、セクション20.7「ストアドプログラムのバイナリロギング」を参照してください。行ベースロギングおよび行ベースレプリケーションに関する詳細は、セクション5.2.4.1「バイナリロギング形式」およびセクション17.1.2「レプリケーション形式」を参照してください。

レプリケーションおよび InnoDB に固有の追加情報については、セクション14.17「InnoDB と MySQL レプリケーション」を参照してください。MySQL Cluster でのレプリケーションに関連する情報は、セクション18.6「MySQL Cluster レプリケーション」を参照してください。

17.4.1.1 レプリケーションと AUTO_INCREMENT

AUTO_INCREMENT、LAST_INSERT_ID()、および TIMESTAMP 値のステートメントレベルレプリケーションは正しく行われますが、次の例外があります。

- MySQL 5.6.10 より前のステートメントベースレプリケーションを使用するときは、スレーブ上のテーブル内の AUTO_INCREMENT カラムは、マスター上の同じカラムに一致する必要があります。つまり、AUTO_INCREMENT カラムは AUTO_INCREMENT カラムに複製される必要があります。(Bug #12669186)
- AUTO_INCREMENT カラムを更新するトリガーまたは関数を呼び出すステートメントは、ステートメントベースレプリケーションでは正しく複製されません。MySQL 5.6 では、このようなステートメントは安全でないとマークされます。(Bug #45677)
- 複合主キーを持ち、この複合キーの先頭カラムでない AUTO_INCREMENT カラムを含むテーブルに INSERT を実行することは、ステートメントベースロギングまたはレプリケーションにとって安全ではありません。MySQL 5.6.6 以降では、このようなステートメントは安全でないとマークされます。(Bug #11754117、Bug #45670)

  この問題は InnoDB ストレージエンジンを使用するテーブルに影響しません。AUTO_INCREMENT カラムを持つ InnoDB テーブルには、自動インクリメントカラムが唯一または左端のカラムであるキーが少なくとも 1 つ必要であるためです。
- ALTER TABLE でテーブルに AUTO_INCREMENT カラムを追加した場合、行の順序がスレーブとマスターで同じにならない場合があります。これが発生するのは、行が番号付けされる順序が、テーブルに使用される固有のストレージエンジンおよび行が挿入された順序に依存するためです。マスターとスレーブで同じ順序を持つことが重要である場合は、行を並べ替えてから AUTO_INCREMENT 番号を割り当てる必要があります。カラム col1 と col2 を持つテーブル t1 に AUTO_INCREMENT カラムを追加するものと仮定すると、次のステートメントは t1 と同じであるけれども AUTO_INCREMENT カラムを持つ新しいテーブル t2 を生成します。

```
CREATE TABLE t2 LIKE t1;
ALTER TABLE t2 ADD id INT AUTO_INCREMENT PRIMARY KEY;
INSERT INTO t2 SELECT * FROM t1 ORDER BY col1, col2;
```

> **重要**
>
> マスターとスレーブの両方で同じ順序を保証するには、ORDER BY 句で t1 のすべてのカラムの名前を指定する必要があります。

上記の指示には CREATE TABLE ... LIKE の制限が適用されます。外部キー定義は DATA DIRECTORY および INDEX DIRECTORY テーブルオプションと同様に無視されます。テーブル定義がこれらの特性を含む場合、t1 の作成に使用したものと同じであるけれども AUTO_INCREMENT カラムを追加した CREATE TABLE ステートメントを使用して、t2 を作成してください。

AUTO_INCREMENT カラムを持つコピーを作成および移入するために使用する方法にかかわらず、最終手順は元のテーブルを削除してコピーの名前を変更することです。

```
DROP t1;
ALTER TABLE t2 RENAME t1;
```

セクションB.5.7.1「ALTER TABLE での問題」も参照してください。

17.4.1.2 レプリケーションと BLACKHOLE テーブル

BLACKHOLE ストレージエンジンはデータを受け入れますが、それを破棄し、格納しません。バイナリロギングを実行するときは、使用している口ギング形式にかかわらず、このようなテーブルへのすべての挿入は常にログが記録されます。更新と削除は、ステートメントベースまたは行ベースのどちらの口ギングが使用されているかによって扱いが異なります。ステートメントベースロギング形式では、BLACKHOLE テーブルに影響するすべ

てのステートメントのログが記録されますが、それらの影響は無視されます。行ベースロギングを使用するときは、このようなテーブルへの更新と削除は単にスキップされ、バイナリログに書き込まれません。MySQL 5.6.12 以降では、これが発生するたびに警告ログが記録されます (Bug #13004581)

このため、BLACKHOLE ストレージエンジンを使用してテーブルに複製するときは、binlog_format サーバー変数を ROW または MIXED ではなく STATEMENT に設定することをお勧めします。

17.4.1.3 レプリケーションと文字セット

次のことは、異なる文字セットを使用する MySQL サーバー間でのレプリケーションに適用されます。

- マスターが MySQL 4.1 を使用する場合、スレーブ MySQL バージョンに関係なく、マスターとスレーブで常に同じグローバル文字セットおよび照合順序を使用する必要があります。(これらは、--character-set-server および --collation-server オプションで制御されます。)そうでない場合、スレーブで重複キーエラーが発生する可能性があります。マスター文字セットで一意のキーがスレーブ文字セットで一意でない場合があるためです。マスターとスレーブが両方とも MySQL 5.0 以降であるときは、これは心配の種にはなりません。

- マスターが MySQL 4.1.3 より古い場合、クライアントの文字セットをそのグローバル値と異なるものにしてはいけません。この文字セットの変更はスレーブに認識されないためです。つまり、クライアントは SET NAMES、SET CHARACTER SET などを使用しないでください。マスターとスレーブの両方が 4.1.3 以降である場合、クライアントは文字セット変数のセッション値を自由に設定できます。これらの設定がバイナリログに書き込まれるため、スレーブに認識されるためです。つまり、クライアントは SET NAMES または SET CHARACTER SET を使用したり、collation_client や collation_server などの変数を設定したりできます。ただし、クライアントはこれらの変数のグローバル値を変更することはできません。前述のように、マスターとスレーブは常に同一グローバル文字セット値を使用する必要があります。これは、ステートメントベースのレプリケーションと行ベースのレプリケーションのどちらを使用している場合にも当てはまります。

- マスターのデータベースの文字セットがグローバル character_set_server 値と異なる場合、データベースのデフォルト文字セットを暗黙的に信頼しないように CREATE TABLE ステートメントを設計してください。推奨される回避策は、CREATE TABLE ステートメントに明示的に文字セットと照合順序を指定することです。

17.4.1.4 CREATE ... IF NOT EXISTS ステートメントのレプリケーション

MySQL は、さまざまな CREATE ... IF NOT EXISTS ステートメントが複製されるときにこれらの値を適用します。

- 各 CREATE DATABASE IF NOT EXISTS ステートメントは、データベースがマスター上にすでに存在しているかどうかにかかわらず、複製されます。

- 同様に、SELECT のない各 CREATE TABLE IF NOT EXISTS ステートメントは、テーブルがマスター上にすでに存在しているかどうかにかかわらず、複製されます。これは CREATE TABLE IF NOT EXISTS ... LIKE を含みます。CREATE TABLE IF NOT EXISTS ... SELECT のレプリケーションは、多少異なるルールに従います。詳細については、セクション17.4.1.5「CREATE TABLE ... SELECT ステートメントのレプリケーション」を参照してください。

- CREATE EVENT IF NOT EXISTS は、ステートメントに指定されたイベントがすでにマスター上に存在しているかどうかにかかわらず、MySQL 5.6 で常に複製されます。

Bug #45574 も参照してください。

17.4.1.5 CREATE TABLE ... SELECT ステートメントのレプリケーション

このセクションでは、MySQL が CREATE TABLE ... SELECT ステートメントを複製する方法について説明します。

MySQL 5.6 は、CREATE TABLE ... SELECT ステートメントが、このステートメントで作成されたテーブル以外のテーブルで変更を行うことを許可しません。これは、以前のバージョンの MySQL (これらのステートメントがそうすることを許可していました) からの動作の変更です。これは、MySQL 5.6 以降のスレーブと前のバージョンの MySQL を実行しているマスターとの間でステートメントベースレプリケーションを使用するときに、マスター上のほかのテーブルで変更を行う CREATE TABLE ... SELECT ステートメントがスレーブ上で失敗し、レプリケーションが停止することを意味します。これが発生しないようにするには、行ベースレプリケーションを使用するか、問題のステートメントをマスター上で実行する前に書き直すか、マスターを MySQL 5.6 (以降) にアップグレードしてください。(マスターをアップグレードすることを選択する場合、このような CREATE TABLE ... SELECT ステートメントは、ほかのテーブルでの副作用をなくすように書き直さないかぎり、アップグレード後に失敗することに留意してください。)これは、行ベースのプリケーションを使用するときは問題になりません。このステートメントのログが、CREATE TABLE ... SELECT 全体としてではなく、CREATE TABLE ステートメントとして記録される (テーブルデータへの変更のログが行挿入イベントとして記録される) ためです。

これらの動作は MySQL バージョンに依存しません。

- CREATE TABLE ... SELECT は常に暗黙的コミットを実行します (セクション13.3.3「暗黙的なコミットを発生させるステートメント」)。
- 目的のテーブルが存在しない場合、ロギングは次のように発生します。IF NOT EXISTS が存在するかどうかは重要ではありません。
  - STATEMENT または MIXED 形式: ステートメントは、書き込まれたものとしてログに記録されます。
  - ROW 形式: ステートメントは、CREATE TABLE ステートメントおよび一連の行挿入イベントとしてログに記録されます。
- ステートメントが失敗した場合、ログは記録されません。これには、宛先テーブルが存在し、IF NOT EXISTS が指定されていないケースが含まれます。

宛先テーブルが存在し、IF NOT EXISTS が指定されているときは、MySQL はバージョンに依存する方法でステートメントを処理します。

5.1.51 より前の MySQL 5.1 および 5.5.6 より前の MySQL 5.5 で (これがオリジナルの動作です):

- STATEMENT または MIXED 形式: ステートメントは、書き込まれたものとしてログに記録されます。
- ROW 形式: ステートメントは、CREATE TABLE ステートメントおよび一連の行挿入イベントとしてログに記録されます。

5.1.51 以降の MySQL 5.1 で:

- STATEMENT または MIXED 形式: ステートメントは、CREATE TABLE および INSERT INTO ... SELECT ステートメントの同等ペアとしてログに記録されます。
- ROW 形式: ステートメントは、CREATE TABLE ステートメントおよび一連の行挿入イベントとしてログに記録されます。

5.5.6 以降の MySQL 5.5 で:

- 何も挿入されず、ログが記録されません。

これらのバージョン依存が発生するのは、MySQL 5.5.6 で、目的のテーブルがすでに存在する場合に行を挿入しないように CREATE TABLE ... SELECT の処理が変更されたことと、MySQL 5.1.51 で、このようなステートメントを 5.1 マスターから 5.5 スレーブに複製する際の上位互換性を保持するように変更されたことによります。詳細については、セクション13.1.17.1「CREATE TABLE ... SELECT 構文」を参照してください。

17.4.1.6 CREATE SERVER、ALTER SERVER、および DROP SERVER のレプリケーション

MySQL 5.6 では、ステートメント CREATE SERVER、ALTER SERVER、および DROP SERVER は、使用中のバイナリロギング形式にかかわらず、バイナリログに書き込まれません。

17.4.1.7 CURRENT_USER() のレプリケーション

次のステートメントでは、CURRENT_USER() 関数を使用して、該当するユーザーまたは定義者の名前 (あるいはそれらのホスト) を置き換えることがサポートされます。そのような場合、CURRENT_USER() は必要に応じて展開されます。

- DROP USER
- RENAME USER
- GRANT
- REVOKE
- CREATE FUNCTION
- CREATE PROCEDURE
- CREATE TRIGGER
- CREATE EVENT

- CREATE VIEW
- ALTER EVENT
- ALTER VIEW
- SET PASSWORD

バイナリロギングが有効のときに CURRENT_USER() または CURRENT_USER がステートメント CREATE FUNCTION、CREATE PROCEDURE、CREATE TRIGGER、CREATE EVENT、CREATE VIEW、または ALTER VIEW のいずれかで定義者として使用されるときは、関数参照はバイナリログに書き込まれる前に展開されるため、これらのステートメントが複製されるときはステートメントはマスターとスレーブの両方で同じユーザーを参照します。CURRENT_USER() または CURRENT_USER は、DROP USER、RENAME USER、GRANT、REVOKE、または ALTER EVENT で使用されるときも、バイナリログに書き込まれる前に展開されます。

17.4.1.8 DROP ... IF EXISTS ステートメントのレプリケーション

DROP DATABASE IF EXISTS、DROP TABLE IF EXISTS、および DROP VIEW IF EXISTS ステートメントは常に複製されます (削除するデータベース、テーブル、またはビューがマスター上に存在しない場合でも)。これは、スレーブがマスターに追い付いたあと、削除するオブジェクトがマスターとスレーブのどちらにも確実に存在しないようにするためです。

ストアドプログラム (ストアドプロシージャーと関数、トリガー、およびイベント) 用の DROP ... IF EXISTS ステートメントも複製されます (削除するストアドプログラムがマスターに存在しない場合でも)。

17.4.1.9 テーブル定義が異なるマスターとスレーブでのレプリケーション

レプリケーションのソースおよびターゲットテーブルは同じである必要はありません。マスター上のテーブルのカラム数が、テーブルのスレーブコピーのものより多くても少なくてもかまいません。また、マスターおよびスレーブ上の対応するテーブルカラムが、一定の条件が適用されますが、異なるデータ型を使用していてもかまいません。

ソースおよびターゲットテーブルの定義が同じでない場合でも、データベースおよびテーブルの名前はマスターとスレーブの両方で同じである必要があります。次の 2 つのセクションで、追加条件について例を示して説明します。

マスターまたはスレーブでカラムが多い場合のレプリケーション

マスターとスレーブコピーでテーブルのカラム数が異なる場合でも、次の条件に従って、テーブルをマスターからスレーブに複製できます。

- 両方のバージョンのテーブルに共通するカラムは、マスターとスレーブで同じ順番に定義する必要があります。

  (これは、両方のテーブルのカラム数が同じ場合でも当てはまります。)

- 両方のバージョンのテーブルに共通するカラムは、追加カラムの前に定義する必要があります。

  これは、スレーブで ALTER TABLE ステートメントを実行して、新しいカラムが両方のテーブルに共通するカラムの範囲内にテーブルに挿入される場合は、次の例で示すように、レプリケーションが失敗することを意味します。

  テーブル t がマスターおよびスレーブ上に存在し、次の CREATE TABLE ステートメントで定義されているとします。

```
CREATE TABLE t (
  c1 INT,
  c2 INT,
  c3 INT
);
```

  ここで示す ALTER TABLE ステートメントがスレーブで実行されるとします。

```
ALTER TABLE t ADD COLUMN cnew1 INT AFTER c3;
```

  上記の ALTER TABLE はスレーブ上で許可されます。テーブル t の両方のバージョンに共通するカラム c1、c2、および c3 が両方のバージョンのテーブルでまとまったままであり、共通しないカラムはそのあとに来るためです。

しかし、次の ALTER TABLE ステートメントをスレーブ上で実行すると、レプリケーションは必ず失敗します。

```
ALTER TABLE t ADD COLUMN cnew2 INT AFTER c2;
```

ここで示す ALTER TABLE ステートメントをスレーブ上で実行すると、新しいカラム cnew2 が両方のバージョンの t に共通するカラムの間に来るため、レプリケーションは失敗します。

- カラム数の多いバージョンのテーブルの「追加」カラムごとに、デフォルト値が必要です。

  カラムのデフォルト値は、その型、DEFAULT オプションで定義されているかどうか、NULL として宣言されているかどうか、作成時に有効であったサーバー SQL モードなど、いくつかの要因で決まります。詳細については、セクション11.6「データ型デフォルト値」を参照してください)。

また、テーブルのスレーブコピーがマスターのコピーよりカラム数が多いときは、テーブルに共通する各カラムは両方のテーブルで同じデータ型を使用する必要があります。

例　次の例は、有効および無効なテーブル定義をいくつか示します。

マスターのカラム数が多い　次のテーブル定義は有効で、正しく複製されます。

```
master> CREATE TABLE t1 (c1 INT, c2 INT, c3 INT);
slave>  CREATE TABLE t1 (c1 INT, c2 INT);
```

次のテーブル定義は、両方のバージョンのテーブルに共通するカラムの定義がスレーブ上とマスター上とで順番が異なるため、エラー 1532 (ER_BINLOG_ROW_RBR_TO_SBR) が発生します。

```
master> CREATE TABLE t1 (c1 INT, c2 INT, c3 INT);
slave>  CREATE TABLE t1 (c2 INT, c1 INT);
```

次のテーブル定義でも、両方のバージョンのテーブルに共通するカラムの定義の前に、マスター上の追加カラムの定義があるため、エラー 1532 が発生します。

```
master> CREATE TABLE t1 (c3 INT, c1 INT, c2 INT);
slave>  CREATE TABLE t1 (c1 INT, c2 INT);
```

スレーブのカラム数が多い　次のテーブル定義は有効で、正しく複製されます。

```
master> CREATE TABLE t1 (c1 INT, c2 INT);
slave>  CREATE TABLE t1 (c1 INT, c2 INT, c3 INT);
```

次の定義では、両方のバージョンのテーブルに共通するカラムが、マスターとスレーブの両方で同じ順序で定義されていないため、エラー 1532 が発生します。

```
master> CREATE TABLE t1 (c1 INT, c2 INT);
slave>  CREATE TABLE t1 (c2 INT, c1 INT, c3 INT);
```

次のテーブル定義でも、両方のバージョンのテーブルに共通のするカラムの定義の前に、スレーブバージョンのテーブルの追加カラムの定義があるため、エラー 1532 が発生します。

```
master> CREATE TABLE t1 (c1 INT, c2 INT);
slave>  CREATE TABLE t1 (c3 INT, c1 INT, c2 INT);
```

次のテーブル定義は、スレーブバージョンのテーブルはマスターバージョンに比べてカラム数が多く、2 つのバージョンのテーブルに共通するカラム c2 が使用するデータ型が異なっているため、失敗します。

```
master> CREATE TABLE t1 (c1 INT, c2 BIGINT);
slave>  CREATE TABLE t1 (c1 INT, c2 INT, c3 INT);
```

データ型が異なるカラムのレプリケーション

同じテーブルのマスターコピーとスレーブコピー上の対応するカラムは、理想的には同じデータ型であるべきです。ただし MySQL 5.1.21 以降では、一定の条件が満たされるかぎり、これは必ずしも厳密に適用されません。

ほかのすべてのことが等しい場合、あるデータ型のカラムから、同じ型、同じサイズまたは幅 (該当する場合は、それより大きい) の別のカラムに複製することは常に可能です。たとえば、CHAR(10) カラムから別の CHAR(10) に、または CHAR(10) カラムから CHAR(25) カラムに、問題なく複製できます。場合によっては、あるデータ型を持つカラム (マスター上) から異なるデータ型を持つカラム (スレーブ上) に複製できることもあります。マ

スターバージョンのカラムのデータ型が、同じサイズまたはそれより大きい型に昇格されるとき、これは属性昇格と呼ばれます。

属性昇格は、ステートメントベースおよび行ベースレプリケーションで使用でき、マスターまたはスレーブで使用されるストレージエンジンに依存しません。ただし、ロギング形式の選択は許可される型変換に影響します。詳細は、このセクションで後述します。

> **重要**
>
> ステートメントベースまたは行ベースのどちらのレプリケーションを使用するときでも、属性昇格を使用する場合はテーブルのスレーブコピーのカラム数をマスターコピーより多くすることはできません。

ステートメントベースのレプリケーション　ステートメントベースレプリケーションを使用するときに従うべき簡単な経験則は、「マスター上で実行されるステートメントがスレーブ上でも正常に実行される場合は、複製も成功するはず」ということです。つまり、スレーブ上のあるカラムの型と互換性のある値をステートメントが使用する場合、そのステートメントは複製できます。たとえば、TINYINT カラムに適合する値を BIGINT カラムにも挿入できます。これは、テーブルのスレーブコピー内の TINYINT カラムの型を BIGINT に変更する場合でも、マスター上で成功するそのカラムへの挿入は、スレーブ上でも成功するはずである、ということに従っています (BIGINT カラムを越えるほど大きな、正当な TINYINT 値を持つことができないため)。

MySQL 5.6.10 より前では、ステートメントベースレプリケーションを使用するときに、AUTO_INCREMENT カラムはマスターとスレーブの両方で同じである必要がありました。そうでない場合、スレーブ上の間違ったテーブルに更新が適用される可能性がありました (Bug #12669186)

行ベースレプリケーション: 属性の昇格と降格　MySQL 5.6 の行ベースレプリケーションは、小さなデータ型と大きなデータ型の間の属性昇格および降格をサポートします。このセクションで後述するように、降格されるカラム値の不可逆 (切り捨て) または非不可逆変換を許可するかどうかを指定することもできます。

不可逆および非不可逆変換　ターゲット型が挿入される値を表現できない場合、変換をどのように扱うかについての決定が必要になります。変換を許可するけれども、ターゲットカラムで「適合」を実現するためにソース値を切り捨てる (または変更する) 場合、不可逆変換と呼ばれることを行います。ソースカラム値をターゲットカラムに適合させるために切り捨てまたは同様の変更を必要としない変換は、非不可逆変換です。

型変換モード (slave_type_conversions 変数)　slave_type_conversions グローバルサーバー変数の設定は、スレーブで使用される型変換モードを制御します。この変数は次の表 (スレーブの型変換動作に対する各モードの影響を示す) からの値セットを取ります。

モード	影響
ALL_LOSSY	このモードでは、情報の損失を意味する型変換が許可されます。 これは非不可逆変換が許可されることを暗示せず、不可逆変換を必要とするまたは変換をまったく必要としないケースのみが許可されることだけを暗示します。たとえば、このモードのみを有効にした場合、INT カラムが TINYINT に変換されること (不可逆変換) は許可されますが、TINYINT カラムが INT カラムに変換されること (非不可逆) は許可されません。このケースで後者の変換を試みると、レプリケーションがスレーブ上のエラーで停止します。
ALL_NON_LOSSY	このモードは、ソース値の切り捨てまたは特別処理を必要とない変換を許可します。すなわち、ターゲット型の範囲がソース型より広い変換を許可します。 このモードを設定することは、不可逆変換が許可されるかどうかに関係ありません。これは ALL_LOSSY モードで制御されます。ALL_NON_LOSSY だけが設定され、ALL_LOSSY はされない場合、データが損失する変換を試みると (INT から TINYINT へ、CHAR(25) から VARCHAR(20) へなど)、スレーブはエラーで停止します。
ALL_LOSSY,ALL_NON_LOSSY	このモードが設定されると、サポートされるすべての型変換が、不可逆変換かどうかにかかわらず、許可されます。
ALL_SIGNED	昇格される整数型を符号付き値として扱います (デフォルト動作)。
ALL_UNSIGNED	昇格される整数型を符号なし値として扱います。
ALL_SIGNED,ALL_UNSIGNED	昇格される整数型を、可能な場合符号付きとして、そうでない場合は符号なしとして扱います。

モード	影響
[empty]	slave_type_conversions が設定されていなときは、属性の昇格または降格は許可されません。これは、ソースおよびターゲットテーブル内のすべてのカラムが同じ型である必要があることを意味します。 このモードがデフォルトです。

整数型が昇格されるときに、符号ありか符号なしかは保持されません。デフォルトでは、スレーブはこのような値をすべて符号付きとして扱います。MySQL 5.6.13 以降では、ALL_SIGNED、ALL_UNSIGNED、またはその両方を使用してこの動作を制御できます。(Bug#15831300) ALL_SIGNED は昇格されるすべての整数型を符号付きとして扱うようにスレーブに指示します。ALL_UNSIGNED はこれらを符号なしとして扱うように指示します。両方を指定すると、スレーブは、可能な場合は値を符号付きとして扱い、そうでない場合は符号なしとして扱います。リストされている順序は重要ではありません。少なくとも ALL_LOSSY または ALL_NONLOSSY のいずれかが使用されていない場合は、ALL_SIGNED も ALL_UNSIGNED も効果を持ちません。

型変換モードを変更するには、新しい slave_type_conversions 設定でスレーブを再起動する必要があります。

サポートされる変換　違うけれども似ているデータ型の間でサポートされる変換を次のリストに示します。

- 整数型 TINYINT、SMALLINT、MEDIUMINT、INT、および BIGINT のいずれかの間。

  これには、これらの型の符号付きおよび符号なしバージョンの間の変換が含まれます。

  不可逆変換は、ソース値をターゲットカラムで許可される最大値 (または最小値) に切り捨てることで行われます。符号なしから符号付き型への非不可逆変換を保証するには、ターゲットカラムがソースカラム内の値の範囲を受け入れるのに十分な大きさである必要があります。たとえば、TINYINT UNSIGNED は、非不可逆に SMALLINT に降格できますが、TINYINT にはできません。

- 小数点型 DECIMAL、FLOAT、DOUBLE、および NUMERIC のいずれかの間。

  FLOAT から DOUBLE へは非不可逆変換です。DOUBLE から FLOAT へは不可逆にしか扱えません。DECIMAL(M,D) から DECIMAL(M',D') への変換 (D' >= D および (M'-D') >= (M-D)) は非不可逆です。M' < M、D' < D、またはその両方の場合、不可逆変換だけを行うことができます。

  いずれかの小数点型の場合、格納される値をターゲット型に適合させることができない場合は、このマニュアルのほかの場所でサーバーに定義される丸めルールに従って値が切り捨てられます。小数点型でこれがどのように実行されるかについては、セクション12.20.4「丸め動作」を参照してください。

- 文字列型 CHAR、VARCHAR、および TEXT のいずれかの間 (異なる幅の間での変換を含む)。

  CHAR、VARCHAR、または TEXT から、同じまたはそれより大きいサイズの CHAR、VARCHAR、または TEXT カラムへの変換は、決して不可逆ではありません。不可逆変換は、スレーブ上の文字列の最初の N 文字だけを挿入することで処理されます。ここで、N はターゲットカラムの幅です。

  > **重要**
  >
  > 異なる文字セットを使用するカラム間のレプリケーションはサポートされません。

- バイナリデータ型 BINARY、VARBINARY、および BLOB のいずれかの間 (異なる幅の間での変換を含む)。

  BINARY、VARBINARY、または BLOB から、同じまたはそれより大きいサイズの BINARY、VARBINARY、または BLOB カラムへの変換は、決して不可逆ではありません。不可逆変換は、スレーブ上の文字列の最初の N バイトだけを挿入することで扱われます。ここで、N はターゲットカラムの幅です。

- 任意の 2 つのサイズの任意の 2 つの BIT カラムの間。

  BIT(M) カラムからの値を BIT(M') カラムに挿入するときは (M' > M)、BIT(M') カラムの最上位ビットがクリアされ (ゼロに設定され)、BIT(M) 値の M ビットが BIT(M') カラムの最下位ビットとして設定されます。

  ソース BIT(M) カラムからの値をターゲット BIT(M') カラムに挿入するときは (M' < M)、BIT(M') カラムの可能な最大値が割り当てられます。つまり、「すべてが設定された」値がターゲットカラムに割り当てられます。

前のリストにない型の間の変換は許可されません。

MySQL 5.5.3 以前でのレプリケーション型変換　MySQL 5.5.3 より前は、行ベースバイナリロギングで、TINYINT から BIGINT へなど、異なる INT サブ型の間で複製することはできませんでした。行ベースロギングを使用するときは、これらの型のカラムへの変更が、バイナリログ内で互いに異なる方法で表現されていたた

めです。(ただし、行ベースレプリケーションを使用して BLOB から TEXT に複製することはできました。BLOB および TEXT カラムへの変更がバイナリログ内で同じ形式を使用して表現されていたためです。)

MySQL 5.5.3 より前の行ベースレプリケーションを使用するときに、属性昇格でサポートされる変換を次の表に示します。

変換前 (マスター)	変換後 (スレーブ)
BINARY	CHAR
BLOB	TEXT
CHAR	BINARY
DECIMAL	NUMERIC
NUMERIC	DECIMAL
TEXT	BLOB
VARBINARY	VARCHAR
VARCHAR	VARBINARY

> 注記
>
> すべてのケースで、スレーブ上のカラムのサイズまたは幅は、マスター上のカラムのもの以上にする必要があります。たとえば、マスター上の CHAR(10) カラムから、スレーブ上の BINARY(10) または BINARY(25) を使用するカラムに複製できましたが、マスター上の CHAR(10) カラムからスレーブ上の BINARY(5) カラムに複製することはできませんでした。
>
> プリフィクスを持つ一意インデックス (主キーを含む) は、マスターとスレーブの両方で同じ長さのプリフィクスを使用する必要があります。このような場合、異なるプリフィクス長は許可されません。プリフィクス長がマスターとスレーブで異なる一意でないインデックスを使用することは可能ですが、これによって重大なパフォーマンス問題が発生する場合があります (特に、マスター上で使用されるプリフィクスの方が長いとき)。これは、ある長さの 2 つの一意プリフィクスが、長さが短くなると一意でなくなる場合があるという事実によります。たとえば、単語 catalogue と catamount は、それぞれ 5 文字のプリフィクス catal と catam を持ちますが、4 文字の同じプリフィクス (cata) を共有します。これにより、このようなインデックスを使用するクエリーが、同じインデックスのスレーブ定義でマスター上のものより短いプリフィクスが使用されるときに、スレーブ上での実行効率が低下する可能性があります。
>
> DECIMAL および NUMERIC カラムの場合、仮数 (M) と小数以下の桁数 (D) の両方が、マスターに比べてスレーブ上で同じまたはそれより大きいサイズである必要があります。たとえば、NUMERIC(5,4) から DECIMAL(6,4) へのレプリケーションは機能しましたが、NUMERIC(5,4) から DECIMAL(5,3) へのレプリケーションはしませんでした。

MySQL 5.5.3 より前は、行ベースレプリケーションを使用するときに、MySQL レプリケーションは次のデータ型とほかのデータ型との間で属性昇格をサポートしませんでした。

- INT (TINYINT、SMALLINT、MEDIUMINT、BIGINT を含む)。

  INT サブ型間の昇格 (たとえば、SMALLINT から BIGINT へ) も、MySQL 5.5.3 より前はサポートされませんでした。

- SET または ENUM。
- FLOAT または DOUBLE。
- 日付、時間、またはその両方に関連するデータ型のすべて: DATE、TIME、DATETIME、TIMESTAMP、および YEAR。

17.4.1.10 レプリケーションと DIRECTORY テーブルオプション

DATA DIRECTORY または INDEX DIRECTORY テーブルオプションが、マスターサーバー上の CREATE TABLE ステートメントで使用されている場合、そのテーブルオプションはスレーブでも使用されます。このため、対応するディレクトリがスレーブホストファイルシステムに存在しない場合、または存在するけれどもスレーブサーバーにアクセスできない場合は、問題が発生する可能性があります。これはスレーブ上で NO_DIR_IN_CREATE サーバー SQL モードを使用することでオーバーライドでき、これによってスレーブは CREATE TABLE ステー

トメントを複製するときに DATA DIRECTORY および INDEX DIRECTORY テーブルオプションを無視します。その結果、テーブルのデータベースディレクトリ内に MyISAM データおよびインデックスファイルが作成されます。

詳細は、セクション5.1.7「サーバー SQL モード」を参照してください。

17.4.1.11 呼び出される機能のレプリケーション

ユーザー定義関数 (UDF) やストアドプログラム (ストアドプロシージャーと関数、トリガー、およびイベント) などの呼び出される機能のレプリケーションには、次の特徴があります。

- 機能の影響は常に複製されます。
- 次のステートメントはステートメントベースレプリケーションを使用して複製されます。
  - CREATE EVENT
  - ALTER EVENT
  - DROP EVENT
  - CREATE PROCEDURE
  - DROP PROCEDURE
  - CREATE FUNCTION
  - DROP FUNCTION
  - CREATE TRIGGER
  - DROP TRIGGER

  ただし、これらのステートメントを使用して作成、変更、または削除される機能の影響は、行ベースレプリケーションを使用して複製されます。

  > **注記**
  >
  > 呼び出される機能をステートメントベースレプリケーションを使用して複製しようとすると、警告が生成されます: Statement is not safe to log in statement format。たとえば、ステートメントベースプリケーションで UDF を複製しようとすると、MySQL サーバーは現在 UDF が決定的かどうかを判断できないため、この警告が生成されます。呼び出される機能の影響が決定的であることを確実にわかっている場合は、このような警告を安全に無視できます。

- CREATE EVENT および ALTER EVENT の場合:
  - イベントのステータスは、指定された状態にかかわらず、スレーブ上で SLAVESIDE_DISABLED に設定されます (これは DROP EVENT には適用されません)。
  - イベントが作成されたマスターは、スレーブ上でそのサーバー ID によって識別されます。INFORMATION_SCHEMA.EVENTS 内の ORIGINATOR カラムおよび mysql.event 内の originator カラムにこの情報が格納されます。詳しくは、セクション21.7「INFORMATION_SCHEMA EVENTS テーブル」およびセクション13.7.5.19「SHOW EVENTS 構文」を参照してください。
- 機能実装は、マスターが失敗してもイベント処理を失うことなくスレーブをマスターとして使用できるように、再生可能な状態でスレーブ上に存在します。

別のサーバー (レプリケーションマスターとして動作していました) 上で作成されたスケジュールされたイベントが MySQL サーバー上にあるかどうかを判断するには、ここで示すような方法で INFORMATION_SCHEMA.EVENTS テーブルを照会してください。

```
SELECT EVENT_SCHEMA, EVENT_NAME
   FROM INFORMATION_SCHEMA.EVENTS
   WHERE STATUS = 'SLAVESIDE_DISABLED';
```

また、SHOW EVENTS ステートメントを次のように使用できます。

```
SHOW EVENTS
```

```
  WHERE STATUS = 'SLAVESIDE_DISABLED';
```

このようなイベントを持つレプリケーションスレーブをレプリケーションマスターに昇格するときは、ALTER EVENT event_name ENABLED を使用して各イベントを有効にする必要があります。ここで、event_name はイベントの名前です。

複数のマスターがこのスレーブ上でイベントを作成することに使用され、サーバー ID master_id を持つマスター上で作成されたイベントのみを識別したい場合は、ここで示すように、EVENTS テーブルに対する先ほどのクエリーを変更して ORIGINATOR カラムを追加してください。

```
SELECT EVENT_SCHEMA, EVENT_NAME, ORIGINATOR
  FROM INFORMATION_SCHEMA.EVENTS
  WHERE STATUS = 'SLAVESIDE_DISABLED'
  AND   ORIGINATOR = 'master_id'
```

同じような方法で SHOW EVENTS ステートメントで ORIGINATOR を使用できます。

```
SHOW EVENTS
  WHERE STATUS = 'SLAVESIDE_DISABLED'
  AND   ORIGINATOR = 'master_id'
```

マスターから複製されたイベントを有効にする前に、スレーブ上で MySQL イベントスケジューラを無効にし (SET GLOBAL event_scheduler = OFF; などのステートメントを使用して)、必要な ALTER EVENT ステートメントを実行し、サーバーを再起動し、その後スレーブ上でイベントスケジューラを再度有効にしてください (SET GLOBAL event_scheduler = ON; などのステートメントを使用)。

あとで新しいマスターをレプリケーションスレーブに降格する場合は、ALTER EVENT ステートメントで有効にしたすべてのイベントを手動で無効にする必要があります。これは、前に示した SELECT ステートメントからのイベントの名前を別個のテーブルに格納するか、ALTER EVENT ステートメントを使用してイベントを識別する共通プリフィクス (replicated_ など) でそれらの名前を変更することで、行うことができます。

イベントの名前を変更した場合は、このサーバーをレプリケーションスレーブに降格するときに、ここで示すように EVENTS テーブルを照会することでイベントを識別できます。

```
SELECT CONCAT(EVENT_SCHEMA, '.', EVENT_NAME) AS 'Db.Event'
    FROM INFORMATION_SCHEMA.EVENTS
    WHERE INSTR(EVENT_NAME, 'replicated_') = 1;
```

17.4.1.12 レプリケーションと浮動小数点値

ステートメントベースレプリケーションでは、値は 10 進からバイナリに変換されます。それらの表現を 10 進とバイナリの間で変換すると近似値になる場合があるため、浮動小数点値を含む比較が不正確になります。これは、浮動小数点値を明示的に使用したり、浮動小数点に暗黙的に変換された値を使用したりする演算に当てはまります。コンピュータアーキテクチャーや MySQL をビルドするために使用されたコンパイラなどの違いにより、浮動小数点値の比較がマスターおよびスレーブサーバーで異なる結果になる場合があります。セクション 12.2「式評価での型変換」およびセクションB.5.5.8「浮動小数点値に関する問題」を参照してください。

17.4.1.13 レプリケーションと小数秒サポート

MySQL 5.6.4 以降では、マイクロ秒 (6 桁) までの精度を持つ TIME、DATETIME、および TIMESTAMP 値の小数秒に対応できるようになりました。セクション11.3.6「時間値での小数秒」を参照してください。

小数秒を理解するマスターサーバーから理解しない古いスレーブに複製する際、問題が発生することがあります。

- fsp (小数秒精度) 値が 0 より大きいカラムを含む CREATE TABLE ステートメントの場合、レプリケーションはパーサーエラーで失敗します。
- fsp 値が 0 の一時データ型を使用するステートメントは、ステートメントベースロギングでは機能しますが、行ベースロギングでは機能しません。後者の場合、このデータ型がマスター上で持つバイナリ形式と型コードは、スレーブ上のものと異なります。
- いくつかの式の結果がマスター上とスレーブ上で異なります。例: マスター上では、timestamp システム変数はマイクロ秒小数部を含む値を返し、スレーブ上では整数を返します。マスター上では、現在時間を含む結果を返す関数 (CURTIME()、SYSDATE()、UTC_TIMESTAMP() など) は引数を fsp 値として解釈し、戻り値はその多くの桁の小数秒部を含みます。スレーブ上では、これらの関数は引数を許可しますが無視します。

17.4.1.14 レプリケーションと FLUSH

一部の形式の FLUSH ステートメント (FLUSH LOGS、FLUSH MASTER、FLUSH SLAVE、および FLUSH TABLES WITH READ LOCK) は、それらがスレーブに複製される場合に問題が発生する可能性があるため、ログに記録されません。構文例は、セクション13.7.6.3「FLUSH 構文」を参照してください。 FLUSH TABLES、ANALYZE TABLE、OPTIMIZE TABLE、および REPAIR TABLE ステートメントはバイナリログに書き込まれるため、スレーブに複製されます。これらのステートメントはテーブルデータを変更しないため、通常は問題ではありません。

ただし、ある状況では、この動作が問題になる場合があります。mysql データベース内の権限テーブルを複製し、これらのテーブルを GRANT を使用しないで直接更新する場合は、スレーブ上で FLUSH PRIVILEGES を発行して新しい権限を有効にする必要があります。また、MERGE テーブルの一部である MyISAM テーブルの名前を変更するときに FLUSH TABLES を使用する場合は、スレーブ上で FLUSH TABLES を手動で発行する必要があります。NO_WRITE_TO_BINLOG またはその エイリアスの LOCAL を指定しない場合、これらのステートメントはバイナリログに書き込まれます。

17.4.1.15 レプリケーションとシステム関数

一部の関数は条件によっては適切に複製されません。

- USER()、CURRENT_USER() (または CURRENT_USER)、UUID()、VERSION()、および LOAD_FILE() 関数は、変更なしで複製されますが、行ベースレプリケーションが有効である場合を除いてスレーブ上での機能に信頼性がありません。(セクション17.1.2「レプリケーション形式」を参照してください。)

  USER() および CURRENT_USER() は、MIXED モード使用時に行ベースレプリケーションを使用して自動的に複製され、STATEMENT モードでは警告を生成します。(セクション17.4.1.7「CURRENT_USER() のレプリケーション」も参照してください。)これは、VERSION() および RAND() にも当てはまります。

- NOW() の場合、バイナリログはタイムスタンプを含みます。これは、マスター上でこの関数への呼び出しによって戻される値がスレーブに複製されることを意味します。タイムゾーンが異なる MySQL サーバー間で複製するときの予期しない結果を回避するには、マスターとスレーブの両方でタイムゾーンを設定してください。セクション17.4.1.30「レプリケーションとタイムゾーン」も参照してください。

  タイムゾーンが異なるサーバー間で複製するときに発生する可能性のある問題を説明するために、マスターはニューヨークにあり、スレーブはストックホルムにあり、両方のサーバーが現地時間を使用しているものとします。さらに、ここで示すようにマスター上で、テーブル mytable を作成し、このテーブルで INSERT ステートメントを実行してから、テーブルから選択するものとします。

```
mysql> CREATE TABLE mytable (mycol TEXT);
Query OK, 0 rows affected (0.06 sec)

mysql> INSERT INTO mytable VALUES ( NOW() );
Query OK, 1 row affected (0.00 sec)

mysql> SELECT * FROM mytable;
+---------------------+
| mycol               |
+---------------------+
| 2009-09-01 12:00:00 |
+---------------------+
1 row in set (0.00 sec)
```

  ストックホルムの現地時間はニューヨークより 6 時間遅れます。このため、スレーブ上で SELECT NOW() を完全に同じタイミングで発行すると、値 2009-09-01 18:00:00 が戻されます。このため、上記の CREATE TABLE および INSERT ステートメントが複製されたあとに、mytable のスレーブコピーから選択すると、mycol に値 2009-09-01 18:00:00 が含まれると予想できます。しかし、これはそうなりません。mytable のスレーブコピーから選択すると、マスターとまったく同じ結果になります。

```
mysql> SELECT * FROM mytable;
+---------------------+
| mycol               |
+---------------------+
| 2009-09-01 12:00:00 |
+---------------------+
1 row in set (0.00 sec)
```

  SYSDATE() 関数は、NOW() とは異なり、レプリケーションに安全ではありません。バイナリログ内で SET TIMESTAMP ステートメントに影響されず、ステートメントベースロギングが使用される場合は非決定的であるためです。行ベースロギングを使用する場合は、これは問題ではありません。

  ほかの方法は --sysdate-is-now オプションを使用することで、SYSDATE() が NOW() のエイリアスになります。正しく機能するには、これをマスターとスレーブで行う必要があります。このような場合でも、この関数

によって警告が発行されますが、--sysdate-is-now がマスターとスレーブの両方で使用されるかぎり安全に無視できます。

MySQL 5.5.1 以降では、SYSDATE() は MIXED モード使用時に行ベースレプリケーションを使用して自動的に複製され、STATEMENT モードで警告を生成します。(Bug #47995)

セクション17.4.1.30「レプリケーションとタイムゾーン」も参照してください。

- 次の制限は、ステートメントベースレプリケーションにのみ適用され、行ベースレプリケーションには適用されません。ユーザーレベルロックを扱う GET_LOCK()、RELEASE_LOCK()、IS_FREE_LOCK()、および IS_USED_LOCK() 関数は、スレーブがマスター上の並列コンテキストを知ることなく複製されます。したがって、スレーブ上の内容が違ってしまうため、これらの関数を使用してマスターテーブルに挿入しないでください。たとえば、INSERT INTO mytable VALUES(GET_LOCK(...)) などのステートメントを発行しないでください。

  MySQL 5.5.1 以降では、これらの関数は MIXED モード使用時に行ベースレプリケーションを使用して自動的に複製され、STATEMENT モードで警告を生成します。(Bug #47995)

ステートメントベースレプリケーションが有効のときに前述の制限に対する回避策として、問題のある関数結果をユーザー変数に保存して、後続のステートメントでその変数を参照する方法を使用できます。たとえば、次の単一行 INSERT は、UUID() 関数を参照するため問題があります。

```
INSERT INTO t VALUES(UUID());
```

この問題を回避するには、代わりにこれを実行してください。

```
SET @my_uuid = UUID();
INSERT INTO t VALUES(@my_uuid);
```

このステートメントの連続は複製されます。@my_uuid の値が INSERT ステートメントの前にユーザー変数イベントとしてバイナリログに格納されて INSERT で使用できるためです。

同じ概念が複数行挿入に適用されますが、使用するのが面倒です。2 行挿入の場合、このようにできます。

```
SET @my_uuid1 = UUID(); @my_uuid2 = UUID();
INSERT INTO t VALUES(@my_uuid1),(@my_uuid2);
```

ただし、行数が多いか不明の場合、この回避策は困難であるか実用的でありません。たとえば、次のステートメントを個々のユーザー変数が各行に関連付けられているものに変換することはできません。

```
INSERT INTO t2 SELECT UUID(), * FROM t1;
```

ストアドファンクション内で、RAND() は、関数の実行中に 1 回だけ呼び出されるかぎり、正しく複製されます。(関数実行タイムスタンプと乱数シードをマスターとスレーブで同じ暗黙的入力と見なすことができます。)

FOUND_ROWS() と ROW_COUNT() 関数がステートメントベースレプリケーションを使用して複製されるときは、信頼性がありません。回避策は、関数呼び出しの結果をユーザー変数に格納してから、INSERT ステートメントでこれを使用することです。たとえば、mytable という名前のテーブルに結果を格納する場合は、普通は次のように実行するかもしれません。

```
SELECT SQL_CALC_FOUND_ROWS FROM mytable LIMIT 1;
INSERT INTO mytable VALUES( FOUND_ROWS() );
```

しかし、mytable を複製する場合は、次のように SELECT ... INTO を使用してから変数をテーブルに格納することをお勧めします。

```
SELECT SQL_CALC_FOUND_ROWS INTO @found_rows FROM mytable LIMIT 1;
INSERT INTO mytable VALUES(@found_rows);
```

このようにすることで、ユーザー変数はコンテキストの一部として複製され、スレーブ上で正しく適用されます。

これらの関数は、MIXED モード使用時に行ベースレプリケーションを使用して自動的に複製され、STATEMENT モードで警告を生成します。(Bug #12092、Bug #30244)

MySQL 5.6.15 より前は、--replicate-ignore-db や --replicate-do-table などのフィルタリングオプションがスレーブで有効になっていた場合、LAST_INSERT_ID() の値は正しく複製されませんでした。(Bug #17234370、BUG# 69861)

17.4.1.16 レプリケーションと LIMIT

DELETE、UPDATE、および INSERT ... SELECT ステートメント内の LIMIT 句のステートメントベースレプリケーションは、影響を受ける行の順序が未定義のため、安全ではありません。(このようなステートメントは、ORDER BY 句も含んでいる場合にのみ、ステートメントベースレプリケーションで正しく複製できます。) このようなステートメントに遭遇したときは:

- STATEMENT モード使用時は、このステートメントがステートメントベースレプリケーションで安全でないという警告が発行されるようになりました。

  現在のところ、STATEMENT モード使用時は、LIMIT を含む DML ステートメントに対して警告が発行されます (それらが ORDER BY 句も含んでいても (それによって決定的にされても))。これは既知の問題です。(Bug #42851)

- MIXED モード使用時は、このステートメントは行ベースモードを使用して自動的に複製されるようになりました。

17.4.1.17 レプリケーションと LOAD DATA INFILE

LOAD DATA INFILE ステートメントは、MySQL 4.0 以前が動作するマスターから MySQL 5.5.0 以前が動作するスレーブに常に正しく複製されるとはかぎりませんでした。ステートメントベースレプリケーションを使用するときは、LOAD DATA INFILE ステートメント CONCURRENT オプションは複製されませんでした。この問題は MySQL 5.5.0 で解決されました。MySQL 5.1 以降で行ベースレプリケーションを使用するときは、この問題は CONCURRENT オプション処理に影響しません。(Bug #34628)

MySQL 5.6 以降は、LOAD DATA INFILE は安全でないと見なされています (セクション17.1.2.3「バイナリロギングでの安全および安全でないステートメントの判断」を参照してください)。ステートメントベースロギング形式を使用時に警告が発行され、混合形式ロギング使用時は行ベース形式を使用してログが記録されます。

17.4.1.18 レプリケーションと REPAIR TABLE

破損または損傷したテーブルで使用されるとき、REPAIR TABLE ステートメントはリカバリできない行を削除できます。ただし、このステートメントによって実行されたテーブルデータのそのような変更は複製されないため、マスターとスレーブが同期を失う可能性があります。このため、マスター上のテーブルが損傷し、REPAIR TABLE を使用してそれを修復する場合は、REPAIR TABLE を使用する前にまずレプリケーションを停止し (まだ実行中の場合)、その後テーブルのマスターおよびスレーブコピーを比較して相違を手動で訂正する準備を整えてから、レプリケーションを再開するようにしてください。

17.4.1.19 レプリケーションとマスターまたはスレーブシャットダウン

マスターサーバーをシャットダウンしてあとで再起動することは安全です。スレーブがマスターへの接続を失った場合、スレーブはすぐに再接続しようとし、それが失敗した場合は定期的に再試行します。デフォルトでは 60 秒ごとに再試行します。これは、CHANGE MASTER TO ステートメントで変更できます。スレーブはネットワーク接続停止にも対応できます。ただし、スレーブは、マスターからのデータを slave_net_timeout 秒間受け取らなかったあとにのみ、ネットワーク停止を認識します。停止時間が短い場合は、slave_net_timeout を減らすことをお勧めします。セクション5.1.4「サーバーシステム変数」を参照してください。

マスター側で予期しないシャットダウン (たとえば、クラッシュ) が発生すると、マスターバイナリログファイルがフラッシュされていないため、マスターバイナリログにスレーブが読み取った最新の位置より前の最終位置が記録される可能性があります。このため、マスターが復帰したときにスレーブを複製できないことがあります。マスター my.cnf ファイルで sync_binlog=1 を設定することは、マスターがそのバイナリログをフラッシュする頻度が上がることになるため、この問題を最小化するのに役立ちます。

スレーブをクリーンにシャットダウンすることは、中断した場所が追跡されるため安全です。ただし、スレーブが一時テーブルを開いていないことを考慮してください。セクション17.4.1.22「レプリケーションと一時テーブル」を参照してください。予期しないシャットダウンにより問題が発生する場合があります (特に、ディスクキャッシュがディスクにフラッシュされない状態で問題が発生した場合)。

- トランザクションについては、スレーブはコミットしてから relay-log.info を更新します。これら 2 つの操作の間にクラッシュが発生した場合、リレーログ処理はファイルが示す情報より先に進み、スレーブは再起動されたあとにリレーログ内の最後のトランザクションからイベントを再実行します。

- 同様な問題は、スレーブが relay-log.info を更新したけれども、その書き込みがディスクにフラッシュされる前にサーバーホストがクラッシュした場合にも発生する可能性があります。これが発生する可能性を最小化するには、スレーブ my.cnf ファイルに sync_relay_log_info=1 を設定してください。sync_relay_log_info のデフォルト値は 0 で、この場合はディスクへの書き込みは強制されません。サーバーはオペレーティングシステムに依存して随時ファイルをフラッシュします。

システムのこのような問題に対する耐障害性は、優れた無停電電源があると大幅に向上します。

17.4.1.20 レプリケーションと max_allowed_packet

max_allowed_packet は、MySQL サーバーとクライアント (レプリケーションスレーブを含みます) の間の単一メッセージのサイズに上限を設定します。大きなカラム値 (TEXT または BLOB カラムで見つかる場合があります) を複製していて、max_allowed_packet がマスター上で小さすぎる場合は、マスターがエラーで失敗し、スレーブは I/O スレッドをシャットダウンします。max_allowed_packet がスレーブ上で小さすぎる場合も、スレーブは I/O スレッドを停止します。

行ベースレプリケーションは現在、更新された行のすべてのカラムとカラム値をマスターからスレーブに送信します (更新で実際には変更されなかったカラムの値を含みます)。これは、行ベースレプリケーションを使用して大きなカラム値を複製するときに、複製されるテーブル内でもっとも大きい行を格納できるだけの大きさに max_allowed_packet を設定するように気をつける必要があります (更新だけを複製したり、比較的小さい値だけを挿入したりする場合でも)。

17.4.1.21 レプリケーションと MEMORY テーブル

マスターサーバーがシャットダウンして再起動すると、その MEMORY テーブルは空になります。マスターはこの影響をスレーブに複製するために、起動後に所定の MEMORY テーブルを最初に使用するときに、空にすべきテーブルの DELETE ステートメントをバイナリログに書き込むことで、そのテーブルを空にする必要があることをスレーブに通知するイベントのログを記録します。

スレーブサーバーがシャットダウンして再起動すると、その MEMORY テーブルは空になります。これによってスレーブはマスターとの同期を失うため、ほかの障害が発生したりスレーブが停止したりすることがあります。

- マスターから受け取る行形式更新および削除が「'memory_table' にレコードが見つかりません」で失敗する場合があります。
- INSERT INTO ... SELECT FROM memory_table などのステートメントがマスターおよびスレーブ上で異なる行セットを挿入することがあります。

MEMORY テーブルを複製するスレーブを再起動する安全な方法は、最初にマスター上の MEMORY テーブルからのすべての行をドロップまたは削除して、これらの変更がスレーブに複製されるまで待つことです。すると、スレーブを再起動しても安全です。

場合によっては、別の再起動方法を適用できることがあります。binlog_format=ROW のときは、スレーブを再起動する前に slave_exec_mode=IDEMPOTENT を設定すれば、スレーブの停止を回避できます。これによってスレーブは複製を継続できますが、MEMORY テーブルは依然としてマスター上のものと異なります。アプリケーションロジックが MEMORY テーブルの内容を安全に失うことができるようなものである場合 (たとえば、MEMORY テーブルがキャッシュに使用されている場合) は、これでかまいません。slave_exec_mode=IDEMPOTENT がすべてのテーブルにグローバルに適用されるため、MEMORY でないテーブルでほかのレプリケーションエラーを隠すことができます。

(上記の方法は MySQL Cluster には適用されません。slave_exec_mode は常に IDEMPOTENT で、変更できません。)

MEMORY テーブルのサイズは、max_heap_table_size システム変数の値によって制限され、これは複製されません (セクション17.4.1.34「レプリケーションと変数」を参照してください)。max_heap_table_size での変更は、変更後に ALTER TABLE ... ENGINE = MEMORY または TRUNCATE TABLE を使用して作成または更新された MEMORY テーブルに、またはサーバー再起動後にすべての MEMORY テーブルに反映されます。この変数の値をマスター上で増やしスレーブ上でそうしない場合は、マスター上のテーブルがスレーブ上のものより大きくなることができるため、挿入がマスター上で成功するけれどもスレーブ上で Table is full エラーで失敗します:これは既知の問題です (Bug #48666)。このような場合、マスター上と同様にスレーブ上で max_heap_table_size のグローバル値を設定してから、レプリケーションを再開する必要があります。また、新しい値がそれぞれで完全 (グローバル) に反映されることを保証するために、マスターとスレーブの両方の MySQL サーバーを再起動することをお勧めします。

MEMORY テーブルに関する詳細は、セクション15.3「MEMORY ストレージエンジン」を参照してください。

17.4.1.22 レプリケーションと一時テーブル

binlog_format=ROW のときは次の段落の説明は適用されません。なぜなら、その場合は一時テーブルは複製されないためです。これは、スレーブによる計画外シャットダウンの場合、スレーブ上の一時テーブルが失われることがないことを意味します。このセクションの残りは、ステートメントベースまたは混合形式レプリケーションを使用するときにのみ適用されます。ステートメントベース形式を使用して安全にログを記録できる一時テーブ

ルを使用するステートメントでは、binlog_format が STATEMENT または MIXED のときは常に、スレーブ上で複製された一時テーブルを失うことが問題となる可能性があります。行ベースレプリケーションと一時テーブルの詳細は、一時テーブルの行ベースロギングを参照してください。

一時テーブルを使用するときの安全なスレーブシャットダウン　スレーブサーバーを停止する場合 (スレーブスレッドだけではなく)、およびスレーブ上でまだ実行されていない更新で使用するために開いている一時テーブルを複製した場合を除いて、一時テーブルは複製されます。スレーブサーバーを停止した場合、これらの更新が必要とする一時テーブルは、スレーブが再起動されたときに使用できなくなっています。この問題を回避するために、一時テーブルが開いている間はスレーブをシャットダウンしないでください。代わりに、次の手順を使用してください。

1. STOP SLAVE SQL_THREAD ステートメントを発行します。
2. SHOW STATUS を使用して Slave_open_temp_tables 変数の値を確認します。
3. 値が 0 でない場合は、スレーブ SQL スレッドを START SLAVE SQL_THREAD で再起動してから、あとで手順を繰り返します。
4. 値が 0 のときは、mysqladmin shutdown コマンドを発行してスレーブを停止します。

一時テーブルとレプリケーションオプション　デフォルトでは、すべての一時テーブルが複製されます。これは、対応する --replicate-do-db、--replicate-do-table、または --replicate-wild-do-table オプションが有効かどうかに関係なく発生します。ただし、--replicate-ignore-table および --replicate-wild-ignore-table オプションは一時テーブルで受け付けられます。

ステートメントベースまたは混合形式レプリケーションを使用するときに推奨される操作は、複製したくない一時テーブルに名前を付けるためだけに使用するプリフィクスを指定してから、そのプリフィクスを照合するために --replicate-wild-ignore-table オプションを使用することです。たとえば、このようなすべてのテーブルに norep で始まる名前を付けてから (たとえば、norepmytable、norepyourtable など)、テーブルが複製されることを回避するために --replicate-wild-ignore-table=norep% を使用します。

17.4.1.23 mysql システムデータベースのレプリケーション

mysql データベース内のテーブルに行われたデータ変更ステートメントは、binlog_format の値に従って複製されます。この値が MIXED の場合、これらのステートメントは行ベース形式を使用して複製されます。ただし、通常はこの情報を間接的に更新するステートメント (GRANT、REVOKE、およびトリガー、ストアドルーチン、ビューを操作するステートメントなど) は、ステートメントベースレプリケーションを使用してスレーブに複製されます。

17.4.1.24 レプリケーションとクエリーオプティマイザ

データ変更が非決定的であるような方法でステートメントが書き込まれた場合、マスターおよびスレーブ上のデータが違ってくる可能性があります。つまり、クエリーオプティマイザの出番です。(一般的に、これはレプリケーション以外であっても良い行動ではありません。)非決定的なステートメントの例には、ORDER BY 句なしの LIMIT を使用する DELETE または UPDATE ステートメントが含まれます。これらの説明の詳細は、セクション17.4.1.16「レプリケーションと LIMIT」を参照してください。

17.4.1.25 レプリケーションと予約語

古いマスターから新しいスレーブに複製しようとするとき、およびスレーブで実行中の新しい MySQL バージョンで予約語である識別子をマスター上で使用するときに、問題が発生する可能性があります。たとえば、4.1 以降のスレーブに複製している 4.0 マスター上で名前が current_user のテーブルカラムを使用しているときです。CURRENT_USER は MySQL 4.1 以降で予約語であるためです。このような場合、レプリケーションはエラー 1064 You have an error in your SQL syntax... で失敗する可能性があります (予約語を使用して名前が付けられたデータベースまたはテーブル、または予約語を使用して名前が付けられたカラムを持つテーブルが、レプリケーションから除外されていても)。これは、各 SQL イベントが実行前にスレーブによって解析される必要があるため、スレーブはどのデータベースオブジェクトが影響されるかがわかるという事実によります。スレーブは、イベントが解析されたあとにのみ、--replicate-do-db、--replicate-do-table, --replicate-ignore-db、および --replicate-ignore-table で定義されたフィルタリングルールを適用できます。

スレーブが予約語と見なす、マスター上のデータベース、テーブル、またはカラム名の問題を回避するには、次のいずれかを行ってください。

- マスター上で 1 つまたは複数の ALTER TABLE ステートメントを使用して、これらの名前がスレーブ上で予約語と見なされるデータベースオブジェクトの名前を変更するか、古い名前を使用する SQL ステートメントを変更して代わりに新しい名前を使用してください。

- これらのデータベースオブジェクト名を使用する SQL ステートメントで、それらの名前をバッククォート文字 (`) で囲まれた識別子として書いてください。

MySQL バージョン別の予約語の一覧については、「MySQL Server Version Reference」の「Reserved Words」を参照してください。識別子を囲むルールについては、セクション9.2「スキーマオブジェクト名」を参照してください。

17.4.1.26 レプリケーション中のスレーブエラー

ステートメントがマスターとスレーブの両方で同じエラー (同じエラーコード) を返した場合、エラーログは記録されますが、レプリケーションは継続します。

ステートメントがマスターとスレーブで異なるエラーを返した場合、スレーブ SQL スレッドは終了し、スレーブはそのエラーログにメッセージを書き込み、データベース管理者がそのエラーについて何をするかを決めるまで待ちます。これには、ステートメントがマスターまたはスレーブ上で (両方ではなく) エラーを返した場合が含まれます。この問題に対処するには、手動でスレーブに接続して問題の原因を判断してください。これには、SHOW SLAVE STATUS が役立ちます。それから問題を解決して START SLAVE を実行してください。たとえば、スレーブを再起動する前に、存在しないテーブルの作成が必要な場合があります。

このエラーコード検証動作が望ましくない場合は、--slave-skip-errors オプションでエラーの一部またはすべてを隠す (無視する) ことができます。

MyISAM などの非トランザクションストレージエンジンの場合、テーブルを不完全に更新してエラーコードを返すだけのステートメントが存在する場合があります。これはたとえば、1 つの行がキー制約に違反する複数行挿入で、または長い更新ステートメントが一部の行を更新したあとに強制終了された場合に発生する可能性があります。これがマスター上で発生した場合は、スレーブはステートメントの実行を予期しますが、同じエラーコードで終了します。そうしない場合は、スレーブ SQL スレッドはすでに説明したように停止します。

マスターおよびスレーブ上で異なるストレージエンジンを使用するテーブル間で複製する場合、同じステートメントが、一方のバージョンのテーブルに実行し、もう一方で実行しないときに異なるエラーを返したり、一方のバージョンのテーブルでエラーが発生し、もう一方では発生しなかったりすることがあることに留意してください。たとえば、MyISAM は外部キー制約を無視するため、マスター上の InnoDB テーブルにアクセスする INSERT または UPDATE ステートメントで外部キー違反が発生することがあっても、スレーブ上の MyISAM バージョンの同じテーブルで実行された同じステートメントはこのようなエラーを返さずにレプリケーションが停止します。

17.4.1.27 サーバー側ヘルプテーブルのレプリケーション

サーバーは、HELP ステートメントの情報を格納するテーブルを mysql データベース内で保持します (セクション13.8.3「HELP 構文」を参照してください)。これらのテーブルは、セクション5.1.10「サーバー側のヘルプ」で説明したとおり、手動でロードできます。

ヘルプテーブル内容は MySQL リファレンスマニュアルから抽出されます。各 MySQL リリースシリーズに固有のマニュアルバージョンがあるため、ヘルプの内容も各シリーズに固有です。一般的には、サーバーバージョンに一致するバージョンのヘルプ内容をロードしてください。これはレプリケーションに密接な関係があります。たとえば、MySQL 5.5 マスターサーバーには MySQL 5.5 ヘルプ内容をロードしますが、5.6 ヘルプ内容がより適している MySQL 5.6 スレーブサーバーには必ずしもその内容を複製する必要はありません。

このセクションでは、サーバーがレプリケーションに参加するときに、ヘルプテーブル内容の更新をどのように管理するかについて説明します。サーバーバージョンがこのタスクの 1 つの要因です。もう 1 つは、ヘルプテーブル構造がマスターとスレーブで異なる場合があることです。

ヘルプ内容が fill_help_tables.sql という名前のファイルに格納されているとします。MySQL 配布では、このファイルは share または share/mysql ディレクトリの下にあり、最新版はいつでも https://dev.mysql.com/doc/index-other.htmlからダウンロードできます。

ヘルプテーブルをアップグレードするには、次の手順を使用します。ここで説明する mysql コマンドの接続パラメータは示されていません。どのような場合でも、mysql データベース内のテーブルを変更するための権限を持つ、root などのアカウントを使用してサーバーに接続してください。

1. mysql_upgrade をまずスレーブ上で実行してからマスター上で実行することで、サーバーをアップグレードします。スレーブを最初にアップグレードするというのが一般原則です。

2. ヘルプテーブル内容をマスターからそのスレーブに複製するかどうかを決めます。しない場合は、マスターと各スレーブに個別に内容をロードします。または、マスターおよびそのスレーブ上のヘルプテーブル構造の非互換を確認し、ある場合は解決してから、内容をマスターにロードしてそこからスレーブに複製します。

ヘルプテーブル内容をロードするためのこれらの 2 つの方法をこれから詳しく説明します。

スレーブへのレプリケーションなしでヘルプテーブル内容をロードする

ヘルプテーブル内容をレプリケーションなしでロードするには、サーバーバージョンに適した内容を含む fill_help_tables.sql ファイルを使用して、マスターおよび各スレーブで個々に次のコマンドを実行します (コマンドは 1 行に入力します)。

```
mysql --init-command="SET sql_log_bin=0"
  mysql < fill_help_tables.sql
```

レプリケーショントポロジ内でスレーブがほかのスレーブのマスターとしても動作している場合は、各サーバー (スレーブを含む) に --init-command オプションを使用します。SET ステートメントはバイナリロギングを抑止します。アップグレードする各サーバーでコマンドが実行されると、完了です。

スレーブへのレプリケーションありでヘルプテーブル内容をロードする

ヘルプテーブル内容を複製しない場合は、マスターとそのスレーブ間のヘルプテーブル非互換を確認します。help_category および help_topic テーブル内の url カラムは、最初は CHAR(128) でしたが、長い URL に対応するために新しい MySQL バージョンでは TEXT です。ヘルプテーブル構造を確認するには、次のステートメントを使用します。

```
SELECT TABLE_NAME, COLUMN_NAME, COLUMN_TYPE
FROM INFORMATION_SCHEMA.COLUMNS
WHERE TABLE_SCHEMA = 'mysql'
AND COLUMN_NAME = 'url';
```

古い構造のテーブルの場合、ステートメントは次の結果を返します。

```
+---------------+-------------+-------------+
| TABLE_NAME    | COLUMN_NAME | COLUMN_TYPE |
+---------------+-------------+-------------+
| help_category | url         | char(128)   |
| help_topic    | url         | char(128)   |
+---------------+-------------+-------------+
```

新しい構造のテーブルの場合、ステートメントは次の結果を返します。

```
+---------------+-------------+-------------+
| TABLE_NAME    | COLUMN_NAME | COLUMN_TYPE |
+---------------+-------------+-------------+
| help_category | url         | text        |
| help_topic    | url         | text        |
+---------------+-------------+-------------+
```

マスターとスレーブの両方が古い構造の場合、または両方が新しい構造の場合は、互換であるため、マスターで次のコマンドを実行することでヘルプテーブル内容を複製できます。

```
mysql mysql < fill_help_tables.sql
```

テーブル内容はマスターにロードされてから、スレーブに複製されます。

マスターとスレーブのヘルプテーブルが非互換の場合は (一方のサーバーが古い構造で、もう一方が新しい構造)、ヘルプテーブル内容を最終的に複製しないか、または内容を複製できるようにテーブル構造を互換にするかを選択できます。

- 最終的に内容を複製しないことを決めた場合、すでに説明したように、--init-command オプション付きの mysql を使用してマスターとスレーブを個々にアップグレードします。
- そうではなく、テーブル構造に互換することを決めた場合は、サーバー上の古い構造のテーブルをアップグレードします。マスターサーバーのテーブル構造が古いとします。次のステートメントを実行することで、テーブルを新しい構造に手動でアップグレードします (すでに新しい構造のスレーブに変更が複製されることを回避するために、ここではバイナリロギングが無効になっています)。

  ```
  SET sql_log_bin=0;
  ALTER TABLE mysql.help_category ALTER COLUMN url TEXT;
  ALTER TABLE mysql.help_topic ALTER COLUMN url TEXT;
  ```

  次に、マスターで次のコマンドを実行します。

  ```
  mysql mysql < fill_help_tables.sql
  ```

テーブル内容はマスターにロードされてから、スレーブに複製されます。

17.4.1.28 レプリケーションとサーバー SQL モード

マスターとスレーブで異なるサーバー SQL モード設定を使用すると、同じ INSERT ステートメントがマスターとスレーブで異なる方法で処理され、マスターとスレーブが同期を失う場合があります。最適な結果を得るために、マスターとスレーブとで常に同じサーバー SQL モードを使用してください。このアドバイスは、ステートメントベースまたは行ベースのどちらのレプリケーションを使用しているかにかかわらず、適用されます。

パーティション化されたテーブルを複製する場合、マスターとスレーブで異なる SQL モードを使用すると、問題が発生する可能性が高くなります。少なくとも、所定のテーブルのマスターおよびスレーブコピー間で、パーティションへのデータ配分が違ってくる可能性が高くなります。マスター上で成功するパーティション化されたテーブルへの挿入が、スレーブ上で失敗する可能性もあります。

詳細は、セクション5.1.7「サーバー SQL モード」を参照してください。

17.4.1.29 レプリケーション再試行とタイムアウト

グローバルシステム変数 slave_transaction_retries は、次のようにレプリケーションに影響します: InnoDB デッドロックが原因で、または InnoDB innodb_lock_wait_timeout 値あるいは NDB TransactionDeadlockDetectionTimeout または TransactionInactiveTimeout 値を越えたことが原因で、スレーブ SQL スレッドがトランザクションの実行に失敗した場合、スレーブはトランザクションを slave_transaction_retries 回自動的に再試行してから エラーで停止します。デフォルト値は 10 です。合計再試行回数は SHOW STATUS の出力で確認できます。セクション5.1.6「サーバーステータス変数」を参照してください。

17.4.1.30 レプリケーションとタイムゾーン

デフォルトでは、マスターおよびスレーブサーバーは、それらが同じタイムゾーンであることを想定します。タイムゾーンが異なるサーバー間で複製する場合、マスターとスレーブの両方でタイムゾーンを設定する必要があります。そうでない場合は、NOW() または FROM_UNIXTIME() 関数を使用するステートメントなど、マスター上のローカル時間に依存するステートメントが適切に複製されません。MySQL サーバーが動作するタイムゾーンは、mysqld_safe スクリプトの --timezone=timezone_name オプションを使用するか、TZ 環境変数を設定することで、設定します。セクション17.4.1.15「レプリケーションとシステム関数」も参照してください。

マスターが MySQL 4.1 以前の場合はさらに、マスターとスレーブの両方が同じデフォルト接続タイムゾーンを使用するべきです。つまり、--default-time-zone パラメータをマスターとスレーブの両方で同じ値にしてください。

CONVERT_TZ(...,...,@@SESSION.time_zone) は、マスターとスレーブの両方が MySQL 5.0.4 以降を実行している場合にのみ、正しく複製されます。

17.4.1.31 レプリケーションとトランザクション

同じトランザクション内にトランザクションおよび非トランザクションステートメントを混在させる　一般的に、レプリケーション環境でトランザクションおよび非トランザクションテーブルの両方を更新するトランザクションは避けるべきです。トランザクション (または一時) および非トランザクションテーブルの両方にアクセスしてそれらに書き込むステートメントを使用することも避けるべきです。

MySQL 5.5.2 以降では、サーバーはバイナリロギングに次のルールを使用します。

- トランザクション内の最初のステートメントが非トランザクションである場合、それらはすぐにバイナリログに書き込まれます。トランザクション内の残りのステートメントはキャッシュされ、トランザクションがコミットされるまでバイナリログに書き込まれません。(トランザクションがロールバックされた場合、キャッシュされたステートメントは、ロールバックできない非トランザクション変更を行う場合にのみバイナリログに書き込まれます。それ以外の場合は、それらは破棄されます。)

- ステートメントベースロギングの場合、非トランザクションステートメントのロギングは binlog_direct_non_transactional_updates システム変数によって影響されます。この変数が OFFの場合 (デフォルト)、ロギングは上記のとおりになります。この変数が ON の場合、非トランザクションステートメントがトランザクション内のどこで発生しても (最初の非トランザクションステートメントだけではありません)、ただちにロギングが発生します。ほかのステートメントはトランザクションキャッシュに保持され、トランザクションがコミットしたときにログが記録されます。binlog_direct_non_transactional_updates は行形式または混合形式バイナリロギングに影響しません。

トランザクション、非トランザクション、および混合ステートメント

これらのルールを適用する場合、サーバーは、非トランザクションテーブルだけを変更する場合にはステートメントを非トランザクションと見なし、トランザクションテーブルだけを変更する場合にはトランザクションと見なします。MySQL 5.6 では、非トランザクションおよびトランザクションテーブルの両方を参照し、使用されるすべてのテーブルを更新するステートメントは、「混合」 ステートメントと見なされます。(以前の MySQL リリースシリーズでは、非トランザクションおよびトランザクションテーブルの両方を変更するステートメントが、混合と見なされていました。)混合ステートメントは、トランザクションステートメントと同様に、キャッシュされ、トランザクションがコミットするときにログが記録されます。

トランザクションテーブルを更新する混合ステートメントは、次のアクションのいずれかを実行する場合も、安全ではないと見なされます。

- トランザクションテーブルを更新または読み取る
- 非トランザクションテーブルを読み取り、トランザクション分離レベルが REPEATABLE_READ 未満

トランザクション内でトランザクションテーブル更新に続く混合ステートメントは、次のアクションのいずれかを実行する場合、安全ではないと見なされます。

- テーブルを更新し、一時テーブルから読み取る
- 非トランザクションテーブルを更新し、binlog_direct_non_trans_update が OFF

詳細については、セクション17.1.2.3「バイナリロギングでの安全および安全でないステートメントの判断」を参照してください。

> **注記**
>
> 混合ステートメントは混合バイナリロギング形式とは関係ありません。

トランザクション内でトランザクションおよび非トランザクションテーブルへの更新が混在している状況で、バイナリログ内のステートメントの順序は正しく、必要なすべてのステートメントがバイナリログに書き込まれます (ROLLBACK の場合でも)。ただし、最初の接続トランザクションが完了する前に 2 番目の接続が非トランザクションテーブルを更新するときは、ステートメントログが記録される順序が乱れる場合があります。最初の接続によって実行されているトランザクションの状態にかかわらず、2 番目の接続更新が実行された直後に書き込まれるためです。

マスターとスレーブで異なるストレージエンジンを使用する　スレーブ上の非トランザクションテーブルを使用してマスター上のトランザクションテーブルを複製できます。たとえば、InnoDB マスターテーブルを MyISAM スレーブテーブルとして複製できます。ただし、これを行う場合、スレーブが BEGIN ブロックの始めで再起動したために BEGIN ... COMMIT ブロックの途中でスレーブが停止した場合に問題が発生します。

MySQL 5.6 では、マスター上の MyISAM テーブルからスレーブ上のトランザクションテーブル (InnoDB ストレージエンジンを使用するテーブルなど) にトランザクションを複製することも安全です。このような場合 (MySQL 5.5.0 以降)、マスター上で発行された AUTOCOMMIT=1 ステートメントが複製されて、スレーブ上で AUTOCOMMIT モードが適用されます。

スレーブのストレージエンジンタイプが非トランザクションの場合、トランザクションおよび非トランザクションテーブルへの更新が混在するマスター上のトランザクションは、マスタートランザクションテーブルとスレーブ非トランザクションテーブルの間でデータの不整合が発生する可能性があるため、避けるべきです。つまり、このようなトランザクションはレプリケーションの同期を失って、マスターストレージエンジン独自の動作になる可能性があります。現在のところ、MySQL はこれについて警告を発行しないため、トランザクションテーブルをマスターからスレーブ上の非トランザクションテーブルに複製するときは十分に気を付けるようにしてください。

トランザクション内でバイナリロギング形式を変更する　MySQL 5.5.3 以降では、トランザクションが進行中のときは binlog_format システム変数は読み取り専用です。(Bug #47863)

各トランザクション (autocommit トランザクションを含む) は、BEGIN ステートメントで始まり、COMMIT または ROLLBACK ステートメントで終わるかのように、バイナリログに記録されます。MySQL 5.6 では、この真実は、非トランザクションストレージエンジン (MyISAM など) を使用するテーブルに影響するステートメントにも適用されます。

17.4.1.32 レプリケーションとトリガー

ステートメントベースレプリケーションでは、マスター上で実行されたトリガーはスレーブでも実行されます。行ベースレプリケーションでは、マスター上で実行されたトリガーはスレーブ上で実行されません。代わりに、トリガー実行によって発生するマスター上の行変更は複製され、スレーブ上で適用されます。

この動作は設計によります。行ベースレプリケーション時に、スレーブがトリガーおよびそれらによって発生する行変更を適用した場合、変更は実際にはスレーブ上で 2 回適用されるため、マスターとスレーブでデータが違ってきます。

マスターとスレーブの両方でトリガーを実行したい場合 (おそらく、マスターとスレーブでトリガーが違うため)、ステートメントベースレプリケーションを使用する必要があります。ただし、スレーブ側トリガーを有効にするために、ステートメントベースレプリケーションを排他的に使用する必要はありません。この効果が必要なステートメントでのみステートメントベースレプリケーションに切り替え、残りの時間は行ベースレプリケーションを使用することで十分です。

AUTO_INCREMENT カラムを更新するトリガー (または関数) を呼び出すステートメントは、ステートメントベースレプリケーションを使用して正しく複製されません。MySQL 5.6 はこのようなステートメントを安全でないとマークします。(Bug #45677)

17.4.1.33 レプリケーションと TRUNCATE TABLE

TRUNCATE TABLE は通常は DML ステートメントと見なされるため、バイナリロギングモードが ROW または MIXED のときは行ベース形式を使用してログが記録されて複製されることが予想されます。しかしこのことが、InnoDB などのトランザクションストレージエンジン (トランザクション分離レベルが READ COMMITTED または READ UNCOMMITTED (ステートメントベースロギングを排除)) を使用するテーブルを STATEMENT または MIXED モードでログを記録または複製するときに、問題を発生させました。

TRUNCATE TABLE は、ロギングおよびレプリケーション目的のときはステートメントとしてログを記録し複製できるように、DML ではなく DDL として扱われます。ただし、レプリケーションスレーブ上の InnoDB およびほかのトランザクションテーブルに適用されるこのステートメントの効果は、このようなテーブルを制御するルール (セクション13.1.33「TRUNCATE TABLE 構文」 を参照) に依然として従います。(Bug #36763)

17.4.1.34 レプリケーションと変数

システム変数は、次の変数を除き (セッションスコープで使用されるとき)、STATEMENT モード使用時は正しく複製されません。

- auto_increment_increment
- auto_increment_offset
- character_set_client
- character_set_connection
- character_set_database
- character_set_server
- collation_connection
- collation_database
- collation_server
- foreign_key_checks
- identity
- last_insert_id
- lc_time_names
- pseudo_thread_id
- sql_auto_is_null
- time_zone
- timestamp
- unique_checks

MIXED モード使用時に、前述のリスト内の変数がセッションスコープで使用されるときはステートメントベースから行ベースロギングに切り替わります。セクション5.2.4.3「混合形式のバイナリロギング形式」を参照してください。

NO_DIR_IN_CREATE モードの場合を除いて、sql_mode も複製されます。スレーブは常に、NO_DIR_IN_CREATE にその独自の値を保持します (マスター上での変更にかかわらず)。これは、すべてのレプリケーション形式に当てはまります。

ただし、mysqlbinlog が SET @@sql_mode = mode ステートメントを解析したときに、NO_DIR_IN_CREATEを含む完全な mode 値が受信サーバーに渡されます。このため、このようなステートメントのレプリケーションは、STATEMENT モード使用時は安全でない場合があります。

default_storage_engine および storage_engine のシステム変数は、ロギングモードにかかわらず複製されません。これは、異なるストレージエンジン間のレプリケーションを容易にすることを意図しています。

read_only システム変数は複製されません。さらに、この変数を有効にした場合の一時テーブル、テーブルロック、および SET PASSWORD ステートメントに関する効果は、MySQL バージョンごとに異なります。

max_heap_table_size システム変数は複製されません。スレーブ上でこの変数の値を増やさずにマスター上で増やすと、最終的にスレーブ上でTable is fullエラーになる可能性があります (スレーブの対応するものより大きくなることが許可されたマスター上のMEMORY テーブルで INSERT ステートメントを実行しようとするとき)。詳細については、セクション17.4.1.21「レプリケーションと MEMORY テーブル」を参照してください。

ステートメントベースレプリケーションで、セッション変数は、テーブルを更新するステートメントで使用されるときに正しく複製されません。たとえば、次のシーケンスのステートメントはマスターとスレーブで同じデータを挿入しません。

```
SET max_join_size=1000;
INSERT INTO mytable VALUES(@@max_join_size);
```

これは、一般的なシーケンスには適用されません。

```
SET time_zone=...;
INSERT INTO mytable VALUES(CONVERT_TZ(..., ..., @@time_zone));
```

セッション変数のレプリケーションは、行ベースレプリケーションが使用されているときは問題ではありません。このケースでは、セッション変数は常に安全に複製されます。セクション17.1.2「レプリケーション形式」を参照してください。

MySQL 5.6 では、次のセッション変数はロギング形式にかかわらず、バイナリログに書き込まれ、バイナリログを解析するときにレプリケーションスレーブによって受け付けられます。

- sql_mode
- foreign_key_checks
- unique_checks
- character_set_client
- collation_connection
- collation_database
- collation_server
- sql_auto_is_null

> **重要**
>
> 文字セットと照合順序に関係するセッション変数はバイナリログに書き込まれるけれども、異なる文字セット間のレプリケーションはサポートされません。

混乱の可能性を減らすため、マスターとスレーブの両方で lower_case_table_names システム変数に常に同じ設定を使用することをお勧めします (特に、ファイルシステムが大文字小文字を区別するプラットフォームで MySQL を実行しているとき)。

> **注記**
>
> 以前のバージョンの MySQL で、大文字小文字を区別するファイルシステムが使用されているとき、この変数をスレーブで 1 に、マスターで異なる値に設定すると、レプリケーションが失敗することがありました。この問題は MySQL 5.6.1 で解決されています。(Bug #37656)

17.4.1.35 レプリケーションとビュー

ビューは常にスレーブに複製されます。ビューは、それらが参照するテーブルではなく、それら独自の名前でフィルタリングされます。これは、通常は replication-ignore-table ルールで除外されるテーブルがビューに含まれている場合でも、そのビューをスレーブに複製できることを意味します。このため、通常はセキュリティー上の理由でフィルタリングされるテーブルデータをビューが複製しないように、気を付けるようにしてください。

テーブルから同じ名前のビューへのレプリケーションは、ステートメントベースロギング使用時はサポートされますが、行ベースロギング使用時はされません。MySQL 5.6.11 以降で、行ベースロギングが有効のときにそうしようとすると、エラーになります。(Bug #11752707、Bug #43975)

17.4.2 MySQL バージョン間のレプリケーション互換性

MySQL は、あるメジャーバージョンから次以降のメジャーバージョンへのレプリケーションをサポートします。たとえば、MySQL 4.1 を実行するマスターから MySQL 5.0 を実行するスレーブ、MySQL 5.0 を実行するマスターから MySQL 5.1 を実行するスレーブなどへの複製ができます。

ただし、古いマスターから新しいスレーブに複製するときに、マスターがスレーブ上で使用されているバージョンの MySQL でもうサポートされていないステートメントを使用したり、そのような動作に依存したりする場合は、問題が発生することがあります。たとえば、MySQL 5.5 では、CREATE TABLE ... SELECT ステートメントが、作成済みでないテーブルを変更することが許可されますが、MySQL 5.6 では許可されなくなりました (セクション17.4.1.5「CREATE TABLE ... SELECT ステートメントのレプリケーション」を参照してください)。

2 つを超える MySQL Server バージョンを使用することは、マスターまたはスレーブ MySQL サーバーの数にかかわらず、複数のマスターを使用するレプリケーションセットアップでサポートされません。この制限は、メジャーバージョンだけではなく、同じメジャーバージョン内のマイナーバージョンにも適用されます。たとえば、チェーンまたは循環レプリケーションセットアップを使用している場合、MySQL 5.6.1、MySQL 5.6.2、および MySQL 5.6.4 は同時に使用できませんが、これらのリリースの任意の 2 つを一緒に使用することはできます。

場合によっては、マスターと、マスターよりメジャーバージョン 2 つ以上新しいスレーブとの間で複製できます。ただし、MySQL 4.1 以前を実行するマスターから MySQL 5.1 以降を実行するスレーブに複製しようとすることに関する既知の問題があります。このような問題に対処するために、2 つの間に中間バージョンを実行する MySQL サーバーを挿入できます。たとえば、MySQL 4.1 マスターから MySQL 5.1 スレーブに直接複製するのではなく、MySQL 4.1 サーバーから MySQL 5.0 サーバーに複製してから、MySQL 5.0 サーバーから MySQL 5.1 サーバーに複製できます。

> **重要**
>
> レプリケーション (およびほかの) 機能は継続的に改善されているので、その MySQL メジャーバージョン内で使用可能な最新リリースを使用することを強くお勧めします。また、MySQL メジャーバージョンの早期リリースを使用するマスターとスレーブは GA (本番環境) リリースに更新することをお勧めします (そのメジャーバージョンで後者を入手できるようになったとき)。

新しいマスターから古いスレーブに複製できる場合がありますが、一般的にはサポートされません。これはいくつかの要因によります。

- バイナリログ形式の変更　バイナリログ形式はメジャーリリース間で変わることがあります。下位互換性を維持しようと試みてはいますが、これがいつか可能なわけではありません。たとえば、MySQL 5.0 に実装されたバイナリログ形式は、以前のバージョンで使用されたものから大幅に変わりました (特に、文字セット、LOAD DATA INFILE、およびタイムゾーンの扱いに関して)。これは、MySQL 5.0 (以降) マスターから MySQL 4.1 (以前) スレーブへのレプリケーションは一般的にサポートされないことを意味します。

  これは、レプリケーションサーバーのアップグレードにも密接な関係があります。詳細は、セクション17.4.3「レプリケーションセットアップをアップグレードする」を参照してください。

- 行ベースレプリケーションの使用　行ベースレプリケーションが MySQL 5.1.5 で実装されたため、行ベースレプリケーションを使用して MySQL 5.6 以降のマスターから MySQL 5.1.5 より古いスレーブに複製することはできません。

  行ベースレプリケーションの詳細は、セクション17.1.2「レプリケーション形式」を参照してください。

- SQL 非互換　複製されるステートメントが、マスター上で利用できるけれどもスレーブ上で利用できない SQL 機能を使用する場合は、ステートメントベースレプリケーションを使用して新しいマスターから古いスレーブに複製することはできません。

  ただし、マスターとスレーブの両方が行ベースレプリケーションをサポートし、複製するデータ定義ステートメントにマスター上で見つかるけれどもスレーブ上で見つからない SQL 機能に依存するものがない場合は、行

ベースレプリケーションを使用してデータ変更ステートメントの効果を複製できます (マスター上で実行される DDL がスレーブ上でサポートされない場合でも)。

- MySQL 5.6 の重要な変数　以前の MySQL バージョンに複製するときに無効にする必要がある機能が、MySQL 5.6 に追加されています。非互換を回避するには、MySQL 5.6 マスターで次の変数を設定してください。
  - binlog_checksum=NONE
  - binlog_row_image=FULL
  - binlog_rows_query_log_events=OFF
  - log_bin_use_v1_row_events=1
  - gtid_mode=OFF

潜在的なレプリケーション問題の詳細は、セクション17.4.1「レプリケーションの機能と問題」を参照してください。

17.4.3 レプリケーションセットアップをアップグレードする

レプリケーションセットアップに関与するサーバーをアップグレードするときに、アップグレードの手順は現在のサーバーバージョンとアップグレード後のバージョンによって異なります。

このセクションは、古いバージョンの MySQL から MySQL 5.6 にレプリケーションをアップグレードすることに割り当てられます。4.0 サーバーは 4.0.3 以降であるべきです。マスター上のレプリケーションユーザーには、ハッシュ方式の変更により、MySQL バージョン 4.1.1 以降で生成されたパスワードハッシュが必要です。詳細は、セクション6.1.2.4「MySQL でのパスワードハッシュ」を参照してください。

以前の MySQL リリースシリーズから 5.6 にマスターをアップグレードするときは、最初にこのマスターのすべてのスレーブが同じ 5.6.x リリースを使用していることを確認してください。そうでない場合は、まずスレーブをアップグレードしてください。各スレーブをアップグレードするには、シャットダウンし、適切な 5.6.x バージョンにアップグレードし、再起動してから、レプリケーションを再開してください。アップグレード後にスレーブによって作成されるリレーログは 5.6 形式です。

厳密な SQL モードでの操作に影響する変更があると、更新後のスレーブでレプリケーションが失敗する場合があります。たとえば、MySQL 5.6.13 以降では、サーバーは、厳密モード (STRICT_TRANS_TABLES または STRICT_ALL_TABLES) で時間データ型の DEFAULT 値への 0 の挿入を制限します。ステートメントベースロギング (binlog_format=STATEMENT) を使用する場合にレプリケーションで非互換になるのは、スレーブがアップグレードされた場合、アップグレードされていないマスターが エラーなしで実行するステートメントが、スレーブ上で失敗することがあり、するとレプリケーションが停止します。これに対処するには、マスター上のすべての新しいステートメントを停止し、スレーブが追い付くのを待ちます。それからスレーブをアップグレードします。あるいは、新しいステートメントを停止できない場合は、マスターで一時的に行ベースのロギングに変更し (binlog_format=ROW)、すべてのスレーブが、この変更の時点までに生成されたすべてのバイナリログを処理するまで待ちます。それからスレーブをアップグレードします。

スレーブがアップグレードされたあとに、マスターをシャットダウンし、スレーブと同じ 5.6.x リリースにアップグレードしてから、再起動します。一時的にマスターを行ベースロギングに変更した場合は、ステートメントベースロギングに戻します。5.6 マスターは、アップグレード前に書き込まれた古いバイナリログを読み取って、それらを 5.6 スレーブに送信できます。スレーブは、古い形式を認識し、適切に処理します。アップグレード後にマスターによって作成されるバイナリログは 5.6 形式です。これらも 5.6 スレーブによって認識されます。

つまり、MySQL 5.6 にアップグレードするときは、マスターを 5.6 にアップグレードする前に、スレーブが MySQL 5.6 である必要があります。5.6 から古いバージョンへのダウングレードは、それほど簡単には機能しません。ダウングレードに進む前に 5.6 バイナリログまたはリレーログを削除できるように、それらが完全に処理されたことを確認する必要があります。

レプリケーションセットアップを以前のバージョンにダウングレードすることは、ステートメントベースから行ベースレプリケーションに切り替えたあと、および最初の行ベースステートメントが binlog に書き込まれたあとには実行できません。セクション17.1.2「レプリケーション形式」を参照してください。

アップグレードによっては、ある MySQL シリーズから次のものに移行するときに、データベースオブジェクトの削除と再作成が必要になる場合があります。たとえば、照合順序の変更には、テーブルインデックスの再構築が必要な場合があります。このような操作が必要な場合は、詳細をセクション2.11.1.3「MySQL 5.5 から 5.6 へのアップグレード」で参照してください。これらの操作をスレーブとマスターで個別に実行し、これらの操作を

マスターからスレーブに複製することを無効にするのがもっとも安全です。 これを実現するには、次の手順を使用してください。

1. すべてのスレーブを停止してそれらをアップグレードします。マスターに接続しないように、--skip-slave-start オプションでそれらを再起動します。データベースオブジェクトの再作成に必要なテーブル修復または再構築操作 (REPAIR TABLE、ALTER TABLE の使用など)、またはテーブルまたはトリガーのダンプおよびリロードを実行します。

2. マスター上でバイナリログを無効にします。マスターを再起動しないでこれを行うには、SET sql_log_bin = 0 ステートメントを実行します。または、マスターを停止して、--log-bin オプションなしで再起動します。マスターを再起動する場合、クライアント接続を拒否することもお勧めします。たとえば、すべてのクライアントが TCP/IP を使用して接続する場合は、マスターを再起動するときに --skip-networking オプションを使用します。

3. バイナリログが無効の状態で、データベースオブジェクトの再作成に必要なテーブル修復または再構築操作を実行します。これらの操作がログに記録されてあとでスレーブに送信されることを回避するには、この手順の間バイナリログを無効にする必要があります。

4. マスター上でバイナリログを再度有効にします。以前に sql_log_bin を 0 に設定した場合、SET sql_log_bin = 1 ステートメントを実行します。バイナリログを無効にするためにマスターを再起動した場合は、クライアントとスレーブが接続できるように --log-bin あり、--skip-networking なしで再起動します。

5. 今度は --skip-slave-start オプションなしで、スレーブを再起動します。

グローバルトランザクション識別子を持つレプリケーションが MySQL 5.6.7 で導入されました。GTID をサポートしないバージョンの MySQL から、サポートするバージョンに既存のレプリケーションセットアップをアップグレードする場合、GTID ベースレプリケーションのすべての要件をセットアップが満たすことを確認する前に、マスターまたはスレーブで GTID を有効にしないでください。セクション17.1.3.2「GTID を使用したレプリケーションのセットアップ」を参照してください。GTID ベースレプリケーションを使用するために既存のレプリケーションセットアップを変換することに関する情報が含まれています。

17.4.4 レプリケーションのトラブルシューティング

指示に従ってもレプリケーションセットアップが機能しない場合、最初に行うことはエラーログでメッセージを確認することです。多くのユーザーは、問題が発生したあとにこれを十分に実行せずに、時間を失います。

エラーログから何が問題だったのかがわからない場合は、次の手法を試してください。

- マスターでバイナリロギングが有効になっていることを、SHOW MASTER STATUS ステートメントを発行して確認します。ロギングが有効であると、Position はゼロではありません。バイナリロギングが有効でない場合、--log-bin オプションでマスターを実行していることを確認してください。

- マスターとスレーブの両方が --server-id オプションで起動されたこと、および ID 値が各サーバーで一意であることを確認します。

- スレーブが動作していることを確認します。SHOW SLAVE STATUS を使用して、Slave_IO_Running および Slave_SQL_Runningの値が両方とも Yes であるかどうかを確認してください。そうでない場合、スレーブサーバーを起動するときに使用したオプションを確認してください。たとえば、--skip-slave-start は、START SLAVE ステートメントを発行するまでスレーブスレッドが起動するのを妨げます。

- スレーブが動作している場合、マスターとの接続が確立されたかどうかを確認します。SHOW PROCESSLIST を使用して I/O スレッドと SQL スレッドを見つけ、それらの State カラムをチェックしてそれらに何が表示されているかを確認してください。セクション17.2.1「レプリケーション実装の詳細」を参照してください。I/O スレッド状態が Connecting to master である場合、次のことを確認してください。

  - マスター上でレプリケーションに使用されているユーザーの権限を確認します。

  - マスターのホスト名前が正しいこと、正しいポートを使用してマスターに接続していることを確認します。レプリケーションに使用されるポートは、クライアントネットワーク通信に使用されるポートと同じです (デフォルトは 3306 です)。ホスト名の場合、その名前が正しい IP アドレスに解決されることを確認してください。

  - ネットワークがマスターまたはスレーブ上で無効化されていないことを確認します。構成ファイルで skip-networking オプションを探してください。存在する場合、コメントアウトするか、削除してください。

  - マスターにファイアウォールまたは IP フィルタリング構成がある場合、MySQL に使用されるネットワークポートがフィルタリングされていないことを確認します。

  - ping または traceroute/tracert を使用してホストに到達することで、マスターに到達できることを確認します。

- スレーブが以前は動作していたのに停止した場合、通常はマスターで成功したステートメントの一部がスレーブで失敗したことが原因です。マスターの適切なスナップショットを作成し、スレーブスレッド以外にスレーブでそのデータを変更しなかった場合、これは決して発生しないはずです。スレーブが突然停止する場合、それはバグであるか、またはセクション17.4.1「レプリケーションの機能と問題」で説明した既知のレプリケーション制限のいずれかが発生しています。バグの場合は、報告方法の説明をセクション17.4.5「レプリケーションバグまたは問題を報告する方法」で参照してください。

- マスター上で成功したステートメントがスレーブ上で実行することを拒否する場合に、スレーブのデータベースを削除してマスターから新しいスナップショットをコピーすることによる完全なデータベース再同期を実行できない場合は、次の手順を試みてください。

  1. スレーブ上の影響されるテーブルがマスターテーブルと異なるかどうかを判断します。これがどのように発生したかを理解しようとしてください。それから、スレーブのテーブルをマスターのものと同じにして START SLAVE を実行してください。

  2. 前述の手順が機能しない、または当てはまらない場合は、手動で更新を行ってから (必要な場合)、マスターからの次のステートメントを無視することが安全かどうかを理解しようとしてください。

  3. スレーブがマスターからの次のステートメントをスキップできると判断した場合、次のステートメントを発行します。

     ```
     mysql> SET GLOBAL sql_slave_skip_counter = N;
     mysql> START SLAVE;
     ```

     マスターからの次のステートメントが AUTO_INCREMENT または LAST_INSERT_ID() を使用しない場合、N の値は 1 であるべきです。そうでない場合は、値は 2 であるべきです。AUTO_INCREMENT または LAST_INSERT_ID() を使用するステートメントに値 2 を使用する理由は、それらがマスターのバイナリログ内で 2 つのイベントを必要とすることです。

     セクション13.4.2.4「SET GLOBAL sql_slave_skip_counter 構文」も参照してください。

  4. スレーブがマスターと完全に同期された状態で起動したこと、およびスレーブスレッド以外で使用されるテーブルをだれも更新しなかったことがわかっている場合は、おそらくこの矛盾はバグの結果です。最新バージョンの MySQL を実行している場合は、この問題を報告してください。古いバージョンを実行している場合は、最新の本番環境リリースにアップグレードして問題が持続するかどうかを判断してみてください。

17.4.5 レプリケーションバグまたは問題を報告する方法

関係するユーザーエラーはないと判断したけれども、依然としてレプリケーションがまったく機能しないか安定しない場合は、バグレポートを送る時期です。バグを突き止めるため、できるだけ多くの情報をユーザーから入手する必要があります。優れたバグレポートを準備するために、ある程度の時間と労力を費やしてくださるようにお願いします。

そのバグをはっきりと示す再現可能なテストケースがある場合は、セクション1.6「質問またはバグをレポートする方法」で示す手順を使用してオラクルのバグデータベースに入力してくださるようにお願いします。「幽霊のような」問題 (意のままに再現できないもの) の場合は、次の手順を使用してください。

1. ユーザーエラーが関係していないことを確認します。たとえば、スレーブスレッド以外の場所でスレーブを更新すると、データが同期を失い、更新時に一意キー違反が発生することがあります。この場合、スレーブスレッドは停止し、ユーザーが手動でテーブルをクリーンアップして同期された状態に戻すまで待ちます。 これは、レプリケーションの問題ではありません。外部干渉の問題でレプリケーションが失敗しています。

2. --log-slave-updates および --log-bin オプションでスレーブを実行します。これらのオプションにより、スレーブはマスターから受け取る更新のログをその独自のバイナリログに記録します。

3. レプリケーション状態をリセットする前のすべての証拠を保存します。情報がなかったり、不完全な情報しかなかったりすると、オラクルでは問題を突き止めることが困難または不可能になります。収集すべき証拠:

   - マスターからのすべてのバイナリログファイル
   - スレーブからのすべてのバイナリログファイル
   - 問題を発見した時点の、マスターからの SHOW MASTER STATUS の出力

- 問題を発見した時点の、スレーブからの SHOW SLAVE STATUS の出力
- マスターおよびスレーブからのエラーログ

4. mysqlbinlog を使用してバイナリログを調べます。次のことは、問題のステートメントを見つけるのに役立つはずです。log_file および log_pos は、SHOW SLAVE STATUS からの Master_Log_File および Read_Master_Log_Pos 値です。

```
shell> mysqlbinlog --start-position=log_pos log_file | head
```

問題の証拠を収集したあとは、まずそれらを個別のテストケースとして切り分けてみてください。そのうえで、セクション1.6「質問またはバグをレポートする方法」での指示を使用して問題およびできるだけ多くの情報をオラクルのバグデータベースに入力してください。

第 18 章 MySQL Cluster NDB 7.3 および MySQL Cluster NDB 7.4

目次

18.1 MySQL Cluster の概要 2010
18.1.1 MySQL Cluster の主な概念 2011
18.1.2 MySQL Cluster のノード、ノードグループ、レプリカ、およびパーティション 2014
18.1.3 MySQL Cluster のハードウェア、ソフトウェア、およびネットワーク要件 2016
18.1.4 MySQL Cluster の開発履歴 2018
18.1.5 InnoDB を使用した MySQL Server と MySQL Cluster の比較 2020
18.1.6 MySQL Cluster の既知の制限 2023
18.2 MySQL Cluster のインストール 2031
18.2.1 MySQL Cluster Auto-Installer 2033
18.2.2 Linux での MySQL Cluster のインストール 2047
18.2.3 Windows での MySQL Cluster のインストール 2052
18.2.4 MySQL Cluster の初期構成 2059
18.2.5 MySQL Cluster の初期起動 2061
18.2.6 テーブルとデータを含む MySQL Cluster の例 2062
18.2.7 MySQL Cluster の安全なシャットダウンと再起動 2064
18.2.8 MySQL Cluster NDB 7.3 のアップグレードとダウングレード 2065
18.3 MySQL Cluster の構成 2066
18.3.1 MySQL Cluster の簡易テストセットアップ 2067
18.3.2 MySQL Cluster の構成ファイル 2069
18.3.3 MySQL Cluster 構成パラメータの概要 2141
18.3.4 MySQL Cluster 用の MySQL Server オプションおよび変数 2162
18.3.5 MySQL Cluster での高速インターコネクトの使用 2220
18.4 MySQL Cluster プログラム 2222
18.4.1 ndbd — MySQL Cluster データノードデーモン 2222
18.4.2 ndbinfo_select_all — ndbinfo テーブルからの選択 2228
18.4.3 ndbmtd — MySQL Cluster データノードデーモン (マルチスレッド) 2229
18.4.4 ndb_mgmd — MySQL Cluster 管理サーバーデーモン 2230
18.4.5 ndb_mgm — MySQL Cluster 管理クライアント 2237
18.4.6 ndb_blob_tool — MySQL Cluster テーブルの BLOB および TEXT カラムのチェックおよび修復 2238
18.4.7 ndb_config — MySQL Cluster 構成情報の抽出 2240
18.4.8 ndb_cpcd — NDB 開発のためのテストの自動化 2247
18.4.9 ndb_delete_all — NDB テーブルからのすべての行の削除 2247
18.4.10 ndb_desc — NDB テーブルの表示 2248
18.4.11 ndb_drop_index — NDB テーブルからのインデックスの削除 2251
18.4.12 ndb_drop_table — NDB テーブルの削除 2252
18.4.13 ndb_error_reporter — NDB エラーレポートユーティリティー 2253
18.4.14 ndb_index_stat — NDB インデックス統計ユーティリティー 2254
18.4.15 ndb_print_backup_file — NDB バックアップファイルの内容の出力 2258
18.4.16 ndb_print_file — NDB ディスクデータファイル内容の出力 2259
18.4.17 ndb_print_schema_file — NDB スキーマファイル内容の出力 2259
18.4.18 ndb_print_sys_file — NDB システムファイル内容の出力 2260
18.4.19 ndbd_redo_log_reader — クラスタ Redo ログ内容のチェックおよび出力 2260
18.4.20 ndb_restore — MySQL Cluster バックアップのリストア 2261
18.4.21 ndb_select_all — NDB テーブルの行の出力 2272
18.4.22 ndb_select_count — NDB テーブルの行数の出力 2275
18.4.23 ndb_setup.py — MySQL Cluster のブラウザベース自動インストーラの開始 2276
18.4.24 ndb_show_tables — NDB テーブルのリストの表示 2278
18.4.25 ndb_size.pl — NDBCLUSTER サイズ要件エスティメータ 2279
18.4.26 ndb_waiter — MySQL Cluster が指定したステータスになるまで待機する 2282
18.4.27 MySQL Cluster プログラムに共通するオプション — MySQL Cluster プログラムに共通するオプション 2284
18.5 MySQL Cluster の管理 2287
18.5.1 MySQL Cluster の起動フェーズのサマリー 2288
18.5.2 MySQL Cluster 管理クライアントのコマンド 2289
18.5.3 MySQL Cluster のオンラインバックアップ 2292

18.5.4 MySQL Cluster での MySQL サーバーの使用法 2296
18.5.5 MySQL Cluster のローリング再起動の実行 2297
18.5.6 MySQL Cluster で生成されたイベントレポート 2299
18.5.7 MySQL Cluster ログメッセージ 2308
18.5.8 MySQL Cluster のシングルユーザーモード 2317
18.5.9 クイックリファレンス: MySQL Cluster の SQL ステートメント 2317
18.5.10 ndbinfo MySQL Cluster 情報データベース 2319
18.5.11 MySQL Cluster のセキュリティー上の問題 2341
18.5.12 MySQL Cluster ディスクデータテーブル 2347
18.5.13 MySQL Cluster データノードのオンライン追加 2353
18.5.14 MySQL Cluster の配布された MySQL 権限 2362
18.5.15 NDB API 統計のカウンタと変数 2365
18.6 MySQL Cluster レプリケーション 2375
18.6.1 MySQL Cluster レプリケーション: 略語と記号 2376
18.6.2 MySQL Cluster レプリケーションの一般要件 2376
18.6.3 MySQL Cluster レプリケーションの既知の問題 2377
18.6.4 MySQL Cluster レプリケーションスキーマとテーブル 2382
18.6.5 レプリケーションのための MySQL Cluster の準備 2385
18.6.6 MySQL Cluster レプリケーションの起動 (レプリケーションチャネルが 1 つ) 2386
18.6.7 2 つのレプリケーションチャネルを使用する MySQL Cluster レプリケーション 2388
18.6.8 MySQL Cluster レプリケーションを使用したフェイルオーバーの実装 2389
18.6.9 MySQL Cluster レプリケーションを使用した MySQL Cluster バックアップ 2390
18.6.10 MySQL Cluster レプリケーション: マルチマスターと循環レプリケーション 2395
18.6.11 MySQL Cluster レプリケーションの競合解決 2399
18.7 MySQL Cluster リリースノート 2410

この章には、分散コンピューティング環境に適した MySQL の高可用性および高冗長性バージョンである MySQL Cluster に関する情報が含まれています。MySQL Cluster の最近のバージョンでは、NDB ストレージエンジン (NDBCLUSTER) のバージョン 7 を使用して、MySQL Server およびその他のソフトウェアを含む複数のコンピュータをクラスタ内で動作させることができます。本番環境で使用可能な最新リリースである MySQL Cluster NDB 7.3 には、NDB バージョン 7.3 が組み込まれています。この章には、NDB ストレージエンジンのバージョン 7.4 を使用し、開発者マイルストーンリリースで入手可能になった MySQL Cluster NDB 7.4 に関する情報も含まれています。

NDB ストレージエンジンのサポートは、オラクルによってビルドされた標準の MySQL Server 5.6 バイナリには含まれていません。代わりに、オラクルから提供される MySQL Cluster バイナリのユーザーは、サポートされるプラットフォームに対応する MySQL Cluster の最新のバイナリリリースにアップグレードするようにしてください。これらには、ほとんどの Linux 配布で機能するはずの RPM が含まれています。ソースからビルドする MySQL Cluster ユーザーは、MySQL Cluster 用に提供されるソースを使用するようにしてください。(ソースを入手できる場所については、このセクションで後述します。)

この章には、MySQL Cluster NDB 7.3 リリース (5.6.22-ndb-7.3.9 まで) および MySQL Cluster NDB 7.4 リリース (5.6.22-ndb-7.4.4 まで) に関する情報が含まれています。現在、MySQL Cluster NDB 7.3 リリースシリーズは一般提供 (GA) されており、MySQL Cluster 7.4 は開発者プレビューが入手可能です。MySQL Cluster NDB 7.2、MySQL Cluster NDB 7.1、および MySQL Cluster NDB 7.0 は以前の GA リリースです。これらは引き続きサポートされますが、新規の配備には MySQL Cluster NDB 7.3 を使用することをお勧めします。

サポートされるプラットフォーム　MySQL Cluster は現在数多くのプラットフォームで利用可能であり、サポートされています。オペレーティングシステムバージョン、オペレーティングシステム配布、およびハードウェアプラットフォームの特定の組み合わせで利用可能な正確なサポートレベルについては、https://www.mysql.com/support/supportedplatforms/cluster.html を参照してください。

可用性　サポートされるプラットフォーム用の MySQL Cluster のバイナリおよびソースパッケージは、https://dev.mysql.com/downloads/cluster/ から入手できます。

MySQL Cluster のリリース番号　MySQL Cluster は、メインラインの MySQL Server 5.6 のリリースシリーズとはやや異なるリリースパターンに従っています。このマニュアルおよび MySQL のその他のドキュメントでは、「NDB」で始まるバージョン番号を使用して、これらおよびそれ以降の MySQL Cluster リリースを識別します。このバージョン番号は、そのリリースで使用されている NDBCLUSTER ストレージエンジンのバージョン番号であり、その MySQL Cluster リリースのベースになっている MySQL Server バージョンのバージョン番号ではありません。

MySQL Cluster ソフトウェアで使用されるバージョン文字列　MySQL Cluster のプログラムに表示されるバージョン文字列には、この書式が使用されます。

```
mysql-mysql_server_version-ndb-ndb_engine_version
```

mysql_server_version は、その MySQL Cluster リリースのベースになっている MySQL Server のバージョンを表します。すべての MySQL Cluster NDB 7.3 リリースおよび現在の MySQL Cluster NDB 7.4 リリースでは、これは「5.6」です。ndb_engine_version は、このリリースの MySQL Cluster ソフトウェアで使用されている NDB ストレージエンジンのバージョンです。次に示すように、mysql クライアントでこの書式が使用されていることがわかります。

```
shell> mysql
Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 2
Server version: 5.6.22-ndb-7.4.4 Source distribution

Type 'help;' or '\h' for help. Type '\c' to clear the buffer.

mysql> SELECT VERSION()\G
*************************** 1. row ***************************
VERSION(): 5.6.22-ndb-7.4.4
1 row in set (0.00 sec)
```

このバージョン文字列は、ndb_mgm クライアントの SHOW コマンドの出力にも表示されます。

```
ndb_mgm> SHOW
Connected to Management Server at: localhost:1186
Cluster Configuration
---------------------
[ndbd(NDB)]     2 node(s)
id=1    @10.0.10.6  (5.6.22-ndb-7.4.4, Nodegroup: 0, *)
id=2    @10.0.10.8  (5.6.22-ndb-7.4.4, Nodegroup: 0)

[ndb_mgmd(MGM)] 1 node(s)
id=3    @10.0.10.2  (5.6.22-ndb-7.4.4)

[mysqld(API)]   2 node(s)
id=4    @10.0.10.10  (5.6.22-ndb-7.4.4)
id=5 (not connected, accepting connect from any host)
```

このバージョン文字列によって、その MySQL Cluster リリースの分岐元であるメインラインの MySQL のバージョンと、使用されている NDB ストレージエンジンのバージョンを特定できます。たとえば、MySQL Cluster NDB 7.3.2 (MySQL Server 5.6 をベースとする最初の MySQL Cluster 本番リリース) の完全なバージョン文字列は mysql-5.6.11-ndb-7.3.2 です。これによって、次のことを特定できます。

- バージョン文字列の「-ndb-」より前の部分はベースとなる MySQL Server のバージョンであるため、これは MySQL Cluster NDB 7.3.2 が MySQL 5.6.11 から派生し、MySQL 5.6 から MySQL 5.6.11 までのすべての機能強化とバグ修正を含んでいることを意味します。
- バージョン文字列の「-ndb-」よりあとの部分は NDB (または NDBCLUSTER) ストレージエンジンのバージョン番号を表しているため、MySQL Cluster NDB 7.3.2 ではバージョン 7.3.2 の NDBCLUSTER ストレージエンジンが使用されています。

新しい MySQL Cluster リリースは、NDB ストレージエンジンの更新に従って番号付けされ、メインラインの MySQL Server リリースとは必ずしも 1 対 1 で対応しません。たとえば、MySQL Cluster NDB 7.3.2 は (前述のように) MySQL 5.6.11 をベースとしていますが、MySQL Cluster NDB 7.3.1 は MySQL 5.6.10 をベースとしていました (バージョン文字列: mysql-5.6.10-ndb-7.3.1)。

標準の MySQL 5.6 リリースとの互換性　　標準の MySQL スキーマおよびアプリケーションの多くが MySQL Cluster を使用しても機能しますが、MySQL Cluster を使用して未変更のアプリケーションやデータベーススキーマを実行すると、互換性がやや低下したり、パフォーマンスが最適でなくなったりする可能性があることも事実です (セクション18.1.6「MySQL Cluster の既知の制限」を参照してください)。これらの問題のほとんどは克服できますが、これは、スキーマ、クエリー、およびアプリケーションに変更を加える可能性を考慮せずに、既存の (たとえば、MyISAM や InnoDB を使用する) アプリケーションデータストアを NDB ストレージエンジンを使用するものに切り替えることがほとんど不可能であることも意味します。さらに、MySQL Server と MySQL Cluster のコードベースは大幅に異なるため、標準の mysqld を MySQL Cluster に付属するバージョンの mysqld と簡単に取り替えて使用することはできません。

MySQL Cluster の開発ソースツリー　　MySQL Cluster の開発ツリーには、https://code.launchpad.net/~mysql/ からもアクセスできます。

https://code.launchpad.net/~mysql/ で管理されている MySQL Cluster の開発ソースは、GPL でライセンス付与されています。Bazaar を使った MySQL ソースの取得およびユーザー自身によるソースのビルドについては、セクション2.9.3「開発ソースツリーを使用して MySQL をインストールする」を参照してください。

> **注記**
>
> MySQL Server 5.6 と同じように、MySQL Cluster NDB 7.3 および MySQL Cluster NDB 7.4 リリースは CMake を使用してビルドされています。

現在、MySQL Cluster NDB 7.2 および MySQL Cluster NDB 7.3 リリースは一般提供 (GA) されています。新規の配備には MySQL Cluster NDB 7.3 を使用することをお勧めします。MySQL Cluster NDB 7.1 は以前の GA リリースで、現在も保守中です。MySQL Cluster NDB 7.0 以前のバージョンは、アクティブな開発が終了しました。MySQL Cluster NDB 7.3 に追加された主な機能の概要については、セクション18.1.4「MySQL Cluster の開発履歴」を参照してください。

この章はまだ完成されたものではなく、その内容は MySQL Cluster の継続的な進化に応じて改訂されます。MySQL Cluster に関する追加情報については、http://www.mysql.com/products/cluster/ の MySQL Web サイトを参照してください。

追加のリソース　MySQL Cluster の詳細は、次の場所を参照してください。

- MySQL Cluster に関するよくある質問の回答については、セクションA.10「MySQL 5.6 FAQ: MySQL Cluster」を参照してください。
- MySQL Cluster メーリングリスト: http://lists.mysql.com/cluster。
- MySQL Cluster フォーラム: https://forums.mysql.com/list.php?25。
- MySQL Cluster の多くのユーザーや開発者が MySQL Cluster に関する自身の経験をブログで公開し、PlanetMySQL でブログのフィードを利用できるようにしています。

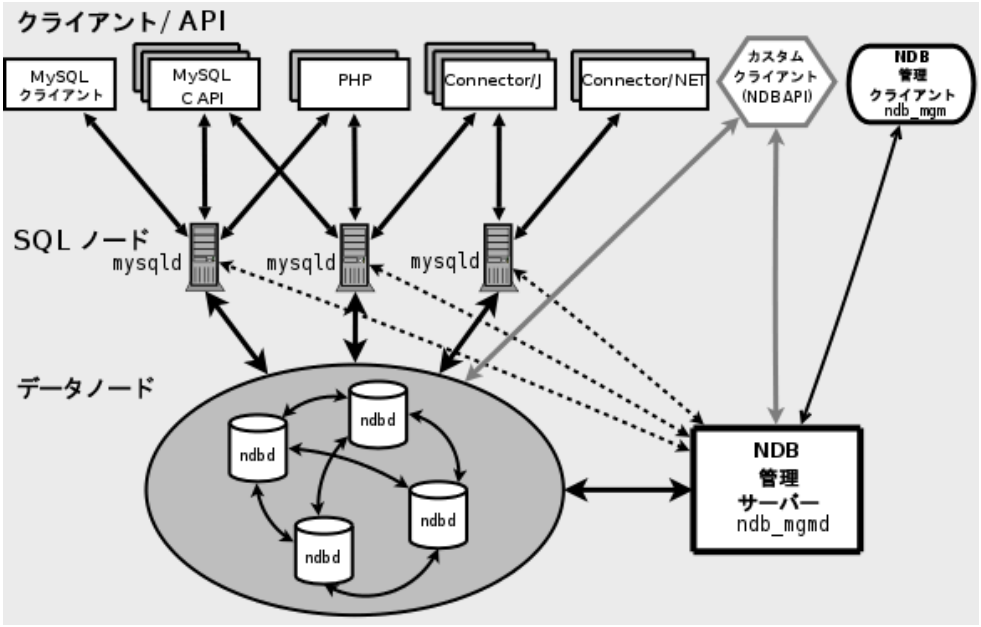
18.1 MySQL Cluster の概要

MySQL Cluster はシェアードナッシングシステムでのインメモリーデータベースのクラスタリングを可能にするテクノロジです。シェアードナッシングアーキテクチャーでは、非常に安価なハードウェアでシステムが動作し、ハードウェアやソフトウェアの要件が最小限に抑えられます。

MySQL Cluster は、単一障害点が発生しないように設計されています。シェアードナッシングシステムでは、各コンポーネントに固有のメモリーとディスクが用意され、ネットワーク共有、ネットワークファイルシステム、SAN などの共有ストレージメカニズムの使用は推奨またはサポートされません。

MySQL Cluster は、標準の MySQL Server と NDB (「Network DataBase」 を表します) と呼ばれるインメモリーのクラスタ化されたストレージエンジンを統合したものです。このドキュメントでは、NDB はセットアップのストレージエンジン固有の部分を表し、「MySQL Cluster」 は 1 つ以上の MySQL Server と NDB ストレージエンジンの組み合わせを表します。

MySQL Cluster は、それぞれが 1 つ以上のプロセスを実行するホストと呼ばれる一連のコンピュータで構成されます。ノードと呼ばれるこれらのプロセスには、MySQL Server (NDB データへのアクセス用)、データノード (データのストレージ用)、1 つ以上の管理サーバー、および場合によってはその他の特殊なデータアクセスプログラムが含まれます。MySQL Cluster 内のこれらのコンポーネントの関係をここに示します。



これらすべてのプログラムが一体となって MySQL Cluster を形成します (セクション18.4「MySQL Cluster プログラム」を参照してください)。NDB ストレージエンジンによってデータが格納されると、データノードにテーブル (およびテーブルデータ) が格納されます。このようなテーブルには、クラスタ内のほかのすべての MySQL Server (SQL ノード) から直接アクセスできます。したがって、クラスタにデータを格納する給料計算アプリケーションでは、あるアプリケーションが社員の給料を更新すると、このデータをクエリーするほかのすべての MySQL サーバーがこの変更をただちに認識できます。

MySQL Cluster の SQL ノードでは mysqld サーバーデーモンが使用されますが、これは MySQL 5.6 配布に付属する mysqld バイナリとはいくつかの重要な点で異なっており、これら 2 つのバージョンの mysqld を交換することはできません。

さらに、MySQL Cluster に接続されていない MySQL Server は、NDB ストレージエンジンを使用できず、MySQL Cluster のデータにアクセスできません。

MySQL Cluster のデータノードに格納されたデータはミラー化できます。クラスタは、少数のトランザクションがトランザクション状態の消失によって中止されること以外の影響を受けずに、個々のデータノードの障害を処理できます。トランザクションの失敗はトランザクションアプリケーションが処理すると想定されるため、これが問題の原因になることはないはずです。

個々のノードは、停止して再起動したあと、システム (クラスタ) に再度参加できます。構成の変更やソフトウェアのアップグレードを行うときは、ローリング再起動 (すべてのノードが順に再起動される) が使用されます (セクション18.5.5「MySQL Cluster のローリング再起動の実行」を参照してください)。ローリング再起動は、新しいデータノードをオンラインで追加するプロセスの一部としても使用されます (セクション18.5.13「MySQL Cluster データノードのオンライン追加」を参照してください)。データノードの詳細、MySQL Cluster でのデータノードの構成方法、およびデータノードによる MySQL Cluster データの処理方法と格納方法については、セクション18.1.2「MySQL Cluster のノード、ノードグループ、レプリカ、およびパーティション」を参照してください。

MySQL Cluster データベースのバックアップとリストアは、MySQL Cluster 管理クライアントにある NDB 固有の機能、および MySQL Cluster 配布に含まれる ndb_restore プログラムを使用して行うことができます。詳細は、セクション18.5.3「MySQL Cluster のオンラインバックアップ」およびセクション18.4.20「ndb_restore — MySQL Cluster バックアップのリストア」を参照してください。mysqldump および MySQL Server でこの目的のために用意されている標準の MySQL 機能を使用することもできます。詳細は、セクション4.5.4「mysqldump — データベースバックアッププログラム」を参照してください。

MySQL Cluster ノードのノード間通信では、標準の 100M ビット/秒またはより高速な Ethernet ハードウェアを使用する TCP/IP など、多くの異なる転送メカニズムを使用できます。MySQL Cluster では、高速なスケーラブルコヒーラントインタフェース (SCI) プロトコルを使用することもできますが、これは MySQL Cluster を使用する上で必須ではありません。SCI には特別なハードウェアとソフトウェアが必要です。SCI および MySQL Cluster での SCI の使用方法の詳細は、セクション18.3.5「MySQL Cluster での高速インターコネクトの使用」を参照してください。

18.1.1 MySQL Cluster の主な概念

NDBCLUSTER (NDB とも呼ばれる) は、高可用性とデータ永続性の機能を備えたインメモリーストレージエンジンです。

NDBCLUSTER ストレージエンジンはさまざまなフェイルオーバーとロードバランシングのオプションを使って構成できますが、クラスタレベルのストレージエンジンから始めるのがもっとも簡単です。MySQL Cluster の NDB ストレージエンジンには、クラスタ自体の内部にあるほかのデータにのみ依存する完全なデータセットが格納されます。

MySQL Cluster の「クラスタ」部分は、MySQL Server とは別個に構成されます。MySQL Cluster では、クラスタの各部分がノードとみなされます。

> **注記**
>
> 「ノード」という用語は、多くのコンテキストではコンピュータを示すために使用されますが、MySQL Cluster について説明するときはプロセスを意味します。1 台のコンピュータで複数のノードを実行できます。1 つ以上のクラスタノードを実行しているコンピュータに対しては、クラスタホストという用語を使用します。

クラスタノードには 3 つのタイプがあります。最小限の MySQL Cluster 構成には少なくとも 3 つのノードがあり、各ノードがこれらのタイプのそれぞれに対応します。

- 管理ノード: このタイプのノードの役割は、構成データの提供、ノードの起動と停止、バックアップの実行などの機能を実行して、MySQL Cluster 内のほかのノードを管理することです。このノードタイプはほかのノードの構成を管理するため、このタイプのノードは最初に (ほかのノードより先に) 起動するようにしてください。MGM ノードは ndb_mgmd コマンドで起動します。

- データノード: このタイプのノードにはクラスタのデータが格納されます。データノードの数は、レプリカの数にフラグメントの数を乗算した数です (セクション18.1.2「MySQL Cluster のノード、ノードグループ、レプリカ、およびパーティション」を参照してください)。たとえば、レプリカが 2 つあり、それぞれにフラグメントが 2 つある場合は、4 つのデータノードが必要です。データストレージとしては 1 つのレプリカで十分ですが、それでは冗長性がありません。したがって、レプリカを 2 つ以上にして、冗長性 (およびその結果としての高可用性) を確保することをお勧めします。データノードは、ndbd (セクション18.4.1「ndbd — MySQL Cluster データノードデーモン」を参照してください) または ndbmtd (セクション18.4.3「ndbmtd — MySQL Cluster データノードデーモン (マルチスレッド)」を参照してください) コマンドで起動します。

  通常、MySQL Cluster テーブルはディスクではなくメモリー内に完全に格納されます (MySQL Cluster をインメモリーデータベースと呼ぶ理由はここにあります)。ただし、MySQL Cluster の一部のデータはディスクに格納できます。詳細は、セクション18.5.12「MySQL Cluster ディスクデータテーブル」を参照してください。

- SQL ノード: これは、クラスタデータにアクセスするノードです。MySQL Cluster の場合、SQL ノードは NDBCLUSTER ストレージエンジンを使用する従来の MySQL Server です。SQL ノードは、--ndbcluster および --ndb-connectstring オプション (これらについては、この章の別の場所で説明します) を指定して起動される mysqld プロセスです。場合によっては、追加の MySQL Server オプションも指定できます。

  SQL ノードは、実際には MySQL Cluster データにアクセスするアプリケーションが指定された特別なタイプの API ノードです。API ノードのもう 1 つの例は、クラスタのバックアップをリストアするために使われる ndb_restore ユーティリティーです。NDB API を使用してこのようなアプリケーションを作成できます。NDB API の基本情報については、Getting Started with the NDB APIを参照してください。

> **重要**
>
> 本番環境で 3 ノードセットアップの採用を期待するのは現実的ではありません。このような構成には冗長性がありません。MySQL Cluster の高可用性機能の恩恵を受けるには、複数のデータノードと SQL ノードを使用する必要があります。複数の管理ノードを使用することも、強くお勧めします。

MySQL Cluster のノード間、ノードグループ間、レプリカ間、およびパーティション間の関係の概要については、セクション18.1.2「MySQL Cluster のノード、ノードグループ、レプリカ、およびパーティション」を参照してください。

クラスタの構成には、クラスタ内の各ノードの構成と、ノード間の個々の通信リンクの設定が含まれます。MySQL Cluster は現在、データノードがプロセッサの性能、メモリースペース、および帯域幅に関して均一になるよう開発されています。さらに、一元管理の構成を提供するため、クラスタのすべての構成データが全体として 1 つの構成ファイルに格納されます。

管理サーバーは、クラスタ構成ファイルとクラスタログを管理します。クラスタ内の各ノードは、管理サーバーから構成データを取得するため、管理サーバーがどこにあるかを特定する手段を必要とします。データノードで

注目に値するイベントが発生すると、そのノードはこれらのイベントに関する情報を管理サーバーに転送し、管理サーバーはその情報をクラスタログに書き込みます。

さらに、任意の数のクラスタクライアントプロセスまたはアプリケーションが存在する可能性があります。これらには、標準の MySQL クライアント、NDB 専用の API プログラム、管理クライアントなどが含まれます。次のいくつかの段落で、これらについて説明します。

標準の MySQL クライアント　MySQL Cluster は、PHP、Perl、C、C++、Java、Python、Ruby などで作成された既存の MySQL アプリケーションとともに使用できます。このようなクライアントアプリケーションは、スタンドアロンの MySQL Server とやり取りする場合とほとんど同じ方法で、MySQL Cluster の SQL ノードとして機能する MySQL Server に SQL ステートメントを送信し、その MySQL Server から応答を受信します。

MySQL Cluster をデータソースとして使用する MySQL クライアントは、複数の MySQL Server に接続する機能を利用してロードバランシングとフェイルオーバーを実現するように変更できます。たとえば、Connector/J 5.0.6 以降を使用する Java クライアントは、jdbc:mysql:loadbalance:// URL (Connector/J 5.1.7 で改良済み) を使用してロードバランシングを透過的に実現できます。MySQL Cluster での Connector/J の使用方法の詳細は、Using Connector/J with NDB Clusterを参照してください。

NDB クライアントプログラム　クラスタに接続されている可能性がある MySQL Server をバイパスし、高レベルの C++ API である NDB API を使用して NDBCLUSTER ストレージエンジンから直接 MySQL Cluster データにアクセスするクライアントプログラムを作成できます。このようなアプリケーションは、データへの SQL インタフェースを必要としない特殊な目的に役立ちます。詳細は、The NDB APIを参照してください。

Java 用 MySQL Cluster Connector を使用して、MySQL Cluster 用に NDB 専用の Java アプリケーションを作成することもできます。この MySQL Cluster Connector には、NDBCLUSTER に直接接続する Hibernate や JPA のようなオブジェクトリレーショナルマッピングの永続性フレームワークに似た高レベルのデータベース API である ClusterJ が含まれているため、MySQL Server へのアクセスは必要ありません。MySQL Cluster NDB 7.1 以降では、ClusterJ と JDBC の長所を生かした MySQL Cluster 用の OpenJPA 実装である ClusterJPA もサポートされます。ID ルックアップやその他の高速処理は ClusterJ を使用して (MySQL Server をバイパスして) 実行されますが、MySQL のクエリーオプティマイザによるメリットが得られるより複雑なクエリーは、JDBC を使用して MySQL Server 経由で送信されます。詳細は、Java and NDB ClusterおよびThe ClusterJ API and Data Object Modelを参照してください。

MySQL Cluster NDB 7.3 以降では、Node.js を使用して JavaScript で作成されたアプリケーションもサポートされます。JavaScript 用の MySQL Connector には、MySQL Server だけでなく、NDB ストレージエンジンに直接アクセスするためのアダプタも含まれています。この Connector を使用するアプリケーションは、通常イベント駆動型であり、ClusterJ に採用されているものと多くの点で似ているドメインオブジェクトモデルを使用します。詳細は、MySQL NoSQL Connector for JavaScriptを参照してください。

memcached バージョン 1.6 以降用のロード可能な ndbmemcache ストレージエンジンとして実装されている MySQL Cluster 用の Memcache API を使用して、memcache プロトコルを使用してアクセスされる永続的な MySQL Cluster データストアを提供できます。

標準の memcached キャッシュエンジンは、MySQL Cluster NDB 7.3 以降の配布に含まれています。各 memcached サーバーは、MySQL Cluster に格納されたデータに直接アクセスしますが、データをローカルでキャッシュして、このローカルキャッシュから要求を処理することもできます。

詳細は、ndbmemcache—Memcache API for NDB Cluster (DEPRECATED)を参照してください。

管理クライアント　これらのクライアントは、管理サーバーに接続して、ノードの正常な起動と停止、メッセージトレースの開始と停止 (デバッグバージョンのみ)、ノードのバージョンとステータスの表示、バックアップの開始と停止などのコマンドを提供します。このタイプのプログラムの例として、MySQL Cluster に付属の ndb_mgm 管理クライアントがあります (セクション18.4.5「ndb_mgm — MySQL Cluster 管理クライアント」を参照してください)。このようなアプリケーションは、1 つ以上の MySQL Cluster 管理サーバーと直接通信する C 言語 API である MGM API を使用して作成できます。詳細は、The MGM APIを参照してください。

オラクルは、多数のノードを含む MySQL Cluster の再起動など、MySQL Cluster の多くの複雑な管理タスクを簡略化する高度なコマンド行インタフェースを備えた MySQL Cluster Manager も提供しています。MySQL Cluster Manager クライアントは、MySQL Cluster に関連する mysqld サーバーのオプションや変数に加えて、ほとんどのノード構成パラメータの値を取得および設定するためのコマンドもサポートします。詳細は、MySQL™ Cluster Manager 1.3.6 User Manualを参照してください。

イベントログ　MySQL Cluster は、イベントをカテゴリ (起動、シャットダウン、エラー、チェックポイントなど)、優先度、および重大度別に記録します。すべてのレポート可能イベントの完全なリストについては、セクション18.5.6「MySQL Cluster で生成されたイベントレポート」を参照してください。イベントログには、ここに示す 2 つのタイプがあります。

- クラスタログ: クラスタに関する必要なすべてのレポート可能イベントが全体として保持されます。

- ノードログ: 個々のノードごとに保持される個別のログです。

  > **注記**
  >
  > 通常の状況では、クラスタログのみを保持して調べるだけで必要十分です。ノードログを調べる必要があるのは、アプリケーション開発やデバッグの場合だけです。

チェックポイント　一般的には、データをディスクに保存したときにチェックポイントに達したといいます。MySQL Cluster に限定した場合、チェックポイントはコミットされたトランザクションがディスクに格納された時点です。NDB ストレージエンジンについては 2 つのタイプのチェックポイントがあり、その組み合わせによってクラスタのデータを一貫して確認し続けることができます。次のリストにそれらを示します。

- ローカルチェックポイント (LCP): これは、1 つのノードに固有のチェックポイントです。ただし、LCP はクラスタ内のすべてのノードである程度同時に発生します。LCP は、ノードのすべてのデータをディスクに保存する必要があるため、通常は数分間隔で発生します。正確な間隔は、ノードに格納されているデータの量、クラスタのアクティビティーのレベル、およびその他の要因によって異なります。
- グローバルチェックポイント (GCP): GCP は、すべてのノードのトランザクションが同期し、Redo ログがディスクにフラッシュされたときに、数秒間隔で発生します。

ローカルチェックポイントとグローバルチェックポイントによって作成されるファイルおよびディレクトリの詳細は、NDB Cluster Data Node File System Directoryを参照してください。

18.1.2 MySQL Cluster のノード、ノードグループ、レプリカ、およびパーティション

このセクションでは、MySQL Cluster が格納するデータを分割および複製する方法について説明します。

次の各段落では、このトピックを理解する上で中心となるいくつかの概念について説明します。

(データ) ノード　レプリカ (そのノードがメンバーとなっているノードグループに割り当てられたパーティション (次を参照してください) のコピー) を格納する ndbd プロセスです。

各データノードは、別個のコンピュータに配置するようにしてください。1 つのコンピュータで複数の ndbd プロセスをホストすることも可能ですが、そのような構成はサポートされていません。

通常、ndbd プロセスに言及するときは、「ノード」と「データノード」を区別せずに使用します。この説明で管理ノード (ndb_mgmd プロセス) と SQL ノード (mysqld プロセス) に言及するときは、そのように明示します。

ノードグループ　ノードグループは、1 つ以上のノードで構成され、パーティションまたはレプリカ (次の項目を参照してください) のセットを格納します。

MySQL Cluster 内のノードグループの数を直接構成することはできません。これは、ここに示すように、データノードの数とレプリカの数 (NoOfReplicas 構成パラメータ) の関数です。

```
[number_of_node_groups] = number_of_data_nodes / NoOfReplicas
```

したがって、4 つのデータノードを含む MySQL Cluster で、config.ini ファイルの NoOfReplicas が 1 に設定されている場合は 4 つのノードグループが存在し、NoOfReplicas が 2 に設定されている場合は 2 つのノードグループが存在し、NoOfReplicas が 4 に設定されている場合は 1 つのノードグループが存在します。レプリカについてはこのセクションで後述します。NoOfReplicas の詳細は、セクション18.3.2.6「MySQL Cluster データノードの定義」を参照してください。

> **注記**
>
> MySQL Cluster 内のすべてのノードグループに同じ数のデータノードが含まれている必要があります。

実行中の MySQL Cluster にオンラインで新しいノードグループ (およびその結果としての新しいデータノード) を追加できます。詳細は、セクション18.5.13「MySQL Cluster データノードのオンライン追加」を参照してください。

パーティション　これは、クラスタに格納されるデータの一部分です。クラスタに参加するノードと同じ数のクラスタパーティションが存在します。各ノードは、そのノードに割り当てられたパーティションの少なくとも 1 つのコピー (つまり、少なくとも 1 つレプリカ) をクラスタで利用可能な状態に維持する役割を担います。

レプリカは完全に 1 つのノードに属します。ノードには複数のレプリカを格納できます (そして通常はそのようにします)。

NDB とユーザー定義のパーティション化　通常、MySQL Cluster は NDBCLUSTER テーブルを自動的にパーティション化します。ただし、NDBCLUSTER テーブルにユーザー定義のパーティション化を適用することもできます。これには、次の制限があります。

1. NDB テーブルには、KEY および LINEAR KEY パーティション化スキームのみを使用できます。
2. ndbd を使用する場合、NDB テーブルに対して明示的に定義できるパーティションの最大数は 8 * [ノードグループの数] です。(MySQL Cluster 内のノードグループの数は、このセクションで前述した方法で決定されます。)

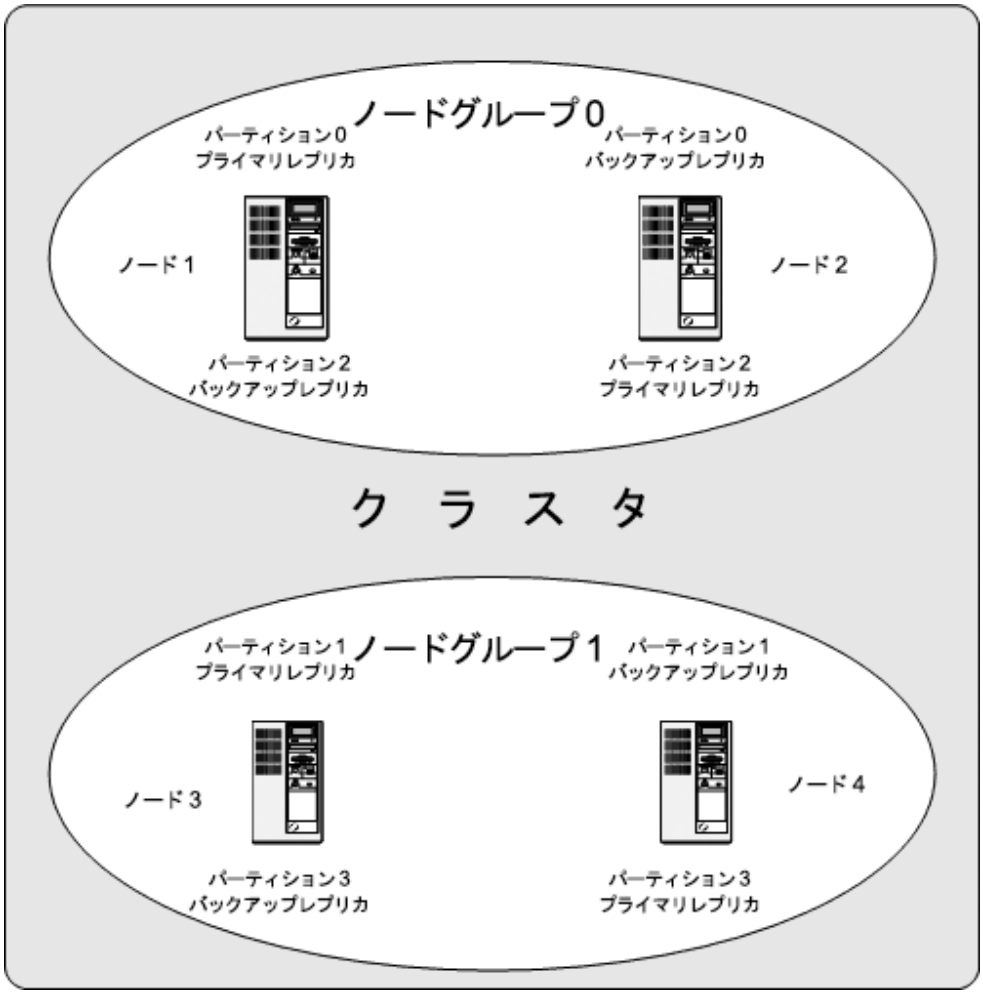
   ndbmtd を使用する場合、この最大数は MaxNoOfExecutionThreads 構成パラメータの値によって決定されるローカルクエリーハンドラスレッドの数にも影響されます。このような場合、NDB テーブルに対して明示的に定義できるパーティションの最大数は 4 * MaxNoOfExecutionThreads * [ノードグループの数] になります。

   詳細は、セクション18.4.3「ndbmtd — MySQL Cluster データノードデーモン (マルチスレッド)」を参照してください。

MySQL Cluster とユーザー定義のパーティション化に関する詳細は、セクション18.1.6「MySQL Cluster の既知の制限」およびセクション19.6.2「ストレージエンジンに関連するパーティショニング制限」を参照してください。

レプリカ　これは、クラスタパーティションのコピーです。レプリカは、ノードグループ内の各ノードに格納されます。パーティションレプリカと呼ばれることもあります。レプリカの数は、ノードグループあたりのノード数と同じになります。

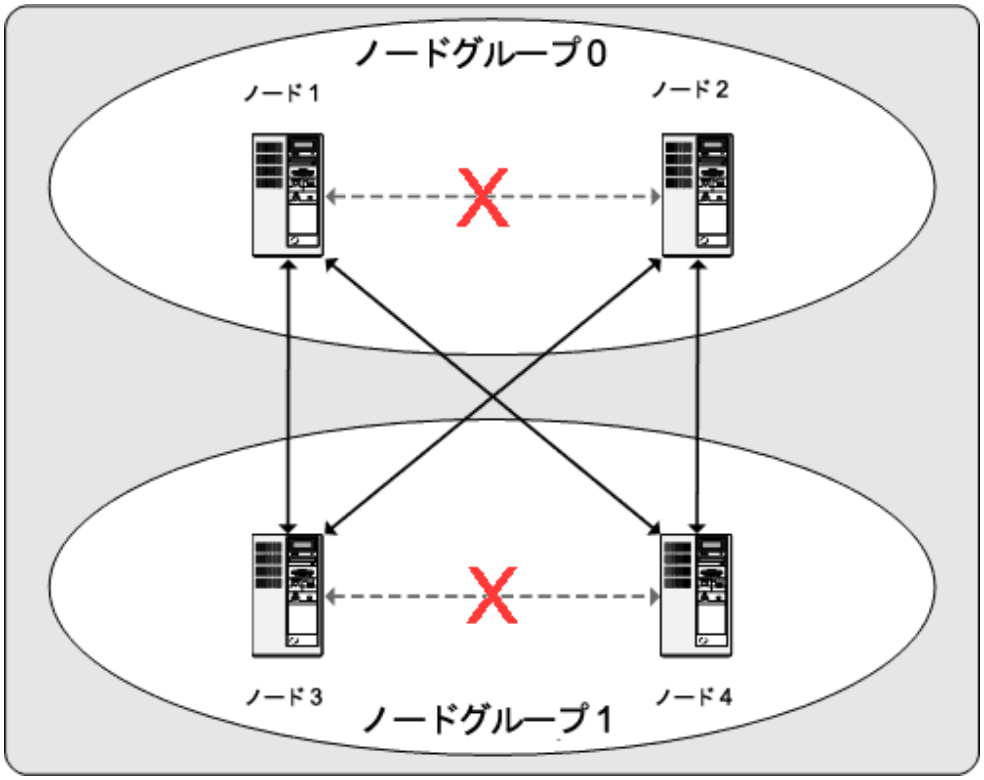
次の図に、4 つのデータノードを含む MySQL Cluster を示します。これらのノードはそれぞれ 2 ノードから成る 2 つのノードグループに配置されています。ノード 1 および 2 はノードグループ 0 に属し、ノード 3 および 4 はノードグループ 1 に属します。ここにはデータ (ndbd) ノードのみを表示しています。正常に機能するクラスタにはクラスタ管理用の ndb_mgm プロセスと、クラスタに格納されているデータにアクセスするための SQL ノードが少なくとも 1 つ必要ですが、この図ではわかりやすくするためにこれらを省略しています。



クラスタに格納されたデータは、0、1、2、3 の番号が付けられた 4 つのパーティションに分割されています。各パーティションは同じノードグループに (複数のコピーで) 格納されます。パーティションは、次のように各ノードグループに交互に格納されます。

- パーティション 0 はノードグループ 0 に格納されます。プライマリレプリカ (プライマリコピー) がノード 1 に格納され、バックアップレプリカ (パーティションのバックアップコピー) がノード 2 に格納されます。
- パーティション 1 は別のノードグループ (ノードグループ 1) に格納されます。このパーティションのプライマリレプリカがノード 3 に格納され、そのバックアップレプリカがノード 4 に格納されます。
- パーティション 2 はノードグループ 0 に格納されます。ただし、その 2 つのレプリカはパーティション 0 とは逆に配置されます。プライマリレプリカがノード 2 に格納され、バックアップレプリカがノード 1 に格納されます。
- パーティション 3 はノードグループ 1 に格納され、その 2 つのレプリカはパーティション 1 とは逆に配置されます。つまり、プライマリレプリカがノード 4 に格納され、バックアップレプリカがノード 3 に格納されます。

これは、MySQL Cluster の継続的な運用に関して次のような意味を持っています。クラスタに参加している各ノードグループで少なくとも 1 つのノードが動作しているかぎり、クラスタはすべてのデータの完全なコピーを保持し、実行可能な状態を維持します。これを次の図に示します。



この例では、クラスタはそれぞれに 2 つのノードを含む 2 つのノードグループで構成されており、クラスタを「存続」させるには、ノードグループ 0 の少なくとも 1 つのノードとノードグループ 1 の少なくとも 1 つのノードの任意の組み合わせ (図の矢印で示したもの) があれば十分です。ただし、どちらかのノードグループの両方のノードに障害が発生した場合、残りの 2 つのノード (X 印を付けた矢印で示したもの) では不十分です。どちらの場合も、クラスタからパーティション全体が消失するため、すべてのクラスタデータの完全なセットへのアクセスを提供できなくなります。

18.1.3 MySQL Cluster のハードウェア、ソフトウェア、およびネットワーク要件

MySQL Cluster の長所の 1 つは、汎用のハードウェアで実行できるため、すべてのライブデータをメモリーに格納するために大量の RAM を必要とする以外は、ハードウェアに関して特別な要件がないことです。(ディスクデータテーブルを使用してこの要件を削減することもできます。詳細は、セクション18.5.12「MySQL Cluster ディスクデータテーブル」を参照してください。)当然ながら、複数の高速な CPU を使用すればパフォーマンスは向上します。ほかの MySQL Cluster プロセスのメモリー要件は、比較的少量です。

MySQL Cluster のソフトウェア要件も小規模です。ホストのオペレーティングシステムに、MySQL Cluster をサポートするための特別なモジュール、サービス、アプリケーション、または構成は必要ありません。サポートされるオペレーティングシステムについては、標準のインストールで十分のはずです。MySQL のソフトウェア要件は単純です。必要なものは MySQL Cluster の本番リリースだけです。MySQL Cluster を使用できるようにするだけなら、必ずしも MySQL をユーザー自身でコンパイルする必要はありません。ここでは、ユーザーが MySQL Cluster ソフトウェアダウンロードページ (https://dev.mysql.com/downloads/cluster/) から入手できる各自のプラットフォームに適したバイナリを使用していると仮定します。

ノード間の通信については、MySQL Cluster は標準的なトポロジの TCP/IP ネットワークをサポートします。各ホストには、標準の 100M ビット/秒 Ethernet カードが最低限必要です。さらに、クラスタ全体のネットワーク接続を提供するためにスイッチ、ハブ、またはルーターが必要です。次の理由で、クラスタの構成要素でないマシンと共有しない独立したサブネットで MySQL Cluster を実行することをお勧めします。

- セキュリティー　MySQL Cluster ノード間の通信では、暗号化や保護がまったく行われません。MySQL Cluster 内の伝送を保護する唯一の手段は、MySQL Cluster を保護されたネットワークで実行することです。MySQL Cluster を Web アプリケーションのために使用する場合は、クラスタをネットワークの非武装地帯 (DMZ) などに配置せず、ファイアウォールの背後に間違いなく配置するようにしてください。

  詳細は、セクション18.5.11.1「MySQL Cluster のセキュリティーとネットワーク上の問題」を参照してください。

- 効率　MySQL Cluster をプライベートネットワークや保護されたネットワークにセットアップすると、クラスタはクラスタホスト間の帯域幅を独占的に使用できます。MySQL Cluster 用に別個のスイッチを使用すると、MySQL Cluster データを不正アクセスから保護できるだけでなく、ネットワーク上のほかのコンピュータ間の伝送によって発生する障害から MySQL Cluster ノードを確実に保護できます。信頼性を高めるため、デュアルスイッチとデュアルカードを使用して、単一障害点になったネットワークを除去できます。多くのデバイスドライバがこのような通信リンクのフェイルオーバーをサポートしています。

ネットワーク通信と待機時間　MySQL Cluster では、データノードと API ノード (SQL ノードを含む) 間、およびデータノードと別のデータノード間でクエリーや更新を実行するために通信を行う必要があります。これらのプロセス間通信の待機時間は、ユーザークエリーの観測されるパフォーマンスと待機時間に直接影響を与える可能性があります。さらに、ノードのサイレント障害が発生した場合でも一貫性とサービスを維持するため、MySQL Cluster では長時間にわたるノードからの通信の喪失をノード障害として扱うハートビートとタイムアウトのメカニズムが使用されます。これは、冗長性の低下につながる可能性があります。ノードグループ内の最後のノードに障害が発生すると、MySQL Cluster はデータの一貫性を維持するためにシャットダウンすることを思い出してください。したがって、強制的なシャットダウンのリスクを増やさないようにするため、ノード間通信の中断はできるかぎり回避すべきです。

データノードまたは API ノードに障害が発生すると、障害が発生したノードに関係するコミットされていないすべてのトランザクションが中止されます。データノードをリカバリするには、データノードのサービスを再開する前に、障害が発生したノードのデータを残存するデータノードのデータと同期し、ディスクベースの redo ログとチェックポイントログを再構築する必要があります。このリカバリには時間がかかる場合があり、その間、クラスタは冗長性が低下した状態で動作します。

ハートビートは、すべてのノードでハートビート信号が遅延なく生成されるかどうかに依存しています。ノードに過大な負荷がかかったり、マシンの CPU がほかのプログラムと共有されるために不十分だったり、スワッピングによる遅延が発生したりすると、これは不可能になります。ハートビート生成の遅延が十分に大きくなると、ほかのノードは応答が遅いノードを障害発生ノードとして扱います。

このように遅いノードを障害発生ノードとして扱うことは、ノードの遅い動作がクラスタの残りの部分にどの程度影響するかによって、望ましい場合とそうでない場合があります。MySQL Cluster の HeartbeatIntervalDbDb や HeartbeatIntervalDbApi などのタイムアウト値を設定するときは、高コストの可能性がある誤判定を回避しながら、迅速な検出、フェイルオーバー、サービス再開が実現されるように配慮する必要があります。

データノード間の通信待機時間が LAN 環境での予想値 (およそ 100 マイクロ秒) より大きくなると予想される場合は、タイムアウトのパラメータを増やして、許容する待機時間が構成したタイムアウトの範囲に十分に収まるようにする必要があります。このようにタイムアウトを増やすと、最大の障害検出時間および (その結果としての) サービスリカバリ時間にも類似の効果があります。

LAN 環境は、通常、安定した小さい待機時間で構成できるため、高速フェイルオーバーによる冗長性を提供できます。個々のリンク障害は、TCP レベルで認識できる最小限の制御された (MySQL Cluster が正常に動作する) 待機時間でリカバリできます。WAN 環境では、待機時間に幅があり、冗長性についてもフェイルオーバーに時間がかかる可能性があります。個々のリンク障害では、エンドツーエンドの接続をリストアする前に、ルート変更を伝播する必要があります。これは、TCP レベルでは個々のチャネルの大きな待機時間として現れます。これらのシナリオで最悪の場合に観測される TCP 待機時間は、IP 層で障害を回避するために行われる再ルーティングの最大時間に関連しています。

SCI のサポート　MySQL Cluster に高速なスケーラブルコヒーラントインタフェース (SCI) を使用することもできますが、これは必要条件ではありません。このプロトコルおよび MySQL Cluster での使用方法の詳細は、セクション18.3.5「MySQL Cluster での高速インターコネクトの使用」を参照してください。

18.1.4 MySQL Cluster の開発履歴

次の 2 つのセクションでは、MySQL Cluster NDB 7.3 および MySQL Cluster NDB 7.4 での MySQL Cluster 実装の変更点について、MySQL Cluster NDB 7.2 以前のリリースと比較して説明します。もっとも注目に値する変更点と機能を次の 2 つの表に示します。

MySQL Cluster NDB 7.3
MySQL Cluster NDB 7.3 は MySQL 5.6 をベースにしています。MySQL Server 5.6 の新機能の詳細は、セクション1.4「MySQL 5.6 の新機能」を参照してください。
MySQL Cluster NDB 7.3 は、テーブルの外部キー制約をサポートします。詳細は、セクション 1.7.3.2「FOREIGN KEY の制約」およびセクション13.1.17.2「外部キー制約の使用」を参照してください。
MySQL Cluster NDB 7.3 は、JavaScript 用 MySQL NoSQL Connector を使用して Node.js のサポートを提供します。詳細は、MySQL NoSQL Connector for JavaScriptを参照してください。

MySQL Cluster NDB 7.4
MySQL Cluster NDB 7.4 は MySQL 5.6 をベースにしています (MySQL Server 5.6 の新機能の詳細は、セクション1.4「MySQL 5.6 の新機能」を参照してください)。
競合例外テーブルに対する拡張機能を含む、MySQL Cluster レプリケーションの競合検出および解決の拡張 (セクション18.6.11「MySQL Cluster レプリケーションの競合解決」を参照してください)
循環 (「アクティブ-アクティブ」) レプリケーションの管理の改善 (ndb_slave_conflict_role によるプライマリ/セカンダリの割り当て)
memory_per_fragment テーブルでのフラグメント単位のメモリー使用状況レポート
次の拡張機能を含む多数のパフォーマンス改善 • 初期メモリー割り当ての高速化 • ローカルチェックポイントの並列化の強化 (LCP でサポートされるフラグメント数が 2 から 32 に増加) このバージョンで導入された一連の構成パラメータ (MaxDiskWriteSpeed、MaxDiskWriteSpeedOtherNoderestart、MaxDiskWriteSpeedOwnRestart) によって、LCP 中のディスク書き込みの制御が改善されました このバージョンの ndbinfo データベースに追加された disk_write_speed_base、disk_write_speed_aggregate、および disk_write_speed_aggregate_node テーブルで、最近のディスク書き込みに関する情報を入手できます • バックアップから MySQL Cluster をリストアする時間の短縮 • NDB 受信スレッドの最適化
ノード再起動中のエラーおよびその他のレポートの改善

これらのセクションには、MySQL Cluster NDB 7.3 リリースの 5.6.22-ndb-7.3.9 (現在一般提供リリースとして入手可能) まで、および MySQL Cluster NDB 7.4 リリースの 5.6.22-ndb-7.4.4 (現在開発者マイルストーンリリースとして入手可能) までに関する情報が含まれています。MySQL Cluster NDB 7.2 および MySQL Cluster NDB 7.1 は、以前の GA リリースシリーズであり、現在もサポートされていますが、新規の配備には MySQL Cluster NDB 7.3 を使用することをお勧めします。

18.1.4.1 MySQL Cluster NDB 7.3 での MySQL Cluster の開発

MySQL Cluster NDB 7.3 では、MySQL Cluster に対して次の改善が行われました。

- MySQL Server 5.6 ベースになった　MySQL Cluster NDB 7.3 は、MySQL Server 5.6 をベースにしているため、MySQL Cluster ユーザーは MySQL 5.6 の拡張性とパフォーマンスモニタリングの改善によるメリットを得ることができます。MySQL 5.6 と同様に、MySQL Cluster NDB 7.3 では構成とソースからのビルドに CMake が使用されます。MySQL 5.6 での変更点と改善点の詳細は、セクション1.4「MySQL 5.6 の新機能」を参照してください。

- 外部キー　NDB ストレージエンジンバージョン 7.3.0 以降を使用して作成されたテーブルでは、外部キー制約がサポートされます。(これには、すべての MySQL Cluster NDB 7.3 リリースが含まれます。)MySQL 5.6 および MySQL Cluster NDB 7.3 での外部キーの処理方法に関する一般的な情報は、セクション1.7.3.2「FOREIGN KEY の制約」を参照してください。構文および関連情報については、セクション13.1.17「CREATE TABLE 構文」およびセクション13.1.17.2「外部キー制約の使用」を参照してください。

- Node.js のサポート　MySQL Cluster NDB 7.3 は、Node.js を使用して JavaScript で作成されたアプリケーションもサポートします。JavaScript 用の MySQL Connector には、MySQL Server だけでなく、NDB ストレージエンジンに直接アクセスするためのアダプタも含まれています。この Connector を使用するアプリケーションは、通常イベント駆動型であり、ClusterJ に採用されているものと多くの点で似ているドメインオブジェクトモデルを使用します。詳細は、MySQL NoSQL Connector for JavaScriptを参照してください。

MySQL Cluster NDB 7.3 は、MySQL Cluster の多くの複雑な管理タスクを簡略化できる高度なコマンド行インタフェースを備えた MySQL Cluster Manager でもサポートされます。詳細は、MySQL™ Cluster Manager 1.3.6 User Manualを参照してください。

18.1.4.2 MySQL Cluster NDB 7.4 での MySQL Cluster の開発

MySQL Cluster NDB 7.4 では、MySQL Cluster に対して次の改善が行われました。

- 競合検出および解決の拡張　例外テーブルのメタカラムに対して予約されたカラム名名前空間 NDB$ が採用され、メインテーブルのカラムの任意のサブセットが元のテーブルの主キーの一部でなくても、それらを記録できるようになりました。

  例外テーブルのカラムとの一致が名前と型だけで行われるようになったため、元の主キーを完全に記録する必要がなくなりました。メインテーブルの主キーの一部ではないカラムの値を例外テーブルに記録することも可能になりました。

  読み取りの競合検出が可能になりました。競合するトランザクションによって読み取られたすべての行がフラグ付けされ、例外テーブルに記録されます。同じトランザクションに挿入された行は、読み取りまたは記録の対象となる行に含まれません。この読み取り追跡は、スレーブが事前に ndb_log_exclusive_reads を設定する必要がある排他的な読み取りロックを持っているかどうかに依存します。詳細と例については、読み取り競合の検出と解決を参照してください。

  既存の例外テーブルも引き続きサポートされます。詳細は、セクション18.6.11「MySQL Cluster レプリケーションの競合解決」を参照してください。

- 循環 (「アクティブ-アクティブ」) レプリケーションでのロール管理　MySQL Cluster の循環 (「アクティブ-アクティブ」) レプリケーショントポロジを使用する場合は、ndb_slave_conflict_role サーバーシステム変数を使用して、プライマリまたはセカンダリのいずれかのロールを特定の MySQL Cluster に割り当てることができます。この変数には、PRIMARY、SECONDARY、PASS、NULL のいずれかの値を指定できます。デフォルトは NULL です。これは、プライマリとして機能する MySQL Cluster からフェイルオーバーするときに使用できます。

  競合解決機能の効果が無視されるパススルー状態 (PASS) もサポートされます。詳細は、ndb_slave_conflict_role 変数の説明およびセクション18.6.11「MySQL Cluster レプリケーションの競合解決」を参照してください。

- フラグメント単位のメモリー使用状況レポート　メモリー使用状況に関するデータを、MySQL Cluster NDB 7.4.1 の ndbinfo 情報データベースに追加された memory_per_fragment ビューから個々の MySQL Cluster フラグメント単位で取得できるようになりました。詳細は、セクション18.5.10.17「ndbinfo memory_per_fragment テーブル」を参照してください。

- ノードの再起動の改善

  MySQL Cluster NDB 7.4 には、データノードの再起動にかかる時間を短縮する多数の改善が含まれています。次のリストで、これらについて説明します。

  - ノードの起動時に割り当てられるメモリーは割り当てられるまで使用できないため、オペレーティングシステムは必要な実際の物理メモリーを確保します。MySQL Cluster の以前のバージョンでは、割り当てられたメモリーの各ページにアクセスするプロセスはシングルスレッド化されていたため、比較的時間がかかっていました。このプロセスがマルチスレッドで実装し直されました。16 個のスレッドによるテストでは、シングルスレッドの場合より約 3 倍速いアクセス時間が観測されました。

  - ローカルチェックポイントの並列化が強化され、MySQL Cluster NDB 7.4 では LCP でサポートされるフラグメント数が以前の 2 から 32 に増加しました。これにより、使用されなかった CPU パワーの利用率が大幅に向上し、LCP が最大 10 倍高速化されます。この高速化によってノードの再起動時間も大幅に短縮されます。

- 新しい ndbinfo テーブルである disk_write_speed_base、disk_write_speed_aggregate、および disk_write_speed_aggregate_node によってディスク書き込みのレポートが提供されます。これらのテーブルは、使用中の各 LDM スレッドのディスク書き込み速度に関する情報を提供します。

  このリリースでは、現在のノードまたは別のノードが現在再起動されているとき、あるいは現在再起動されているノードがないときの LCP およびバックアップの書き込み速度を制御するため、MinDiskWriteSpeed、MaxDiskWriteSpeed、MaxDiskWriteSpeedOtherNodeRestart、および MaxDiskWriteSpeedOwnRestart データノード構成パラメータも追加されています。

  これらの変更の目的は、DiskCheckpointSpeed および DiskCheckpointSpeedInRestart 構成パラメータを使ったディスク書き込みの構成の代替となることです。これら 2 つのパラメータは現在非推奨になっており、今後の MySQL Cluster リリースで削除される予定です。

- パフォーマンスにとって重要であることが判明したポイントで検出された遅延シグナルを通常の (遅延しない) シグナルに置換することで、バックアップから MySQL Cluster をリストアする時間が短縮されます。これらの不要な遅延シグナルを除去または置換することで、MySQL Cluster のバックアップやバックアップからの MySQL Cluster のリストアにかかる時間が大幅に短縮されるはずです。

- NDB の受信スレッドに関連する複数の内部メソッドが最適化され、NDB による SQL 処理の効率が向上しました。受信スレッドは、場合によっては 1 秒間に数百万件の受信レコードを処理する必要があるため、MySQL Cluster のデータノードからレコードを取得するときに不要な作業を実行したり、リソースを浪費したりしないようにすることが重要です。

- MySQL Cluster の起動フェーズのレポートが改善され、起動中の出力回数が増加しました。セクション 18.5.1「MySQL Cluster の起動フェーズのサマリー」を参照してください。

18.1.5 InnoDB を使用した MySQL Server と MySQL Cluster の比較

MySQL Server では、ストレージエンジンに数多くの選択肢があります。NDBCLUSTER と InnoDB は、どちらもトランザクション対応の MySQL ストレージエンジンとして機能するため、MySQL Server のユーザーが MySQL Cluster に興味を持つこともあります。NDB は、MySQL 5.6 のデフォルトの InnoDB ストレージエンジンの代替候補またはアップグレード候補とみなされています。NDB と InnoDB には共通する特徴もありますが、アーキテクチャーや実装が異なるため、既存の MySQL Server アプリケーションや使用シナリオの中には MySQL Cluster に適合するものもありますが、すべてが適合するわけではありません。

このセクションでは、MySQL Cluster NDB 7.3 で使用されている NDB ストレージエンジンの特徴を、MySQL 5.6 で使用されている InnoDB と比較して説明します。次のいくつかのセクションでは、技術的な比較を示します。多くの場合、MySQL Cluster をいつどこで使用するかは、すべての要因を考慮に入れながらケースバイケースで決定する必要があります。このドキュメントでは、考えられるあらゆる使用シナリオの詳細を示すことはしませんが、一般的なタイプのアプリケーションが InnoDB バックエンドに比べて NDB にどの程度適合するかについて、ごく一般的なガイダンスを示します。

MySQL Cluster NDB 7.3 では MySQL 5.6 をベースとする mysqld が使用されますが、これには InnoDB 1.1 のサポートも含まれます。MySQL Cluster で InnoDB テーブルを使用することは可能ですが、このようなテーブルはクラスタ化されません。MySQL Server 5.6 で MySQL Cluster NDB 7.3 配布のプログラムやライブラリを使用すること (またはその逆) もできません。

一般的なビジネスアプリケーションの中には MySQL Cluster と MySQL Server (ほとんどの場合、InnoDB ストレージエンジンを使用) のどちらかで実行可能なタイプがあることも事実ですが、アーキテクチャーと実装には重要な違いがあります。これらの違いのサマリーについては、セクション18.1.5.1「NDB および InnoDB ストレージエンジンの違い」を参照してください。これらの違いのため、一部の使用シナリオは明らかにどちらか一方のエンジンに適しています。セクション18.1.5.2「NDB と InnoDB のワークロード」を参照してください。これは、NDB または InnoDB とともに使用するのが適切なアプリケーションのタイプにも影響を与えます。それぞれが一般的なタイプのデータベースアプリケーションでの使用にどの程度適合するかの比較については、セクション18.1.5.3「NDB および InnoDB の機能使用のサマリー」を参照してください。

NDB および MEMORY ストレージエンジンの相対的な特徴については、MEMORY または MySQL Cluster を使用する場合を参照してください。

MySQL ストレージエンジンに関する追加情報は、第15章「代替ストレージエンジン」を参照してください。

18.1.5.1 NDB および InnoDB ストレージエンジンの違い

MySQL Cluster の NDB ストレージエンジンは、分散型のシェアードナッシングアーキテクチャーを使用して実装されているため、InnoDB とは多くの点で動作が異なります。NDB の操作に慣れていないユーザーにとって

は、NDB の持つ分散型の性質によって、トランザクション、外部キー、テーブルの制限、およびその他の特性に関して予期しない動作が発生する可能性があります。次の表にそれらを示します。

機能	InnoDB 1.1	MySQL Cluster NDB 7.3、MySQL Cluster NDB 7.4
MySQL Server のバージョン	5.6	5.6
InnoDB のバージョン	InnoDB 5.6.23	InnoDB 5.6.23
MySQL Cluster のバージョン	N/A	NDB 7.3.9
ストレージの制限	64T バイト	3T バイト (それぞれ 64G バイトの RAM を備えた 48 個のデータノードを基にした現実的な上限。ディスクベースのデータや BLOB によって増える可能性があります)
外部キー	はい	MySQL Cluster NDB 7.3 より前のバージョン: いいえ。(MyISAM と同様に無視されます) MySQL Cluster NDB 7.3 では利用可能です。
トランザクション	すべての標準タイプ	READ COMMITTED
MVCC	はい	いいえ
データ圧縮	はい	いいえ (MySQL Cluster のチェックポイントおよびバックアップファイルは圧縮できます)
大きな行のサポート (> 14K)	VARBINARY、VARCHAR、BLOB、および TEXT カラムでサポートされます	BLOB および TEXT カラムでのみサポートされます (これらの型を使用して非常に大量のデータを格納すると、MySQL Cluster のパフォーマンスが低下する可能性があります)
レプリケーションのサポート	MySQL レプリケーションを使用した非同期および準同期レプリケーション	MySQL Cluster 内での自動的な同期レプリケーション。 MySQL レプリケーションを使用した MySQL Cluster 間の非同期レプリケーション
読み取り操作のスケールアウト	はい (MySQL レプリケーション)	はい (MySQL Cluster 内の自動パーティション化、MySQL レプリケーション)
書き込み操作のスケールアウト	アプリケーションレベルのパーティション化 (シャーディング) が必要です	はい (MySQL Cluster 内の自動パーティション化がアプリケーションに対して透過的に行われます)
高可用性 (HA)	追加のソフトウェアが必要です	はい (99.999% の稼働時間を実現するように設計されています)
ノードの障害リカバリとフェイルオーバー	追加のソフトウェアが必要です	自動 (MySQL Cluster アーキテクチャーの重要な要素)
ノードの障害リカバリにかかる時間	30 秒以上	通常は 1 秒未満
リアルタイムのパフォーマンス	いいえ	はい
インメモリーテーブル	いいえ	はい (必要に応じて一部のデータをディスクに格納できます。インメモリーと

機能	InnoDB 1.1	MySQL Cluster NDB 7.3、MySQL Cluster NDB 7.4
		ディスクの両方のデータストレージが永続的です)
ストレージエンジンへの NoSQL アクセス	ネイティブの memcached インタフェースが開発中です (MySQL Dev Zone の記事「MySQL Cluster 7.2 (DMR2): NoSQL, Key/Value, Memcached」を参照してください)	はい Memcached、Node.js/ JavaScript、Java、JPA、C++、HTTP/REST を含む複数の API
同時および並列書き込み	サポートされません	同時書き込みのために最適化された最大 48 個のライター
競合検出および解決 (複数のレプリケーションマスター)	いいえ	はい
ハッシュインデックス	いいえ	はい
オンラインのノード追加	MySQL レプリケーションを使った読み取り専用レプリカ	はい (すべてのノードタイプ)
オンラインのアップグレード	いいえ	はい
オンラインのスキーマ変更	はい (MySQL 5.6 の一部として)。	はい。

18.1.5.2 NDB と InnoDB のワークロード

MySQL Cluster には、高可用性、高速フェイルオーバー、高スループット、および低待機時間を必要とするアプリケーションに対応するのに最適なさまざまな固有の属性があります。MySQL Cluster には、その分散型アーキテクチャーと複数ノード実装による固有の制約も存在し、そのために一部のワークロードが適切に実行されないことがあります。データベース駆動型アプリケーションの一般的なワークロードのタイプに関する NDB ストレージエンジンと InnoDB ストレージエンジンの動作の主な違いを、次の表にいくつか示します。

ワークロード	InnoDB	MySQL Cluster (NDB)
大容量 OLTP アプリケーション	はい	はい
DSS アプリケーション (データマート、分析)	はい	制限付き (サイズが 3T バイトを超える OLTP データセット間の結合操作は不可)
カスタムアプリケーション	はい	はい
パッケージ化されたアプリケーション	はい	制限付き (ほとんどが主キーアクセス)。 MySQL Cluster NDB 7.3 は外部キーをサポートします。
ネットワーク内電気通信アプリケーション (HLR、HSS、SDP)	いいえ	はい
セッション管理およびキャッシュ	はい	はい
電子商取引アプリケーション	はい	はい
ユーザープロファイル管理、AAA プロトコル	はい	はい

18.1.5.3 NDB および InnoDB の機能使用のサマリー

InnoDB と NDB の機能に対するアプリケーション機能要件を比較すると、その一部は明らかにどちらか一方のストレージエンジンと高い互換性があります。

次の表に、サポートされるアプリケーション機能を、一般に各機能がより適合するストレージエンジンに従って示します。

InnoDB がより適合するアプリケーション要件	NDB がより適合するアプリケーション要件
• 外部キー	• 書き込みのスケーリング

InnoDB がより適合するアプリケーション要件	NDB がより適合するアプリケーション要件
注記 MySQL Cluster NDB 7.3 は外部キーをサポートします。 • 完全なテーブルスキャン • 非常に大規模なデータベース、行、またはトランザクション • READ COMMITTED 以外のトランザクション	• 99.999% の稼働時間 • オンラインのノード追加とオンラインのスキーマ操作 • 複数の SQL および NoSQL API (NDB Cluster APIs: Overview and Concepts を参照してください) • リアルタイムのパフォーマンス • BLOB カラムの限定的な使用 • 外部キーがサポートされますが、外部キーの使用は高スループットでのパフォーマンスに影響する可能性があります。

18.1.6 MySQL Cluster の既知の制限

以降のセクションでは、MySQL Cluster の現在のリリースに含まれる既知の制限について、MyISAM および InnoDB ストレージエンジンを使用した場合に使用できる機能と比較して説明します。MySQL バグデータベース (http://bugs.mysql.com) で 「Cluster」 カテゴリをチェックすると、MySQL バグデータベース (http://bugs.mysql.com) の 「MySQL Server:」 の下にある次のカテゴリで既知のバグが見つかります。これらのバグは、MySQL Cluster の今後のリリースで修正される予定です。

- MySQL Cluster
- Cluster Direct API (NDBAPI)
- Cluster Disk Data
- Cluster Replication
- ClusterJ

この情報は、規定された条件に関して完結するように意図されています。不具合が発生した場合は、セクション 1.6「質問またはバグをレポートする方法」に示す手順を使用して MySQL バグデータベースにそれらを報告できます。MySQL Cluster NDB 7.3 の問題を修正する予定がない場合、その問題はリストに追加されます。

MySQL Cluster NDB 7.3 で解決された MySQL Cluster NDB 7.2 の問題のリストについては、セクション 18.1.6.11「MySQL Cluster NDB 7.3 で解決された以前の MySQL Cluster の問題」を参照してください。

> 注記
>
> MySQL Cluster レプリケーションに固有の制限およびその他の問題については、セクション18.6.3「MySQL Cluster レプリケーションの既知の問題」を参照してください。

18.1.6.1 MySQL Cluster での SQL 構文の不適合

次のリストに示すように、特定の MySQL 機能に関連する SQL ステートメントの中には、NDB テーブルで使用したときにエラーを生成するものがあります。

- 一時テーブル　一時テーブルはサポートされません。NDB ストレージエンジンを使用する一時テーブルを作成しようとしたり、NDB を使用するように既存の一時テーブルを変更しようとしたりすると、失敗して次のエラーが発生します:Table storage engine 'ndbcluster' does not support the create option 'TEMPORARY'.
- NDB テーブルのインデックスとキー　MySQL Cluster テーブルのインデックスとキーには、次の制限があります。
  - カラム幅　3072 バイトを超える幅の NDB テーブルカラムに対するインデックスを作成しようとすると、成功しますが、インデックスに実際に使用されるのは最初の 3072 バイトだけです。このような場合は、警告 Specified key was too long; max key length is 3072 bytes が発行され、SHOW CREATE TABLE ステートメントではインデックスの長さが 3072 と表示されます。
  - TEXT および BLOB カラム　TEXT または BLOB データ型のどちらかを使用する NDB テーブルカラムに対するインデックスは作成できません。

- FULLTEXT インデックス　NDB ストレージエンジンは、FULLTEXT インデックスをサポートしません。これは、MyISAM および (MySQL 5.6.4 以降の) InnoDB テーブルでのみ可能です。

  ただし、NDB テーブルの VARCHAR カラムに対するインデックスは作成できます。

- USING HASH キーと NULL　一意キーおよび主キーで NULL 可能カラムを使用すると、これらのカラムを使用するクエリーは完全なテーブルスキャンとして処理されます。この問題を回避するには、カラムを NOT NULL にするか、USING HASH オプションを指定せずにインデックスを再作成します。

- プリフィクス　プリフィクスインデックスはありません。インデックスはカラム全体にのみ付けることができます。(NDB のカラムインデックスのサイズは、このセクションで前述したように、カラムの幅 (バイト単位) と常に同じ (最大 3072 バイトまで) です。追加情報については、セクション18.1.6.6「MySQL Cluster の未サポート機能または欠落している機能」を参照してください。)

- BIT カラム　BIT カラムを主キー、一意キー、またはインデックスにすることはできません。また、複合の主キー、一意キー、またはインデックスの一部にすることもできません。

- AUTO_INCREMENT カラム　ほかの MySQL ストレージエンジンと同様に、NDB ストレージエンジンはテーブルあたり最大 1 つの AUTO_INCREMENT カラムを処理できます。ただし、明示的な主キーがないクラスタテーブルの場合は、AUTO_INCREMENT カラムが自動的に定義され、「隠し」主キーとして使用されます。このため、明示的な AUTO_INCREMENT カラムを持つテーブルは、PRIMARY KEY オプションを同時に使用してカラムを宣言しないかぎり定義できません。テーブルの主キーでない AUTO_INCREMENT カラムを持つテーブルを作成しようとして、NDB ストレージエンジンを使用すると、失敗してエラーが発生します。

- 外部キーに関する制限　MySQL Cluster NDB 7.3 での外部キー制約のサポートは、InnoDB によるサポートと同等ですが、次の制限があります。

  - 外部キーとして参照されるすべてのカラムは、それがテーブルの主キーでない場合、明示的な一意キーを必要とします。

  - 参照先が親テーブルの主キーである場合、ON UPDATE CASCADE はサポートされません。

  - SET DEFAULT はサポートされません。(InnoDB でもサポートされません。)

  - NO ACTION キーワードは受け入れられますが、RESCRICT として扱われます。(InnoDB でも同じです。)

  - MySQL Cluster NDB 7.3.5 以前のバージョンでは、別のテーブルのインデックスを参照する外部キーを含むテーブルを作成したときに、インデックス内のカラムの順序が一致しなくても内部で適切なエラーが返されないことがあったため、外部キーを作成できるように見える場合がありました。MySQL Cluster NDB 7.3.5 ではこの問題が部分的に修正され、内部で使用されるエラーがほとんどの場合に機能するように改善されましたが、親インデックスが一意のインデックスである場合は、まだこの状況が発生する可能性があります。(Bug #18094360)

  詳細は、セクション13.1.17.2「外部キー制約の使用」およびセクション1.7.3.2「FOREIGN KEY の制約」を参照してください。

- MySQL Cluster とジオメトリデータ型
  NDB テーブルでは、ジオメトリデータ型 (WKT および WKB) がサポートされます。ただし、空間インデックスはサポートされません。

- 文字セットとバイナリログファイル　現在、ndb_apply_status および ndb_binlog_index テーブルは latin1 (ASCII) 文字セットを使用して作成されています。バイナリログの名前はこのテーブルに記録されるため、ラテン語以外の文字を使用する名前が付けられたバイナリログファイルはこれらのテーブルで正しく参照されません。これは既知の問題であり、現在修正中です。(Bug #50226)

  この問題を回避するには、バイナリログファイルの名前を付けるときや、--basedir、--log-bin、または --log-bin-index オプションを設定するときに、Latin-1 文字のみを使用してください。

- ユーザー定義のパーティション化を使用した NDBCLUSTER テーブルの作成　MySQL Cluster でのユーザー定義のパーティション化のサポートは、[LINEAR] KEY のパーティション化に限定されます。CREATE TABLE ステートメントで ENGINE=NDB または ENGINE=NDBCLUSTER とともにほかのパーティショニングタイプを使用すると、エラーが発生します。

  デフォルトのパーティション化スキーム　すべての MySQL Cluster テーブルは、デフォルトではテーブルの主キーをパーティション化キーとして使用して KEY によってパーティション化されます。テーブルに明示的に設定された主キーがない場合は、代わりに NDB ストレージエンジンによって自動的に作成された「隠し」主

キーが使用されます。これらおよび関連する問題の追加情報については、セクション19.2.5「KEY パーティショニング」を参照してください。

ユーザーパーティション化された NDBCLUSTER テーブルが次の 2 つの要件のどちらかまたは両方を満たさない原因となる CREATE TABLE および ALTER TABLE ステートメントは許可されず、失敗してエラーが発生します。

1. テーブルに明示的な主キーが存在する必要があります。

2. テーブルのパーティショニング式に指定されたすべてのカラムが主キーの一部である必要があります。

例外　空のカラムリストを使用して (つまり、PARTITION BY [LINEAR] KEY() を使用して) ユーザーパーティション化された NDBCLUSTER テーブルを作成する場合は、明示的な主キーは必要ありません。

NDBCLUSTER テーブルの最大パーティション数　ユーザー定義のパーティション化を使用したときに NDBCLUSTER テーブルに定義できるパーティションの最大数は、ノードグループあたり 8 個です。(MySQL Cluster ノードグループの詳細は、セクション18.1.2「MySQL Cluster のノード、ノードグループ、レプリカ、およびパーティション」を参照してください。

DROP PARTITION の未サポート　ALTER TABLE ... DROP PARTITION を使用して NDB テーブルからパーティションを削除することはできません。クラスタテーブルでは ALTER TABLE に対する別のパーティション化拡張機能 (ADD PARTITION、REORGANIZE PARTITION、および COALESCE PARTITION) がサポートされますが、コピーが使用されるため、最適化されません。セクション19.3.1「RANGE および LIST パーティションの管理」およびセクション13.1.7「ALTER TABLE 構文」を参照してください。

- 行ベースのレプリケーション
  MySQL Cluster で行ベースのレプリケーションを使用するときは、バイナリロギングを無効にすることができません。つまり、NDB ストレージエンジンは sql_log_bin の値を無視します。(Bug #16680)

18.1.6.2 MySQL Cluster の制限と標準の MySQL の制限との違い

このセクションでは、MySQL Cluster に含まれる制限のうち、標準の MySQL に含まれる制限とは異なる (または含まれない) ものを示します。

メモリーの使用量とリカバリ　ほかのストレージエンジンと同様、NDB テーブルにデータを挿入したときに消費されたメモリーは、削除したときに自動的にリカバリされません。代わりに、次のルールが適用されます。

- NDB テーブルに対して DELETE ステートメントを実行すると、削除された行で以前使用されていたメモリーが同じテーブルでの挿入にかぎって再利用可能になります。ただし、OPTIMIZE TABLE を実行すると、このメモリーの一般的な再利用が可能になります。

  クラスタのローリング再起動が行われた場合も、削除された行で使用されていたメモリーが解放されます。セクション18.5.5「MySQL Cluster のローリング再起動の実行」を参照してください。

- NDB テーブルに対して DROP TABLE または TRUNCATE TABLE 操作を実行すると、このテーブルで使用されていたメモリーが解放され、任意の NDB テーブルで (同じテーブルでも別の NDB テーブルでも) 再利用可能になります。

  > **注記**
  >
  > TRUNCATE TABLE によってテーブルが削除され、再作成されることを思い出してください。セクション13.1.33「TRUNCATE TABLE 構文」を参照してください。

- クラスタの構成によって課される制限
  多くの構成可能なハード制限がありますが、クラスタ内の利用可能なメインメモリーによって制限が設定されます。セクション18.3.2「MySQL Cluster の構成ファイル」の構成パラメータの完全なリストを参照してください。ほとんどの構成パラメータはオンラインでアップグレードできます。これらのハード制限には次のようなものがあります。

  - データベースのメモリーサイズとインデックスのメモリーサイズ (それぞれ DataMemory と IndexMemory)。

    DataMemory は 32K バイトのページとして割り当てられます。各 DataMemory ページは、使用されたときに特定のテーブルに割り当てられます。割り当てられたあとは、テーブルを削除した場合を除いてこのメモリーを解放できません。

    詳細は、セクション18.3.2.6「MySQL Cluster データノードの定義」を参照してください。

- 1 つのトランザクションで実行できる操作の最大数は、構成パラメータ MaxNoOfConcurrentOperations および MaxNoOfLocalOperations を使用して設定されます。

  > **注記**
  >
  > 一括ロード、TRUNCATE TABLE、および ALTER TABLE は、複数のトランザクションを実行することによって特別なケースとして扱われるため、この制限が適用されません。

- テーブルとインデックスに関するさまざまな制限。たとえば、クラスタ内の順序付けされたインデックスの最大数は MaxNoOfOrderedIndexes によって決定され、テーブルあたりの順序付けされたインデックスの最大数は 16 です。

- ノードとデータオブジェクトの最大数　クラスタノードとメタデータオブジェクトの数には、次の制限が適用されます。

  - データノードの最大数は 48 です。

    データノードは 1 から 48 までの (これらを含む) 範囲のノード ID を持つ必要があります。(管理ノードと API ノードには、1 から 255 までの (これらを含む) 範囲のノード ID が使用される場合があります。)

  - MySQL Cluster 内のノードの最大合計数は 255 です。この数には、すべての SQL ノード (MySQL Server)、API ノード (MySQL Server 以外のクラスタにアクセスするアプリケーション)、データノード、および管理サーバーが含まれます。

  - メタデータオブジェクトの最大数は、MySQL Cluster の現在のバージョンでは 20320 です。この制限はハードコーディングされています。

  詳細は、セクション18.1.6.11「MySQL Cluster NDB 7.3 で解決された以前の MySQL Cluster の問題」を参照してください。

18.1.6.3 MySQL Cluster でのトランザクション処理に関する制限

MySQL Cluster には、トランザクションの処理に関していくつかの制限があります。これらには、次のものが含まれます。

- トランザクション分離レベル　NDBCLUSTER ストレージエンジンは、READ COMMITTED トランザクション分離レベルのみをサポートします。(たとえば、InnoDB は READ COMMITTED、READ UNCOMMITTED、REPEATABLE READ、および SERIALIZABLE をサポートします。)これがクラスタデータベースのバックアップとリストアに与える影響については、セクション18.5.3.4「MySQL Cluster バックアップのトラブルシューティング」を参照してください。)

- トランザクションと BLOB または TEXT カラム　NDBCLUSTER は、MySQL から認識できるテーブルに MySQL の BLOB または TEXT データ型のいずれかを使用するカラム値の一部のみを格納します。BLOB または TEXT の残りの部分は、MySQL からアクセスできない別の内部テーブルに格納されます。これに関連して、これらの型のカラムを含むテーブルに対して SELECT ステートメントを実行するときに常に注意すべき問題が 2 つ発生します。

  1. MySQL Cluster テーブルからの SELECT の場合: SELECT に BLOB または TEXT カラムが含まれる場合は、READ COMMITTED トランザクション分離レベルが読み取りロック付きの読み取りに変換されます。これは一貫性を保証するために行われます。

  2. 一意キーのルックアップを使用して BLOB または TEXT データ型のいずれかを使用するカラムを取得し、1 つのトランザクション内で実行される SELECT の場合は、共有読み取りロックがトランザクションの期間中 (つまり、トランザクションがコミットまたは中止されるまで) そのテーブルに保持されます。

     インデックスまたはテーブルスキャンを使用するクエリーでは、BLOB または TEXT カラムを含む NDB テーブルが対象であっても、この問題は発生しません。

     たとえば、次の CREATE TABLE ステートメントによって定義されたテーブル t について考えます。

```
CREATE TABLE t (
    a INT NOT NULL AUTO_INCREMENT PRIMARY KEY,
    b INT NOT NULL,
    c INT NOT NULL,
    d TEXT,
    INDEX i(b),
    UNIQUE KEY u(c)
```

```
) ENGINE = NDB,
```

t に対する次のクエリーでは、1 つ目のクエリーが主キールックアップを使用し、2 つ目が一意キールックアップを使用しているため、いずれも共有読み取りロックが発生します。

```
SELECT * FROM t WHERE a = 1;

SELECT * FROM t WHERE c = 1;
```

しかし、ここに示す 4 つのクエリーでは、いずれも共有読み取りロックは発生しません。

```
SELECT * FROM t WHERE b 1;

SELECT * FROM t WHERE d = '1';

SELECT * FROM t;

SELECT b,c WHERE a = 1;
```

その理由は、これら 4 つのクエリーのうち、1 つ目はインデックススキャンを使用し、2 つ目と 3 つ目はテーブルスキャンを使用し、4 つ目は (主キールックアップを使用していますが) BLOB または TEXT カラムの値を取得していないためです。

BLOB または TEXT カラムを取得する一意キールックアップを使用するクエリーを回避するか、このようなクエリーを回避できない場合はトランザクションのコミットをできるだけあとで行うようにすると、共有読み取りロックによる問題を最小限に抑えることができます。

- ロールバック　部分的なトランザクションおよびトランザクションの部分的なロールバックはありません。重複キーまたは同様のエラーが発生すると、トランザクション全体がロールバックされます。

  この動作は、個々のステートメントをロールバックできる InnoDB などのほかのトランザクション対応のストレージエンジンと異なります。

- トランザクションとメモリー使用量
  この章の別の場所で説明したように、MySQL Cluster では大規模なトランザクションが適切に処理されません。多数の操作を含む 1 つの大規模なトランザクションを実行するより、少数の操作を含む複数の小規模なトランザクションを実行する方が適切です。特に考慮すべき点は、大規模なトランザクションが非常に大量のメモリーを必要とすることです。このため、いくつかの MySQL ステートメントのトランザクション動作に対して次のリストに示すような影響があります。

  - TRUNCATE TABLE は、NDB テーブルに対して使用した場合、トランザクションになりません。TRUNCATE TABLE でテーブルを空にできない場合は、成功するまでこれを再実行する必要があります。

  - DELETE FROM (WHERE 句が内場合も含む) は、トランザクションになります。テーブルに非常に多くの行が含まれる場合は、複数の DELETE FROM ... LIMIT ... ステートメントを使用して削除操作を「ひとまとめに」すると、パフォーマンスが向上することがあります。テーブルを空にすることが目的である場合は、代わりに TRUNCATE TABLE を使用することをお勧めします。

  - LOAD DATA ステートメント　LOAD DATA INFILE は、NDB テーブルに使用した場合、トランザクションになりません。

    > **重要**
    >
    > LOAD DATA INFILE ステートメントを実行すると、NDB エンジンは通信ネットワークの利用率が高くなるように不規則な間隔でコミットを実行します。このようなコミットが発生するタイミングを事前に知ることはできません。

  - ALTER TABLE とトランザクション　ALTER TABLE の一部として NDB テーブルをコピーする場合、コピーの作成はトランザクションになりません。(いずれにしても、コピーが削除されたときにこの操作はロールバックされます。)

- トランザクションと COUNT() 関数　MySQL Cluster レプリケーションを使用する場合、スレーブでの COUNT() 関数のトランザクション一貫性は保証されません。つまり、1 つのトランザクション内でテーブル内の行数を変更する一連のステートメント (INSERT、DELETE、またはその両方) をマスターで実行しているときに、スレーブで SELECT COUNT(*) FROM table クエリーを実行すると、中間の結果が返される可能性があります。これは、SELECT COUNT(...) がダーティー読み取りを行うために発生し、NDB ストレージエンジンのバグではありません。(詳細は、Bug #31321 を参照してください。)

18.1.6.4 MySQL Cluster のエラー処理

ノードの起動、停止、および再起動によって、トランザクションが失敗する原因となる一時的なエラーが発生する場合があります。これらには、次のケースが含まれます。

- 一時エラー　ノードを最初に起動するときは、エラー 1204 Temporary failure, distribution changedおよび同様の一時エラーが発生する可能性があります。
- ノード障害によるエラー　データノードの停止または障害によって、いくつかの異なるノード障害エラーが発生する場合があります。(ただし、クラスタの計画的なシャットダウンを実行したときに中止されるトランザクションは存在しないはずです。)

これらのいずれの場合も、生成されたエラーはアプリケーションの内部で処理する必要があります。これは、トランザクションの再試行によって行うべきです。

セクション18.1.6.2「MySQL Cluster の制限と標準の MySQL の制限との違い」も参照してください。

18.1.6.5 MySQL Cluster 内のデータベースオブジェクトに関する制限

NDBCLUSTER ストレージエンジンを使用するときは、テーブルやインデックスなどの一部のデータベースオブジェクトに関してさまざまな制限があります。

- データベース名とテーブル名　NDB ストレージエンジンを使用するときは、データベース名とテーブル名の最大許容長はどちらも 63 文字です。

  MySQL Cluster NDB 7.3.8 以降では、この制限を超える長さのデータベース名またはテーブル名を使用するステートメントは該当するエラーで失敗します。(Bug #19550973)

- データベースオブジェクトの数　1 つの MySQL Cluster に含まれるすべての NDB データベースオブジェクト (データベース、テーブル、およびインデックスを含む) の最大数は、20320 に制限されています。
- テーブルあたりの属性数　特定のテーブルに属すことができる属性 (つまり、カラムとインデックス) の最大数は 512 です。
- キーあたりの属性数　キーあたりの属性の最大数は 32 です。
- 行サイズ　1 行の最大許容サイズは 14000 バイトです (MySQL Cluster NDB 7.0 の時点で)。この合計のうち、個々の BLOB または TEXT カラムは 256 + 8 = 264 バイトを占めます。

18.1.6.6 MySQL Cluster の未サポート機能または欠落している機能

NDB テーブルでは、ほかのストレージエンジンでサポートされるいくつかの機能がサポートされません。MySQL Cluster でこれらの機能を使用しようとすると、それ自体ではエラーは発生しませんが、その機能がサポートまたは適用されることを予期するアプリケーションでエラーが発生する可能性があります。

- インデックスのプリフィクス　NDBCLUSTER テーブルでは、インデックスのプリフィクスはサポートされません。CREATE TABLE、ALTER TABLE、CREATE INDEX などのステートメントでインデックス指定の一部としてプリフィクスが使用された場合、そのプリフィクスは無視されます。
- セーブポイントとロールバック　セーブポイントおよびセーブポイントへのロールバックは、MyISAM と同様に無視されます。
- コミットの持続性　ディスクに対する永続的なコミットはありません。コミットはレプリケートされますが、コミット時にログがディスクにフラッシュされる保証はありません。
- レプリケーション　ステートメントベースのレプリケーションはサポートされません。クラスタのレプリケーションを設定するときは、--binlog-format=ROW (または --binlog-format=MIXED) を使用してください。詳細は、セクション18.6「MySQL Cluster レプリケーション」を参照してください。

  グローバルトランザクション ID (GTID) を使用したレプリケーションは、MySQL Cluster と互換性がなく、MySQL Cluster NDB 7.3 または MySQL Cluster NDB 7.4 ではサポートされません。MySQL Cluster レプリケーションの失敗を含む問題が発生する可能性が非常に高いため、NDB ストレージエンジンを使用するときは GTID を有効にしないでください。

  > **注記**
  >
  > NDB でのトランザクション処理に関する制限の詳細は、セクション18.1.6.3「MySQL Cluster でのトランザクション処理に関する制限」を参照してください。

18.1.6.7 MySQL Cluster のパフォーマンスに関する制限

次のパフォーマンスの問題は、MySQL Cluster に固有の問題または MySQL Cluster で特に顕著な問題です。

- Range scans　NDB ストレージエンジンへの順次アクセスによって発生するクエリーのパフォーマンスの問題があります。多数の範囲スキャンの実行も、MyISAM や InnoDB に比べて負荷が高くなります。
- Records in range の信頼性　Records in range 統計は使用可能ですが、完全なテストや正式なサポートは行われていません。このため、場合によっては最適でないクエリー計画が生成されることがあります。必要な場合は、USE INDEX または FORCE INDEX を使用して実行プランを変更できます。これを行う方法の詳細は、セクション13.2.9.3「インデックスヒントの構文」を参照してください。
- 一意のハッシュインデックス　USING HASH で作成された一意のハッシュインデックスは、NULL がキーの一部として指定された場合はテーブルへのアクセスに使用できません。

18.1.6.8 MySQL Cluster に限定された問題

NDBCLUSTER ストレージエンジンに固有の制限を次に示します。

- マシンアーキテクチャー　クラスタ内で使用されるすべてのマシンが同じアーキテクチャーである必要があります。つまり、ノードをホストするすべてのマシンがビッグエンディアンまたはリトルエンディアンのどちらかである必要があり、両者を組み合わせて使用することはできません。たとえば、PowerPC で実行されている管理ノードから x86 マシンで実行されているデータノードに指示することはできません。この制限は、クラスタの SQL ノードにアクセスする可能性がある mysql またはその他のクライアントを実行するだけのマシンには適用されません。
- バイナリロギング
  MySQL Cluster には、バイナリロギングに関して次の制限または規制があります。
  - sql_log_bin は、データ操作では効果がありませんが、スキーマ操作ではサポートされます。
  - MySQL Cluster は、BLOB カラムがあっても主キーがないテーブルのバイナリログを生成できません。
  - ステートメントを実行する mysqld に配置されていないクラスタバイナリログには、次のスキーマ操作だけが記録されます。
    - CREATE TABLE
    - ALTER TABLE
    - DROP TABLE
    - CREATE DATABASE / CREATE SCHEMA
    - DROP DATABASE / DROP SCHEMA
    - CREATE TABLESPACE
    - ALTER TABLESPACE
    - DROP TABLESPACE
    - CREATE LOGFILE GROUP
    - ALTER LOGFILE GROUP
    - DROP LOGFILE GROUP

セクション18.1.6.10「複数の MySQL Cluster ノードに関する制限」も参照してください。

18.1.6.9 MySQL Cluster のディスクデータストレージに関する制限

ディスクデータオブジェクトの最大数と最小数　ディスクデータオブジェクトには、次の最大数および最小数が適用されます。

- テーブルスペースの最大数: 2^{32} (4294967296)
- テーブルスペースあたりのデータファイルの最大数: 2^{16} (65536)
- データファイルの最大サイズ: 理論的な限界は 64G ですが、実際的な上限は 32G です。これは、1M のエクステント 32768 個分に相当します。

MySQL Cluster のディスクデータテーブルは最大 1 つのテーブルスペースを使用できます。つまり、1 つの NDB テーブルによってディスク上に格納できるデータ量 (バイト単位) の理論的な上限は 32G * 65536 = 2251799813685248 (およそ 2 ペタバイト) です。

- テーブルスペースのデータファイルあたりのエクステントの理論的な最大数は 2^{16} (65536) です。ただし、実際的に推奨されるデータファイルあたりのエクステントの最大数は 2^{15} (32768) です。

  テーブルスペースのデータファイルのエクステントが取り得る最小および最大サイズは、それぞれ 32K および 2G です。詳細は、セクション13.1.18「CREATE TABLESPACE 構文」を参照してください。

ディスクデータテーブルとディスクレスモード　クラスタをディスクレスモードで実行しているときは、ディスクデータテーブルを使用できません。

18.1.6.10 複数の MySQL Cluster ノードに関する制限

複数の SQL ノード
複数の MySQL サーバーを MySQL Cluster の SQL ノードとして使用することに関する NDBCLUSTER ストレージエンジン固有の問題を次に示します。

- 分散型のテーブルロックがない　LOCK TABLES は、ロックが発行された SQL ノードに対してのみ機能します。クラスタ内のほかの SQL ノードでは、このロックは「認識」されません。これは、操作の一部としてテーブルをロックするステートメントによって発行されたロックにも当てはまります。(例については、次の項目を参照してください。)

- ALTER TABLE 操作　複数の MySQL サーバー (SQL ノード) が実行されているときは、ALTER TABLE によるロックは完全ではありません。(前の項目で説明したように、MySQL Cluster は分散型のテーブルロックをサポートしません。)

複数の管理ノード
複数の管理サーバーを使用する場合:

- ノード ID の自動割り当てが複数の管理サーバー間で機能しないため、接続文字列にノードの明示的な ID を指定する必要があります。

- 管理サーバーを起動すると、まず同じ MySQL Cluster 内にほかの管理サーバーがあるかどうかがチェックされ、ほかの管理サーバーへの接続に成功すると、その構成データが使用されます。つまり、管理サーバーの --reload および --initial 起動オプションは、それが実行中の唯一の管理サーバーでないかぎり、無視されます。また、複数の管理ノードを含む MySQL Cluster のローリング再起動を実行すると、管理サーバーはその MySQL Cluster で実行中の唯一の管理サーバーである場合にかぎって、それ自身の構成ファイルを読み取ります。詳細は、セクション18.5.5「MySQL Cluster のローリング再起動の実行」を参照してください。

複数のネットワークアドレス　1 つのデータノードに対する複数のネットワークアドレスはサポートされません。これらを使用すると、問題が発生しやすくなります。データノードに障害が発生すると、SQL ノードはデータノードが停止したことの確認を待機しますが、そのデータノードへの別のルートが開いたままであるため、この確認を受信できません。これにより、クラスタは実質的に操作不能になります。

> **注記**
>
> 1 つのデータノードで複数のネットワークハードウェアインタフェース (Ethernet カードなど) を使用できますが、これらは同じアドレスにバインドする必要があります。これは、config.ini ファイルで接続ごとに複数の [tcp] セクションを使用できないことも意味します。詳細は、セクション18.3.2.8「MySQL クラスタの TCP/IP 接続」を参照してください。

18.1.6.11 MySQL Cluster NDB 7.3 で解決された以前の MySQL Cluster の問題

MySQL Cluster NDB 7.3 では、MySQL Cluster の以前のバージョンに存在したいくつかの制限および関連する問題が解決されました。これらについて、次のリストで簡単に説明します。

- 外部キーのサポート　NDB テーブルの外部キー制約が、InnoDB ストレージエンジンでのサポートと同様の方法でサポートされました。

  > **注記**
  >
  > ユーザーパーティション化された InnoDB テーブルの場合とは異なり、KEY または LINEAR KEY によってパーティション化された NDB テーブルで外部キーがサポートされます。

MySQL での外部キーのサポートの詳細は、セクション1.7.3.2「FOREIGN KEY の制約」を参照してください。MySQL でサポートされる外部キーの構文の詳細は、セクション13.1.17.2「外部キー制約の使用」を参照してください。

18.2 MySQL Cluster のインストール

このセクションでは、MySQL Cluster の計画、インストール、構成、および実行の基本について説明します。セクション18.3「MySQL Cluster の構成」の例ではクラスタリングのさまざまなオプションと構成について詳しく説明しますが、ここで概説するガイドラインと手順に従うことで、データの可用性と保護に関する最小要件を満たす使用可能な MySQL Cluster が構築されます。

MySQL Cluster のリリースバージョンでのアップグレードまたはダウングレードについては、セクション18.2.8「MySQL Cluster NDB 7.3 のアップグレードとダウングレード」を参照してください。

このセクションでは、ハードウェアとソフトウェアの要件、ネットワークの問題、MySQL Cluster のインストール、基本的な構成の問題、クラスタの開始、停止、および再起動、サンプルデータベースのロード、およびクエリーの実行について説明します。

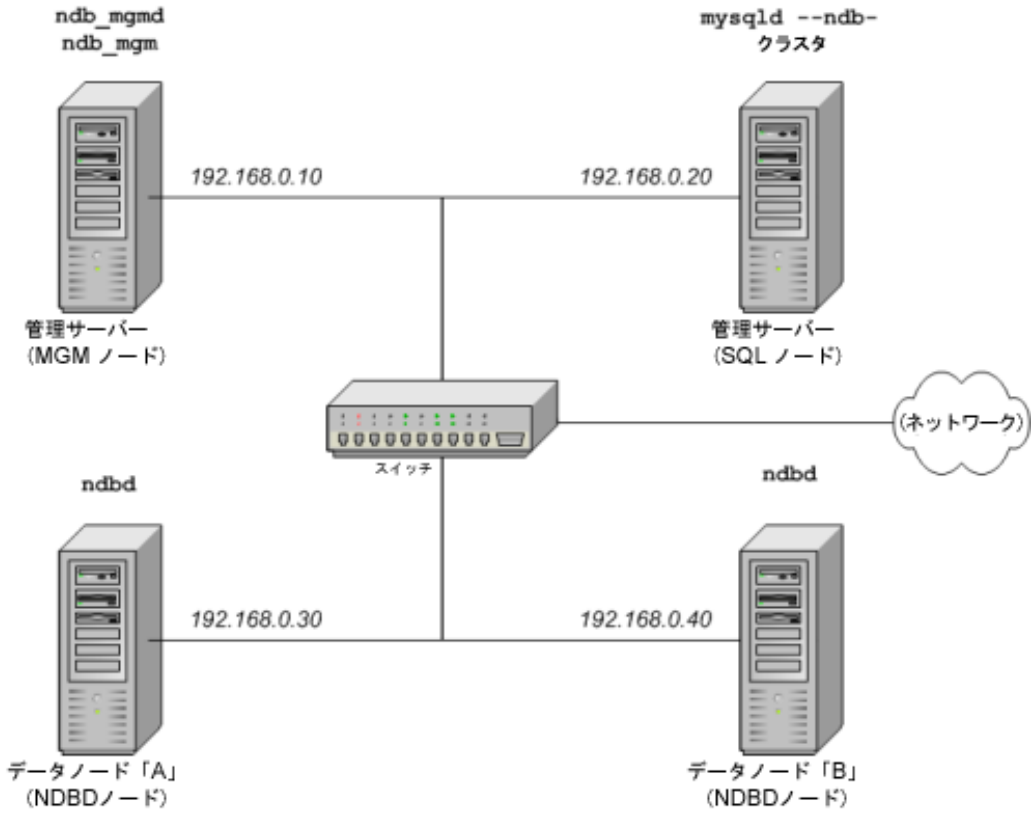
MySQL Cluster NDB 7.3 以降では、Web ベースのグラフィカルインストーラである MySQL Cluster Auto-Installer が MySQL Cluster 配布の一部として提供されます。Auto-Installer を使用すると、1 台 (テスト用) または複数のホストコンピュータに対して MySQL Cluster の基本的なインストールとセットアップを実行できます。詳細は、セクション18.2.1「MySQL Cluster Auto-Installer」を参照してください。

仮定　以降のセクションでは、クラスタの物理的構成とネットワーク構成に関していくつかの仮定を立てています。これらの仮定については、次のいくつかの段落で説明します。

クラスタノードとホストコンピュータ　このクラスタは、ここに示す 4 つのノードで構成されます。各ノードは別個のホストコンピュータ上に配置され、一般的な Ethernet ネットワーク上に固定のネットワークアドレスを持っています。

ノード	IP アドレス
管理ノード (mgmd)	192.168.0.10
SQL ノード (mysqld)	192.168.0.20
データノード "A" (ndbd)	192.168.0.30
データノード "B" (ndbd)	192.168.0.40

これをより明確に表したものが次の図です。



ネットワークアドレス設定　　簡略化 (および信頼性) のため、この操作手順では数値の IP アドレスのみを使用します。ただし、ネットワーク上で DNS 解決が利用可能な場合は、クラスタ構成時に IP アドレスの代わりにホスト名を使用できます。また、hosts ファイル (通常、Linux およびその他の Unix 系オペレーティングシステムでは /etc/hosts、Windows では C:\WINDOWS\system32\drivers\etc\hosts、または使用しているオペレーティングシステムの同等のファイル) が使用可能な場合は、ホスト検索を行う手段として使用することもできます。

hosts ファイルの潜在的な問題　　クラスタノードにホスト名を使用しようとしたときによくある問題は、一部のオペレーティングシステム (一部の Linux 配布を含む) がインストール中にシステム独自のホスト名を /etc/hosts に設定する方法が原因で発生します。ndb1 および ndb2 というホスト名を持つ 2 台のマシンがどちらも cluster ネットワークドメインに含まれる場合について考えます。Red Hat Linux (CentOS や Fedora などの一部の派生バージョンを含む) では、これらのマシンの /etc/hosts ファイルに次のエントリが設定されます。

```
# ndb1 /etc/hosts:
127.0.0.1   ndb1.cluster ndb1 localhost.localdomain localhost
```

```
# ndb2 /etc/hosts:
127.0.0.1   ndb2.cluster ndb2 localhost.localdomain localhost
```

SUSE Linux (OpenSUSE を含む) では、マシンの /etc/hosts ファイルにこれらのエントリが設定されます。

```
# ndb1 /etc/hosts:
127.0.0.1       localhost
127.0.0.2       ndb1.cluster ndb1
```

```
# ndb2 /etc/hosts:
127.0.0.1       localhost
127.0.0.2       ndb2.cluster ndb2
```

どちらの場合も、ndb1 は ndb1.cluster をループバック IP アドレスにルーティングしますが、DNS から ndb2.cluster のパブリック IP アドレスを取得します。一方、ndb2 は ndb2.cluster をループバックアドレスにルーティングし、ndb1.cluster のパブリックアドレスを取得します。その結果、各データノードは管理サーバーに接続しますが、ほかのデータノードが接続したことを検出できないため、データノードが起動中にハングアップしたように見えます。

> **注意**
>
> config.ini では localhost とほかのホスト名または IP アドレスを混在できません。これらの理由により、このようなケースの (config.ini のすべての HostName エントリで IP ア

ドレスを使用する以外の) 解決策は、すべてのクラスタホストの /etc/hosts から完全修飾ホスト名を削除し、config.ini で使用することです。

ホストコンピュータのタイプ　このインストールシナリオに含まれる各ホストコンピュータは、標準的な構成でディスクにインストールされたサポート対象のオペレーティングシステムを実行し、不要なサービスを実行していない Intel ベースのデスクトップ PC です。標準の TCP/IP ネットワーク機能を含む中核的なオペレーティングシステムで十分です。また、簡略化のため、すべてのホストのファイルシステムが完全に同じように設定されていると仮定します。そうでない場合は、状況に応じてこれらの手順を適用してください。

ネットワークハードウェア　各マシンには標準の 100M ビット/秒または 1 ギガビット Ethernet カードが (カードに対応するドライバとともに) 取り付けられ、4 台のホストすべてがスイッチなどの標準仕様の Ethernet ネットワークアプライアンスを介して接続されています。(すべてのマシンで同じスループットのネットワークカードを使用してください。つまり、クラスタ内の 4 台のマシンで 100M ビット/秒カードを使用するか、または 4 台のマシンで 1 ギガビットカードを使用してください。)MySQL Cluster は 100M ビット/秒のネットワークで動作しますが、ギガビット Ethernet ではパフォーマンスがさらに向上します。

重要

MySQL Cluster は、スループットが 100M ビット/秒未満のネットワークや長い待機時間が発生するネットワークで使用できるように設計されていません。特にこの理由により、インターネットなどの広域ネットワークを介して MySQL Cluster を実行することは、成功する可能性が低く、本番環境ではサポートされていません。

サンプルデータ　ここでは、MySQL Web サイトからダウンロードできる world データベースを使用します (https://dev.mysql.com/doc/index-other.html を参照してください)。ここでは、オペレーティングシステムと必要な MySQL Cluster プロセスを実行し、(データノードで) データベースを格納するための十分なメモリーが各マシンにあると仮定します。

MySQL のインストールに関する一般的な情報は、第2章「MySQL のインストールと更新」を参照してください。Linux およびその他の Unix 系オペレーティングシステムに対する MySQL Cluster のインストールについては、セクション18.2.2「Linux での MySQL Cluster のインストール」を参照してください。Windows オペレーティングシステムに対する MySQL Cluster のインストールについては、セクション18.2.3「Windows での MySQL Cluster のインストール」を参照してください。

MySQL Cluster のハードウェア、ソフトウェア、およびネットワーク要件に関する一般的な情報は、セクション18.1.3「MySQL Cluster のハードウェア、ソフトウェア、およびネットワーク要件」を参照してください。

18.2.1 MySQL Cluster Auto-Installer

- セクション18.2.1.1「MySQL Cluster Auto-Installer の要件」
- セクション18.2.1.2「MySQL Cluster Auto-Installer の概要」
- セクション18.2.1.3「MySQL Cluster Auto-Installer の使用」

このセクションでは、MySQL Cluster NDB 7.3.1 から MySQL Cluster 配布の一部として組み込まれた Web ベースのグラフィカル構成インストーラについて説明します。説明するトピックには、インストーラとその各部分の概要、インストーラを実行するためのソフトウェアおよびその他の要件、GUI の操作、およびインストーラを使用して 1 台以上のホストコンピュータに MySQL Cluster をセットアップし、起動または停止する方法が含まれます。

18.2.1.1 MySQL Cluster Auto-Installer の要件

このセクションでは、サポートされるオペレーティングプラットフォームとソフトウェア、必要なソフトウェア、および MySQL Cluster Auto-Installer を実行するためのその他の前提条件について説明します。

サポートされるプラットフォーム　MySQL Cluster Auto-Installer は、Linux、Windows、Solaris、および MacOS X の最近のバージョンに対応する MySQL Cluster NDB 7.3 以降のほとんどの配布で使用できます。MySQL Cluster および MySQL Cluster Auto-Installer のプラットフォームサポートの詳細は、https://www.mysql.com/support/supportedplatforms/cluster.html を参照してください。

サポートされる Web ブラウザ　この Web ベースのインストーラは、Firefox および Microsoft Internet Explorer の最近のバージョンでサポートされます。また、Opera、Safari、および Chrome の最近のバージョンでも動作しますが、これらのブラウザとの互換性に関する詳細なテストは行われていません。

必要なソフトウェア (セットアップホスト)　Auto-Installer を実行するホストには、次のソフトウェアがインストールされている必要があります。

- Python 2.6 以降　Auto-Installer には、Python インタプリタおよび標準ライブラリが必要です。これらがシステムにまだインストールされていない場合は、システムのパッケージマネージャーを使用して追加できる可能性があります。そうでない場合は、http://python.org/download/ からダウンロードできます。

- Paramiko 1.7.7.1 以降　これは、SSH を使用してリモートホストと通信するために必要です。http://www.lag.net/paramiko/ からダウンロードできます。Paramiko は、使用しているシステムのパッケージマネージャーから入手することもできます。

- Pycrypto バージョン 2.6 以降　この暗号化モジュールは Paramiko が必要とします。使用しているシステムのパッケージマネージャーを使用して入手できない場合は、https://www.dlitz.net/software/pycrypto/ からダウンロードできます。

構成ツールの Windows バージョンには、前のリストに示したすべてのソフトウェアが含まれているため、別個にインストールする必要はありません。

Paramiko および Pycrypto ライブラリは、MySQL Cluster ノードをリモートホストに配備する場合にのみ必要です。インストーラを実行する同じホストにすべてのノードを配置する場合は必要ありません。

必要なソフトウェア (リモートホスト)　MySQL Cluster ノードを配備するリモートホストに必要な唯一のソフトウェアは、Linux および Solaris システムでは通常デフォルトでインストールされる SSH サーバーです。Windows では、いくつかの代わりとなるものが使用可能です。概要については、http://en.wikipedia.org/wiki/Comparison_of_SSH_servers を参照してください。

複数のホストを使用するときの追加要件は、次の段落で説明するように、SSH と適切な鍵またはユーザー資格証明を使用してリモートホストで認証できるようにすることです。

認証とセキュリティー　Auto-Installer では、リモートアクセスに対してここに示す 3 つの基本的なセキュリティーまたは認証メカニズムを使用できます。

- SSH　セキュアシェル接続を使用すると、バックエンドからリモートホストでのアクションを実行できるようになります。そのためには、リモートホストで SSH サーバーが実行されている必要があります。また、インストーラを実行するシステムユーザーがユーザー名とパスワードまたは公開鍵と秘密鍵を使用してリモートサーバーにアクセスできる必要があります。

  > **重要**
  >
  > システムの root アカウントは安全性がきわめて低いため、リモートアクセスには決して使用しないでください。また、mysqld はシステムの root によって正常に起動できません。これらの理由やその他の理由から、システムの root ではく、ターゲットシステム上の通常のユーザーアカウントの SSH 資格証明を提供してください。この問題の詳細は、セクション6.1.5「MySQL を通常ユーザーとして実行する方法」を参照してください。

- HTTPS　Web ブラウザのフロントエンドとバックエンド間のリモート通信は、デフォルトでは暗号化されません。これは、ユーザーの SSH パスワードなどの情報がすべてのユーザーに判読可能な平文で転送されることを意味します。リモートクライアントからの通信を暗号化するには、バックエンドに証明書を用意し、フロントエンドが HTTP ではなく HTTPS を使用してバックエンドと通信する必要があります。HTTPS を有効にするには、自己署名証明書を発行するのがもっとも簡単です。証明書を発行したあとは、それが使用されていることを確認する必要があります。これを行うには、コマンド行から --use-https および --cert-file オプションを指定して ndb_setup.py コマンドを起動します。

- 証明書ベースの認証　バックエンドの ndb_setup.py プロセスは、ローカルホストだけでなくリモートホストに対してもコマンドを実行できます。これは、バックエンドに接続した任意のユーザーがコマンドの実行方法を管理できることを意味します。バックエンドへの不要な接続を拒否するには、クライアントを認証するための証明書が必要です。この場合は、証明書がユーザーによって発行され、ブラウザにインストールされ、バックエンドで認証に使用できるようになっている必要があります。この要件を (パスワードまたは鍵による認証と組み合わせて、またはその代わりに) 設定するには、--ca-certs-file オプションを指定して ndb_setup.py を起動します。

クライアントブラウザを Auto-Installer のバックエンドと同じホストで実行する場合は、セキュアな認証の必要性または要件はありません。

MySQL のより一般的なセキュリティー情報については、セクション18.5.11「MySQL Cluster のセキュリティー上の問題」(MySQL Cluster を配備するときに考慮に入れるセキュリティーの検討事項について説明しています) および第6章「セキュリティー」も参照してください。

18.2.1.2 MySQL Cluster Auto-Installer の概要

MySQL Cluster Auto-Installer は 2 つのコンポーネントで構成されます。フロントエンドは、Firefox や Microsoft Internet Explorer などの標準的な Web ブラウザ内でロードおよび実行される Web ページとして実装された GUI クライアントです (セクション18.2.1.1「MySQL Cluster Auto-Installer の要件」を参照してください)。バックエンドは、ローカルマシンまたはユーザーがアクセスできる別のホストで実行されるサーバープロセス (ndb_setup.py) です。

これら 2 つのコンポーネント (クライアントとサーバー) は、標準の HTTP 要求および応答を使用して相互に通信します。バックエンドでは、バックエンドのユーザーがアクセスを許可された任意のホスト上の MySQL Cluster ソフトウェアプログラムを管理できます。MySQL Cluster ソフトウェアが別のホスト上にある場合、バックエンドは SSH を利用してアクセスし、Paramiko ライブラリを使用してリモートでコマンドを実行します (セクション18.2.1.1「MySQL Cluster Auto-Installer の要件」を参照してください)。

このセクションの残りの部分では、主に Web クライアントについて説明します。コマンド行ツールの使用方法の詳細は、セクション18.4.23「ndb_setup.py — MySQL Cluster のブラウザベース自動インストーラの開始」を参照してください。

MySQL Cluster Auto-Installer のインタフェース　このセクションでは、MySQL Cluster Auto-Installer のレイアウトとナビゲーションについて説明します。Auto-Installer を Web ブラウザで最初に開くと、ここに示すような「ようこそ」画面が表示されます。

インストーラの UI にアクセスするには、「Create New MySQL Cluster」または「Continue Previous Cluster Configuration」のいずれかのオプションを選択します。Auto-Installer の一般的な画面には、次の要素が含まれています。

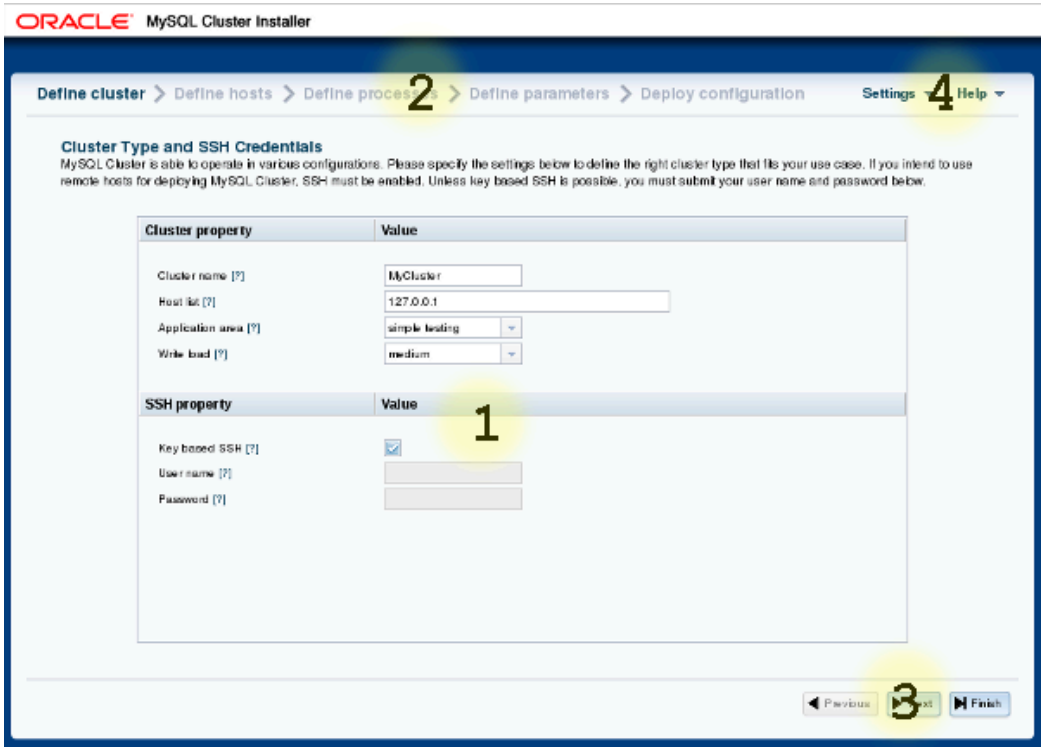
1. 表示パネル　構成設定に関するデータとそれらを変更するためのコントロールが表示される中央の領域です。

2. ブレッドクラムナビゲーション　GUI の左上および中央上部にあるブレッドクラムナビゲーションバーは、MySQL Cluster の構成の各ステップに対応する画面にリンクする一連のタイトルで構成されます。ブレッドクラムを使用すると、これらの段階の間を任意の順序でジャンプできます。

3. 順次ナビゲーション　これは、「Previous」、「Next」、および「Finished」というラベルが付いた一連のボタンで構成され、GUI の右下隅にあります。順次ナビゲーションは、指示された順序でステップ間を移動するために使用します。

4. 「設定」および「ヘルプ」メニュー　これらのメニューは、GUI の右上隅 (ブレッドクラムナビゲーションバーの右側) にあります。「設定」では、Auto-Installer の構成設定を確認 (場合によっては変更) できます。「ヘルプ」は、インストーラの組み込みヘルプファイルにアクセスするために使用できます。

この図は、前述の各要素の場所を Auto-Installer の一般的なページで示したものです。図に重ね合わせた番号は、前のリストの番号に対応しています。

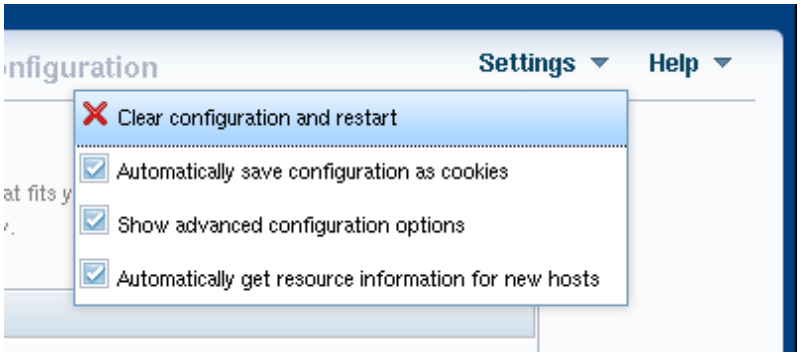


このセクションの残りの部分では、表示パネルを除くすべての要素について詳しく説明します。セクション 18.2.1.3「MySQL Cluster Auto-Installer の使用」では、表示領域に表示されるパネルについて、各パネルの機能とそこに含まれるコントロールとともに説明します。

任意および順次ナビゲーション　Auto-Installer には、MySQL Cluster 配備のセットアップと構成の異なる段階をカバーするページのいずれかが表示されます。ページ間を移動するには、次の 2 つの方法のいずれかを使用します。1 つ目は、さまざまなページのタイトルを表示するブレッドクラムトレイルナビゲーションツールバーです(現在のページのタイトルは強調表示され、無効になっています)。この中から目的のページのタイトルを選択すると、任意のページに任意の順序で移動できます。このツールバーを次に示します。

Auto-Installer に用意されている 2 つ目のナビゲーションメカニズムは、ページの右下にある順次ナビゲーションボタン「Next」、「Previous」、および「Finish」で構成されます。これを使用すると、特定の順序で次または前のページに移動したり、最後のページに移動したりできます。このボタンは必要に応じて有効および無効にして、最後のページをとばして進めないようにしたりできます。

「設定」および「ヘルプ」メニュー　これらのメニューは、このセクションの冒頭で説明したように、GUI の右上隅に並んで配置されています。「設定」メニューの詳細を次に示します。



次のリストでは、「設定」メニューのエントリについて説明します。

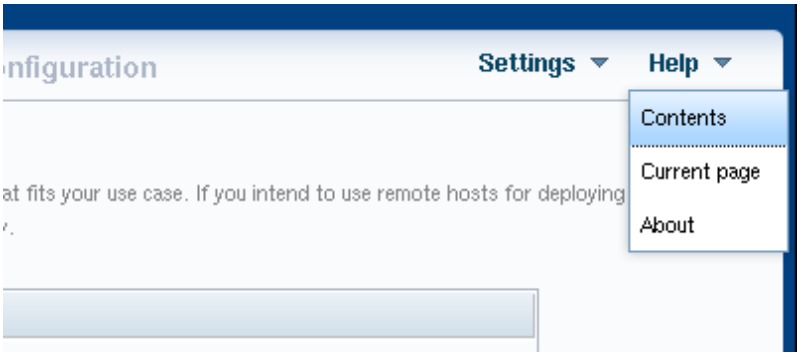
- 「Clear configuration and restart」: すべてのホストおよびプロセスを削除し、すべてのパラメータ値をデフォルトに戻し、インストーラを最初のページからやり直します。
- 「Automatically save configuration as cookies」: 構成情報 (ホスト名、プロセスデータ、パラメータ値など) をブラウザの Cookie として保存します。このオプションを選択すると、SSH パスワードを除くすべての情報が保存されます。つまり、ブラウザを終了して再起動したときに、前のセッションと同じ構成作業を最後に中止したポイントから続行できます。

  SSH パスワードは保存されないため、使用している場合は、新しいセッションの開始時にもう一度入力する必要があります。
- 「Show advanced configuration options」: Auto-Installer に高度な構成パラメータを表示し、ユーザーが設定できるようにします。

  高度なパラメータは、一度設定すると、明示的に変更またはリセットするまで構成ファイル内で使用され続けます。これは、高度なパラメータが現在インストーラに表示されているかどうかは関係ありません。メニュー項目を無効にしても、パラメータの値はリセットされません。
- 「Automatically get resource information for new hosts」: 新しいホストのハードウェアリソース情報を自動的にクエリーして、構成オプションおよび値を事前に移入します。この場合、提案された値は必須ではありませんが、インストーラの対応する編集オプションを使用して明示的に変更しないかぎり、その値が使用されます。

インストーラのナビゲーション要素と同様に、ユーザーがそれまでに行なった選択に応じて、「設定」メニューの 1 つ以上のエントリが無効になることがあります。

展開時の「ヘルプ」メニューを次に示します。



「ヘルプ」メニューには、次のリストで説明する複数のオプションがあります。

- 「Content」: 組み込みのユーザーガイドを表示します。これは、別のブラウザウィンドウで開くため、ワークフローを中断せずにインストーラと同時に使用できます。
- 「現在のページ」: 組み込みのユーザーガイドの、インストーラに現在表示されているページについて説明するセクションを開きます。
- 「この製品について」: ここに示すように、インストーラの名前とこのインストーラが提供される MySQL Cluster 配布のバージョン番号が表示された小さいダイアログを表示します。

Auto-Installer では、ほとんどの入力ウィジェットでツールチップ形式のコンテキスト依存ヘルプも提供されます。これらのツールチップの 1 つは、ウィジェットやウィジェットラベルの横に表示されることがある小さい疑問符にマウスを合わせたときに表示されます。

また、MySQL Cluster の構成パラメータの名前は MySQL Cluster のオンラインドキュメント内の対応する説明にリンクされているため、特定のパラメータの名前をクリックすると、そのパラメータのドキュメントが別のウィンドウに表示されます。

18.2.1.3 MySQL Cluster Auto-Installer の使用

- MySQL Cluster Auto-Installer の起動
- MySQL Cluster Auto-Installer の「ようこそ」画面
- MySQL Cluster Auto-Installer の「Define Cluster」画面
- MySQL Cluster Auto-Installer の「Define Hosts」画面
- MySQL Cluster Auto-Installer の「Define Processes」画面
- MySQL Cluster Auto-Installer の「Define Attributes」画面
- MySQL Cluster Auto-Installer の「Deploy Cluster」画面

MySQL Cluster Auto-Installer は複数のページで構成され、各ページは MySQL Cluster を構成して配備するために使用されるプロセス内のステップに対応しています。各ページについて次に示します。

- 「ようこそ」: 新しい MySQL Cluster の構成または既存の MySQL Cluster 構成の続行のいずれかを選択して、Auto-Installer の使用を開始します。
- 「Define Cluster」: クラスタ全体の基本的な情報 (名前、ホスト、負荷タイプなど) を設定します。ここでは、必要に応じてリモートホストへのアクセスに使用する SSH 認証のタイプを設定することもできます。
- 「Define Hosts」: MySQL Cluster プロセスを実行するホストを特定します。
- 「Define Processes」: 各クラスタホストに特定のタイプのプロセスを 1 つ以上割り当てます。
- 「Define Attributes」: プロセスまたはプロセスタイプの構成属性を設定します。
- 「Deploy Cluster」: 前に設定された構成でクラスタを配備します。また、配備されたクラスタを開始および停止します。

次のセクションでは、これらの各ページの目的と機能について、リストに示した順に詳しく説明します。

MySQL Cluster Auto-Installer の起動

Auto-Installer は、MySQL Cluster ソフトウェアとともに提供されます。(セクション18.2「MySQL Cluster のインストール」を参照してください。)このセクションでは、インストーラの起動方法について説明します。起動するには、ndb_setup.py 実行可能ファイルを起動します。ndb_setup.py は、MySQL Cluster インストールディレクトリ内の bin にあります。一般的な場所は、Linux システムでは /usr/local/mysql/bin、Windows システムでは C:\Program Files\MySQL\MySQL Server 5.6\bin ですが、これは MySQL Cluster ソフトウェアをインストールしたシステム上の場所によって異なる可能性があります。

Windows では、MySQL Cluster インストールディレクトリ内の setup.bat を実行してインストーラを起動することもできます。コマンド行から起動するときは、ndb_setup.py と同じオプションを指定できます。

ndb_setup.py は、その動作に影響を与える複数のオプションを指定して起動できますが、通常はデフォルト設定を使用するだけで十分です。その場合は、次の 2 つの方法で ndb_setup.py を起動できます。

1. 端末で MySQL Cluster の bin ディレクトリに移動し、このようにコマンド行から引数やオプションを追加せずに起動します。

   ```
   shell> ndb_setup
   ```

   これは、オペレーティングシステムに関係なく有効です。

2. ファイルブラウザ (Windows の Windows Explorer、Linux の Konqueror、Dolphin、Nautilus など) で MySQL Cluster の bin ディレクトリに移動し、ndb_setup.py ファイルアイコンを (通常はダブルクリックによって) アクティブ化します。これは Windows で有効ですが、ほとんどの一般的な Linux デスクトップでも有効です。

   Windows では、MySQL Cluster インストールディレクトリに移動して、setup.bat ファイルアイコンをアクティブ化することもできます。

どちらの場合も、ndb_setup.py を起動すると、システムのデフォルトの Web ブラウザで Auto-Installer の「ようこそ」画面が開きます。

場合によっては、Auto-Installer に含まれている Web サーバーの実行用として異なるポートを指定するなど、インストーラのデフォルト以外の設定を使用する必要があります。その場合は、必要なデフォルトをオーバーライ

ドする値を含む起動オプションを指定して ndb_setup.py を起動する必要があります。Windows システムで同じ起動オプションを使用するには、MySQL Cluster ソフトウェア配布でそのようなプラットフォーム用に提供されている setup.bat ファイルを使用します。これはコマンド行を使用して実行できますが、これらのオプションを 1 つ以上使用しながらデスクトップまたはファイルブラウザからインストーラを起動する必要がある場合は、適切な起動を含むスクリプトまたはバッチファイルを作成し、ファイルブラウザでそのファイルアイコンをダブルクリックしてインストーラを起動することもできます。(Linux システムでは、最初にスクリプトファイルを実行可能にする必要もあります。)MySQL Cluster Auto-Installer の高度な起動オプションについては、セクション 18.4.23「ndb_setup.py — MySQL Cluster のブラウザベース自動インストーラの開始」を参照してください。

MySQL Cluster Auto-Installer の「ようこそ」画面

「ようこそ」画面は、ここに示すように、ndb_setup.py を起動したときにデフォルトのブラウザにロードされます。

この画面には、インストーラに入るための選択肢として次の 2 つが表示されます。続行するには、そのいずれかを選択する必要があります。

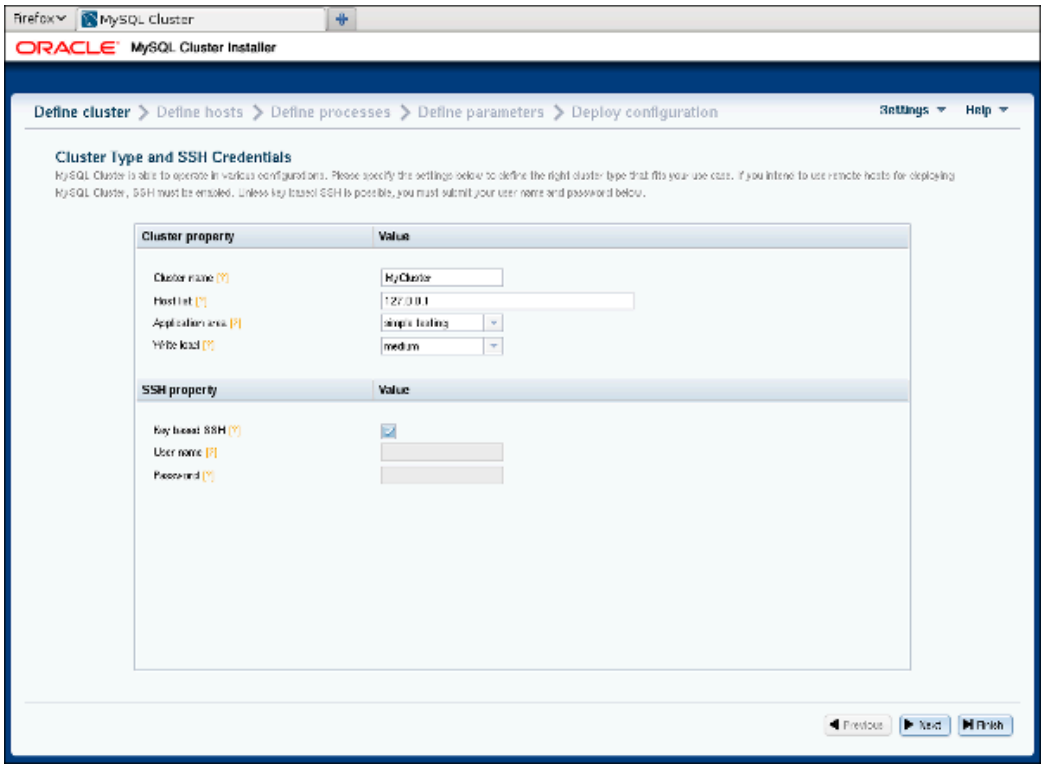
1. 「Create New MySQL Cluster」: 完全に新しいクラスタをセットアップして配備するために Auto-Installer を起動します。

2. 「Continue Previous Cluster Configuration」: 前のセッションが終了した同じポイントから、前の設定をすべて保持した状態で Auto-Installer を起動します。

2 つ目のオプションでは、ブラウザが前のセッションの Cookie にアクセスできる必要があります。これらの Cookie は、セッション中に生成された構成やその他の情報を格納するメカニズムとして使用されています。つまり、Auto-Installer で前のセッションを続行するには、前のセッションと同じホストで実行される同じブラウザを使用する必要があります。

MySQL Cluster Auto-Installer の「Define Cluster」画面

「Define Cluster」画面は、「ようこそ」画面での選択に続いて表示される最初の画面で、クラスタの一般的なプロパティーを設定するために使用されます。「Define Cluster」画面のレイアウトを次に示します。



「Define Cluster」画面では、このリストで説明するクラスタの一般的なプロパティーを設定できます。

- 「Cluster name」: クラスタを識別する名前。デフォルトは MyCluster です。
- 「Host list」: クラスタプロセスを実行する 1 台以上のホストのカンマ区切りリスト。デフォルトでは、これは 127.0.0.1 です。このリストにリモートホストを追加した場合は、指定した「SSH Credentials」を使用して接続できる必要があります。
- 「Application type」: 次のいずれかを選択します。

  1. 「Simple testing」: 小規模なテストに対応する最小限のリソース使用。これはデフォルトです。本番環境を目的にしたものではありません。
  2. 「Web」: 特定のハードウェアに合わせてパフォーマンスを最大化します。
  3. 「Real-time」: 障害の発生したクラスタプロセスの検出にかかる時間を最小限に抑えるため、タイムアウトへの感度を最大限にしながらパフォーマンスを最大化します。

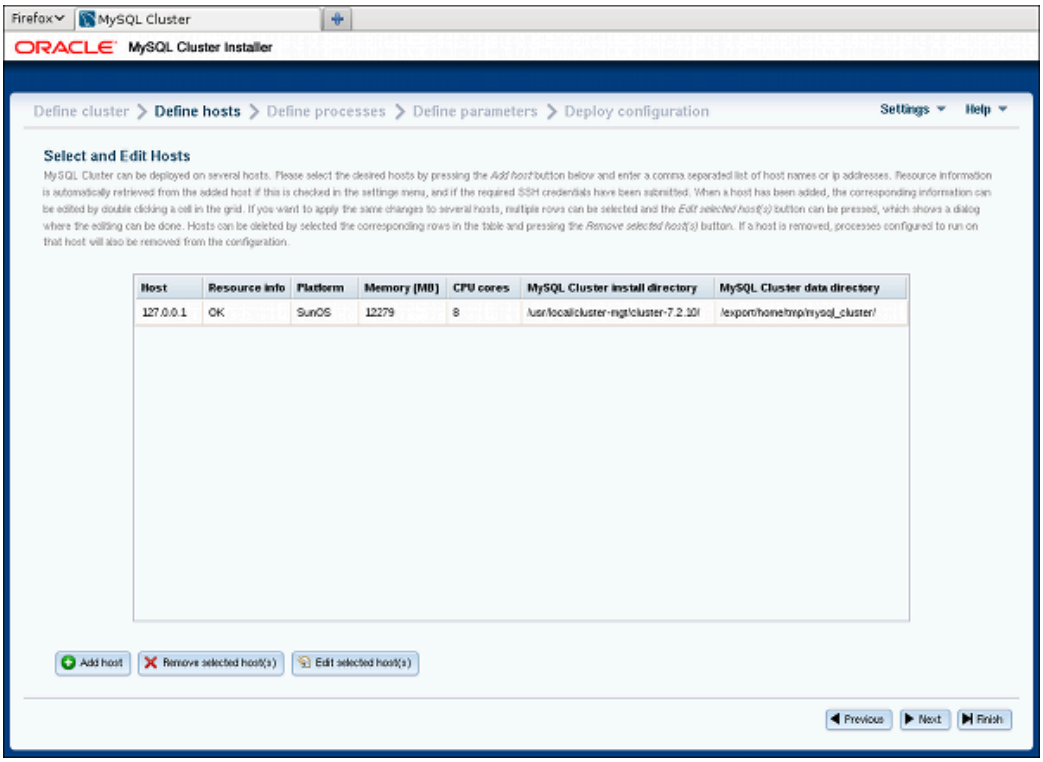
- 「Write load」: クラスタ全体で予測される書き込み数のレベルを選択します。次のいずれかのレベルを選択できます。

  1. 「Low」: 予想される負荷に、1 秒あたり 100 件未満の書き込みトランザクションが含まれます。
  2. 「Medium」: 予想される負荷に、1 秒あたり 100-1000 件の書き込みトランザクションが含まれます。
  3. 「High」: 予想される負荷に、1 秒あたり 1000 件を超える書き込みトランザクションが含まれます。

- 「SSH Credentials」: 「Key-Based SSH」を選択するか、「User」および「Password」資格証明を入力します。この SSH 鍵またはユーザー名とパスワードは、「Host list」に指定したリモートホストに接続するときに必要になります。デフォルトでは、「Key-Based SSH」が選択され、「User」および「Password」フィールドは空になっています。

MySQL Cluster Auto-Installer の「Define Hosts」画面

ここに示す「Define Hosts」画面では、各クラスタホストのいくつかの主要なプロパティーを表示および指定できます。



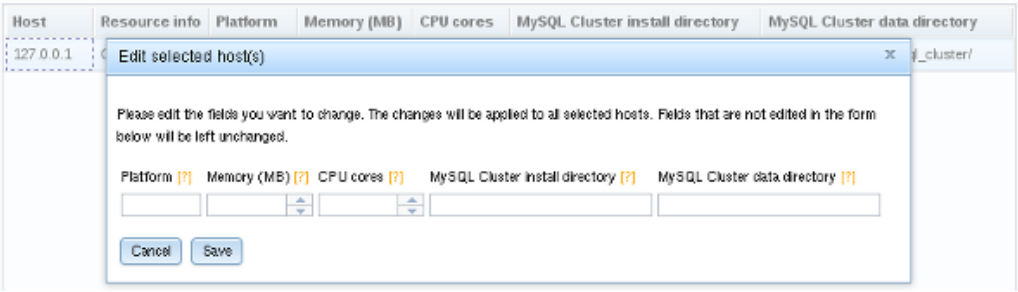
現在入力されているホストが各種の情報とともにグリッドに表示されます。ホストを追加するには、「Add hosts」ボタンをクリックし、(「Define Cluster」画面のホストリストを編集するときと同じように) カンマで区切られた 1 つ以上のホスト名、IP アドレス、またはその両方のリストを入力します。

同様に、「Remove selected host(s)」というラベルのボタンを使用して 1 台以上のホストを削除できます。この方法でホストを削除すると、そのホストに構成されたプロセスもすべて削除されます。

「設定」メニューで「Automatically get resource information for new hosts」にチェックマークを付けると、Auto-Installer はプラットフォーム名、メモリーの量、および CPU コアの数を取得して、それらを自動的に設定しようとします。このステータスは「Resource info」カラムに表示されます。リモートホストからの情報は即座にフェッチできず、特に Windows を実行しているホストからは、ある程度時間がかかることがあります。

「Define Cluster」画面の SSH ユーザー資格情報を変更すると、このツールは情報が欠落しているホストのハードウェア情報をリフレッシュしようとします。ただし、特定のフィールドがすでに編集されている場合、ユーザーが指定した情報がホストからフェッチされた値で上書きされることはありません。

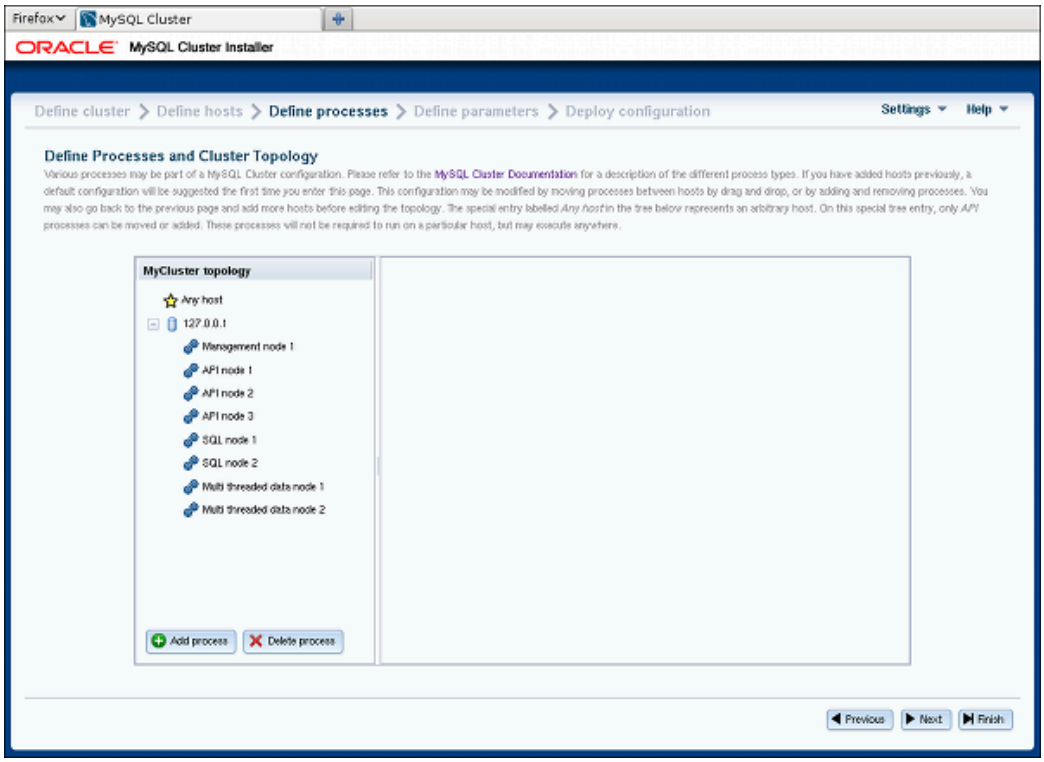
ハードウェアリソース情報、プラットフォーム名、インストールディレクトリ、およびデータディレクトリは、ユーザーがグリッド内の対応するセルをクリックし、1 台以上のホストを選択し、「Edit selected host(s)」というラベルのボタンをクリックすると編集できます。これにより、ここに示すように、これらのフィールドを編集できるダイアログボックスが表示されます。



複数のホストを選択すると、編集した値が選択したすべてのホストに適用されます。

MySQL Cluster Auto-Installer の「Define Processes」画面

ここに示す「Define Processes」画面では、MySQL Cluster プロセス (ノード) をクラスタホストに割り当てることができます。



この画面の左側には、クラスタホストと各ホストで実行するように設定されたプロセスを示すツリーが含まれています。右側には、ツリーで現在選択されている項目に関する情報を表示するパネルがあります。

特定のクラスタでこの画面に最初にアクセスすると、ホストの数に基づいてデフォルトのプロセスセットが自動的に定義されます。その後「Define Hosts」画面に戻り、すべてのホストを削除して新しいホストを追加した場合も、新しいデフォルトのプロセスセットが定義されます。

MySQL Cluster プロセスには次のタイプがあります。

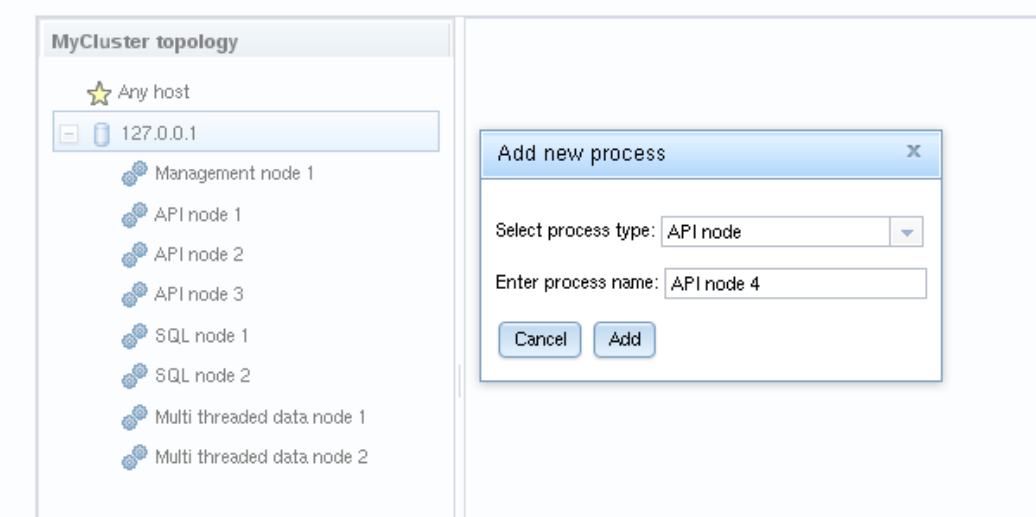
- 管理ノード　個々のデータノードの停止、ノードやクラスタのステータスのクエリー、バックアップの作成などの管理タスクを実行します。実行可能ファイル: ndb_mgmd。
- シングルスレッドのデータノード　データを格納し、クエリーを実行します。実行可能ファイル: ndbd。
- マルチスレッドのデータノード　並列で実行される複数のワーカースレッドによってデータを格納し、クエリーを実行します。実行可能ファイル: ndbmtd。
- SQL ノード　NDB に対して SQL クエリーを実行するための MySQL サーバー。実行可能ファイル: mysqld。
- API ノード　SQL を使用せずに、NDB API またはその他の低レベルのクライアント API を使用して NDB 内のデータにアクセスするクライアント。詳細は、MySQL NDB Cluster API Developer Guideを参照してください。

プロセス (ノード) タイプの詳細は、セクション18.1.1「MySQL Cluster の主な概念」を参照してください。

ツリーに表示されるプロセスには、簡単に識別できるように (たとえば、SQL node 1、SQL node 2 のように) ホストごとにタイプ別の連番が付けられています。

個々の管理ノード、データノード、または SQL プロセスは、特定のホストに割り当てる必要があり、ほかのホストでは実行できません。API ノードは 1 台のホストに割り当てることもできますが、これは必須ではありません。代わりに、ほかのホストとは別にツリー内に表示される「Any host」エントリに別途割り当てることができます。このエントリは、任意のホストで実行できるプロセスのプレースホルダとして機能します。この「Any host」エントリを使用できるのは API プロセスだけです。

プロセスの追加　特定のホストに新しいプロセスを追加するには、ツリー内のそのホストのエントリを右クリックし、表示された「Add process」ポップアップを選択するか、プロセスツリー内のホストを選択して、プロセスツリーの下にある「Add process」ボタンをクリックします。これらのアクションのどちらを実行した場合も、ここに示すようなプロセス追加ダイアログが開きます。



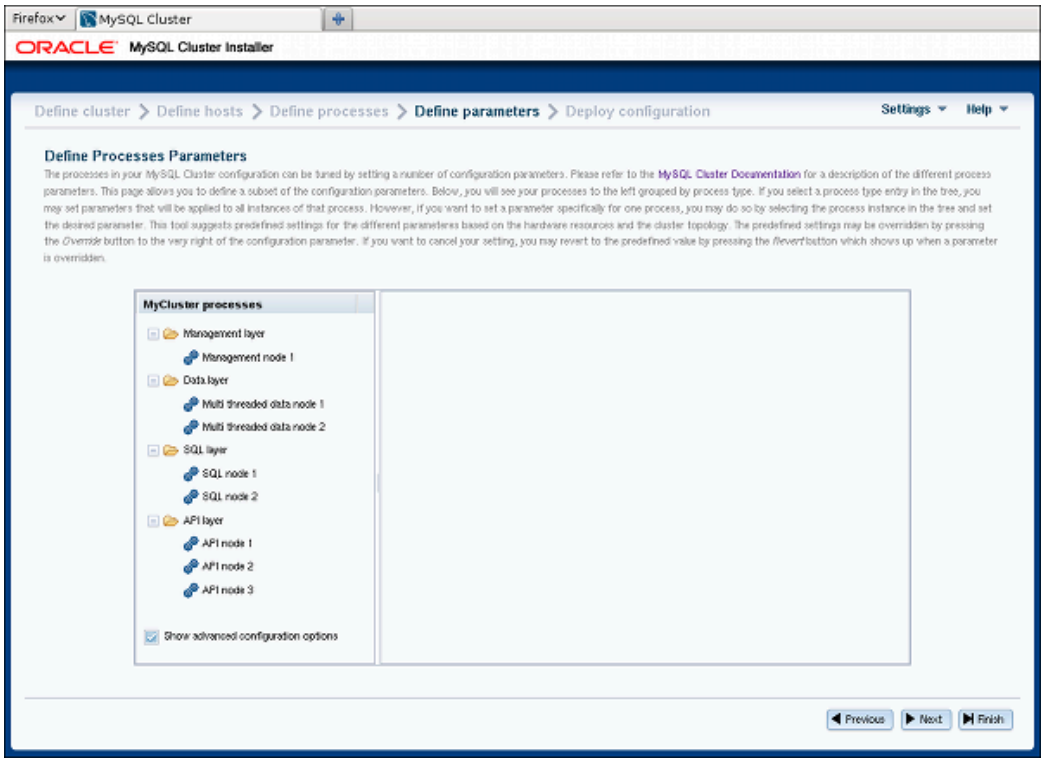
ここでは、このセクションで前述した利用可能なプロセスタイプを選択できます。必要に応じて、提案された値の代わりに任意のプロセス名を入力することもできます。

プロセスの削除　プロセスを削除するには、ツリー内のプロセスを右クリックし、表示されるポップアップメニューから「delete process」を選択します。または、プロセスを選択して、プロセスツリーの下にある「delete process」ボタンを使用します。

プロセスツリー内のプロセスを選択すると、そのプロセスに関する情報がツリーの右側のパネルに表示され、ここでプロセス名 (場合によってはプロセスタイプ) を変更できます。重要: 現在のところ、シングルスレッドデータノード (ndbd) をマルチスレッドデータノード (ndbmtd) に (またはその逆に) 変更できます。ほかのプロセスタイプには変更できません。ほかのプロセスタイプ間で変更する場合は、最初に元のプロセスを削除し、次に目的のタイプの新しいプロセスを追加する必要があります。

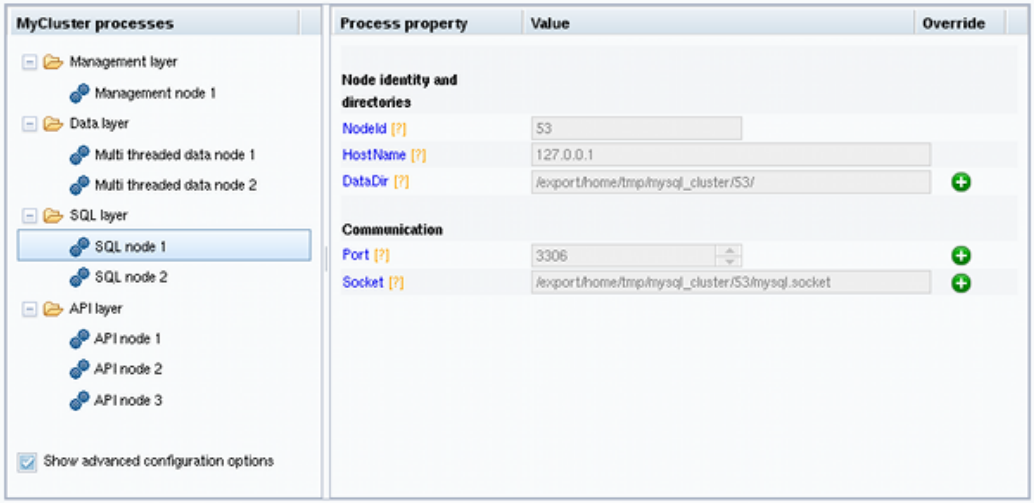
MySQL Cluster Auto-Installer の「Define Attributes」画面

この画面のレイアウトは「Define Processes」画面とほぼ同じで、左側にプロセスツリーが表示されます。その画面のツリーとは異なり、「Define Attributes」プロセスツリーはプロセスまたはノードのタイプで分類され (この場合、シングルスレッドおよびマルチスレッドデータノードは同じタイプとみなされます)、各グループに「Management Layer」、「Data Layer」、「SQL Layer」、および「API Layer」というラベルが付けられています。このツリーの右側のパネルには、現在選択されている項目に関する情報が表示されます。「Define Attributes」画面をここに示します。

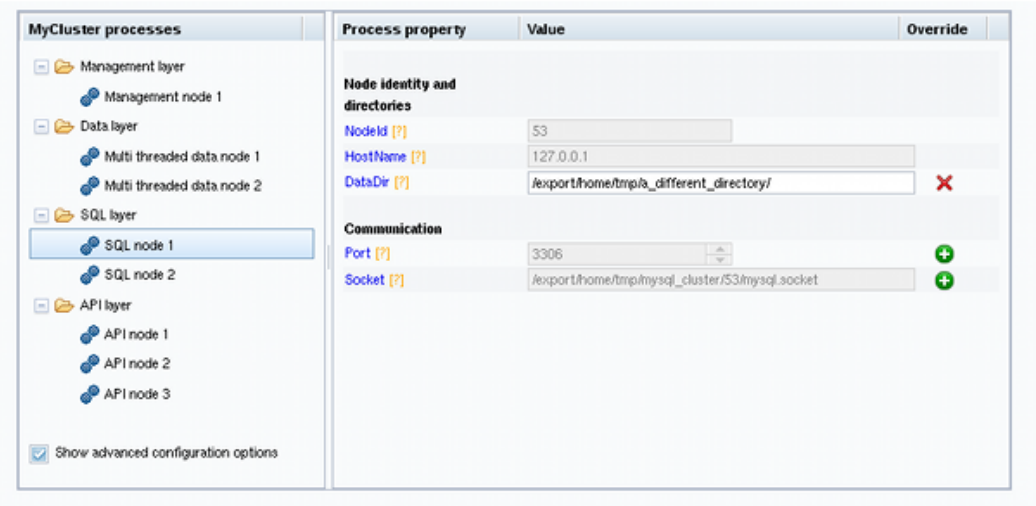


プロセスツリーの下に「Show advanced configuration」というラベルのチェックボックスがあります。このボックスにチェックマークを付けると、情報ペインに高度なオプションが表示されます。これらのオプションは、表示されているかどうかに関係なく設定および使用されます。

1 つのプロセスの属性を編集するには、ツリーからそのプロセスを選択します。クラスタに含まれる同じタイプのプロセスすべての属性を編集するには、いずれかの「Layer」フォルダを選択します。プロセス単位で設定された特定の属性の値は、該当するプロセスに以前に適用されていた属性のグループ単位の設定をオーバーライドします。このような情報パネルの例 (SQL プロセスの場合) をここに示します。



情報パネルに表示された属性の一部については、その右側に、この属性の値をオーバーライドできることを示すプラス記号が付いたボタンが表示されます。この + ボタンをクリックすると、属性の入力ウィジェットがアクティブ化され、属性の値を変更できるようになります。値がオーバーライドされている場合は、ここに示すように、このボタンが X を示すボタンに変更されます。



属性の横にある X ボタンをクリックすると、その属性に対する変更が取り消され、ただちに事前定義値に戻ります。

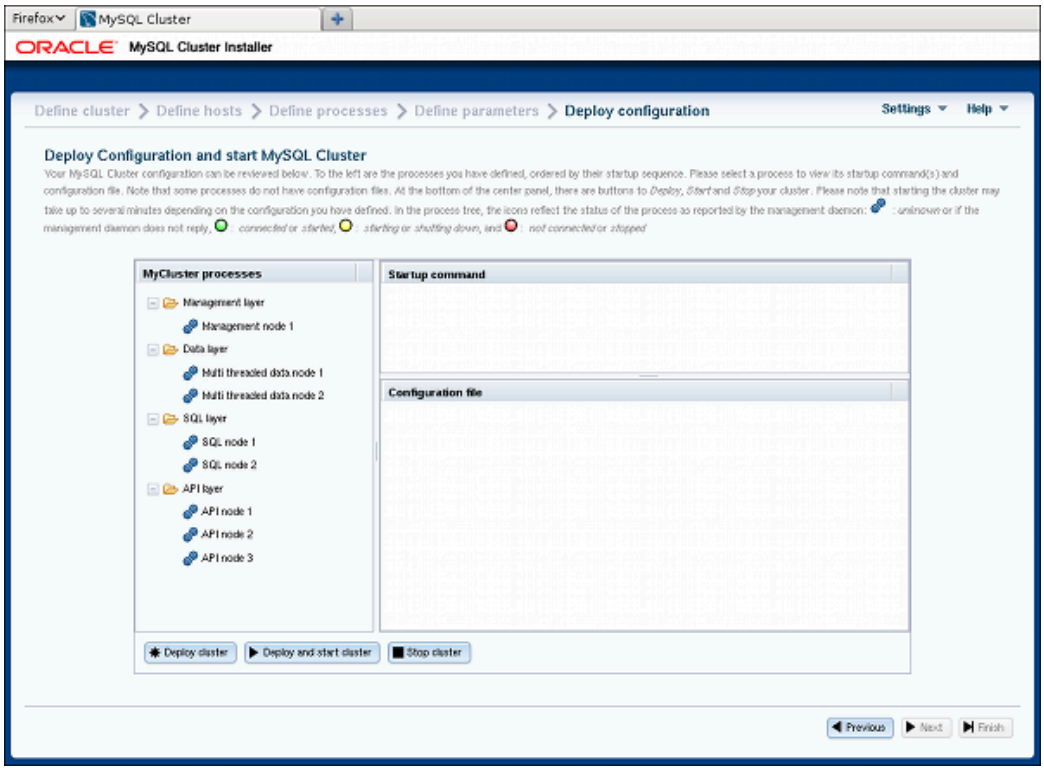
すべての構成属性には、インストーラがホスト名、ノード ID、ノードタイプなどの要因に基づいて計算した事前定義値があります。ほとんどの場合、これらの値はそのままにしておくことができます。これらの操作をまだ習熟していない場合は、属性値を変更する前に該当するドキュメントを読むことを強くお勧めします。この情報を見つけやすくするため、情報パネルに表示される個々の属性名は、MySQL Cluster のオンラインドキュメント内の対応する説明にリンクされています。

MySQL Cluster Auto-Installer の「Deploy Cluster」画面

この画面では、次のタスクを実行できます。

- 適用されるプロセス起動コマンドと構成ファイルを確認します
- すべてのクラスタホストで必要なファイルおよびディレクトリを作成して、構成ファイルを配布します (つまり、現在の構成に従ってクラスタを配備します)
- クラスタを起動および停止します

「Deploy Cluster」画面をここに示します。



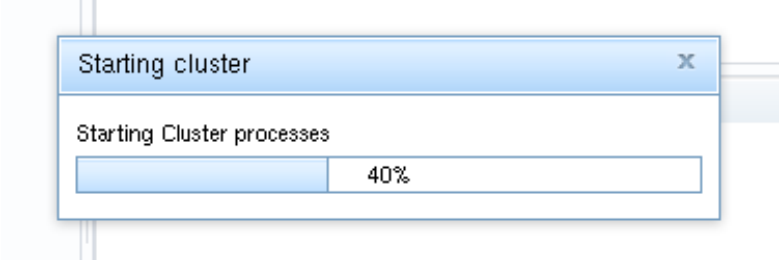
「Define Attributes」画面と同じように、この画面の左側にはプロセスタイプで分類されたプロセスツリーが表示されます。各プロセスの横には、プロセスの現在のステータスを色で示すステータスアイコン (実行中の場合は緑、起動または停止中の場合は黄、プロセスが停止している場合は赤) が表示されます。

プロセスツリーの右側には、2 つの情報パネルが表示されます。上側のパネルには起動コマンド (選択したプロセスを起動するのに必要なコマンド) が表示されます。(プロセスによっては、初期化が必要な場合など、複数のコマンドが必要な場合があります。)下側のパネルには、指定したプロセスの構成ファイル (あれば) の内容が表示されます。現在のところ、構成ファイルがあるプロセスタイプは管理ノードプロセスだけです。ほかのプロセスタイプは、プロセスの起動時にコマンド行パラメータを使用して、または必要に応じて管理ノードからリアルタイムで構成情報を取得して構成されます。

プロセスツリーのすぐ下には、3 つのボタンがあります。これらは、次のリストで説明する機能を実行し、各機能を示すラベルが付いています。

- 「Deploy cluster」: 構成が有効かどうか検証します。クラスタホスト上に必要なディレクトリを作成し、各ホストに構成ファイルを配布します。進行状況バーに、配備の進行状況が表示されます。
- 「Start cluster」: 「Deploy cluster」と同様にクラスタを配備し、その後、すべてのクラスタプロセスを正しい順序で起動します。

  これらのプロセスの起動には、ある程度時間がかかることがあります。完了までの推定時間が長すぎる場合は、起動プロセスを取り消しまたは続行できます。ここに示すように、進行状況バーに起動手順の現在のステータスが示されます。

  前述したプロセスツリーの横にあるプロセスステータスアイコンも、各プロセスのステータスに合わせて更新されます。

- 「Stop cluster」: クラスタを起動したあとは、これを使用して停止できます。クラスタの起動と同様に、クラスタは瞬時にシャットダウンされず、ある程度時間がかかることがあります。クラスタの起動時に表示される

のと同様の進行状況バーが表示され、プロセスツリーの横にあるプロセスステータスアイコンと同じように、クラスタのシャットダウン手順の現在の大まかなステータスが示されます。

MySQL Cluster NDB 7.3.3 より前のバージョンでは、SQL ノードはコマンド行で指定されたすべてのオプションを使用して起動されました。MySQL Cluster NDB 7.3.3 以降では、Auto-Installer によってクラスタ内の各 mysqld プロセスに適したオプションを含む my.cnf ファイルが生成されます。(Bug #16994782)

18.2.2 Linux での MySQL Cluster のインストール

このセクションでは、Linux およびその他の Unix 系オペレーティングシステムでの MySQL Cluster のインストール方法について説明します。次のセクションでは Linux オペレーティングシステムについて言及しますが、記載されている指示と手順は、サポートされるほかの Unix 系プラットフォームにも簡単に適用できます。Windows システムに固有の手動のインストールおよびセットアップ手順については、セクション18.2.3「Windows での MySQL Cluster のインストール」を参照してください。

個々の MySQL Cluster ホストコンピュータに適切な実行可能プログラムがインストールされている必要があります。SQL ノードを実行するホストには、MySQL Server バイナリ (mysqld) がインストールされている必要があります。管理ノードには、管理サーバーデーモン (ndb_mgmd) が必要です。データノードには、データノードデーモン (ndbd または ndbmtd) が必要です。管理ノードホストおよびデータノードホストに MySQL Server バイナリをインストールする必要はありません。管理サーバーホストには管理クライアント (ndb_mgm) もインストールすることをお勧めします。

Linux での MySQL Cluster のインストールは、オラクルが提供する事前コンパイル済みバイナリ (.tar.gz アーカイブとしてダウンロード) を使用するか、RPM パッケージ (これもオラクルから入手可能) を使用するか、ソースコードから実行できます。次のセクションでは、これら 3 つのすべてのインストール方法について説明します。

使用する方法に関係なく、クラスタを起動する前に、MySQL Cluster バイナリのインストールに続いてすべてのクラスタノードの構成ファイルを作成する必要もあります。セクション18.2.4「MySQL Cluster の初期構成」を参照してください。

18.2.2.1 Linux での MySQL Cluster バイナリリリースのインストール

このセクションでは、オラクルが提供する事前コンパイル済みバイナリからクラスタノードの各タイプに対応する適切な実行可能ファイルをインストールするために必要なステップについて説明します。

事前コンパイル済みバイナリを使用してクラスタをセットアップする場合は、各クラスタホストのインストールプロセスの最初のステップとして、MySQL Cluster ダウンロード領域から最新の MySQL Cluster NDB 7.3 以降のバイナリアーカイブ (mysql-cluster-gpl-7.3.9-linux-i686-glibc23.tar.gz または mysql-cluster-gpl-7.4.4-linux-i686-glibc23.tar.gz) をダウンロードします。ここでは、このファイルが各マシンの /var/tmp ディレクトリに配置されていると仮定します。(カスタムバイナリが必要な場合は、セクション2.9.3「開発ソースツリーを使用して MySQL をインストールする」を参照してください。)

> **注記**
>
> インストールが完了しても、バイナリはまだ起動しないでください。ノードの構成に続いてその実行方法を示します (セクション18.2.4「MySQL Cluster の初期構成」を参照してください)。

SQL ノード　SQL ノードのホストとして指定された各マシンで、システムの root ユーザーとして次のステップを実行します。

1. /etc/passwd および /etc/group ファイルを調べて (または、オペレーティングシステムが提供する何らかのユーザーおよびグループ管理ツールを使用して)、システムに mysql グループと mysql ユーザーがすでに存在しているかどうかを確認します。一部の OS 配布では、オペレーティングシステムのインストールプロセスの一部としてこれらが作成されます。まだ存在しない場合は、mysql ユーザーグループを新規作成し、このグループに mysql ユーザーを追加します。

   ```
   shell> groupadd mysql
   shell> useradd -g mysql mysql
   ```

   useradd および groupadd の構文は、Unix のバージョンによって多少異なる場合があり、また adduser および addgroup などの別な名前を使用している場合もあります。

2. ダウンロードしたファイルを含むディレクトリに移動し、アーカイブを解凍して、mysql という名前で mysql ディレクトリへのシンボリックリンクを作成します。実際のファイルおよびディレクトリ名は、MySQL Cluster のバージョン番号によって異なります。

   ```
   shell> cd /var/tmp
   ```

```
shell> tar -C /usr/local -xzvf mysql-cluster-gpl-7.4.4-linux2.6.tar.gz
shell> ln -s /usr/local/mysql-cluster-gpl-7.4.4-linux2.6-i686 /usr/local/mysql
```

3. mysql ディレクトリに移動し、システムデータベースを作成するための付属のスクリプトを実行します。

```
shell> cd mysql
shell> scripts/mysql_install_db --user=mysql
```

4. MySQL サーバーおよびデータディレクトリで必要なアクセス許可を設定します。

```
shell> chown -R root .
shell> chown -R mysql data
shell> chgrp -R mysql .
```

5. MySQL 起動スクリプトを適切なディレクトリにコピーし、実行可能にして、オペレーティングシステムがブートしたときに起動するように設定します。

```
shell> cp support-files/mysql.server /etc/rc.d/init.d/
shell> chmod +x /etc/rc.d/init.d/mysql.server
shell> chkconfig --add mysql.server
```

(起動スクリプトのディレクトリは、オペレーティングシステムとバージョンによって異なります。たとえば、一部の Linux 配布では /etc/init.d です。)

ここでは、起動スクリプトへのリンクを作成するために Red Hat の chkconfig を使用します。使用しているプラットフォームのこの目的に適した何らかの手段 (Debian の update-rc.d など) を使用してください。

前述の各ステップは、SQL ノードを配置するマシンごとに繰り返す必要があります。

データノード　データノードのインストールには、mysqld バイナリは必要ありません。MySQL Cluster のデータノード実行可能ファイル ndbd (シングルスレッド) または ndbmtd (マルチスレッド) のみが必要です。これらのバイナリも、.tar.gz アーカイブに含まれています。ここでも、このアーカイブが /var/tmp に配置されていると仮定します。

システムの root として (つまり、sudo、su root、または使用しているシステムでシステム管理者アカウントの特権を一時的に持つための同等のコマンドを使用したあとで)、次のステップを実行してデータノードホストにデータノードバイナリをインストールします。

1. /var/tmp ディレクトリに移動し、アーカイブに含まれる ndbd および ndbmtd バイナリを /usr/local/bin などの適切なディレクトリに抽出します。

```
shell> cd /var/tmp
shell> tar -zxvf mysql-5.6.22-ndb-7.4.4-linux-i686-glibc23.tar.gz
shell> cd mysql-5.6.22-ndb-7.4.4-linux-i686-glibc23
shell> cp bin/ndbd /usr/local/bin/ndbd
shell> cp bin/ndbmtd /usr/local/bin/ndbmtd
```

(ndb_mgm および ndb_mgmd を実行可能ファイルのディレクトリにコピーしたあとは、ダウンロードしたアーカイブを解凍したときに作成されたディレクトリ (およびディレクトリ内のファイル) を /var/tmp から安全に削除できます。)

2. ファイルをコピーしたディレクトリに移動して、両方のファイルを実行可能にします。

```
shell> cd /usr/local/bin
shell> chmod +x ndb*
```

前述のステップは、データノードホストごとに繰り返してください。

MySQL Cluster データノードを実行するために必要なのはいずれかのデータノード実行可能ファイルだけですが、前の説明では ndbd と ndbmtd の両方のインストール方法を示しました。MySQL Cluster をインストールまたはアップグレードするときは、どちらか一方のみを使用する予定であっても、あとでもう一方に変更する場合の時間と問題が減るため、このようにすることをお勧めします。

> **注記**
>
> データノードをホストする各マシン上のデータディレクトリは /usr/local/mysql/data です。この情報は、管理ノードを構成するときに重要になります。(セクション 18.2.4「MySQL Cluster の初期構成」を参照してください。)

管理ノード　管理ノードのインストールには、mysqld バイナリは必要ありません。MySQL Cluster の管理サーバー (ndb_mgmd) のみが必要です。ほとんどの場合、管理クライアント (ndb_mgm) もインストールする必要があ

ります。これらのバイナリは、どちらも .tar.gz アーカイブに含まれています。ここでも、このアーカイブが /var/tmp に配置されていると仮定します。

システムの root として、次のステップを実行して管理ノードホストに ndb_mgmd および ndb_mgm をインストールします。

1. /var/tmp ディレクトリに移動し、アーカイブに含まれる ndb_mgm および ndb_mgmd を /usr/local/bin などの適切なディレクトリに抽出します。

```
shell> cd /var/tmp
shell> tar -zxvf mysql-5.6.22-ndb-7.4.4-linux2.6-i686.tar.gz
shell> cd mysql-5.6.22-ndb-7.4.4-linux2.6-i686
shell> cp bin/ndb_mgm* /usr/local/bin
```

(ndb_mgm および ndb_mgmd を実行可能ファイルのディレクトリにコピーしたあとは、ダウンロードしたアーカイブを解凍したときに作成されたディレクトリ (およびディレクトリ内のファイル) を /var/tmp から安全に削除できます。)

2. ファイルをコピーしたディレクトリに移動して、両方のファイルを実行可能にします。

```
shell> cd /usr/local/bin
shell> chmod +x ndb_mgm*
```

セクション18.2.4「MySQL Cluster の初期構成」では、この例の MySQL Cluster に含まれるすべてのノード用の構成ファイルを作成しています。

18.2.2.2 RPM からの MySQL Cluster のインストール

このセクションでは、オラクルが提供する RPM パッケージを使用して MySQL Cluster ノードの各タイプに対応する適切な実行可能ファイルをインストールするために必要なステップについて説明します。

RPM は、32 ビットと 64 ビットの両方の Linux プラットフォームで使用できます。これらの RPM のファイル名には、次のパターンが使用されています。

```
MySQL-Cluster-component-producttype-ndbversion.distribution.architecture.rpm

component:= {server | client [| other]}

producttype:= {gpl | advanced}

ndbversion:= major.minor.release

distribution:= {sles10 | rhel5 | el6}

architecture:= {i386 | x86_64}
```

component は、server または client です。(ほかの値になる可能性もありますが、MySQL Cluster の正常なインストールには server および client コンポーネントのみが必要であるため、ここでは説明しません。)https://dev.mysql.com/downloads/cluster/ からダウンロードした Community RPM の producttype は常に gpl です。advanced は商用リリースを示すために使用されます。ndbversion は、3 つの部分から成る (7.3.x または 7.4.x 形式の) NDB ストレージエンジンのバージョン番号を表します。distribution は、sles11 (SUSE Enterprise Linux 11)、rhel5 (Oracle Linux 5、Red Hat Enterprise Linux 4 および 5)、el6 (Oracle Linux 6、Red Hat Enterprise Linux 6) のいずれかです。architecture は、32 ビット RPM の場合は i386、64 ビットバージョンの場合は x86_64 です。

MySQL Cluster では、1 つ (場合によっては 2 つ) の RPM が必要です。

- NDBCLUSTER ストレージエンジンのサポート付きで (つまり、MySQL Cluster の SQL ノードとして) MySQL Server を実行するために必要なコアファイルと、管理ノード、データノード、および ndb_mgm クライアントのバイナリを含むすべての MySQL Cluster 実行可能ファイルを提供する server RPM (たとえば、MySQL-Cluster-server-gpl-7.3.9-1.sles11.i386.rpm または MySQL-Cluster-server-gpl-7.4.4-1.sles11.i386.rpm)。MySQL Cluster をインストールするには、常にこの RPM が必要です。

- MySQL サーバーを管理する機能を持つ独自のクライアントアプリケーションがない場合は、mysql クライアントを提供する client RPM (たとえば、MySQL-Cluster-client-gpl-7.3.9-1.sles11.i386.rpm または MySQL-Cluster-client-gpl-7.4.4-1.sles11.i386.rpm) も入手してインストールしてください。

RPM ファイル名に含まれる MySQL Cluster のバージョン番号 (ここでは、MySQL Cluster NDB 7.3 と MySQL Cluster NDB 7.4 のどちらをインストールするかに応じて 7.3.9 または 7.4.4 として示したもの) は、実際に使用するバージョンによって異なります。インストールするすべてのクラスタ RPM のバージョン番号が同じになってい

ることが非常に重要です。architecture の指定も、RPM をインストールするマシンに適合するようにしてください。特に、32 ビットオペレーティングシステムでは 64 ビット RPM を使用できないことを留意してください。

データノード　クラスタのデータノードをホストするコンピュータには、server RPM のみをインストールする必要があります。これを行うには、この RPM をデータノードホストにコピーし、システムの root ユーザーとして次のコマンドを実行します。示された RPM の名前は、必要に応じて MySQL Web サイトからダウンロードした RPM の名前と一致するように置き換えてください。

```
shell> rpm -Uhv MySQL-Cluster-server-gpl-7.3.9-1.sles11.i386.rpm
```

または

```
shell> rpm -Uhv MySQL-Cluster-server-gpl-7.4.4-1.sles11.i386.rpm
```

これによってすべての MySQL Cluster バイナリがインストールされますが、MySQL Cluster のデータノードを実行するために実際に必要なのは ndbd または ndbmtd プログラム (どちらも /usr/sbin にあります) だけです。

SQL ノード　クラスタの SQL ノードをホストするために使用される各マシンで、システムの root ユーザーとして次のコマンドを実行して server RPM をインストールします。示された RPM の名前は、必要に応じて MySQL Web サイトからダウンロードした RPM の名前と一致するように置き換えてください。

```
shell> rpm -Uhv MySQL-Cluster-server-gpl-7.3.9-1.sles11.i386.rpm
```

または

```
shell> rpm -Uhv MySQL-Cluster-server-gpl-7.4.4-1.sles11.i386.rpm
```

これにより、NDB ストレージエンジンのサポートを含む MySQL サーバーバイナリ (mysqld) と必要なすべての MySQL Server サポートファイルが /usr/sbin ディレクトリにインストールされます。また、mysql.server および mysqld_safe 起動スクリプトも (それぞれ /usr/share/mysql および /usr/bin に) インストールされます。RPM インストーラでは、一般的な構成の問題 (必要に応じて mysql ユーザーおよびグループを作成するなど) に自動的に対応します。

SQL ノード (MySQL サーバー) を管理するには、ここに示すように client RPM もインストールするようにしてください。

```
shell> rpm -Uhv MySQL-Cluster-client-gpl-7.3.9-1.sles11.i386.rpm
```

または

```
shell> rpm -Uhv MySQL-Cluster-client-gpl-7.4.4-1.sles11.i386.rpm
```

これにより、mysql クライアントプログラムがインストールされます。

管理ノード　MySQL Cluster の管理サーバーをインストールするには、server RPM のみを使用する必要があります。この RPM を管理ノードをホストするためのコンピュータにコピーし、システムの root ユーザーとして次のコマンドを実行して、この RPM をインストールします (示された RPM の名前は、必要に応じて MySQL Web サイトからダウンロードした server RPM の名前と一致するように置き換えてください)。

```
shell> rpm -Uhv MySQL-Cluster-server-gpl-7.3.9-1.sles11.i386.rpm
```

または

```
shell> rpm -Uhv MySQL-Cluster-server-gpl-7.4.4-1.sles11.i386.rpm
```

この RPM によってほかの多くのファイルがインストールされますが、管理ノードを実行するために実際に必要なのは管理サーバーバイナリ ndb_mgmd (/usr/sbin ディレクトリにあります) だけです。server RPM によって、NDB の管理クライアントである ndb_mgm もインストールされます。

オラクルが提供する RPM を使った MySQL のインストールに関する一般的な情報は、セクション2.5.5「RPM パッケージを使用して MySQL を Linux にインストールする」を参照してください。

RPM からインストールしたあとは、セクション18.2.4「MySQL Cluster の初期構成」の説明に従ってクラスタを構成する必要もあります。

> **注記**
>
> MySQL Cluster NDB 7.1 で使用されていた一部の RPM は、MySQL Cluster NDB 7.3 で非推奨になり、廃止されました。これには、以前の MySQL-Cluster-clusterj、MySQL-Cluster-extra、MySQL-Cluster-management、MySQL-Cluster-storage、および MySQL Cluster-tools RPM が含まれます。このパッケージの以前の内容は、現在 MySQL-Cluster-server RPM に含まれています。

18.2.2.3 Linux でのソースからの MySQL Cluster のビルド

このセクションでは、Linux およびその他の Unix 系プラットフォームでの MySQL Cluster のコンパイルについて説明します。ソースからの MySQL Cluster のビルドは、標準の MySQL Server のビルドとほぼ同じですが、ここで説明するいくつかの重要な点が異なります。ソースからの MySQL のビルドに関する一般的な情報は、セクション2.9「ソースから MySQL をインストールする」を参照してください。Windows プラットフォームでの MySQL Cluster のコンパイルについては、セクション18.2.3.2「Windows でのソースからの MySQL Cluster のコンパイルとインストール」を参照してください。

MySQL Cluster をビルドするには、MySQL Cluster ソースを使用する必要があります。これらは、MySQL Cluster のダウンロードページ (https://dev.mysql.com/downloads/cluster/) から入手できます。アーカイブされたソースファイルには、mysql-cluster-gpl-7.3.9.tar.gz (MySQL Cluster NDB 7.3) や mysql-cluster-gpl-7.4.4.tar.gz (MySQL Cluster NDB 7.4) のような名前が付いています。launchpad.net から MySQL の開発ソースを入手することもできます。標準の MySQL Server 5.6 ソースからの MySQL Cluster のビルドはサポートされません。

CMake に WITH_NDBCLUSTER_STORAGE_ENGINE オプションを指定すると、管理ノード、データノード、およびその他の MySQL Cluster プログラムのバイナリがビルドされます。また、これによって NDB ストレージエンジンのサポート付きで mysqld がコンパイルされます。このオプションは、MySQL Cluster NDB 7.3 以降のソースではデフォルトで有効になっています。

> **重要**
>
> MySQL Cluster NDB 7.3 以降では、WITH_NDB_JAVA オプションがデフォルトで有効になっています。つまり、デフォルトでは CMake でシステム上の Java の場所が見つからなかった場合に構成プロセスが失敗します。Java および ClusterJ のサポートを有効にしない場合は、-DWITH_NDB_JAVA=OFF を使用してビルドを構成することで、これを明示的に示す必要があります。必要な場合は、WITH_CLASSPATH を使用して Java クラスパスを指定します。

MySQL Cluster のビルドに固有の CMake オプションの詳細は、MySQL Cluster をコンパイルするためのオプションを参照してください。

make && make install (または使用しているシステムの同等のコマンド) を実行すると、同じ場所に事前コンパイル済みバイナリを解凍した場合と同じ結果が得られます。

管理ノード　ソースからビルドしてデフォルトの make install を実行すると、管理サーバーと管理クライアントのバイナリ (ndb_mgmd と ndb_mgm) が /usr/local/mysql/bin に見つかります。管理ノードホストに配置する必要があるのは ndb_mgmd だけですが、同じホストマシンに ndb_mgm も配置することをお勧めします。これらの実行可能ファイルは、どちらもホストマシンのファイルシステム上の特定の場所に配置する必要はありません。

データノード　データノードホストに配置する必要がある実行可能ファイルは、データノードバイナリ ndbd または ndbmtd だけです。(たとえば、mysqld をホストマシンに配置する必要はありません。)ソースからビルドすると、デフォルトではこのファイルは /usr/local/mysql/bin ディレクトリに配置されます。複数のデータノードホストにインストールする場合、ほかのマシンにコピーする必要があるのは ndbd または ndbmtd だけです。(これは、すべてのデータノードホストで同じアーキテクチャーとオペレーティングシステムが使用されていることが前提です。そうでない場合は、異なるプラットフォームごとに別個にコンパイルする必要がある可能性があります。)データノードバイナリは、その場所が既知であるかぎり、ホストのファイルシステム上の特定の場所に配置する必要はありません。

ソースから MySQL Cluster をコンパイルする場合、マルチスレッドのデータノードバイナリをビルドするために特別なオプションは必要ありません。NDB ストレージエンジンのサポート付きでビルドを構成すると、自動的に ndbmtd がビルドされます。make install を実行すると、ndbmtd バイナリは mysqld、ndbd、および ndb_mgm とともにインストールの bin ディレクトリに配置されます。

SQL ノード　クラスタリングのサポート付きで MySQL をコンパイルし、デフォルトのインストールを実行 (システムの root ユーザーとして make install を使用) すると、mysqld は /usr/local/mysql/bin に配置されます。セクション2.9「ソースから MySQL をインストールする」に示したステップに従って、mysqld を使用できるようにします。複数の SQL ノードを実行する場合は、同じ mysqld 実行可能ファイルと関連するサポートファイルのコピーを複数のマシンで使用できます。これを実行するもっとも簡単な方法は、/usr/local/mysql ディレクトリ全体およびその内部に含まれているすべてのディレクトリとファイルをほかの SQL ノードホストにコピーし、各マシンでセクション2.9「ソースから MySQL をインストールする」のステップを繰り返すことです。デフォルトではない PREFIX オプションを指定してビルドを構成する場合は、それに合わせてディレクトリを調整する必要があります。

セクション18.2.4「MySQL Cluster の初期構成」では、この例の MySQL Cluster に含まれるすべてのノード用の構成ファイルを作成しています。

18.2.3 Windows での MySQL Cluster のインストール

このセクションでは、Windows ホストでの MySQL Cluster のインストール手順について説明します。Windows 用の MySQL Cluster NDB 7.3 および MySQL Cluster NDB 7.4 バイナリは、https://dev.mysql.com/downloads/cluster/ から入手できます。Windows でオラクルが提供するバイナリリリースから MySQL Cluster をインストールする方法については、セクション18.2.3.1「Windows でのバイナリリリースからの MySQL Cluster のインストール」を参照してください。

Windows では、Microsoft Visual Studio を使用してソースから MySQL Cluster をコンパイルしてインストールすることもできます。詳細は、セクション18.2.3.2「Windows でのソースからの MySQL Cluster のコンパイルとインストール」を参照してください。

18.2.3.1 Windows でのバイナリリリースからの MySQL Cluster のインストール

このセクションでは、オラクルが提供する MySQL Cluster の no-install バイナリリリースを使用した MySQL Cluster の基本的なインストールについて説明します。ここでは、次の表に示すように、このセクションの冒頭 (セクション18.2「MySQL Cluster のインストール」を参照してください) で概説したのと同じ 4 ノードのセットアップを使用します。

ノード	IP アドレス
管理 (MGMD) ノード	192.168.0.10
MySQL サーバー (SQL) ノード	192.168.0.20
データ (NDBD) ノード "A"	192.168.0.30
データ (NDBD) ノード "B"	192.168.0.40

ほかのプラットフォームと同様に、SQL ノードを実行する MySQL Cluster ホストコンピュータには MySQL Server バイナリ (mysqld.exe) をインストールする必要があります。このホストには MySQL クライアント (mysql.exe) も配置するようにしてください。管理ノードおよびデータノードに MySQL Server バイナリをインストールする必要はありません。各管理サーバーには、管理サーバーデーモン (ndb_mgmd.exe) が必要です。各データノードには、データノードデーモン (ndbd.exe または ndbmtd.exe) が必要です。この例では ndbd.exe をデータノード実行可能ファイルと呼びますが、代わりにこのプログラムのマルチスレッドバージョンである ndbmtd.exe をまったく同じ方法でインストールできます。管理サーバーホストには、管理クライアント (ndb_mgm.exe) もインストールするようにしてください。このセクションでは、MySQL Cluster ノードの各タイプに対応する適切な Windows バイナリをインストールするのに必要なステップについて説明します。

> 注記
>
> ほかの Windows プログラムと同様に、MySQL Cluster 実行可能ファイルの名前には .exe ファイル拡張子が付けられています。ただし、コマンド行からこれらのプログラムを起動するときに .exe 拡張子を含める必要はありません。そのため、このドキュメントでは多くの場合、これらのプログラムを mysqld、mysql、ndb_mgmd などと呼びます。ここでは、(たとえば) mysqld と mysqld.exe のどちらの呼び方であっても、どちらの名前も同じもの (MySQL Server プログラム) を意味しています。

オラクルの no-install バイナリを使用して MySQL Cluster をセットアップする場合は、インストールプロセスの最初のステップとして、https://dev.mysql.com/downloads/cluster/ から最新の MySQL Cluster Windows バイナリアーカイブをダウンロードします。このアーカイブには、mysql-cluster-gpl-noinstall-ver-winarch.zip という形式のファイル名が付けられます。ver は NDB ストレージエンジンのバージョン (7.3.1 など) であり、arch はアーキテクチャー (32 ビットバイナリの場合は 32、64 ビットバイナリの場合は 64) です。たとえば、MySQL Cluster NDB 7.3.1 の 32 ビット Windows システム用 no-install アーカイブの名前は mysql-cluster-gpl-noinstall-7.3.1-win32.zip です。

32 ビット MySQL Cluster バイナリは Windows の 32 ビットと 64 ビットの両方のバージョンで実行できますが、64 ビット MySQL Cluster バイナリは Windows の 64 ビットバージョンでのみ使用できます。64 ビット CPU を搭載したコンピュータで Windows の 32 ビットバージョンを使用する場合は、32 ビット MySQL Cluster バイナリを使用する必要があります。

インターネットからダウンロードしたりマシン間でコピーしたりする必要があるファイルの数を最小限に抑えるため、ここでは SQL ノードを実行するコンピュータから始めます。

SQL ノード　ここでは、IP アドレスが 192.168.0.20 であるコンピュータ上の C:\Documents and Settings\username\My Documents\Downloads (username は現在のユーザーの名前です) ディレクトリに no-install アーカイブのコピーが配置されていると仮定します。(この名前は、コマンド行で ECHO %USERNAME% を使用する

と表示されます。)MySQL Cluster 実行可能ファイルを Windows サービスとしてインストールおよび実行するには、このユーザーが Administrators グループのメンバーになります。

アーカイブからすべてのファイルを抽出します。このタスクには、Windows Explorer に組み込まれた抽出ウィザードが適しています。(別のアーカイブプログラムを使用する場合は、アーカイブからすべてのファイルとディレクトリが抽出されたこと、アーカイブのディレクトリ構造が維持されていることを確認してください。)出力先のディレクトリを求められたら、C:\ と入力します。抽出ウィザードによってディレクトリ C:\mysql-cluster-gpl-noinstall-ver-winarch にアーカイブが抽出されます。このディレクトリの名前を C:\mysql に変更します。

MySQL Cluster バイナリを C:\mysql\bin 以外のディレクトリにインストールすることもできますが、その場合は、この手順に示されているパスをそれに合わせて変更する必要があります。特に、MySQL Server (SQL ノード) バイナリを C:\mysql または C:\Program Files\MySQL\MySQL Server 5.6 以外の場所にインストールした場合や、SQL ノードのデータディレクトリが C:\mysql\data または C:\Program Files\MySQL\MySQL Server 5.6\data 以外の場所にある場合は、SQL ノードの起動時に、追加の構成オプションをコマンド行で使用するか、my.ini または my.cnf ファイルに追加する必要があります。標準以外の場所で実行するように MySQL Server を構成する方法の詳細は、セクション2.3.5「非インストール Zip アーカイブを使用して Microsoft Windows に MySQL をインストールする」を参照してください。

MySQL Cluster サポートを含む MySQL Server を MySQL Cluster の一部として実行するには、--ndbcluster および --ndb-connectstring オプションを指定して起動する必要があります。これらのオプションは、コマンド行で指定することもできますが、通常はオプションファイルに設定する方が便利です。そのためには、メモ帳などのテキストエディタで新しいテキストファイルを作成します。このファイルに次の構成情報を入力します。

```
[mysqld]
# Options for mysqld process:
ndbcluster                # run NDB storage engine
ndb-connectstring=192.168.0.10  # location of management server
```

この MySQL Server が使用するほかのオプション (セクション2.3.5.2「オプションファイルの作成」を参照してください) を必要に応じて追加できますが、このファイルには少なくともここに示したオプションを含める必要があります。このファイルを C:\mysql\my.ini として保存します。これで、SQL ノードのインストールとセットアップが完了します。

データノード　Windows ホストの MySQL Cluster データノードには、ndbd.exe または ndbmtd.exe のいずれか 1 つの実行可能ファイルのみが必要です。この例では、ndbd.exe を使用すると仮定しますが、ndbmtd.exe を使用するときも同じ手順が適用されます。データノードを実行する各コンピュータ (IP アドレスが 192.168.0.30 および 192.168.0.40 のコンピュータ) で、C:\mysql、C:\mysql\bin、および C:\mysql\cluster-data の各ディレクトリを作成します。次に、no-install アーカイブをダウンロードして抽出したコンピュータで、C:\mysql\bin ディレクトリ内の ndbd.exe を見つけます。このファイルを 2 台のデータノードホストの C:\mysql\bin ディレクトリにそれぞれコピーします。

データノードを MySQL Cluster の一部として機能させるには、各ノードに対して管理サーバーのアドレスまたはホスト名を指定する必要があります。この情報を指定するには、各データノードプロセスの起動時にコマンド行で --ndb-connectstring または -c オプションを使用します。ただし、通常はオプションファイルにこの情報を指定することをお勧めします。そのためには、メモ帳などのテキストエディタで新しいテキストファイルを作成して、次のテキストを入力します。

```
[mysql_cluster]
# Options for data node process:
ndb-connectstring=192.168.0.10  # location of management server
```

このファイルをデータノードホストに C:\mysql\my.ini として保存します。もう一方のデータノードホストで同じ内容を含むテキストファイルをもう 1 つ作成し、それを C:mysql\my.ini として保存するか、my.ini ファイルを 1 つ目のデータノードホストから 2 つ目のデータノードホストにコピーし、そのコピーを 2 つ目のデータノードの C:\mysql ディレクトリに確実に配置します。これで、両方のデータノードホストを MySQL Cluster で使用できるようになりました。あとは、管理ノードをインストールして構成するだけです。

管理ノード　MySQL Cluster 管理ノードのホストとして使用するコンピュータに必要な実行可能プログラムは、管理サーバープログラム ndb_mgmd.exe だけです。ただし、起動された MySQL Cluster を管理するため、管理サーバーと同じマシンに MySQL Cluster 管理クライアントプログラム ndb_mgm.exe もインストールしてください。no-install アーカイブをダウンロードして抽出したマシンで、これら 2 つのプログラムを見つけます。これは、SQL ノードホストの C:\mysql\bin ディレクトリになります。IP アドレスが 192.168.0.10 であるコンピュータに C:\mysql\bin ディレクトリを作成し、両方のプログラムをこのディレクトリにコピーします。

ここで、ndb_mgmd.exe が使用する 2 つの構成ファイルを作成してください。

1. 管理ノード自体に固有の構成データを提供するローカル構成ファイル。通常、このファイルに指定する必要があるのは、MySQL Cluster グローバル構成ファイル (項目 2 を参照してください) の場所だけです。

このファイルを作成するには、メモ帳などのテキストエディタで新しいテキストファイルを作成し、次の情報を入力します。

```
[mysql_cluster]
# Options for management node process
config-file=C:/mysql/bin/config.ini
```

このファイルをプレーンテキストファイル C:\mysql\bin\my.ini として保存します。

2. 管理ノードが MySQL Cluster 全体を制御する構成情報を取得できるグローバル構成ファイル。このファイルには、少なくとも MySQL Cluster 内の各ノード用のセクションと、管理ノードおよびすべてのデータノードの IP アドレスまたはホスト名 (HostName 構成パラメータ) が含まれている必要があります。また、次の追加情報も含めることをお勧めします。

   - SQL ノードの IP アドレスまたはホスト名
   - 各データノードに割り当てられたデータメモリーおよびインデックスメモリー (DataMemory および IndexMemory 構成パラメータ)
   - レプリカの数 (NoOfReplicas 構成パラメータを使用します。セクション18.1.2「MySQL Cluster のノード、ノードグループ、レプリカ、およびパーティション」を参照してください)
   - 各データノードがデータおよびログファイルを格納するディレクトリと、管理ノードがログファイルを保持するディレクトリ (どちらの場合も、DataDir 構成パラメータ)

   メモ帳などのテキストエディタを使用して新しいテキストファイルを作成し、次の情報を入力します。

```
[ndbd default]
# Options affecting ndbd processes on all data nodes:
NoOfReplicas=2                  # Number of replicas
DataDir=C:/mysql/cluster-data      # Directory for each data node's data files
                                # Forward slashes used in directory path,
                                # rather than backslashes. This is correct;
                                # see Important note in text
DataMemory=80M    # Memory allocated to data storage
IndexMemory=18M   # Memory allocated to index storage
                # For DataMemory and IndexMemory, we have used the
                # default values. Since the "world" database takes up
                # only about 500KB, this should be more than enough for
                # this example Cluster setup.

[ndb_mgmd]
# Management process options:
HostName=192.168.0.10             # Hostname or IP address of management node
DataDir=C:/mysql/bin/cluster-logs   # Directory for management node log files

[ndbd]
# Options for data node "A":
                            # (one [ndbd] section per data node)
HostName=192.168.0.30         # Hostname or IP address

[ndbd]
# Options for data node "B":
HostName=192.168.0.40         # Hostname or IP address

[mysqld]
# SQL node options:
HostName=192.168.0.20         # Hostname or IP address
```

このファイルをプレーンテキストファイル C:\mysql\bin\config.ini として保存します。

> **重要**
>
> Windows 上の MySQL Cluster が使用するプログラムオプションまたは構成ファイルでは、ディレクトリパスを指定するときに単独のバックスラッシュ文字 (\) を使用できません。代わりに、個々のバックスラッシュ文字を 2 つ目のバックスラッシュ (\\) でエスケープするか、バックスラッシュをスラッシュ文字 (/) に置き換えてください。たとえば、MySQL Cluster の config.ini ファイルの [ndb_mgmd] セクションから抜粋した次の行は機能しません。
>
> ```
> DataDir=C:\mysql\bin\cluster-logs
> ```
>
> 代わりに、次のいずれかを使用できます。

```
DataDir=C:\\mysql\\bin\\cluster-logs  # Escaped backslashes
```

```
DataDir=C:/mysql/bin/cluster-logs     # Forward slashes
```

簡潔さと読みやすさのため、Windows 上の MySQL Cluster プログラムのオプションや構成ファイルで使用するディレクトリパスには、スラッシュを使用することをお勧めします。

18.2.3.2 Windows でのソースからの MySQL Cluster のコンパイルとインストール

オラクルは、ほとんどのユーザーに適合する事前コンパイル済みの Windows 用 MySQL Cluster バイナリを提供しています。ただし、必要な場合はソースコードから Windows 用の MySQL Cluster をコンパイルすることもできます。これを実行する手順は、標準の Windows 用 MySQL Server バイナリをコンパイルする場合の手順とほぼ同じであり、同じツールを使用します。ただし、2 つの大きな違いがあります。

- MySQL Cluster をビルドするには、https://dev.mysql.com/downloads/cluster/ から入手できる MySQL Cluster ソースを使用する必要があります。

  標準の MySQL Server のソースコードから MySQL Cluster をビルドしようとすると、失敗する可能性が高く、オラクルではサポートされません。

- CMake で使用するほかのビルドオプションに加えて、WITH_NDBCLUSTER_STORAGE_ENGINE または WITH_NDBCLUSTER オプションを使用してビルドを構成する必要があります。(WITH_NDBCLUSTER は WITH_NDBCLUSTER_STORAGE_ENGINE のエイリアスとしてサポートされており、まったく同じように機能します。)

  重要

  MySQL Cluster NDB 7.3 以降では、WITH_NDB_JAVA オプションがデフォルトで有効になっています。つまり、デフォルトでは CMake でシステム上の Java の場所が見つからなかった場合に構成プロセスが失敗します。Java および ClusterJ のサポートを有効にしない場合は、-DWITH_NDB_JAVA=OFF を使用してビルドを構成することで、これを明示的に示す必要があります。(Bug #12379735) 必要な場合は、WITH_CLASSPATH を使用して Java クラスパスを指定します。

MySQL Cluster のビルドに固有の CMake オプションの詳細は、MySQL Cluster をコンパイルするためのオプションを参照してください。

ビルドプロセスが完了したら、コンパイルされたバイナリを含む Zip アーカイブを作成できます。Windows システムでこのタスクを実行するのに必要なコマンドについては、セクション2.9.2「標準ソース配布を使用して MySQL をインストールする」を参照してください。MySQL Cluster バイナリは、生成されたアーカイブの bin ディレクトリに含まれています。このアーカイブは no-install アーカイブと同等であり、同じ方法でインストールおよび構成できます。詳細は、セクション18.2.3.1「Windows でのバイナリリリースからの MySQL Cluster のインストール」を参照してください。

18.2.3.3 Windows での MySQL Cluster の初期起動

MySQL Cluster の実行可能ファイルと必要な構成ファイルを配置したあと、クラスタの初期起動を行うには、単純にクラスタ内のすべてのノードで MySQL Cluster の実行可能ファイルを起動します。各クラスタノードプロセスは、それが配置されているホストコンピュータ上で別個に起動する必要があります。最初に管理ノード、次にデータノード、最後に SQL ノードを起動するようにしてください。

1. 管理ノードホストでは、コマンド行から次のコマンドを発行して管理ノードプロセスを起動します。ここに示すような出力が表示されます。

   ```
   C:\mysql\bin> ndb_mgmd
   2010-06-23 07:53:34 [MgmtSrvr] INFO -- NDB Cluster Management Server. mysql-5.6.22-ndb-7.4.4
   2010-06-23 07:53:34 [MgmtSrvr] INFO -- Reading cluster configuration from 'config.ini'
   ```

   管理ノードプロセスは、ロギング出力をコンソールに出力し続けます。管理ノードは Windows サービスとして実行されていないため、これは正常な動作です。(Linux などの Unix 系プラットフォームで MySQL Cluster を使用したことがある場合は、これに関する Windows での管理ノードのデフォルトの動作が実質的に Unix システムの動作 (デフォルトでは Unix デーモンプロセスとして実行されます) と逆であることに気付くでしょう。この動作は、Windows で実行される MySQL Cluster データノードプロセスにも当てはまります。)このため、ndb_mgmd.exe が実行されているウィンドウを閉じないでください。閉じると、管理ノードプロセスが強制終了されます。(MySQL Cluster プロセスを Windows サービスとしてインストールして実行する方法については、セクション18.2.3.4「Windows サービスとしての MySQL Cluster プロセスのインストール」を参照してください。)

必須の -f オプションで、管理ノードにグローバル構成ファイル (config.ini) がある場所を指示します。このオプションの長形式は --config-file です。

> **重要**
>
> MySQL Cluster 管理ノードは、config.ini から読み取った構成データをキャッシュします。構成キャッシュが作成されたあとは、強制的に読み取りを行わないかぎり、その後の起動時に config.ini ファイルは無視されます。つまり、このファイル内のエラーが原因で管理ノードが起動に失敗した場合は、エラーを修正してから、管理ノードに config.ini を再度読み取らせる必要があります。これを行うには、コマンド行で --reload または --initial オプションを指定して ndb_mgmd.exe を起動します。これらのオプションは、どちらも構成キャッシュをリフレッシュする機能を持っています。
>
> 管理ノードの my.ini ファイルでこれらのオプションのいずれかを使用することは、必要ないか、または推奨されません。

ndb_mgmd で使用できるオプションに関する追加情報は、セクション18.4.4「ndb_mgmd — MySQL Cluster 管理サーバーデーモン」およびセクション18.4.27「MySQL Cluster プログラムに共通するオプション — MySQL Cluster プログラムに共通するオプション」を参照してください。

2. 各データノードホストで、ここに示すコマンドを実行してデータノードプロセスを起動します。

```
C:\mysql\bin> ndbd
2010-06-23 07:53:46 [ndbd] INFO -- Configuration fetched from 'localhost:1186', generation: 1
```

いずれの場合も、データノードプロセスによって生成される出力の最初の行は前の例で示したものと似ていますが、そのあとにロギング出力の行が追加されます。管理ノードと同様に、データノードは Windows サービスとして実行されていないため、これは正常な動作です。このため、データノードプロセスが実行されているコンソールウィンドウを閉じないでください。閉じると、ndbd.exe が強制終了されます。(詳細はセクション18.2.3.4「Windows サービスとしての MySQL Cluster プロセスのインストール」を参照してください。)

3. SQL ノードをまだ起動しないでください。データノードの起動 (しばらく時間がかかることがあります) が完了するまでは、SQL ノードをクラスタに接続できません。代わりに、管理ノードホストの新しいコンソールウィンドウで、管理ノードホストの C:\mysql\bin にある MySQL Cluster 管理クライアント ndb_mgm.exe を起動します。(CTRL+C を入力して ndb_mgmd.exe が実行されているコンソールウィンドウを再利用しないでください。これを行うと管理ノードが強制終了されます。)生成される出力は次のようになります。

```
C:\mysql\bin> ndb_mgm
-- NDB Cluster -- Management Client --
ndb_mgm>
```

ndb_mgm> というプロンプトが表示された場合、これは管理クライアントが MySQL Cluster の管理コマンドを受信できるようになったことを示します。管理クライアントのプロンプトで ALL STATUS と入力すると、データノードの起動時のステータスを確認できます。このコマンドによって、データノードの起動シーケンスのレポートが実行され、次のように表示されます。

```
ndb_mgm> ALL STATUS
Connected to Management Server at: localhost:1186
Node 2: starting (Last completed phase 3) (mysql-5.6.22-ndb-7.4.4)
Node 3: starting (Last completed phase 3) (mysql-5.6.22-ndb-7.4.4)

Node 2: starting (Last completed phase 4) (mysql-5.6.22-ndb-7.4.4)
Node 3: starting (Last completed phase 4) (mysql-5.6.22-ndb-7.4.4)

Node 2: Started (version 7.4.4)
Node 3: Started (version 7.4.4)

ndb_mgm>
```

> **注記**
>
> 管理クライアントで発行されるコマンドでは大文字と小文字が区別されません。ここでは、コマンドの標準形式として大文字を使用しますが、ndb_mgm クライア

> ントに入力するときに、この表記法に従う必要はありません。詳細は、セクション 18.5.2「MySQL Cluster 管理クライアントのコマンド」を参照してください。

ALL STATUS によって生成される出力は、データノードの起動速度、使用する MySQL Cluster ソフトウェアのリリースバージョン番号、およびその他の要因によって、ここに示すものと異なる可能性があります。重要なのは、両方のデータノードの起動を確認したときに、SQL ノードの起動準備が整うことです。

ndb_mgm.exe は実行したままにできます。MySQL Cluster のパフォーマンスに悪影響を与えることはありません。次のステップではこれを使用して、起動した SQL ノードがクラスタに接続したか確認します。

4. SQL ノードホストとして指定したコンピュータで、コンソールウィンドウを開き、MySQL Cluster バイナリを解凍したディレクトリ (この例に従っている場合、これは C:\mysql\bin です) に移動します。

   SQL ノードを起動するには、ここに示すように、コマンド行で mysqld.exe を起動します。

   ```
   C:\mysql\bin> mysqld --console
   ```

   --console オプションによって、コンソールにロギング情報が書き込まれます。これは問題が発生したときに役立つことがあります。(SQL ノードが問題なく実行されていることを確認できたら、SQL ノードを停止してから --console オプションを指定せずに起動すると、ロギングが通常どおり実行されるようになります。)

   管理ノードホストの管理クライアント (ndb_mgm.exe) が実行されているコンソールウィンドウで、SHOW コマンドを入力します。ここに示すような出力が生成されます。

   ```
   ndb_mgm> SHOW
   Connected to Management Server at: localhost:1186
   Cluster Configuration
   ---------------------
   [ndbd(NDB)]     2 node(s)
   id=2    @192.168.0.30  (Version: 5.6.22-ndb-7.4.4, Nodegroup: 0, *)
   id=3    @192.168.0.40  (Version: 5.6.22-ndb-7.4.4, Nodegroup: 0)

   [ndb_mgmd(MGM)] 1 node(s)
   id=1    @192.168.0.10  (Version: 5.6.22-ndb-7.4.4)

   [mysqld(API)]   1 node(s)
   id=4    @192.168.0.20  (Version: 5.6.22-ndb-7.4.4)
   ```

   また、SQL ノードが MySQL Cluster に接続されているか確認するには、mysql クライアント (mysql.exe) で SHOW ENGINE NDB STATUS ステートメントを使用します。

これで、MySQL Cluster の NDBCLUSTER ストレージエンジンを使用してデータベースオブジェクトとデータを操作する準備ができました。詳細と例については、セクション18.2.6「テーブルとデータを含む MySQL Cluster の例」を参照してください。

ndb_mgmd.exe、ndbd.exe、および ndbmtd.exe を Windows サービスとしてインストールすることもできます。これを行う方法については、セクション18.2.3.4「Windows サービスとしての MySQL Cluster プロセスのインストール」を参照してください)。

18.2.3.4 Windows サービスとしての MySQL Cluster プロセスのインストール

MySQL Cluster が要求どおりに実行されていることを確認できたら、管理ノードとデータノードを Windows サービスとしてインストールできます。これにより、Windows の起動または停止に合わせて各プロセスを自動的に起動および停止できます。また、この場合は、コマンド行で適切な NET START または NET STOP コマンドを使用するか、Windows のグラフィカルな「サービス」ユーティリティーを使用すると、このプロセスを制御できます。

プログラムを Windows サービスとしてインストールする場合は、通常、システム上の Administrator 権利を持つアカウントを使用する必要があります。

管理ノードを Windows 上のサービスとしてインストールするには、ここに示すように、管理ノードをホストするマシンのコマンド行で --install オプションを使用して ndb_mgmd.exe を起動します。

```
C:\> C:\mysql\bin\ndb_mgmd.exe --install
Installing service 'MySQL Cluster Management Server'
  as '"C:\mysql\bin\ndbd.exe" "--service=ndb_mgmd"'
Service successfully installed.
```

> **重要**
>
> MySQL Cluster プログラムを Windows サービスとしてインストールするときは、常に完全なパスを指定するようにしてください。そうしないと、サービスのインストールが失敗し、次のエラーが発生します: The system cannot find the file specified。

--install オプションは、ndb_mgmd.exe に指定できるほかのオプションより先に使用する必要があります。ただし、このようなオプションはオプションファイルに指定することをお勧めします。オプションファイルが ndb_mgmd.exe --help の出力に示されるデフォルトの場所のいずれにも存在しない場合は、--config-file オプションを使用するとその場所を指定できます。

これで、このように管理サーバーを起動および停止できるようになります。

```
C:\> NET START ndb_mgmd
The MySQL Cluster Management Server service is starting.
The MySQL Cluster Management Server service was started successfully.

C:\> NET STOP ndb_mgmd
The MySQL Cluster Management Server service is stopping..
The MySQL Cluster Management Server service was stopped successfully.
```

ここに示すように、管理サーバーを Windows サービスとして起動または停止するときに記述名を使用することもできます。

```
C:\> NET START 'MySQL Cluster Management Server'
The MySQL Cluster Management Server service is starting.
The MySQL Cluster Management Server service was started successfully.

C:\> NET STOP  'MySQL Cluster Management Server'
The MySQL Cluster Management Server service is stopping..
The MySQL Cluster Management Server service was stopped successfully.
```

ただし、通常は短いサービス名を指定するか、サービスのインストール時にデフォルトのサービス名を使用できるように設定して、サービスの起動または停止時にその名前を参照する方が簡単です。ndb_mgmd 以外のサービス名を指定するには、この例に示すように --install オプションを追加します。

```
C:\> C:\mysql\bin\ndb_mgmd.exe --install=mgmd1
Installing service 'MySQL Cluster Management Server'
  as '"C:\mysql\bin\ndb_mgmd.exe" "--service=mgmd1"'
Service successfully installed.
```

これで、このように指定した名前を使用してサービスを起動または停止できるようになります。

```
C:\> NET START mgmd1
The MySQL Cluster Management Server service is starting.
The MySQL Cluster Management Server service was started successfully.

C:\> NET STOP mgmd1
The MySQL Cluster Management Server service is stopping..
The MySQL Cluster Management Server service was stopped successfully.
```

管理ノードサービスを削除するには、ここに示すように --remove オプションを指定して ndb_mgmd.exe を起動します。

```
C:\> C:\mysql\bin\ndb_mgmd.exe --remove
Removing service 'MySQL Cluster Management Server'
Service successfully removed.
```

デフォルト以外のサービス名を使用してサービスをインストールした場合は、このように --remove オプションの値としてこの名前を渡すと、サービスを削除できます。

```
C:\> C:\mysql\bin\ndb_mgmd.exe --remove=mgmd1
Removing service 'mgmd1'
Service successfully removed.
```

Windows サービスとしての MySQL Cluster データノードプロセスのインストールは、ここに示すように、ndbd.exe (または ndbmtd.exe) の --install オプションを使用して同じように実行できます。

```
C:\> C:\mysql\bin\ndbd.exe --install
Installing service 'MySQL Cluster Data Node Daemon' as '"C:\mysql\bin\ndbd.exe" "--service=ndbd"'
Service successfully installed.
```

これで、次の例に示すように net start または net stop でデフォルトのサービス名または記述名を使用してデータノードを起動または停止できます。

```
C:\> NET START ndbd
The MySQL Cluster Data Node Daemon service is starting.
The MySQL Cluster Data Node Daemon service was started successfully.

C:\> NET STOP ndbd
The MySQL Cluster Data Node Daemon service is stopping..
The MySQL Cluster Data Node Daemon service was stopped successfully.

C:\> NET START 'MySQL Cluster Data Node Daemon'
The MySQL Cluster Data Node Daemon service is starting.
The MySQL Cluster Data Node Daemon service was started successfully.

C:\> NET STOP 'MySQL Cluster Data Node Daemon'
The MySQL Cluster Data Node Daemon service is stopping..
The MySQL Cluster Data Node Daemon service was stopped successfully.
```

データノードサービスを削除するには、ここに示すように --remove オプションを指定して ndbd.exe を起動します。

```
C:\> C:\mysql\bin\ndbd.exe --remove
Removing service 'MySQL Cluster Data Node Daemon'
Service successfully removed.
```

ndb_mgmd.exe (および mysqld.exe) と同様に、ndbd.exe を Windows サービスとしてインストールするときは、このようにサービスの名前を --install の値として指定し、サービスの起動または停止時にそれを使用することもできます。

```
C:\> C:\mysql\bin\ndbd.exe --install=dnode1
Installing service 'dnode1' as '"C:\mysql\bin\ndbd.exe" "--service=dnode1"'
Service successfully installed.

C:\> NET START dnode1
The MySQL Cluster Data Node Daemon service is starting.
The MySQL Cluster Data Node Daemon service was started successfully.

C:\> NET STOP dnode1
The MySQL Cluster Data Node Daemon service is stopping..
The MySQL Cluster Data Node Daemon service was stopped successfully.
```

データノードサービスのインストール時にサービス名を指定した場合は、ここに示すように、それを削除するときもこの名前を (--remove オプションの値として渡すことで) 使用できます。

```
C:\> C:\mysql\bin\ndbd.exe --remove=dnode1
Removing service 'dnode1'
Service successfully removed.
```

Windows サービスとしての SQL ノードのインストール、サービスの起動、サービスの停止、およびサービスの削除も、mysqld --install、NET START、NET STOP、および mysqld --remove を使用して同じように行います。追加情報については、セクション2.3.5.7「Windows のサービスとして MySQL を起動する」を参照してください。

18.2.4 MySQL Cluster の初期構成

このセクションでは、構成ファイルの作成と編集によるインストール済み MySQL Cluster の手動構成について説明します。

MySQL Cluster (NDB バージョン 7.3 以降) には、別のアプリケーションでテキストファイルを編集せずに構成を行うために使用できる GUI インストーラも用意されています。詳細は、セクション18.2.1「MySQL Cluster Auto-Installer」を参照してください。

ここで使用する 4 ノードおよび 4 ホストの MySQL Cluster (クラスタノードとホストコンピュータを参照してください) では、4 つ (ノードホストごとに 1 つずつ) の構成ファイルを作成する必要があります。

- 個々のデータノードまたは SQL ノードには my.cnf ファイルが必要です。このファイルは、管理ノードが配置されたノードを指定する接続文字列と、このホスト (データノードをホストしているマシン) の MySQL サーバーに対して NDBCLUSTER ストレージエンジンを有効にするように指示する行の 2 つの情報を提供します。

  接続文字列の詳細は、セクション18.3.2.3「MySQL Cluster の接続文字列」を参照してください。

- 管理ノードには config.ini ファイルが必要です。このファイルは、保持するレプリカの数、各データノードのデータおよびインデックスに割り当てるメモリーの量、データノードの場所、各データノードのデータを保存するディスク上の場所、および SQL ノードの場所を指示します。

データノードと SQL ノードの構成　データノードに必要な my.cnf ファイルはかなり単純です。この構成ファイルは、/etc ディレクトリに配置され、任意のテキスト工ディタを使用して編集できます。(このファイルが存在しない場合は作成してください。)例:

```
shell> vi /etc/my.cnf
```

> **注記**
>
> ここでは、vi を使用してこのファイルを作成しますが、どのテキスト工ディタでも同じように機能します。

このセットアップ例の各データノードおよび SQL ノードでは、my.cnf はこのようになります。

```
[mysqld]
# Options for mysqld process:
ndbcluster                      # run NDB storage engine

[mysql_cluster]
# Options for MySQL Cluster processes:
ndb-connectstring=192.168.0.10  # location of management server
```

上記の情報を入力したら、このファイルを保存してテキスト工ディタを終了します。これを、データノード「A」、データノード「B」、および SQL ノードをホストしているマシンで実行します。

> **重要**
>
> 上記の my.cnf ファイルの [mysqld] および [mysql_cluster] セクションの ndbcluster および ndb-connectstring パラメータを使用して mysqld プロセスを起動したあとは、クラスタが実際に起動しないと、CREATE TABLE または ALTER TABLE ステートメントを実行できません。そうでない場合、これらのステートメントはエラーで失敗します。これは意図的なものです。

管理ノードの構成　管理ノードの構成では、最初のステップとして構成ファイルを配置するディレクトリを作成し、その後構成ファイル自体を作成します。例 (root として実行します):

```
shell> mkdir /var/lib/mysql-cluster
shell> cd /var/lib/mysql-cluster
shell> vi config.ini
```

ここで使用する典型的なセットアップでは、config.ini ファイルは次のようになります。

```
[ndbd default]
# Options affecting ndbd processes on all data nodes:
NoOfReplicas=2    # Number of replicas
DataMemory=80M    # How much memory to allocate for data storage
IndexMemory=18M   # How much memory to allocate for index storage
                  # For DataMemory and IndexMemory, we have used the
                  # default values. Since the "world" database takes up
                  # only about 500KB, this should be more than enough for
                  # this example Cluster setup.

[tcp default]
# TCP/IP options:
portnumber=2202   # This the default; however, you can use any
                  # port that is free for all the hosts in the cluster
                  # Note: It is recommended that you do not specify the port
                  # number at all and simply allow the default value to be used
                  # instead

[ndb_mgmd]
# Management process options:
hostname=192.168.0.10          # Hostname or IP address of MGM node
datadir=/var/lib/mysql-cluster  # Directory for MGM node log files

[ndbd]
# Options for data node "A":
                               # (one [ndbd] section per data node)
hostname=192.168.0.30          # Hostname or IP address
datadir=/usr/local/mysql/data   # Directory for this data node's data files

[ndbd]
# Options for data node "B":
hostname=192.168.0.40          # Hostname or IP address
datadir=/usr/local/mysql/data   # Directory for this data node's data files

[mysqld]
```

```
# SQL node options:
hostname=192.168.0.20          # Hostname or IP address
                               # (additional mysqld connections can be
                               # specified for this node for various
                               # purposes such as running ndb_restore)
```

> **注記**
>
> world データベースは https://dev.mysql.com/doc/ からダウンロードできます (「Examples」 の下にあります)。

すべての構成ファイルを作成し、最小限のオプションを指定したら、クラスタの起動とすべてのプロセスの実行確認に進む準備が整います。これを行う方法については、セクション18.2.5「MySQL Cluster の初期起動」で説明します。

使用可能な MySQL Cluster 構成パラメータとその使用方法の詳細は、セクション18.3.2「MySQL Cluster の構成ファイル」およびセクション18.3「MySQL Cluster の構成」を参照してください。MySQL Cluster のバックアップ作成に関する構成については、セクション18.5.3.3「MySQL Cluster バックアップ用の構成」を参照してください。

> **注記**
>
> クラスタ管理ノードのデフォルトポートは 1186 です。データノードのデフォルトポートは 2202 です。ただし、クラスタではすでに開放されているポートからデータノードのポートを自動的に割り当てることができます。

18.2.5 MySQL Cluster の初期起動

構成後のクラスタを起動することは、それほど難しいことではありません。各クラスタノードプロセスは、配置されているホストで別個に起動する必要があります。最初に管理ノード、次にデータノード、最後に SQL ノードを起動してください。

1. 管理ホストで、システムシェルから次のコマンドを発行して管理ノードプロセスを起動します。

   ```
   shell> ndb_mgmd -f /var/lib/mysql-cluster/config.ini
   ```

   ndb_mgmd をはじめて起動するときは、-f または --config-file オプションを使用してその構成ファイルの場所を指定する必要があります。(詳細は、セクション18.4.4「ndb_mgmd — MySQL Cluster 管理サーバーデーモン」を参照してください。)

   ndb_mgmd で使用できる追加のオプションについては、セクション18.4.27「MySQL Cluster プログラムに共通するオプション — MySQL Cluster プログラムに共通するオプション」を参照してください。

2. 個々のデータノードホストで、このコマンドを実行して ndbd プロセスを起動します。

   ```
   shell> ndbd
   ```

3. SQL ノードを配置するクラスタホストで RPM ファイルを使用して MySQL をインストールした場合は、付属の起動スクリプトを使用して SQL ノードの MySQL サーバープロセスを起動できます (起動するようにしてください)。

すべてが順調に進み、クラスタが正しくセットアップされると、クラスタは使用できる状態になります。これをテストするには、ndb_mgm 管理ノードクライアントを起動します。ここに示すような出力が表示されますが、使用している MySQL の特定のバージョンによって出力内容がやや異なる場合があります。

```
shell> ndb_mgm
-- NDB Cluster -- Management Client --
ndb_mgm> SHOW
Connected to Management Server at: localhost:1186
Cluster Configuration
---------------------
[ndbd(NDB)]     2 node(s)
id=2    @192.168.0.30  (Version: 5.6.22-ndb-7.4.4, Nodegroup: 0, *)
id=3    @192.168.0.40  (Version: 5.6.22-ndb-7.4.4, Nodegroup: 0)

[ndb_mgmd(MGM)] 1 node(s)
id=1    @192.168.0.10  (Version: 5.6.22-ndb-7.4.4)

[mysqld(API)]   1 node(s)
id=4    @192.168.0.20  (Version: 5.6.22-ndb-7.4.4)
```

ここでは、SQL ノードが [mysqld(API)] として参照されています。これは、mysqld プロセスが MySQL Cluster の API ノードとして機能していることを反映しています。

> **注記**
>
> SHOW の出力で MySQL Cluster の特定の SQL ノードまたはその他の API ノードに関して表示されている IP アドレスは、SQL または API ノードが管理ノードではなくクラスタデータノードに接続するために使用するアドレスです。

これで、MySQL Cluster 内のデータベース、テーブル、およびデータを操作する準備が整いました。簡単な説明については、セクション18.2.6「テーブルとデータを含む MySQL Cluster の例」を参照してください。

18.2.6 テーブルとデータを含む MySQL Cluster の例

> **注記**
>
> このセクションの情報は、Unix と Windows のプラットフォームで実行される MySQL Cluster に適用されます。

MySQL Cluster 内のデータベーステーブルおよびデータの操作は、標準の MySQL の場合とほとんど違いはありません。留意すべき重要なポイントが 2 つあります。

- クラスタ内でレプリケートされるテーブルでは、NDBCLUSTER ストレージエンジンを使用する必要があります。これを指定するには、テーブルの作成時に ENGINE=NDBCLUSTER または ENGINE=NDB オプションを使用します。

  ```
  CREATE TABLE tbl_name (col_name column_definitions) ENGINE=NDBCLUSTER;
  ```

  または、異なるストレージエンジンを使用する既存のテーブルに対して ALTER TABLE を使用して、NDBCLUSTER を使用するようにテーブルを変更します。

  ```
  ALTER TABLE tbl_name ENGINE=NDBCLUSTER;
  ```

- すべての NDBCLUSTER テーブルには主キーがあります。テーブルの作成時にユーザーが主キーを定義しなかった場合は、NDBCLUSTER ストレージエンジンが隠し主キーを自動的に生成します。このようなキーは、ほかのテーブルインデックスと同じサイズの領域を占有します。(これらの自動的に作成されるインデックスを格納する十分なメモリーがないために問題が発生することは、珍しくありません。)

mysqldump の出力を使用して既存のデータベースからテーブルをインポートする場合は、SQL スクリプトをテキストエディタで開いて、テーブル作成ステートメントに ENGINE オプションを追加するか、既存の ENGINE オプションを置き換えることができます。MySQL Cluster をサポートしない別の MySQL サーバーに world サンプルデータベースがあり、City テーブルをエクスポートする必要があるとします。

```
shell> mysqldump --add-drop-table world City > city_table.sql
```

この結果作成される city_table.sql ファイルには、このテーブル作成ステートメント (およびテーブルデータをインポートするために必要な INSERT ステートメント) が格納されます。

```
DROP TABLE IF EXISTS `City`;
CREATE TABLE `City` (
  `ID` int(11) NOT NULL auto_increment,
  `Name` char(35) NOT NULL default '',
  `CountryCode` char(3) NOT NULL default '',
  `District` char(20) NOT NULL default '',
  `Population` int(11) NOT NULL default '0',
  PRIMARY KEY  (`ID`)
) ENGINE=MyISAM DEFAULT CHARSET=latin1;

INSERT INTO `City` VALUES (1,'Kabul','AFG','Kabol',1780000);
INSERT INTO `City` VALUES (2,'Qandahar','AFG','Qandahar',237500);
INSERT INTO `City` VALUES (3,'Herat','AFG','Herat',186800);
(remaining INSERT statements omitted)
```

MySQL がこのテーブルに対して NDBCLUSTER ストレージエンジンを使用するか確認する必要があります。これを行うには 2 つの方法があります。1 つは、テーブル定義をクラスタデータベースにインポートする前に変更することです。例として City テーブルを使用し、定義の ENGINE オプションを次のように変更します。

```
DROP TABLE IF EXISTS `City`;
CREATE TABLE `City` (
  `ID` int(11) NOT NULL auto_increment,
  `Name` char(35) NOT NULL default '',
  `CountryCode` char(3) NOT NULL default '',
```

```
 `District` char(20) NOT NULL default '',
 `Population` int(11) NOT NULL default '0',
 PRIMARY KEY  (`ID`)
) ENGINE=NDBCLUSTER DEFAULT CHARSET=latin1;

INSERT INTO `City` VALUES (1,'Kabul','AFG','Kabol',1780000);
INSERT INTO `City` VALUES (2,'Qandahar','AFG','Qandahar',237500);
INSERT INTO `City` VALUES (3,'Herat','AFG','Herat',186800);
(remaining INSERT statements omitted)
```

クラスタ化されたデータベースの一部になる個々のテーブルの定義で、これを行う必要があります。これを行うもっとも簡単な方法は、定義を含むファイルに対して検索置換を実行し、TYPE=engine_name または ENGINE=engine_name のすべてのインスタンスを ENGINE=NDBCLUSTER に置換することです。ファイルを変更したくない場合は、変更していないファイルを使用してテーブルを作成してから、ALTER TABLE を使用してストレージエンジンを変更します。詳細はこのセクションで後述します。

クラスタの SQL ノードで world という名前のデータベースがすでに作成されていることを前提に、mysql コマンド行クライアントを使用して city_table.sql を読み取り、対応するテーブルを通常の方法で作成して移入します。

```
shell> mysql world < city_table.sql
```

非常に重要な留意点として、前述のコマンドは SQL ノードを実行しているホスト (この場合は、IP アドレスが 192.168.0.20 のマシン) で実行する必要があります。

SQL ノードで world データベース全体のコピーを作成するには、非クラスタサーバーで mysqldump を使用して、(たとえば) /tmp ディレクトリの world.sql という名前のファイルにデータベースをエクスポートします。次に、前述のとおりにテーブル定義を変更し、このようにファイルをクラスタの SQL ノードにインポートします。

```
shell> mysql world < /tmp/world.sql
```

ファイルを別の場所に保存する場合は、それに合わせて前述の手順を調整してください。

SQL ノードでの SELECT クエリーの実行は、MySQL サーバーのほかのインスタンスでの実行と違いはありません。コマンド行からクエリーを実行するには、最初に通常の方法で (Enter password: プロンプトで root パスワードを指定して) MySQL Monitor に口グインする必要があります。

```
shell> mysql -u root -p
Enter password:
Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 1 to server version: 5.6.22-ndb-7.4.4

Type 'help;' or '\h' for help. Type '\c' to clear the buffer.

mysql>
```

ここでは、MySQL サーバーの root アカウントを使用し、MySQL サーバーのインストールに関する標準的なセキュリティー対策 (強力な root パスワードの設定を含む) に従っていると仮定します。詳細は、セクション 2.10.2「最初の MySQL アカウントのセキュリティー設定」を参照してください。

クラスタノードが相互にアクセスするときは MySQL の権限システムが利用されないことを考慮に入れてください。MySQL ユーザーアカウント (root アカウントを含む) の設定または変更は、SQL ノードにアクセスするアプリケーションにのみ影響し、ノード間のやり取りには影響しません。詳細は、セクション18.5.11.2「MySQL Cluster と MySQL 権限」を参照してください。

SQL スクリプトをインポートする前にテーブル定義の ENGINE 句を変更しなかった場合は、この時点で次のステートメントを実行してください。

```
mysql> USE world;
mysql> ALTER TABLE City ENGINE=NDBCLUSTER;
mysql> ALTER TABLE Country ENGINE=NDBCLUSTER;
mysql> ALTER TABLE CountryLanguage ENGINE=NDBCLUSTER;
```

データベースの選択とそのデータベース内のテーブルに対する SELECT クエリーの実行も、MySQL Monitor の終了と同様に通常の方法で行います。

```
mysql> USE world;
mysql> SELECT Name, Population FROM City ORDER BY Population DESC LIMIT 5;
+-----------+------------+
| Name      | Population |
+-----------+------------+
| Bombay    |   10500000 |
| Seoul     |    9981619 |
| São Paulo |    9968485 |
| Shanghai  |    9696300 |
| Jakarta   |    9604900 |
```

```
+-----------+------------+
5 rows in set (0.34 sec)

mysql> \q
Bye

shell>
```

MySQL を使用するアプリケーションは、標準の API を使用して NDB テーブルにアクセスできます。アプリケーションは、管理ノードやデータノードではなく、SQL ノードにアクセスする必要があります。この簡単な例では、ネットワーク上の別の場所にある Web サーバーで実行されている PHP 5.X の mysqli 拡張を使用して、ここに示す SELECT ステートメントを実行する方法を示しています。

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"
 "http://www.w3.org/TR/html4/loose.dtd">
<html>
<head>
 <meta http-equiv="Content-Type"
       content="text/html; charset=iso-8859-1">
 <title>SIMPLE mysqli SELECT</title>
</head>
<body>
<?php
 # connect to SQL node:
 $link = new mysqli('192.168.0.20', 'root', 'root_password', 'world');
 # parameters for mysqli constructor are:
 #   host, user, password, database

 if( mysqli_connect_errno() )
   die("Connect failed: " . mysqli_connect_error());

 $query = "SELECT Name, Population
          FROM City
          ORDER BY Population DESC
          LIMIT 5";

 # if no errors...
 if( $result = $link->query($query) )
 {
?>
<table border="1" width="40%" cellpadding="4" cellspacing ="1">
 <tbody>
 <tr>
   <th width="10%">City</th>
   <th>Population</th>
 </tr>
<?
   # then display the results...
   while($row = $result->fetch_object())
     printf("<tr>\n  <td align=\"center\">%s</td><td>%d</td>\n</tr>\n",
           $row->Name, $row->Population);
?>
 </tbody
</table>
<?
 # ...and verify the number of rows that were retrieved
   printf("<p>Affected rows: %d</p>\n", $link->affected_rows);
 }
 else
   # otherwise, tell us what went wrong
   echo mysqli_error();

 # free the result set and the mysqli connection object
 $result->close();
 $link->close();
?>
</body>
</html>
```

ここでは、Web サーバーで実行されているプロセスが SQL ノードの IP アドレスに到達できると仮定します。

同様の方法で、MySQL C API、Perl-DBI、Python-mysql、または MySQL Connector を使用して、MySQL で通常実行する同じデータの定義や操作のタスクを実行できます。

18.2.7 MySQL Cluster の安全なシャットダウンと再起動

クラスタをシャットダウンするには、管理ノードをホストしているマシン上のシェルで次のコマンドを入力します。

```
shell> ndb_mgm -e shutdown
```

この -e オプションは、シェルから ndb_mgm クライアントにコマンドを渡すために使用されます。(このオプションの詳細は、セクション18.4.27「MySQL Cluster プログラムに共通するオプション — MySQL Cluster プログラムに共通するオプション」を参照してください。)このコマンドによって、ndb_mgm、ndb_mgmd、およびすべての ndbd または ndbmtd プロセスが適切な方法で終了します。SQL ノードは mysqladmin shutdown およびその他の方法で終了できます。Windows プラットフォームでは、SQL ノードを Windows サービスとしてインストールした場合に NET STOP MYSQL を使用できます。

Unix プラットフォームでクラスタを再起動するには、次のコマンドを実行します。

- 管理ホスト (このセットアップ例では 192.168.0.10):

```
shell> ndb_mgmd -f /var/lib/mysql-cluster/config.ini
```

- 個々のデータノードホスト (192.168.0.30 および 192.168.0.40):

```
shell> ndbd
```

- ndb_mgm クライアントを使用して、両方のデータノードが正常に起動したか確認します。
- SQL ホスト (192.168.0.20):

```
shell> mysqld_safe &
```

Windows プラットフォームでは、すべての MySQL Cluster プロセスをデフォルトのサービス名を使用して Windows サービスとしてインストールした場合に (セクション18.2.3.4「Windows サービスとしての MySQL Cluster プロセスのインストール」を参照してください)、次のようにクラスタを再起動できます。

- 管理ホスト (このセットアップ例では 192.168.0.10) で、次のコマンドを実行します。

```
C:\> NET START ndb_mgmd
```

- 個々のデータノードホスト (192.168.0.30 および 192.168.0.40) で、次のコマンドを実行します。

```
C:\> NET START ndbd
```

- 管理ノードホストで、ndb_mgm クライアントを使用して管理ノードと両方のデータノードが正常に起動したか確認します (セクション18.2.3.3「Windows での MySQL Cluster の初期起動」を参照してください)。
- SQL ノードホスト (192.168.0.20) で、次のコマンドを実行します。

```
C:\> NET START mysql
```

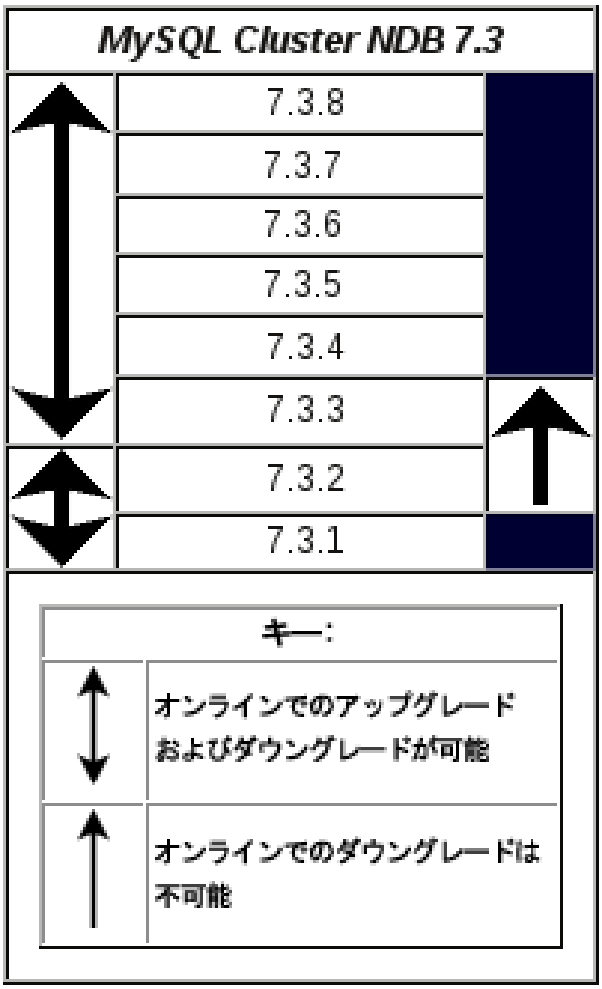
本番設定では、通常、クラスタを完全にシャットダウンすることは望ましくありません。多くの場合、構成変更やクラスタのハードウェアまたはソフトウェア (あるいはその両方) のアップグレードを実行して個々のホストマシンをシャットダウンする必要が生じた場合でも、クラスタのローリング再起動を実行することで、クラスタ全体をシャットダウンせずに処理を完了できます。この実行方法の詳細は、セクション18.5.5「MySQL Cluster のローリング再起動の実行」を参照してください。

18.2.8 MySQL Cluster NDB 7.3 のアップグレードとダウングレード

このセクションでは、異なる MySQL Cluster NDB 7.3 リリース間でのアップグレードおよびダウングレードの実行に関する MySQL Cluster ソフトウェアおよびテーブルファイルの互換性、および互換性のマトリックスと注意事項について説明します。アップグレードまたはダウングレードを試行する前に、ユーザーは MySQL Cluster のインストールと構成にすでに習熟していることが求められます。セクション18.3「MySQL Cluster の構成」を参照してください。

> **重要**
>
> このセクションでは、MySQL バージョン間の NDBCLUSTER に関する互換性のみを考慮しますが、考慮すべき問題はほかにもある場合があります。ほかの MySQL ソフトウェアのアップグレードまたはダウングレードと同様に、MySQL Cluster ソフトウェアのアップグレードまたはダウングレードを試行する前に、移行先および移行元の MySQL バージョンの MySQL マニュアルの関連部分を確認することを強くお勧めします。セクション2.11.1「MySQL のアップグレード」を参照してください。

ここに示す表は、MySQL Cluster NDB 7.3 の異なるリリース間での MySQL Cluster のアップグレードおよびダウングレードの互換性に関する情報を示しています。表の直後に、MySQL Cluster NDB 7.3 リリースシリーズへの、リリースシリーズからの、またはリリースシリーズ間のアップグレードおよびダウングレードに関する追加の注意事項を示します。



注意事項: MySQL Cluster NDB 7.3

サポートされるバージョン　MySQL Cluster NDB 7.3 (7.3.2 移行) へのアップグレードについては、MySQL Cluster の次のバージョンがサポートされます。

- MySQL Cluster NDB 7.2 GA リリース (7.2.4 以降)
- MySQL Cluster NDB 7.1 GA リリース (7.1.3 以降)
- MySQL Cluster NDB 7.0 GA リリース (7.0.5 以降)
- MySQL Cluster NDB 7.1 にアップグレードできる MySQL Cluster NDB 6.3 GA リリース (6.3.8 以降)

MySQL Cluster NDB 6.3 以降の最近のリリースで使用される NDB API、ClusterJ、およびその他のアプリケーションは、書き換えや再コンパイルをしなくても MySQL Cluster NDB 7.3.2 以降で引き続き使用できます。

MySQL Cluster NDB 7.3.3 以降から MySQL Cluster NDB 7.3.2 以前にオンラインでダウングレードすることはできません。MySQL Cluster NDB 7.3.2 からその後の MySQL Cluster NDB 7.3 リリースへのオンラインでのアップグレードはサポートされます。

18.3 MySQL Cluster の構成

MySQL Cluster の一部である MySQL サーバーが通常の (クラスタ化されていない) MySQL サーバーと大きく異なる点は、NDB ストレージエンジンを採用していることです。このエンジンは NDBCLUSTER と呼ばれることもありますが、NDB が推奨されます。

不要なリソースの割り当てを避けるため、デフォルトでは NDB ストレージエンジンを無効にするようにサーバーが構成されます。NDB を有効にするには、サーバーの my.cnf 構成ファイルを変更するか、--ndbcluster オプションを指定してサーバーを起動する必要があります。

この MySQL サーバーはクラスタの一部であるため、管理ノードにアクセスしてクラスタ構成データを取得する方法も認識する必要があります。デフォルトの動作では、localhost 上の管理ノードを検索します。ただし、それがほかの場所であることを指定する必要がある場合は、my.cnf で、または mysql クライアントを使用してこれを行います。NDB ストレージエンジンを使用する前に、少なくとも 1 つの管理ノードと必要なデータノードが使用可能である必要があります。

--ndbcluster および MySQL Cluster に固有のその他の mysqld オプションの詳細は、セクション18.3.4.2「MySQL Cluster 用の MySQL Server オプション」を参照してください。

MySQL Cluster NDB 7.3.1 以降では、MySQL Cluster Auto-Installer を使用して、ブラウザベースの GUI を使用する 1 台以上のホストに MySQL Cluster をセットアップおよび配備できます。詳細は、セクション18.2.1「MySQL Cluster Auto-Installer」を参照してください。

MySQL Cluster のインストールに関する一般情報は、セクション18.2「MySQL Cluster のインストール」を参照してください。

18.3.1 MySQL Cluster の簡易テストセットアップ

ここでは、基本を習得するため、実用的な MySQL Cluster のもっとも簡単な構成について説明します。その後、この章の関連するほかのセクションに示した情報から、必要なセットアップを設計できるようになります。

最初に、システムの root ユーザーとして次のコマンドを実行して、/var/lib/mysql-cluster などの構成ディレクトリを作成する必要があります。

```
shell> mkdir /var/lib/mysql-cluster
```

このディレクトリで、次の情報を含む config.ini という名前のファイルを作成します。必要に応じて、HostName および DataDir をシステムの適切な値に置き換えます。

```
# file "config.ini" - showing minimal setup consisting of 1 data node,
# 1 management server, and 3 MySQL servers.
# The empty default sections are not required, and are shown only for
# the sake of completeness.
# Data nodes must provide a hostname but MySQL Servers are not required
# to do so.
# If you don't know the hostname for your machine, use localhost.
# The DataDir parameter also has a default value, but it is recommended to
# set it explicitly.
# Note: [db], [api], and [mgm] are aliases for [ndbd], [mysqld], and [ndb_mgmd],
# respectively. [db] is deprecated and should not be used in new installations.

[ndbd default]
NoOfReplicas= 1

[mysqld  default]
[ndb_mgmd default]
[tcp default]

[ndb_mgmd]
HostName= myhost.example.com

[ndbd]
HostName= myhost.example.com
DataDir= /var/lib/mysql-cluster

[mysqld]
[mysqld]
[mysqld]
```

これで、ndb_mgmd 管理サーバーを起動できるようになりました。デフォルトでは現在の作業ディレクトリ内にある config.ini ファイルの読み取りが試行されるため、このファイルが配置されているディレクトリに場所を移動してから ndb_mgmd を起動します。

```
shell> cd /var/lib/mysql-cluster
shell> ndb_mgmd
```

次に、ndbd を実行して 1 つのデータノードを起動します。

```
shell> ndbd
```

ndbd の起動時に使用できるコマンド行オプションについては、セクション18.4.27「MySQL Cluster プログラムに共通するオプション — MySQL Cluster プログラムに共通するオプション」を参照してください。

デフォルトでは、ndbd は localhost のポート 1186 で管理サーバーを検索します。

> **注記**
>
> バイナリ tarball から MySQL をインストールした場合は、ndb_mgmd および ndbd サーバーのパスを明示的に指定する必要があります。(通常、これらは /usr/local/mysql/bin にあります。)

最後に、MySQL データディレクトリ (通常は /var/lib/mysql または /usr/local/mysql/data) に場所を変更して、NDB ストレージエンジンを有効にするのに必要なオプションが my.cnf ファイルに含まれていることを確認します。

```
[mysqld]
ndbcluster
```

これで、MySQL サーバーを通常どおり起動できるようになりました。

```
shell> mysqld_safe --user=mysql &
```

しばらく待ってから、MySQL サーバーが適切に実行されていることを確認します。「mysql ended」という通知が表示された場合は、サーバーの .err ファイルをチェックして、どのような不具合があったかを調べます。

ここまで問題なく進んだ場合は、クラスタを使用し始めることができます。サーバーに接続して、NDBCLUSTER ストレージエンジンが有効になっていることを確認します。

```
shell> mysql
Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 1 to server version: 5.6.23

Type 'help;' or '\h' for help. Type '\c' to clear the buffer.

mysql> SHOW ENGINES\G
...
*************************** 12. row ***************************
Engine: NDBCLUSTER
Support: YES
Comment: Clustered, fault-tolerant, memory-based tables
*************************** 13. row ***************************
Engine: NDB
Support: YES
Comment: Alias for NDBCLUSTER
...
```

前の出力例に表示されている行番号は、サーバーの構成方法によっては、使用しているシステムで表示されるものと異なる可能性があります。

NDBCLUSTER テーブルを作成してみます。

```
shell> mysql
mysql> USE test;
Database changed

mysql> CREATE TABLE ctest (i INT) ENGINE=NDBCLUSTER;
Query OK, 0 rows affected (0.09 sec)

mysql> SHOW CREATE TABLE ctest \G
*************************** 1. row ***************************
       Table: ctest
Create Table: CREATE TABLE `ctest` (
  `i` int(11) default NULL
) ENGINE=ndbcluster DEFAULT CHARSET=latin1
1 row in set (0.00 sec)
```

ノードが適切にセットアップされたことをチェックするには、管理クライアントを起動します。

```
shell> ndb_mgm
```

クラスタのステータスに関するレポートを取得するには、管理クライアント内で SHOW コマンドを使用します。

```
ndb_mgm> SHOW
Cluster Configuration
---------------------
[ndbd(NDB)]     1 node(s)
id=2    @127.0.0.1  (Version: 5.6.22-ndb-7.4.4, Nodegroup: 0, *)

[ndb_mgmd(MGM)] 1 node(s)
id=1    @127.0.0.1  (Version: 5.6.22-ndb-7.4.4)

[mysqld(API)]   3 node(s)
```

```
id=3    @127.0.0.1  (Version: 5.6.22-ndb-7.4.4)
id=4 (not connected, accepting connect from any host)
id=5 (not connected, accepting connect from any host)
```

この時点で、機能する MySQL Cluster が正常にセットアップされました。これで、ENGINE=NDBCLUSTER またはその工イリアスである ENGINE=NDB を指定して作成したテーブルを使用して、クラスタにデータを格納できます。

18.3.2 MySQL Cluster の構成ファイル

MySQL Cluster を構成するには、2 つのファイルを操作する必要があります。

- my.cnf: MySQL Cluster のすべての実行可能ファイルのオプションを指定します。このファイルは、(これまでの MySQL の使用経験からご存じのとおり) クラスタ内で実行されている個々の実行可能ファイルからアクセスできる必要があります。
- config.ini: このファイルは、グローバル構成ファイルとも呼ばれますが、MySQL Cluster 管理サーバーからのみ読み取られます。管理サーバーに格納された情報は、クラスタに参加しているすべてのプロセスに配布されます。config.ini には、クラスタに関与する各ノードの記述が含まれています。これには、データノードに関する構成パラメータと、クラスタ内のすべてのノード間の接続に関する構成パラメータが含まれます。このファイルに含まれる可能性があるセクションと各セクションに配置される構成パラメータの種類を簡単に確認するには、「config.ini ファイルのセクション」を参照してください。

構成データのキャッシュ　MySQL Cluster NDB 7.3 以降では、NDB はステートフル構成を使用します。管理サーバーは、再起動されるたびにグローバル構成ファイルを読み取るのではなく、最初の起動時に構成をキャッシュし、その後は次の条件のいずれかが true である場合にのみグローバル構成ファイルを読み取ります。

- --initial オプションを使用すると管理サーバーが起動します　この場合は、グローバル構成ファイルが再度読み取られ、既存のキャッシュファイルが削除され、管理サーバーによって新しい構成キャッシュが作成されます。
- --reload オプションを使用すると管理サーバーが起動します　この場合は、管理サーバーのキャッシュとグローバル構成ファイルが比較されます。これらが異なる場合は、管理サーバーによって新しい構成キャッシュが作成されます。既存の構成キャッシュは保持されますが、使用されません。管理サーバーのキャッシュとグローバル構成ファイルに同じ構成データが含まれている場合は、既存のキャッシュが使用され、新しいキャッシュは作成されません。
- --config-cache オプションを使用すると管理サーバーが起動します　このオプションを使用すると、管理サーバーは構成キャッシュを完全にバイパスします。この場合、管理サーバーは存在する可能性がある構成ファイルを無視し、常に config.ini ファイルからその構成データを読み取ります。
- 構成キャッシュがみつかりません　この場合、管理サーバーはグローバル構成ファイルを読み取り、そのファイルと同じ構成データを含むキャッシュを作成します。

構成キャッシュファイル　管理サーバーは、デフォルトで MySQL インストールディレクトリ内の mysql-cluster という名前のディレクトリに構成キャッシュファイルを作成します。(Unix システムでソースから MySQL Cluster をビルドした場合、デフォルトの場所は /usr/local/mysql-cluster です。)これは、--configdir オプションを指定して管理サーバーを起動すると、実行時にオーバーライドできます。構成キャッシュファイルは、ndb_node_id_config.bin.seq_id というパターンで命名されるバイナリファイルです。node_id は管理サーバーのクラスタ内のノード ID で、seq_id はキャッシュの識別子です。キャッシュファイルには、作成された順に seq_id を使用して連番が付けられます。管理サーバーは、seq_id で特定される最新のキャッシュファイルを使用します。

> **注記**
>
> あとの構成キャッシュファイルを削除するか、seq_id が大きくなるように前のキャッシュファイルの名前を変更すると、前の構成にロールバックできます。ただし、構成キャッシュファイルはバイナリ形式で書き込まれるため、その内容を手動で編集しないでください。

MySQL Cluster 管理サーバーの --configdir、--config-cache、--initial、および --reload オプションの詳細は、セクション18.4.4「ndb_mgmd — MySQL Cluster 管理サーバーデーモン」を参照してください。

クラスタ構成の改良とこのプロセスを簡素化する試みを継続的に行なっています。下位互換性の維持に努めていますが、場合によっては互換性のない変更が行われる可能性があります。下位互換のない変更の場合には、クラスタのユーザーに事前に通知するように努めます。このような変更を発見し、それに関する情報が提供されていない場合は、セクション1.6「質問またはバグをレポートする方法」に示した手順を使用して MySQL バグデータベースに報告してください。

18.3.2.1 MySQL Cluster 構成の基本的な例

MySQL Cluster をサポートするには、次の例に示すように my.cnf を更新する必要があります。これらのパラメータを実行可能ファイルの起動時にコマンド行に指定することもできます。

> **注記**
>
> ここに示すオプションを、config.ini グローバル構成ファイルで使用されるものと混同しないようにしてください。グローバル構成オプションについては、このセクションで後述します。

```
# my.cnf
# example additions to my.cnf for MySQL Cluster
# (valid in MySQL 5.6)

# enable ndbcluster storage engine, and provide connection string for
# management server host (default port is 1186)
[mysqld]
ndbcluster
ndb-connectstring=ndb_mgmd.mysql.com


# provide connection string for management server host (default port: 1186)
[ndbd]
connect-string=ndb_mgmd.mysql.com

# provide connection string for management server host (default port: 1186)
[ndb_mgm]
connect-string=ndb_mgmd.mysql.com

# provide location of cluster configuration file
[ndb_mgmd]
config-file=/etc/config.ini
```

(接続文字列の詳細は、セクション18.3.2.3「MySQL Cluster の接続文字列」を参照してください。)

```
# my.cnf
# example additions to my.cnf for MySQL Cluster
# (will work on all versions)

# enable ndbcluster storage engine, and provide connection string for management
# server host to the default port 1186
[mysqld]
ndbcluster
ndb-connectstring=ndb_mgmd.mysql.com:1186
```

> **重要**
>
> 前に示したように、my.cnf ファイルの [mysqld] に NDBCLUSTER および ndb-connectstring パラメータを指定して mysqld プロセスを起動したあとは、クラスタを実際に起動しないと、CREATE TABLE または ALTER TABLE ステートメントを実行できません。そうでない場合、これらのステートメントはエラーで失敗します。これは意図的なものです。

クラスタの my.cnf ファイルでは、すべての実行可能ファイルによって読み取られ、使用される設定で別個の [mysql_cluster] セクションを使用することもできます。

```
# cluster-specific settings
[mysql_cluster]
ndb-connectstring=ndb_mgmd.mysql.com:1186
```

my.cnf ファイルに設定できる追加の NDB 変数については、セクション18.3.4.3「MySQL Cluster のシステム変数」を参照してください。

MySQL Cluster のグローバル構成ファイルには、慣例で config.ini という名前が付けられています (ただし、これは必須ではありません)。これは、必要に応じて ndb_mgmd が起動時に読み取るため、読み取りが可能な任意の場所に配置できます。構成の場所と名前は、コマンド行の ndb_mgmd で --config-file=path_name を使用して指定します。このオプションにデフォルト値はなく、ndb_mgmd で構成キャッシュを使用する場合、このオプションは無視されます。

MySQL Cluster のグローバル構成では、複数のセクションからなる INI 形式が使用されます。各セクションの先頭には (角括弧で囲まれた) セクション見出しが付き、そのあとに適切なパラメータ名と値が続きます。標準の INI 形式との違いの 1 つは、パラメータ名と値を等号 (「=」) だけでなくコロン (「:」) で区切ることもできることです (ただし、等号が推奨されます)。もう 1 つの違いは、セクションがセクション名で一意に識別されないことで

す。代わりに、一意のセクション (同じタイプの 2 つの異なるノードなど) はセクション内のパラメータとして指定された一意の ID で識別されます。

デフォルト値は、ほとんどのパラメータに定義されており、config.ini で指定することもできます。デフォルト値のセクションを作成するには、セクション名に default という単語を追加します。たとえば、[ndbd] セクションには特定のデータノードに適用されるパラメータが含まれていますが、[ndbd default] セクションにはすべてのデータノードに適用されるパラメータが含まれています。すべてのデータノードが同じデータメモリーサイズを使用するとします。これらすべてを構成するには、データメモリーサイズを指定する DataMemory 行を含む [ndbd default] セクションを作成します。

> **注記**
>
> MySQL Cluster の一部の古いリリースでは、NoOfReplicas のデフォルト値が存在しないため、常に [ndbd default] セクションに明示的にその値を指定する必要がありました。現在、このパラメータのデフォルト値はほとんどの一般的な使用シナリオでの推奨設定である 2 になっていますが、今後もこのパラメータを明示的に設定することが推奨されます。

グローバル構成ファイルでは、クラスタに関与するコンピュータとノード、およびノードをどのコンピュータに配置するかを定義する必要があります。1 つの管理サーバー、2 つのデータノード、および 2 つの MySQL サーバーで構成されるクラスタ用の簡単な構成ファイルの例をここに示します。

```
# file "config.ini" - 2 data nodes and 2 SQL nodes
# This file is placed in the startup directory of ndb_mgmd (the
# management server)
# The first MySQL Server can be started from any host. The second
# can be started only on the host mysqld_5.mysql.com

[ndbd default]
NoOfReplicas= 2
DataDir= /var/lib/mysql-cluster

[ndb_mgmd]
Hostname= ndb_mgmd.mysql.com
DataDir= /var/lib/mysql-cluster

[ndbd]
HostName= ndbd_2.mysql.com

[ndbd]
HostName= ndbd_3.mysql.com

[mysqld]
[mysqld]
HostName= mysqld_5.mysql.com
```

> **注記**
>
> 前の例は、MySQL Cluster に習熟するために必要な最小限の初期構成であり、本番環境の設定としては不十分です。完全な初期構成例については、セクション 18.3.2.2「MySQL Cluster の推奨される初期構成」を参照してください。

config.ini ファイルには、各ノードに固有のセクションがあります。たとえば、このクラスタには 2 つのデータノードがあるため、前に示した構成ファイルにはこれらのノードを定義する 2 つの [ndbd] セクションがあります。

> **注記**
>
> config.ini ファイルでは、セクション見出しと同じ行にコメントを配置しないでください。これを行うと、管理サーバーはこのような構成ファイルを解析できないため、起動しなくなります。

config.ini ファイルのセクション

次のリストで説明するように、config.ini 構成ファイルでは 6 つの異なるセクションを使用できます。

- [computer]: クラスタホストを定義します。これは、実行可能な MySQL Cluster の構成に必須ではありませんが、大規模なクラスタをセットアップするときに使用すると便利です。詳細は、セクション18.3.2.4「MySQL Cluster 内のコンピュータの定義」を参照してください。
- [ndbd]: クラスタデータノード (ndbd プロセス) を定義します。詳細は、セクション18.3.2.6「MySQL Cluster データノードの定義」を参照してください。

- [mysqld]: クラスタの MySQL サーバーノード (SQL ノードや API ノードとも呼ばれる) を定義します。SQL ノードの構成については、セクション18.3.2.7「MySQL Cluster 内の SQL ノードおよびその他の API ノードの定義」を参照してください。
- [mgm] または [ndb_mgmd]: クラスタの管理サーバー (MGM) ノードを定義します。管理ノードの構成については、セクション18.3.2.5「MySQL Cluster 管理サーバーの定義」を参照してください。
- [tcp]: TCP/IP がデフォルトの接続プロトコルであるクラスタノード間の TCP/IP 接続を定義します。通常、[tcp] または [tcp default] セクションは、クラスタが自動的に処理するため、MySQL Cluster をセットアップするときは必要ありません。ただし、状況によっては、クラスタが提供するデフォルトをオーバーライドするために必要になることがあります。使用可能な TCP/IP 構成パラメータと使用方法については、セクション18.3.2.8「MySQL クラスタの TCP/IP 接続」を参照してください。(場合によっては、セクション18.3.2.9「直接接続を使用する MySQL Cluster の TCP/IP 接続」も参考になる可能性があります。)
- [shm]: ノード間の共有メモリー接続を定義します。これは、MySQL 5.6 ではデフォルトで有効になっていますが、まだ実験的なものだと考えるようにしてください。SHM 相互接続については、セクション18.3.2.10「MySQL Cluster の共有メモリー接続」を参照してください。
- [sci]: クラスタデータノード間のスケーラブルコヒーラントインタフェース接続を定義します。このような接続には、無償で入手でき MySQL Cluster 配布には含まれないソフトウェアと、専用のハードウェアが必要です。SCI 相互接続の詳細は、セクション18.3.2.11「MySQL Cluster での SCI トランスポート接続」を参照してください。

セクションごとに default 値を定義できます。my.cnf または my.ini ファイルに指定されるパラメータと異なり、すべてのクラスタパラメータ名は大文字と小文字を区別しません。

18.3.2.2 MySQL Cluster の推奨される初期構成

MySQL Cluster で最良のパフォーマンスを実現できるかどうかは、次を含むいくつかの要因に依存します。

- MySQL Cluster ソフトウェアのバージョン
- データノードおよび SQL ノードの数
- ハードウェア
- オペレーティングシステム
- 格納されるデータの量
- クラスタ稼働時の負荷のサイズとタイプ

このため、最適な構成を獲得するプロセスは反復的なものになる可能性が高く、その結果は個々の MySQL Cluster 配備によって大きく異なる可能性があります。クラスタが動作するプラットフォームや MySQL Cluster のデータを使用するアプリケーションに変更が加えられたときは、構成の変更も必要になる可能性があります。これらの理由により、すべての使用シナリオに適した単一の構成を提示することは不可能です。ただし、このセクションでは推奨されるベース構成を示します。

初期の config.ini ファイル　MySQL Cluster NDB 7.3 以降を実行するクラスタ構成の開始点として、次の config.ini ファイルを推奨します。

```
# TCP PARAMETERS

[tcp default]
SendBufferMemory=2M
ReceiveBufferMemory=2M

# Increasing the sizes of these 2 buffers beyond the default values
# helps prevent bottlenecks due to slow disk I/O.

# MANAGEMENT NODE PARAMETERS

[ndb_mgmd default]
DataDir=path/to/management/server/data/directory

# It is possible to use a different data directory for each management
# server, but for ease of administration it is preferable to be
# consistent.

[ndb_mgmd]
HostName=management-server-A-hostname
# NodeId=management-server-A-nodeid

[ndb_mgmd]
```

```
HostName=management-server-B-hostname
# NodeId=management-server-B-nodeid

# Using 2 management servers helps guarantee that there is always an
# arbitrator in the event of network partitioning, and so is
# recommended for high availability. Each management server must be
# identified by a HostName. You may for the sake of convenience specify
# a NodeId for any management server, although one will be allocated
# for it automatically; if you do so, it must be in the range 1-255
# inclusive and must be unique among all IDs specified for cluster
# nodes.

# DATA NODE PARAMETERS

[ndbd default]
NoOfReplicas=2

# Using 2 replicas is recommended to guarantee availability of data;
# using only 1 replica does not provide any redundancy, which means
# that the failure of a single data node causes the entire cluster to
# shut down. We do not recommend using more than 2 replicas, since 2 is
# sufficient to provide high availability, and we do not currently test
# with greater values for this parameter.

LockPagesInMainMemory=1

# On Linux and Solaris systems, setting this parameter locks data node
# processes into memory. Doing so prevents them from swapping to disk,
# which can severely degrade cluster performance.

DataMemory=3072M
IndexMemory=384M

# The values provided for DataMemory and IndexMemory assume 4 GB RAM
# per data node. However, for best results, you should first calculate
# the memory that would be used based on the data you actually plan to
# store (you may find the ndb_size.pl utility helpful in estimating
# this), then allow an extra 20% over the calculated values. Naturally,
# you should ensure that each data node host has at least as much
# physical memory as the sum of these two values.

# ODirect=1

# Enabling this parameter causes NDBCLUSTER to try using O_DIRECT
# writes for local checkpoints and redo logs; this can reduce load on
# CPUs. We recommend doing so when using MySQL Cluster on systems running
# Linux kernel 2.6 or later.

NoOfFragmentLogFiles=300
DataDir=path/to/data/node/data/directory
MaxNoOfConcurrentOperations=100000

SchedulerSpinTimer=400
SchedulerExecutionTimer=100
RealTimeScheduler=1
# Setting these parameters allows you to take advantage of real-time scheduling
# of NDB threads to achieve increased throughput when using ndbd. They
# are not needed when using ndbmtd; in particular, you should not set
# RealTimeScheduler for ndbmtd data nodes.

TimeBetweenGlobalCheckpoints=1000
TimeBetweenEpochs=200
DiskCheckpointSpeed=10M
DiskCheckpointSpeedInRestart=100M
RedoBuffer=32M

# CompressedLCP=1
# CompressedBackup=1
# Enabling CompressedLCP and CompressedBackup causes, respectively, local
checkpoint files and backup files to be compressed, which can result in a space
savings of up to 50% over noncompressed LCPs and backups.

# MaxNoOfLocalScans=64
MaxNoOfTables=1024
MaxNoOfOrderedIndexes=256

[ndbd]
HostName=data-node-A-hostname
# NodeId=data-node-A-nodeid
```

```
LockExecuteThreadToCPU=1
LockMaintThreadsToCPU=0
# On systems with multiple CPUs, these parameters can be used to lock NDBCLUSTER
# threads to specific CPUs

[ndbd]
HostName=data-node-B-hostname
# NodeId=data-node-B-nodeid

LockExecuteThreadToCPU=1
LockMaintThreadsToCPU=0

# You must have an [ndbd] section for every data node in the cluster;
# each of these sections must include a HostName. Each section may
# optionally include a NodeId for convenience, but in most cases, it is
# sufficient to allow the cluster to allocate node IDs dynamically. If
# you do specify the node ID for a data node, it must be in the range 1
# to 48 inclusive and must be unique among all IDs specified for
# cluster nodes.

# SQL NODE / API NODE PARAMETERS

[mysqld]
# HostName=sql-node-A-hostname
# NodeId=sql-node-A-nodeid

[mysqld]

[mysqld]

# Each API or SQL node that connects to the cluster requires a [mysqld]
# or [api] section of its own. Each such section defines a connection
# 「slot」; you should have at least as many of these sections in the
# config.ini file as the total number of API nodes and SQL nodes that
# you wish to have connected to the cluster at any given time. There is
# no performance or other penalty for having extra slots available in
# case you find later that you want or need more API or SQL nodes to
# connect to the cluster at the same time.
# If no HostName is specified for a given [mysqld] or [api] section,
# then any API or SQL node may use that slot to connect to the
# cluster. You may wish to use an explicit HostName for one connection slot
# to guarantee that an API or SQL node from that host can always
# connect to the cluster. If you wish to prevent API or SQL nodes from
# connecting from other than a desired host or hosts, then use a
# HostName for every [mysqld] or [api] section in the config.ini file.
# You can if you wish define a node ID (NodeId parameter) for any API or
# SQL node, but this is not necessary; if you do so, it must be in the
# range 1 to 255 inclusive and must be unique among all IDs specified
# for cluster nodes.
```

推奨される SQL ノードの my.cnf オプション　MySQL Cluster SQL ノードとして機能する MySQL サーバーは、常にコマンド行または my.cnf に --ndbcluster および --ndb-connectstring オプションを指定して起動する必要があります。さらに、セットアップでほかのオプションを必要とする場合を除き、クラスタ内のすべての mysqld プロセスに次のオプションを設定してください。

- --ndb-use-exact-count=0
- --ndb-index-stat-enable=0
- --ndb-force-send=1
- --engine-condition-pushdown=1

18.3.2.3 MySQL Cluster の接続文字列

MySQL Cluster 管理サーバー (ndb_mgmd) を除き、MySQL Cluster の一部となる各ノードには、管理サーバーの場所を指定する接続文字列が必要です。この接続文字列は、管理サーバーとの接続を確立するときに使用されるほか、クラスタ内でのノードの役割によっては、ほかのタスクを実行するときにも使用されます。接続文字列の構文は次のとおりです。

```
[nodeid=node_id, ]host-definition[, host-definition[, ...]]

host-definition:
   host_name[:port_number]
```

node_id は、config.ini 内のノードを識別する 1 以上の整数です。host_name は、有効なインターネット名または IP アドレスを表す文字列です。port_number は、TCP/IP ポート番号を参照する整数です。

```
example 1 (long):   "nodeid=2,myhost1:1100,myhost2:1100,192.168.0.3:1200"
example 2 (short):  "myhost1"
```

何も指定されなかった場合は、デフォルトの接続文字列値として localhost:1186 が使用されます。接続文字列から port_num が省略された場合、デフォルトポートは 1186 です。このポートは、この目的のために IANA によって割り当てられたものであるため、ネットワーク上で常に使用可能です (詳細は、http://www.iana.org/assignments/port-numbersを参照してください)。

複数のホスト定義を列挙すると、複数の冗長管理サーバーを指定できます。MySQL Cluster のデータまたは API ノードは、正常な接続が確立されるまで、各ホスト上の連続する管理サーバーへの接続を指定された順序で試行します。

接続文字列には、管理サーバーに接続する複数のネットワークインタフェースを持つノードで使用される 1 つ以上のバインドアドレスを指定することもできます。バインドアドレスは、ホスト名またはネットワークアドレスと、オプションのポート番号で構成されます。この拡張された接続文字列の構文をここに示します。

```
[nodeid=node_id, ]
   [bind-address=host-definition, ]
   host-definition[; bind-address=host-definition]
   host-definition[; bind-address=host-definition]
   [, ...]]

host-definition:
   host_name[:port_number]
```

接続文字列で管理ホストを指定する前に 1 つのバインドアドレスを使用すると、そのアドレスは管理ホストのいずれかに接続するためのデフォルトとして使用されます (特定の管理サーバーにオーバーライドされた場合を除きます。例については、このセクションで後述します)。たとえば、次の接続文字列によって、このノードは接続先の管理サーバーに関係なく 192.168.178.242 を使用します。

```
bind-address=192.168.178.242, poseidon:1186, perch:1186
```

管理ホストの定義のあとにバインドアドレスを指定すると、そのアドレスはその管理ノードへの接続でのみ使用されます。次の接続文字列について考えます。

```
poseidon:1186;bind-address=localhost, perch:1186;bind-address=192.168.178.242
```

この場合、ノードは poseidon という名前のホストで実行されている管理サーバーに接続するために localhost を使用し、perch という名前のホストで実行されている管理サーバーに接続するために 192.168.178.242 を使用します。

デフォルトのバインドアドレスを指定してから、1 台以上の管理ホストにこのデフォルトをオーバーライドできます。次の例では、ホスト poseidon で実行されている管理サーバーに接続するために localhost が使用されます。192.168.178.242 は、最初に (どの管理サーバーの定義よりも前に) 指定されているため、デフォルトのバインドアドレスであり、ホスト perch および orca 上の管理サーバーに接続するために使用されます。

```
bind-address=192.168.178.242,poseidon:1186;bind-address=localhost,perch:1186,orca:2200
```

接続文字列を指定するには、いくつかの異なる方法があります。

- 各実行可能ファイルには、起動時に管理サーバーを指定できる固有のコマンド行オプションがあります。(各実行可能ファイルのドキュメントを参照してください。)
- 管理サーバーの my.cnf ファイルの [mysql_cluster] セクションに接続文字列を配置することで、クラスタ内のすべてのノードの接続文字列を同時に設定することもできます。
- 下位互換性のため、同じ構文を使用するオプションがほかに 2 つ用意されています。

  1. NDB_CONNECTSTRING 環境変数を設定して、接続文字列を含めます。
  2. 各実行可能ファイル用の接続文字列を Ndb.cfg という名前のテキストファイルに記述し、このファイルを実行可能ファイルの起動ディレクトリに配置します。

  ただし、これらは現在非推奨であり、新しいインストールに使用しないでください。

推奨される接続文字列の指定方法は、各実行可能ファイルのコマンド行または my.cnf ファイルでの設定です。

18.3.2.4 MySQL Cluster 内のコンピュータの定義

[computer] セクションは、システム内の各ノードのホスト名を定義しないで済む方法を提供する以外は、基本的には重要ではありません。ここに示すすべてのパラメータは必須です。

- Id

有効なバージョン	型/単位	デフォルト	範囲/値	再起動タイプ
NDB 7.3.0	文字列	[none]	...	IS

これは、構成ファイルに含まれるホストコンピュータを参照するために使用する一意の識別子です。

> **重要**
>
> コンピュータ ID は、管理、API、またはデータノードに使用するノード ID とは異なります。ノード ID の場合と異なり、config.ini ファイルの [computer] セクションでは、Id の代わりに NodeId を使用できません。

- HostName

有効なバージョン	型/単位	デフォルト	範囲/値	再起動タイプ
NDB 7.3.0	名前または IP アドレス	[none]	...	N

これは、コンピュータのホスト名または IP アドレスです。

18.3.2.5 MySQL Cluster 管理サーバーの定義

[ndb_mgmd] セクションは、管理サーバーの動作を構成するために使用されます。複数の管理サーバーが採用されている場合は、それらのすべてに共通するパラメータを [ndb_mgmd default] セクションに指定できます。[mgm] と [mgm default] はこれらの古いエイリアスで、下位互換性に対応しています。

次のリストに示すパラメータはすべてオプションであり、省略した場合はデフォルト値が使用されます。

> **注記**
>
> ExecuteOnComputer パラメータと HostName パラメータがどちらも存在しない場合は、デフォルト値の localhost が両方に使用されます。

- Id

有効なバージョン	型/単位	デフォルト	範囲/値	再起動タイプ
NDB 7.3.0	符号なし	[none]	1 - 255	IS

クラスタ内の各ノードは一意の ID を持っています。管理ノードの場合、これは 1-255 の (これらを含む) 範囲の整数値で表されます。この ID はすべての内部クラスタメッセージでノードを解決するために使用されるため、ノードのタイプに関係なく、各 MySQL Cluster ノードで一意である必要があります。

> **注記**
>
> データノードの ID は、49 未満にする必要があります。多数のデータノードを配備する予定の場合は、管理ノード (および API ノード) のノード ID を 48 より大きい値に制限することをお勧めします。

Id パラメータを使った管理ノードの識別は、NodeId を優先するため非推奨になりました。Id は、下位互換性に引き続き対応しますが、現在は警告を生成するようになっており、MySQL Cluster の今後のバージョンで削除される予定です。

- NodeId

有効なバージョン	型/単位	デフォルト	範囲/値	再起動タイプ
NDB 7.3.0	符号なし	[none]	1 - 255	IS

クラスタ内の各ノードは一意の ID を持っています。管理ノードでは、これは 1-255 の (これらを含む) 範囲の整数値で表されます。この ID はすべての内部クラスタメッセージでノードを解決するために使用されるため、ノードのタイプに関係なく、各 MySQL Cluster ノードで一意である必要があります。

> **注記**
>
> データノードの ID は、49 未満にする必要があります。多数のデータノードを配備する予定の場合は、管理ノード (および API ノード) のノード ID を 48 より大きい値に制限することをお勧めします。

NodeId は、管理ノードを識別時の使用が推奨されるパラメータ名です。古い Id は、下位互換性に引き続き対応しますが、現在は非推奨であり、使用時に警告を生成します。また、MySQL Cluster の今後のリリースで削除される予定です。

- ExecuteOnComputer

有効なバージョン	型/単位	デフォルト	範囲/値	再起動タイプ
NDB 7.3.0	name	[none]	...	S

これは、config.ini ファイルの [computer] セクションに定義されたいずれかのコンピュータに設定されている Id を参照します。

- PortNumber

有効なバージョン	型/単位	デフォルト	範囲/値	再起動タイプ
NDB 7.3.0	符号なし	1186	0 - 64K	N

これは、管理サーバーが構成要求と管理コマンドを待機するポートの番号です。

- HostName

有効なバージョン	型/単位	デフォルト	範囲/値	再起動タイプ
NDB 7.3.0	名前または IP アドレス	[none]	...	N

このパラメータを指定すると、管理ノードを配置するコンピュータのホスト名が定義されます。localhost 以外のホスト名を指定するには、このパラメータまたは ExecuteOnComputer のどちらかが必要です。

- LogDestination

有効なバージョン	型/単位	デフォルト	範囲/値	再起動タイプ
NDB 7.3.0	{CONSOLE|SYSLOG|FILE}	[テキストを参照]	...	N

このパラメータは、クラスタのロギング情報の送信先を指定します。これには CONSOLE、SYSLOG、FILE の 3 つのオプションがあり、FILE がデフォルトです。

- CONSOLE では、stdout にログが出力されます。

```
CONSOLE
```

- SYSLOG では、syslog 機能にログが送信されます。指定できる値は、auth、authpriv、cron、daemon、ftp、kern、lpr、mail、news、syslog、user、uucp、local0、local1、local2、local3、のいずれかです。

> **注記**
>
> すべてのオペレーティングシステムで必ずしもすべての機能がサポートされるとはかぎりません。

```
SYSLOG:facility=syslog
```

- FILE では、クラスタのログ出力が同じマシン上の通常のファイルにパイプされます。次の値を指定できます。
  - filename: ログファイルの名前。

    MySQL Cluster NDB 7.3 以降では、このような場合に使用されるデフォルトのログファイル名は ndb_nodeid_cluster.log です (一部の古いバージョンでは、filename を設定せずに FILE を指定した場合に使用されるデフォルトのログファイル名は logger.log でした)。
  - maxsize: ロギングが新しいファイルにロールオーバーされる前のファイルの最大サイズ (バイト単位)。これが発生すると、古いログファイルの名前が変更され、ファイル名の末尾に .N が追加されます (N はこの名前に対してまだ使用されていない次の番号です)。
  - maxfiles: ログファイルの最大数。

```
FILE:filename=cluster.log,maxsize=1000000,maxfiles=6
```

FILE パラメータのデフォルト値は、FILE:filename=ndb_node_id_cluster.log,maxsize=1000000,maxfiles=6 です (node_id はノードの ID です)。

ここに示すように、複数のログの保存先をセミコロンで区切って指定できます。

```
CONSOLE;SYSLOG:facility=local0;FILE:filename=/var/log/mgmd
```

- ArbitrationRank

有効なバージョン	型/単位	デフォルト	範囲/値	再起動タイプ
NDB 7.3.0	0-2	1	0 - 2	N

このパラメータは、アービトレータとして機能するノードを定義するために使用されます。アービトレータにできるのは、管理ノードと SQL ノードだけです。ArbitrationRank には次のいずれかの値を指定できます。

- 0: このノードはアービトレータとして使用できません。
- 1: このノードは高い優先度を持っており、優先度の低いノードより優先的にアービトレータになります。
- 2: 優先度の高いノードがこの目的で使用できない場合にのみアービトレータとして使用される、優先度の低いノードを示します。

通常は、管理サーバーの ArbitrationRank を 1 (管理ノードのデフォルト) に設定し、すべての SQL ノードを 0 (SQL ノードのデフォルト) に設定することにより、管理サーバーをアービトレータとして構成してください。

すべての管理および SQL ノードで ArbitrationRank を 0 に設定するか、config.ini グローバル構成ファイルの [ndbd default] セクションに Arbitration パラメータを設定することで、アービトレーションを完全に無効化できます。Arbitration を設定すると、ArbitrationRank の設定はすべて無視されます。

- ArbitrationDelay

有効なバージョン	型/単位	デフォルト	範囲/値	再起動タイプ
NDB 7.3.0	ミリ秒	0	0 - 4294967039 (0xFFFFFEFF)	N

アービトレーション要求に対する管理サーバーの応答をミリ秒数だけ遅らせるための整数値。デフォルトでは、この値は 0 です。通常、これを変更する必要はありません。

- DataDir

有効なバージョン	型/単位	デフォルト	範囲/値	再起動タイプ
NDB 7.3.0	パス	.	...	N

これは、管理サーバーの出力ファイルを配置するディレクトリを指定します。これらのファイルには、クラスタのログファイル、プロセスの出力ファイル、およびデーモンのプロセス ID (PID) ファイルが含まれます。(ログファイルでは、LogDestination の FILE パラメータをこのセクションで前述したように設定することで、この場所をオーバーライドできます。)

このパラメータのデフォルト値は、ndb_mgmd が配置されているディレクトリです。

- PortNumberStats

有効なバージョン	型/単位	デフォルト	範囲/値	再起動タイプ
NDB 7.3.0	符号なし	[none]	0 - 64K	N

このパラメータは、MySQL Cluster 管理サーバーから統計情報を取得するために使用されるポート番号を指定します。デフォルト値はありません。

- Wan

有効なバージョン	型/単位	デフォルト	範囲/値	再起動タイプ
NDB 7.3.0	ブール	false	true、false	N

WAN の TCP 設定をデフォルトとして使用します。

- HeartbeatThreadPriority

有効なバージョン	型/単位	デフォルト	範囲/値	再起動タイプ
NDB 7.3.0	文字列	[none]	...	S

管理および API ノードのハートビートスレッドのスケジューリングポリシーと優先度を設定します。

このパラメータを設定するための構文をここに示します。

```
HeartbeatThreadPriority = policy[, priority]

policy:
  {FIFO | RR}
```

このパラメータを設定するときは、ポリシーを指定する必要があります。これは、FIFO (先入れ先出し) または RR (ラウンドロビン) のいずれかです。オプションで、ポリシー値のあとに優先度 (整数) を指定できます。

- TotalSendBufferMemory

有効なバージョン	型/単位	デフォルト	範囲/値	再起動タイプ
NDB 7.3.0	バイト	256K	0 - 4294967039 (0xFFFFFEFF)	N

このパラメータは、すべての構成済みトランスポータ間で共有される送信バッファーメモリーの、このノードに割り当てられるメモリー合計量を決定するために使用されます。

このパラメータを設定する場合、許容される最小値は 256K バイト、最大値は 4294967039 です。TotalSendBufferMemory の動作と使用、および送信バッファーメモリーパラメータの構成の詳細は、セクション18.3.2.12「MySQL Cluster の送信バッファーパラメータの構成」を参照してください。

- HeartbeatIntervalMgmdMgmd

有効なバージョン	型/単位	デフォルト	範囲/値	再起動タイプ
NDB 7.3.0	ミリ秒	1500	100 - 4294967039 (0xFFFFFEFF)	N
NDB 7.3.3	ミリ秒	1500	100 - 4294967039 (0xFFFFFEFF)	N

別の管理ノードがこの管理ノードに接続しているかどうかを判定するために使用されるハートビートメッセージの間隔を指定します。管理ノードは、この間隔を 3 回待ったあとで接続の切断を宣言します。このため、デフォルト設定の 1500 ミリ秒では、管理ノードはおよそ 1600 ミリ秒待ってからタイムアウトします。

このパラメータは MySQL Cluster NDB 7.3.3 で追加されました。(Bug #16426805)

> 注記
>
> 管理ノードの構成を変更したあとは、新しい構成を有効にするためにクラスタのローリング再起動を実行する必要があります。
>
> 実行中の MySQL Cluster に新しい管理サーバーを追加するには、既存の config.ini ファイルを変更したあとで、すべてのクラスタノードのローリング再起動を実行する必要もあります。複数の管理ノードを使用するときに発生する問題の詳細は、セクション18.1.6.10「複数の MySQL Cluster ノードに関する制限」を参照してください。

18.3.2.6 MySQL Cluster データノードの定義

クラスタのデータノードの動作を構成するには、[ndbd] および [ndbd default] セクションを使用します。

データノードのプロセスとして ndbd バイナリと ndbmtd バイナリのどちらを使用する場合でも、セクション名としては常に [ndbd] と [ndbd default] が使用されます。

バッファーサイズ、プールサイズ、タイムアウトなどを制御する数多くのパラメータがあります。唯一の必須パラメータは、ExecuteOnComputer または HostName のいずれかです。これは、ローカルの [ndbd] セクションで定義する必要があります。

NoOfReplicas パラメータは、すべてのクラスタデータノードに共通であるため、[ndbd default] セクションに定義してください。NoOfReplicas を設定する必要は必ずしもありませんが、これを明示的に設定することをお勧めします。

ほとんどのデータノードパラメータは [ndbd default] セクションに設定します。[ndbd] セクションでの変更が許可されるのは、ローカル値の設定を明示的に規定されているパラメータだけです。HostName、NodeId、および ExecuteOnComputer (存在する場合) は、config.ini のほかのセクションではなく、ローカルの [ndbd] で定義する必要があります。つまり、これらのパラメータの設定は 1 つのデータノードに固有のものです。

メモリー使用状況やバッファーサイズに影響を与えるパラメータでは、1024、1024 * 1024、または 1024 * 1024 * 1024 の単位を示すサフィクスとして K、M、または G を使用できます。(たとえば、100K は 100 * 1024 = 102400 を意味します。)現在のところ、パラメータの名前と値では大文字と小文字を区別します。

MySQL Cluster ディスクデータのテーブルに固有の構成パラメータについては、このセクションで後述します (ディスクデータの構成パラメータを参照してください)。

これらすべてのパラメータは、ndbmtd (ndbd のマルチスレッドバージョン) にも適用されます。追加の 3 つのデータノード構成パラメータ (MaxNoOfExecutionThreads、ThreadConfig、および NoOfFragmentLogParts) は、ndbmtd にのみ適用されます。これらを ndbd に使用しても、無効になります。詳細は、マルチスレッドの構成パラメータ (ndbmtd)を参照してください。セクション18.4.3「ndbmtd — MySQL Cluster データノードデーモン (マルチスレッド)」も参照してください。

データノードの識別　NodeId または Id の値 (つまり、データノードの識別子) は、ノード起動時のコマンド行、または構成ファイルで割り当てることができます。

- Id

有効なバージョン	型/単位	デフォルト	範囲/値	再起動タイプ
NDB 7.3.0	符号なし	[none]	1 - 48	IS

一意のノード ID は、クラスタのすべての内部メッセージでノードのアドレスとして使用されます。データノードでは、これは 1-48 の (これらを含む) 範囲の整数です。クラスタ内の各ノードは、一意の識別子を持っている必要があります。

NodeId は、データノードの識別時での使用が推奨されるパラメータ名です。古い Id は、下位互換性に引き続き対応しますが、現在は非推奨であり、使用時に警告を生成します。Id は MySQL Cluster の今後のリリースで削除される予定です。

- NodeId

有効なバージョン	型/単位	デフォルト	範囲/値	再起動タイプ
NDB 7.3.0	符号なし	[none]	1 - 48	IS

一意のノード ID は、クラスタのすべての内部メッセージでノードのアドレスとして使用されます。データノードでは、これは 1-48 の (これらを含む) 範囲の整数です。クラスタ内の各ノードは、一意の識別子を持っている必要があります。

NodeId は、データノードの識別時での使用が推奨されるパラメータ名です。Id は、下位互換性に引き続き対応しますが、現在は非推奨であり、使用時に警告を生成します。また、MySQL Cluster の今後のバージョンで削除される予定です。

- ExecuteOnComputer

有効なバージョン	型/単位	デフォルト	範囲/値	再起動タイプ
NDB 7.3.0	name	[none]	...	S

これは、[computer] セクションに定義されたいずれかのコンピュータに設定されている Id を参照します。

- HostName

有効なバージョン	型/単位	デフォルト	範囲/値	再起動タイプ
NDB 7.3.0	名前または IP アドレス	localhost	...	N

このパラメータを指定すると、データノードを配置するコンピュータのホスト名が定義されます。localhost 以外のホスト名を指定するには、このパラメータまたは ExecuteOnComputer のどちらかが必要です。

- ServerPort

有効なバージョン	型/単位	デフォルト	範囲/値	再起動タイプ
NDB 7.3.0	符号なし	[none]	1 - 64K	N

クラスタ内の各ノードは、ほかのノードに接続するためにポートを使用します。デフォルトでは、このポートは同じホストコンピュータ上の 2 つのノードで同じポート番号を受け取らないよう動的に割り当てられるため、通常はこのパラメータの値を指定する必要はありません。

ただし、データノードと API ノード (SQL ノードを含む) 間の通信を許可するため、ファイアウォールで特定のポートを開く必要がある場合は、config.ini ファイルの [ndbd] セクションまたは (複数のデータノードに対してこれを行う必要がある場合は) [ndbd default] セクションでこのパラメータを必要なポートの番号に設定すると、SQL ノード、API ノード、またはその両方からの着信接続のためにその番号のポートを開くことができます。

> **注記**
>
> データノードから管理ノードへの接続は ndb_mgmd 管理ポート (管理サーバーの PortNumber。セクション18.3.2.5「MySQL Cluster 管理サーバーの定義」を参照してください) を使用して行われるため、データノードからポートへの送信接続は常に許可されます。

- TcpBind_INADDR_ANY

  このパラメータを TRUE または 1 に設定すると、IP_ADDR_ANY がバインドされ、任意の場所から接続できるようになります (自動生成接続の場合)。デフォルトは FALSE (0) です。

- NodeGroup

有効なバージョン	型/単位	デフォルト	範囲/値	再起動タイプ
NDB 7.3.0		[none]	0 - 65536	IS

このパラメータを使用して、データノードを特定のノードグループに割り当てることができます。これはクラスタを最初に起動したときは読み取り専用であり、オンラインでデータノードを別のノードグループに再割り当てすることはできません。config.ini ファイルの [ndbd default] セクションでこのパラメータを使用することは、通常望ましくありません。また、ノードをノードグループに割り当てるときは、ノードグループに無効な数のノードが割り当てられないように注意する必要があります。

NodeGroup パラメータの主な用途は、ローリング再起動を実行せずに実行中の MySQL Cluster に新しいノードグループを追加することです。そのためには、これを 65536 (最大値) に設定します。NodeGroup の値は、すべてのクラスタデータノードではなく、あとで起動され、新しいノードグループとしてクラスタに追加されるノードにのみ設定する必要があります。詳細は、セクション18.5.13.3「MySQL Cluster データノードのオンライン追加: 詳細な例」を参照してください。

- NoOfReplicas

有効なバージョン	型/単位	デフォルト	範囲/値	再起動タイプ
NDB 7.3.0	整数	2	1 - 4	IS

このグローバルパラメータは、[ndbd default] セクションでのみ設定でき、クラスタに格納されている各テーブルのレプリカの数を定義します。このパラメータは、ノードグループのサイズも指定します。ノードグループは、すべてが同じ情報を格納するノードのセットです。

ノードグループは暗黙的に形成されます。最初のノードグループはノード ID がもっとも小さいデータノードのセットで形成され、次のノードグループはノード ID が次に小さいデータノードのセットで形成されます (以下同様)。例として、4 つのデータノードがあり、NoOfReplicas が 2 に設定されているとします。4 つのデータノードのノード ID はそれぞれ 2、3、4、および 5 です。この場合、1 つ目のノードグループはノード 2 および 3 で形成され、2 つ目のノードグループはノード 4 および 5 で形成されます。1 つのハードウェアの障害によっ

てクラスタ全体が機能停止するため、同じノードグループ内のノードが同じコンピュータに配置されないようにクラスタを構成することが重要です。

ノード ID が指定されていない場合は、データノードの順序がノードグループの決定要因になります。明示的な割り当てが行われたかどうかに関係なく、管理クライアントの SHOW コマンドの出力に表示されます。

NoOfReplicas のデフォルト値は 2 です。これは、ほとんどの一般的な使用シナリオで推奨される設定です。

指定可能な最大値は 4 ですが、現時点で実際にサポートされている値は 1 と 2 だけです。

> **重要**
>
> NoOfReplicas を 1 に設定することは、すべてのクラスタデータのコピーが 1 つだけ存在することを意味します。この場合は、そのデータに格納されているデータの追加のコピーが存在しないため、1 つのデータノードの喪失によってクラスタが機能停止します。

このパラメータの値は、クラスタ内のデータノードの数で均等に割れる必要があります。たとえば、データノードが 2 つの場合、2/3 と 2/4 はどちらも小数値になるため、NoOfReplicas は 1 または 2 である必要があります。データノードが 4 つの場合、NoOfReplicas は 1、2、または 4 である必要があります。

- DataDir

有効なバージョン	型/単位	デフォルト	範囲/値	再起動タイプ
NDB 7.3.0	パス	.	...	IN

このパラメータは、トレースファイル、ログファイル、PID ファイル、およびエラーログが配置されるディレクトリを指定します。

デフォルトはデータノードプロセスの作業ディレクトリです。

- FileSystemPath

有効なバージョン	型/単位	デフォルト	範囲/値	再起動タイプ
NDB 7.3.0	パス	DataDir	...	IN

このパラメータは、メタデータ用に作成されるすべてのファイル、Redo ログ、Undo ログ (ディスクデータテーブル用)、およびデータファイルが配置されるディレクトリを指定します。デフォルトは、DataDir で指定されたディレクトリです。

> **注記**
>
> このディレクトリは、ndbd プロセスが開始される前に存在している必要があります。

MySQL Cluster の推奨されるディレクトリ階層には /var/lib/mysql-cluster が含まれており、この下にノードのファイルシステム用のディレクトリが作成されます。このサブディレクトリの名前にはノード ID が含まれます。たとえば、ノード ID が 2 の場合、このサブディレクトリの名前は ndb_2_fs です。

- BackupDataDir

有効なバージョン	型/単位	デフォルト	範囲/値	再起動タイプ
NDB 7.3.0	パス	[テキストを参照]	...	IN

このパラメータは、バックアップが配置されるディレクトリを指定します。

> **重要**
>
> この値の末尾には常に '/BACKUP' という文字列が追加されます。たとえば、BackupDataDir の値を /var/lib/cluster-data に設定すると、すべてのバックアップが /var/lib/cluster-data/BACKUP の下に格納されます。これは、事実上のデフォルトのバックアップ場所が FileSystemPath パラメータで指定された場所の下にある BACKUP という名前のディレクトリであることも意味しています。

データメモリー、インデックスメモリー、および文字列メモリー

DataMemory と IndexMemory は、実際のレコードとインデックスの格納に使用されるメモリーセグメントのサイズを指定する [ndbd] パラメータです。これらの値を設定するときは、DataMemory と IndexMemory がどのように使用されるかを理解することが重要です。これらは、通常、クラスタでの実際の使用状況を反映して更新する必要があるためです。

- DataMemory

有効なバージョン	型/単位	デフォルト	範囲/値	再起動タイプ
NDB 7.3.0	バイト	80M	1M - 1024G	N

このパラメータは、データベースレコードの格納に使用できるスペースの量 (バイト単位) を定義します。この値によって指定される量の全体がメモリー内に割り当てられるため、それに対応できるだけの十分な物理メモリーがマシンに搭載されていることが非常に重要です。

DataMemory によって割り当てられたメモリーは、実際のレコードとインデックスの両方を格納するために使用されます。各レコードには 16 バイトのオーバーヘッドがあります。各レコードは 128 バイトのページオーバーヘッドを含む 32K バイトのページに格納されるため、レコードごとに追加の量が発生します (次を参照してください)。また、各レコードが 1 つのページのみに格納されるため、ページごとに少量の無駄が発生します。

可変サイズのテーブル属性については、DataMemory から割り当てられた別個のデータページにデータが格納されます。可変長レコードは、可変サイズ部分を参照するための 4 バイトの追加オーバーヘッドを含む固定サイズ部分を使用します。可変サイズ部分には、2 バイトのオーバーヘッドと属性あたり 2 バイトが含まれます。

最大レコードサイズは 14000 バイトです。

DataMemory によって定義されたメモリースペースは、順序付けされたインデックスを格納するためにも使用されます。これには、レコードあたり約 10 バイトが使用されます。各テーブル行は順序付けされたインデックスで表されます。ユーザーの間では共通して、すべてのインデックスが IndexMemory によって割り当てられたメモリーに格納されるという誤解がありますが、これは正しくありません。主キーと一意のハッシュインデックスのみがこのメモリーを使用します。順序付けされたインデックスは、DataMemory によって割り当てられたメモリーを使用します。ただし、主キーまたは一意のハッシュインデックスを作成するときに、インデックス作成ステートメントに USING HASH を指定しなかった場合は、同じキーに対して順序付けされたインデックスも作成されます。これは、管理クライアントで ndb_desc -d db_name table_name を実行すると確認できます。

現在、MySQL Cluster ではハッシュインデックスにパーティションあたり最大 512M バイトを使用できます。つまり、場合によってはndb_mgm -e "ALL REPORT MEMORYUSAGE" で DataMemory に大きい空き領域が示されていてもTable is fullというエラーがMySQL クライアントアプリケーションで表示されることがあります。このため、データの負荷が高いノードでデータノードを再起動するときに問題が発生する場合もあります。CREATE TABLE に MAX_ROWS オプションを使用すると、MySQL Cluster のテーブルのために追加のパーティションを作成して、ハッシュインデックスに使用できるメモリーを増やすように NDB に強制できます。通常、テーブルに格納することが予期されている行数の 2 倍の数値を MAX_ROWS に設定すれば十分です。MinFreePct 構成パラメータを使用して、ノード再起動の問題を回避することもできます。(Bug #13436216)

DataMemory によって割り当てられるメモリースペースは 32K バイトのページで構成され、テーブルのフラグメントに割り当てられます。各テーブルは、通常、クラスタ内のデータノードと同じ数のフラグメントにパーティション化されます。このため、各ノードには NoOfReplicas での設定と同じ数のフラグメントがあります。

ページの割り当て後、テーブルを削除する以外の方法で空きページのプールに戻すことは、現在不可能です。(これは、特定のページに割り当てられた DataMemory ページをほかのテーブルで使用できないことも意味します。)データノードのリカバリを実行すると、すべてのレコードがほかの稼働しているノードから空のパーティションに挿入されるため、パーティションも圧縮されます。

DataMemory メモリースペースには、Undo 情報も含まれています。更新のたびに、変更されていないレコードのコピーが DataMemory に割り当てられます。順序付けされたテーブルインデックスにも各コピーへの参照が含まれています。一意のハッシュインデックスは、一意のインデックスカラムが更新されたときだけ更新されます。その場合は、コミット時にインデックステーブル内の新しいエントリが挿入され、古いエントリが削除されます。このため、クラスタを使用するアプリケーションによって実行される大規模なトランザクションを処理できるだけの十分なメモリーを割り当てる必要もあります。いずれにしても、次の理由から、少数の大規

模なトランザクションを実行するよりも、多数の小規模なトランザクションを使用する方がメリットがあります。

- 大規模なトランザクションは小規模なトランザクションより高速に実行できません
- 大規模なトランザクションでは、トランザクション障害の発生時に消失して繰り返す必要がある操作の数が多くなります
- 大規模なトランザクションはより多くのメモリーを使用します

DataMemory のデフォルト値は 80M バイトです。最小値は 1M バイトです。最大サイズはありませんが、実際には、制限に達したときにプロセスがスワップを開始しないように最大サイズを決定する必要があります。この制限は、マシン上の使用可能な物理 RAM 量と、オペレーティングシステムが 1 つのプロセスに振り向けることができるメモリー量によって決まります。32 ビットオペレーティングシステムでは通常プロセスあたり 2-4G バイトに制限されますが、64 ビットオペレーティングシステムではより多くのメモリーを使用できます。このため、大規模なデータベースでは 64 ビットオペレーティングシステムを使用することをお勧めします。

- IndexMemory

有効なバージョン	型/単位	デフォルト	範囲/値	再起動タイプ
NDB 7.3.0	バイト	18M	1M - 1T	N

このパラメータは、MySQL Cluster 内のハッシュインデックスに使用されるストレージの量を制御します。ハッシュインデックスは、常に主キーのインデックス、一意のインデックス、および一意の制約に使用されます。主キーまたは一意のインデックスを定義すると、2 つのインデックスが作成され、そのうちの 1 つのハッシュインデックスが、すべてのタプルへのアクセスとロックの処理に使用されます。このインデックスは、一意の制約を適用するためにも使用されます。

ハッシュインデックスのサイズは、この式を使用して見積もることができます。

```
size  = ( (fragments * 32K) + (rows * 18) )
* replicas
```

fragments はフラグメントの数、replicas はレプリカの数 (通常 2)、rows は行の数です。テーブルに 100 万個の行、8 個のフラグメント、2 つのレプリカが含まれる場合、予想されるインデックスのメモリー使用量は、ここに示すように計算されます。

```
((8 * 32K) + (1000000 * 18)) * 2 = ((8 * 32768) + (1000000 * 18)) * 2
= (262144 + 18000000) * 2
= 18262144 * 2 = 36524288 bytes = ~35MB
```

MySQL Cluster NDB 7.2 以降では、順序付けされたインデックスのインデックス統計が (有効になっている場合は) mysql.ndb_index_stat_sample テーブルに格納されます。このテーブルにはハッシュインデックスがあるため、これによってインデックスのメモリー使用量が追加されます。特定の順序付けされたインデックスの行数の上限は、次のように計算できます。

```
sample_size= key_size + ((key_attributes + 1) * 4)

sample_rows = IndexStatSaveSize
          * ((0.01 * IndexStatSaveScale * log2(rows * sample_size)) + 1)
          / sample_size
```

前の式で、key_size は順序付けされたインデックスキーのサイズ (バイト単位)、key_attributes は順序付けされたインデックスキー内の属性の数、rows はベーステーブルの行数です。

テーブル t1 に、100 万個の行と、2 つの 4 バイト整数に対する ix1 という名前の順序付けされたインデックスがあるとします。また、IndexStatSaveSize と IndexStatSaveScale がこのデフォルト値 (それぞれ 32K および 100) に設定されているとします。前の 2 つの式を使用して、次のように計算できます。

```
sample_size = 8  + ((1 + 2) * 4) = 20 bytes

sample_rows = 32K
          * ((0.01 * 100 * log2(1000000*20)) + 1)
          / 20
          = 32768 * ( (1 * ~16.811) +1) / 20
          = 32768 * ~17.811 / 20
```

```
            = ~29182 rows
```

予想されるインデックスのメモリー使用量は 2 * 18 * 29182 = 約 1050550 バイトです。

IndexMemory のデフォルト値は 18M バイトです。最小は 1M バイトです。

- StringMemory

有効なバージョン	型/単位	デフォルト	範囲/値	再起動タイプ
NDB 7.3.0	% またはバイト	25	0 - 4294967039 (0xFFFFFEFF)	S

このパラメータは、テーブル名などの文字列用に割り当てられるメモリー量を決定するもので、config.ini ファイルの [ndbd] または [ndbd default] セクションに指定されます。0 から 100 までの (これらを含む) 値は、デフォルトの最大値のパーセントとして解釈されます。デフォルトの最大値は、テーブルの数、テーブル名のサイズ、.FRM ファイルの最大サイズ、MaxNoOfTriggers、カラム名の最大サイズ、およびデフォルトの最大カラム値を含むいくつかの係数に基づいて計算されます。

100 を超える値は、バイト数として解釈されます。

デフォルト値は 25 (つまり、デフォルトの最大の 25%) です。

ほとんどの状況ではデフォルト値で十分ですが、クラスタテーブルの数が非常に多い (1000 個以上) 場合は、エラー 773 Out of string memory, please modify StringMemory config parameter: Permanent error: Schema errorが発生する可能性があるので、その場合はこの値を増やすようにしてください。25 (25%) であれば、多すぎることはなく、もっとも極端な状況を除き、このエラーは繰り返し発生しません。

次の例は、1 つのテーブルでメモリーがどのように使用されるかを示しています。このテーブル定義について考えます。

```
CREATE TABLE example (
  a INT NOT NULL,
  b INT NOT NULL,
  c INT NOT NULL,
  PRIMARY KEY(a),
  UNIQUE(b)
) ENGINE=NDBCLUSTER;
```

各レコードに、12 バイトのデータと 12 バイトのオーバーヘッドがあります。NULL 可能カラムがない場合は、4 バイトのオーバーヘッドを節約できます。さらに、カラム a および b に対して 2 つの順序付けされたインデックスがあり、レコードごとにおよそ 10 バイトが消費されます。ベーステーブルには主キーのハッシュインデックスがあり、レコードあたりおよそ 29 バイトが使用されます。b を主キーとし、a をカラムとする別のテーブルによって一意の制約が実装されています。この別のテーブルでは、8 バイトのレコードデータと 12 バイトのオーバーヘッドに加えて、example テーブル内のレコードあたり 29 バイトのインデックスメモリーが追加で消費されます。

したがって、100 万個のレコードに対して、主キーと一意の制約のハッシュインデックスを処理するには 58M バイトのインデックスメモリーが必要です。さらに、ベーステーブルのレコード、一意のインデックステーブル、および 2 つの順序付けされたインデックステーブルに 64M バイトが必要です。

ハッシュインデックスがかなりのメモリー量を占有しますが、データへのアクセスは非常に高速になります。これらは、MySQL Cluster では一意制約を処理するためにも使用されます。

現在、唯一のパーティション化アルゴリズムはハッシュ化であり、順序付けされたインデックスは各ノードでローカルに使用されます。したがって、一般的なケースで順序付けされたインデックスを使用して一意制約を処理することはできません。

IndexMemory と DataMemory の両方で重要な点は、各ノードグループのすべてのデータメモリーとインデックスメモリーの合計がデータベースの合計サイズになることです。各ノードグループはレプリケートされた情報の格納に使用されるため、2 つのレプリカを含むノードが 4 つある場合は、2 つのノードグループが作成されます。したがって、使用可能なデータメモリーの合計はデータノードごとに 2 * DataMemory です。

すべてのノードで DataMemory と IndexMemory を同じ値に設定することを強くお勧めします。データの分布はクラスタ内のすべてのノードで均等であるため、各ノードで使用できるスペースの最大量はクラスタ内でもっとも小さいノードの最大量と同じになります。

DataMemory と IndexMemory は変更できますが、これらのいずれかを減らすのは危険です。その場合、ノードまたは MySQL Cluster 全体がメモリースペースの不足によって再起動できなくなる可能性が高くなります。これらの値を増やすことは問題ありませんが、このようなアップグレードはソフトウェアのアップグレードと同じ方

法で行うことをお勧めします。つまり、最初に構成ファイルを更新し、次に管理サーバーを起動して、各データノードを順に再起動します。

MinFreePct　DataMemory と IndexMemory を含むデータノードリソースの一定割合 (デフォルトでは 5%) は、データノードが再起動するときにそのメモリーを使い果たさないように予備として残されます。これは、MinFreePct データノード構成パラメータを使用して調整できます (デフォルトは 5)。

有効なバージョン	型/単位	デフォルト	範囲/値	再起動タイプ
NDB 7.3.0	符号なし	5	0 - 100	N

更新によって、使用されるインデックスメモリーの量は増えません。挿入はただちに有効になりますが、行の削除はトランザクションがコミットされるまで実際には行われません。

トランザクションパラメータ　次に説明する [ndbd] のいくつかのパラメータは、システムで処理できる並列トランザクションの数とトランザクションのサイズに影響を与えるという点で重要です。MaxNoOfConcurrentTransactions は、ノード内で実行できる並列トランザクションの数を設定します。MaxNoOfConcurrentOperations は、同時に更新フェーズに入ったり、ロックしたりできるレコードの数を設定します。

これらのパラメータはどちらも (特に MaxNoOfConcurrentOperations は)、デフォルト値を使用せずに特定の値を設定するユーザーのターゲットになる可能性があります。デフォルト値は、小規模なトランザクションを使用するシステムで過大なメモリーが使用されないように設定されています。

MaxDMLOperationsPerTransaction は、特定のトランザクションで実行できる DML 操作の最大数を設定します。

- MaxNoOfConcurrentTransactions

有効なバージョン	型/単位	デフォルト	範囲/値	再起動タイプ
NDB 7.3.0	整数	4096	32 - 4294967039 (0xFFFFFEFF)	N

各クラスタデータノードには、クラスタ内の個々のアクティブなトランザクションに対するトランザクションレコードが必要です。トランザクションを調整するタスクは、すべてのデータノードに分配されます。クラスタ内のトランザクションレコードの合計数は、特定のノード内のトランザクションの数にクラスタ内のノードの数を掛けた値です。

トランザクションレコードは、個々の MySQL サーバーに割り当てられます。MySQL サーバーへの個々の接続には、少なくとも 1 つのトランザクションレコードと、その接続でアクセスされるテーブルごとに追加のトランザクションオブジェクトが必要です。つまり、クラスタ内のトランザクション合計数の妥当な最小値を次のように表すことができます。

```
MinTotalNoOfConcurrentTransactions =
    (maximum number of tables accessed in any single transaction + 1)
    * number of SQL nodes
```

クラスタを使用する 10 個の SQL ノードがあるとします。10 個のテーブルが関与する 1 回の結合には、11 個のトランザクションレコードが必要です。トランザクション内にこのような結合が 10 回ある場合、このトランザクションには MySQL サーバーあたり 10 * 11 = 110 個のトランザクションレコード (または合計 110 * 10 = 1100 個のトランザクションレコード) が必要です。各データノードは、MinTotalNoOfConcurrentTransactions / データノード数が処理されます。この場合、4 個のデータノードを含む MySQL Cluster では、各データノードの MaxNoOfConcurrentTransactions を 1100/4 = 275 に設定することになります。さらに、1 つのノードグループですべての並列トランザクションに対応できるようにすることで、障害リカバリを提供します。つまり、各データノードの MaxNoOfConcurrentTransactions を、MinTotalNoOfConcurrentTransactions/ノードグループ数と同じ数のトランザクションに対応できるだけの十分な数に設定します。このクラスタにノードグループが 1 つだけある場合は、MaxNoOfConcurrentTransactions を 1100 (クラスタ全体の並列トランザクションの数と同じ) に設定するようにしてください。

また、各トランザクションに少なくとも 1 つの操作が関与します。このため、MaxNoOfConcurrentTransactions に設定する値は常に MaxNoOfConcurrentOperations の値以下にします。

このパラメータは、すべてのクラスタデータノードで同じ値に設定する必要があります。これは、データノードの機能停止時に、残存するもっとも古いノードが機能停止したノードで実行されていたトランザクションのトランザクション状態を再作成するためです。

ローリング再起動を使用してこの値を変更できますが、実行中のクラスタ上のトラフィック量は、発生するトランザクションが新旧どちらか低い方のレベルを超えない程度にする必要があります。

デフォルト値は 4096 です。

- MaxNoOfConcurrentOperations

有効なバージョン	型/単位	デフォルト	範囲/値	再起動タイプ
NDB 7.3.0	整数	32K	32 - 4294967039 (0xFFFFFEFF)	N

このパラメータの値は、トランザクションのサイズと数に合わせて調整することをお勧めします。少数の操作とレコードしか関与しないトランザクションを実行する場合は、通常、このパラメータのデフォルト値で十分です。多数のレコードが関与する大規模なトランザクションを実行する場合は、通常、その値を増やす必要があります。

クラスタデータを更新する各トランザクションでは、トランザクションコーディネータと実際の更新が実行されるノードの両方でレコードが保持されます。これらのレコードには、ロールバック、ロックキュー、およびその他の目的に使用する Undo レコードを見つけるために必要な状態情報が含まれています。

このパラメータは、最低でも、トランザクションで同時に更新されるレコードの数をクラスタデータノードの数で割った値に設定するようにしてください。たとえば、4 つのデータノードがあるクラスタで、トランザクションを使用して 100 万件の並列更新を処理することが予想される場合は、この値を 1000000/4 = 250000 に設定します。障害からの回復力を提供できるように、このパラメータを、個々のデータノードでノードグループの負荷を処理できる十分な大きさの値に設定することをお勧めします。つまり、並列操作の合計数/ノードグループの数と同じ値に設定するようにしてください。(ノードグループが 1 つだけの場合、これはクラスタ全体の並列操作の合計数と同じです。)

各トランザクションには常に少なくとも 1 つの操作が関与しているため、MaxNoOfConcurrentOperations の値は常に MaxNoOfConcurrentTransactions の値以上にします。

ロックを設定する読み取りクエリーでも、操作レコードが作成されます。ノード間の分布が完全でない場合に対応できるように、個々のノード内にある程度の追加スペースが割り当てられます。

クエリーで一意のハッシュインデックスが使用されると、実際にはトランザクション内のレコードあたり 2 つの操作レコードが使用されます。1 つ目のレコードはインデックステーブルの読み取りを表し、2 つ目はベーステーブルに対する操作を扱います。

デフォルト値は 32768 です。

このパラメータは、実際には別個に構成できる 2 つの値を扱います。これらのうち 1 つ目は、トランザクションコーディネータによって配置される操作レコードの数を指定します。2 つ目の部分は、データベースにローカルに格納される操作レコードの数を指定します。

8 ノードのクラスタで実行される非常に大規模なトランザクションでは、そのトランザクションに関与する読み取り、更新、および削除と同じ数の操作レコードがトランザクションコーディネータ内に必要です。ただし、これらの操作レコードは 8 個のノードすべてに分散しています。このため、1 つの非常に大規模なトランザクション用にシステムを構成する必要がある場合は、この 2 つの部分を別個に構成することをお勧めします。MaxNoOfConcurrentOperations は常に、ノードのトランザクションコーディネータ部分の操作レコード数を計算するために使用されます。

操作レコードのメモリー要件を考慮することも重要です。これらは、レコードあたり約 1K バイトを消費します。

- MaxNoOfLocalOperations

有効なバージョン	型/単位	デフォルト	範囲/値	再起動タイプ
NDB 7.3.0	整数	UNDEFINED	32 - 4294967039 (0xFFFFFEFF)	N

デフォルトでは、このパラメータは 1.1 * MaxNoOfConcurrentOperations として計算されます。これは、多くの同時トランザクションを含み、そのいずれもが大規模ではないシステムに適合します。一度に 1 つの非常に大規模なトランザクションが実行され、多数のノードがある場合は、このパラメータを明示的に指定してデフォルトをオーバーライドすることをお勧めします。

- MaxDMLOperationsPerTransaction

有効なバージョン	型/単位	デフォルト	範囲/値	再起動タイプ
NDB 7.3.0	操作 (DML)	4294967295	32 - 4294967295	N

このパラメータは、トランザクションのサイズを制限します。これを超える数の DML 操作が必要になった場合、トランザクションは中止されます。トランザクションあたりの最小操作数は 32 ですが、MaxDMLOperationsPerTransaction を 0 に設定すると、トランザクションあたりの DML 操作の数に対する制限を無効にできます。最大 (およびデフォルト) は 4294967295 です。

トランザクションの一時ストレージ　次の一連の [ndbd] パラメータは、クラスタトランザクションの一部であるステートメントの実行時に一時ストレージを決定するために使用されます。このステートメントが完了し、クラスタがコミットまたはロールバックを待機しているときに、すべてのレコードが解放されます。

これらのパラメータのデフォルト値は、ほとんどの状況に適しています。ただし、多数の行または操作が関与するトランザクションをサポートする必要がある場合は値を増やして、システム内の並列性を高める必要があることがあります。一方、比較的小規模なトランザクションを必要とするアプリケーションでは、これらの値を減らしてメモリーを節約できます。

- MaxNoOfConcurrentIndexOperations

有効なバージョン	型/単位	デフォルト	範囲/値	再起動タイプ
NDB 7.3.0	整数	8K	0 - 4294967039 (0xFFFFFEFF)	N

一意のハッシュインデックスを使用するクエリーでは、クエリーの実行フェーズで別の一時的な操作レコードのセットが使用されます。このパラメータは、そのレコードプールのサイズを設定します。このため、このレコードはクエリーの一部を実行している間だけ割り当てられます。この部分の実行が完了した直後にレコードは解放されます。中止とコミットを処理するために必要な状態では通常の操作レコードによって処理され、プールサイズは MaxNoOfConcurrentOperations パラメータで設定されます。

このパラメータのデフォルト値は 8192 です。一意のハッシュインデックスを使用して非常に高い並列性を実現させるまれなケースでのみ、この値を増やす必要があります。クラスタに高いレベルの並列性が必要ないことを DBA が確信している場合は、小さい値を使用してメモリーを節約できます。

- MaxNoOfFiredTriggers

有効なバージョン	型/単位	デフォルト	範囲/値	再起動タイプ
NDB 7.3.0	整数	4000	0 - 4294967039 (0xFFFFFEFF)	N

MaxNoOfFiredTriggers のデフォルト値は 4000 です。ほとんどの状況では、これで十分です。場合によっては、クラスタに並列性はそれほど必要ないと DBA が確信している場合は、値を減らすこともできます。

一意のハッシュインデックスに影響を与える操作が実行されたときに、レコードが作成されます。一意のハッシュインデックスを使用してテーブル内のレコードを挿入または削除したり、一意のハッシュインデックスの一部であるカラムを更新したりすると、インデックステーブルで挿入または削除が発生します。生成されたレコードは、このインデックステーブル操作を発生させた元の操作の完了を待機している間、インデックステーブル操作を表すために使用されます。この操作は短期間ですが、一意のハッシュインデックスのセットを含むベーステーブルに対して多数の並列書き込み操作が行われる状況では、レコードプール内に多数のレコードを必要とする可能性があります。

- TransactionBufferMemory

有効なバージョン	型/単位	デフォルト	範囲/値	再起動タイプ
NDB 7.3.0	バイト	1M	1K - 4294967039 (0xFFFFFEFF)	N

このパラメータの影響を受けるメモリーは、インデックステーブルの更新と一意のインデックスの読み取り時に発生した操作を追跡するために使用されます。このメモリーは、これらの操作のキーおよびカラムの情報を格納するために使用されます。このパラメータの値をデフォルトから変更する必要があるのは、非常にまれなケースだけです。

TransactionBufferMemory のデフォルト値は 1M バイトです。

通常の読み取りおよび書き込み操作でも同じようなバッファーが使用されますが、その使用期間はさらに短くなります。コンパイル時のパラメータ ZATTRBUF_FILESIZE (ndb/src/kernel/blocks/Dbtc/Dbtc.hpp にあります) は 4000 * 128 バイト (500K バイト) に設定されています。キー情報用の同様のバッファー ZDATABUF_FILESIZE (これも Dbtc.hpp にあります) には、4000 * 16 = 62.5K バイトのバッファースペースが含まれています。Dbtc は、トランザクションのコーディネーションを扱うモジュールです。

スキャンとバッファリング　(ndb/src/kernel/blocks/Dblqh/Dblqh.hpp 内の) Dblqh モジュールには、読み取りと更新に影響を与える追加の [ndbd] パラメータがあります。これらには、デフォルトで 10000 * 128 バイト (1250K バイト) に設定される ZATTRINBUF_FILESIZE と、デフォルトで 10000 * 16 バイト (およそ 156K バイト) のバッファースペースに設定される ZDATABUF_FILE_SIZE が含まれます。これまでに、これらのコンパイル時制限のいずれかを増やすことを示すユーザーからの報告または弊社の大規模なテストの結果はありません。

- MaxNoOfConcurrentScans

有効なバージョン	型/単位	デフォルト	範囲/値	再起動タイプ
NDB 7.3.0	整数	256	2 - 500	N

このパラメータは、クラスタ内で実行できる並列スキャンの数を制御するために使用されます。各トランザクションコーディネータは、このパラメータで定義された数の並列スキャンを処理できます。各スキャンクエリーは、すべてのパーティションを並列でスキャンして実行されます。各パーティションスキャンでは、パーティションが配置されているノード内のスキャンレコードが使用されます。レコードの数は、このパラメータの値にノードの数を掛けた値です。クラスタは、クラスタ内のすべてのノードで同時に発生した MaxNoOfConcurrentScans 個のスキャンを維持できます。

スキャンは、実際には 2 つのケースで実行されます。これらのうち 1 つ目のケースは、クエリーを処理するためのハッシュまたは順序付けされたインデックスが存在しないときに発生します。この場合は、フルテーブルスキャンを実行するとクエリーが実行されます。2 つ目のケースは、クエリーをサポートするハッシュインデックスが存在せず、順序付けされたインデックスが存在する場合に発生します。順序付けされたインデックスの使用は、並列の範囲スキャンの実行を意味します。この順序はローカルのパーティションでのみ維持されるため、すべてのパーティションでインデックススキャンを実行する必要があります。

MaxNoOfConcurrentScans のデフォルト値は 256 です。最大値は 500 です。

- MaxNoOfLocalScans

有効なバージョン	型/単位	デフォルト	範囲/値	再起動タイプ
NDB 7.3.0	整数	[テキストを参照]	32 - 4294967039 (0xFFFFFEFF)	N

多くのスキャンが完全に並列化されていない場合の、ローカルスキャンレコードの数を指定します。ローカルスキャンレコードの数が指定されていない場合は、ここに示すように計算されます。

```
4 * MaxNoOfConcurrentScans * [# data nodes] + 2
```

最小値は 32 です。

- BatchSizePerLocalScan

有効なバージョン	型/単位	デフォルト	範囲/値	再起動タイプ
NDB 7.3.0	整数	256	1 - 992	N

このパラメータは、並列スキャン操作を処理するために使用されるロックレコードの数を計算するために使用されます。

BatchSizePerLocalScan は、SQL ノードで定義される BatchSize と深い関係があります。

- LongMessageBuffer

有効なバージョン	型/単位	デフォルト	範囲/値	再起動タイプ
NDB 7.3.0	バイト	64M	512K - 4294967039 (0xFFFFFEFF)	N
NDB 7.3.1	バイト	4M	512K - 4294967039 (0xFFFFFEFF)	N

有効なバージョン	型/単位	デフォルト	範囲/値	再起動タイプ
NDB 7.3.5	バイト	64M	512K - 4294967039 (0xFFFFFEFF)	N

これは、個々のノード内およびノード間でメッセージを渡すために使用される内部バッファーです。デフォルトは 64M バイトです。(MySQL Cluster NDB 7.3.5 より前では、これは 4M バイトでした。)

ほとんどの場合、このパラメータをデフォルトから変更する必要はありません。

- MaxParallelScansPerFragment

有効なバージョン	型/単位	デフォルト	範囲/値	再起動タイプ
NDB 7.3.0	バイト	256	1 - 4294967039 (0xFFFFFEFF)	N

順次処理のキューイングが開始される前に許可される並列スキャン (TUP スキャンおよび TUX スキャン) の最大数を構成できます。これを増やすことにより、多数のスキャンを並列で実行するときに未使用の CPU を利用して、パフォーマンスを向上させることができます。

MySQL Cluster NDB 7.3 以降では、このパラメータのデフォルト値は 256 です。

メモリー割り当て

MaxAllocate

有効なバージョン	型/単位	デフォルト	範囲/値	再起動タイプ
NDB 7.3.0	符号なし	32M	1M - 1G	N

これは、テーブル用のメモリーを割り当てるときに使用するメモリーユニットの最大サイズです。NDB で Out of memoryエラーが発生したが、クラスタログまたは DUMP 1000 (DUMP 1000を参照してください) の出力を調べると、使用可能なすべてのメモリーがまだ使用されていないことが明らかな場合は、このパラメータ (または MaxNoOfTables、あるいはその両方) の値を増やすと、NDB が十分なメモリーを使用できるようになります。

ハッシュマップサイズ

DefaultHashMapSize

有効なバージョン	型/単位	デフォルト	範囲/値	再起動タイプ
NDB 7.3.0	LDM スレッド	3840	0 - 3840	N

MySQL Cluster NDB 7.2.7 以降では、前のリリース (240) より大きなテーブルハッシュマップのデフォルトサイズ (3840) が使用されます。MySQL Cluster NDB 7.2.11 以降では、このパラメータを使用して NDB が使用するテーブルハッシュマップのサイズを構成できます (以前は、この値はハードコーディングされていました)。DefaultHashMapSize には、3 つの値 (0、240、3840) のいずれかを指定できます。これらの値とその効果について、次の表で説明します。

値	説明/効果
0	クラスタ内のすべてのデータおよび API ノードの中で、このパラメータに設定されたもっとも小さい値を (あれば) 使用します。どのデータまたは API ノードにも設定されていない場合は、デフォルト値を使用します。
240	MySQL Cluster NDB 7.2.7 より前のすべての MySQL Cluster リリースでデフォルトで使用されていた元のハッシュマップサイズです。
3840	MySQL Cluster NDB 7.2.7 以降でデフォルトで使用される、より大きなハッシュマップサイズ

このパラメータの主な用途は、より大きなハッシュマップサイズ (3840) がデフォルトである MySQL Cluster NDB 7.2.7 以降の MySQL Cluster バージョンと以前のリリース (デフォルトが 240 だった) との間のアップグレードと (特に) ダウングレードを容易にすることです。これは、この変更に下位互換性がない (Bug #14800539) ためです。このパラメータを 240 に設定してから、この値を使用している古いバージョンからのアップグレードを実行すると、クラスタはテーブルハッシュマップに対して引き続き小さいサイズを使用するため、アップグレード後のテーブルでも前のバージョンとの互換性が維持されます。DefaultHashMapSize は個々のデータノード、API ノード、またはその両方で設定できますが、config.ini ファイルの [ndbd default] セクションに一度だけ設定する方法をお勧めします。

このパラメータを増やしたあと、既存のテーブルで新しいサイズを利用するには、そのテーブルに対して ALTER TABLE ... REORGANIZE PARTITION を実行します。その後、そのテーブルでより大きなハッシュマップサイズを使用できるようになります。これは、ローリング再起動の実行に加えて行います。ローリング再起動を実行すると、より大きなハッシュマップを新しいテーブルで使用できるようになりますが、既存のテーブルでは使用できません。

DefaultHashMapSize を 3840 に設定してテーブルを作成または変更したあと、オンラインでこのパラメータを減らす方法は、現在サポートされていません。

ロギングとチェックポイント　次の [ndbd] パラメータは、ログとチェックポイントの動作を制御します。

- NoOfFragmentLogFiles

有効なバージョン	型/単位	デフォルト	範囲/値	再起動タイプ
NDB 7.3.0	整数	16	3 - 4294967039 (0xFFFFFEFF)	IN

このパラメータは、ノードの Redo ログファイルの数 (つまり、Redo ログギングに割り当てられるスペースの量) を設定します。Redo ログファイルはリング状に編成されるため、セット内の最初と最後のログファイル (それぞれ「head」および「tail」ログファイルとも呼ばれる) が交わらないことが非常に重要です。これらが互いに近寄りすぎると、新しいログレコードを追加する余裕がないため、ノードは更新を含むすべてのトランザクションを中止し始めます。

Redo ログレコードは、挿入されたあと、必要な数のローカルチェックポイントが完了するまで削除されません。(MySQL Cluster NDB 7.3 以降では、2 つのローカルチェックポイントのみが必要です)。チェックポイントの頻度は、この章で別途説明する固有の構成パラメータセットによって決定されます。

デフォルトのパラメータ値は 16 です。これは、デフォルトでは 16M バイトのファイル 4 個が 16 セット (合計 1024M バイト) あることを意味します。個々のログファイルのサイズは、FragmentLogFileSize パラメータを使用して構成できます。非常に多くの更新が必要なシナリオでは、Redo ログに対して十分なスペースを提供するため、NoOfFragmentLogFiles の値を 300 以上に設定する必要がある場合があります。

チェックポイントが遅く、データベースへの書き込みが多いためにログファイルがいっぱいになって、リカバリに悪影響を与えずにログの末尾をカットできない場合は、すべての更新トランザクションが内部エラーコード 410「Out of log file space temporarily」によって中止されます。この状態は、チェックポイントが完了してログの末尾を前に移動できるようになるまで続きます。

> **重要**
>
> このパラメータは「実行中に」変更できません。--initial を使用してノードを再起動する必要があります。実行中のクラスタのすべてのデータノードでこの値を変更する必要がある場合は、(各データノードの起動時に --initial を使用して) ノードのローリング再起動を使用します。

- FragmentLogFileSize

有効なバージョン	型/単位	デフォルト	範囲/値	再起動タイプ
NDB 7.3.0	バイト	16M	4M - 1G	IN

このパラメータを設定すると、Redo ログファイルのサイズを直接制御できます。これは、MySQL Cluster が負荷の高い状態で稼働し、フラグメントログファイルを閉じるまでに時間がかかって新しいファイルを開けない (一度に開けるフラグメントログファイルは 2 つだけです) 場合に便利です。フラグメントログファイルのサイズを増やすと、クラスタ内で新しいフラグメントログファイルを開く必要が生じるまでの時間が長くなります。このパラメータのデフォルト値は 16M です。

フラグメントログファイルの詳細は、NoOfFragmentLogFiles の説明を参照してください。

- InitFragmentLogFiles

有効なバージョン	型/単位	デフォルト	範囲/値	再起動タイプ
NDB 7.3.0	[値を参照]	SPARSE	SPARSE、FULL	IN

デフォルトでは、データノードの初期起動の実行時はフラグメントログファイルがまばらに作成されます。つまり、使用するオペレーティングシステムとファイルシステムによって、必ずしもすべてのバイトがディスクに書き込まれるわけではありません。ただし、このパラメータを使用すると、使用されているプラットフォー

ムやファイルシステムのタイプに関係なく、この動作をオーバーライドしてすべてのバイトが強制的に書き込まれるように設定できます。InitFragmentLogFiles は 2 つの値のいずれかを取ります。

- SPARSE。フラグメントログファイルがまばらに作成されます。これはデフォルト値です。
- FULL。フラグメントログファイルのすべてのバイトが強制的にディスクに書き込まれます。

オペレーティングシステムとファイルシステムによっては、InitFragmentLogFiles=FULL を設定することで、Redo ログへの書き込み時の I/O エラーを解消できる場合があります。

- MaxNoOfOpenFiles

有効なバージョン	型/単位	デフォルト	範囲/値	再起動タイプ
NDB 7.3.0	符号なし	0	20 - 4294967039 (0xFFFFFEFF)	N

このパラメータは、開いたファイルに割り当てる内部スレッド数の上限を設定します。このパラメータの変更が必要な状況が発生した場合は、それをバグとして報告するようにしてください。

デフォルト値は 0 です。ただし、このパラメータに設定できる最小値は 20 です。

- InitialNoOfOpenFiles

有効なバージョン	型/単位	デフォルト	範囲/値	再起動タイプ
NDB 7.3.0	ファイル	27	20 - 4294967039 (0xFFFFFEFF)	N

このパラメータは、開いたファイルに割り当てる初期の内部スレッド数を設定します。

デフォルト値は 27 です。

- MaxNoOfSavedMessages

有効なバージョン	型/単位	デフォルト	範囲/値	再起動タイプ
NDB 7.3.0	整数	25	0 - 4294967039 (0xFFFFFEFF)	N

このパラメータは、古いファイルを上書きする前に保持されるトレースファイルの最大数を設定します。トレースファイルは、何らかの理由でノードがクラッシュしたときに生成されます。

デフォルトは 25 個のトレースファイルです。

- MaxLCPStartDelay

有効なバージョン	型/単位	デフォルト	範囲/値	再起動タイプ
NDB 7.3.0	秒	0	0 - 600	N

データノードの並列リカバリでは、実際にはテーブルデータのみが並列でコピーされ、同期されます。ディクショナリ情報やチェックポイント情報などのメタデータの同期は順次に行われます。また、ディクショナリおよびチェックポイント情報のリカバリは、ローカルチェックポイントの実行と並列で実行できません。つまり、多くのデータノードを同時に起動したり再起動したりすると、データノードがローカルチェックポイントの実行中に待たされて、ノードのリカバリ時間が長くなる可能性があります。

より多くの (場合によってはすべての) データノードでメタデータの同期を完了できるように、ローカルチェックポイントの遅延を強制できます。各データノードでメタデータの同期が完了したあとは、ローカルチェックポイントの実行中でも、すべてのデータノードで並列にテーブルデータをリカバリできます。このような遅延を強制するには、MaxLCPStartDelay を設定します。これは、データノードでメタデータの同期が継続されている間にクラスタがローカルチェックポイントの開始を待機する秒数を決定します。このパラメータは、すべてのデータノードで同じになるように config.ini ファイルの [ndbd default] セクションに設定するようにしてください。最大値は 600、デフォルトは 0 です。

- LcpScanProgressTimeout

有効なバージョン	型/単位	デフォルト	範囲/値	再起動タイプ
NDB 7.3.0	second	60	0 - 4294967039 (0xFFFFFEFF)	N
NDB 7.3.3	second	60	0 - 4294967039 (0xFFFFFEFF)	N

ローカルチェックポイントのフラグメントスキャンウォッチドッグは、ローカルチェックポイントの一部として実行されている各フラグメントスキャンが進行していないかどうか定期的にチェックし、特定の時間が経過しても進行しない場合はそのノードをシャットダウンします。MySQL Cluster NDB 7.3.3 より前では、この間隔は常に 60 秒でした (Bug #16630410)。MySQL Cluster NDB 7.3.3 以降では、LcpScanProgressTimeout データノード構成パラメータを使用してこの間隔を設定できます。これは、LCP のフラグメントスキャンウォッチドッグがノードをシャットダウンするまでローカルチェックポイントの停滞が許可される最大時間を設定します。

デフォルト値は 60 秒です (以前のリリースと互換性があります)。このパラメータを 0 に設定すると、LCP のフラグメントスキャンウォッチドッグが完全に無効化されます。

メタデータオブジェクト　次の一連の [ndbd] パラメータは、インデックス、イベント、およびクラスタ間のレプリケーションで使用される属性、テーブル、インデックス、およびトリガーオブジェクトの最大数を定義するために使用されるメタデータオブジェクトのプールサイズを定義します。これらはクラスタに対する「推奨」の役割を果たすだけであり、指定されなかったものはすべて示されたデフォルト値に戻ります。

- MaxNoOfAttributes

有効なバージョン	型/単位	デフォルト	範囲/値	再起動タイプ
NDB 7.3.0	整数	1000	32 - 4294967039 (0xFFFFFEFF)	N

このパラメータは、クラスタに定義できる属性の推奨最大数を設定します。MaxNoOfTables と同様、厳密な上限の役割を果たすためのものではありません。

(古い MySQL Cluster リリースでは、このパラメータは特定の操作に対する厳密な制限として扱われることがありました。このため、MySQL Cluster レプリケーションでレプリケートできる数を超えるテーブルを作成したときに問題が発生し、MaxNoOfAttributes を超える数の属性を作成した (状況によってはできない) ときに混乱を招くことがありました。)

デフォルト値は 1000 です。指定可能な最小値は 32 です。最大は 4294967039 です。すべてのメタデータがサーバー上で完全にレプリケートされるため、各属性はノードあたり 200 バイト前後のストレージを消費します。

MaxNoOfAttributes を設定するときは、今後実行する可能性がある ALTER TABLE ステートメントを事前に用意することが重要です。これは、クラスタテーブルで ALTER TABLE の実行中に、元のテーブルに含まれる 3 倍の数の属性が使用されるためであり、この数の 2 倍を許可することをお勧めします。たとえば、属性の数がもっとも多い (greatest_number_of_attributes) MySQL Cluster テーブルに 100 個の属性がある場合、MaxNoOfAttributes の適切な初期値は 6 * greatest_number_of_attributes = 600 です。

また、テーブルあたりの平均属性数を見積もり、それに MaxNoOfTables を掛けることもお勧めします。この値が前の段落で得られた値より大きい場合は、大きい値を使用するようにしてください。

必要なすべてのテーブルが問題なく作成できると仮定して、パラメータの構成後に実際に ALTER TABLE を試行し、この数が十分であることを確認するようにしてください。これが成功しなかった場合は、MaxNoOfAttributes をさらに MaxNoOfTables の数だけ増やして、再度テストしてください。

- MaxNoOfTables

有効なバージョン	型/単位	デフォルト	範囲/値	再起動タイプ
NDB 7.3.0	整数	128	8 - 20320	N

テーブルオブジェクトは、クラスタ内のテーブルごと、および一意のハッシュインデックスごとに割り当てられます。このパラメータは、クラスタ全体のテーブルオブジェクトの推奨最大数を設定します。MaxNoOfAttributes と同様、厳密な上限の役割を果たすためのものではありません。

(古い MySQL Cluster リリースでは、このパラメータは特定の操作に対する厳密な制限として扱われることがありました。このため、MySQL Cluster レプリケーションでレプリケートできる数を超えるテーブルを作成したときに問題が発生し、MaxNoOfTables を超える数のテーブルを作成した (状況によってはできない) ときに混乱を招くことがありました。)

BLOB データ型を持つ各属性では、BLOB データの大部分を格納するために追加のテーブルが使用されます。テーブルの合計数を定義するときは、これらのテーブルも考慮に入れる必要があります。

このパラメータのデフォルト値は 128 です。最小値は 8、最大値は 20320 です。各テーブルオブジェクトは、ノードあたりおよそ 20K バイトを消費します。

> **注記**
>
> MaxNoOfTables、MaxNoOfOrderedIndexes、および MaxNoOfUniqueHashIndexes の合計が 2^{32} - 2 (4294967294) を超えないようにする必要があります。

- MaxNoOfOrderedIndexes

有効なバージョン	型/単位	デフォルト	範囲/値	再起動タイプ
NDB 7.3.0	整数	128	0 - 4294967039 (0xFFFFFEFF)	N

クラスタ内の順序付けされたインデックスごとに、インデックスされた内容とそのストレージセグメントを記述するオブジェクトが割り当てられます。デフォルトでは、そのように定義された各インデックスによって、順序付けされたインデックスも定義されます。個々の一意のインデックスおよび主キーには、順序付けされたインデックスとハッシュインデックスの両方が含まれています。MaxNoOfOrderedIndexes は、システム内で一度に使用できる順序付けされたインデックスの合計数を設定します。

このパラメータのデフォルト値は 128 です。各インデックスオブジェクトは、ノードあたりおよそ 10K バイトのデータを消費します。

> **注記**
>
> MaxNoOfTables、MaxNoOfOrderedIndexes、および MaxNoOfUniqueHashIndexes の合計が 2^{32} - 2 (4294967294) を超えないようにする必要があります。

- MaxNoOfUniqueHashIndexes

有効なバージョン	型/単位	デフォルト	範囲/値	再起動タイプ
NDB 7.3.0	整数	64	0 - 4294967039 (0xFFFFFEFF)	N

主キー以外の個々の一意のインデックスには、一意キーをインデックスされたテーブルの主キーにマップする専用のテーブルが割り当てられます。デフォルトでは、個々の一意のインデックスに対して順序付けされたインデックスも定義されます。これを禁止するには、一意のインデックスを定義するときに USING HASH オプションを指定する必要があります。

デフォルト値は 64 です。各インデックスは、ノードあたりおよそ 15K バイトを消費します。

> **注記**
>
> MaxNoOfTables、MaxNoOfOrderedIndexes、および MaxNoOfUniqueHashIndexes の合計が 2^{32} - 2 (4294967294) を超えないようにする必要があります。

- MaxNoOfTriggers

有効なバージョン	型/単位	デフォルト	範囲/値	再起動タイプ
NDB 7.3.0	整数	768	0 - 4294967039 (0xFFFFFEFF)	N

個々の一意のハッシュインデックスには、内部更新、挿入、および削除のトリガーが割り当てられます。(これは、一意のハッシュインデックスごとに 3 つのトリガーが作成されることを意味します。)ただし、順序付け

されたインデックスに必要なトリガーオブジェクトは 1 つだけです。バックアップでも、クラスタ内の通常のテーブルごとに 3 つのトリガーオブジェクトが使用されます。

クラスタ間のレプリケーションでも、内部トリガーが使用されます。

このパラメータは、クラスタ内のトリガーオブジェクトの最大数を設定します。

デフォルト値は 768 です。

- MaxNoOfIndexes

  このパラメータは非推奨であり、MySQL Cluster の今後のバージョンで削除される予定です。代わりに MaxNoOfOrderedIndexes および MaxNoOfUniqueHashIndexes を使用するようにしてください。

  このパラメータは、一意のハッシュインデックスでのみ使用されます。　クラスタ内に定義された一意のハッシュインデックスごとに 1 つのレコードがこのプールに存在している必要があります。

  このパラメータのデフォルト値は 128 です。

- MaxNoOfSubscriptions

有効なバージョン	型/単位	デフォルト	範囲/値	再起動タイプ
NDB 7.3.0	符号なし	0	0 - 4294967039 (0xFFFFFEFF)	N

  MySQL Cluster 内の各 NDB テーブルには、NDB カーネル内のサブスクリプションが必要です。一部の NDB API アプリケーションでは、このパラメータの変更が必要または望ましい場合があります。ただし、SQL ノードとして機能する MySQL サーバーを通常どおりに使用する場合は、これを行う必要はありません。

  MaxNoOfSubscriptions のデフォルト値は 0 です。これは、MaxNoOfTables と等価として扱われます。各サブスクリプションは 108 バイトを消費します。

- MaxNoOfSubscribers

有効なバージョン	型/単位	デフォルト	範囲/値	再起動タイプ
NDB 7.3.0	符号なし	0	0 - 4294967039 (0xFFFFFEFF)	N

  このパラメータは、MySQL Cluster レプリケーションの使用時にのみ関連します。デフォルト値は 0 です。これは、2 * MaxNoOfTables として扱われます。つまり、2 つの MySQL サーバー (1 つはレプリケーションマスターとして機能し、もう 1 つはスレーブとして機能します) のそれぞれに対して NDB テーブルあたり 1 つのサブスクリプションがあります。各サブスクライバは 16 バイトのメモリーを使用します。

  循環レプリケーション、マルチマスターレプリケーション、および 3 つ以上の MySQL サーバーが関与するその他のレプリケーションセットアップを使用するときは、このパラメータをレプリケーションに含まれる mysqld プロセスの数 (これは通常 (常にではありませんが) クラスタの数と同じです) まで増やすようにしてください。たとえば、3 つの MySQL Cluster を使用する循環レプリケーションセットアップがあり、1 つの mysqld が各クラスタに接続され、これらの mysqld プロセスがそれぞれマスターおよびスレーブとして機能する場合は、MaxNoOfSubscribers を 3 * MaxNoOfTables に設定するようにしてください。

  詳細は、セクション18.6「MySQL Cluster レプリケーション」を参照してください。

- MaxNoOfConcurrentSubOperations

有効なバージョン	型/単位	デフォルト	範囲/値	再起動タイプ
NDB 7.3.0	符号なし	256	0 - 4294967039 (0xFFFFFEFF)	N

  このパラメータは、クラスタ内のすべての API ノードが一度に実行できる操作数の上限を設定します。通常の操作ではデフォルト値 (256) で十分ですが、API ノードが多数あり、それぞれが大量の操作を同時に実行しているシナリオでのみ、調整が必要になる可能性があります。

ブールパラメータ　データノードの動作は、ブール値を取る一連の [ndbd] パラメータにも影響されます。これらの各パラメータは、TRUE として指定する場合は 1 または Y に設定し、FALSE として指定する場合は 0 または N に設定します。

- LateAlloc

有効なバージョン	型/単位	デフォルト	範囲/値	再起動タイプ
NDB 7.3.0	数値	1	0 - 1	N

管理サーバーへの接続が確立されたあとで、このデータノードのメモリーを割り当てます。デフォルトで有効。

- LockPagesInMainMemory

有効なバージョン	型/単位	デフォルト	範囲/値	再起動タイプ
NDB 7.3.0	数値	0	0 - 2	N

Solaris と Linux を含むいくつかのオペレーティングシステムでは、プロセスをメモリーにロックすると、ディスクへのスワップを回避できます。これを使用すると、クラスタのリアルタイム特性が保証されやすくなります。

このパラメータは、整数値 0、1、または 2 のいずれかを取ります。これらの機能を次のリストに示します。

- 0: ロックを無効にします。これはデフォルト値です。
- 1: プロセスのメモリーを割り当てたあとでロックを実行します。
- 2: プロセスのメモリーを割り当てる前にロックを実行します。

権限のないユーザーがページをロックできるようにオペレーティングシステムが構成されていない場合は、このパラメータを利用するデータノードプロセスをシステムの root として実行する必要がある可能性があります。(LockPagesInMainMemory では、mlockall 関数が使用されます。Linux kernel 2.6.9 以降では、権限のないユーザーが max locked memory の制限に従ってメモリーをロックできます。詳細は、ulimit -l および http://linux.die.net/man/2/mlock を参照してください)。

> **注記**
>
> 古い MySQL Cluster リリースでは、このパラメータはブール型でした。デフォルト設定だった 0 または false では、ロックが無効化されました。1 または true では、プロセスのメモリーが割り当てられたあとでそのロックが有効化されました。MySQL Cluster NDB 7.3 以降では、このパラメータの値に true または false を使用すると、エラーとして処理されます。

> **重要**
>
> glibc 2.10 以降では、glibc が共有プールでのロック競合を減らすためにスレッド単位のアリーナを使用し、これによって実メモリーが消費されます。通常、データノードプロセスは起動後にメモリー割り当てを実行しないため、スレッド単位のアリーナを必要としません。(このアロケータの違いがパフォーマンスに重大な影響を与えることはありません。)
>
> この glibc の動作は MALLOC_ARENA_MAX 環境変数を使用して構成できるようになっていますが、glibc 2.16 より前では、このメカニズムのバグが原因でこの変数を 8 未満に設定できなかったため、無駄になったメモリーを再利用できませんでした。(Bug #15907219。この問題の詳細は、http://sourceware.org/bugzilla/show_bug.cgi?id=13137 も参照してください。)
>
> この問題の考えられる回避策は、LD_PRELOAD 環境変数を使用して jemalloc メモリー割り当てライブラリをプリロードし、glibc に付属するライブラリの代わりに使用することです。

- StopOnError

有効なバージョン	型/単位	デフォルト	範囲/値	再起動タイプ
NDB 7.3.0	ブール	1	0、1	N

このパラメータは、データノードプロセスでエラー状態が発生したときに、プロセスを終了するか、自動的に再起動するか指定します。

このパラメータのデフォルト値は 1 です。これは、デフォルトで、エラー発生時にデータノードプロセスが停止することを意味します。

- CrashOnCorruptedTuple

有効なバージョン	型/単位	デフォルト	範囲/値	再起動タイプ
NDB 7.3.0	ブール	true	true、false	S

このパラメータを有効にすると、破損したタプルが検出されたときにデータノードが強制的にシャットダウンされます。MySQL Cluster NDB 7.3 以降では、これはデフォルトで有効になっています。

- Diskless

有効なバージョン	型/単位	デフォルト	範囲/値	再起動タイプ
NDB 7.3.0	true|false (1|0)	false	true、false	IS

MySQL Cluster のテーブルをディスクレスとして指定できます。これは、テーブルがディスクにチェックポイントされず、ロギングが発生しないことを意味します。このようなテーブルはメインメモリーにのみ存在します。ディスクレステーブルを使用すると、クラッシュ発生時にテーブルもテーブル内のレコードも失われます。ただし、ディスクレスモードで動作するときは、ディスクレスコンピュータで ndbd を実行できます。

> **重要**
>
> この機能を使用すると、クラスタ全体がディスクレスモードで動作します。

この機能を有効にすると、クラスタのオンラインバックアップが無効になります。また、クラスタの部分的起動ができなくなります。

Diskless はデフォルトで無効になっています。

- ODirect

有効なバージョン	型/単位	デフォルト	範囲/値	再起動タイプ
NDB 7.3.0	ブール	false	true、false	N

このパラメータを有効にすると、NDB が LCP、バックアップ、および Redo ログで O_DIRECT 書き込みを試行し、多くの場合 kswapd と CPU の使用率が低下します。Linux 上で MySQL Cluster を使用するときに 2.6 以降のカーネルを使用する場合は、ODirect を有効にしてください。

ODirect はデフォルトで無効になっています。

- RestartOnErrorInsert

有効なバージョン	型/単位	デフォルト	範囲/値	再起動タイプ
NDB 7.3.0	エラーコード	2	0 - 4	N

この機能は、コードの個々のブロックをテストの一部として実行する際に、エラーの挿入が可能なデバッグバージョンをビルドする場合にのみ利用できます。

この機能はデフォルトで無効になっています。

- CompressedBackup

有効なバージョン	型/単位	デフォルト	範囲/値	再起動タイプ
NDB 7.3.0	ブール	false	true、false	N

このパラメータを 1 に設定すると、バックアップファイルが圧縮されます。使用される圧縮は gzip --fast と同等であり、データノードで非圧縮バックアップファイルの格納に必要なスペースの 50% 以上を節約できます。圧縮バックアップは、個々のデータノードまたは (config.ini ファイルの [ndbd default] セクションにこのパラメータを設定することで) すべてのデータノードで有効化できます。

> **重要**
>
> 圧縮バックアップを、この機能をサポートしない MySQL バージョンを実行するクラスタにリストアすることはできません。

デフォルト値は 0 (無効) です。

- CompressedLCP

有効なバージョン	型/単位	デフォルト	範囲/値	再起動タイプ
NDB 7.3.0	ブール	false	true、false	N

このパラメータを 1 に設定すると、ローカルチェックポイントが圧縮されます。使用される圧縮は gzip --fast と同等であり、データノードで非圧縮チェックポイントファイルの格納に必要なスペースの 50% 以上を節約できます。圧縮 LCP は、個々のデータノードまたは (config.ini ファイルの [ndbd default] セクションにこのパラメータを設定することで) すべてのデータノードで有効化できます。

> **重要**
>
> 圧縮ローカルチェックポイントを、この機能をサポートしない MySQL バージョンを実行するクラスタにリストアすることはできません。

デフォルト値は 0 (無効) です。

タイムアウト、間隔、およびディスクページングの制御

クラスタデータノード内のさまざまなアクション間のタイムアウトと間隔を指定する、いくつかの [ndbd] パラメータがあります。ほとんどのタイムアウト値はミリ秒単位で指定します。この例外については、該当する箇所で説明します。

- TimeBetweenWatchDogCheck

有効なバージョン	型/単位	デフォルト	範囲/値	再起動タイプ
NDB 7.3.0	ミリ秒	6000	70 - 4294967039 (0xFFFFFEFF)	N

メインスレッドがある時点で無限ループに陥ることを防ぐため、「ウォッチドッグ」スレッドがメインスレッドをチェックします。このパラメータは、チェック間のミリ秒数を指定します。プロセスが 3 回のチェック後も同じ状態の場合、ウォッチドッグスレッドはプロセスを強制終了します。

このパラメータは、実験のため、または口ーカルの状態に合わせて簡単に変更できます。これをノード単位で指定することもできますが、指定することはほとんどありません。

デフォルトのタイムアウトは 6000 ミリ秒 (6 秒) です。

- TimeBetweenWatchDogCheckInitial

有効なバージョン	型/単位	デフォルト	範囲/値	再起動タイプ
NDB 7.3.0	ミリ秒	6000	70 - 4294967039 (0xFFFFFEFF)	N

これは TimeBetweenWatchDogCheck パラメータとほぼ同じですが、TimeBetweenWatchDogCheckInitial は、メモリー割り当てが行われる初期起動段階でデータベースノード内部の実行チェック間に経過する時間を制御します。

デフォルトのタイムアウトは 6000 ミリ秒 (6 秒) です。

- StartPartialTimeout

有効なバージョン	型/単位	デフォルト	範囲/値	再起動タイプ
NDB 7.3.0	ミリ秒	30000	0 - 4294967039 (0xFFFFFEFF)	N

このパラメータは、クラスタの初期化ルーチンが呼び出されるまでにクラスタがデータノードの起動を待つ時間を指定します。このタイムアウトは、可能なかぎり、クラスタの部分的起動を回避するために使用されます。

このパラメータは、クラスタの初期起動または初期再起動の実行時にオーバーライドされます。

デフォルト値は 30000 ミリ秒 (30 秒) です。0 にするとタイムアウトが無効になり、すべてのノードが使用可能な場合にのみクラスタを起動できるようになります。

- StartPartitionedTimeout

有効なバージョン	型/単位	デフォルト	範囲/値	再起動タイプ
NDB 7.3.0	ミリ秒	60000	0 - 4294967039 (0xFFFFFEFF)	N

StartPartialTimeout ミリ秒の待機後にクラスタが起動できる状態になっても、まだパーティション化された状態の可能性がある場合、クラスタはこのタイムアウトが経過するまで待機します。StartPartitionedTimeout を 0 に設定すると、クラスタは無期限で待機します。

このパラメータは、クラスタの初期起動または初期再起動の実行時にオーバーライドされます。

デフォルトのタイムアウトは 60000 ミリ秒 (60 秒) です。

- StartFailureTimeout

有効なバージョン	型/単位	デフォルト	範囲/値	再起動タイプ
NDB 7.3.0	ミリ秒	0	0 - 4294967039 (0xFFFFFEFF)	N

データノードでこのパラメータで指定された時間内に起動シーケンスが完了されなかった場合、ノードの起動は失敗します。このパラメータを 0 (デフォルト値) に設定すると、データノードのタイムアウトは適用されません。

このパラメータでは、0 以外の値はミリ秒単位で測定されます。非常に多くのデータを含むデータノードでは、このパラメータを増やすようにしてください。たとえば、数ギガバイトのデータを含むデータノードの場合、ノードを再起動するのに 10-15 分 (つまり、600000-1000000 ミリ秒) の時間が必要です。

- StartNoNodeGroupTimeout

有効なバージョン	型/単位	デフォルト	範囲/値	再起動タイプ
NDB 7.3.0	ミリ秒	15000	0 - 4294967039 (0xFFFFFEFF)	N

Nodegroup = 65536 でデータノードを構成すると、そのノードはどのノードグループにも割り当てられないものとみなされます。その場合、クラスタは StartNoNodegroupTimeout ミリ秒待機してから、そのようなノードを --nowait-nodes オプションに渡されたリストに追加されたものとして扱い、起動します。デフォルト値は 15000 です (つまり、管理サーバーは 15 秒待機します)。このパラメータを 0 に設定すると、クラスタは無期限に待機します。

StartNoNodegroupTimeout はクラスタ内のすべてのデータノードで同じにする必要があります。そのため、これは個々のデータノードではなく、常に config.ini ファイルの [ndbd default] セクションに設定するようにしてください。

詳細は、セクション18.5.13「MySQL Cluster データノードのオンライン追加」を参照してください。

- HeartbeatIntervalDbDb

有効なバージョン	型/単位	デフォルト	範囲/値	再起動タイプ
NDB 7.3.0	ミリ秒	5000	10 - 4294967039 (0xFFFFFEFF)	N

障害の発生したノードを検出する主な方法の 1 つは、ハートビートを使用することです。このパラメータは、ハートビート信号の送信回数と予想される受信回数を指定します。3 回連続でハートビート間隔が失敗すると、そのノードはデッドと宣言されます。したがって、ハートビートメカニズムによる障害検出の最大時間は、ハートビート間隔の 4 倍になります。

MySQL Cluster NDB 7.3 以降では、デフォルトのハートビート間隔は 5000 ミリ秒 (5 秒) です。このパラメータを大幅に変更しないでください。また、ノード間の違いが大きくならないようにしてください。あるノードが 5000 ミリ秒を使用し、それを監視するノードが 1000 ミリ秒を使用する場合、そのノードは非常に速くデッドと宣言されます。このパラメータはオンラインのソフトウェアアップグレード中に変更できますが、少しずつ増やしてください。

ネットワーク通信と待機時間も参照してください。

- HeartbeatIntervalDbApi

有効なバージョン	型/単位	デフォルト	範囲/値	再起動タイプ
NDB 7.3.0	ミリ秒	1500	100 - 4294967039 (0xFFFFFEFF)	N

各データノードは各 MySQL サーバー (SQL ノード) にハートビート信号を送信して、接続されたままであるか確認します。MySQL サーバーが時間内にハートビートを送信できなかった場合、「デッド」と宣言されます。その場合、進行中のすべてのトランザクションが完了し、リソースが解放されます。以前の MySQL インスタンスによって開始されたすべてのアクティビティーが完了するまで、SQL ノードは再接続できません。この判定に使用される 3 ハートビートの条件は、HeartbeatIntervalDbDb で説明したものと同じです。

デフォルトの間隔は 1500 ミリ秒 (1.5 秒) です。個々のデータノードはほかのすべてのデータノードと関係なく接続中の MySQL サーバーを監視するため、この間隔は各データノードで異なることがあります。

詳細は、ネットワーク通信と待機時間を参照してください。

- HeartbeatOrder

有効なバージョン	型/単位	デフォルト	範囲/値	再起動タイプ
NDB 7.3.0	数値	0	0 - 65535	S

データノードは、各データノードが直前のデータノードをモニターする循環的な方法で、相互にハートビートを送信します。ハートビートが特定のデータノードによって検出されなかった場合、このノードは循環の直前のデータノードを「デッド」(つまり、クラスタからアクセスできなくなった) と宣言します。データノードがデッドであるという判定はグローバルに行われます。つまり、データノードがデッドと宣言されると、そのノードはクラスタ内のすべてのノードからそのようにみなされます。

異なるホストに配置されたデータノード間のハートビートが (たとえば、非常に長いハートビート間隔や一時的な接続の問題によって) ほかのノードペア間のハートビートに比べて遅すぎるために、データノードがまだクラスタの一部として機能しているにもかかわらず、デッドと宣言される可能性があります。

このような状況では、データノード間のハートビートの転送順序が、特定のデータノードがデッドと宣言されるかどうかに影響している可能性があります。この宣言が不必要に発生すると、それがノードグループの損失を招き、さらにクラスタの障害につながる可能性があります。

次の表に示すように、2 台のホストコンピュータ host1 および host2 で実行されている 4 つのデータノード A、B、C、および D があり、これらのデータノードが 2 つのノードグループを構成しているセットアップについて考えます。

ノードグループ	host1 で実行されているノード	host2 で実行されているノード
ノードグループ 0:	ノード A	ノード B
ノードグループ 1:	ノード C	ノード D

ハートビートが A->B->C->D->A の順に転送されるとします。この場合、ホスト間のハートビートが失われると、ノード B がノード A をデッドと宣言し、ノード C が B をデッドと宣言します。この結果、ノードグルー

プ 0 が失われるため、クラスタが機能停止します。一方、転送の順序が A->B->D->C->A である (およびほかのすべての条件が前述のままである) 場合は、ハートビートが失われると、ノード A および D がデッドと宣言されます。この場合、各ノードグループに 1 つずつノードが残っているため、クラスタは存続します。

HeartbeatOrder 構成パラメータは、ユーザーがハートビートの転送順序を構成できるようにします。HeartbeatOrder のデフォルト値は 0 です。すべてのデータノードでデフォルト値を使用できるようにすると、ハートビートの転送順序は NDB によって決定されます。このパラメータを使用する場合は、クラスタ内のすべてのデータノードで 0 以外の値 (最大 65535) に設定する必要があります。また、この値は各データノードで一意である必要があります。これにより、ハートビートが HeartbeatOrder 値の小さいノードから大きいノードに順に転送される (その後、HeartbeatOrder がもっとも大きいノードからもっとも小さいノードに転送され、循環が完結する) ようになります。たとえば、前述のシナリオでハートビートの転送順序 A->B->D->C->A を強制するために、値を連続させる必要はありません。ここに示すような HeartbeatOrder 値を設定できます。

ノード	HeartbeatOrder
A	10
B	20
C	30
D	25

このパラメータを使用して実行中の MySQL Cluster 内のハートビートの転送順序を変更するには、最初にグローバル構成 (config.ini) ファイルにクラスタ内の各データノードの HeartbeatOrder を設定する必要があります。変更を有効にするには、次のいずれかを実行する必要があります。

- クラスタ全体の完全なシャットダウンと再起動。
- 2 回連続のクラスタのローリング再起動。両方のローリング再起動で、すべてのノードが同じ順序で再起動される必要があります。

DUMP 908 を使用すると、データノードログでこのパラメータの効果を観察できます。

- ConnectCheckIntervalDelay

有効なバージョン	型/単位	デフォルト	範囲/値	再起動タイプ
NDB 7.3.0	文字列	0	0 - 4294967039 (0xFFFFFEFF)	N

このパラメータは、データノード間の接続チェックを有効にします。ConnectCheckIntervalDelay 秒の間隔以内に応答できないデータノードは疑いありとみなされ、そのような間隔が 2 回続いたあとでデッドとみなされます。

このパラメータのデフォルト値は 0 です。これは、MySQL Cluster NDB 7.1 からの変更点です。

- TimeBetweenLocalCheckpoints

有効なバージョン	型/単位	デフォルト	範囲/値	再起動タイプ
NDB 7.3.0	4 バイトワードの数 (底 2 の対数として)	20	0 - 31	N

このパラメータは、新しいローカルポイントを開始するまでの待機時間を指定しないという点で例外です。どちらかといえば、比較的少数の更新が行われるクラスタ内でローカルチェックポイントが実行されないようにするために使用されます。更新回数が多いほとんどのクラスタでは、直前のローカルチェックポイントが完了した直後に新しいローカルチェックポイントが開始される可能性があります。

前のローカルチェックポイントの開始以降に実行されたすべての書き込み操作のサイズが追加されます。このパラメータは、4 バイトワードの数の底 2 の対数として指定される点でも例外的です。したがって、デフォルト値の 20 は 4M バイト ($4 * 2^{20}$) の書き込み操作を意味し、21 は 8M バイトを意味し、最大値の 31 は 8G バイトの書き込み操作と同等です。

クラスタ内のすべての書き込み操作がまとめて追加されます。TimeBetweenLocalCheckpoints を 6 以下に設定すると、ローカルチェックポイントはクラスタのワークロードと無関係に途切れることなく連続的に実行されます。

- TimeBetweenGlobalCheckpoints

有効なバージョン	型/単位	デフォルト	範囲/値	再起動タイプ
NDB 7.3.0	ミリ秒	2000	20 - 32000	N

トランザクションがコミットされると、データがミラー化されているノードのメインメモリーでコミットされます。ただし、トランザクションログレコードはコミットの一部としてディスクにフラッシュされません。トランザクションを少なくとも 2 台の自立的なホストマシン上で安全にコミットすることで、持続性に関する妥当な基準が満たされるはずだということがこの動作の根拠です。

また、最悪のケース (クラスタの完全なクラッシュ) も適切に処理されるようにすることが重要です。この実行を保証するために、特定の間隔以内に行われるすべてのトランザクションはグローバルチェックポイントに取り込まれます。これは、ディスクにフラッシュされたコミット済みのトランザクションとみなすことができます。つまり、トランザクションはコミットプロセスの一部としてグローバルチェックポイントグループに組み込まれます。その後、このグループのログレコードがディスクにフラッシュされると、トランザクションのグループ全体がクラスタ内のコンピュータ上のディスクに安全にコミットされます。

このパラメータは、グローバルチェックポイント間の間隔を定義します。デフォルトは 2000 ミリ秒です。

- TimeBetweenEpochs

有効なバージョン	型/単位	デフォルト	範囲/値	再起動タイプ
NDB 7.3.0	ミリ秒	100	0 - 32000	N

このパラメータは、MySQL Cluster レプリケーションの同期エポック間の間隔を定義します。デフォルト値は 100 ミリ秒です。

TimeBetweenEpochs は、MySQL Cluster レプリケーションのパフォーマンスを向上させるために使用できる「micro-GCP」 の実装の一部です。

- TimeBetweenEpochsTimeout

有効なバージョン	型/単位	デフォルト	範囲/値	再起動タイプ
NDB 7.3.0	ミリ秒	0	0 - 256000	N

このパラメータは、MySQL Cluster レプリケーションの同期エポックのタイムアウトを定義します。このパラメータで決定された時間内にノードがグローバルチェックポイントに参加できなかった場合、ノードはシャットダウンされます。MySQL Cluster NDB 7.3 以降では、デフォルト値は 0 です (つまり、タイムアウトは無効になっています)。

TimeBetweenEpochsTimeout は、MySQL Cluster レプリケーションのパフォーマンスを向上させるために使用できる 「micro-GCP」 の実装の一部です。

GCP の保存に 1 分を超える時間がかかった場合、または GCP の保存に 10 秒を超える時間がかかった場合は、常にこのパラメータの現在の値と警告がクラスタログに書き込まれます。

このパラメータを 0 に設定すると、保存のタイムアウト、コミットのタイムアウト、またはその両方によって発生する GCP の停止を無効にします。このパラメータに指定できる最大値は 256000 ミリ秒です。

- MaxBufferedEpochs

有効なバージョン	型/単位	デフォルト	範囲/値	再起動タイプ
NDB 7.3.0	エポック	100	0 - 100000	N

サブスクライブするノードが遅延できる未処理のエポック数。この数を超えると、遅延するサブスクライバが切断されます。

ほとんどの通常の操作では、デフォルト値の 100 で十分です。サブスクライブするノードの遅延によって切断が発生する場合、その原因は通常、プロセスまたはスレッドに関するネットワークまたはスケジューリングの問題です。(まれな状況ですが、NDB クライアントのバグによってこの問題が起きることもあります。)エポックが大きいときは、この値をデフォルトより小さく設定するのが望ましい場合もあります。

切断することで、クライアントの問題によってデータノードサービスが影響を受け、データをバッファリングするためのメモリーが不足し、最終的にシャットダウンすることはなくなります。代わりに、切断の結果とし

て (たとえば、バイナリログのギャップイベントによって) クライアントが影響を受け、クライアントは強制的にプロセスを再接続または再起動します。

- MaxBufferedEpochBytes

有効なバージョン	型/単位	デフォルト	範囲/値	再起動タイプ
NDB 7.3.0	バイト	26214400	26214400 (0x01900000) - 4294967039 (0xFFFFFEFF)	N

このノードがエポックをバッファリングするために割り当てるバイトの合計数。

- TimeBetweenInactiveTransactionAbortCheck

有効なバージョン	型/単位	デフォルト	範囲/値	再起動タイプ
NDB 7.3.0	ミリ秒	1000	1000 - 4294967039 (0xFFFFFEFF)	N

タイムアウトの処理は、各トランザクションのタイマーをこのパラメータで指定された間隔ごとに 1 回チェックして実行されます。したがって、このパラメータを 1000 ミリ秒に設定すると、すべてのトランザクションのタイムアウトが毎秒 1 回チェックされます。

デフォルト値は 1000 ミリ秒 (1 秒) です。

- TransactionInactiveTimeout

有効なバージョン	型/単位	デフォルト	範囲/値	再起動タイプ
NDB 7.3.0	ミリ秒	[テキストを参照]	0 - 4294967039 (0xFFFFFEFF)	N

このパラメータは、トランザクションが中止されるまでに、同じトランザクション内の操作間で経過が許容される最大時間を指定します。

このパラメータのデフォルトは 4G です (最大も同じです)。トランザクションがロックを保持する期間が長すぎないようにする必要があるリアルタイムデータベースでは、このパラメータを比較的小さい値に設定するようにしてください。単位はミリ秒です。

- TransactionDeadlockDetectionTimeout

有効なバージョン	型/単位	デフォルト	範囲/値	再起動タイプ
NDB 7.3.0	ミリ秒	1200	50 - 4294967039 (0xFFFFFEFF)	N

ノードは、トランザクションを含むクエリーの実行時に、クラスタ内のほかのノードが応答するのを待機してから処理を続行します。応答の失敗は、次のいずれかの理由で発生します。

- ノードが「デッド」です
- 操作がロックキューに入りました
- アクションの実行を要求したノードに非常に高い負荷がかかっている可能性があります。

このタイムアウトパラメータは、トランザクションが中止されるまでにトランザクションコーディネータが別のノードによるクエリーの実行を待機する時間を指定するもので、ノード障害の処理とデッドロックの検出において重要です。

デフォルトのタイムアウト値は 1200 ミリ秒 (1.2 秒) です。

このパラメータの最小は 50 ミリ秒です。

- DiskSyncSize

有効なバージョン	型/単位	デフォルト	範囲/値	再起動タイプ
NDB 7.3.0	バイト	4M	32K - 4294967039 (0xFFFFFEFF)	N

これは、ローカルチェックポイントファイルにデータをフラッシュするまでに格納される最大バイト数です。これは、パフォーマンスを大幅に低下させる可能性がある書き込みバッファリングを禁止するために行われます。このパラメータは、TimeBetweenLocalCheckpoints の代わりに使用するためのものではありません。

> **注記**
>
> ODirect が有効になっている場合は、DiskSyncSize を設定する必要はありません。実際、そのような場合は、その値が無視されます。

デフォルト値は 4M (4 メガバイト) です。

- DiskCheckpointSpeed

有効なバージョン	型/単位	デフォルト	範囲/値	再起動タイプ
NDB 7.3.0	バイト	10M	1M - 4294967039 (0xFFFFFEFF)	N

ローカルチェックポイント中にディスクに転送されるデータの量 (バイト/秒)。この割り当ては DML 操作とバックアップ (バックアップログギングは除く) で共有されます。これは、集約的な DML の実行中に開始されたバックアップが Redo ログバッファーの大量発生によって妨害され、競合が深刻な場合は完全に失敗する可能性があることを意味します。

デフォルト値は 10M (10 メガバイト/秒) です。

MySQL Cluster 7.4.1 以降では、このパラメータは非推奨であり、代わりに構成パラメータ MinDiskWriteSpeed, MaxDiskWriteSpeed、MaxDiskWriteSpeedOtherNodeRestart、および MaxDiskWriteSpeedOwnRestart を使用して LCP とバックアップの書き込み速度を制御できます。

- DiskCheckpointSpeedInRestart

有効なバージョン	型/単位	デフォルト	範囲/値	再起動タイプ
NDB 7.3.0	バイト	100M	1M - 4294967039 (0xFFFFFEFF)	N

再起動操作の一部であるローカルチェックポイント中にディスクに転送されるデータの量 (バイト/秒)。

デフォルト値は 100M (100 メガバイト/秒) です。

MySQL Cluster 7.4.1 以降では、このパラメータは非推奨であり、代わりに構成パラメータ MinDiskWriteSpeed, MaxDiskWriteSpeed、MaxDiskWriteSpeedOtherNodeRestart、および MaxDiskWriteSpeedOwnRestart を使用して LCP とバックアップの書き込み速度を制御できます。

- NoOfDiskPagesToDiskAfterRestartTUP

このパラメータは非推奨であり、MySQL Cluster の今後のバージョンで削除される予定です。代わりに DiskCheckpointSpeedInRestart および DiskSyncSize を使用してください。MySQL Cluster 7.4.1 以降では、代わりにそのリリースで導入された構成パラメータ MinDiskWriteSpeed, MaxDiskWriteSpeed、MaxDiskWriteSpeedOtherNodeRestart、および MaxDiskWriteSpeedOwnRestart を使用するようにしてください。

- MaxDiskWriteSpeed

有効なバージョン	型/単位	デフォルト	範囲/値	再起動タイプ
NDB 7.4.1	数値	20M	1M - 1024G	S

この MySQL Cluster で (このデータノードまたはその他のデータノードによって) 再起動が行われていない場合に、ローカルチェックポイントおよびバックアップ操作によるディスク書き込みの最大速度 (バイト/秒) を設定します。

このデータノードの再起動中に許可されるディスク書き込みの最大速度を設定するには、MaxDiskWriteSpeedOwnRestart を使用します。ほかのデータノードの再起動中に許可されるディスク書き込みの最大速度を設定するには、MaxDiskWriteSpeedOtherNodeRestart を使用します。すべての LCP およびバックアップ操作によるディスク書き込みの最小速度を調整するには、MinDiskWriteSpeed を設定します。

MaxDiskWriteSpeed は MySQL Cluster NDB 7.4.1 で追加されました。

- MaxDiskWriteSpeedOtherNodeRestart

有効なバージョン	型/単位	デフォルト	範囲/値	再起動タイプ
NDB 7.4.1	数値	50M	1M - 1024G	S

この MySQL Cluster に含まれるこのノード以外の 1 つ以上のデータノードが再起動している場合に、ローカルチェックポイントおよびバックアップ操作によるディスク書き込みの最大速度 (バイト/秒) を設定します。

このデータノードの再起動中に許可されるディスク書き込みの最大速度を設定するには、MaxDiskWriteSpeedOwnRestart を使用します。クラスタ内ではデータノードが再起動されていない場合に許可されるディスク書き込みの最大速度を設定するには、MaxDiskWriteSpeed を使用します。すべての LCP およびバックアップ操作によるディスク書き込みの最小速度を調整するには、MinDiskWriteSpeed を設定します。

MaxDiskWriteSpeedOtherNodeRestart は MySQL Cluster NDB 7.4.1 で追加されました。

- MaxDiskWriteSpeedOwnRestart

有効なバージョン	型/単位	デフォルト	範囲/値	再起動タイプ
NDB 7.4.1	数値	200M	1M - 1024G	S

このデータノードの再起動時の、ローカルチェックポイントおよびバックアップ操作によるディスク書き込みの最大速度 (バイト/秒) を設定します。

ほかのデータノードの再起動中に許可されるディスク書き込みの最大速度を設定するには、MaxDiskWriteSpeedOtherNodeRestart を使用します。クラスタ内ではデータノードが再起動されていない場合に許可されるディスク書き込みの最大速度を設定するには、MaxDiskWriteSpeed を使用します。すべての LCP およびバックアップ操作によるディスク書き込みの最小速度を調整するには、MinDiskWriteSpeed を設定します。

MaxDiskWriteSpeedOwnRestart は MySQL Cluster NDB 7.4.1 で追加されました。

- MinDiskWriteSpeed

有効なバージョン	型/単位	デフォルト	範囲/値	再起動タイプ
NDB 7.4.1	数値	10M	1M - 1024G	S

ローカルチェックポイントおよびバックアップ操作によるディスク書き込みの最小速度 (バイト/秒) を設定します。

さまざまな条件下で LCP およびバックアップに許可されるディスク書き込みの最大速度は、パラメータ MaxDiskWriteSpeed、MaxDiskWriteSpeedOwnRestart、および MaxDiskWriteSpeedOtherNodeRestart を使用して調整できます。詳細は、これらのパラメータの説明を参照してください。

MinDiskWriteSpeed は MySQL Cluster NDB 7.4.1 で追加されました。

- NoOfDiskPagesToDiskAfterRestartACC

  このパラメータは非推奨であり、MySQL Cluster の今後のバージョンで削除される予定です。MySQL Cluster NDB 7.3 では、代わりに DiskCheckpointSpeedInRestart および DiskSyncSize を使用してください。MySQL Cluster NDB 7.4.1 以降では、パラメータ MinDiskWriteSpeed、MaxDiskWriteSpeed、MaxDiskWriteSpeedOtherNodeRestart、および MaxDiskWriteSpeedOwnRestart を使用するようにしてください。

- NoOfDiskPagesToDiskDuringRestartTUP (非推奨)

  このパラメータは非推奨であり、MySQL Cluster の今後のバージョンで削除される予定です。MySQL Cluster NDB 7.3 では、代わりに DiskCheckpointSpeedInRestart および DiskSyncSize を使用してください。MySQL Cluster NDB 7.4.1 以降では、パラメータ MinDiskWriteSpeed、MaxDiskWriteSpeed、MaxDiskWriteSpeedOtherNodeRestart、および MaxDiskWriteSpeedOwnRestart を使用するようにしてください。

- NoOfDiskPagesToDiskDuringRestartACC (非推奨)

  このパラメータは非推奨であり、MySQL Cluster の今後のバージョンで削除される予定です。MySQL Cluster NDB 7.3 では、代わりに DiskCheckpointSpeedInRestart および DiskSyncSize を使用してください。MySQL Cluster NDB 7.4.1 以降では、パラメータ MinDiskWriteSpeed、MaxDiskWriteSpeed、MaxDiskWriteSpeedOtherNodeRestart、および MaxDiskWriteSpeedOwnRestart を使用するようにしてください。

- ArbitrationTimeout

有効なバージョン	型/単位	デフォルト	範囲/値	再起動タイプ
NDB 7.3.0	ミリ秒	7500	10 - 4294967039 (0xFFFFFEFF)	N

  このパラメータは、データノードがアービトレーションメッセージに対するアービトレータからの応答を待機する時間を設定します。これを超えた場合は、ネットワークが切断されたとみなされます。

  MySQL Cluster NDB 7.3 以降では、デフォルト値は 7500 ミリ秒 (7.5 秒) です。

- Arbitration

有効なバージョン	型/単位	デフォルト	範囲/値	再起動タイプ
NDB 7.3.0	列挙	デフォルト	デフォルト、無効、WaitExternal	N

  Arbitration パラメータを使用して、このパラメータに指定できる 3 つの値のいずれかに対応するアービトレーションスキームを選択できます。

  - デフォルト　この場合は、管理および API ノードの ArbitrationRank 設定で指定されるとおりに、アービトレーションが正常に進行します。これはデフォルト値です。
  - Disabled　config.ini ファイルの [ndbd default] セクションに Arbitration = Disabled を設定すると、すべての管理および API ノードで ArbitrationRank の 0 を設定した場合と同じ結果が得られます。Arbitration をこのように設定すると、ArbitrationRank の設定はすべて無視されます。
  - WaitExternal　Arbitration パラメータを使用して、クラスタが内部でアービトレーションを処理する代わりに ArbitrationTimeout で指定された時間が経過するまで外部のクラスタマネージャーアプリケーションによるアービトレーションの実行を待機する方法で、アービトレーションを構成することもできます。これを行うには、config.ini ファイルの [ndbd default] セクションに Arbitration = WaitExternal を設定します。WaitExternal の設定で最良の結果を得るためには、ArbitrationTimeout を外部クラスタマネージャーがアービトレーションを実行するのに必要な間隔の 2 倍に設定することをお勧めします。

    > **重要**
    >
    > このパラメータは、クラスタ構成ファイルの [ndbd default] セクションでのみ使用するようにしてください。Arbitration を個々のデータノードで異なる値に設定すると、クラスタの動作は不特定になります。

バッファリングとロギング　上級ユーザーは、いくつかの [ndbd] 構成パラメータを使用するとことで、ノードプロセスが使用するリソースをより細かく制御したり、必要に応じてさまざまなバッファーサイズを調整したりできます。

これらのバッファーは、ログレコードをディスクに書き込む際に、ファイルシステムに対するフロントエンドとして使用されます。ノードがディスクレスモードで実行されている場合は、NDB ストレージエンジンのファイルシステム抽象化レイヤーによってディスク書き込みが「偽装」されるため、ペナルティーなしでこれらのパラメータを最小値に設定できます。

- UndoIndexBuffer

有効なバージョン	型/単位	デフォルト	範囲/値	再起動タイプ
NDB 7.3.0	符号なし	2M	1M - 4294967039 (0xFFFFFEFF)	N

このパラメータでサイズが設定される Undo インデックスバッファーは、ローカルチェックポイント中に使用されます。NDB ストレージエンジンは、使用可能な Redo ログとともに、チェックポイントの一貫性に基づくリカバリスキームを使用します。システム全体の書き込みをブロックせずに一貫性のあるチェックポイントを作成するため、ローカルチェックポイントの実行中に Undo ロギングが行われます。Undo ロギングは、一度に 1 つのテーブルフラグメントに対してアクティブ化されます。この最適化が可能なのは、テーブルが完全にメインメモリーに格納されているためです。

Undo インデックスバッファーは、主キーのハッシュインデックスの更新で使用されます。挿入と削除では、ハッシュインデックスが再編成されます。NDB ストレージエンジンは、すべての物理的な変更をシステムの再起動時に取り消すことができるように、変更を 1 つのインデックスページにマップする Undo ログレコードを書き込みます。また、ローカルチェックポイントの開始時に、各フラグメントのアクティブな挿入操作も記録します。

読み取りと更新では、ロックビットが設定され、ハッシュインデックスエントリのヘッダーが更新されます。これらの変更はページ書き込みアルゴリズムによって処理されるため、これらの操作に Undo ロギングは必要ありません。

このバッファーはデフォルトで 2M バイトです。最小値は 1M バイトです。ほとんどのアプリケーションではこれで十分です。大規模なトランザクションや大規模な主キーとともに非常に大規模または多数の挿入および削除を実行するアプリケーションでは、このバッファーのサイズを増やす必要がある場合があります。このバッファーが小さすぎる場合、NDB ストレージエンジンは内部エラーコード 677「Index UNDO buffers overloaded」を発行します。

> **重要**
>
> ローリング再起動中にこのパラメータの値を減らすのは安全ではありません。

- UndoDataBuffer

有効なバージョン	型/単位	デフォルト	範囲/値	再起動タイプ
NDB 7.3.0	符号なし	16M	1M - 4294967039 (0xFFFFFEFF)	N

このパラメータは、Undo データバッファーのサイズを設定します。Undo データバッファーは、Undo インデックスバッファーとほぼ同じ機能を実行しますが、インデックスメモリーではなくデータメモリーに関して使用されます。このバッファーは、フラグメントのローカルチェックポイントフェーズで挿入、削除、および更新のために使用されます。

Undo ログエントリは記録される操作が増えるにつれて大きくなる傾向があるため、このバッファーも対応するインデックスメモリーより大きくなります (デフォルト値は 16M バイトです)。

一部のアプリケーションでは、このメモリー量が不必要に大きい場合があります。そのような場合は、このサイズを最小の 1M バイトに減らすことができます。

ほとんどの場合、このバッファーのサイズを増やす必要はありません。そのような必要がある場合は、データベースの更新アクティビティーによって発生する負荷をディスクが実際に処理できているかどうかをチェックすることをお勧めします。このバッファーのサイズを増やしても、ディスクスペースの不足を解消することはできません。

このバッファーが小さすぎて過密状態になった場合、NDB ストレージエンジンは次の内部エラーコードを発行します: 891 (Data UNDO buffers overloaded)。

> **重要**
>
> ローリング再起動中にこのパラメータの値を減らすのは安全ではありません。

- RedoBuffer

有効なバージョン	型/単位	デフォルト	範囲/値	再起動タイプ
NDB 7.3.0	バイト	32M	1M - 4294967039 (0xFFFFFEFF)	N

更新アクティビティーも、すべて記録する必要があります。Redo ログにより、システムが再起動されるたびにこれらの更新を再現できるようになります。NDB のリカバリアルゴリズムは、Undo ログとともにデータの「ファジー」チェックポイントを使用し、Redo ログを適用してリストアポイントまでのすべての変更を再現します。

RedoBuffer は、Redo ログが書き込まれるバッファーのサイズを設定します。デフォルト値は 32M バイトです。最小値は 1M バイトです。

このバッファーが小さすぎる場合、NDBストレージエンジンは内部エラーコード 1221 (REDO log buffers overloaded)を発行します。このため、クラスタ構成のオンライン変更の一部として RedoBuffer の値を減らそうとする場合は注意してください。

ndbmtd は、LDM スレッドごとに別個のバッファーを割り当てます (ThreadConfig を参照してください)。たとえば、4 つの LDM スレッドがある場合、ndbmtd データノードには実際には 4 つのバッファーがあり、それぞれに RedoBuffer バイト (合計で 4 * RedoBuffer バイト) が割り当てられます。

- EventLogBufferSize

有効なバージョン	型/単位	デフォルト	範囲/値	再起動タイプ
NDB 7.3.0	バイト	8192	0 - 64K	S

データノード内部の NDB ログイベントに使用される循環バッファーのサイズを制御します。

ログメッセージの制御　クラスタの管理では、stdout に送信されるさまざまなイベントタイプのログメッセージ数を制御できることが非常に重要です。イベントカテゴリごとに、16 個の設定可能なイベントレベル (番号 0-15) があります。特定のイベントカテゴリのイベントレポートをレベル 15 に設定すると、そのカテゴリのすべてのイベントレポートが stdout に送信されます。0 に設定すると、そのカテゴリのイベントレポートがまったく行われなくなります。

デフォルトでは、起動メッセージのみが stdout に送信され、残りのイベントレポートのレベルはデフォルトの 0 に設定されます。これは、これらのメッセージが管理サーバーのクラスタログにも送信されるためです。

管理クライアントで同じようなレベルのセットを設定すると、クラスタログに記録するイベントのレベルを指定できます。

- LogLevelStartup

有効なバージョン	型/単位	デフォルト	範囲/値	再起動タイプ
NDB 7.3.0	整数	1	0 - 15	N

プロセスの起動中に生成されるイベントのレポートレベル。

デフォルトのレベルは 1 です。

- LogLevelShutdown

有効なバージョン	型/単位	デフォルト	範囲/値	再起動タイプ
NDB 7.3.0	整数	0	0 - 15	N

ノードの正常なシャットダウンの一部として生成されるイベントのレポートレベル。

デフォルトのレベルは 0 です。

- LogLevelStatistic

有効なバージョン	型/単位	デフォルト	範囲/値	再起動タイプ
NDB 7.3.0	整数	0	0 - 15	N

主キー読み取りの数、更新の数、挿入の数、バッファーの使用状況に関する情報などの統計イベントのレポートレベル。

デフォルトのレベルは 0 です。

- LogLevelCheckpoint

有効なバージョン	型/単位	デフォルト	範囲/値	再起動タイプ
NDB 7.3.0	ログレベル	0	0 - 15	N

ローカルおよびグローバルチェックポイントによって生成されるイベントのレポートレベル。

デフォルトのレベルは 0 です。

- LogLevelNodeRestart

有効なバージョン	型/単位	デフォルト	範囲/値	再起動タイプ
NDB 7.3.0	整数	0	0 - 15	N

ノード再起動中に生成されるイベントのレポートレベル。

デフォルトのレベルは 0 です。

- LogLevelConnection

有効なバージョン	型/単位	デフォルト	範囲/値	再起動タイプ
NDB 7.3.0	整数	0	0 - 15	N

クラスタノード間の接続によって生成されるイベントのレポートレベル。

デフォルトのレベルは 0 です。

- LogLevelError

有効なバージョン	型/単位	デフォルト	範囲/値	再起動タイプ
NDB 7.3.0	整数	0	0 - 15	N

クラスタ全体のエラーおよび警告によって生成されるイベントのレポートレベル。これらのエラーは、ノードの障害の原因にはなりませんが、レポートする価値はあると考えられます。

デフォルトのレベルは 0 です。

- LogLevelCongestion

有効なバージョン	型/単位	デフォルト	範囲/値	再起動タイプ
NDB 7.3.0	levelr	0	0 - 15	N

輻輳によって生成されるイベントのレポートレベル。これらのエラーは、ノードの障害の原因にはなりませんが、レポートする価値はあると考えられます。

デフォルトのレベルは 0 です。

- LogLevelInfo

有効なバージョン	型/単位	デフォルト	範囲/値	再起動タイプ
NDB 7.3.0	整数	0	0 - 15	N

クラスタの一般的な状態に関する情報として生成されるイベントのレポートレベル。

デフォルトのレベルは 0 です。

- MemReportFrequency

有効なバージョン	型/単位	デフォルト	範囲/値	再起動タイプ
NDB 7.3.0	符号なし	0	0 - 4294967039 (0xFFFFFEFF)	N

このパラメータは、データノードのメモリー使用状況レポートをクラスタログに記録する頻度を制御します。これは、レポート間の秒数を表す整数値です。

各データノードのデータメモリーおよびインデックスメモリーの使用状況は、config.ini ファイルに設定された DataMemory および IndexMemory の割合と 32K バイトのページ数の両方でそれぞれ記録されます。たとえば、DataMemory が 100M バイトであり、特定のデータノードがデータメモリーのストレージとして 50M バイトを使用している場合、クラスタログの対応する行はこのようになります。

```
2006-12-24 01:18:16 [MgmSrvr] INFO -- Node 2: Data usage is 50%(1280 32K pages of total 2560)
```

MemReportFrequency は必須のパラメータではありません。使用する場合は、config.ini の [ndbd default] セクションですべてのクラスタデータノード用に設定することも、構成ファイルの対応する [ndbd] セクションで個々のデータノード用に設定またはオーバーライドすることもできます。最小値 (デフォルト値も同じ) は 0 です。この場合は、セクション18.5.6.2「MySQL Cluster ログイベント」の統計イベントの説明で指摘したように、メモリー使用量が特定の割合 (80%、90%、および 100%) に達したときにのみ、メモリーのレポートが記録されます。

- StartupStatusReportFrequency

有効なバージョン	型/単位	デフォルト	範囲/値	再起動タイプ
NDB 7.3.0	秒	0	0 - 4294967039 (0xFFFFFEFF)	N

--initial を指定してデータノードを起動すると、起動フェーズ 4 (セクション18.5.1「MySQL Cluster の起動フェーズのサマリー」を参照してください) で Redo ログファイルが初期化されます。NoOfFragmentLogFiles、FragmentLogFileSize、またはその両方に非常に大きな値を設定すると、この初期化に長い時間がかかることがあります。StartupStatusReportFrequency 構成パラメータを使用して、このプロセスの進行に関するレポートを定期的に記録するよう強制できます。この場合、ここに示すように、初期化されたファイルの数とスペースの量に関する進行状況がクラスタログにレポートされます。

```
2009-06-20 16:39:23 [MgmSrvr] INFO -- Node 1: Local redo log file initialization status:
#Total files: 80, Completed: 60
#Total MBytes: 20480, Completed: 15557
2009-06-20 16:39:23 [MgmSrvr] INFO -- Node 2: Local redo log file initialization status:
#Total files: 80, Completed: 60
#Total MBytes: 20480, Completed: 15570
```

これらのレポートは、起動フェーズ 4 で StartupStatusReportFrequency 秒ごとに記録されます。StartupStatusReportFrequency が 0 (デフォルト) の場合は、Redo ログファイルの初期化プロセスの開始時と完了時にのみ、レポートがクラスタログに書き込まれます。

デバッグパラメータ　MySQL Cluster NDB 7.3 以降では、DictTrace を使用して、テーブルの作成および削除によって生成されたイベントのトレースを記録できます。このパラメータは、NDB のカーネルコードをデバッグする場合にのみ役立ちます。DictTrace は整数値を取ります。現在サポートされる唯一の値は 0 (デフォルト。ロギングなし) と 1 (ロギングが有効) です。

バックアップパラメータ　このセクションで説明する [ndbd] パラメータは、オンラインバックアップの実行用に確保されるメモリーバッファーを定義します。

- BackupDataBufferSize

有効なバージョン	型/単位	デフォルト	範囲/値	再起動タイプ
NDB 7.3.0	バイト	16M	0 - 4294967039 (0xFFFFFEFF)	N

バックアップの作成では、データをディスクに転送するために使用される 2 つのバッファーがあります。バックアップデータバッファーは、ノードのテーブルをスキャンして記録されるデータを書き込むために使用されます。このバッファーへの書き込みが BackupWriteSize で指定されたレベルに達すると、ページがディスクに転送されます。バックアッププロセスでは、データをディスクにフラッシュしながら、スペースがなくなるまでこのバッファーに書き込み続けることができます。これが発生すると、バックアッププロセスでスキャンを

一時停止し、ディスク書き込みがある程度完了してメモリーが解放され、スキャンを続行できるようになるまで待機します。

このパラメータのデフォルト値は 16M バイトです。

- BackupLogBufferSize

有効なバージョン	型/単位	デフォルト	範囲/値	再起動タイプ
NDB 7.3.0	バイト	16M	0 - 4294967039 (0xFFFFFEFF)	N

バックアップログバッファーはバックアップデータバッファーと同じような役割を果たしますが、バックアップの実行中に行われたテーブル書き込みのログを生成するために使用されます。これらのページの書き込みにはバックアップデータバッファーと同じ原則が適用されますが、バックアップログバッファーのスペースがなくなると、バックアップは失敗します。そのため、バックアップログバッファーのサイズは、バックアップ実行中の書き込みアクティビティーによって発生する負荷を処理するのに十分な大きさである必要があります。セクション18.5.3.3「MySQL Cluster バックアップ用の構成」を参照してください。

ほとんどのアプリケーションでは、このパラメータのデフォルト値で十分です。実際、バックアップログバッファーがいっぱいになった場合より、ディスク書き込みの速度が不十分な場合の方が、バックアップが失敗する可能性は高くなります。ディスクサブシステムがアプリケーションから発生する書き込み負荷に合わせて構成されていない場合は、クラスタが必要な操作を実行できない可能性が高くなります。

ディスクやネットワーク接続よりもプロセッサがボトルネックになるような方法でクラスタノードを構成することをお勧めします。

このパラメータのデフォルト値は 16M バイトです。

- BackupMemory

有効なバージョン	型/単位	デフォルト	範囲/値	再起動タイプ
NDB 7.3.0	バイト	32M	0 - 4294967039 (0xFFFFFEFF)	N

このパラメータは、単純に BackupDataBufferSize と BackupLogBufferSize の合計です。

MySQL Cluster NDB 7.3 以降では、このパラメータのデフォルト値は 16M バイト + 16M バイト = 32M バイトです。

> **重要**
>
> BackupDataBufferSize と BackupLogBufferSize の合計が BackupMemory のデフォルト値を超える場合は、config.ini ファイルでこのパラメータをこれらの合計に明示的に設定する必要があります。

- BackupReportFrequency

有効なバージョン	型/単位	デフォルト	範囲/値	再起動タイプ
NDB 7.3.0	秒	0	0 - 4294967039 (0xFFFFFEFF)	N

このパラメータは、バックアップ中に管理クライアントでバックアップステータスレポートを発行する頻度と、そのようなレポートをクラスタログに書き込む頻度を制御します (クラスタイベントロギングで許可するように構成されている場合。ロギングとチェックポイントを参照してください)。BackupReportFrequency は、バックアップステータスレポート間の時間 (秒単位) を表します。

デフォルト値は 0 です。

- BackupWriteSize

有効なバージョン	型/単位	デフォルト	範囲/値	再起動タイプ
NDB 7.3.0	バイト	256K	2K - 4294967039 (0xFFFFFEFF)	N

このパラメータは、バックアップログおよびバックアップデータバッファーによってディスクに書き込まれるメッセージのデフォルトサイズを指定します。

このパラメータのデフォルト値は 256K バイトです。

- BackupMaxWriteSize

有効なバージョン	型/単位	デフォルト	範囲/値	再起動タイプ
NDB 7.3.0	バイト	1M	2K - 4294967039 (0xFFFFFEFF)	N

このパラメータは、バックアップログおよびバックアップデータバッファーによってディスクに書き込まれるメッセージの最大サイズを指定します。

このパラメータのデフォルト値は 1M バイトです。

> **重要**
>
> これらのパラメータを指定するときは、次の関係が有効である必要があります。そうでない場合は、データノードを起動できません。

- BackupDataBufferSize >= BackupWriteSize + 188K バイト
- BackupLogBufferSize >= BackupWriteSize + 16K バイト
- BackupMaxWriteSize >= BackupWriteSize

MySQL Cluster のリアルタイムパフォーマンスパラメータ

このセクションで説明する [ndbd] パラメータは、マルチプロセッサのデータノードホスト上の特定の CPU に対するスレッドのスケジューリングとロックで使用されます。

> **注記**
>
> これらのパラメータを使用するには、システムの root としてデータノードプロセスを実行する必要があります。

- LockExecuteThreadToCPU

有効なバージョン	型/単位	デフォルト	範囲/値	再起動タイプ
NDB 7.3.0	CPU ID	64K	0 - 64K	N

ndbd で使用する場合、このパラメータ (現在は文字列) は NDBCLUSTER の実行スレッドを処理するために割り当てられる CPU の ID を指定します。ndbmtd で使用する場合、このパラメータの値は実行スレッドを処理するために割り当てられる CPU ID のカンマ区切りリストです。リスト内の各 CPU ID は、0-65535 の (これらを含む) 範囲の整数にします。

指定する ID の数は、MaxNoOfExecutionThreads によって指定される実行スレッドの数と一致するようにしてください。ただし、このパラメータを使用したときに、特定の順序でスレッドが CPU に割り当てられる保証はありません。ThreadConfig を使用すると、このようなより細かい制御が可能になります。

LockExecuteThreadToCPU にデフォルト値はありません。

- LockMaintThreadsToCPU

有効なバージョン	型/単位	デフォルト	範囲/値	再起動タイプ
NDB 7.3.0	CPU ID	[none]	0 - 64K	N

このパラメータは、NDBCLUSTER の管理スレッドを処理するために割り当てられる CPU の ID を指定します。

このパラメータの値は、0-65535 の (これらを含む) 範囲の整数です。MySQL Cluster NDB 7.3 以降では、デフォルト値はありません。

- RealtimeScheduler

有効なバージョン	型/単位	デフォルト	範囲/値	再起動タイプ
NDB 7.3.0	ブール	false	true、false	N

このパラメータを 1 に設定すると、データノードスレッドのリアルタイムスケジューリングが有効になります。

MySQL Cluster NDB 7.3.3 より前では、このパラメータは ndbmtd を実行するノードで正常に機能しませんでした。(Bug #16961971)

デフォルトは 0 (スケジューリングが無効) です。

- SchedulerExecutionTimer

有効なバージョン	型/単位	デフォルト	範囲/値	再起動タイプ
NDB 7.3.0	マイクロ秒	50	0 - 11000	N

このパラメータは、スレッドがスケジューラで実行されてから送信されるまでの時間 (ミリ秒) を指定します。これを 0 に設定すると、応答時間が最小化されます。スループットを向上させるため、この値を増やすことができますが、その代償として応答時間は長くなります。

デフォルトは 50 マイクロ秒です。弊社のテストでは、これによって高負荷のケースで実質的に要求を遅延させずにスループットが若干向上することがわかっています。

- SchedulerSpinTimer

有効なバージョン	型/単位	デフォルト	範囲/値	再起動タイプ
NDB 7.3.0	マイクロ秒	0	0 - 500	N

このパラメータは、スレッドがスケジューラで実行されてからスリープ状態になるまでの時間 (ミリ秒) を指定します。

デフォルト値は 0 です。

- BuildIndexThreads

有効なバージョン	型/単位	デフォルト	範囲/値	再起動タイプ
NDB 7.3.0	数値	0	0 - 128	S

このパラメータは、システムまたはノードの起動中に順序付けされたインデックスの再構築時、および ndb_restore --rebuild-indexes の実行時に作成するスレッドの数を指定します。これは、データノードあたり複数のフラグメントがテーブルに存在するとき (たとえば、CREATE TABLE で MAX_ROWS オプションが使用されたときなど) にのみサポートされます。

このパラメータを 0 (デフォルト) に設定すると、マルチスレッドによる順序付けされたインデックスの構築が無効になります。

このパラメータは、ndbd または ndbmtd を使用するときにサポートされます。

データノードの初期再起動中にマルチスレッドによる構築を有効にするには、TwoPassInitialNodeRestartCopy データノード構成パラメータを TRUE に設定します。

- TwoPassInitialNodeRestartCopy

有効なバージョン	型/単位	デフォルト	範囲/値	再起動タイプ
NDB 7.3.0	ブール	false	true、false	N

データノードの初期再起動でマルチスレッドによる順序付けされたインデックスの構築を有効にするには、この構成パラメータを TRUE に設定します。これにより、ノードの初期再起動中にデータの 2 パスコピーが有効になります。

同時に BuildIndexThreads を 0 以外の値に設定する必要があります。

- Numa

有効なバージョン	型/単位	デフォルト	範囲/値	再起動タイプ
NDB 7.3.0	ブール	1	...	N

NDB は、Non-Uniform Memory Access の設定と (発生するタイムアウトによる) マルチ CPU システムの影響をかなり受けやすくなっています。この事実と、ほとんどの MySQL Cluster ユーザーが numactl を採用しないことから、Linux システムで ndbd を実行しているときは、NUMA のサポートはデフォルトで無視されます。Linux システムが NUMA をサポートしていて、データノードのメモリーを NUMA で制御する必要がある場合は、このパラメータを 0 に設定できます。

Numa 構成パラメータは、libnuma.so がインストールされている Linux システムでのみサポートされます。

マルチスレッドの構成パラメータ (ndbmtd)　ndbmtd は、デフォルトではシングルスレッドプロセスとして動作するため、2 つの方法のいずれかを使用して、複数のスレッドを使用するように構成する必要があります。どちらの場合も、config.ini ファイルに構成パラメータを設定する必要があります。1 つ目の方法では、MaxNoOfExecutionThreads 構成パラメータに適切な値を設定します。MySQL Cluster NDB 7.3 以降では、ThreadConfig を使用して ndbmtd のマルチスレッドに関するより複雑なルールを設定できる 2 つ目の方法もサポートされます。このパラメータとマルチスレッドデータノードでの使用について、次のいくつかの段落で説明します。

- MaxNoOfExecutionThreads

有効なバージョン	型/単位	デフォルト	範囲/値	再起動タイプ
NDB 7.3.0	整数	2	2 - 36	IS
NDB 7.3.3	整数	2	2 - 72	IS

このパラメータは、ndbmtd によって使用される実行スレッドの数を制御します。最大数は 72 です (MySQL Cluster NDB 7.3.3 より前では、これは 36 でした)。このパラメータは、config.ini ファイルの [ndbd] または [ndbd default] セクションに設定しますが、ndbmtd 専用であり、ndbd には適用されません。

MaxNoOfExecutionThreads を設定すると、storage/ndb/kernel/src/vm/mt_thr_config.cpp ファイルのマトリックスで決定される各タイプのスレッド数が設定されます。この表は、MySQL Cluster NDB 7.4.3 以降で MaxNoOfExecutionThreads に指定できる値に対応するこれらのスレッド数を示しています (Bug #75220、Bug #20215689)。(以前のバージョンの MySQL Cluster に適用されるマトリックスに関する情報を示す表は、このあとにあります。)MySQL Cluster NDB 7.4.3 で変更された値を含む行は、強調表示のテキストで示しています。

MaxNoOfExecutionThreads の値	LDM スレッド	TC スレッド	送信スレッド	受信スレッド
0 .. 3	1	1	0	1
4 .. 6	2	1	0	1
7 .. 8	4	1	0	1
9	4	2	0	1
10	4	2	1	1
11	4	3	1	1
12	4	3	1	2
13	4	3	2	2
14	4	4	2	2
15	4	5	2	2
16	8	3	1	2
17	8	4	1	2
18	8	4	2	2
19	8	5	2	2
20	10	4	2	2

MaxNoOfExecutionThreads の値	LDM スレッド	TC スレッド	送信スレッド	受信スレッド
21	10	5	2	2
22	10	5	2	3
23	10	6	2	3
24	12	5	2	3
25	12	6	2	3
26	12	6	3	3
27	12	7	3	3
28	12	7	3	4
29	12	8	3	4
30	12	8	4	4
31	12	9	4	4
32	16	8	3	3
33	16	8	3	4
34	16	8	4	4
35	16	9	4	4
36	16	10	4	4
37	16	10	4	5
38	16	11	4	5
39	16	11	5	5
40	20	10	4	4
41	20	10	4	5
42	20	11	4	5
43	20	11	5	5
44	20	12	5	5
45	20	12	5	6
46	20	13	5	6
47	20	13	6	6
48	24	12	5	5
49	24	12	5	6
50	24	13	5	6
51	24	13	6	6
52	24	14	6	6
53	24	14	6	7
54	24	15	6	7
55	24	15	7	7
56	24	16	7	7
57	24	16	7	8
58	24	17	7	8
59	24	17	8	8
60	24	18	8	8
61	24	18	8	9
62	24	19	8	9
63	24	19	9	9

MaxNoOfExecutionThreads の値	LDM スレッド	TC スレッド	送信スレッド	受信スレッド
64	32	16	7	7
65	32	16	7	8
66	32	17	7	8
67	32	17	8	8
68	32	18	8	8
69	32	18	8	9
70	32	19	8	9
71	32	20	8	9
72	32	20	8	10

次の表に、MySQL CLuster NDB 7.4.2 以前の MaxNoOfExecutionThreads の値に対応する各タイプのスレッド数を取得する方法を示します。この表は MySQL Cluster NDB 7.3.2 以前でも使用できますが、これらのバージョンでは MaxNoOfExecutionThreads の最大値が 36 であるため、この表の 36 より大きい値に対応する行は、MySQL Cluster NDB 7.3.3 より前には適用されません。

MaxNoOfExecutionThreads の値	LDM スレッド	TC スレッド	送信スレッド	受信スレッド
0 .. 3	1	1	0	1
4 .. 6	2	1	0	1
7 .. 8	4	1	0	1
9	4	2	0	1
10	4	2	1	1
11	4	3	1	1
12	4	3	1	2
13	4	3	2	2
14	4	4	2	2
15	4	5	2	2
16	8	3	1	2
17	8	4	1	2
18	8	4	2	2
19	8	5	2	2
20	8	5	2	3
21	8	5	3	3
22	8	6	3	3
23	8	7	3	3
24	12	5	2	3
25	12	6	2	3
26	12	6	3	3
27	12	7	3	3
28	12	7	3	4
29	12	8	3	4
30	12	8	4	4
31	12	9	4	4
32	16	8	3	3
33	16	8	3	4
34	16	8	4	4

MaxNoOfExecutionThreads の値	LDM スレッド	TC スレッド	送信スレッド	受信スレッド
35	16	9	4	4
36	16	10	4	4
37	16	10	4	5
38	16	11	4	5
39	16	11	5	5
40	16	12	5	5
41	16	12	5	6
42	16	13	5	6
43	16	13	6	6
44	16	14	6	6
45	16	14	6	7
46	16	15	6	7
47	16	15	7	7
48	24	12	5	5
49	24	12	5	6
50	24	13	5	6
51	24	13	6	6
52	24	14	6	6
53	24	14	6	7
54	24	15	6	7
55	24	15	7	7
56	24	16	7	7
57	24	16	7	8
58	24	17	7	8
59	24	17	8	8
60	24	18	8	8
61	24	18	8	9
62	24	19	8	9
63	24	19	9	9
64	32	16	7	7
65	32	16	7	8
66	32	17	7	8
67	32	17	8	8
68	32	18	8	8
69	32	18	8	9
70	32	19	8	9
71	32	20	8	9
72	32	20	8	10

MySQL Cluster NDB 7.3 以降では、SUMA (レプリケーション) スレッドが常に 1 つ存在します。

LDM スレッドの数が NoOfFragmentLogParts を超えないようにする必要があります。このため、このパラメータの値がデフォルト (4) の場合は、MaxNoOfExecutionThreads を 16 以上に設定したときに、この値も増やす

必要があります。つまり、NoOfFragmentLogParts を、前の表に示された MaxNoOfExecutionThreads のこのパラメータの値に対応する LDM スレッド数の値に設定してください。

スレッドタイプについては、このセクションで後述します (ThreadConfig を参照してください)。

このパラメータを許容範囲外の値に設定すると、管理サーバーの起動が中止され、次の工ラーが発生します: Error line number: Illegal value value for parameter MaxNoOfExecutionThreads。

MaxNoOfExecutionThreads では、値 0 または 1 は NDB の内部で 2 に切り上げられるため、2 がこのパラメータのデフォルト値および最小値とみなされます。

MaxNoOfExecutionThreads は、通常、使用可能な CPU スレッド数と同じ値に設定し、各タイプのスレッド数を一般的なワークロードに合わせて割り当てることを目的としています。指定された CPU に特定のスレッドを割り当てるものではありません。提供された設定を変更することが望ましい場合や、スレッドを CPU にバインドする場合は、代わりに必要なタイプ、CPU、またはその両方に直接スレッドを割り当てることができる ThreadConfig を使用するようにしてください。

マルチスレッドのデータノードプロセスでは、少なくともここに示す 5 個のスレッドを常に生成します。

- 1 個のローカルクエリーハンドラ (LDM) スレッド
- 1 個のトランザクションコーディネータ (TC) スレッド
- 1 個の送信スレッド

  (このセクションの別の場所で説明するように、独立した送信スレッドを採用しないようにすることもできます。)
- 1 個の受信スレッド
- 1 個のサブスクリプションマネージャー (SUMA またはレプリケーション) スレッド

LDM スレッドの数を変更するには、このパラメータと ThreadConfig のどちらを使用して変更する場合でも、常にシステムの再起動が必要です。クラスタの IndexMemory の使用率が 50% を超える場合、これを変更するにはクラスタの初期再起動が必要です。(このような場合は、最大で 30-35% の IndexMemory 使用率が推奨されます。)それ以外の場合は、ノード間でリソースの使用率と LDM スレッドの割り当てのバランスを取ることができず、結果として LDM スレッドの利用が不十分または過剰になり、最終的にデータノードに障害が発生します。

- NoOfFragmentLogParts

有効なバージョン	型/単位	デフォルト	範囲/値	再起動タイプ
NDB 7.3.0	数値	4	4、8、12、16	IN
NDB 7.3.3	数値	4	4、8、12、16、24、32	IN

この ndbmtd に属する Redo ログのログファイルグループの数を設定します。MySQL Cluster NDB 7.3.3 より前では、この値は 4 から 16 までの (これらを含む) 4 の偶数倍である必要があります。MySQL Cluster NDB 7.3.3 以降では、最大は 32 です。以前と同様に、値は 4 の偶数倍である必要があります。

ndbmtd が使用する LQH スレッドの数が NoOfFragmentLogParts を超えないようにする必要があります。また、MaxNoOfExecutionThreads を増やすとこの数が増える場合があります。詳細は、このパラメータの説明を参照してください。

- ThreadConfig

有効なバージョン	型/単位	デフォルト	範囲/値	再起動タイプ
NDB 7.3.0	文字列	''	...	IS

このパラメータは、ndbmtd で異なるタイプのスレッドを別の CPU に割り当てるために使用されます。この値は、次の構文を持つ形式の文字列です。

```
ThreadConfig := entry[,entry[,...]]

entry := type={param[,param[,...]]}

type := ldm | main | recv | send | rep | io
```

```
param := count=number | cpubind=cpu_list
```

パラメータのリストを囲む中括弧 ({...}) は、リスト内にパラメータが 1 つしかない場合でも必要です。

param (パラメータ) には、特定タイプのスレッドの数 (count)、特定タイプのスレッドのバインド先となる CPU (cpubind)、またはその両方を指定します。

type 属性は、NDB のスレッドタイプを表します。MySQL Cluster NDB 7.3 以降でサポートされるスレッドタイプと各タイプで許容される count 値の範囲を、次のリストに示します。

- ldm: データを処理するローカルクエリーハンドラ (DBLQH カーネルブロック)。使用される LDM スレッドの数が多いほど、データがより高度にパーティション化されます。各 LDM スレッドには、固有のデータおよびインデックスパーティションのセットと固有の Redo ログが保持されます。MySQL Cluster NDB 7.3.3 以降では、このようなスレッドの最大数は 32 です。MySQL Cluster NDB 7.3.2 以前では、最大は 16 です。

  > **重要**
  >
  > LDM スレッドの数を変更するには、システムの再起動をクラスタの操作に対して効果的かつ安全に行う必要があります。(これは、MaxNoOfExecutionThreads を使用して実行する場合にもあてはまります。)IndexMemory の使用率が 50% を超える場合は、クラスタの初期再起動が必要です。このような場合は、IndexMemory の使用率を最大で 30-35% にすることをお勧めします。そうでない場合は、IndexMemory および DataMemory の使用率と LDM スレッドのノード間の割り当てのバランスが取れず、最終的にはデータノードの障害につながる可能性があります。

- tc: 進行中のトランザクションの状態を含むトランザクションコーディネータスレッド (DBTC カーネルブロック)。MySQL Cluster NDB 7.3 では、TC スレッドの数を構成できます。MySQL Cluster NDB 7.3.3 以降では、合計 32 個が使用可能です。以前は 16 個でした。

  新しいトランザクションをそれぞれ 1 つの新しい TC スレッドに適切に割り当てることができます。ほとんどの場合、この実行が保証されるには、2 つの LDM スレッドに対して 1 つの TC スレッドで十分です。読み取りの数に比べて書き込みの数が少ないケースでは、4 つの LQH スレッドあたり 1 つの TC スレッドがあればトランザクション状態を維持できる可能性があります。逆に、非常に多くの更新を実行するアプリケーションでは、LDM スレッドに対する TC スレッドの比率を 1 に近づける (たとえば、4 つの LDM スレッドに対して TC スレッドを 3 つにする) 必要がある場合があります。

  範囲: (NDB 7.3.3 以降) 1-32、(NDB 7.3.2 以前) 1-16。

- main: スキーマ管理を提供するデータディクショナリおよびトランザクションコーディネータ (DBDIH および DBTC カーネルブロック)。これは、常に 1 つの専用スレッドで処理されます。

  範囲: 1 のみ。

- recv: 受信スレッド (CMVMI カーネルブロック)。各受信スレッドは、MySQL Cluster 内のほかのノードと通信するための 1 つ以上のソケット (ノードあたり 1 ソケット) を処理します。MySQL Cluster NDB 7.3 以降では、複数の受信スレッドがサポートされます。MySQL Cluster NDB 7.3.2 以前では、このようなスレッドの最大数は 8 です。MySQL Cluster NDB 7.3.3 以降では、最大は 16 です。

  範囲: (NDB 7.3.3 以降) 1-16、(NDB 7.3.2 以前) 1-8。

- send: 送信スレッド (CMVMI カーネルブロック)。スループットを向上させるため、1 つ以上 (最大 8 個) の独立した専用スレッドから送信を実行できます。

  以前は、すべてのスレッドが自身の送信を直接処理していました。今でも、送信スレッドの数を 0 に設定することでこれを行うことができます (MaxNoOfExecutionThreads を 9 に設定した場合もこれが行われます)。これを行うと、スループットに悪影響を及ぼす可能性がありますが、場合によっては待機時間が減る可能性もあります。

  範囲: (NDB 7.3.3 以降) 0-16、(NDB 7.3.2 以前) 0-8。

- rep: レプリケーションスレッド (SUMA カーネルブロック)。非同期のレプリケーション操作は、常に 1 つの専用スレッドで処理されます。

  範囲: 1 のみ。

- io: ファイルシステムおよびその他の操作。これらは、要求の厳しいタスクではなく、常に 1 つの I/O 専用スレッドでまとめて処理されます。

  範囲: 1 のみ。

簡単な例:

```
# Example 1.

ThreadConfig=ldm={count=2,cpubind=1,2},main={cpubind=12},rep={cpubind=11}

# Example 2.

Threadconfig=main={cpubind=0},ldm={count=4,cpubind=1,2,5,6},io={cpubind=3}
```

通常、スレッドの使用法を構成するときは、データノードホストの 1 つ以上の CPU をオペレーティングシステムおよびその他のタスク用に確保します。したがって、24 個の CPU を搭載したホストマシンでは、20 個の CPU スレッド (8 個の LDM スレッド、4 個の TC スレッド (LDM スレッドの半分)、3 個の送信スレッド、3 個の受信スレッド、およびスキーマ管理、非同期レプリケーション、I/O 操作のそれぞれに 1 個のスレッド) を使用できます (ほかの用途のために 4 個を残します)。(これは、MaxNoOfExecutionThreads を 20 に設定したときに使用されるスレッドの分布とほぼ同じです。)次の ThreadConfig 設定では、これらの割り当てを行い、さらにこれらのスレッドを特定の CPU にバインドしています。

```
ThreadConfig=ldm{count=8,cpubind=1,2,3,4,5,6,7,8},main={cpubind=9},io={cpubind=9}, \
rep={cpubind=10},tc{count=4,cpubind=11,12,13,14},recv={count=3,cpubind=15,16,17}, \
send{count=3,cpubind=18,19,20}
```

ほとんどの場合、ここに示す例で行なったように、main (スキーマ管理) スレッドと I/O スレッドは同じ CPU にバインドできます。

ThreadConfig を使用することで得られる高い安定性を生かすには、CPU が隔離されていて、割り込みを受けたり、オペレーティングシステムによってほかのタスクをスケジュールされたりしない必要があります。多くの Linux システムでは、/etc/sysconfig/irqbalance の IRQBALANCE_BANNED_CPUS を 0xFFFFF0 に設定し、grub.conf の isolcpus ブートオプションを使用することで、これを実現できます。具体的な情報については、オペレーティングシステムまたはプラットフォームのドキュメントを参照してください。

ディスクデータの構成パラメータ　ディスクデータの動作に影響を与える構成パラメータには、次が含まれます。

- DiskPageBufferMemory

有効なバージョン	型/単位	デフォルト	範囲/値	再起動タイプ
NDB 7.3.0	バイト	64M	4M - 1T	N

  これは、ディスク上のページをキャッシュするために使用されるスペースの量を指定するもので、config.ini ファイルの [ndbd] または [ndbd default] セクションで設定されます。これは、バイト単位で測定されます。各ページは最大 32K バイトを占有します。これは、常に N * 32K バイトのメモリーがクラスタディスクデータのストレージに使用されることを意味します (N は負でない整数です)。

  このパラメータのデフォルト値は 64M (32K バイトのページ 2000 個分) です。

  ndbinfo.diskpagebuffer テーブルをクエリーすると、不要なディスクシークを最小限に抑えるためにこのパラメータの値を増やすべきかどうか判定しやすくなります。詳細は、セクション18.5.10.12「ndbinfo diskpagebuffer テーブル」を参照してください。

- SharedGlobalMemory

有効なバージョン	型/単位	デフォルト	範囲/値	再起動タイプ
NDB 7.3.0	バイト	128M	0 - 64T	N

  このパラメータは、テーブルスペース、ログファイルグループ、Undo ファイル、およびデータファイル用のログバッファー、ディスク操作 (ページ要求や待機キューなど)、およびメタデータに使用されるメモリーの量を指定します。この共有グローバルメモリープールから、CREATE LOGFILE GROUP および ALTER LOGFILE GROUP ステートメントで使用される INITIAL_SIZE および UNDO_BUFFER_SIZE オプション (InitialLogFileGroup データノード構成パラメータの設定によって示されるこれらのオプションのデフォルト値を含む) のメモリー要件を満たすために使用されるメモリーも提供されます。SharedGlobalMemory は、config.ini 構成ファイルの [ndbd] または [ndbd default] セクションに設定でき、バイト単位で測定されます。

デフォルト値は 128M です。

- DiskIOThreadPool

有効なバージョン	型/単位	デフォルト	範囲/値	再起動タイプ
NDB 7.3.0	スレッド	2	0 - 4294967039 (0xFFFFFEFF)	N

このパラメータは、ディスクデータファイルへのアクセスに使用される未バインドスレッドの数を指定します。DiskIOThreadPool が導入される前は、ディスクデータファイルごとに 1 つのスレッドしか生成されなかったため、特に非常に大きいデータファイルを使用する場合に、パフォーマンスの問題が発生する可能性がありました。DiskIOThreadPool を使用すると、たとえば、並列で動作する複数のスレッドを使用して 1 つの大きなデータファイルにアクセスできます。

このパラメータは、ディスクデータ I/O スレッドにのみ適用されます。

このパラメータの最適な値は、使用しているハードウェアと構成によって異なり、次の要因を含みます。

- ディスクデータファイルの物理的な分布　データファイル、Undo ログファイル、およびデータノードファイルシステムを別々の物理ディスクに配置することで、パフォーマンスを向上させることができます。これらの一部またはすべてのファイルを使用してこれを行う場合は、各ディスク上のファイルを別個のスレッドで処理できるようにするために、DiskIOThreadPool を大きな値に設定できます。
- ディスクのパフォーマンスとタイプ　ディスクデータファイルの処理用に提供できるスレッドの数は、ディスクの速度とスループットにも依存します。ディスクの速度が速く、スループットが高いほど、より多くの I/O スレッドを使用できます。弊社のテスト結果では、従来のディスクよりソリッドステートディスクドライブの方が (つまり DiskIOThreadPool の値が大きいほど) はるかに多くのディスク I/O スレッドを処理できることを示しています。

このパラメータのデフォルト値は 2 です。

- ディスクデータのファイルシステムパラメータ　次のリストのパラメータを使用すると、シンボリックリンクを使用せずに MySQL Cluster ディスクデータのファイルを特定のディレクトリに配置できるようになります。

  - FileSystemPathDD

有効なバージョン	型/単位	デフォルト	範囲/値	再起動タイプ
NDB 7.3.0	ファイル名	[テキストを参照]	...	IN

このパラメータを指定すると、MySQL Cluster ディスクデータのデータファイルと Undo ログファイルが指定されたディレクトリに配置されます。データファイル、Undo ログファイル、またはその両方をオーバーライドするには、FileSystemPathDataFiles、FileSystemPathUndoFiles、またはその両方の値をこれらのパラメータの説明に従って指定します。データファイルでは CREATE TABLESPACE または ALTER TABLESPACE ステートメントの ADD DATAFILE 句にパスを指定し、Undo ログファイルでは CREATE LOGFILE GROUP または ALTER LOGFILE GROUP ステートメントの ADD UNDOFILE 句でパスを指定してオーバーライドすることもできます。FileSystemPathDD が指定されていない場合は、FileSystemPath が使用されます。

特定のデータノードで FileSystemPathDD ディレクトリが指定された場合 (config.ini ファイルの [ndbd default] セクションでこのパラメータが指定された場合を含む)、--initial を指定してデータノードを起動すると、ディレクトリ内のすべてのファイルが削除されます。

  - FileSystemPathDataFiles

有効なバージョン	型/単位	デフォルト	範囲/値	再起動タイプ
NDB 7.3.0	ファイル名	[テキストを参照]	...	IN

このパラメータを指定すると、MySQL Cluster ディスクデータのデータファイルが指定されたディレクトリに配置されます。これは、FileSystemPathDD に設定された値をオーバーライドします。特定のデータファイルのパラメータをオーバーライドするには、そのデータファイルを作成するために使用される CREATE TABLESPACE または ALTER TABLESPACE ステートメントの ADD DATAFILE 句でパスを

指定します。FileSystemPathDataFiles が指定されていない場合は、FileSystemPathDD が使用されます (FileSystemPathDD も設定されていない場合は、FileSystemPath が使用されます)。

特定のデータノードで FileSystemPathDataFiles ディレクトリが指定された場合 (config.ini ファイルの [ndbd default] セクションにこのパラメータが指定された場合を含む)、--initial を指定してそのデータノードを起動すると、ディレクトリ内のすべてのファイルが削除されます。

- FileSystemPathUndoFiles

有効なバージョン	型/単位	デフォルト	範囲/値	再起動タイプ
NDB 7.3.0	ファイル名	[テキストを参照]	...	IN

このパラメータを指定すると、MySQL Cluster ディスクデータの Undo ログファイルが指定されたディレクトリに配置されます。これは、FileSystemPathDD に設定された値をオーバーライドします。特定のデータファイルについてこのパラメータをオーバーライドするには、そのデータファイルを作成するために使用される CREATE LOGFILE GROUP または CREATE LOGFILE GROUP ステートメントの ADD UNDO 句にパスを指定します。FileSystemPathUndoFiles が指定されていない場合は、FileSystemPathDD が使用されます (FileSystemPathDD も設定されていない場合は、FileSystemPath が使用されます)。

特定のデータノードで FileSystemPathUndoFiles ディレクトリが指定された場合 (config.ini ファイルの [ndbd default] セクションにこのパラメータが指定された場合を含む)、--initial を指定してそのデータノードを起動すると、ディレクトリ内のすべてのファイルが削除されます。

詳細は、セクション18.5.12.1「MySQL Cluster ディスクデータオブジェクト」を参照してください。

- ディスクデータのオブジェクト作成パラメータ　次の 2 つのパラメータを使用すると、クラスタを最初に起動したときに、SQL ステートメントを使用せずにディスクデータのログファイルグループ、テーブルスペース、またはその両方を作成できます。

  - InitialLogFileGroup

有効なバージョン	型/単位	デフォルト	範囲/値	再起動タイプ
NDB 7.3.0	文字列	[テキストを参照]	...	S

このパラメータを使用して、クラスタの初期起動の実行時に作成されるログファイルグループを指定できます。InitialLogFileGroup は、ここに示すように指定されます。

```
InitialLogFileGroup = [name=name;] [undo_buffer_size=size;] file-specification-list

file-specification-list:
   file-specification[; file-specification[; ...]]

file-specification:
   filename:size
```

ログファイルグループの name はオプションであり、デフォルト値は DEFAULT-LG です。undo_buffer_size もオプションです。省略した場合のデフォルト値は 64M です。file-specification は、それぞれ 1 つの Undo ログファイルに対応し、file-specification-list に少なくとも 1 つ指定する必要があります。Undo ログファイルは、FileSystemPath、FileSystemPathDD、および FileSystemPathUndoFiles に設定された値に従って、CREATE LOGFILE GROUP または ALTER LOGFILE GROUP ステートメントの結果として作成された場合と同じように配置されます。

次について考えます。

```
InitialLogFileGroup = name=LG1; undo_buffer_size=128M; undo1.log:250M; undo2.log:150M
```

これは、次の SQL ステートメントと同等です。

```
CREATE LOGFILE GROUP LG1
   ADD UNDOFILE 'undo1.log'
   INITIAL_SIZE 250M
   UNDO_BUFFER_SIZE 128M
   ENGINE NDBCLUSTER;

ALTER LOGFILE GROUP LG1
   ADD UNDOFILE 'undo2.log'
   INITIAL_SIZE 150M
```

```
  ENGINE NDBCLUSTER;
```

このログファイルグループは、--initial を指定してデータノードを起動したときに作成されます。

初期のログファイルグループのリソースは、SharedGlobalMemory データノード構成パラメータの値でサイズが決定されるグローバルメモリープールから取得されます。このパラメータの設定値が小さすぎて、ログファイルグループの初期サイズまたは Undo バッファーサイズとして InitialLogFileGroup に設定された値が大きすぎる場合は、クラスタの起動時にデフォルトのログファイルグループが作成されないか、クラスタの起動が完全に失敗する可能性があります。

このパラメータを使用する場合は、常に config.ini ファイルの [ndbd default] セクションに設定するようにしてください。異なるデータノードに異なる値を設定した場合の MySQL Cluster の動作は定義されていません。

- InitialTablespace

有効なバージョン	型/単位	デフォルト	範囲/値	再起動タイプ
NDB 7.3.0	文字列	[テキストを参照]	...	S

このパラメータを使用して、クラスタの初期起動を実行したときに作成される MySQL Cluster ディスクデータのテーブルスペースを指定できます。InitialTablespace は、ここに示すように指定されます。

```
InitialTablespace = [name=name;] [extent_size=size;] file-specification-list
```

テーブルスペースの name はオプションであり、デフォルト値は DEFAULT-TS です。extent_size もオプションです。デフォルト値は 1M です。file-specification-list には、InitialLogfileGroup パラメータに示された同じ構文が使用されます。唯一の違いは、InitialTablespace で使用される file-specification がそれぞれ 1 つのデータファイルに対応することです。file-specification-list には、少なくとも 1 つを指定する必要があります。データファイルは、FileSystemPath、FileSystemPathDD、および FileSystemPathDataFiles に設定された値に従って、CREATE TABLESPACE または ALTER TABLESPACE ステートメントの結果として作成された場合と同じように配置されます。

たとえば、config.ini ファイルの [ndbd default] セクションに InitialTablespace を指定する次の行について考えます (InitialLogfileGroup と同様、異なるデータノードに異なる値を設定したときの MySQL Cluster の動作は定義されていないため、このパラメータは常に [ndbd default] セクションに設定するようにしてください)。

```
InitialTablespace = name=TS1; extent_size=8M; data1.dat:2G; data2.dat:4G
```

これは、次の SQL ステートメントと同等です。

```
CREATE TABLESPACE TS1
   ADD DATAFILE 'data1.dat'
   EXTENT_SIZE 8M
   INITIAL_SIZE 2G
   ENGINE NDBCLUSTER;

ALTER TABLESPACE TS1
   ADD DATAFILE 'data2.dat'
   INITIAL_SIZE 4G
   ENGINE NDBCLUSTER;
```

このテーブルスペースは、--initial を指定してデータノードを起動したときに作成され、その後 MySQL Cluster ディスクデータのテーブルを作成するたびに使用できます。

ディスクデータと GCP 停止エラー　　ディスクデータテーブルを使用するときに発生するエラー:Node nodeid killed this node because GCP stop was detected(エラー 2303) などは、多くの場合、「GCP 停止エラー」と呼ばれています。このようなエラーは、Redo ログが十分な速さでディスクにフラッシュされていない場合に発生します。これは通常、ディスクの速度が低く、ディスクのスループットが十分でないことが原因です。

高速なディスクを使用し、ディスクデータファイルをデータノードファイルシステムとは別のディスクに配置することで、これらのエラーを回避しやすくなります。TimeBetweenGlobalCheckpoints の値を減らすと、グローバルチェックポイントごとに書き込まれるデータの量が減りやすくなるため、グローバルチェックポイントを書き込もうとしたときに Redo ログバッファーのオーバーフローをある程度回避できる可能性があります。ただし、この値を減らすと GCP を書き込むことができる時間も減るため、注意が必要です。

前述の DiskPageBufferMemory について示した考慮事項に加えて、DiskIOThreadPool 構成パラメータを正しく設定することも非常に重要です。DiskIOThreadPool の設定値が大きすぎると、GCP 停止エラーが発生する可能性が非常に高くなります (Bug #37227)。

GCP の停止は、保存またはコミットのタイムアウトによって発生することがあります。TimeBetweenEpochsTimeout データノード構成パラメータは、コミットのタイムアウトを指定します。ただし、このパラメータを 0 に設定することで、両方のタイプのタイムアウトを無効にできます。

送信バッファーメモリーの割り当てを構成するためのパラメータ　送信バッファーメモリーは、すべてのトランスポータ間で共有されるメモリープールから動的に割り当てられます。これは、送信バッファーのサイズを必要に応じて調整できることを意味します。(以前は、クラスタ内のすべてのノードで NDB カーネルが固定サイズの送信バッファーを使用していました。これは、ノードの起動時に割り当てられ、ノードの実行中は変更できませんでした。)TotalSendBufferMemory および OverLoadLimit データノード構成パラメータを使用して、このメモリー割り当ての制限を設定できます。これらのパラメータ (および SendBufferMemory) の使用方法の詳細は、セクション18.3.2.12「MySQL Cluster の送信バッファーパラメータの構成」を参照してください。

- ExtraSendBufferMemory

  このパラメータは、TotalSendBufferMemory、SendBufferMemory、またはその両方を使用して設定されたメモリーに加えて割り当てられるトランスポータ送信バッファーメモリーの量を指定します。

- TotalSendBufferMemory

  このパラメータは、MySQL Cluster NDB 6.4.0 以降で使用可能です。これは、すべての構成済みトランスポータ間で共有される送信バッファーメモリーの、このノード上で割り当てられるメモリーの合計量を指定するために使用されます。

  このパラメータを設定する場合、許容される最小値は 256K バイト、最大値は 4294967039 です。

- ReservedSendBufferMemory

  このパラメータは、MySQL Cluster NDB 6.4.0 以降の NDBCLUSTER のソースコード内にあります。しかし、現在は有効になっていません。

  このパラメータは MySQL Cluster NDB 7.2 で非推奨になり、MySQL Cluster の今後のバージョンで削除される予定です (Bug #11760629、Bug #53053)。

TotalSendBufferMemory の動作と使用、 および MySQL Cluster での送信バッファーメモリーパラメータの構成の詳細は、セクション18.3.2.12「MySQL Cluster の送信バッファーパラメータの構成」を参照してください。

セクション18.5.13「MySQL Cluster データノードのオンライン追加」も参照してください。

Redo ログのオーバーコミット処理　Redo ログをディスクにフラッシュする時間が長すぎる場合の、データノードによる操作の処理を制御できます。これは、特定の Redo ログのフラッシュが RedoOverCommitLimit 秒より長い時間をかけて RedoOverCommitCounter 回を超える回数分行われ、保留中のトランザクションが中止されたときに発生します。これが発生すると、トランザクションを送信した API ノードは、コミットされるはずだった操作を (DefaultOperationRedoProblemAction の指定に従って) キューに配置して再試行するか、中止することで処理できます。API ノードでこのアクションが実行される前に超過が許されるタイムアウトと回数を設定するためのデータノード構成パラメータについて、次のリストで説明します。

- RedoOverCommitCounter

有効なバージョン	型/単位	デフォルト	範囲/値	再起動タイプ
NDB 7.3.0	数値	3	0 - 4294967039 (0xFFFFFEFF)	N

  特定の Redo ログをディスクに書き込もうとしたときに RedoOverCommitLimit を超過した回数がこの回数を超えると、結果としてコミットされなかったトランザクションはすべて中止され、トランザクションの発生元である API ノードは、これらのトランザクションを構成する操作を DefaultOperationRedoProblemAction の値に従って (操作をキューに入れて再試行するか、中止して) 処理します。

  RedoOverCommitCounter のデフォルト値は 3 です。0 に設定すると、この制限は無効になります。

- RedoOverCommitLimit

有効なバージョン	型/単位	デフォルト	範囲/値	再起動タイプ
NDB 7.3.0	秒	20	0 - 4294967039 (0xFFFFFEFF)	N

  このパラメータは、特定の Redo ログをディスクに書き込もうとしたときにタイムアウトするまでの上限 (秒単位) を設定します。データノードがこの Redo ログをフラッシュしようとして RedoOverCommitLimit よりも長

い時間がかかった回数が保存され、RedoOverCommitCounter と比較されます。フラッシュにかかる時間が長すぎた回数がこのパラメータの値より多くなったときは、フラッシュタイムアウトの結果としてコミットされなかったトランザクションがすべて中止されます。これが発生すると、トランザクションの発生元である API ノードは、これらのトランザクションを構成する操作を DefaultOperationRedoProblemAction の設定に従って (操作をキューに入れて再試行するか、中止することで) 処理します。

デフォルトでは、RedoOverCommitLimit は 20 秒です。0 に設定すると、Redo ログのフラッシュタイムアウトのチェックが無効になります。このパラメータは MySQL Cluster NDB 7.1.10 で追加されました。

再起動試行の制御　MaxStartFailRetries および StartFailRetryDelay データノード構成パラメータを使用すると、データノードによる起動失敗時の再起動試行を細かく制御できます。

MaxStartFailRetries は、データノードの起動を中止するまでに行われる再試行の合計数を制限します。StartFailRetryDelay は、再試行間の秒数を設定します。これらのパラメータをここに示します。

- StartFailRetryDelay

有効なバージョン	型/単位	デフォルト	範囲/値	再起動タイプ
NDB 7.3.0	符号なし	0	0 - 4294967039 (0xFFFFFEFF)	N

このパラメータを使用して、データノードによる起動失敗時の再試行間の秒数を設定します。デフォルトは 0 (遅延なし) です。

StopOnError が 0 でない場合は、このパラメータと MaxStartFailRetries の両方が無視されます。

- MaxStartFailRetries

有効なバージョン	型/単位	デフォルト	範囲/値	再起動タイプ
NDB 7.3.0	符号なし	3	0 - 4294967039 (0xFFFFFEFF)	N

このパラメータを使用して、データノードによる起動失敗時の再試行回数を制限します。デフォルトは 3 回です。

StopOnError が 0 でない場合は、このパラメータと StartFailRetryDelay の両方が無視されます。

NDB インデックス統計のパラメータ　次のリストのパラメータは、MySQL Cluster NDB 7.2.1 で導入された NDB インデックス統計の生成に関連しています。

- IndexStatAutoCreate

有効なバージョン	型/単位	デフォルト	範囲/値	再起動タイプ
NDB 7.3.0	ブール	false	false、true	S

インデックスが作成されたときの自動統計収集を有効または無効にします。デフォルトで無効になっています。

このパラメータは MySQL Cluster NDB 7.2.1 で追加されました。

- IndexStatAutoUpdate

有効なバージョン	型/単位	デフォルト	範囲/値	再起動タイプ
NDB 7.3.0	ブール	false	false、true	S

インデックス変更のモニタリングを有効または無効にし、変更が検出される自動統計更新をトリガーします。更新をトリガーするために必要な変更の量およびレベルは、IndexStatTriggerPct および IndexStatTriggerScale オプションの設定によって決定されます。

このパラメータは MySQL Cluster NDB 7.2.1 で追加されました。

- IndexStatSaveSize

有効なバージョン	型/単位	デフォルト	範囲/値	再起動タイプ
NDB 7.3.0	バイト	32768	0 - 4294967039 (0xFFFFFEFF)	IN

NDB システムテーブルおよび mysqld メモリーキャッシュに保存される特定のインデックスの統計に対して許容される最大スペース (バイト単位)。これは、IndexMemory を消費します。

サイズ制限に関係なく、常に少なくとも 1 つのサンプルが生成されます。このサイズは、IndexStatSaveScale によって調整されます。

このパラメータは MySQL Cluster NDB 7.2.1 で追加されました。

大規模なインデックスでは、IndexStatSaveSize に指定されたサイズが IndexStatTriggerPct に 0.01 を掛けた値によって調整されます。これは、さらにインデックスサイズの底 2 の対数で乗算されます。IndexStatTriggerPct を 0 に設定すると、この調整が無効になります。

- IndexStatSaveScale

有効なバージョン	型/単位	デフォルト	範囲/値	再起動タイプ
NDB 7.3.0	パーセンテージ	100	0 - 4294967039 (0xFFFFFEFF)	IN

大規模なインデックスでは、IndexStatSaveSize に指定されたサイズが IndexStatTriggerPct に 0.01 を掛けた値によって調整されます。これは、さらにインデックスサイズの底 2 の対数で乗算されます。IndexStatTriggerPct を 0 に設定すると、この調整が無効になります。

- IndexStatTriggerPct

有効なバージョン	型/単位	デフォルト	範囲/値	再起動タイプ
NDB 7.3.0	パーセンテージ	100	0 - 4294967039 (0xFFFFFEFF)	IN

インデックス統計更新をトリガーする更新内の変更割合。この値は、IndexStatTriggerScale によって調整されます。このトリガーを完全に無効にするには、IndexStatTriggerPct を 0 に設定します。

このパラメータは MySQL Cluster NDB 7.2.1 で追加されました。

- IndexStatTriggerScale

有効なバージョン	型/単位	デフォルト	範囲/値	再起動タイプ
NDB 7.3.0	パーセンテージ	100	0 - 4294967039 (0xFFFFFEFF)	IN

大規模なインデックスでは、この量に 0.01 を掛けた値で IndexStatTriggerPct を調整します。値 0 で調整が無効になります。

このパラメータは MySQL Cluster NDB 7.2.1 で追加されました。

- IndexStatUpdateDelay

有効なバージョン	型/単位	デフォルト	範囲/値	再起動タイプ
NDB 7.3.0	秒	60	0 - 4294967039 (0xFFFFFEFF)	IN

特定のインデックスの自動インデックス統計更新間の最小遅延 (秒数)。この変数を 0 に設定すると、遅延が無効になります。デフォルトは 60 秒です。

このパラメータは MySQL Cluster NDB 7.2.1 で追加されました。

18.3.2.7 MySQL Cluster 内の SQL ノードおよびその他の API ノードの定義

config.ini ファイルの [mysqld] および [api] セクションでは、クラスタデータへのアクセスに使用される MySQL サーバー (SQL ノード) およびその他のアプリケーション (API ノード) が定義されます。示されているどのパラ

メータも必須ではありません。コンピュータ名またはホスト名が指定されていない場合は、任意のホストでこの SQL または API ノードを使用できます。

一般的には、[mysqld] セクションはクラスタへの SQL インタフェースを提供する MySQL サーバーを示すために使用され、[api] セクションはクラスタデータにアクセスする mysqld プロセス以外のアプリケーションのために使用されますが、この 2 つの指定は実際には同義です。たとえば、SQL ノードとして機能する MySQL サーバーのパラメータを [api] セクションに指定できます。

> **注記**
>
> MySQL Cluster 用の MySQL サーバーオプションについては、セクション 18.3.4.2「MySQL Cluster 用の MySQL Server オプション」を参照してください。MySQL Cluster に関連する MySQL サーバーシステム変数については、セクション 18.3.4.3「MySQL Cluster のシステム変数」を参照してください。

- Id

有効なバージョン	型/単位	デフォルト	範囲/値	再起動タイプ
NDB 7.3.0	符号なし	[none]	1 - 255	IS

Id は、すべてのクラスタ内部メッセージ内でノードを識別するために使用される整数値です。許容される値の範囲は、1-255 (これらを含む) です。この値は、ノードのタイプに関係なく、クラスタ内の各ノードで一意である必要があります。

> **注記**
>
> データノード ID は、使用する MySQL Cluster のバージョンに関係なく、49 未満である必要があります。多数のデータノードを配備する予定の場合は、API ノード (および管理ノード) のノード ID を 48 より大きい値に制限することをお勧めします。

NodeId は、API ノードを識別するときに使用することが推奨されるパラメータ名です。(Id は、下位互換性に引き続き対応しますが、現在は非推奨であり、使用時に警告を生成します。また、これは今後削除される予定です。)

- ConnectionMap

有効なバージョン	型/単位	デフォルト	範囲/値	再起動タイプ
NDB 7.3.0	文字列	[none]	...	N

接続するデータノードを指定します。

- NodeId

有効なバージョン	型/単位	デフォルト	範囲/値	再起動タイプ
NDB 7.3.0	符号なし	[none]	1 - 255	IS

NodeId は、すべてのクラスタ内部メッセージ内でノードを識別するために使用される整数値です。許容される値の範囲は、1-255 (これらを含む) です。この値は、ノードのタイプに関係なく、クラスタ内の各ノードで一意である必要があります。

> **注記**
>
> データノード ID は、使用する MySQL Cluster のバージョンに関係なく、49 未満である必要があります。多数のデータノードを配備する予定の場合は、API ノード (および管理ノード) のノード ID を 48 より大きい値に制限することをお勧めします。

NodeId は、管理ノードを識別するときに使用することが推奨されるパラメータ名です。MySQL Cluster の非常に古いバージョンでは、この目的のために エイリアス (Id) が使用されていました。これは、下位互換性に引き続き対応していますが、現在は非推奨であり、使用時に警告を生成します。また、MySQL Cluster の今後のリリースで削除される予定です。

- ExecuteOnComputer

有効なバージョン	型/単位	デフォルト	範囲/値	再起動タイプ
NDB 7.3.0	name	[none]	...	S

これは、構成ファイルの [computer] セクションに定義されたいずれかのコンピュータ (ホスト) に設定されている Id を参照します。

- HostName

有効なバージョン	型/単位	デフォルト	範囲/値	再起動タイプ
NDB 7.3.0	名前または IP アドレス	[none]	...	N

このパラメータを指定すると、SQL ノード (API ノード) が配置されるコンピュータのホスト名が定義されます。ホスト名を指定するには、このパラメータまたは ExecuteOnComputer のいずれかが必要です。

config.ini ファイルの特定の [mysql] または [api] セクションに HostName または ExecuteOnComputer が指定されていない場合、SQL または API ノードはネットワーク接続を確立できる任意のホストから対応する「スロット」を使用して管理サーバーホストマシンに接続できます。これは、ほかに指定されていない場合 localhost が HostName として使用されるデータノードのデフォルトの動作とは異なります。

- ArbitrationRank

有効なバージョン	型/単位	デフォルト	範囲/値	再起動タイプ
NDB 7.3.0	0-2	0	0 - 2	N

このパラメータは、アービトレータとして機能できるノードを定義します。管理ノードと SQL ノードの両方がアービトレータになることができます。値 0 は、指定されたノードがアービトレータとして使用されないことを意味します。値 1 は、ノードにアービトレータとしての高い優先度を与えます。値 2 は低い優先度を与えます。通常の構成では、管理サーバーの ArbitrationRank が 1 (管理ノードのデフォルト) に設定され、各 SQL ノードでは 0 (SQL ノードのデフォルト) に設定されるため、管理サーバーがアービトレータとして使用されます。

すべての管理および SQL ノードで ArbitrationRank を 0 に設定すると、アービトレーションを完全に無効化できます。このパラメーラをオーバーライドしてアービトレーションを制御することもできます。そのためには、config.ini グローバル構成ファイルの [ndbd default] セクションに Arbitration パラメータを設定します。

- ArbitrationDelay

有効なバージョン	型/単位	デフォルト	範囲/値	再起動タイプ
NDB 7.3.0	ミリ秒	0	0 - 4294967039 (0xFFFFFEFF)	N

このパラメータを 0 (デフォルト) 以外の値に設定すると、アービトレーション要求に対するアービトレータの応答が指定されたミリ秒数だけ遅延します。通常、この値を変更する必要はありません。

- BatchByteSize

有効なバージョン	型/単位	デフォルト	範囲/値	再起動タイプ
NDB 7.3.0	バイト	16K	1024 - 1M	N

フルテーブルスキャンまたはインデックスの範囲スキャンに変換されるクエリーでは、最良のパフォーマンスを得るため、適切にサイズ調整されたバッチでレコードをフェッチすることが重要です。レコード数 (BatchSize) とバイト数 (BatchByteSize) の両方で、適切なサイズを設定できます。実際のバッチサイズは両方のパラメータによって制限されます。

このパラメータの設定方法によっては、クエリーの実行速度の変化率が 40% を超える可能性があります。

このパラメータはバイト単位で測定されます。MySQL Cluster NDB 7.3 以降では、デフォルト値は 16K です。

- BatchSize

有効なバージョン	型/単位	デフォルト	範囲/値	再起動タイプ
NDB 7.3.0	レコード	256	1 - 992	N

このパラメータはレコード数で測定され、デフォルトで 256 に設定されます。最大サイズは 992 です。

- ExtraSendBufferMemory

有効なバージョン	型/単位	デフォルト	範囲/値	再起動タイプ
NDB 7.3.0	バイト	0	0 - 4294967039 (0xFFFFFEFF)	N

このパラメータは、TotalSendBufferMemory、SendBufferMemory、またはその両方を使用して設定されたメモリーに加えて割り当てられるトランスポータ送信バッファーメモリーの量を指定します。

- HeartbeatThreadPriority

有効なバージョン	型/単位	デフォルト	範囲/値	再起動タイプ
NDB 7.3.0	文字列	[none]	...	S

このパラメータを使用して、管理および API ノードのハートビートスレッドのスケジューリングポリシーと優先度を設定します。このパラメータを設定するための構文をここに示します。

```
HeartbeatThreadPriority = policy[, priority]

policy:
  {FIFO | RR}
```

このパラメータを設定するときは、ポリシーを指定する必要があります。これは、FIFO (先入れ先出し) または RR (ラウンドロビン) のいずれかです。このあとに、オプションで優先度 (整数) を指定できます。

- MaxScanBatchSize

有効なバージョン	型/単位	デフォルト	範囲/値	再起動タイプ
NDB 7.3.0	バイト	256K	32K - 16M	N

バッチサイズは、各データノードから送信される各バッチのサイズです。多数のノードから並列で受信される過大なデータ量から MySQL サーバーを保護するため、ほとんどのスキャンは並列で実行されます。このパラメータは、すべてのノードの合計バッチサイズに対する制限を設定します。

このパラメータのデフォルトの値は 256K バイトです。最大サイズは 16M バイトです。

- TotalSendBufferMemory

有効なバージョン	型/単位	デフォルト	範囲/値	再起動タイプ
NDB 7.3.0	バイト	256K	0 - 4294967039 (0xFFFFFEFF)	N

このパラメータは、すべての構成済みトランスポータ間で共有される送信バッファーメモリーの、このノードに割り当てられるメモリー合計量を決定するために使用されます。

このパラメータを設定する場合、許容される最小値は 256K バイト、最大値は 4294967039 です。TotalSendBufferMemory の動作と使用、および MySQL Cluster での送信バッファーメモリーパラメータの構成の詳細は、セクション18.3.2.12「MySQL Cluster の送信バッファーパラメータの構成」を参照してください。

- AutoReconnect

有効なバージョン	型/単位	デフォルト	範囲/値	再起動タイプ
NDB 7.3.0	ブール	false	true、false	N

このパラメータは、デフォルトで false です。これによって、切断された API ノード (SQL ノードとして機能する MySQL サーバーを含む) が既存の接続を再使用しようとせずに、クラスタへの新しい接続を強制的に使用するようになります (接続を再使用すると、動的に割り当てられたノード ID を使用するときに問題が発生することがあるため)。(Bug #45921)

> **注記**
>
> このパラメータは、NDB API を使用してオーバーライドできます。詳細は、Ndb_cluster_connection::set_auto_reconnect() および Ndb_cluster_connection::get_auto_reconnect() を参照してください。

- DefaultOperationRedoProblemAction

有効なバージョン	型/単位	デフォルト	範囲/値	再起動タイプ
NDB 7.3.0	列挙	QUEUE	ABORT、QUEUE	S

このパラメータは、(RedoOverCommitLimit および RedoOverCommitCounter とともに) Redo ログをディスクにフラッシュする時間が長すぎる場合にデータノードによる操作の処理を制御します。これは、特定の Redo ログのフラッシュが RedoOverCommitLimit 秒より長い時間をかけて RedoOverCommitCounter 回を超える回数分行われ、保留中のトランザクションが中止されたときに発生します。

この発生時は、ノードはここに示す DefaultOperationRedoProblemAction の値に応じた 2 つの方法のいずれかで応答できます。

- ABORT: 中止されたトランザクションに含まれる保留中の操作も中止されます。
- QUEUE: 中止されたトランザクションに含まれる保留中の操作がキューに入れられ、再試行されます。

- DefaultHashMapSize

有効なバージョン	型/単位	デフォルト	範囲/値	再起動タイプ
NDB 7.3.0	LDM スレッド	3840	0 - 3840	N

MySQL Cluster NDB 7.2.7 以降では、前のリリース (240) より大きなテーブルハッシュマップのデフォルトサイズ (3840) が使用されます。MySQL Cluster NDB 7.2.11 以降では、このパラメータを使用して NDB が使用するテーブルハッシュマップのサイズを構成できます (以前は、この値はハードコーディングされていました)。DefaultHashMapSize には、3 つの値 (0、240、3840) のいずれかを指定できます。これらの値とその効果について、次の表で説明します。

値	説明/効果
0	クラスタ内のすべてのデータおよび API ノードの中で、このパラメータに設定されたもっとも小さい値を (あれば) 使用します。どのデータまたは API ノードにも設定されていない場合は、デフォルト値を使用します。
240	MySQL Cluster NDB 7.2.7 より前のすべての MySQL Cluster リリースでデフォルトで使用されていた元のハッシュマップサイズです。
3840	MySQL Cluster NDB 7.2.7 以降でデフォルトで使用される、より大きなハッシュマップサイズ

このパラメータの主な用途は、より大きなハッシュマップサイズ (3840) がデフォルトである MySQL Cluster NDB 7.2.7 以降の MySQL Cluster バージョンと以前のリリース (デフォルトが 240 だった) との間のアップグレードと (特に) ダウングレードを容易にすることです。これは、この変更に下位互換性がない (Bug #14800539) ためです。このパラメータを 240 に設定してから、この値を使用している古いバージョンからのアップグレードを実行すると、クラスタはテーブルハッシュマップに対して引き続き小さいサイズを使用するため、アップグレード後のテーブルでも前のバージョンとの互換性が維持されます。DefaultHashMapSize は個々のデータノード、API ノード、またはその両方で設定できますが、config.ini ファイルの [ndbd default] セクションに一度だけ設定する方法をお勧めします。

このパラメータを増やしたあと、既存のテーブルで新しいサイズを利用するには、そのテーブルに対して ALTER TABLE ... REORGANIZE PARTITION を実行します。その後、そのテーブルでより大きなハッシュマップサイズを使用できるようになります。これは、ローリング再起動の実行に加えて行います。ローリング再起動を実行すると、より大きなハッシュマップを新しいテーブルで使用できるようになりますが、既存のテーブルでは使用できません。

DefaultHashMapSize を 3840 に設定してテーブルを作成または変更したあと、オンラインでこのパラメータを減らす方法は、現在サポートされていません。

- Wan

有効なバージョン	型/単位	デフォルト	範囲/値	再起動タイプ
NDB 7.3.0	ブール	false	true、false	N

WAN の TCP 設定をデフォルトとして使用します。

- ConnectBackoffMaxTime

有効なバージョン	型/単位	デフォルト	範囲/値	再起動タイプ
NDB 7.3.0	整数	0	0 - 4294967039 (0xFFFFFEFF)	N

MySQL Cluster NDB 7.3.7 および MySQL Cluster NDB 7.4.2 以降では、MySQL Cluster 内に起動されていないデータノードが多数ある場合に、このパラメータの値を増やし、クラスタ内のまだ機能し始めていないデータノードへの接続試行を回避して、管理ノードへの大量のトラフィックを抑えることができます。API ノードが新しいデータノードに接続されていない場合は、StartConnectBackoffMaxTime パラメータの値が適用されます。それ以外の場合は、ConnectBackoffMaxTime を使用して接続試行間の待機時間の長さ (ミリ秒) が決定されます。

このパラメータの経過時間を計算する際に、ノードの接続試行中に経過する時間は考慮されません。タイムアウトは、100 ミリ秒の遅延から始まり、約 100 ミリ秒の分解能で適用されます。後続の試行のたびに、ConnectBackoffMaxTime ミリ秒 (最大 100000 ミリ秒 (100 秒)) に達するまでこの期間の長さが倍加されます。

API ノードがデータノードに接続され、そのノードが (ハートビートメッセージで) ほかのデータノードに接続したことを報告すると、データノードに対する接続試行はこのパラメータの影響を受けなくなり、その後接続されるまで 100 ミリ秒間隔で行われます。データノードが起動すると、これが発生したことが API ノードに通知されるまでに HeartbeatIntervalDbApi を要します。

- StartConnectBackoffMaxTime

有効なバージョン	型/単位	デフォルト	範囲/値	再起動タイプ
NDB 7.3.0	整数	1500	0 - 4294967039 (0xFFFFFEFF)	N

MySQL Cluster NDB 7.3.7 および MySQL Cluster NDB 7.4.2 以降では、MySQL Cluster 内に起動されていないデータノードが多数ある場合に、このパラメータの値を増やし、クラスタ内のまだ機能し始めていないデータノードへの接続試行を回避して、管理ノードへの大量のトラフィックを抑えることができます。API ノードが新しいデータノードに接続されていない場合は、StartConnectBackoffMaxTime パラメータの値が適用されます。それ以外の場合は、ConnectBackoffMaxTime を使用して接続試行間の待機時間の長さ (ミリ秒) が決定されます。

このパラメータの経過時間を計算する際に、ノードの接続試行中に経過する時間は考慮されません。タイムアウトは、100 ミリ秒の遅延から始まり、約 100 ミリ秒の分解能で適用されます。後続の試行のたびに、StartConnectBackoffMaxTime ミリ秒 (最大 100000 ミリ秒 (100 秒)) に達するまでこの期間の長さが倍加されます。

API ノードがデータノードに接続され、そのノードが (ハートビートメッセージで) ほかのデータノードに接続したことを報告すると、データノードに対する接続試行はこのパラメータの影響を受けなくなり、その後接続されるまで 100 ミリ秒間隔で行われます。データノードが起動すると、これが発生したことが API ノードに通知されるまでに HeartbeatIntervalDbApi を要します。

ここに示すように、mysql クライアントの SHOW STATUS を使用して、MySQL Cluster SQL ノードとして実行されている MySQL サーバーから情報を取得することもできます。

```
mysql> SHOW STATUS LIKE 'ndb%';
+-----------------------------+--------------+
| Variable_name               | Value        |
+-----------------------------+--------------+
| Ndb_cluster_node_id         | 5            |
| Ndb_config_from_host        | 192.168.0.112 |
| Ndb_config_from_port        | 1186         |
| Ndb_number_of_storage_nodes | 4            |
+-----------------------------+--------------+
4 rows in set (0.02 sec)
```

このステートメントの出力に表示されるステータス変数については、セクション18.3.4.4「MySQL Cluster のステータス変数」を参照してください。

> **注記**
>
> 実行中の MySQL Cluster の構成に新しい SQL または API ノードを追加するには、config.ini ファイル (複数の管理サーバーを使用する場合は複数のファイル) に新しい

> [mysqld] または [api] セクションを追加したあとで、すべてのクラスタノードのローリング再起動を実行する必要があります。これは、新しい SQL または API ノードをクラスタに接続する前に実行する必要があります。
>
> 新しい SQL または API ノードがクラスタ構成内の以前に使用されていない API スロットを使用してクラスタに接続する場合、クラスタの再起動を実行する必要はありません。

18.3.2.8 MySQL クラスタの TCP/IP 接続

TCP/IP は、MySQL Cluster に含まれるノード間のすべての接続で使用されるデフォルトのトランスポートメカニズムです。通常、TCP/IP 接続を定義する必要はありません。MySQL Cluster は、すべてのデータノード、管理ノード、および SQL または API ノードに対してそのような接続を自動的に設定します。

> **注記**
>
> このルールの例外については、セクション18.3.2.9「直接接続を使用する MySQL Cluster の TCP/IP 接続」を参照してください。

デフォルトの接続パラメータをオーバーライドするには、config.ini ファイルで 1 つ以上の [tcp] セクションを使用して接続を定義する必要があります。各 [tcp] セクションは、2 つの MySQL Cluster クラスタノード間の TCP/IP 接続を定義します。これには、最低でも NodeId1 および NodeId2 パラメータと、オーバーライドする接続パラメータを含める必要があります。

これらのパラメータのデフォルト値を [tcp default] セクションに設定して変更することもできます。

> **重要**
>
> config.ini ファイル内の [tcp] セクションは、最後に (ファイル内のほかのすべてのセクションのあとで) 指定するようにしてください。ただし、[tcp default] セクションについては、これは必須ではありません。この要件は、MySQL Cluster 管理サーバーが config.ini ファイルを読み取る方法に関する既知の問題です。

config.ini ファイルの [tcp] および [tcp default] セクションに設定できる接続パラメータをここに示します。

- NodeId1

有効なバージョン	型/単位	デフォルト	範囲/値	再起動タイプ
NDB 7.3.0	数値	[none]	...	N

2 つのノード間の接続を識別するには、構成ファイルの [tcp] セクションに NodeId1 および NodeId2 の値としてノード ID を指定する必要があります。これらは、各ノードに対する一意の Id 値であり、セクション18.3.2.7「MySQL Cluster 内の SQL ノードおよびその他の API ノードの定義」で説明しているものと同じです。

- NodeId2

有効なバージョン	型/単位	デフォルト	範囲/値	再起動タイプ
NDB 7.3.0	数値	[none]	...	N

2 つのノード間の接続を識別するには、構成ファイルの [tcp] セクションに NodeId1 および NodeId2 の値としてノード ID を指定する必要があります。これらは、各ノードに対する一意の Id 値であり、セクション18.3.2.7「MySQL Cluster 内の SQL ノードおよびその他の API ノードの定義」で説明しているものと同じです。

- HostName1

有効なバージョン	型/単位	デフォルト	範囲/値	再起動タイプ
NDB 7.3.0	名前または IP アドレス	[none]	...	N

HostName1 および HostName2 パラメータを使用すると、2 つのノード間の特定の TCP 接続で使用する特定のネットワークインタフェースを指定できます。これらのパラメータに使用する値は、ホスト名または IP アドレスです。

- HostName2

有効なバージョン	型/単位	デフォルト	範囲/値	再起動タイプ
NDB 7.3.0	名前または IP アドレス	[none]	...	N

HostName1 および HostName2 パラメータを使用すると、2 つのノード間の特定の TCP 接続で使用する特定のネットワークインタフェースを指定できます。これらのパラメータに使用する値は、ホスト名または IP アドレスです。

- OverloadLimit

有効なバージョン	型/単位	デフォルト	範囲/値	再起動タイプ
NDB 7.3.0	バイト	0	0 - 4294967039 (0xFFFFFEFF)	N

送信バッファーにこれより多くの未送信バイトがあるときは、接続が過負荷状態であるとみなされます。

このパラメータを使用すると、接続を過負荷状態であるとみなす前に送信バッファーに存在する未送信データ量を決定できます。詳細は、セクション18.3.2.12「MySQL Cluster の送信バッファーパラメータの構成」を参照してください。

- SendBufferMemory

有効なバージョン	型/単位	デフォルト	範囲/値	再起動タイプ
NDB 7.3.0	符号なし	2M	256K - 4294967039 (0xFFFFFEFF)	N

TCP トランスポータは、オペレーティングシステムに対する送信呼び出しを実行する前に、バッファーを使用してすべてのメッセージを格納します。このバッファーが 64K バイトに達すると、その内容が送信されます。これは、一連のメッセージが実行されたときにも送信されます。一時的な過負荷状態に対応するため、より大きな送信バッファーを定義することもできます。

このパラメータが明示的に設定されている場合は、メモリーが各トランスポータ専用でなくなります。代わりに、使用された値によって、(使用可能なメモリーの合計、つまり TotalSendBufferMemory のうち) 単一のトランスポータが使用できるメモリー量に関する厳密な制限が示されます。MySQL Cluster でトランスポータ送信バッファーメモリーの動的割り当てを構成する方法の詳細は、セクション18.3.2.12「MySQL Cluster の送信バッファーパラメータの構成」を参照してください。

送信バッファーのデフォルトサイズは 2M バイトです。これは、ほとんどの状況で推奨されるサイズです。最小サイズは 64K バイトです。理論的な最大は 4G バイトです。

- SendSignalId

有効なバージョン	型/単位	デフォルト	範囲/値	再起動タイプ
NDB 7.3.0	ブール	[テキストを参照]	true、false	N

配信されたメッセージデータグラムを再トレースできるようにするには、各メッセージを識別する必要があります。このパラメータを Y に設定すると、メッセージ ID がネットワーク経由で転送されます。この機能は、製品ビルドではデフォルトで無効になっており、-debug ビルドで有効になります。

- Checksum

有効なバージョン	型/単位	デフォルト	範囲/値	再起動タイプ
NDB 7.3.0	ブール	false	true、false	N

このパラメータはブールパラメータです (Y または 1 に設定すると有効になり、N または 0 に設定すると無効になります)。デフォルトでは無効になっています。有効にすると、送信バッファーに配置される前にすべてのメッセージのチェックサムが計算されます。この機能によって、メッセージが送信バッファーでの待機中に (またはトランスポートメカニズムによって) 破損していないことが確認されます。

- PortNumber (廃止)

  以前は、ほかのノードからの接続を待機するために使用するポート番号をこれで指定していました。このパラメータは今後使用しないようにしてください。代わりに ServerPort データノード構成パラメータを使用します。

- ReceiveBufferMemory

有効なバージョン	型/単位	デフォルト	範囲/値	再起動タイプ
NDB 7.3.0	バイト	2M	16K - 4294967039 (0xFFFFFEFF)	N

  TCP/IP ソケットからデータを受信するときに使用するバッファーのサイズを指定します。

  このパラメータのデフォルト値は 2M バイトです。指定可能な最小値は 16K バイトです。理論的な最大は 4G バイトです。

- TCP_RCV_BUF_SIZE

有効なバージョン	型/単位	デフォルト	範囲/値	再起動タイプ
NDB 7.3.0	符号なし	70080	1 - 2G	N
NDB 7.3.1	符号なし	0	0 - 2G	N

  TCP トランスポータの初期化時に設定される受信バッファーのサイズを指定します。MySQL Cluster NDB 7.3.1 より前では、デフォルトは 70080、最小は 1 でした。MySQL Cluster NDB 7.3.1 以降では、デフォルトおよび最小値は 0 です。その場合は、オペレーティングシステムまたはプラットフォームによってこの値が設定されます。ほとんどの一般的な使用ケースでは、デフォルトが推奨されます。

- TCP_SND_BUF_SIZE

有効なバージョン	型/単位	デフォルト	範囲/値	再起動タイプ
NDB 7.3.0	符号なし	71540	1 - 2G	N
NDB 7.3.1	符号なし	0	0 - 2G	N

  TCP トランスポータの初期化時に設定される送信バッファーのサイズを指定します。MySQL Cluster NDB 7.3.1 より前では、デフォルトは 71540、最小は 1 でした。MySQL Cluster NDB 7.3.1 以降では、デフォルトおよび最小値は 0 です。その場合は、オペレーティングシステムまたはプラットフォームによってこの値が設定されます。ほとんどの一般的な使用ケースでは、デフォルトが推奨されます。

- TCP_MAXSEG_SIZE

有効なバージョン	型/単位	デフォルト	範囲/値	再起動タイプ
NDB 7.3.0	符号なし	0	0 - 2G	N

  TCP トランスポータの初期化時に設定されるメモリーのサイズを決定します。ほとんどの一般的な使用ケースでは、デフォルトが推奨されます。

- TcpBind_INADDR_ANY

  このパラメータを TRUE または 1 に設定すると、IP_ADDR_ANY がバインドされ、任意の場所から接続できるようになります (自動生成接続の場合)。デフォルトは FALSE (0) です。

18.3.2.9 直接接続を使用する MySQL Cluster の TCP/IP 接続

データノード間の直接接続を使用するクラスタをセットアップするには、クラスタの config.ini ファイルの [tcp] セクションに、そのように接続されるデータノードのクロスオーバー IP アドレスを明示的に指定する必要があります。

次の例では、少なくとも 4 台 (管理サーバー、SQL ノード、および 2 つのデータノード用に 1 台ずつ) のホストを含むクラスタについて考えます。クラスタ全体が LAN の 172.23.72.* サブネットに配置されています。次に示すように、通常のネットワーク接続に加えて、2 つのデータノードが標準のクロスオーバーケーブルにより直接接続されており、1.1.0.* アドレス範囲内の IP アドレスを使用して相互に直接通信します。

```
# Management Server
```

```
[ndb_mgmd]
Id=1
HostName=172.23.72.20

# SQL Node
[mysqld]
Id=2
HostName=172.23.72.21

# Data Nodes
[ndbd]
Id=3
HostName=172.23.72.22

[ndbd]
Id=4
HostName=172.23.72.23

# TCP/IP Connections
[tcp]
NodeId1=3
NodeId2=4
HostName1=1.1.0.1
HostName2=1.1.0.2
```

HostName1 および HostName2 パラメータは、直接接続を指定するときにのみ使用されます。

データノード間の直接 TCP 接続を使用すると、データノードがスイッチ、ハブ、ルーターなどの Ethernet デバイスをバイパスできるようになり、クラスタの待機時間が減るため、クラスタ全体の効率が向上する可能性があります。3 つ以上のデータノードでこのような直接接続を最大限に活用するには、各データノードと同じノードグループ内のほかのすべてのデータノート間で直接接続を行う必要があります。

18.3.2.10 MySQL Cluster の共有メモリー接続

MySQL Cluster は、共有メモリートランスポータを使用し、可能な場合は自動的にそれを構成しようとします。config.ini ファイルの [shm] セクションでは、クラスタ内のノード間の共有メモリー接続を明示的に定義します。共有メモリーを接続方法として明示的に定義するときは、少なくとも NodeId1、NodeId2、および ShmKey を定義する必要があります。ほかのすべてのパラメータには、ほとんどのケースで適切に機能するデフォルト値があります。

> **重要**
>
> SHM 機能は、あくまでも実験的なものとみなされます。これは現在のどの MySQL Cluster リリースでも公式にサポートされておらず、テスト結果から、トランスポータに TCP/IP を使用した場合に比べて SHM のパフォーマンスが明らかに優れているとは言えません。
>
> これらの理由により、特定のケースで SHM が正しく機能するようになるかどうかは、ユーザー自身で、または無償のリソース (フォーラム、メーリングリスト) を使用して判定する必要があります。

- NodeId1

有効なバージョン	型/単位	デフォルト	範囲/値	再起動タイプ
NDB 7.3.0	数値	[none]	...	N

2 つのノード間の接続を識別するには、各ノードのノード識別子を NodeId1 および NodeId2 として指定する必要があります。

- NodeId2

有効なバージョン	型/単位	デフォルト	範囲/値	再起動タイプ
NDB 7.3.0	数値	[none]	...	N

2 つのノード間の接続を識別するには、各ノードのノード識別子を NodeId1 および NodeId2 として指定する必要があります。

- HostName1

有効なバージョン	型/単位	デフォルト	範囲/値	再起動タイプ
NDB 7.3.0	名前または IP アドレス	[none]	...	N

HostName1 および HostName2 パラメータを使用すると、2 つのノード間の特定の SHM 接続で使用する特定のネットワークインタフェースを指定できます。これらのパラメータに使用する値は、ホスト名または IP アドレスです。

- HostName2

有効なバージョン	型/単位	デフォルト	範囲/値	再起動タイプ
NDB 7.3.0	名前または IP アドレス	[none]	...	N

HostName1 および HostName2 パラメータを使用すると、2 つのノード間の特定の SHM 接続で使用する特定のネットワークインタフェースを指定できます。これらのパラメータに使用する値は、ホスト名または IP アドレスです。

- OverloadLimit

有効なバージョン	型/単位	デフォルト	範囲/値	再起動タイプ
NDB 7.3.0	バイト	0	0 - 4294967039 (0xFFFFFEFF)	N

送信バッファーにこれより多くの未送信バイトがあるときは、接続が過負荷状態であるとみなされます。

このパラメータを使用すると、接続を過負荷状態であるとみなす前に送信バッファーに存在する未送信データ量を決定できます。詳細は、セクション18.3.2.12「MySQL Cluster の送信バッファーパラメータの構成」を参照してください。

- ShmKey

有効なバージョン	型/単位	デフォルト	範囲/値	再起動タイプ
NDB 7.3.0	符号なし	[none]	0 - 4294967039 (0xFFFFFEFF)	N

共有メモリーセグメントを設定するときは、整数で表されるノード ID を使用して、通信に使用する共有メモリーセグメントを一意に識別します。デフォルト値はありません。

- ShmSize

有効なバージョン	型/単位	デフォルト	範囲/値	再起動タイプ
NDB 7.3.0	バイト	1M	64K - 4294967039 (0xFFFFFEFF)	N

各 SHM 接続には、送信側と読み取り側でノード間のメッセージを配置する共有メモリーセグメントがあります。このセグメントのサイズは ShmSize で定義されます。デフォルト値は 1M バイトです。

- SendSignalId

有効なバージョン	型/単位	デフォルト	範囲/値	再起動タイプ
NDB 7.3.0	ブール	false	true、false	N

配信されたメッセージの経路をたどるには、各メッセージに一意の識別子を設定する必要があります。このパラメータを Y に設定すると、メッセージ ID もネットワーク経由で転送されるようになります。この機能は、製品ビルドではデフォルトで無効になっており、-debug ビルドで有効になります。

- Checksum

有効なバージョン	型/単位	デフォルト	範囲/値	再起動タイプ
NDB 7.3.0	ブール	true	true、false	N

このパラメータはブール (Y/N) パラメータであり、デフォルトで無効になっています。有効にすると、送信バッファーに配置される前にすべてのメッセージのチェックサムが計算されます。

この機能によって、送信バッファーで待機中のメッセージの破損が回避されます。これは、トランスポート時のデータ破損に対するチェックとしても機能します。

- SigNum

有効なバージョン	型/単位	デフォルト	範囲/値	再起動タイプ
NDB 7.3.0	符号なし	[none]	0 - 4294967039 (0xFFFFFEFF)	N

共有メモリートランスポータを使用すると、共有メモリー内に使用可能な新しいデータが発生したときに、プロセスからもう一方のプロセスにオペレーティングシステムシグナルが送信されます。そのシグナルが既存のシグナルと競合する場合は、このパラメータを使用して変更できます。これは、SHM の使用時に、異なるオペレーティングシステムで異なるシグナル番号が使用されるために発生する可能性があります。

SigNum のデフォルト値は 0 です。したがって、共有メモリートランスポータの使用時にクラスタログ内のエラーを回避するために、これを設定する必要があります。通常、このパラメータは config.ini ファイルの [shm default] セクションで 10 に設定します。

18.3.2.11 MySQL Cluster での SCI トランスポート接続

config.ini ファイルの [sci] セクションでは、クラスタノード間の SCI (スケーラブルコヒーラントインタフェース) 接続が明示的に定義されます。MySQL Cluster での SCI トランスポータの使用は、--with-ndb-sci=/your/path/to/SCI を使用して MySQL バイナリがビルドされている場合にのみサポートされます。path には、最低でも SISCI のライブラリとヘッダーファイルを含む lib および include ディレクトリを含むディレクトリを指定するようにしてください。(SCI の詳細は、セクション18.3.5「MySQL Cluster での高速インターコネクトの使用」を参照してください。)

また、SCI には専用のハードウェアが必要です。

ndbd プロセス間の通信にのみ SCI トランスポータを使用することを強くお勧めします。SCI トランスポータを使用すると ndbd プロセスがスリープしないことにも注意してください。このため、ndbd プロセスが少なくとも 2 個の CPU を専用で使用できるマシンでのみ、SCI トランスポータを使用するようにしてください。ndbd プロセスごとに少なくとも 1 つの CPU と、オペレーティングシステムのアクティビティーを処理するために少なくとも 1 つの CPU を予備として残します。

- NodeId1

有効なバージョン	型/単位	デフォルト	範囲/値	再起動タイプ
NDB 7.3.0	数値	[none]	...	N

2 つのノード間の接続を識別するには、各ノードのノード識別子を NodeId1 および NodeId2 として指定する必要があります。

- NodeId2

有効なバージョン	型/単位	デフォルト	範囲/値	再起動タイプ
NDB 7.3.0	数値	[none]	...	N

2 つのノード間の接続を識別するには、各ノードのノード識別子を NodeId1 および NodeId2 として指定する必要があります。

- Host1SciId0

有効なバージョン	型/単位	デフォルト	範囲/値	再起動タイプ
NDB 7.3.0	符号なし	[none]	0 - 4294967039 (0xFFFFFEFF)	N

これは、(NodeId1 によって識別される) 1 つ目のクラスタノード上の SCI ノード ID を識別します。

- Host1SciId1

有効なバージョン	型/単位	デフォルト	範囲/値	再起動タイプ
NDB 7.3.0	符号なし	0	0 - 4294967039 (0xFFFFFEFF)	N

ノード間で別個のネットワークを使用する 2 つの SCI カード間のフェイルオーバー用に SCI トランスポータを設定できます。これは、1 つ目のノードで使用されるノード ID と 2 つ目の SCI カードを識別します。

- Host2SciId0

有効なバージョン	型/単位	デフォルト	範囲/値	再起動タイプ
NDB 7.3.0	符号なし	[none]	0 - 4294967039 (0xFFFFFEFF)	N

これは、(NodeId2 によって識別される) 2 つ目のクラスタノード上の SCI ノード ID を識別します。

- Host2SciId1

有効なバージョン	型/単位	デフォルト	範囲/値	再起動タイプ
NDB 7.3.0	符号なし	0	0 - 4294967039 (0xFFFFFEFF)	N

2 つの SCI カードを使用してフェイルオーバーを提供するときに、このパラメータは 2 つ目のノードで使用される 2 つ目の SCI カードを識別します。

- HostName1

有効なバージョン	型/単位	デフォルト	範囲/値	再起動タイプ
NDB 7.3.0	名前または IP アドレス	[none]	...	N

HostName1 および HostName2 パラメータを使用すると、2 つのノード間の特定の SCI 接続で使用する特定のネットワークインタフェースを指定できます。これらのパラメータに使用する値は、ホスト名または IP アドレスです。

- HostName2

有効なバージョン	型/単位	デフォルト	範囲/値	再起動タイプ
NDB 7.3.0	名前または IP アドレス	[none]	...	N

HostName1 および HostName2 パラメータを使用すると、2 つのノード間の特定の SCI 接続で使用する特定のネットワークインタフェースを指定できます。これらのパラメータに使用する値は、ホスト名または IP アドレスです。

- SharedBufferSize

有効なバージョン	型/単位	デフォルト	範囲/値	再起動タイプ
NDB 7.3.0	符号なし	10M	64K - 4294967039 (0xFFFFFEFF)	N

各 SCI トランスポートには、2 つのノード間の通信に使用される共有メモリーセグメントがあります。ほとんどのアプリケーションでは、このセグメントのサイズをデフォルト値の 1M バイトに設定すれば十分です。より小さい値を使用すると、多数の並列挿入を実行するときに問題が発生する可能性があります。共有バッファーが小さすぎる場合は、これによって ndbd プロセスがクラッシュする可能性もあります。

- SendLimit

有効なバージョン	型/単位	デフォルト	範囲/値	再起動タイプ
NDB 7.3.0	符号なし	8K	128 - 32K	N

SCI ネットワークで転送されるメッセージは、その前に SCI メディアの直前の小さなバッファーに保存されます。デフォルトでは、これは 8K バイトに設定されます。弊社のベンチマークによれば、64K バイトでパフォーマンスが最高になりますが、16K バイトではこの数パーセント以内であり、8K バイトを超える値に増やしても利点は (あるとしても) ほとんどありませんでした。

- SendSignalId

有効なバージョン	型/単位	デフォルト	範囲/値	再起動タイプ
NDB 7.3.0	ブール	true	true、false	N

配信されたメッセージをトレースするには、各メッセージを一意に識別する必要があります。このパラメータを Y に設定すると、メッセージ ID がネットワーク経由で転送されます。この機能は、製品ビルドではデフォルトで無効になっており、-debug ビルドで有効になります。

- Checksum

有効なバージョン	型/単位	デフォルト	範囲/値	再起動タイプ
NDB 7.3.0	ブール	false	true、false	N

このパラメータはブール値であり、デフォルトで無効になっています。Checksum を有効にすると、送信バッファーに配置される前にすべてのメッセージのチェックサムが計算されます。この機能によって、送信バッファーで待機中のメッセージの破損が回避されます。これは、トランスポート時のデータ破損に対するチェックとしても機能します。

- OverloadLimit

有効なバージョン	型/単位	デフォルト	範囲/値	再起動タイプ
NDB 7.3.0	バイト	0	0 - 4294967039 (0xFFFFFEFF)	N

送信バッファーにこれより多くの未送信バイトがあるときは、接続が過負荷状態であるとみなされます。詳細は、セクション18.3.2.12「MySQL Cluster の送信バッファーパラメータの構成」を参照してください。

18.3.2.12 MySQL Cluster の送信バッファーパラメータの構成

以前の NDB カーネルでは、クラスタ内の各ノードに対してサイズが 2M バイトに固定された送信バッファーが採用され、このバッファーはノードの起動時に割り当てられていました。このバッファーのサイズは、クラスタの起動後に変更できなかったため、事前にトランスポータソケットに対する最大負荷に対応できる十分な大きさにする必要がありました。しかし、多くの場合ほとんどのメモリーが使用されないため、これは非効率的なメモリーの使用方法であり、その結果多数の API ノードに対応して拡張したときに、大量のリソースが浪費される可能性がありました。

この問題は、最終的には (MySQL Cluster NDB 7.0 で) すべてのトランスポータが共有するプールからメモリーを動的に割り当てる統合された送信バッファーを採用することで解決されました。これにより、送信バッファーのサイズを必要に応じて調整できます。統合された送信バッファーの構成は、次のパラメータを設定すると実現できます。

- TotalSendBufferMemory　このパラメータは、すべてのタイプの MySQL Cluster ノードに対して設定できます。つまり、config.ini ファイルの [ndbd]、[mgm]、および [api] (または [mysql]) セクションに設定できます。これは、各ノードによって割り当てられ、すべての構成済みトランスポータ間での使用に設定されるメモリーの合計量 (バイト単位) を表します。設定する場合、その最小は 256K バイト、最大は 4294967039 です。

  既存の構成との下位互換性を確保するため、このパラメータのデフォルト値は、すべての構成済みトランスポータの最大送信バッファーサイズの合計にトランスポータあたり 32K バイト (1 ページ) を加えた値になっています。最大は、次の表に示すように、トランスポータのタイプによって異なります。

トランスポータ	最大送信バッファーサイズ (バイト)
TCP	SendBufferMemory (デフォルト = 2M)

トランスポータ	最大送信バッファーサイズ (バイト)
SCI	SendLimit (デフォルト = 8K) + 16K
SHM	20K

これにより、既存の構成が MySQL Cluster NDB 6.3 以前とほぼ同じように機能し、各トランスポータで同じ量のメモリーと送信バッファースペースを使用できます。ただし、特定のトランスポータで使用されていないメモリーは、ほかのトランスポータでも使用できません。

- OverloadLimit　このパラメータは、config.ini ファイルの [tcp] セクションで使用され、接続が過負荷状態であるとみなされる前に送信バッファーに存在する未送信データの量 (バイト単位) を表します。このような過負荷状態が発生すると、過負荷の接続に影響を与えるトランザクションが失敗し、過負荷のステータスが終了するまで次のエラーが発生します: NDB API エラー 1218 (Send Buffers overloaded in NDB kernel)。デフォルト値は 0 です。その場合、特定の接続に対する実質的な過負荷の制限は SendBufferMemory * 0.8 として計算されます。このパラメータの最大値は 4G です。
- SendBufferMemory　この値は、単一のトランスポータが TotalSendBufferMemory で指定されたプール全体から使用できるメモリー量に対する厳密な制限を表します。ただし、すべての構成済みトランスポータの SendBufferMemory の合計は、特定のノードに対して設定された TotalSendBufferMemory より大きくなることがあります。多数のノードが使用されているときは、すべてのトランスポータがメモリーの最大量を同時に必要としないかぎり、このような方法でメモリーが節約されます。

18.3.3 MySQL Cluster 構成パラメータの概要

次の 4 つのセクションでは、クラスタの機能を管理するために config.ini ファイルで使用される MySQL Cluster 構成パラメータのサマリー表を示します。各表は、クラスタノードプロセスのいずれかのタイプ (ndbd、ndb_mgmd、および mysqld) に対応するパラメータのリストであり、パラメータのタイプと、場合に応じてそのデフォルト、最小、および最大値を含んでいます。

これらの表は、特定の構成パラメータの値を変更するために必要な再起動のタイプ (ノードの再起動またはシステムの再起動) と、--initial を指定して再起動を実行する必要があるかどうかも示しています。

ノードの再起動またはノードの初期再起動を実行するときは、クラスタのすべてのデータノードを順に再起動する必要があります (ローリング再起動とも呼ばれます)。node としてマークされたクラスタ構成パラメータは、オンラインで (つまり、この方法でクラスタをシャットダウンすることなく) 更新できます。ノードの初期再起動では、--initial オプションを指定して各 ndbd プロセスを再起動する必要があります。

システムの再起動では、クラスタ全体を完全にシャットダウンして再起動する必要があります。システムの初期再起動では、クラスタのバックアップを取り、シャットダウン後にクラスタファイルシステムを消去して、再起動後にバックアップからリストアする必要があります。

どのクラスタ再起動でも、クラスタのすべての管理サーバーを再起動して、更新された構成パラメータ値を読み取ることができるようにする必要があります。

> **重要**
>
> 数値型のクラスタパラメータの値は通常問題なく増やすことができますが、そのような調整は比較的小さいインクリメントで徐々に行うことをお勧めします。多くのパラメータは、ローリング再起動を使用してオンラインで増やすことができます。
>
> ただし、このようなパラメータ値の削減は、ノードの再起動、ノードの初期再起動、またはクラスタの完全なシステムの再起動のどれを使用するかに関係なく、安易に行うべきではありません。その場合は、事前に入念な計画とテストを行うことをお勧めします。これは、メモリー使用やディスクスペースに関連するパラメータ (MaxNoOfTables、MaxNoOfOrderedIndexes、MaxNoOfUniqueHashIndexes など) に関して特に当てはまります。さらに、一般的には、メモリーおよびディスク使用に関連する構成パラメータは単純なノードの再起動により値を増やすことができますが、値を減らすにはノードの初期再起動が必要になります。

これらのパラメータの一部は、複数のタイプのクラスタノードの構成に使用できるため、複数の表に表示されている場合があります。

> **注記**
>
> これらの表では、多くの場合、最大値として 4294967039 が表示されています。この値は、NDBCLUSTER のソースで MAX_INT_RNIL として定義されており、0xFFFFFEFF、または $2^{32} - 2^8 - 1$ と等価です。

18.3.3.1 MySQL Cluster データノードの構成パラメータ

このセクションのサマリー表では、MySQL Cluster データノードを構成するために config.ini ファイルの [ndbd] または [ndbd default] セクションで使用されるパラメータについて説明します。これらの各パラメータに関する詳しい説明およびその他の追加情報については、セクション18.3.2.6「MySQL Cluster データノードの定義」を参照してください。

これらのパラメータは、ndbd のマルチスレッドバージョンである ndbmtd にも適用されます。詳細は、セクション18.4.3「ndbmtd — MySQL Cluster データノードデーモン (マルチスレッド)」を参照してください。

再起動のタイプ　MySQL Cluster 構成パラメータの変更は、クラスタが再起動されるまで有効になりません。サマリー表では、特定のパラメータの変更に必要な再起動のタイプを次のように示します。

- N - ノードの再起動: このパラメータは、ローリング再起動を使用して更新できます (セクション18.5.5「MySQL Cluster のローリング再起動の実行」を参照してください)。
- S - システムの再起動: このパラメータの変更を有効にするには、クラスタを完全にシャットダウンしてから再起動する必要があります。
- I - 初期再起動: --initial オプションを使用してデータノードを再起動する必要があります。

再起動のタイプの詳細は、セクション18.3.3「MySQL Cluster 構成パラメータの概要」を参照してください。

MySQL Cluster NDB 7.3 以降では、新しいデータノードグループを実行中のクラスタにオンラインで追加する方法がサポートされます。詳細は、セクション18.5.13「MySQL Cluster データノードのオンライン追加」を参照してください。

表 18.1 データノード構成パラメータ

パラメータ名	型または単位 デフォルト値 最小値/最大値または許可される値	再起動タイプ	バージョン ... (以降)
Arbitration	列挙 デフォルト デフォルト、無効、WaitExternal	N	NDB 7.3.0
ArbitrationTimeout	ミリ秒 7500 10 / 4294967039 (0xFFFFFEFF)	N	NDB 7.3.0
BackupDataBufferSize	バイト 16M 0 / 4294967039 (0xFFFFFEFF)	N	NDB 7.3.0
BackupDataDir	パス FileSystemPath ...	IN	NDB 7.3.0
BackupLogBufferSize	バイト 16M 0 / 4294967039 (0xFFFFFEFF)	N	NDB 7.3.0
BackupMaxWriteSize	バイト 1M 2K / 4294967039 (0xFFFFFEFF)	N	NDB 7.3.0
BackupMemory	バイト	N	NDB 7.3.0

パラメータ名	型または単位 デフォルト値 最小値/最大値または許可される値	再起動タイプ	バージョン ... (以降)
	32M 0 / 4294967039 (0xFFFFFEFF)		
BackupReportFrequency	秒 0 0 / 4294967039 (0xFFFFFEFF)	N	NDB 7.3.0
BackupWriteSize	バイト 256K 2K / 4294967039 (0xFFFFFEFF)	N	NDB 7.3.0
BatchSizePerLocalScan	整数 256 1 / 992	N	NDB 7.3.0
BuildIndexThreads	数値 0 0 / 128	S	NDB 7.3.0
CompressedBackup	ブール false true、false	N	NDB 7.3.0
CompressedLCP	ブール false true、false	N	NDB 7.3.0
ConnectCheckIntervalDelay	文字列 0 0 / 4294967039 (0xFFFFFEFF)	N	NDB 7.3.0
CrashOnCorruptedTuple	ブール true true、false	S	NDB 7.3.0
DataDir	パス	IN	NDB 7.3.0
DataMemory	バイト 80M 1M / 1024G	N	NDB 7.3.0
DefaultHashMapSize	LDM スレッド 3840 0 / 3840	N	NDB 7.3.0
DictTrace	バイト 未定義 0 / 100	N	NDB 7.3.0

パラメータ名	型または単位 デフォルト値 最小値/最大値または許可される値	再起動タイプ	バージョン ... (以降)
DiskCheckpointSpeed	バイト 10M 1M / 4294967039 (0xFFFFFEFF)	N	NDB 7.3.0
DiskCheckpointSpeedInRestart	バイト 100M 1M / 4294967039 (0xFFFFFEFF)	N	NDB 7.3.0
DiskIOThreadPool	スレッド 2 0 / 4294967039 (0xFFFFFEFF)	N	NDB 7.3.0
Diskless	true|false (1|0) false true、false	IS	NDB 7.3.0
DiskPageBufferMemory	バイト 64M 4M / 1T	N	NDB 7.3.0
DiskSyncSize	バイト 4M 32K / 4294967039 (0xFFFFFEFF)	N	NDB 7.3.0
ExecuteOnComputer	name [none] ...	S	NDB 7.3.0
ExtraSendBufferMemory	バイト 0 0 / 32G	N	NDB 7.3.0
FileSystemPath	パス DataDir ...	IN	NDB 7.3.0
FileSystemPathDataFiles	ファイル名 [テキストを参照] ...	IN	NDB 7.3.0
FileSystemPathDD	ファイル名 FileSystemPath ...	IN	NDB 7.3.0
FileSystemPathUndoFiles	ファイル名 [テキストを参照] ...	IN	NDB 7.3.0

パラメータ名	型または単位 デフォルト値 最小値/最大値または許可される値	再起動タイプ	バージョン ... (以降)
FragmentLogFileSize	バイト 16M 4M / 1G	IN	NDB 7.3.0
HeartbeatIntervalDbApi	ミリ秒 1500 100 / 4294967039 (0xFFFFFEFF)	N	NDB 7.3.0
HeartbeatIntervalDbDb	ミリ秒 5000 10 / 4294967039 (0xFFFFFEFF)	N	NDB 7.3.0
HeartbeatOrder	数値 0 0 / 65535	S	NDB 7.3.0
HostName	名前または IP アドレス localhost ...	N	NDB 7.3.0
Id	符号なし [none] 1 / 48	IS	NDB 7.3.0
IndexMemory	バイト 18M 1M / 1T	N	NDB 7.3.0
IndexStatAutoCreate	ブール false false、true	S	NDB 7.3.0
IndexStatAutoUpdate	ブール false false、true	S	NDB 7.3.0
IndexStatSaveScale	パーセンテージ 100 0 / 4294967039 (0xFFFFFEFF)	IN	NDB 7.3.0
IndexStatSaveSize	バイト 32768 0 / 4294967039 (0xFFFFFEFF)	IN	NDB 7.3.0
IndexStatTriggerPct	パーセンテージ 100 0 / 4294967039 (0xFFFFFEFF)	IN	NDB 7.3.0

パラメータ名	型または単位 デフォルト値 最小値/最大値または許可される値	再起動タイプ	バージョン ... (以降)
IndexStatTriggerScale	パーセンテージ 100 0 / 4294967039 (0xFFFFFEFF)	IN	NDB 7.3.0
IndexStatUpdateDelay	秒 60 0 / 4294967039 (0xFFFFFEFF)	IN	NDB 7.3.0
InitFragmentLogFiles	[値を参照] SPARSE SPARSE、FULL	IN	NDB 7.3.0
InitialLogFileGroup	文字列 [テキストを参照] ...	S	NDB 7.3.0
InitialNoOfOpenFiles	ファイル 27 20 / 4294967039 (0xFFFFFEFF)	N	NDB 7.3.0
InitialTablespace	文字列 [テキストを参照] ...	S	NDB 7.3.0
LateAlloc	数値 1 0 / 1	N	NDB 7.3.0
LcpScanProgressTimeout	second 60 0 / 4294967039 (0xFFFFFEFF)	N	NDB 7.3.3
LockExecuteThreadToCPU	CPU ID 64K 0 / 64K	N	NDB 7.3.0
LockMaintThreadsToCPU	CPU ID [none] 0 / 64K	N	NDB 7.3.0
LockPagesInMainMemory	数値 0 0 / 2	N	NDB 7.3.0
LogLevelCheckpoint	ログレベル 0 0 / 15	N	NDB 7.3.0
LogLevelCongestion	levelr 0	N	NDB 7.3.0

パラメータ名	型または単位 デフォルト値 最小値/最大値または許可される値	再起動タイプ	バージョン ... (以降)
	0 / 15		
LogLevelConnection	整数 0 0 / 15	N	NDB 7.3.0
LogLevelError	整数 0 0 / 15	N	NDB 7.3.0
LogLevelInfo	整数 0 0 / 15	N	NDB 7.3.0
LogLevelNodeRestart	整数 0 0 / 15	N	NDB 7.3.0
LogLevelShutdown	整数 0 0 / 15	N	NDB 7.3.0
LogLevelStartup	整数 1 0 / 15	N	NDB 7.3.0
LogLevelStatistic	整数 0 0 / 15	N	NDB 7.3.0
LongMessageBuffer	バイト 64M 512K / 4294967039 (0xFFFFFEFF)	N	NDB 7.3.5
MaxAllocate	符号なし 32M 1M / 1G	N	NDB 7.3.0
MaxBufferedEpochs	エポック 100 0 / 100000	N	NDB 7.3.0
MaxBufferedEpochBytes	バイト 26214400 26214400 (0x01900000) / 4294967039 (0xFFFFFEFF)	N	NDB 7.3.0
MaxDiskWriteSpeed	数値 20M 1M / 1024G	S	NDB 7.4.1

パラメータ名	型または単位 デフォルト値 最小値/最大値または許可される値	再起動タイプ	バージョン ... (以降)
MaxDiskWriteSpeedOtherNodeRestart	数値 50M 1M / 1024G	S	NDB 7.4.1
MaxDiskWriteSpeedOwnRestart	数値 200M 1M / 1024G	S	NDB 7.4.1
MaxDMLOperationsPerTransaction	操作 (DML) 4294967295 32 / 4294967295	N	NDB 7.3.0
MaxLCPStartDelay	秒 0 0 / 600	N	NDB 7.3.0
MaxNoOfAttributes	整数 1000 32 / 4294967039 (0xFFFFFEFF)	N	NDB 7.3.0
MaxNoOfConcurrentIndexOperations	整数 8K 0 / 4294967039 (0xFFFFFEFF)	N	NDB 7.3.0
MaxNoOfConcurrentOperations	整数 32K 32 / 4294967039 (0xFFFFFEFF)	N	NDB 7.3.0
MaxNoOfConcurrentScans	整数 256 2 / 500	N	NDB 7.3.0
MaxNoOfConcurrentSubOperations	符号なし 256 0 / 4294967039 (0xFFFFFEFF)	N	NDB 7.3.0
MaxNoOfConcurrentTransactions	整数 4096 32 / 4294967039 (0xFFFFFEFF)	N	NDB 7.3.0
MaxNoOfFiredTriggers	整数 4000 0 / 4294967039 (0xFFFFFEFF)	N	NDB 7.3.0
MaxNoOfLocalOperations	整数 UNDEFINED 32 / 4294967039 (0xFFFFFEFF)	N	NDB 7.3.0

パラメータ名	型または単位 デフォルト値 最小値/最大値または許可される値	再起動タイプ	バージョン ... (以降)
MaxNoOfLocalScans	整数 [テキストを参照] 32 / 4294967039 (0xFFFFFEFF)	N	NDB 7.3.0
MaxNoOfOpenFiles	符号なし 0 20 / 4294967039 (0xFFFFFEFF)	N	NDB 7.3.0
MaxNoOfOrderedIndexes	整数 128 0 / 4294967039 (0xFFFFFEFF)	N	NDB 7.3.0
MaxNoOfSavedMessages	整数 25 0 / 4294967039 (0xFFFFFEFF)	N	NDB 7.3.0
MaxNoOfSubscribers	符号なし 0 0 / 4294967039 (0xFFFFFEFF)	N	NDB 7.3.0
MaxNoOfSubscriptions	符号なし 0 0 / 4294967039 (0xFFFFFEFF)	N	NDB 7.3.0
MaxNoOfTables	整数 128 8 / 20320	N	NDB 7.3.0
MaxNoOfTriggers	整数 768 0 / 4294967039 (0xFFFFFEFF)	N	NDB 7.3.0
MaxNoOfUniqueHashIndexes	整数 64 0 / 4294967039 (0xFFFFFEFF)	N	NDB 7.3.0
MaxParallelScansPerFragment	バイト 256 1 / 4294967039 (0xFFFFFEFF)	N	NDB 7.3.0
MaxStartFailRetries	符号なし 3 0 / 4294967039 (0xFFFFFEFF)	N	NDB 7.3.0
MemReportFrequency	符号なし	N	NDB 7.3.0

パラメータ名	型または単位 デフォルト値 最小値/最大値または許可される値	再起動タイプ	バージョン ... (以降)
	0 0 / 4294967039 (0xFFFFFEFF)		
MinDiskWriteSpeed	数値 10M 1M / 1024G	S	NDB 7.4.1
MinFreePct	符号なし 5 0 / 100	N	NDB 7.3.0
NodeGroup	 [none] 0 / 65536	IS	NDB 7.3.0
NodeId	符号なし [none] 1 / 48	IS	NDB 7.3.0
NoOfFragmentLogFiles	整数 16 3 / 4294967039 (0xFFFFFEFF)	IN	NDB 7.3.0
NoOfReplicas	整数 2 1 / 4	IS	NDB 7.3.0
Numa	ブール 1 ...	N	NDB 7.3.0
ODirect	ブール false true、false	N	NDB 7.3.0
RealtimeScheduler	ブール false true、false	N	NDB 7.3.0
RedoBuffer	バイト 32M 1M / 4294967039 (0xFFFFFEFF)	N	NDB 7.3.0
RedoOverCommitCounter	数値 3 0 / 4294967039 (0xFFFFFEFF)	N	NDB 7.3.0
RedoOverCommitLimit	秒 20	N	NDB 7.3.0

パラメータ名	型または単位 デフォルト値 最小値/最大値または許可される値	再起動タイプ	バージョン ... (以降)
	0 / 4294967039 (0xFFFFFEFF)		
ReservedSendBufferMemory	バイト 256K 0 / 4294967039 (0xFFFFFEFF)	N	NDB 7.3.0
RestartOnErrorInsert	エラーコード 2 0 / 4	N	NDB 7.3.0
SchedulerExecutionTimer	マイクロ秒 50 0 / 11000	N	NDB 7.3.0
SchedulerSpinTimer	マイクロ秒 0 0 / 500	N	NDB 7.3.0
ServerPort	符号なし [none] 1 / 64K	N	NDB 7.3.0
SharedGlobalMemory	バイト 128M 0 / 64T	N	NDB 7.3.0
StartFailRetryDelay	符号なし 0 0 / 4294967039 (0xFFFFFEFF)	N	NDB 7.3.0
StartFailureTimeout	ミリ秒 0 0 / 4294967039 (0xFFFFFEFF)	N	NDB 7.3.0
StartNoNodeGroupTimeout	ミリ秒 15000 0 / 4294967039 (0xFFFFFEFF)	N	NDB 7.3.0
StartPartialTimeout	ミリ秒 30000 0 / 4294967039 (0xFFFFFEFF)	N	NDB 7.3.0
StartPartitionedTimeout	ミリ秒 60000 0 / 4294967039 (0xFFFFFEFF)	N	NDB 7.3.0
StartupStatusReportFrequency	秒 0	N	NDB 7.3.0

パラメータ名	型または単位 デフォルト値 最小値/最大値または許可される値	再起動タイプ	バージョン ... (以降)
	0 / 4294967039 (0xFFFFFEFF)		
StopOnError	ブール 1 0、1	N	NDB 7.3.0
StringMemory	% またはバイト 25 0 / 4294967039 (0xFFFFFEFF)	S	NDB 7.3.0
TcpBind_INADDR_ANY	ブール false true、false	N	NDB 7.3.0
TimeBetweenEpochs	ミリ秒 100 0 / 32000	N	NDB 7.3.0
TimeBetweenEpochsTimeout	ミリ秒 0 0 / 256000	N	NDB 7.3.0
TimeBetweenGlobalCheckpoints	ミリ秒 2000 20 / 32000	N	NDB 7.3.0
TimeBetweenInactiveTransactionAbortCheck	ミリ秒 1000 1000 / 4294967039 (0xFFFFFEFF)	N	NDB 7.3.0
TimeBetweenLocalCheckpoints	4 バイトワードの数 (底 2 の対数として) 20 0 / 31	N	NDB 7.3.0
TimeBetweenWatchDogCheck	ミリ秒 6000 70 / 4294967039 (0xFFFFFEFF)	N	NDB 7.3.0
TimeBetweenWatchDogCheckInitial	ミリ秒 6000 70 / 4294967039 (0xFFFFFEFF)	N	NDB 7.3.0
TotalSendBufferMemory	バイト 256K 0 / 4294967039 (0xFFFFFEFF)	N	NDB 7.3.0

パラメータ名	型または単位 デフォルト値 最小値/最大値または許可される値	再起動タイプ	バージョン ... (以降)
TransactionBufferMemory	バイト 1M 1K / 4294967039 (0xFFFFFEFF)	N	NDB 7.3.0
TransactionDeadlockDetectionTimeout	ミリ秒 1200 50 / 4294967039 (0xFFFFFEFF)	N	NDB 7.3.0
TransactionInactiveTimeout	ミリ秒 [テキストを参照] 0 / 4294967039 (0xFFFFFEFF)	N	NDB 7.3.0
TwoPassInitialNodeRestartCopy	ブール false true、false	N	NDB 7.3.0
UndoDataBuffer	符号なし 16M 1M / 4294967039 (0xFFFFFEFF)	N	NDB 7.3.0
UndoIndexBuffer	符号なし 2M 1M / 4294967039 (0xFFFFFEFF)	N	NDB 7.3.0

表 18.2 マルチスレッドデータノード構成パラメータ

パラメータ名	型または単位 デフォルト値 最小値/最大値または許可される値	再起動タイプ	バージョン ... (以降)
MaxNoOfExecutionThreads	整数 2 2 / 72	IS	NDB 7.3.3
NoOfFragmentLogParts	数値 4 4、8、12、16、24、32	IN	NDB 7.3.3
ThreadConfig	文字列 '' ...	IS	NDB 7.3.0

18.3.3.2 MySQL Cluster 管理ノードの構成パラメータ

このセクションのサマリー表では、MySQL Cluster 管理ノードを構成するために config.ini ファイルの [ndb_mgmd] または [mgm] セクションで使用するパラメータについて説明します。各パラメータ関する詳しい説

明およびその他の追加情報については、セクション18.3.2.5「MySQL Cluster 管理サーバーの定義」を参照してください。

再起動のタイプ　MySQL Cluster 構成パラメータの変更は、クラスタが再起動されるまで有効になりません。サマリー表では、特定のパラメータの変更に必要な再起動のタイプを次のように示します。

- N - ノードの再起動: このパラメータは、ローリング再起動を使用して更新できます (セクション 18.5.5「MySQL Cluster のローリング再起動の実行」を参照してください)。
- S - システムの再起動: このパラメータの変更を有効にするには、クラスタを完全にシャットダウンしてから再起動する必要があります。
- I - 初期再起動: --initial オプションを使用してデータノードを再起動する必要があります。

再起動のタイプの詳細は、セクション18.3.3「MySQL Cluster 構成パラメータの概要」を参照してください。

表 18.3 管理ノード構成パラメータ

パラメータ名	型または単位 デフォルト値 最小値/最大値または許可される値	再起動タイプ	バージョン ... (以降)
ArbitrationDelay	ミリ秒 0 0 / 4294967039 (0xFFFFFEFF)	N	NDB 7.3.0
ArbitrationRank	0-2 1 0 / 2	N	NDB 7.3.0
DataDir	パス	N	NDB 7.3.0
ExecuteOnComputer	name [none] ...	S	NDB 7.3.0
HeartbeatIntervalMgmdMgmd	ミリ秒 1500 100 / 4294967039 (0xFFFFFEFF)	N	NDB 7.3.3
HeartbeatThreadPriority	文字列 [none] ...	S	NDB 7.3.0
HostName	名前または IP アドレス [none] ...	N	NDB 7.3.0
Id	符号なし [none] 1 / 255	IS	NDB 7.3.0
LogDestination	{CONSOLE|SYSLOG|FILE} [テキストを参照]	N	NDB 7.3.0

パラメータ名	型または単位 デフォルト値 最小値/最大値または許可される値	再起動タイプ	バージョン ... (以降)
	...		
MaxNoOfSavedEvents	符号なし 100 0 / 4294967039 (0xFFFFFEFF)	N	NDB 7.3.0
NodeId	符号なし [none] 1 / 255	IS	NDB 7.3.0
PortNumber	符号なし 1186 0 / 64K	N	NDB 7.3.0
PortNumberStats	符号なし [none] 0 / 64K	N	NDB 7.3.0
TotalSendBufferMemory	バイト 256K 0 / 4294967039 (0xFFFFFEFF)	N	NDB 7.3.0
wan	ブール false true、false	N	NDB 7.3.0

> 注記
>
> 管理ノードの構成に変更を加えたあとは、新しい構成を有効にするために、クラスタのローリング再起動を実行する必要があります。詳細は、セクション18.3.2.5「MySQL Cluster 管理サーバーの定義」を参照してください。
>
> 実行中の MySQL Cluster に新しい管理サーバーを追加するには、既存の config.ini ファイルを変更したあとで、すべてのクラスタノードのローリング再起動も実行する必要があります。複数の管理ノードを使用するときに発生する問題の詳細は、セクション18.1.6.10「複数の MySQL Cluster ノードに関する制限」を参照してください。

18.3.3.3 MySQL Cluster SQL ノードおよび API ノードの構成パラメータ

このセクションのサマリー表では、MySQL Cluster SQL ノードおよび API ノードを構成するために config.ini ファイルの [mysqld] および [api] セクションで使用するパラメータについて説明します。各パラメータに関する詳しい説明およびその他の追加情報については、セクション18.3.2.7「MySQL Cluster 内の SQL ノードおよびその他の API ノードの定義」を参照してください。

> 注記
>
> MySQL Cluster 用の MySQL サーバーオプションについては、セクション18.3.4.2「MySQL Cluster 用の MySQL Server オプション」を参照してください。MySQL Cluster に関連する MySQL サーバーシステム変数については、セクション18.3.4.3「MySQL Cluster のシステム変数」を参照してください。

再起動のタイプ　MySQL Cluster 構成パラメータの変更は、クラスタが再起動されるまで有効になりません。サマリー表では、特定のパラメータの変更に必要な再起動のタイプを次のように示します。

- N - ノードの再起動: このパラメータは、ローリング再起動を使用して更新できます (セクション18.5.5「MySQL Cluster のローリング再起動の実行」を参照してください)。

- S - システムの再起動: このパラメータの変更を有効にするには、クラスタを完全にシャットダウンしてから再起動する必要があります。
- I - 初期再起動: --initial オプションを使用してデータノードを再起動する必要があります。

再起動のタイプの詳細は、セクション18.3.3「MySQL Cluster 構成パラメータの概要」を参照してください。

表 18.4 SQL ノード/API ノード構成パラメータ

パラメータ名	型または単位 デフォルト値 最小値/最大値または許可される値	再起動タイプ	バージョン ... (以降)
ArbitrationDelay	ミリ秒 0 0 / 4294967039 (0xFFFFFEFF)	N	NDB 7.3.0
ArbitrationRank	0-2 0 0 / 2	N	NDB 7.3.0
AutoReconnect	ブール false true、false	N	NDB 7.3.0
BatchByteSize	バイト 16K 1024 / 1M	N	NDB 7.3.0
BatchSize	レコード 256 1 / 992	N	NDB 7.3.0
ConnectBackoffMaxTime	整数 0 0 / 4294967039 (0xFFFFFEFF)	N	NDB 7.3.0
ConnectionMap	文字列 [none] ...	N	NDB 7.3.0
DefaultHashMapSize	LDM スレッド 3840 0 / 3840	N	NDB 7.3.0
DefaultOperationRedoProblemAction	列挙 QUEUE ABORT、QUEUE	S	NDB 7.3.0
EventLogBufferSize	バイト 8192 0 / 64K	S	NDB 7.3.0
ExecuteOnComputer	name [none] ...	S	NDB 7.3.0
ExtraSendBufferMemory	バイト	N	NDB 7.3.0

パラメータ名	型または単位 デフォルト値 最小値/最大値または許可される値	再起動タイプ	バージョン ... (以降)
	0 0 / 4294967039 (0xFFFFFEFF)		
HeartbeatThreadPriority	文字列 [none] ...	S	NDB 7.3.0
HostName	名前または IP アドレス [none] ...	N	NDB 7.3.0
Id	符号なし [none] 1 / 255	IS	NDB 7.3.0
MaxScanBatchSize	バイト 256K 32K / 16M	N	NDB 7.3.0
NodeId	符号なし [none] 1 / 255	IS	NDB 7.3.0
StartConnectBackoffMaxTime	整数 1500 0 / 4294967039 (0xFFFFFEFF)	N	NDB 7.3.0
TotalSendBufferMemory	バイト 256K 0 / 4294967039 (0xFFFFFEFF)	N	NDB 7.3.0
wan	ブール false true、false	N	NDB 7.3.0

> 注記
>
> 実行中の MySQL Cluster の構成に新しい SQL または API ノードを追加するには、config.ini ファイル (複数の管理サーバーを使用する場合は複数のファイル) に新しい [mysqld] または [api] セクションを追加したあとで、すべてのクラスタノードのローリング再起動を実行する必要があります。これは、新しい SQL または API ノードをクラスタに接続する前に実行する必要があります。
>
> 新しい SQL または API ノードがクラスタ構成内の以前に使用されていない API スロットを使用してクラスタに接続する場合、クラスタの再起動を実行する必要はありません。

18.3.3.4 その他の MySQL Cluster 構成パラメータ

このセクションのサマリー表では、MySQL Cluster 管理ノードを構成するために config.ini ファイルの [computer]、[tcp]、[shm]、および [sci] セクションで使用するパラメータについて説明します。個々のパラメータに関する詳しい説明およびその他の追加情報については、セクション18.3.2.8「MySQL クラスタの TCP/IP

接続」、セクション18.3.2.10「MySQL Cluster の共有メモリー接続」、またはセクション18.3.2.11「MySQL Cluster での SCI トランスポート接続」を必要に応じて参照してください。

再起動のタイプ　MySQL Cluster 構成パラメータの変更は、クラスタが再起動されるまで有効になりません。サマリー表では、特定のパラメータを変更するために必要な再起動のタイプを次のように示します。

- N - ノードの再起動: このパラメータは、ローリング再起動を使用して更新できます (セクション 18.5.5「MySQL Cluster のローリング再起動の実行」を参照してください)。
- S - システムの再起動: このパラメータの変更を有効にするには、クラスタを完全にシャットダウンしてから再起動する必要があります。
- I - 初期再起動: --initial オプションを使用してデータノードを再起動する必要があります。

再起動のタイプの詳細は、セクション18.3.3「MySQL Cluster 構成パラメータの概要」を参照してください。

表 18.5 コンピュータ構成パラメータ

パラメータ名	型または単位 デフォルト値 最小値/最大値または許可される値	再起動タイプ	バージョン ... (以降)
HostName	名前または IP アドレス [none] ...	N	NDB 7.3.0
Id	文字列 [none] ...	IS	NDB 7.3.0

表 18.6 TCP 構成パラメータ

パラメータ名	型または単位 デフォルト値 最小値/最大値または許可される値	再起動タイプ	バージョン ... (以降)
Checksum	ブール false true、false	N	NDB 7.3.0
Group	符号なし 55 0 / 200	N	NDB 7.3.0
NodeId1	数値 [none] ...	N	NDB 7.3.0
NodeId2	数値 [none] ...	N	NDB 7.3.0
NodeIdServer	数値 [none] ...	N	NDB 7.3.0
OverloadLimit	バイト 0	N	NDB 7.3.0

パラメータ名	型または単位 デフォルト値 最小値/最大値または許可される値	再起動タイプ	バージョン ... (以降)
	0 / 4294967039 (0xFFFFFEFF)		
PortNumber	符号なし [none] 0 / 64K	N	NDB 7.3.0
Proxy	文字列 [none] ...	N	NDB 7.3.0
ReceiveBufferMemory	バイト 2M 16K / 4294967039 (0xFFFFFEFF)	N	NDB 7.3.0
SendBufferMemory	符号なし 2M 256K / 4294967039 (0xFFFFFEFF)	N	NDB 7.3.0
SendSignalId	ブール [テキストを参照] true、false	N	NDB 7.3.0
TCP_MAXSEG_SIZE	符号なし 0 0 / 2G	N	NDB 7.3.0
TCP_RCV_BUF_SIZE	符号なし 0 0 / 2G	N	NDB 7.3.1
TCP_SND_BUF_SIZE	符号なし 0 0 / 2G	N	NDB 7.3.1
TcpBind_INADDR_ANY	ブール false true、false	N	NDB 7.3.0

表 18.7 共有メモリー構成パラメータ

パラメータ名	型または単位 デフォルト値 最小値/最大値または許可される値	再起動タイプ	バージョン ... (以降)
Checksum	ブール true true、false	N	NDB 7.3.0

パラメータ名	型または単位 デフォルト値 最小値/最大値または許可される値	再起動タイプ	バージョン ... (以降)
Group	符号なし 35 0 / 200	N	NDB 7.3.0
NodeId1	数値 [none] ...	N	NDB 7.3.0
NodeId2	数値 [none] ...	N	NDB 7.3.0
NodeIdServer	数値 [none] ...	N	NDB 7.3.0
OverloadLimit	バイト 0 0 / 4294967039 (0xFFFFFEFF)	N	NDB 7.3.0
PortNumber	符号なし [none] 0 / 64K	N	NDB 7.3.0
SendSignalId	ブール false true、false	N	NDB 7.3.0
ShmKey	符号なし [none] 0 / 4294967039 (0xFFFFFEFF)	N	NDB 7.3.0
ShmSize	バイト 1M 64K / 4294967039 (0xFFFFFEFF)	N	NDB 7.3.0
Signum	符号なし [none] 0 / 4294967039 (0xFFFFFEFF)	N	NDB 7.3.0

表 18.8 SCI 構成パラメータ

パラメータ名	型または単位 デフォルト値 最小値/最大値または許可される値	再起動タイプ	バージョン ... (以降)
Checksum	ブール	N	NDB 7.3.0

パラメータ名	型または単位 デフォルト値 最小値/最大値または許可される値	再起動タイプ	バージョン ... (以降)
	false true、false		
Group	符号なし 15 0 / 200	N	NDB 7.3.0
Host1ScId0	符号なし [none] 0 / 4294967039 (0xFFFFFEFF)	N	NDB 7.3.0
Host1ScId1	符号なし 0 0 / 4294967039 (0xFFFFFEFF)	N	NDB 7.3.0
Host2ScId0	符号なし [none] 0 / 4294967039 (0xFFFFFEFF)	N	NDB 7.3.0
Host2ScId1	符号なし 0 0 / 4294967039 (0xFFFFFEFF)	N	NDB 7.3.0
NodeId1	数値 [none] ...	N	NDB 7.3.0
NodeId2	数値 [none] ...	N	NDB 7.3.0
NodeIdServer	数値 [none] ...	N	NDB 7.3.0
OverloadLimit	バイト 0 0 / 4294967039 (0xFFFFFEFF)	N	NDB 7.3.0
PortNumber	符号なし [none] 0 / 64K	N	NDB 7.3.0
SendLimit	符号なし 8K 128 / 32K	N	NDB 7.3.0
SendSignalId	ブール true	N	NDB 7.3.0

パラメータ名	型または単位 デフォルト値 最小値/最大値または許可される値	再起動タイプ	バージョン ... (以降)
	true、false		
SharedBufferSize	符号なし 10M 64K / 4294967039 (0xFFFFFEFF)	N	NDB 7.3.0

18.3.4 MySQL Cluster 用の MySQL Server オプションおよび変数

このセクションでは、MySQL Cluster に固有の MySQL サーバーオプションとサーバーおよびステータス変数について説明します。これらの使用に関する一般情報と、MySQL Cluster に固有でないその他のオプションおよび変数については、セクション5.1「MySQL Server」を参照してください。

クラスタ構成ファイル (通常、config.ini という名前) で使用する MySQL Cluster 構成パラメータについては、セクション18.3「MySQL Cluster の構成」を参照してください。

18.3.4.1 MySQL Cluster の mysqld オプションおよび変数のリファレンス

次の表に、MySQL Cluster 内で SQL ノードとして実行されているときの mysqld に適用されるコマンド行オプションとサーバーおよびステータス変数のリストを示します。mysqld で使用できるすべてのコマンド行オプションとサーバーおよびステータス変数を示す表については、セクション5.1.1「サーバーオプションおよび変数リファレンス」を参照してください。

表 18.9 MySQL Cluster 用の MySQL Server オプションおよび変数: MySQL Cluster NDB 7.3-7.4

オプションまたは変数名		
コマンド行	システム変数	ステータス変数
オプションファイル	スコープ	動的
メモ		
Handler_discover		
いいえ	いいえ	はい
いいえ	両方	いいえ
説明: テーブルが検出された回数		
Ndb_api_wait_exec_complete_count_session		
いいえ	いいえ	はい
いいえ	セッション	いいえ
説明: このクライアントセッションで、操作の実行が完了するのを待機する間にスレッドがブロックされた回数。		
Ndb_api_wait_exec_complete_count_slave		
いいえ	いいえ	はい
いいえ	グローバル	いいえ
説明: このスレーブによって、操作の実行が完了するのを待機する間にスレッドがブロックされた回数。		
Ndb_api_wait_exec_complete_count		
いいえ	いいえ	はい
いいえ	グローバル	いいえ
説明: この MySQL Server (SQL ノード) によって、操作の実行が完了するのを待機する間にスレッドがブロックされた回数。		
Ndb_api_wait_scan_result_count_session		
いいえ	いいえ	はい

オプションまたは変数名		
コマンド行	システム変数	ステータス変数
オプションファイル	スコープ	動的
メモ		
いいえ	セッション	いいえ
説明: このクライアントセッションで、スキャンベースの信号を待機する間にスレッドがブロックされた回数。 Ndb_api_wait_scan_result_count_slave		
いいえ	いいえ	はい
いいえ	グローバル	いいえ
説明: このスレーブによって、スキャンベースの信号を待機する間にスレッドがブロックされた回数。 Ndb_api_wait_scan_result_count		
いいえ	いいえ	はい
いいえ	グローバル	いいえ
説明: この MySQL Server (SQL ノード) によって、スキャンベースの信号を待機する間にスレッドがブロックされた回数。 Ndb_api_wait_meta_request_count_session		
いいえ	いいえ	はい
いいえ	セッション	いいえ
説明: このクライアントセッションで、メタデータベースの信号を待機する間にスレッドがブロックされた回数。 Ndb_api_wait_meta_request_count_slave		
いいえ	いいえ	はい
いいえ	グローバル	いいえ
説明: このスレーブによって、メタデータベースの信号を待機する間にスレッドがブロックされた回数。 Ndb_api_wait_meta_request_count		
いいえ	いいえ	はい
いいえ	グローバル	いいえ
説明: この MySQL Server (SQL ノード) によって、メタデータベースの信号を待機する間にスレッドがブロックされた回数。 Ndb_api_wait_nanos_count_session		
いいえ	いいえ	はい
いいえ	セッション	いいえ
説明: このクライアントセッションで、データノードからのある種の信号の待機にかかった合計時間 (ナノ秒)。 Ndb_api_wait_nanos_count_slave		
いいえ	いいえ	はい
いいえ	グローバル	いいえ
説明: このスレーブによって、データノードからのある種の信号の待機にかかった合計時間 (ナノ秒)。 Ndb_api_wait_nanos_count		
いいえ	いいえ	はい
いいえ	グローバル	いいえ
説明: この MySQL Server (SQL ノード) によって、データノードからのある種の信号の待機にかかった合計時間 (ナノ秒)。 Ndb_api_bytes_sent_count_session		
いいえ	いいえ	はい
いいえ	セッション	いいえ
説明: このクライアントセッションで、データノードに送信されたデータ量 (バイト単位)。 Ndb_api_bytes_sent_count_slave		

オプションまたは変数名		
コマンド行	システム変数	ステータス変数
オプションファイル	スコープ	動的
メモ		
いいえ	いいえ	はい
いいえ	グローバル	いいえ
説明: このスレーブによって、データノードに送信されたデータ量 (バイト単位)。 Ndb_api_bytes_sent_count		
いいえ	いいえ	はい
いいえ	グローバル	いいえ
説明: この MySQL Server (SQL ノード) によって、データノードに送信されたデータ量 (バイト単位)。 Ndb_api_bytes_received_count_session		
いいえ	いいえ	はい
いいえ	セッション	いいえ
説明: このクライアントセッションで、データノードから受信されたデータ量 (バイト単位)。 Ndb_api_bytes_received_count_slave		
いいえ	いいえ	はい
いいえ	グローバル	いいえ
説明: このスレーブによって、データノードから受信されたデータ量 (バイト単位)。 Ndb_api_bytes_received_count		
いいえ	いいえ	はい
いいえ	グローバル	いいえ
説明: この MySQL Server (SQL ノード) によって、データノードから受信されたデータ量 (バイト単位)。 Ndb_api_trans_start_count_session		
いいえ	いいえ	はい
いいえ	セッション	いいえ
説明: このクライアントセッションで開始されたトランザクションの数。 Ndb_api_trans_start_count_slave		
いいえ	いいえ	はい
いいえ	グローバル	いいえ
説明: このスレーブにより開始されたトランザクションの数。 Ndb_api_trans_start_count		
いいえ	いいえ	はい
いいえ	グローバル	いいえ
説明: この MySQL Server (SQL ノード) により開始されたトランザクションの数。 Ndb_api_trans_commit_count_session		
いいえ	いいえ	はい
いいえ	セッション	いいえ
説明: このクライアントセッションでコミットされたトランザクションの数。 Ndb_api_trans_commit_count_slave		
いいえ	いいえ	はい
いいえ	グローバル	いいえ
説明: このスレーブによりコミットされたトランザクションの数。 Ndb_api_trans_commit_count		
いいえ	いいえ	はい

オプションまたは変数名		
コマンド行	システム変数	ステータス変数
オプションファイル	スコープ	動的
メモ		
いいえ	グローバル	いいえ
説明: この MySQL Server (SQL ノード) によりコミットされたトランザクションの数。 Ndb_api_trans_abort_count_session		
いいえ	いいえ	はい
いいえ	セッション	いいえ
説明: このクライアントセッションで中止されたトランザクションの数。 Ndb_api_trans_abort_count_slave		
いいえ	いいえ	はい
いいえ	グローバル	いいえ
説明: このスレーブにより中止されたトランザクションの数。 Ndb_api_trans_abort_count		
いいえ	いいえ	はい
いいえ	グローバル	いいえ
説明: この MySQL Server (SQL ノード) により中止されたトランザクションの数。 Ndb_api_trans_close_count_session		
いいえ	いいえ	はい
いいえ	セッション	いいえ
説明: このクライアントセッションで、中止されたトランザクションの数 (TransCommitCount と TransAbortCount との合計より大きい場合があります)。 Ndb_api_trans_close_count_slave		
いいえ	いいえ	はい
いいえ	グローバル	いいえ
説明: このスレーブによって中止されたトランザクションの数 (TransCommitCount と TransAbortCount との合計より大きい場合があります)。 Ndb_api_trans_close_count		
いいえ	いいえ	はい
いいえ	グローバル	いいえ
説明: この MySQL Server (SQL ノード) によって中止されたトランザクションの数 (TransCommitCount と TransAbortCount との合計より大きい場合があります)。 Ndb_api_pk_op_count_session		
いいえ	いいえ	はい
いいえ	セッション	いいえ
説明: このクライアントセッションで、主キーに基づいた、または主キーを使用した操作の数。 Ndb_api_pk_op_count_slave		
いいえ	いいえ	はい
いいえ	グローバル	いいえ
説明: このスレーブによる、主キーに基づいた、または主キーを使用した操作の数。 Ndb_api_pk_op_count		
いいえ	いいえ	はい
いいえ	グローバル	いいえ
説明: この MySQL Server (SQL ノード) による、主キーに基づいた、または主キーを使用した操作の数。 Ndb_api_uk_op_count_session		

オプションまたは変数名		
コマンド行	システム変数	ステータス変数
オプションファイル	スコープ	動的
メモ		
いいえ	いいえ	はい
いいえ	セッション	いいえ
説明: このクライアントセッションで、一意のキーに基づいた、または一意のキーを使用した操作の数。 Ndb_api_uk_op_count_slave		
いいえ	いいえ	はい
いいえ	グローバル	いいえ
説明: このスレーブによる、一意のキーに基づいた、または一意のキーを使用した操作の数。 Ndb_api_uk_op_count		
いいえ	いいえ	はい
いいえ	グローバル	いいえ
説明: この MySQL Server (SQL ノード) による、一意のキーに基づいた、または一意のキーを使用した操作の数。 Ndb_api_table_scan_count_session		
いいえ	いいえ	はい
いいえ	セッション	いいえ
説明: このクライアントセッションで開始された、内部テーブルのスキャンを含むテーブルスキャンの回数。 Ndb_api_table_scan_count_slave		
いいえ	いいえ	はい
いいえ	グローバル	いいえ
説明: このスレーブによって開始された、内部テーブルのスキャンを含むテーブルスキャンの回数。 Ndb_api_table_scan_count		
いいえ	いいえ	はい
いいえ	グローバル	いいえ
説明: この MySQL Server (SQL ノード) によって開始された、内部テーブルのスキャンを含むテーブルスキャンの回数。 Ndb_api_range_scan_count_session		
いいえ	いいえ	はい
いいえ	セッション	いいえ
説明: このクライアントセッションで開始された範囲スキャンの回数。 Ndb_api_range_scan_count_slave		
いいえ	いいえ	はい
いいえ	グローバル	いいえ
説明: このスレーブによって開始された範囲スキャンの回数。 Ndb_api_range_scan_count		
いいえ	いいえ	はい
いいえ	グローバル	いいえ
説明: この MySQL Server (SQL ノード) によって開始された範囲スキャンの回数。 Ndb_api_pruned_scan_count_session		
いいえ	いいえ	はい
いいえ	セッション	いいえ
説明: このクライアントセッションで単一のパーティションにプルーニングされたスキャンの回数。 Ndb_api_pruned_scan_count_slave		

オプションまたは変数名		
コマンド行	システム変数	ステータス変数
オプションファイル	スコープ	動的
メモ		
いいえ	いいえ	はい
いいえ	グローバル	いいえ
説明: このスレーブによって単一のパーティションにプルーニングされたスキャンの回数。 Ndb_api_pruned_scan_count		
いいえ	いいえ	はい
いいえ	グローバル	いいえ
説明: この MySQL Server (SQL ノード) によって単一のパーティションにプルーニングされたスキャンの回数。 Ndb_api_scan_batch_count_session		
いいえ	いいえ	はい
いいえ	セッション	いいえ
説明: このクライアントセッションで受信されたバッチの行数。 Ndb_api_scan_batch_count_slave		
いいえ	いいえ	はい
いいえ	グローバル	いいえ
説明: このスレーブによって受信されたバッチの行数。 Ndb_api_scan_batch_count		
いいえ	いいえ	はい
いいえ	グローバル	いいえ
説明: この MySQL Server (SQL ノード) によって受信されたバッチの行数。 Ndb_api_read_row_count_session		
いいえ	いいえ	はい
いいえ	セッション	いいえ
説明: このクライアントセッションで読み取られた行の合計数。 Ndb_api_read_row_count_slave		
いいえ	いいえ	はい
いいえ	グローバル	いいえ
説明: このスレーブによって読み取られた行の合計数。 Ndb_api_read_row_count		
いいえ	いいえ	はい
いいえ	グローバル	いいえ
説明: この MySQL Server (SQL ノード) によって読み取られた行の合計数。 Ndb_api_trans_local_read_row_count_session		
いいえ	いいえ	はい
いいえ	セッション	いいえ
説明: このクライアントセッションで読み取られた行の合計数。 Ndb_api_trans_local_read_row_count_slave		
いいえ	いいえ	はい
いいえ	グローバル	いいえ
説明: このスレーブによって読み取られた行の合計数。 Ndb_api_trans_local_read_row_count		
いいえ	いいえ	はい

オプションまたは変数名		
コマンド行	システム変数	ステータス変数
オプションファイル	スコープ	動的
メモ		
いいえ	グローバル	いいえ
説明: この MySQL Server (SQL ノード) によって読み取られた行の合計数。 Ndb_api_event_data_count_injector		
いいえ	いいえ	はい
いいえ	グローバル	いいえ
説明: NDB バイナリログインジェクタスレッドによって受信された行変更イベントの数。 Ndb_api_event_data_count		
いいえ	いいえ	はい
いいえ	グローバル	いいえ
説明: この MySQL Server (SQL ノード) によって受信された行変更イベントの数。 Ndb_api_event_nondata_count_injector		
いいえ	いいえ	はい
いいえ	グローバル	いいえ
説明: NDB バイナリログインジェクタスレッドによって受信された、行変更イベント以外のイベントの数。 Ndb_api_event_nondata_count		
いいえ	いいえ	はい
いいえ	グローバル	いいえ
説明: この MySQL Server (SQL ノード) によって受信された、行変更イベント以外のイベントの数。 Ndb_api_event_bytes_count_injector		
いいえ	いいえ	はい
いいえ	グローバル	いいえ
説明: NDB バイナリログインジェクタスレッドによって受信されたイベントのバイト数。 Ndb_api_event_bytes_count		
いいえ	いいえ	はい
いいえ	グローバル	いいえ
説明: この MySQL Server (SQL ノード) によって受信されたイベントのバイト数。 Ndb_conflict_fn_max		
いいえ	いいえ	はい
いいえ	グローバル	いいえ
説明: サーバーが、クラスタレプリケーションに関連する MySQL Cluster の一部である場合、この変数の値は、「大きい方のタイムスタンプを優先」に基づく競合解決が適用された回数を示します Ndb_conflict_fn_old		
いいえ	いいえ	はい
いいえ	グローバル	いいえ
説明: サーバーが、クラスタレプリケーションに関連する MySQL Cluster の一部である場合、この変数の値は、「同じタイムスタンプを優先」の競合解決が適用された回数を示します Ndb_conflict_fn_epoch		
いいえ	いいえ	はい
いいえ	グローバル	いいえ
説明: NDB$EPOCH() 競合検出関数によって競合状態であることが検出された行数 Ndb_conflict_fn_epoch_trans		
いいえ	いいえ	はい

オプションまたは変数名		
コマンド行	システム変数	ステータス変数
オプションファイル	スコープ	動的
メモ		
いいえ	グローバル	いいえ
説明: NDB$EPOCH_TRANS() 競合検出関数によって競合状態であることが検出された行数 Ndb_conflict_trans_row_conflict_count		
いいえ	いいえ	はい
いいえ	グローバル	いいえ
説明: トランザクション競合関数によって競合状態であることが検出された行数 Ndb_conflict_trans_row_reject_count		
いいえ	いいえ	はい
いいえ	グローバル	いいえ
説明: トランザクション競合関数によって競合状態であることが検出されたあと、再調整された行の合計数。Ndb_conflict_trans_row_conflict_count、および競合するトランザクションに含まれているか、または依存しているすべての行が含まれます。 Ndb_conflict_trans_reject_count		
いいえ	いいえ	はい
いいえ	グローバル	いいえ
説明: トランザクション競合関数によって競合状態であることが検出されたあと、拒否されたトランザクションの数。 Ndb_conflict_trans_detect_iter_count		
いいえ	いいえ	はい
いいえ	グローバル	いいえ
説明: エポックトランザクションのコミットに必要な内部反復回数。Ndb_conflict_trans_conflict_commit_count より (わずかに) 大きいか、または等しい値にしてください。 Ndb_conflict_trans_conflict_commit_count		
いいえ	いいえ	はい
いいえ	グローバル	いいえ
説明: トランザクション競合処理を要求したあとにコミットされたエポックトランザクションの数。 Ndb_number_of_data_nodes		
いいえ	いいえ	はい
いいえ	グローバル	いいえ
説明: サーバーが MySQL Cluster の一部である場合、この変数の値はクラスタ内のデータノードの数 Ndb_slave_last_conflict_epoch		
いいえ	はい	いいえ
いいえ	グローバル	いいえ
説明: このスレーブで、競合が検出された直近の NDB エポック。 Ndb_slave_max_replicated_epoch		
いいえ	はい	いいえ
いいえ	グローバル	いいえ
説明: このスレーブで、直近でコミットされた NDB エポック。この値が Ndb_slave_last_conflict_epoch 以上である場合、まだ競合は検出されていません。 ndb_optimization_delay		
いいえ	はい	いいえ
いいえ	グローバル	はい

オプションまたは変数名		
コマンド行	システム変数	ステータス変数
オプションファイル	スコープ	動的
メモ		
説明: NDB テーブルに対して OPTIMIZE TABLE で行のセットを処理する間に待機する時間をミリ秒で設定します。 ndb_table_temporary		
いいえ	はい	いいえ
いいえ	セッション	はい
説明: NDB テーブルはディスク上で永続的ではありません。スキーマファイルは作成されず、テーブルはログに記録されません ndb_table_no_logging		
いいえ	はい	いいえ
いいえ	セッション	はい
説明: この設定が有効なときに作成された NDB テーブルはディスクに対してチェックポイントを設定しません (ただし、テーブルスキーマファイルは作成されます)。テーブルが NDBCLUSTER で作成されるか、またはそれを使用するように変更されたときに有効な設定は、そのテーブルの有効期間にわたって保持されます。 ndb_autoincrement_prefetch_sz		
はい	はい	いいえ
はい	両方	はい
説明: NDB 自動インクリメント値のプリフェッチサイズ ndb-batch-size		
はい	はい	いいえ
はい	グローバル	いいえ
説明: NDB トランザクションバッチで使用するサイズ (バイト単位) ndb-blob-read-batch-bytes		
はい	はい	いいえ
はい	両方	はい
説明: 大規模な BLOB 読み取りがバッチ化されるサイズをバイト単位で指定します。0 = 制限なし。 ndb-blob-write-batch-bytes		
はい	はい	いいえ
はい	両方	はい
説明: 大規模な BLOB 書き込みがバッチ化されるサイズをバイト単位で指定します。0 = 制限なし。 ndb_cache_check_time		
はい	はい	いいえ
はい	グローバル	はい
説明: MySQL クエリーキャッシュによって行われるクラスタ SQL ノードのチェック間のミリ秒数 ndb-cluster-connection-pool		
はい	はい	はい
はい	グローバル	いいえ
説明: MySQL によって使用されるクラスタへの接続の数 ndb_join_pushdown		
いいえ	はい	いいえ
いいえ	両方	はい
説明: データノードへの結合のプッシュダウンを有効化 Ndb_execute_count		
いいえ	いいえ	はい

オプションまたは変数名		
コマンド行	システム変数	ステータス変数
オプションファイル	スコープ	動的
メモ		
いいえ	グローバル	いいえ
説明: 操作によって行われる NDB カーネルへのラウンドトリップの数を指定		
Ndb_scan_count		
いいえ	いいえ	はい
いいえ	グローバル	いいえ
説明: クラスタが最後に起動されたあとに NDB によって実行されたスキャンの合計数		
Ndb_pruned_scan_count		
いいえ	いいえ	はい
いいえ	グローバル	いいえ
説明: クラスタが最後に起動されたあとに NDB によって実行された、パーティションプルーニングが使用可能であったスキャンの数		
ndb-connectstring		
はい	いいえ	いいえ
はい		いいえ
説明: クラスタ構成を配布する管理サーバーへのポイント		
ndb_extra_logging		
はい	はい	いいえ
はい	グローバル	はい
説明: MySQL Cluster スキーマ、接続、およびデータ配布イベントの、MySQL エラーログへのロギングを制御		
ndb_force_send		
はい	はい	いいえ
はい	両方	はい
説明: ほかのスレッドを待機することなく、ただちにバッファーを NDB に強制的に送信		
ndb_use_exact_count		
いいえ	はい	いいえ
いいえ	両方	はい
説明: クエリーを計画する際に正確な行数を使用		
ndb-log-apply-status		
はい	はい	いいえ
はい	グローバル	いいえ
説明: スレーブとして機能している MySQL サーバーが、直接のマスターから受信した mysql.ndb_apply_status の更新を、自身のサーバー ID を使用して自身のバイナリログに記録するようになります。サーバーが --ndbcluster オプションで起動された場合にのみ有効です。		
ndb_log_apply_status		
はい	はい	いいえ
はい	グローバル	いいえ
説明: スレーブとして機能している MySQL サーバーが、直接のマスターから受信した mysql.ndb_apply_status の更新を、自身のサーバー ID を使用して自身のバイナリログに記録するかどうか。		
ndb_log_bin		
はい	はい	いいえ
いいえ	両方	はい
説明: NDB テーブルの更新をバイナリログに書き込みます。--log-bin でバイナリロギングが有効になっている場合にのみ有効です。		

オプションまたは変数名		
コマンド行	システム変数	ステータス変数
オプションファイル	スコープ	動的
メモ		
ndb_log_binlog_index		
はい	はい	いいえ
いいえ	グローバル	はい
説明: エポックとバイナリログの位置の間のマッピングを ndb_binlog_index テーブルに挿入します。デフォルトでは ON になります。サーバー上でバイナリロギングが有効になっている場合にのみ有効です。		
ndb-log-exclusive-reads		
はい	はい	いいえ
はい	両方	はい
説明: 排他的ロックによる主キーの読み取りをログに記録します。読み取りの競合に基づく競合解決を許可します。		
ndb_log_exclusive_reads		
はい	はい	いいえ
はい	両方	はい
説明: 排他的ロックによる主キーの読み取りをログに記録します。読み取りの競合に基づく競合解決を許可します。		
ndb-log-orig		
はい	はい	いいえ
はい	グローバル	いいえ
説明: 発信サーバー ID とエポックを mysql.ndb_binlog_index テーブルに記録します。		
ndb_log_orig		
はい	はい	いいえ
はい	グローバル	いいえ
説明: 発信サーバーの ID とエポックが mysql.ndb_binlog_index テーブルに記録されるかどうか。mysqld を起動するときに --ndb-log-orig オプションを使用して設定します。		
ndb-log-transaction-id		
はい	はい	いいえ
はい	グローバル	いいえ
説明: NDB トランザクション ID をバイナリログに書き込みます。--log-bin-v1-events=OFF が必要です。		
ndb_log_transaction_id		
いいえ	はい	いいえ
いいえ	グローバル	いいえ
説明: NDB トランザクション ID がバイナリログに書き込まれるかどうか。(読み取り専用。)		
ndb_log_updated_only		
はい	はい	いいえ
はい	グローバル	はい
説明: 完全な行 (ON) または更新のみ (OFF) をログに記録		
ndb-log-update-as-write		
はい	はい	いいえ
はい	グローバル	はい
説明: マスターでの更新のロギングを、更新 (OFF) と書き込み (ON) の間で切り替え		
ndb-log-empty-epochs		
はい	はい	いいえ
はい	グローバル	はい

オプションまたは変数名		
コマンド行	システム変数	ステータス変数
オプションファイル	スコープ	動的
メモ		
説明: 有効な場合は、--log-slave-updates が有効であっても、変更がなかったエポックが ndb_apply_status および ndb_binlog_index テーブルに書き込まれるようになります。 ndb_log_empty_epochs		
はい	はい	いいえ
はい	グローバル	はい
説明: 有効な場合は、log_slave_updates が有効であっても、変更がなかったエポックが ndb_apply_status および ndb_binlog_index テーブルに書き込まれます。 ndb_slave_conflict_role		
はい	はい	いいえ
はい	グローバル	はい
説明: スレーブが、競合の検出と解決で果たすロール。値は PRIMARY、SECONDARY、PASS、NONE (デフォルト) のいずれかです。スレーブ SQL スレッドが停止されている場合にのみ変更できます。詳細は、ドキュメントを参照してください。 ndb-transid-mysql-connection-map		
はい	いいえ	いいえ
いいえ		いいえ
説明: ndb_transid_mysql_connection_map プラグインを有効または無効にします。つまり、その名前を持つ INFORMATION_SCHEMA テーブルを有効または無効にします。 Ndb_pushed_reads		
いいえ	いいえ	はい
いいえ	グローバル	いいえ
説明: プッシュダウン結合によってデータノード上で実行された読み取りの数 Ndb_pushed_queries_defined		
いいえ	いいえ	はい
いいえ	グローバル	いいえ
説明: API ノードがデータノードにプッシュダウンしようとした結合の数 Ndb_pushed_queries_dropped		
いいえ	いいえ	はい
いいえ	グローバル	いいえ
説明: API ノードがプッシュダウンしようとしたが、失敗した結合の数 Ndb_pushed_queries_executed		
いいえ	いいえ	はい
いいえ	グローバル	いいえ
説明: 正常にプッシュダウンされ、データノード上で実行された結合の数 ndb-nodeid		
はい	いいえ	はい
はい	グローバル	いいえ
説明: この MySQL サーバーの MySQL Cluster ノード ID ndb-mgmd-host		
はい	いいえ	いいえ
はい		いいえ
説明: 管理サーバーに接続するためのホスト (およびポート (必要な場合)) を設定します ndb-recv-thread-activation-threshold		

オプションまたは変数名		
コマンド行	システム変数	ステータス変数
オプションファイル	スコープ	動的
メモ		
はい	いいえ	いいえ
はい		いいえ
説明: 受信スレッドがクラスタ接続のポーリングを引き継ぐときのアクティブ化のしきい値 (同時にアクティブなスレッドで測定されます)		
ndb_recv_thread_activation_threshold		
いいえ	いいえ	いいえ
いいえ		いいえ
説明: 受信スレッドがクラスタ接続のポーリングを引き継ぐときのアクティブ化のしきい値 (同時にアクティブなスレッドで測定されます)		
ndb-recv-thread-cpu-mask		
はい	いいえ	いいえ
はい		いいえ
説明: 受信者スレッドを特定の CPU にロックするための CPU マスク。16 進数として指定されます。詳細は、ドキュメントを参照してください。		
ndb_recv_thread_cpu_mask		
いいえ	はい	いいえ
いいえ	グローバル	はい
説明: 受信者スレッドを特定の CPU にロックするための CPU マスク。16 進数として指定されます。詳細は、ドキュメントを参照してください。		
ndb-wait-connected		
はい	はい	いいえ
はい	グローバル	いいえ
説明: MySQL サーバーが、MySQL クライアントの接続を受け入れる前にクラスタ管理およびデータノードへの接続を待機する時間 (秒単位)。		
ndb-wait-setup		
はい	はい	いいえ
はい	グローバル	いいえ
説明: MySQL サーバーが NDB エンジンのセットアップの完了を待機する時間 (秒単位)。		
ndbcluster		
はい	いいえ	いいえ
はい		いいえ
説明: NDB Cluster を有効化 (このバージョンの MySQL がサポートしている場合) --skip-ndbcluster によって無効になります		
ndbinfo_database		
いいえ	はい	いいえ
いいえ	グローバル	いいえ
説明: NDB 情報データベースに使用される名前。読み取り専用。		
ndbinfo_max_bytes		
はい	はい	いいえ
いいえ	両方	はい
説明: デバッグにのみ使用されます。		
ndbinfo_max_rows		
はい	はい	いいえ

オプションまたは変数名		
コマンド行	システム変数	ステータス変数
オプションファイル	スコープ	動的
メモ		
いいえ	両方	はい
説明: デバッグにのみ使用されます。		
ndbinfo_offline		
いいえ	はい	いいえ
いいえ	グローバル	はい
説明: ndbinfo データベースをオフラインモードにします。この場合、テーブルまたはビューから行は返されません。		
ndbinfo_show_hidden		
はい	はい	いいえ
いいえ	両方	はい
説明: ndbinfo 内部ベーステーブルを mysql クライアントで表示するかどうか。デフォルトは OFF です。		
ndbinfo_table_prefix		
はい	はい	いいえ
いいえ	両方	はい
説明: ndbinfo 内部ベーステーブルの命名に使用するプリフィクス		
ndbinfo_version		
いいえ	はい	いいえ
いいえ	グローバル	いいえ
説明: ndbinfo エンジンのバージョン。読み取り専用。		
server-id-bits		
はい	はい	いいえ
はい	グローバル	いいえ
説明: サーバーを識別するために実際に使用される server_id 内の最下位ビットの数を設定することにより、NDB API アプリケーションが最上位ビットにアプリケーションデータを格納できるようにします。server_id は 2 の「この値」乗未満である必要があります。		
server_id_bits		
はい	はい	いいえ
はい	グローバル	いいえ
説明: --server-id-bits オプションがデフォルト以外の値に設定された状態でサーバーが起動された場合の server_id の有効な値。		
slave_allow_batching		
はい	はい	いいえ
はい	グローバル	はい
説明: レプリケーションスレーブの更新バッチ化をオンおよびオフにする		
transaction_allow_batching		
いいえ	はい	いいえ
いいえ	セッション	はい
説明: トランザクション内でのステートメントのバッチ化を許可します。使用するには AUTOCOMMIT を無効にします。		
have_ndbcluster		
いいえ	はい	いいえ
いいえ	グローバル	いいえ
説明: mysqld が NDB Cluster テーブルをサポートするかどうか (--ndbcluster オプションで設定されます)。		

オプションまたは変数名		
コマンド行	システム変数	ステータス変数
オプションファイル	スコープ	動的
メモ		
ndb-deferred-constraints		
はい	はい	いいえ
はい	両方	はい
説明: 一意のインデックスに関する制約チェック (サポートされている場合) を、コミット時まで遅延させることを指定します。通常は必要なく、使用されません。テストの目的にのみ使用されます。		
ndb_deferred_constraints		
はい	はい	いいえ
はい	両方	はい
説明: 制約チェック (サポートされている場合) を遅延させることを指定します。通常は必要なく、使用されません。テストの目的にのみ使用されます。		
ndb-distribution		
はい	はい	いいえ
はい	グローバル	はい
説明: NDBCLUSTER の新しいテーブルのデフォルトの配布 (KEYHASH または LINHASH、デフォルトは KEYHASH)		
ndb_distribution		
はい	はい	いいえ
はい	グローバル	はい
説明: NDBCLUSTER の新しいテーブルのデフォルトの配布 (KEYHASH または LINHASH、デフォルトは KEYHASH)		
ndb_eventbuffer_max_alloc		
はい	はい	いいえ
はい	グローバル	はい
説明: NDB API によるイベントのバッファリングのために割り当てることのできる最大メモリー。デフォルトでは 0 (制限なし) になります。		
ndb_index_stat_cache_entries		
はい	はい	いいえ
はい	両方	はい
説明: 開始キーと終了キーの数を決定することにより、統計の粒度を設定		
ndb_index_stat_enable		
はい	はい	いいえ
はい	両方	はい
説明: クエリーの最適化で NDB インデックス統計を使用		
ndb_index_stat_option		
はい	はい	いいえ
はい	両方	はい
説明: NDB インデックス統計の調整可能なオプションのカンマ区切りのリスト。リストはスペースを含んではなりません		
ndb_index_stat_update_freq		
はい	はい	いいえ
はい	両方	はい
説明: 統計キャッシュではなくデータノードにクエリーを実行する頻度		
ndb_optimized_node_selection		

オプションまたは変数名		
コマンド行	システム変数	ステータス変数
オプションファイル	スコープ	動的
メモ		
はい	はい	いいえ
はい	グローバル	いいえ
説明: SQL ノードが、トランザクションコーディネータとして使用するためにクラスタデータノードを選択する方法を決定 ndb_report_thresh_binlog_epoch_slip		
はい	いいえ	いいえ
はい		いいえ
説明: これは、バイナリログステータスをレポートする前に、過ぎていなければならないエポックの数のしきい値です ndb_report_thresh_binlog_mem_usage		
はい	いいえ	いいえ
はい		いいえ
説明: これは、バイナリログステータスをレポートする前に、残っている空きメモリーの割合のしきい値です。 ndb_show_foreign_key_mock_tables		
はい	はい	いいえ
はい	グローバル	はい
説明: foreign_key_checks=0 をサポートするために使用される擬似テーブルを表示します。 ndb_use_transactions		
はい	はい	いいえ
はい	両方	はい
説明: SELECT COUNT(*) クエリー計画中に、強制的に NDB にレコードの数を使用してこのタイプのクエリーを高速化させる ndb_version		
いいえ	はい	いいえ
いいえ	グローバル	いいえ
説明: ビルドおよび NDB エンジンのバージョンを整数として表示します。 ndb_version_string		
いいえ	はい	いいえ
いいえ	グローバル	いいえ
説明: NDB エンジンのバージョンを含むビルド情報を ndb-x.y.z の形式で表示します。 Ndb_cluster_node_id		
いいえ	いいえ	はい
いいえ	両方	いいえ
説明: サーバーが MySQL Cluster ノードとして機能している場合、この変数の値はクラスタ内のそのノード ID Ndb_config_from_host		
いいえ	いいえ	はい
いいえ	両方	いいえ
説明: Cluster 管理サーバーのホスト名または IP アドレス。以前は Ndb_connected_host Ndb_config_from_port		
いいえ	いいえ	はい
いいえ	両方	いいえ
説明: Cluster 管理サーバーに接続するためのポート。以前は Ndb_connected_port		

オプションまたは変数名		
コマンド行	システム変数	ステータス変数
オプションファイル	スコープ	動的
メモ		
Com_show_ndb_status		
いいえ	いいえ	はい
いいえ	両方	いいえ
説明: SHOW NDB STATUS ステートメントの数		

18.3.4.2 MySQL Cluster 用の MySQL Server オプション

このセクションでは、MySQL Cluster に関連する mysqld サーバーオプションについて説明します。MySQL Cluster に固有でない mysqld オプション、および mysqld でのオプションの使用に関する一般的な情報については、セクション5.1.3「サーバーコマンドオプション」を参照してください。

その他の MySQL Cluster プロセス (ndbd、ndb_mgmd、および ndb_mgm) で使用されるコマンド行オプションについては、セクション18.4.27「MySQL Cluster プログラムに共通するオプション — MySQL Cluster プログラムに共通するオプション」を参照してください。NDB のユーティリティープログラム (ndb_desc、ndb_size.pl、ndb_show_tables など) で使用されるコマンド行オプションについては、セクション18.4「MySQL Cluster プログラム」を参照してください。

- --ndb-batch-size=#

表 18.10 ndb-batch-size の型と値の情報

コマンド行	システム変数	ステータス変数
オプションファイル	スコープ	動的
開始バージョン	型	デフォルト、範囲
メモ		
ndb-batch-size		
はい	はい	いいえ
はい	グローバル	いいえ
NDB 7.3	数値	32768 / 0 - 31536000
説明: NDB トランザクションバッチで使用するサイズ (バイト単位)		

これは、NDB のトランザクションバッチで使用されるサイズ (バイト単位) を設定します。

- --ndb-cluster-connection-pool=#

表 18.11 ndb-cluster-connection-pool の型と値の情報

コマンド行	システム変数	ステータス変数
オプションファイル	スコープ	動的
開始バージョン	型	デフォルト、範囲
メモ		
ndb-cluster-connection-pool		
はい	はい	はい
はい	グローバル	いいえ
NDB 7.3	数値	1 / 1 - 63
説明: MySQL によって使用されるクラスタへの接続の数		

このオプションを 1 (デフォルト) より大きい値に設定すると、mysqld プロセスはクラスタに対して複数の接続を使用し、実質的に複数の SQL ノードを模倣します。各接続には、クラスタ構成 (config.ini) ファイル内に固有の [api] または [mysqld] セクションを必要とし、クラスタでサポートされる API 接続の最大数を参照してカウントされます。

2 台のクラスタホストコンピュータがあり、実行している各 SQL ノードの mysqld プロセスが --ndb-cluster-connection-pool=4 を指定して起動されたとします。これは、これらの接続に使用できる API スロットがクラスタに (2 個ではなく) 8 個存在する必要があることを意味します。これらすべての接続は、SQL ノードがクラスタに接続したときに設定され、ラウンドロビン方式でスレッドに割り当てられます。

このオプションは、複数の CPU、複数のコア、またはその両方を搭載したホストマシンで mysqld を実行する場合にのみ有効です。最良の結果を得るには、この値をホストマシンで使用できるコアの合計数より小さくします。これより大きい値に設定すると、パフォーマンスが大幅に低下する可能性があります。

> **重要**
>
> 接続プールを使用する各 SQL ノードは複数の API ノードスロットを占有する (各スロットにはクラスタ内で固有のノード ID がある) ため、接続プールを採用する mysqld プロセスを起動するときは、クラスタ接続文字列の一部としてノード ID を使用しないでください。
>
> --ndb-cluster-connection-pool を使用する際に接続文字列にノード ID を設定すると、SQL ノードのクラスタへの接続試行時に、ノード ID の割り当てエラーが発生します。

- --ndb-blob-read-batch-bytes=bytes

表 18.12 ndb-blob-read-batch-bytes の型と値の情報

コマンド行	システム変数	ステータス変数
オプションファイル	スコープ	動的
開始バージョン	型	デフォルト、範囲
メモ		
ndb-blob-read-batch-bytes		
はい	はい	いいえ
はい	両方	はい
NDB 7.3	数値	65536 / 0 - 4294967295
説明: 大規模な BLOB 読み取りがバッチ化されるサイズをバイト単位で指定します。0 = 制限なし。		

このオプションを使用すると、MySQL Cluster アプリケーションで BLOB データ読み取りのバッチ化のサイズ (バイト単位) を設定できます。現在のトランザクション内で読み取られる BLOB データの量がこのバッチサイズを超えると、保留中のすべての BLOB 読み取り操作がただちに実行されます。

このオプションの最大値は 4294967295 です。デフォルトは 65536 です。0 に設定すると、BLOB 読み取りのバッチ化を無効にします。

> **注記**
>
> NDB API アプリケーションでは、setMaxPendingBlobReadBytes() および getMaxPendingBlobReadBytes() メソッドを使用すると BLOB 書き込みのバッチ化を制御できます。

- --ndb-blob-write-batch-bytes=bytes

表 18.13 ndb-blob-write-batch-bytes の型と値の情報

コマンド行	システム変数	ステータス変数
オプションファイル	スコープ	動的
開始バージョン	型	デフォルト、範囲
メモ		
ndb-blob-write-batch-bytes		
はい	はい	いいえ
はい	両方	はい

コマンド行	システム変数	ステータス変数
オプションファイル	スコープ	動的
開始バージョン	型	デフォルト、範囲
メモ		
NDB 7.3	数値	65536 / 0 - 4294967295
説明: 大規模な BLOB 書き込みがバッチ化されるサイズをバイト単位で指定します。0 = 制限なし。		

このオプションを使用すると、MySQL Cluster アプリケーションで BLOB データ書き込みのバッチ化のサイズ (バイト単位) を設定できます。現在のトランザクション内で書き込まれる BLOB データの量がこのバッチサイズを超えると、保留中のすべての BLOB 書き込み操作がただちに実行されます。

このオプションの最大値は 4294967295 です。デフォルトは 65536 です。0 に設定すると、BLOB 書き込みのバッチ化を無効にします。

> **注記**
>
> NDB API アプリケーションでは、setMaxPendingBlobWriteBytes() および getMaxPendingBlobWriteBytes() メソッドを使用すると BLOB 書き込みのバッチ化を制御できます。

- --ndb-connectstring=connect_string

表 18.14 ndb-connectstring の型と値の情報

コマンド行	システム変数	ステータス変数
オプションファイル	スコープ	動的
開始バージョン	型	デフォルト、範囲
メモ		
ndb-connectstring		
はい	いいえ	いいえ
はい		いいえ
NDB 7.3	文字列	
説明: クラスタ構成を配布する管理サーバーへのポイント		

このオプションは、NDBCLUSTER ストレージエンジンの使用時に、クラスタ構成データを配信する管理サーバーを指定します。構文については、セクション18.3.2.3「MySQL Cluster の接続文字列」を参照してください。

- --ndb-deferred-constraints=[0|1]

表 18.15 ndb-deferred-constraints の型と値の情報

コマンド行	システム変数	ステータス変数
オプションファイル	スコープ	動的
開始バージョン	型	デフォルト、範囲
メモ		
ndb-deferred-constraints		
はい	はい	いいえ
はい	両方	はい
NDB 7.3	整数	0 / 0 - 1

コマンド行	システム変数	ステータス変数
オプションファイル	スコープ	動的
開始バージョン	型	デフォルト、範囲
メモ		
説明: 一意のインデックスに関する制約チェック (サポートされている場合) を、コミット時まで遅延させることを指定します。通常は必要なく、使用されません。テストの目的にのみ使用されます。		

一意のインデックスに対する制約チェックがサポートされている場合に、チェックをコミット時まで遅延するかどうかを制御します。0 がデフォルトです。

このオプションは、MySQL Cluster または MySQL Cluster レプリケーションの動作には通常不要であり、主にテストでの使用を目的としています。

- --ndb-distribution=[KEYHASH|LINHASH]

表 18.16 ndb-distribution の型と値の情報

コマンド行	システム変数	ステータス変数
オプションファイル	スコープ	動的
開始バージョン	型	デフォルト、範囲
メモ		
ndb-distribution		
はい	はい	いいえ
はい	グローバル	はい
NDB 7.3	列挙	KEYHASH / LINHASH, KEYHASH
説明: NDBCLUSTER の新しいテーブルのデフォルトの配布 (KEYHASH または LINHASH、デフォルトは KEYHASH)		

NDB テーブルのデフォルトの分配方法を制御します。KEYHASH (キーハッシュ) または LINHASH (線形ハッシュ) のいずれかに設定できます。KEYHASH がデフォルトです。

- --ndb-mgmd-host=host[:port]

表 18.17 ndb-mgmd-host の型と値の情報

コマンド行	システム変数	ステータス変数
オプションファイル	スコープ	動的
開始バージョン	型	デフォルト、範囲
メモ		
ndb-mgmd-host		
はい	いいえ	いいえ
はい		いいえ
NDB 7.3	文字列	localhost:1186
説明: 管理サーバーに接続するためのホスト (およびポート (必要な場合)) を設定します		

プログラムに接続される単一の管理サーバーのホストとポート番号の設定に使用できます。プログラムの接続情報として複数の管理サーバーのノード ID または参照 (あるいはその両方) が必要な場合は、代わりに --ndb-connectstring オプションを使用します。

- --ndbcluster

表 18.18 ndbcluster の型と値の情報

コマンド行	システム変数	ステータス変数
オプションファイル	スコープ	動的
開始バージョン	型	デフォルト、範囲
メモ		
ndbcluster		
はい	いいえ	いいえ
はい		いいえ
NDB 7.3	ブール	FALSE
説明: NDB Cluster を有効化 (このバージョンの MySQL がサポートしている場合) --skip-ndbcluster によって無効になります		

MySQL Cluster を使用するには、NDBCLUSTER ストレージエンジンが必要です。mysqld バイナリに NDBCLUSTER ストレージエンジンのサポートが含まれている場合、このエンジンはデフォルトで無効になっています。これを有効にするには、--ndbcluster オプションを使用します。エンジンを明示的に無効にするには、--skip-ndbcluster を使用します。

- --ndb-log-apply-status

表 18.19 ndb-log-apply-status の型と値の情報

コマンド行	システム変数	ステータス変数
オプションファイル	スコープ	動的
開始バージョン	型	デフォルト、範囲
メモ		
ndb-log-apply-status		
はい	はい	いいえ
はい	グローバル	いいえ
NDB 7.3	ブール	OFF
説明: スレーブとして機能している MySQL サーバーが、直接のマスターから受信した mysql.ndb_apply_status の更新を、自身のサーバー ID を使用して自身のバイナリログに記録するようになります。サーバーが --ndbcluster オプションで起動された場合にのみ有効です。		

スレーブの mysqld が、マスターのサーバー ID ではなく自身のサーバー ID を使用して、そのマスターから直接受信した更新を自身のバイナリログ内の mysql.ndb_apply_status テーブルに記録するようになります。循環またはチェーンレプリケーション設定では、これによって現在の mysqld のスレーブとして構成されているすべての MySQL サーバーの mysql.ndb_apply_status テーブルに更新が反映されます。

チェーンレプリケーションセットアップでこのオプションを使用すると、ダウンストリーム (スレーブ) クラスタがすべてのアップストリームコントリビュータ (マスター) との相対的な位置を認識できるようになります。

循環レプリケーションセットアップでこのオプションを使用すると、循環の全体が完成するように ndb_apply_status テーブルが変更され、最終的に発信元の MySQL Cluster に反映されます。これにより、マスターとして機能するクラスタも、循環内のほかのクラスタにその変更 (エポック) が適用されたことを認識できます。

このオプションは、MySQL サーバーが --ndbcluster オプションを使用して起動されていない場合は無効になります。

- --ndb-log-empty-epochs=[0|1]

表 18.20 ndb-log-empty-epochs の型と値の情報

コマンド行	システム変数	ステータス変数
オプションファイル	スコープ	動的
開始バージョン	型	デフォルト、範囲
メモ		
ndb-log-empty-epochs		
はい	はい	いいえ
はい	グローバル	はい
NDB 7.3	ブール	OFF
説明: 有効な場合は、--log-slave-updates が有効であっても、変更がなかったエポックが ndb_apply_status および ndb_binlog_index テーブルに書き込まれるようになります。		

--log-slave-updates が有効になっていても、変更がなかったエポックが ndb_apply_status および ndb_binlog_index テーブルに書き込まれるようになります。

このオプションはデフォルトで無効になっています。--ndb-log-empty-epochs を無効にすると、変更がなかったエポックトランザクションがバイナリログに書き込まれなくなりますが、ndb_binlog_index には空のエポックの行も書き込まれます。

--ndb-log-empty-epochs=1 にすると ndb_binlog_index テーブルのサイズがバイナリログのサイズとは関係なく増えるため、クラスタが大部分の時間アイドル状態であると予想される場合でも、ユーザーはこのテーブルの増大を管理できるように準備してください。

- --ndb-log-exclusive-reads=[0|1]

表 18.21 ndb-log-exclusive-reads の型と値の情報

コマンド行	システム変数	ステータス変数
オプションファイル	スコープ	動的
開始バージョン	型	デフォルト、範囲
メモ		
ndb-log-exclusive-reads		
はい	はい	いいえ
はい	両方	はい
NDB 7.3	ブール	0
説明: 排他的ロックによる主キーの読み取りをログに記録します。読み取りの競合に基づく競合解決を許可します。		

MySQL Cluster NDB 7.4.1 以降では、このオプションを使用してサーバーを起動すると、主キーの読み取りが排他ロックで記録され、読み取りの競合に基づく MySQL Cluster レプリケーションの競合検出および解決が可能になります。ndb_log_exclusive_reads システム変数の値を 1 または 0 に設定すると、このロックを実行時に有効または無効にすることもできます。0 (ロックを無効化) がデフォルトです。

詳細は、読み取り競合の検出と解決を参照してください。

- --ndb-log-orig

表 18.22 ndb-log-orig の型と値の情報

コマンド行	システム変数	ステータス変数
オプションファイル	スコープ	動的
開始バージョン	型	デフォルト、範囲
メモ		
ndb-log-orig		
はい	はい	いいえ
はい	グローバル	いいえ
NDB 7.3	ブール	OFF
説明: 発信サーバー ID とエポックを mysql.ndb_binlog_index テーブルに記録します。		

発信元サーバーの ID とエポックを ndb_binlog_index テーブルに記録します。

この場合、ndb_binlog_index 内には特定のエポックに対して複数の (発信元のエポックごとに 1 つの) 行が存在する可能性があります。

詳細は、セクション18.6.4「MySQL Cluster レプリケーションスキーマとテーブル」を参照してください。

- --ndb-log-transaction-id

表 18.23 ndb-log-transaction-id の型と値の情報

コマンド行	システム変数	ステータス変数
オプションファイル	スコープ	動的
開始バージョン	型	デフォルト、範囲
メモ		
ndb-log-transaction-id		
はい	はい	いいえ
はい	グローバル	いいえ
NDB 7.3	ブール	OFF
説明: NDB トランザクション ID をバイナリログに書き込みます。--log-bin-v1-events=OFF が必要です。		

スレーブの mysqld がバイナリログの各行に NDB トランザクション ID を書き込むようになります。このようなロギングでは、バイナリログのバージョン 2 のイベント形式を使用する必要があります。したがって、このオプションを使用するには --log-bin-use-v1-row-events を FALSE に設定する必要があります。

このオプションはメインラインの MySQL Server 5.6 ではサポートされません。NDB$EPOCH_TRANS() 関数を使用する MySQL Cluster レプリケーションの競合検出および解決を有効にする必要があります (NDB $EPOCH_TRANS()を参照してください)。

デフォルト値は FALSE です。

詳細は、セクション18.6.11「MySQL Cluster レプリケーションの競合解決」を参照してください。

- --ndb-nodeid=#

表 18.24 ndb-nodeid の型と値の情報

コマンド行	システム変数	ステータス変数
オプションファイル	スコープ	動的
開始バージョン	型	デフォルト、範囲
メモ		
ndb-nodeid		

コマンド行	システム変数	ステータス変数
オプションファイル	スコープ	動的
開始バージョン	型	デフォルト、範囲
メモ		
はい	いいえ	はい
はい	グローバル	いいえ
5.0.45	数値	/ 1 - 63
5.1.5	数値	/ 1 - 255
説明: この MySQL サーバーの MySQL Cluster ノード ID		

この MySQL サーバーの、MySQL Cluster 内でのノード ID を設定します。

--ndb-nodeid オプションは、--ndb-connectstring で設定されたノード ID を (2 つのオプションの使用順序に関係なく) オーバーライドします。

また、--ndb-nodeid を使用する場合は、config.ini の [mysqld] または [api] セクションに一致するノード ID が存在するか、このファイルに「オープンな」 [mysqld] または [api] セクション (つまり、NodeId または Id パラメータが指定されていないセクション) が存在する必要があります。これは、ノード ID が接続文字列の一部として指定されている場合にも当てはまります。

ノード ID は、その指定方法に関係なく、SHOW STATUS の出力にグローバルステータス変数 Ndb_cluster_node_id の値として表示され、SHOW ENGINE NDBCLUSTER STATUS の出力の connection 行に cluster_node_id として表示されます。

MySQL Cluster SQL ノードの ID の詳細は、セクション18.3.2.7「MySQL Cluster 内の SQL ノードおよびその他の API ノードの定義」を参照してください。

- --ndb_optimization_delay=milliseconds

表 18.25 ndb_optimization_delay の型と値の情報

コマンド行	システム変数	ステータス変数
オプションファイル	スコープ	動的
開始バージョン	型	デフォルト、範囲
メモ		
ndb_optimization_delay		
いいえ	はい	いいえ
いいえ	グローバル	はい
NDB 7.3	数値	10 / 0 - 100000
説明: NDB テーブルに対して OPTIMIZE TABLE で行のセットを処理する間に待機する時間をミリ秒で設定します。		

NDB テーブルに対する OPTIMIZE TABLE ステートメントによる行セット間で待機するミリ秒数を設定します。デフォルトは 10 です。

- --ndb-recv-thread-activation-threshold=threshold

表 18.26 ndb-recv-thread-activation-threshold の型と値の情報

コマンド行	システム変数	ステータス変数
オプションファイル	スコープ	動的
開始バージョン	型	デフォルト、範囲
メモ		
ndb-recv-thread-activation-threshold		
はい	いいえ	いいえ

コマンド行	システム変数	ステータス変数
オプションファイル	スコープ	動的
開始バージョン	型	デフォルト、範囲
メモ		
はい		いいえ
5.6.10-ndb-7.3.1	数値	8 / 0 (MIN_ACTIVATION_THRESHOLD) - 16 (MAX_ACTIVATION_THRESHOLD)
説明: 受信スレッドがクラスタ接続のポーリングを引き継ぐときのアクティブ化のしきい値 (同時にアクティブなスレッドで測定されます)		

現在アクティブなスレッドがこの数に達すると、受信スレッドがクラスタ接続のポーリングを引き継ぎます。

- --ndb-recv-thread-cpu-mask=bitmask

表 18.27 ndb-recv-thread-cpu-mask の型と値の情報

コマンド行	システム変数	ステータス変数
オプションファイル	スコープ	動的
開始バージョン	型	デフォルト、範囲
メモ		
ndb-recv-thread-cpu-mask		
はい	いいえ	いいえ
はい		いいえ
NDB 7.3	ビットマップ	[空]
説明: 受信者スレッドを特定の CPU にロックするための CPU マスク。16 進数として指定されます。詳細は、ドキュメントを参照してください。		

受信者スレッドを特定の CPU にロックする CPU マスクを設定します。これは、16 進数のビットマスクとして指定されます。たとえば、0x33 は受信者スレッドごとに 1 つの CPU を使用します。空の文字列 (受信者スレッドをロックしない) がデフォルトです。

- ndb-transid-mysql-connection-map=state

表 18.28 ndb-transid-mysql-connection-map の型と値の情報

コマンド行	システム変数	ステータス変数
オプションファイル	スコープ	動的
開始バージョン	型	デフォルト、範囲
メモ		
ndb-transid-mysql-connection-map		
はい	いいえ	いいえ
いいえ		いいえ
NDB 7.3	列挙	ON / ON、OFF、FORCE
説明: ndb_transid_mysql_connection_map プラグインを有効または無効にします。つまり、その名前を持つ INFORMATION_SCHEMA テーブルを有効または無効にします。		

INFORMATION_SCHEMA データベースの ndb_transid_mysql_connection_map テーブルを処理するプラグインを有効または無効にします。ON、OFF、FORCE のいずれかの値を取ります。ON (デフォルト) では、プラグインが有効になります。OFF では、プラグインが無効になり、ndb_transid_mysql_connection_map にアクセス

できなくなります。FORCE では、プラグインのロードと起動に失敗した場合に MySQL Server が起動しなくなります。

ndb_transid_mysql_connection_map テーブルプラグインが実行されているかどうかを確認するには、SHOW PLUGINS の出力をチェックします。

- --ndb-wait-connected=seconds

表 18.29 ndb-wait-connected の型と値の情報

コマンド行	システム変数	ステータス変数
オプションファイル	スコープ	動的
開始バージョン	型	デフォルト、範囲
メモ		
ndb-wait-connected		
はい	はい	いいえ
はい	グローバル	いいえ
NDB 7.3	数値	0 / 0 - 31536000
5.1.56-ndb-7.1.16、5.1.56-ndb-7.0.27	数値	30 / 0 - 31536000
説明: MySQL サーバーが、MySQL クライアントの接続を受け入れる前にクラスタ管理およびデータノードへの接続を待機する時間 (秒単位)。		

このオプションは、MySQL サーバーが MySQL クライアントの接続を受け入れる前に MySQL Cluster 管理およびデータノードへの接続が確立されるまで待機する時間を設定します。この時間は秒単位で指定します。デフォルト値は 30 です。

- --ndb-wait-setup=seconds

表 18.30 ndb-wait-setup の型と値の情報

コマンド行	システム変数	ステータス変数
オプションファイル	スコープ	動的
開始バージョン	型	デフォルト、範囲
メモ		
ndb-wait-setup		
はい	はい	いいえ
はい	グローバル	いいえ
NDB 7.3	数値	15 / 0 - 31536000
5.1.56-ndb-7.0.27, 5.1.56-ndb-7.1.16	数値	30 / 0 - 31536000
説明: MySQL サーバーが NDB エンジンのセットアップの完了を待機する時間 (秒単位)。		

この変数は、MySQL サーバーが NDB ストレージエンジンのセットアップが完了されるまで待機しタイムアウトして、NDB を使用不可として扱うまでの時間を示します。この時間は秒単位で指定します。デフォルト値は 30 です。

- --server-id-bits=#

表 18.31 server-id-bits の型と値の情報

コマンド行	システム変数	ステータス変数
オプションファイル	スコープ	動的
開始バージョン	型	デフォルト、範囲
メモ		
server-id-bits		
はい	はい	いいえ
はい	グローバル	いいえ
NDB 7.3	数値	32 / 7 - 32
説明: サーバーを識別するために実際に使用される server_id 内の最下位ビットの数を設定することにより、NDB API アプリケーションが最上位ビットにアプリケーションデータを格納できるようにします。server_id は 2 の「この値」乗未満である必要があります。		

このオプションは、32 ビットの server_id のうち、実際にサーバーを識別する最小有効ビットの数を示します。サーバーが実際には 32 未満のビット数で識別されることを示すことで、残りのビットの一部をほかの目的に使用できるようになります。たとえば、NDB API のイベント API を使用するアプリケーションによって生成される OperationOptions 構造体の AnyValue 内のユーザーデータを格納できます (MySQL Cluster は AnyValue を使用してサーバー ID を格納します)。

サーバーは、レプリケーションループの検出などで server_id から有効なサーバー ID を抽出するときに、残りのビットを無視します。--server-id-bits オプションは、I/O および SQL スレッド内の server_id に基づいてイベントを無視すべきかどうかを決定するときに、サーバー ID の無関係なビットをマスクして除外するために使用されます。

このデータは、mysqlbinlog によってバイナリログから読み取ることができます (それ自体の --server-id-bits オプションを 32 (デフォルト) に設定して実行することが条件になります)。

server_id の値は、2 ^ server_id_bits 未満である必要があります。それ以外の場合、mysqld により起動が拒否されます。

このシステム変数は MySQL Cluster でのみサポートされます。標準の MySQL 5.6 Server ではサポートされません。

- --skip-ndbcluster

表 18.32 skip-ndbcluster の型と値の情報

コマンド行	システム変数	ステータス変数
オプションファイル	スコープ	動的
開始バージョン	型	デフォルト、範囲
メモ		
skip-ndbcluster		
はい	いいえ	いいえ
はい		いいえ
説明: NDB Cluster ストレージエンジンを無効化		

NDBCLUSTER ストレージエンジンを無効にします。これは、NDBCLUSTER ストレージエンジンのサポート付きでビルドされたバイナリのデフォルトです。サーバーは、--ndbcluster オプションが明示的に指定された場合にのみ、このストレージエンジン用にメモリーおよびその他のリソースを割り当てます。例については、セクション18.3.1「MySQL Cluster の簡易テストセットアップ」を参照してください。

18.3.4.3 MySQL Cluster のシステム変数

このセクションでは、MySQL Cluster および NDB ストレージエンジンに固有の MySQL サーバーシステム変数について詳しく説明します。MySQL Cluster に固有でないシステム変数については、セクション5.1.4「サーバーシ

ステム変数」を参照してください。システム変数の使用に関する一般情報は、セクション5.1.5「システム変数の使用」を参照してください。

- have_ndbcluster

  表 18.33 have_ndbcluster の型と値の情報

コマンド行	システム変数	ステータス変数
オプションファイル	スコープ	動的
開始バージョン	型	デフォルト、範囲
メモ		
have_ndbcluster		
いいえ	はい	いいえ
いいえ	グローバル	いいえ
NDB 7.3	ブール	
説明: mysqld が NDB Cluster テーブルをサポートするかどうか (--ndbcluster オプションで設定されます)。		

  mysqld が NDBCLUSTER テーブルをサポートする場合は YES。--skip-ndbcluster が使用されている場合は DISABLED。

  この変数は非推奨であり、MySQL 5.6 で削除されています。代わりに SHOW ENGINES を使用してください。

- ndb_autoincrement_prefetch_sz

  表 18.34 ndb_autoincrement_prefetch_sz の型と値の情報

コマンド行	システム変数	ステータス変数
オプションファイル	スコープ	動的
開始バージョン	型	デフォルト、範囲
メモ		
ndb_autoincrement_prefetch_sz		
はい	はい	いいえ
はい	両方	はい
NDB 7.3	数値	32 / 1 - 256
5.0.56	数値	1 / 1 - 256
5.1.1	数値	32 / 1 - 256
5.1.23	数値	1 / 1 - 256
5.1.16-ndb-6.2.0	数値	32 / 1 - 256
5.1.23-ndb-6.2.10	数値	1 / 1 - 256
5.1.19-ndb-6.3.0	数値	32 / 1 - 256
5.1.23-ndb-6.3.7	数値	1 / 1 - 256
5.1.41-ndb-6.3.31	数値	1 / 1 - 65536
5.1.30-ndb-6.4.0	数値	32 / 1 - 256
5.1.41-ndb-7.0.11	数値	1 / 1 - 65536
5.5.15-ndb-7.2.1	数値	1 / 1 - 65536
説明: NDB 自動インクリメント値のプリフェッチサイズ		

  自動インクリメントカラムのギャップの可能性を指定します。これを最小化するには、1 に設定します。最適化するために大きな値に設定すると、挿入は高速になりますが、一連の挿入に連続した自動インクリメント番号が使用される可能性は減ります。最小およびデフォルト値は 1 です。ndb_autoincrement_prefetch_sz の最大値は 65536 です。

  この変数は、ステートメント間でフェッチされる AUTO_INCREMENT ID の数にのみ影響します。特定のステートメント内では、一度に少なくとも 32 個の ID が取得されます。デフォルト値は 1 です。

> **重要**
>
> この変数は、INSERT ... SELECT を使用して実行される挿入には影響しません。

- ndb_cache_check_time

表 18.35 ndb_cache_check_time の型と値の情報

コマンド行	システム変数	ステータス変数
オプションファイル	スコープ	動的
開始バージョン	型	デフォルト、範囲
メモ		
ndb_cache_check_time		
はい	はい	いいえ
はい	グローバル	はい
NDB 7.3	数値	0 / -
説明: MySQL クエリーキャッシュによって行われるクラスタ SQL ノードのチェック間のミリ秒数		

MySQL クエリーキャッシュによる MySQL Cluster SQL ノードのチェック間で経過するミリ秒数。これを 0 (デフォルトおよび最小値) に設定すると、クエリーキャッシュですべてのクエリーの妥当性がチェックされます。

この変数の推奨最大値は 1000 です。これは、チェックが毎秒 1 回行われることを意味します。より大きな値にすると、チェックが実行され、場合によっては異なる SQL ノードでの更新のために無効になる頻度が少なくなります。これを 2000 より大きな値に設定することは、通常、望ましくありません。

- ndb_deferred_constraints

表 18.36 ndb_deferred_constraints の型と値の情報

コマンド行	システム変数	ステータス変数
オプションファイル	スコープ	動的
開始バージョン	型	デフォルト、範囲
メモ		
ndb_deferred_constraints		
はい	はい	いいえ
はい	両方	はい
NDB 7.3	整数	0 / 0 - 1
説明: 制約チェック (サポートされている場合) を遅延させることを指定します。通常は必要なく、使用されません。テストの目的にのみ使用されます。		

制約チェックがサポートされる場合に、それを遅延するかどうかを制御します。0 がデフォルトです。

この変数は、MySQL Cluster または MySQL Cluster レプリケーションの動作には通常不要であり、主にテストでの使用を目的としています。

- ndb_distribution

表 18.37 ndb_distribution の型と値の情報

コマンド行	システム変数	ステータス変数
オプションファイル	スコープ	動的
開始バージョン	型	デフォルト、範囲
メモ		
ndb_distribution		
はい	はい	いいえ

コマンド行	システム変数	ステータス変数
オプションファイル	スコープ	動的
開始バージョン	型	デフォルト、範囲
メモ		
はい	グローバル	はい
NDB 7.3	列挙	KEYHASH / LINHASH, KEYHASH
説明: NDBCLUSTER の新しいテーブルのデフォルトの配布 (KEYHASH または LINHASH、デフォルトは KEYHASH)		

NDB テーブルのデフォルトの分配方法を制御します。KEYHASH (キーハッシュ) または LINHASH (線形ハッシュ) のいずれかに設定できます。KEYHASH がデフォルトです。

- ndb_eventbuffer_max_alloc

表 18.38 ndb_eventbuffer_max_alloc の型と値の情報

コマンド行	システム変数	ステータス変数
オプションファイル	スコープ	動的
開始バージョン	型	デフォルト、範囲
メモ		
ndb_eventbuffer_max_alloc		
はい	はい	いいえ
はい	グローバル	はい
NDB 7.3	数値	0 / 0 - 4294967295
説明: NDB API によるイベントのバッファリングのために割り当てることのできる最大メモリー。デフォルトでは 0 (制限なし) になります。		

NDB API によるイベントのバッファリングに割り当てる可能な最大メモリー量 (バイト単位) を設定します。0 はデフォルトで、制限が適用されません。

この変数は MySQL Cluster NDB 7.3.3 で追加されました。

- ndb_extra_logging

表 18.39 ndb_extra_logging の型と値の情報

コマンド行	システム変数	ステータス変数
オプションファイル	スコープ	動的
開始バージョン	型	デフォルト、範囲
メモ		
ndb_extra_logging		
はい	はい	いいえ
はい	グローバル	はい
NDB 7.3	数値	0 / -
5.1.19-ndb-6.3.0	数値	1 / -
説明: MySQL Cluster スキーマ、接続、およびデータ配布イベントの、MySQL エラーログへのロギングを制御		

この変数は、NDB ストレージエンジンに固有の情報を MySQL エラーログに記録できるようにします。

この変数を 0 に設定すると、MySQL エラーログに書き込まれる NDB 固有の情報がトランザクション処理に関するものだけになります。0 より大きく 10 より小さい値に設定すると、NDB のテーブルスキーマイベントと接続イベント、競合解決の使用中の有無、および NDB のその他のエラーと情報も記録されます。値を 10 以上に設定すると、NDB の内部に関する情報 (クラスタノード間のデータ分配の進行状況など) も MySQL エラーログに書き込まれます。デフォルトは 1 です。

- ndb_force_send

表 18.40 ndb_force_send の型と値の情報

コマンド行	システム変数	ステータス変数
オプションファイル	スコープ	動的
開始バージョン	型	デフォルト、範囲
メモ		
ndb_force_send		
はい	はい	いいえ
はい	両方	はい
NDB 7.3	ブール	TRUE
説明: ほかのスレッドを待機することなく、ただちにバッファーを NDB に強制的に送信		

ほかのスレッドを待機せずに、バッファーを NDB にただちに送信します。デフォルトは ON です。

- ndb_index_stat_cache_entries

表 18.41 ndb_index_stat_cache_entries の型と値の情報

コマンド行	システム変数	ステータス変数
オプションファイル	スコープ	動的
開始バージョン	型	デフォルト、範囲
メモ		
ndb_index_stat_cache_entries		
はい	はい	いいえ
はい	両方	はい
NDB 7.3	数値	32 / 0 - 4294967295
説明: 開始キーと終了キーの数を決定することにより、統計の粒度を設定		

統計メモリーキャッシュに格納する開始キーと終了キーの数を指定して、統計の粒度を設定します。0 では、キャッシュが行われません。この場合、データノードは常に直接クエリーされます。デフォルト値: 32。

> **注記**
>
> ndb_index_stat_enable が OFF の場合は、この変数を設定しても無効になります。

この変数は MySQL 5.1 で非推奨になり、MySQL Cluster NDB 7.3.5 以降では削除されています。

- ndb_index_stat_enable

表 18.42 ndb_index_stat_enable の型と値の情報

コマンド行	システム変数	ステータス変数
オプションファイル	スコープ	動的
開始バージョン	型	デフォルト、範囲
メモ		
ndb_index_stat_enable		
はい	はい	いいえ
はい	両方	はい
NDB 7.3	ブール	OFF
5.5.15-ndb-7.2.1	ブール	ON

コマンド行	システム変数	ステータス変数
オプションファイル	スコープ	動的
開始バージョン	型	デフォルト、範囲
メモ		
説明: クエリーの最適化で NDB インデックス統計を使用		

クエリーの最適化で NDB インデックス統計を使用します。デフォルトは ON です。

- ndb_index_stat_option

表 18.43 ndb_index_stat_option の型と値の情報

コマンド行	システム変数	ステータス変数
オプションファイル	スコープ	動的
開始バージョン	型	デフォルト、範囲
メモ		
ndb_index_stat_option		
はい	はい	いいえ
はい	両方	はい
NDB 7.3	文字列	loop_enable=1000ms,loop_idle=1000ms,loop_ update_batch=1,read_batch=4,idle_batch=32,c check_delay=10m,delete_batch=8, clean_delay=1m,error_batch=4, error_delay=1m,evict_batch=8,evict_delay=1m, cache_lowpct=90,zero_total=0
5.1.56-ndb-7.1.17	文字列	loop_checkon=1000ms,loop_idle=1000ms,loop update_batch=1,read_batch=4,idle_batch=32,c check_delay=1m,delete_batch=8,clean_delay= error_delay=1m,evict_batch=8,evict_delay=1m, cache_lowpct=90
説明: NDB インデックス統計の調整可能なオプションのカンマ区切りのリスト。リストはスペースを含んではなりません		

この変数は、NDB インデックス統計の生成に関する調整オプションを提供するために使用されます。このリストは、カンマで区切られたオプション名と値の名前/値ペアで構成されます。このリストに空白文字を入れることはできません。

ndb_index_stat_option の設定時に使用していなかったオプションは、デフォルト値から変更できません。たとえば、ndb_index_stat_option = 'loop_idle=1000ms,cache_limit=32M' のように設定できます。

オプションで、時間値のサフィクスとして s (秒) または ms (ミリ秒) を使用できます。整数値のサフィクスとして K、M、または G を使用できます。

この変数を使用して設定できるオプションの名前を次の表に示します。この表には、オプションの簡単な説明、デフォルト値、および (適用可能な場合は) 最小値と最大値も示します。

名前	説明	デフォルト/単位	最小/最大
loop_enable		1000 ms	0/4G
loop_idle	アイドル時のスリープ時間	1000 ms	0/4G
loop_busy	追加作業が待機しているときのスリープ時間	100 ms	0/4G
update_batch		1	0/4G
read_batch		4	1/4G
idle_batch		32	1/4G
check_batch		8	1/4G

名前	説明	デフォルト/単位	最小/最大
check_delay	新しい統計をチェックする頻度	10 ms	1/4G
delete_batch		8	0/4G
clean_delay		1 ms	0/4G
error_batch		4	1/4G
error_delay		1 ms	1/4G
evict_batch		8	1/4G
evict_delay	LRU キャッシュを削除します (読み取り時から)	1 ms	0/4G
cache_limit	この mysqld がキャッシュされたインデックス統計のために使用するメモリーの最大量 (バイト単位)。これを超えたときにキャッシュをクリーンアップします。	32 M	0/4G
cache_lowpct		90	0/100
zero_total	これを 1 に設定すると、ndb_index_stat_status のすべての累積カウンタが 0 にリセットされます。これが行われると、このオプション値も 0 にリセットされます。	0	0/1

- ndb_index_stat_update_freq

表 18.44 ndb_index_stat_update_freq の型と値の情報

コマンド行	システム変数	ステータス変数
オプションファイル	スコープ	動的
開始バージョン	型	デフォルト、範囲
メモ		
ndb_index_stat_update_freq		
はい	はい	いいえ
はい	両方	はい
NDB 7.3	数値	20 / 0 - 4294967295
説明: 統計キャッシュではなくデータノードにクエリーを実行する頻度		

統計キャッシュの代わりにデータノードをクエリーする頻度。たとえば、値 20 (デフォルト) ではクエリーを 20 回に 1 回データノードに送信します。

> **注記**
>
> ndb_index_stat_cache_entries が 0 の場合は、この変数を設定しても無効になります。この場合は、すべてのクエリーがデータノードに直接送信されます。

この変数は MySQL 5.1 で非推奨になり、MySQL Cluster NDB 7.3.5 以降では削除されています。

- ndb_join_pushdown

表 18.45 ndb_join_pushdown の型と値の情報

コマンド行	システム変数	ステータス変数
オプションファイル	スコープ	動的
開始バージョン	型	デフォルト、範囲
メモ		
ndb_join_pushdown		
いいえ	はい	いいえ
いいえ	両方	はい
5.1.51-ndb-7.2.0	ブール	TRUE
説明: データノードへの結合のプッシュダウンを有効化		

この変数は、NDB テーブル上の結合が NDB カーネル (データノード) にプッシュダウンされるかどうかを制御します。以前は、SQL ノードによる NDB の複数アクセスにより結合が処理されていましたが、ndb_join_pushdown を有効にすると、プッシュ可能な結合のすべてがデータノードに送信されます。この場合、結合はデータノード間で分配され、データの複数のコピーに対して並列で実行され、単一のマージされた結果が mysqld に返されます。これにより、このような結合を処理するために必要な SQL ノードとデータノード間のラウンドトリップの回数を大幅に減らすことができます。

デフォルトでは、ndb_join_pushdown は有効になっています。

結合をプッシュ可能にするには、次の条件を満たす必要があります。

1. 比較できるのはカラムのみであり、結合されるすべてのカラムが完全に同じデータ型を使用している必要があります。

   つまり、t1.a = t2.a + constant のような式はプッシュダウンできません。また、(たとえば) INT カラムと BIGINT カラムの結合をプッシュダウンすることもできません。

2. BLOB または TEXT カラムを参照するクエリーはサポートされません。

3. 明示的なロックはサポートされません。ただし、NDB ストレージエンジンの特徴である暗黙的な行ベースのロックは適用されます。

   これは、FOR UPDATE を使用する結合をプッシュダウンできないことを意味します。

4. 結合をプッシュダウンするには、ref、eq_ref、または const アクセスメソッドのいずれか、またはこれらのメソッドの組み合わせを使用して結合の子テーブルにアクセスする必要があります。

   外部結合された子テーブルは、eq_ref のみを使用してプッシュできます。

   プッシュされた結合のルートが eq_ref または const である場合は、eq_ref によって結合された子テーブルのみを追加できます。(ref によって結合されたテーブルは、プッシュされた結合の別のルートになる可能性があります。)

   クエリーオプティマイザで候補となる子テーブルに Using join cache を指定した場合は、そのテーブルは子としてプッシュできません。ただし、プッシュされたテーブルの別のセットのルートになることはできます。

5. [LINEAR] HASH、LIST、または RANGE によって明示的にパーティション化されたテーブルを参照する結合は、現在のところプッシュダウンできません。

特定の結合をプッシュダウンできるかどうかを確認するには、それを EXPLAIN でチェックします。結合をプッシュダウンできる場合は、この例に示すように、出力の Extra カラムに pushed join への参照が表示されます。

```
mysql> EXPLAIN
    ->     SELECT e.first_name, e.last_name, t.title, d.dept_name
    ->         FROM employees e
    ->         JOIN dept_emp de ON e.emp_no=de.emp_no
    ->         JOIN departments d ON d.dept_no=de.dept_no
    ->         JOIN titles t ON e.emp_no=t.emp_no\G
*************************** 1. row ***************************
           id: 1
```

```
  select_type: SIMPLE
        table: d
         type: ALL
possible_keys: PRIMARY
          key: NULL
      key_len: NULL
          ref: NULL
         rows: 9
        Extra: Parent of 4 pushed join@1
*************************** 2. row ***************************
           id: 1
  select_type: SIMPLE
        table: de
         type: ref
possible_keys: PRIMARY,emp_no,dept_no
          key: dept_no
      key_len: 4
          ref: employees.d.dept_no
         rows: 5305
        Extra: Child of 'd' in pushed join@1
*************************** 3. row ***************************
           id: 1
  select_type: SIMPLE
        table: e
         type: eq_ref
possible_keys: PRIMARY
          key: PRIMARY
      key_len: 4
          ref: employees.de.emp_no
         rows: 1
        Extra: Child of 'de' in pushed join@1
*************************** 4. row ***************************
           id: 1
  select_type: SIMPLE
        table: t
         type: ref
possible_keys: PRIMARY,emp_no
          key: emp_no
      key_len: 4
          ref: employees.de.emp_no
         rows: 19
        Extra: Child of 'e' in pushed join@1
4 rows in set (0.00 sec)
```

> **注記**
>
> 内部結合されたテーブルが ref によって結合され、かつその結果がソートされたインデックスによって順序付けまたはグループ化された場合、このインデックスはソートされた行を提供できず、ソートされた一時ファイルへの書き込みが強制されます。

プッシュされた結合の動作については、追加の情報源が 2 つあります。

1. ステータス変数 Ndb_pushed_queries_defined、Ndb_pushed_queries_dropped、Ndb_pushed_queries_executed、および Ndb_pushed_reads。
2. DBSPJ カーネルブロックに属する ndbinfo.counters テーブル内のカウンタ。このカウンタについては、セクション18.5.10.7「ndbinfo counters テーブル」を参照してください。『MySQL Cluster API Developer Guide』のThe DBSPJ Blockも参照してください。

- ndb_log_apply_status

表 18.46 ndb_log_apply_status の型と値の情報

コマンド行	システム変数	ステータス変数
オプションファイル	スコープ	動的
開始バージョン	型	デフォルト、範囲
メモ		
ndb_log_apply_status		
はい	はい	いいえ
はい	グローバル	いいえ

コマンド行	システム変数	ステータス変数
オプションファイル	スコープ	動的
開始バージョン	型	デフォルト、範囲
メモ		
NDB 7.3	ブール	OFF
説明: スレーブとして機能している MySQL サーバーが、直接のマスターから受信した mysql.ndb_apply_status の更新を、自身のサーバー ID を使用して自身のバイナリログに記録するかどうか。		

--ndb-log-apply-status オプションを使用してサーバーが起動されたかどうかを示す読み取り専用の変数。

- ndb_log_bin

表 18.47 ndb_log_bin の型と値の情報

コマンド行	システム変数	ステータス変数
オプションファイル	スコープ	動的
開始バージョン	型	デフォルト、範囲
メモ		
ndb_log_bin		
はい	はい	いいえ
いいえ	両方	はい
NDB 7.3	ブール	ON
説明: NDB テーブルの更新をバイナリログに書き込みます。--log-bin でバイナリロギングが有効になっている場合にのみ有効です。		

NDB テーブルの更新がバイナリログに書き込まれるようになります。サーバーのバイナリロギングがまだ log_bin を使用して有効になっていない場合は、この変数を設定しても無効になります。ndb_log_bin のデフォルトは 1 (ON) です。通常、本番環境でこの値を変更する必要はありません。

- ndb_log_binlog_index

表 18.48 ndb_log_binlog_index の型と値の情報

コマンド行	システム変数	ステータス変数
オプションファイル	スコープ	動的
開始バージョン	型	デフォルト、範囲
メモ		
ndb_log_binlog_index		
はい	はい	いいえ
いいえ	グローバル	はい
NDB 7.3	ブール	ON
説明: エポックとバイナリログの位置の間のマッピングを ndb_binlog_index テーブルに挿入します。デフォルトでは ON になります。サーバー上でバイナリロギングが有効になっている場合にのみ有効です。		

エポックとバイナリログ内の位置とのマッピングが ndb_binlog_index テーブルに挿入されるようになります。サーバーのバイナリロギングがまだ log_bin を使用して有効になっていない場合は、この変数を設定しても無効になります。(また、ndb_log_bin も無効にしないでください。) ndb_log_binlog_index のデフォルトは 1 (ON) です。通常、本番環境でこの値を変更する必要はありません。

- ndb_log_empty_epochs

表 18.49 ndb_log_empty_epochs の型と値の情報

コマンド行	システム変数	ステータス変数
オプションファイル	スコープ	動的
開始バージョン	型	デフォルト、範囲
メモ		
ndb_log_empty_epochs		
はい	はい	いいえ
はい	グローバル	はい
NDB 7.3	ブール	OFF
説明: 有効な場合は、log_slave_updates が有効であっても、変更がなかったエポックが ndb_apply_status および ndb_binlog_index テーブルに書き込まれます。		

この変数を 0 に設定すると、変更がなかったエポックトランザクションがバイナリログに書き込まれなくなりますが、ndb_binlog_index には空のエポックの行も書き込まれます。

- ndb_log_exclusive_reads

表 18.50 ndb_log_exclusive_reads の型と値の情報

コマンド行	システム変数	ステータス変数
オプションファイル	スコープ	動的
開始バージョン	型	デフォルト、範囲
メモ		
ndb_log_exclusive_reads		
はい	はい	いいえ
はい	両方	はい
NDB 7.3	ブール	0
説明: 排他的ロックによる主キーの読み取りをログに記録します。読み取りの競合に基づく競合解決を許可します。		

MySQL Cluster NDB 7.4.1 以降では、この変数によって、主キーの読み取りを排他ロックで記録するかどうかを指定します。これにより、読み取りの競合に基づく MySQL Cluster レプリケーションの競合検出および解決が可能になります。このロックを有効にするには、ndb_log_exclusive_reads の値を 1 に設定します。0 (このようなロックを無効にする) がデフォルトです。

詳細は、読み取り競合の検出と解決を参照してください。

- ndb_log_orig

表 18.51 ndb_log_orig の型と値の情報

コマンド行	システム変数	ステータス変数
オプションファイル	スコープ	動的
開始バージョン	型	デフォルト、範囲
メモ		
ndb_log_orig		
はい	はい	いいえ
はい	グローバル	いいえ
NDB 7.3	ブール	OFF

コマンド行	システム変数	ステータス変数
オプションファイル	スコープ	動的
開始バージョン	型	デフォルト、範囲
メモ		
説明: 発信サーバーの ID とエポックが mysql.ndb_binlog_index テーブルに記録されるかどうか。mysqld を起動するときに --ndb-log-orig オプションを使用して設定します。		

発信元サーバーの ID とエポックが ndb_binlog_index テーブルに記録されるかどうかを示します。--ndb-log-orig サーバーオプションを使用して設定します。

- ndb_log_transaction_id

表 18.52 ndb_log_transaction_id の型と値の情報

コマンド行	システム変数	ステータス変数
オプションファイル	スコープ	動的
開始バージョン	型	デフォルト、範囲
メモ		
ndb_log_transaction_id		
いいえ	はい	いいえ
いいえ	グローバル	いいえ
NDB 7.3	ブール	OFF
説明: NDB トランザクション ID がバイナリログに書き込まれるかどうか。(読み取り専用。)		

この読み取り専用のブールシステム変数は、スレーブの mysqld が (「アクティブ-アクティブ」型の MySQL Cluster レプリケーションを NDB$EPOCH_TRANS() の競合検出とともに使用するために必要な) NDB トランザクション ID をバイナリログに書き込むかどうかを示します。この設定を変更するには、--ndb-log-transaction-id オプションを使用します。

メインラインの MySQL Server 5.6 では ndb_log_transaction_id はサポートされません。

詳細は、セクション18.6.11「MySQL Cluster レプリケーションの競合解決」を参照してください。

- ndb_optimized_node_selection

表 18.53 ndb_optimized_node_selection の型と値の情報

コマンド行	システム変数	ステータス変数
オプションファイル	スコープ	動的
開始バージョン	型	デフォルト、範囲
メモ		
ndb_optimized_node_selection		
はい	はい	いいえ
はい	グローバル	いいえ
NDB 7.3	ブール	ON
5.1.22-ndb-6.3.4	数値	3 / 0 - 3
説明: SQL ノードが、トランザクションコーディネータとして使用するためにクラスタデータノードを選択する方法を決定		

最適化されたノードの選択には、ここの示す 2 つの形式があります。

1. SQL ノードは、近接性を使用してトランザクションコーディネータを指定します。つまり、その SQL ノードから「もっとも近い」データノードがトランザクションコーディネータとして選択されます。この指定では、SQL ノードとの共有メモリー接続を持つデータノードがその SQL ノードから「もっとも近い」とみなされます。次に近いのは、(近接性が低くなる順に) localhost との TCP 接続、SCI 接続、localhost 以外のホストからの TCP 接続です。

2. SQL スレッドは、分布の認識を使用してデータノードを選択します。つまり、特定のトランザクションの最初のステートメントによってアクセスされるクラスタパーティションを収容するデータノードが、トランザクション全体のトランザクションコーディネータとして使用されます。(これは、トランザクションの最初のステートメントが 1 つのクラスタパーティションにしかアクセスしない場合にのみ有効です。)

このオプションは、0、1、2、または 3 のいずれかの整数値を取ります。3 がデフォルトです。これらの値は、ノード選択に次のように影響します。

- 0: ノード選択は最適化されません。SQL スレッドが次のデータノードに進む前に、各データノードがトランザクションコーディネータとして 8 回ずつ採用されます。
- 1: SQL ノードへの近接性を使用してトランザクションコーディネータが指定されます。
- 2: 分布の認識を使用してトランザクションコーディネータが選択されます。ただし、トランザクションの最初のステートメントが複数のクラスタパーティションにアクセスする場合は、SQL ノードはこのオプションを 0 に設定したときに見られるラウンドロビンの動作に戻ります。
- 3: トランザクションコーディネータを指定するために分布の認識を使用できる場合は、それが使用されます。そうでない場合は、近接性を使用してトランザクションコーディネータが選択されます。(これはデフォルトの動作です。)

- ndb_recv_thread_activation_threshold

表 18.54 ndb_recv_thread_activation_threshold の型と値の情報

コマンド行	システム変数	ステータス変数
オプションファイル	スコープ	動的
開始バージョン	型	デフォルト、範囲
メモ		
ndb_recv_thread_activation_threshold		
いいえ	いいえ	いいえ
いいえ		いいえ
5.6.10-ndb-7.3.1	数値	8 / 0 (MIN_ACTIVATION_THRESHOLD) - 16 (MAX_ACTIVATION_THRESHOLD)
説明: 受信スレッドがクラスタ接続のポーリングを引き継ぐときのアクティブ化のしきい値 (同時にアクティブなスレッドで測定されます)		

現在アクティブなスレッドがこの数に達すると、受信スレッドがクラスタ接続のポーリングを引き継ぎます。

この変数のスコープはグローバルです。また、--ndb-recv-thread-activation-threshold オプションを使用して起動時に設定することもできます。

- ndb_recv_thread_cpu_mask

表 18.55 ndb_recv_thread_cpu_mask の型と値の情報

コマンド行	システム変数	ステータス変数
オプションファイル	スコープ	動的
開始バージョン	型	デフォルト、範囲
メモ		
ndb_recv_thread_cpu_mask		
いいえ	はい	いいえ
いいえ	グローバル	はい
NDB 7.3	ビットマップ	[空]
説明: 受信者スレッドを特定の CPU にロックするための CPU マスク。16 進数として指定されます。詳細は、ドキュメントを参照してください。		

受信者スレッドを特定の CPU にロックするための CPU マスク。これは、16 進数のビットマスクとして指定されます。たとえば、0x33 は受信者スレッドごとに 1 つの CPU を使用します。空の文字列がデフォルトです。ndb_recv_thread_cpu_mask をこの値に設定すると、以前に設定された受信者スレッドのロックがすべて削除されます。

この変数のスコープはグローバルです。また、--ndb-recv-thread-cpu-mask オプションを使用して起動時に設定することもできます。

- ndb_report_thresh_binlog_epoch_slip

表 18.56 ndb_report_thresh_binlog_epoch_slip の型と値の情報

コマンド行	システム変数	ステータス変数
オプションファイル	スコープ	動的
開始バージョン	型	デフォルト、範囲
メモ		
ndb_report_thresh_binlog_epoch_slip		
はい	いいえ	いいえ
はい		いいえ
NDB 7.3	数値	3 / 0 - 256
説明: これは、バイナリログステータスをレポートする前に、過ぎていなければならないエポックの数のしきい値です		

これは、バイナリログのステータスを報告するまでに遅延するエポックの数に対するしきい値です。たとえば、値 3 (デフォルト) では、ストレージノードから受信したエポックとバイナリログに適用されたエポックの差が 3 以上になると、ステータスメッセージがクラスタログに送信されます。

- ndb_report_thresh_binlog_mem_usage

表 18.57 ndb_report_thresh_binlog_mem_usage の型と値の情報

コマンド行	システム変数	ステータス変数
オプションファイル	スコープ	動的
開始バージョン	型	デフォルト、範囲
メモ		
ndb_report_thresh_binlog_mem_usage		
はい	いいえ	いいえ
はい		いいえ
NDB 7.3	数値	10 / 0 - 10
説明: これは、バイナリログステータスをレポートする前に、残っている空きメモリーの割合のしきい値です。		

これは、バイナリログのステータスを報告するまでに残存する空きメモリーの割合に対するしきい値です。たとえば、値 10 (デフォルト) では、データノードからのバイナリログデータの受信に使用できるメモリー量が 10% を下回ると、ステータスメッセージがクラスタログに送信されます。

- slave_allow_batching

表 18.58 slave_allow_batching の型と値の情報

コマンド行	システム変数	ステータス変数
オプションファイル	スコープ	動的
開始バージョン	型	デフォルト、範囲
メモ		
slave_allow_batching		

コマンド行	システム変数	ステータス変数
オプションファイル	スコープ	動的
開始バージョン	型	デフォルト、範囲
メモ		
はい	はい	いいえ
はい	グローバル	はい
NDB 7.3	ブール	off
説明: レプリケーションスレーブの更新バッチ化をオンおよびオフにする		

MySQL Cluster レプリケーションのスレーブでバッチ更新が有効かどうか。

現在、この変数は MySQL Cluster に付属する、または MySQL Cluster ソースからビルドされた mysqld でのみ使用可能です。詳細は、セクション18.6.6「MySQL Cluster レプリケーションの起動 (レプリケーションチャネルが 1 つ)」を参照してください。

- ndb_show_foreign_key_mock_tables

表 18.59 ndb_show_foreign_key_mock_tables の型と値の情報

コマンド行	システム変数	ステータス変数
オプションファイル	スコープ	動的
開始バージョン	型	デフォルト、範囲
メモ		
ndb_show_foreign_key_mock_tables		
はい	はい	いいえ
はい	グローバル	はい
NDB 7.3	ブール	OFF
説明: foreign_key_checks=0 をサポートするために使用される擬似テーブルを表示します。		

NDB が foreign_key_checks=0 のサポートに使用するモックテーブルを示します。これを有効にすると、テーブルを作成および削除するときに追加の警告が表示されます。このテーブルの実際の (内部の) 名前は、SHOW CREATE TABLE の出力に表示されます。

- ndb_slave_conflict_role

表 18.60 ndb_slave_conflict_role の型と値の情報

コマンド行	システム変数	ステータス変数
オプションファイル	スコープ	動的
開始バージョン	型	デフォルト、範囲
メモ		
ndb_slave_conflict_role		
はい	はい	いいえ
はい	グローバル	はい
NDB 7.3	列挙	NONE / NONE、PRIMARY、SECONDARY、PASS
説明: スレーブが、競合の検出と解決で果たすロール。値は PRIMARY、SECONDARY、PASS、NONE (デフォルト) のいずれかです。スレーブ SQL スレッドが停止されている場合にのみ変更できます。詳細は、ドキュメントを参照してください。		

循環 (「アクティブ-アクティブ」) レプリケーションセットアップでのこの SQL ノード (および MySQL Cluster) のロールを指定します。ndb_slave_conflict_role では、PRIMARY、SECONDARY、PASS、または NULL (デフォルト) のいずれかの値を取ることができます。ndb_slave_conflict_role を変更するには、その前にスレーブの SQL スレッドを停止する必要があります。また、これを PASS と PRIMARY または SECONDARY

のいずれかの間で直接変更することはできません。変更する場合は、SQL スレッドが停止していることを確認してから、先に SET @@GLOBAL.ndb_slave_conflict_role = 'NONE' を実行する必要があります。

この変数は MySQL Cluster NDB 7.4.1 で追加されました。詳細は、セクション18.6.11「MySQL Cluster レプリケーションの競合解決」を参照してください。

- ndb_table_no_logging

表 18.61 ndb_table_no_logging の型と値の情報

コマンド行	システム変数	ステータス変数
オプションファイル	スコープ	動的
開始バージョン	型	デフォルト、範囲
メモ		
ndb_table_no_logging		
いいえ	はい	いいえ
いいえ	セッション	はい
NDB 7.3	ブール	FALSE
説明: この設定が有効なときに作成された NDB テーブルはディスクに対してチェックポイントを設定しません (ただし、テーブルスキーマファイルは作成されます)。テーブルが NDBCLUSTER で作成されるか、またはそれを使用するように変更されたときに有効な設定は、そのテーブルの有効期間にわたって保持されます。		

この変数を ON または 1 に設定すると、NDB テーブルのチェックポイントがディスクに対して実行されなくなります。より具体的には、この設定は ndb_table_no_logging が有効になっているときに ENGINE NDB を使用して作成または変更されたテーブルに適用され、あとで ndb_table_no_logging が変更された場合でも、テーブルの存続期間にわたって適用され続けます。ここに示すように、テーブル A、B、C、および D を作成 (および変更) して、ndb_table_no_logging の設定も変更したとします。

```
SET @@ndb_table_no_logging = 1;

CREATE TABLE A ... ENGINE NDB;

CREATE TABLE B ... ENGINE MYISAM;
CREATE TABLE C ... ENGINE MYISAM;

ALTER TABLE B ENGINE NDB;

SET @@ndb_table_no_logging = 0;

CREATE TABLE D ... ENGINE NDB;
ALTER TABLE C ENGINE NDB;

SET @@ndb_table_no_logging = 1;
```

前の一連のイベントのあと、テーブル A と B のチェックポイントは実行されません (A は ENGINE NDB を使用して作成され、B は NDB を使用するように変更されましたが、どちらも ndb_table_no_logging の有効時に行われたため)。ただし、テーブル C と D は記録されます (C は NDB を使用するように変更され、D は ENGINE NDB を使用して作成されましたが、どちらも ndb_table_no_logging の無効時に行われたため)。ndb_table_no_logging を 1 または ON に再度設定しても、テーブル C または D のチェックポイントは実行されません。

> 注記
>
> ndb_table_no_logging は、NDB テーブルスキーマファイルの作成では無効になります。これをサポートするには、代わりに ndb_table_temporary を使用してください。

- ndb_table_temporary

表 18.62 ndb_table_temporary の型と値の情報

コマンド行	システム変数	ステータス変数
オプションファイル	スコープ	動的
開始バージョン	型	デフォルト、範囲
メモ		
ndb_table_temporary		
いいえ	はい	いいえ
いいえ	セッション	はい
NDB 7.3	ブール	FALSE
説明: NDB テーブルはディスク上で永続的ではありません。スキーマファイルは作成されず、テーブルはログに記録されません		

この変数を ON または 1 に設定すると、NDB テーブルがディスクに書き込まれなくなります。これは、テーブルスキーマファイルが作成されず、テーブルが記録されないことを意味します。

> **注記**
>
> MySQL Cluster NDB 7.0 以降では、現在のところ、この変数を設定しても無効になります。これは既知の問題です。Bug #34036 を参照してください。

- ndb_use_copying_alter_table

表 18.63 ndb_use_copying_alter_table の型と値の情報

コマンド行	システム変数	ステータス変数
オプションファイル	スコープ	動的
開始バージョン	型	デフォルト、範囲
メモ		
ndb_use_copying_alter_table		
いいえ	はい	いいえ
いいえ	両方	いいえ
説明: MySQL Cluster で ALTER TABLE 操作のコピーを使用		

オンラインの ALTER TABLE 操作で問題が発生したときに、NDB によりテーブルのコピーが強制的に使用されます。デフォルト値は OFF です。

- ndb_use_exact_count

表 18.64 ndb_use_exact_count の型と値の情報

コマンド行	システム変数	ステータス変数
オプションファイル	スコープ	動的
開始バージョン	型	デフォルト、範囲
メモ		
ndb_use_exact_count		
いいえ	はい	いいえ
いいえ	両方	はい
NDB 7.3	ブール	ON
5.1.47-ndb-7.1.8	ブール	OFF

コマンド行	システム変数	ステータス変数
オプションファイル	スコープ	動的
開始バージョン	型	デフォルト、範囲
メモ		
説明: クエリーを計画する際に正確な行数を使用		

SELECT COUNT(*) クエリーでこのタイプのクエリーの高速化を計画するときに、NDB によりレコードのカウントが強制的に使用されます。デフォルト値は OFF で、クエリー全体が高速化されます。

- ndb_use_transactions

表 18.65 ndb_use_transactions の型と値の情報

コマンド行	システム変数	ステータス変数
オプションファイル	スコープ	動的
開始バージョン	型	デフォルト、範囲
メモ		
ndb_use_transactions		
はい	はい	いいえ
はい	両方	はい
NDB 7.3	ブール	ON
説明: SELECT COUNT(*) クエリー計画中に、強制的に NDB にレコードの数を使用してこのタイプのクエリーを高速化させる		

NDB のトランザクションサポートを無効にするには、この変数の値を OFF に設定します (推奨されません)。デフォルトは ON です。

- ndb_version

表 18.66 ndb_version の型と値の情報

コマンド行	システム変数	ステータス変数
オプションファイル	スコープ	動的
開始バージョン	型	デフォルト、範囲
メモ		
ndb_version		
いいえ	はい	いいえ
いいえ	グローバル	いいえ
NDB 7.3	文字列	
説明: ビルドおよび NDB エンジンのバージョンを整数として表示します。		

NDB エンジンのバージョン (合成整数として)。

- ndb_version_string

表 18.67 ndb_version_string の型と値の情報

コマンド行	システム変数	ステータス変数
オプションファイル	スコープ	動的
開始バージョン	型	デフォルト、範囲
メモ		
ndb_version_string		
いいえ	はい	いいえ

コマンド行	システム変数	ステータス変数
オプションファイル	スコープ	動的
開始バージョン	型	デフォルト、範囲
メモ		
いいえ	グローバル	いいえ
NDB 7.3	文字列	
説明: NDB エンジンのバージョンを含むビルド情報を ndb-x.y.z の形式で表示します。		

NDB エンジンのバージョン (ndb-x.y.z 形式)。

- server_id_bits

表 18.68 server_id_bits の型と値の情報

コマンド行	システム変数	ステータス変数
オプションファイル	スコープ	動的
開始バージョン	型	デフォルト、範囲
メモ		
server_id_bits		
はい	はい	いいえ
はい	グローバル	いいえ
NDB 7.3	数値	32 / 7 - 32
説明: --server-id-bits オプションがデフォルト以外の値に設定された状態でサーバーが起動された場合の server_id の有効な値。		

--server-id-bits オプションをデフォルト以外の値に設定してサーバーを起動した場合の server_id の有効値。

server_id の値が 2 の server_id_bits 乗以上の場合、mysqld は起動を拒否します。

このシステム変数は MySQL Cluster でのみサポートされます。server_id_bits は標準の MySQL Server ではサポートされません。

- transaction_allow_batching

表 18.69 transaction_allow_batching の型と値の情報

コマンド行	システム変数	ステータス変数
オプションファイル	スコープ	動的
開始バージョン	型	デフォルト、範囲
メモ		
transaction_allow_batching		
いいえ	はい	いいえ
いいえ	セッション	はい
NDB 7.3	ブール	FALSE

コマンド行	システム変数	ステータス変数
オプションファイル	スコープ	動的
開始バージョン	型	デフォルト、範囲
メモ		
説明: トランザクション内でのステートメントのバッチ化を許可します。使用するには AUTOCOMMIT を無効にします。		

この変数を 1 または ON に設定すると、同じトランザクション内のステートメントのバッチ化が有効になります。この変数を使用するには、先に autocommit を 0 または OFF に設定して無効にする必要があります。これを実行しない場合、transaction_allow_batching を設定しても無効になります。

この変数は、有効にすると「以前の」イメージから読み取りが行われる可能性があるため、書き込みのみを実行するトランザクションで使用するのが安全です。SELECT を発行する前に、(必要に応じて明示的な COMMIT を使用して) 保留中のトランザクションをすべて確実にコミットするようにしてください。

> **重要**
>
> 特定のステートメントの効果が同じトランザクション内の前のステートメントの結果に依存する可能性がある場合は、transaction_allow_batching を使用しないでください。

この変数は、現在のところ MySQL Cluster でのみサポートされます。

次のリストのシステム変数は、すべて ndbinfo 情報データベースに関連するものです。

- ndbinfo_database

表 18.70 ndbinfo_database の型と値の情報

コマンド行	システム変数	ステータス変数
オプションファイル	スコープ	動的
開始バージョン	型	デフォルト、範囲
メモ		
ndbinfo_database		
いいえ	はい	いいえ
いいえ	グローバル	いいえ
NDB 7.3	文字列	ndbinfo
説明: NDB 情報データベースに使用される名前。読み取り専用。		

NDB 情報データベースに使用される名前を示します。デフォルトは ndbinfo です。これは、コンパイル時に値が指定される読み取り専用の変数です。これを設定するには、--ndbinfo-database=name を使用してサーバーを起動します。これにより、この変数で示される値が設定されますが、NDB 情報データベースに使用される名前は実際には変更されません。

- ndbinfo_max_bytes

表 18.71 ndbinfo_max_bytes の型と値の情報

コマンド行	システム変数	ステータス変数
オプションファイル	スコープ	動的
開始バージョン	型	デフォルト、範囲
メモ		
ndbinfo_max_bytes		
はい	はい	いいえ
いいえ	両方	はい
NDB 7.3	数値	0 / -

コマンド行	システム変数	ステータス変数
オプションファイル	スコープ	動的
開始バージョン	型	デフォルト、範囲
メモ		
説明: デバッグにのみ使用されます。		

テストとデバッグでのみ使用されます。

- ndbinfo_max_rows

表 18.72 ndbinfo_max_rows の型と値の情報

コマンド行	システム変数	ステータス変数
オプションファイル	スコープ	動的
開始バージョン	型	デフォルト、範囲
メモ		
ndbinfo_max_rows		
はい	はい	いいえ
いいえ	両方	はい
NDB 7.3	数値	10 / -
説明: デバッグにのみ使用されます。		

テストとデバッグでのみ使用されます。

- ndbinfo_offline

表 18.73 ndbinfo_offline の型と値の情報

コマンド行	システム変数	ステータス変数
オプションファイル	スコープ	動的
開始バージョン	型	デフォルト、範囲
メモ		
ndbinfo_offline		
いいえ	はい	いいえ
いいえ	グローバル	はい
NDB 7.3	ブール	OFF
説明: ndbinfo データベースをオフラインモードにします。この場合、テーブルまたはビューから行は返されません。		

ndbinfo データベースをオフラインモードにします。オフラインモードでは、テーブルやビューが実際には存在しない場合や、存在するが NDB での定義が異なる場合でも、テーブルやビューを開くことができます。このようなテーブル (またはビュー) からは、行は返されません。

- ndbinfo_show_hidden

表 18.74 ndbinfo_show_hidden の型と値の情報

コマンド行	システム変数	ステータス変数
オプションファイル	スコープ	動的
開始バージョン	型	デフォルト、範囲
メモ		
ndbinfo_show_hidden		
はい	はい	いいえ

コマンド行	システム変数	ステータス変数
オプションファイル	スコープ	動的
開始バージョン	型	デフォルト、範囲
メモ		
いいえ	両方	はい
NDB 7.3	ブール	OFF
説明: ndbinfo 内部ベーステーブルを mysql クライアントで表示するかどうか。デフォルトは OFF です。		

ndbinfo データベースのベースとなる内部テーブルが mysql クライアントに表示されるかどうか。デフォルトは OFF です。

- ndbinfo_table_prefix

表 18.75 ndbinfo_table_prefix の型と値の情報

コマンド行	システム変数	ステータス変数
オプションファイル	スコープ	動的
開始バージョン	型	デフォルト、範囲
メモ		
ndbinfo_table_prefix		
はい	はい	いいえ
いいえ	両方	はい
NDB 7.3	文字列	ndb$
説明: ndbinfo 内部ベーステーブルの命名に使用するプリフィクス		

ndbinfo データベースのベーステーブル (ndbinfo_show_hidden を設定して公開されないかぎり、通常は非表示です) の名前に使用されるプリフィクス。これは読み取り専用の変数であり、デフォルト値は 「ndb$」 です。サーバーの起動時に --ndbinfo-table-prefix オプションを使用できますが、その場合はこの変数が設定されるだけで、非表示のベーステーブルの名前に使用される実際のプリフィクスは変更されません。プリフィクス自体はコンパイル時に指定されます。

- ndbinfo_version

表 18.76 ndbinfo_version の型と値の情報

コマンド行	システム変数	ステータス変数
オプションファイル	スコープ	動的
開始バージョン	型	デフォルト、範囲
メモ		
ndbinfo_version		
いいえ	はい	いいえ
いいえ	グローバル	いいえ
NDB 7.3	文字列	
説明: ndbinfo エンジンのバージョン。読み取り専用。		

使用中の ndbinfo エンジンのバージョンを示します (読み取り専用)。

18.3.4.4 MySQL Cluster のステータス変数

このセクションでは、MySQL Cluster および NDB ストレージエンジンに関連する MySQL サーバーステータス変数について詳しく説明します。MySQL Cluster に固有でないステータス変数、およびステータス変数の使用に関する一般的な情報は、セクション5.1.6「サーバーステータス変数」を参照してください。

- Handler_discover

MySQL サーバーは、NDBCLUSTER ストレージエンジンに対して特定の名前を持つテーブルの情報を問い合わせることができます。これは検出と呼ばれます。Handler_discover は、このメカニズムを使用してテーブルが検出された回数を示します。

- Ndb_api_bytes_sent_count_session

  このクライアントセッションでデータノードに送信されたデータ量 (バイト単位)。

  この変数は、SHOW GLOBAL STATUS または SHOW SESSION STATUS を使用して読み取ることができますが、現在のセッションにのみ関連しており、この mysqld のほかのクライアントには影響しません。

  詳細は、セクション18.5.15「NDB API 統計のカウンタと変数」を参照してください。

- Ndb_api_bytes_sent_count_slave

  このスレーブによってデータノードに送信されたデータ量 (バイト単位)。

  この変数は、SHOW GLOBAL STATUS または SHOW SESSION STATUS を使用して読み取ることができますが、スコープは実質的にグローバルです。この MySQL サーバーがレプリケーションのスレーブとして動作していないか、NDB テーブルを使用していない場合、この値は常に 0 です。

  詳細は、セクション18.5.15「NDB API 統計のカウンタと変数」を参照してください。

- Ndb_api_bytes_sent_count

  この MySQL サーバー (SQL ノード) によってデータノードに送信されたデータ量 (バイト単位)。

  この変数は、SHOW GLOBAL STATUS または SHOW SESSION STATUS を使用して読み取ることができますが、スコープは実質的にグローバルです。

  詳細は、セクション18.5.15「NDB API 統計のカウンタと変数」を参照してください。

- Ndb_api_bytes_received_count_session

  このクライアントセッションでデータノードから受信されたデータ量 (バイト単位)。

  この変数は、SHOW GLOBAL STATUS または SHOW SESSION STATUS を使用して読み取ることができますが、現在のセッションにのみ関連しており、この mysqld のほかのクライアントには影響しません。

  詳細は、セクション18.5.15「NDB API 統計のカウンタと変数」を参照してください。

- Ndb_api_bytes_received_count_slave

  このスレーブによってデータノードから受信されたデータ量 (バイト単位)。

  この変数は、SHOW GLOBAL STATUS または SHOW SESSION STATUS を使用して読み取ることができますが、スコープは実質的にグローバルです。この MySQL サーバーがレプリケーションのスレーブとして動作していないか、NDB テーブルを使用していない場合、この値は常に 0 です。

  詳細は、セクション18.5.15「NDB API 統計のカウンタと変数」を参照してください。

- Ndb_api_bytes_received_count

  この MySQL サーバー (SQL ノード) によってデータノードから受信されたデータ量 (バイト単位)。

  この変数は、SHOW GLOBAL STATUS または SHOW SESSION STATUS を使用して読み取ることができますが、スコープは実質的にグローバルです。

  詳細は、セクション18.5.15「NDB API 統計のカウンタと変数」を参照してください。

- Ndb_api_event_data_count_injector

  NDB binlog インジェクタスレッドによって受信された行変更イベントの数。

  この変数は、SHOW GLOBAL STATUS または SHOW SESSION STATUS を使用して読み取ることができますが、スコープは実質的にグローバルです。

  詳細は、セクション18.5.15「NDB API 統計のカウンタと変数」を参照してください。

- Ndb_api_event_data_count

  この MySQL サーバー (SQL ノード) によって受信された行変更イベントの数。

  この変数は、SHOW GLOBAL STATUS または SHOW SESSION STATUS を使用して読み取ることができますが、スコープは実質的にグローバルです。

  詳細は、セクション18.5.15「NDB API 統計のカウンタと変数」を参照してください。

- Ndb_api_event_nondata_count_injector

  NDB バイナリログインジェクタスレッドによって受信された、行変更イベント以外のイベントの数。

  この変数は、SHOW GLOBAL STATUS または SHOW SESSION STATUS を使用して読み取ることができますが、スコープは実質的にグローバルです。

  詳細は、セクション18.5.15「NDB API 統計のカウンタと変数」を参照してください。

- Ndb_api_event_nondata_count

  この MySQL サーバー (SQL ノード) によって受信された、行変更イベント以外のイベントの数。

  この変数は、SHOW GLOBAL STATUS または SHOW SESSION STATUS を使用して読み取ることができますが、スコープは実質的にグローバルです。

  詳細は、セクション18.5.15「NDB API 統計のカウンタと変数」を参照してください。

- Ndb_api_event_bytes_count_injector

  NDB binlog インジェクタスレッドによって受信されたイベントのバイト数。

  この変数は、SHOW GLOBAL STATUS または SHOW SESSION STATUS を使用して読み取ることができますが、スコープは実質的にグローバルです。

  詳細は、セクション18.5.15「NDB API 統計のカウンタと変数」を参照してください。

- Ndb_api_event_bytes_count

  この MySQL サーバー (SQL ノード) によって受信されたイベントのバイト数。

  この変数は、SHOW GLOBAL STATUS または SHOW SESSION STATUS を使用して読み取ることができますが、スコープは実質的にグローバルです。

  詳細は、セクション18.5.15「NDB API 統計のカウンタと変数」を参照してください。

- Ndb_api_pk_op_count_session

  このクライアントセッションで、主キーに基づいた、または主キーを使用した操作の数。これには、BLOB テーブルに対する操作、暗黙的なロック解除操作、自動インクリメント操作、およびユーザーの管理下にある主キー操作が含まれます。

  この変数は、SHOW GLOBAL STATUS または SHOW SESSION STATUS を使用して読み取ることができますが、現在のセッションにのみ関連しており、この mysqld のほかのクライアントには影響しません。

  詳細は、セクション18.5.15「NDB API 統計のカウンタと変数」を参照してください。

- Ndb_api_pk_op_count_slave

  このスレーブによる、主キーに基づいた、または主キーを使用した操作の数。これには、BLOB テーブルに対する操作、暗黙的なロック解除操作、自動インクリメント操作、およびユーザーの管理下にある主キー操作が含まれます。

  この変数は、SHOW GLOBAL STATUS または SHOW SESSION STATUS を使用して読み取ることができますが、スコープは実質的にグローバルです。この MySQL サーバーがレプリケーションのスレーブとして動作していないか、NDB テーブルを使用していない場合、この値は常に 0 です。

  詳細は、セクション18.5.15「NDB API 統計のカウンタと変数」を参照してください。

- Ndb_api_pk_op_count

  この MySQL サーバー (SQL ノード) による、主キーに基づいた、または主キーを使用した操作の数。これには、BLOB テーブルに対する操作、暗黙的なロック解除操作、自動インクリメント操作、およびユーザーの管理下にある主キー操作が含まれます。

  この変数は、SHOW GLOBAL STATUS または SHOW SESSION STATUS を使用して読み取ることができますが、スコープは実質的にグローバルです。

  詳細は、セクション18.5.15「NDB API 統計のカウンタと変数」を参照してください。

- Ndb_api_pruned_scan_count_session

  このクライアントセッションで単一のパーティションにプルーニングされたスキャンの回数。

  この変数は、SHOW GLOBAL STATUS または SHOW SESSION STATUS を使用して読み取ることができますが、現在のセッションにのみ関連しており、この mysqld のほかのクライアントには影響しません。

  詳細は、セクション18.5.15「NDB API 統計のカウンタと変数」を参照してください。

- Ndb_api_pruned_scan_count_slave

  このスレーブによって単一のパーティションにプルーニングされたスキャンの回数。

  この変数は、SHOW GLOBAL STATUS または SHOW SESSION STATUS を使用して読み取ることができますが、スコープは実質的にグローバルです。この MySQL サーバーがレプリケーションのスレーブとして動作していないか、NDB テーブルを使用していない場合、この値は常に 0 です。

  詳細は、セクション18.5.15「NDB API 統計のカウンタと変数」を参照してください。

- Ndb_api_pruned_scan_count

  この MySQL サーバー (SQL ノード) によって単一のパーティションにプルーニングされたスキャンの回数。

  この変数は、SHOW GLOBAL STATUS または SHOW SESSION STATUS を使用して読み取ることができますが、スコープは実質的にグローバルです。

  詳細は、セクション18.5.15「NDB API 統計のカウンタと変数」を参照してください。

- Ndb_api_range_scan_count_session

  このクライアントセッションで開始された範囲スキャンの回数。

  この変数は、SHOW GLOBAL STATUS または SHOW SESSION STATUS を使用して読み取ることができますが、現在のセッションにのみ関連しており、この mysqld のほかのクライアントには影響しません。

  詳細は、セクション18.5.15「NDB API 統計のカウンタと変数」を参照してください。

- Ndb_api_range_scan_count_slave

  このスレーブによって開始された範囲スキャンの回数。

  この変数は、SHOW GLOBAL STATUS または SHOW SESSION STATUS を使用して読み取ることができますが、スコープは実質的にグローバルです。この MySQL サーバーがレプリケーションのスレーブとして動作していないか、NDB テーブルを使用していない場合、この値は常に 0 です。

  詳細は、セクション18.5.15「NDB API 統計のカウンタと変数」を参照してください。

- Ndb_api_range_scan_count

  この MySQL サーバー (SQL ノード) によって開始された範囲スキャンの回数。

  この変数は、SHOW GLOBAL STATUS または SHOW SESSION STATUS を使用して読み取ることができますが、スコープは実質的にグローバルです。

  詳細は、セクション18.5.15「NDB API 統計のカウンタと変数」を参照してください。

- Ndb_api_read_row_count_session

  このクライアントセッションで読み取られた行の合計数。これには、このクライアントセッションで作成された主キー、一意キー、またはスキャン操作によって読み取られたすべての行が含まれます。

  この変数は、SHOW GLOBAL STATUS または SHOW SESSION STATUS を使用して読み取ることができますが、現在のセッションにのみ関連しており、この mysqld のほかのクライアントには影響しません。

  詳細は、セクション18.5.15「NDB API 統計のカウンタと変数」を参照してください。

- Ndb_api_read_row_count_slave

  このスレーブによって読み取られた行の合計数。これには、このスレーブによって作成された主キー、一意キー、またはスキャン操作によって読み取られたすべての行が含まれます。

  この変数は、SHOW GLOBAL STATUS または SHOW SESSION STATUS を使用して読み取ることができますが、スコープは実質的にグローバルです。この MySQL サーバーがレプリケーションのスレーブとして動作していないか、NDB テーブルを使用していない場合、この値は常に 0 です。

  詳細は、セクション18.5.15「NDB API 統計のカウンタと変数」を参照してください。

- Ndb_api_read_row_count

  この MySQL サーバー (SQL ノード) によって読み取られた行の合計数。これには、この MySQL Server (SQL ノード) によって作成された主キー、一意キー、またはスキャン操作によって読み取られたすべての行が含まれます。

  この変数は、SHOW GLOBAL STATUS または SHOW SESSION STATUS を使用して読み取ることができますが、スコープは実質的にグローバルです。

  詳細は、セクション18.5.15「NDB API 統計のカウンタと変数」を参照してください。

- Ndb_api_scan_batch_count_session

  このクライアントセッションで受信された行のバッチ数。1 バッチは、単一のフラグメントから得られた 1 セットのスキャン結果として定義されます。

  この変数は、SHOW GLOBAL STATUS または SHOW SESSION STATUS を使用して読み取ることができますが、現在のセッションにのみ関連しており、この mysqld のほかのクライアントには影響しません。

  詳細は、セクション18.5.15「NDB API 統計のカウンタと変数」を参照してください。

- Ndb_api_scan_batch_count_slave

  このスレーブによって受信された行のバッチ数。1 バッチは、単一のフラグメントから得られた 1 セットのスキャン結果として定義されます。

  この変数は、SHOW GLOBAL STATUS または SHOW SESSION STATUS を使用して読み取ることができますが、スコープは実質的にグローバルです。この MySQL サーバーがレプリケーションのスレーブとして動作していないか、NDB テーブルを使用していない場合、この値は常に 0 です。

  詳細は、セクション18.5.15「NDB API 統計のカウンタと変数」を参照してください。

- Ndb_api_scan_batch_count

  この MySQL Server (SQL ノード) によって受信された行のバッチ数。1 バッチは、単一のフラグメントから得られた 1 セットのスキャン結果として定義されます。

  この変数は、SHOW GLOBAL STATUS または SHOW SESSION STATUS を使用して読み取ることができますが、スコープは実質的にグローバルです。

  詳細は、セクション18.5.15「NDB API 統計のカウンタと変数」を参照してください。

- Ndb_api_table_scan_count_session

  このクライアントセッションで開始された、内部テーブルのスキャンを含むテーブルスキャンの回数。

  この変数は、SHOW GLOBAL STATUS または SHOW SESSION STATUS を使用して読み取ることができますが、現在のセッションにのみ関連しており、この mysqld のほかのクライアントには影響しません。

詳細は、セクション18.5.15「NDB API 統計のカウンタと変数」を参照してください。

- Ndb_api_table_scan_count_slave

  このスレーブによって開始された、内部テーブルのスキャンを含むテーブルスキャンの回数。

  この変数は、SHOW GLOBAL STATUS または SHOW SESSION STATUS を使用して読み取ることができますが、スコープは実質的にグローバルです。この MySQL サーバーがレプリケーションのスレーブとして動作していないか、NDB テーブルを使用していない場合、この値は常に 0 です。

  詳細は、セクション18.5.15「NDB API 統計のカウンタと変数」を参照してください。

- Ndb_api_table_scan_count

  この MySQL サーバー (SQL ノード) によって開始された、内部テーブルのスキャンを含むテーブルスキャンの回数。

  この変数は、SHOW GLOBAL STATUS または SHOW SESSION STATUS を使用して読み取ることができますが、スコープは実質的にグローバルです。

  詳細は、セクション18.5.15「NDB API 統計のカウンタと変数」を参照してください。

- Ndb_api_trans_abort_count_session

  このクライアントセッションで中止されたトランザクションの数。

  この変数は、SHOW GLOBAL STATUS または SHOW SESSION STATUS を使用して読み取ることができますが、現在のセッションにのみ関連しており、この mysqld のほかのクライアントには影響しません。

  詳細は、セクション18.5.15「NDB API 統計のカウンタと変数」を参照してください。

- Ndb_api_trans_abort_count_slave

  このスレーブにより中止されたトランザクションの数。

  この変数は、SHOW GLOBAL STATUS または SHOW SESSION STATUS を使用して読み取ることができますが、スコープは実質的にグローバルです。この MySQL サーバーがレプリケーションのスレーブとして動作していないか、NDB テーブルを使用していない場合、この値は常に 0 です。

  詳細は、セクション18.5.15「NDB API 統計のカウンタと変数」を参照してください。

- Ndb_api_trans_abort_count

  この MySQL サーバー (SQL ノード) により中止されたトランザクションの数。

  この変数は、SHOW GLOBAL STATUS または SHOW SESSION STATUS を使用して読み取ることができますが、スコープは実質的にグローバルです。

  詳細は、セクション18.5.15「NDB API 統計のカウンタと変数」を参照してください。

- Ndb_api_trans_close_count_session

  このクライアントセッションで閉じられたトランザクションの数。一部のトランザクションがロールバックされている場合があるため、この値は Ndb_api_trans_commit_count_session と Ndb_api_trans_abort_count_session の合計より大きくなることがあります。

  この変数は、SHOW GLOBAL STATUS または SHOW SESSION STATUS を使用して読み取ることができますが、現在のセッションにのみ関連しており、この mysqld のほかのクライアントには影響しません。

  詳細は、セクション18.5.15「NDB API 統計のカウンタと変数」を参照してください。

- Ndb_api_trans_close_count_slave

  このスレーブにより閉じられたトランザクションの数。一部のトランザクションがロールバックされている場合があるため、この値は Ndb_api_trans_commit_count_slave と Ndb_api_trans_abort_count_slave の合計より大きくなることがあります。

  この変数は、SHOW GLOBAL STATUS または SHOW SESSION STATUS を使用して読み取ることができますが、スコープは実質的にグローバルです。この MySQL サーバーがレプリケーションのスレーブとして動作していないか、NDB テーブルを使用していない場合、この値は常に 0 です。

  詳細は、セクション18.5.15「NDB API 統計のカウンタと変数」を参照してください。

- Ndb_api_trans_close_count

  この MySQL サーバー (SQL ノード) により閉じられたトランザクションの数。一部のトランザクションがロールバックされている場合があるため、この値は Ndb_api_trans_commit_count と Ndb_api_trans_abort_count の合計より大きくなることがあります。

  この変数は、SHOW GLOBAL STATUS または SHOW SESSION STATUS を使用して読み取ることができますが、スコープは実質的にグローバルです。

  詳細は、セクション18.5.15「NDB API 統計のカウンタと変数」を参照してください。

- Ndb_api_trans_commit_count_session

  このクライアントセッションでコミットされたトランザクションの数。

  この変数は、SHOW GLOBAL STATUS または SHOW SESSION STATUS を使用して読み取ることができますが、現在のセッションにのみ関連しており、この mysqld のほかのクライアントには影響しません。

  詳細は、セクション18.5.15「NDB API 統計のカウンタと変数」を参照してください。

- Ndb_api_trans_commit_count_slave

  このスレーブによりコミットされたトランザクションの数。

  この変数は、SHOW GLOBAL STATUS または SHOW SESSION STATUS を使用して読み取ることができますが、スコープは実質的にグローバルです。この MySQL サーバーがレプリケーションのスレーブとして動作していないか、NDB テーブルを使用していない場合、この値は常に 0 です。

  詳細は、セクション18.5.15「NDB API 統計のカウンタと変数」を参照してください。

- Ndb_api_trans_commit_count

  この MySQL サーバー (SQL ノード) によりコミットされたトランザクションの数。

  この変数は、SHOW GLOBAL STATUS または SHOW SESSION STATUS を使用して読み取ることができますが、スコープは実質的にグローバルです。

  詳細は、セクション18.5.15「NDB API 統計のカウンタと変数」を参照してください。

- Ndb_api_trans_local_read_row_count_session

  このクライアントセッションで読み取られた行の合計数。これには、このクライアントセッションで作成された主キー、一意キー、またはスキャン操作によって読み取られたすべての行が含まれます。

  この変数は、SHOW GLOBAL STATUS または SHOW SESSION STATUS を使用して読み取ることができますが、現在のセッションにのみ関連しており、この mysqld のほかのクライアントには影響しません。

  詳細は、セクション18.5.15「NDB API 統計のカウンタと変数」を参照してください。

- Ndb_api_trans_local_read_row_count_slave

  このスレーブによって読み取られた行の合計数。これには、このスレーブによって作成された主キー、一意キー、またはスキャン操作によって読み取られたすべての行が含まれます。

  この変数は、SHOW GLOBAL STATUS または SHOW SESSION STATUS を使用して読み取ることができますが、スコープは実質的にグローバルです。この MySQL サーバーがレプリケーションのスレーブとして動作していないか、NDB テーブルを使用していない場合、この値は常に 0 です。

詳細は、セクション18.5.15「NDB API 統計のカウンタと変数」を参照してください。

- Ndb_api_trans_local_read_row_count

  この MySQL サーバー (SQL ノード) によって読み取られた行の合計数。これには、この MySQL Server (SQL ノード) によって作成された主キー、一意キー、またはスキャン操作によって読み取られたすべての行が含まれます。

  この変数は、SHOW GLOBAL STATUS または SHOW SESSION STATUS を使用して読み取ることができますが、スコープは実質的にグローバルです。

  詳細は、セクション18.5.15「NDB API 統計のカウンタと変数」を参照してください。

- Ndb_api_trans_start_count_session

  このクライアントセッションで開始されたトランザクションの数。

  この変数は、SHOW GLOBAL STATUS または SHOW SESSION STATUS を使用して読み取ることができますが、現在のセッションにのみ関連しており、この mysqld のほかのクライアントには影響しません。

  詳細は、セクション18.5.15「NDB API 統計のカウンタと変数」を参照してください。

- Ndb_api_trans_start_count_slave

  このスレーブにより開始されたトランザクションの数。

  この変数は、SHOW GLOBAL STATUS または SHOW SESSION STATUS を使用して読み取ることができますが、スコープは実質的にグローバルです。この MySQL サーバーがレプリケーションのスレーブとして動作していないか、NDB テーブルを使用していない場合、この値は常に 0 です。

  詳細は、セクション18.5.15「NDB API 統計のカウンタと変数」を参照してください。

- Ndb_api_trans_start_count

  この MySQL サーバー (SQL ノード) により開始されたトランザクションの数。

  この変数は、SHOW GLOBAL STATUS または SHOW SESSION STATUS を使用して読み取ることができますが、スコープは実質的にグローバルです。

  詳細は、セクション18.5.15「NDB API 統計のカウンタと変数」を参照してください。

- Ndb_api_uk_op_count_session

  このクライアントセッションで、一意キーに基づいた、または一意キーを使用した操作の数。

  この変数は、SHOW GLOBAL STATUS または SHOW SESSION STATUS を使用して読み取ることができますが、現在のセッションにのみ関連しており、この mysqld のほかのクライアントには影響しません。

  詳細は、セクション18.5.15「NDB API 統計のカウンタと変数」を参照してください。

- Ndb_api_uk_op_count_slave

  このスレーブによる、一意キーに基づいた、または一意キーを使用した操作の数。

  この変数は、SHOW GLOBAL STATUS または SHOW SESSION STATUS を使用して読み取ることができますが、スコープは実質的にグローバルです。この MySQL サーバーがレプリケーションのスレーブとして動作していないか、NDB テーブルを使用していない場合、この値は常に 0 です。

  詳細は、セクション18.5.15「NDB API 統計のカウンタと変数」を参照してください。

- Ndb_api_uk_op_count

  この MySQL サーバー (SQL ノード) による、一意キーに基づいた、または一意キーを使用した操作の数。

  この変数は、SHOW GLOBAL STATUS または SHOW SESSION STATUS を使用して読み取ることができますが、スコープは実質的にグローバルです。

  詳細は、セクション18.5.15「NDB API 統計のカウンタと変数」を参照してください。

- Ndb_api_wait_exec_complete_count_session

  このクライアントセッションで、操作の実行が完了するのを待機する間にスレッドがブロックされた回数。これには、すべての execute() 呼び出しと、クライアントの管理下にない BLOB および自動インクリメント操作の暗黙的な実行が含まれます。

  この変数は、SHOW GLOBAL STATUS または SHOW SESSION STATUS を使用して読み取ることができますが、現在のセッションにのみ関連しており、この mysqld のほかのクライアントには影響しません。

  詳細は、セクション18.5.15「NDB API 統計のカウンタと変数」を参照してください。

- Ndb_api_wait_exec_complete_count_slave

  操作の実行が完了するまで待機する間に、このスレーブによってスレッドがブロックされた回数。これには、すべての execute() 呼び出しと、クライアントの管理下にない BLOB および自動インクリメント操作の暗黙的な実行が含まれます。

  この変数は、SHOW GLOBAL STATUS または SHOW SESSION STATUS を使用して読み取ることができますが、スコープは実質的にグローバルです。この MySQL サーバーがレプリケーションのスレーブとして動作していないか、NDB テーブルを使用していない場合、この値は常に 0 です。

  詳細は、セクション18.5.15「NDB API 統計のカウンタと変数」を参照してください。

- Ndb_api_wait_exec_complete_count

  操作の実行が完了するまで待機する間に、この MySQL Server (SQL ノード) によってスレッドがブロックされた回数。これには、すべての execute() 呼び出しと、クライアントの管理下にない BLOB および自動インクリメント操作の暗黙的な実行が含まれます。

  この変数は、SHOW GLOBAL STATUS または SHOW SESSION STATUS を使用して読み取ることができますが、スコープは実質的にグローバルです。

  詳細は、セクション18.5.15「NDB API 統計のカウンタと変数」を参照してください。

- Ndb_api_wait_meta_request_count_session

  このクライアントセッションで、メタデータベースの信号 (DDL 要求、新しいエポック、トランザクションレコードの占有など) を待機する間にスレッドがブロックされた回数。

  この変数は、SHOW GLOBAL STATUS または SHOW SESSION STATUS を使用して読み取ることができますが、現在のセッションにのみ関連しており、この mysqld のほかのクライアントには影響しません。

  詳細は、セクション18.5.15「NDB API 統計のカウンタと変数」を参照してください。

- Ndb_api_wait_meta_request_count_slave

  メタデータベースの信号 (DDL 要求、新しいエポック、トランザクションレコードの占有など) を待機する間に、このスレーブによってスレッドがブロックされた回数。

  この変数は、SHOW GLOBAL STATUS または SHOW SESSION STATUS を使用して読み取ることができますが、スコープは実質的にグローバルです。この MySQL サーバーがレプリケーションのスレーブとして動作していないか、NDB テーブルを使用していない場合、この値は常に 0 です。

  詳細は、セクション18.5.15「NDB API 統計のカウンタと変数」を参照してください。

- Ndb_api_wait_meta_request_count

  メタデータベースの信号 (DDL 要求、新しいエポック、トランザクションレコードの占有など) を待機する間に、この MySQL Server (SQL ノード) によってスレッドがブロックされた回数。

  この変数は、SHOW GLOBAL STATUS または SHOW SESSION STATUS を使用して読み取ることができますが、スコープは実質的にグローバルです。

  詳細は、セクション18.5.15「NDB API 統計のカウンタと変数」を参照してください。

- Ndb_api_wait_nanos_count_session

  このクライアントセッションで、データノードから送信される任意のタイプの信号の待機に費やされた合計時間 (ナノ秒)。

この変数は、SHOW GLOBAL STATUS または SHOW SESSION STATUS を使用して読み取ることができますが、現在のセッションにのみ関連しており、この mysqld のほかのクライアントには影響しません。

詳細は、セクション18.5.15「NDB API 統計のカウンタと変数」を参照してください。

- Ndb_api_wait_nanos_count_slave

  このスレーブによって、データノードから送信される任意のタイプの信号の待機に費やされた合計時間 (ナノ秒)。

  この変数は、SHOW GLOBAL STATUS または SHOW SESSION STATUS を使用して読み取ることができますが、スコープは実質的にグローバルです。この MySQL サーバーがレプリケーションのスレーブとして動作していないか、NDB テーブルを使用していない場合、この値は常に 0 です。

  詳細は、セクション18.5.15「NDB API 統計のカウンタと変数」を参照してください。

- Ndb_api_wait_nanos_count

  この MySQL Server (SQL ノード) によって、データノードから送信される任意のタイプの信号の待機に費やされた合計時間 (ナノ秒)。

  この変数は、SHOW GLOBAL STATUS または SHOW SESSION STATUS を使用して読み取ることができますが、スコープは実質的にグローバルです。

  詳細は、セクション18.5.15「NDB API 統計のカウンタと変数」を参照してください。

- Ndb_api_wait_scan_result_count_session

  このクライアントセッションで、スキャンベースの信号を待機する間 (追加のスキャン結果を待機するときや、スキャンが閉じるまで待機するときなど) にスレッドがブロックされた回数。

  この変数は、SHOW GLOBAL STATUS または SHOW SESSION STATUS を使用して読み取ることができますが、現在のセッションにのみ関連しており、この mysqld のほかのクライアントには影響しません。

  詳細は、セクション18.5.15「NDB API 統計のカウンタと変数」を参照してください。

- Ndb_api_wait_scan_result_count_slave

  スキャンベースの信号を待機する間 (追加のスキャン結果を待機するときや、スキャンが閉じるまで待機するときなど) に、このスレーブによってスレッドがブロックされた回数。

  この変数は、SHOW GLOBAL STATUS または SHOW SESSION STATUS を使用して読み取ることができますが、スコープは実質的にグローバルです。この MySQL サーバーがレプリケーションのスレーブとして動作していないか、NDB テーブルを使用していない場合、この値は常に 0 です。

  詳細は、セクション18.5.15「NDB API 統計のカウンタと変数」を参照してください。

- Ndb_api_wait_scan_result_count

  スキャンベースの信号を待機する間 (追加のスキャン結果を待機するときや、スキャンが閉じるまで待機するときなど) に、この MySQL Server (SQL ノード) によってスレッドがブロックされた回数。

  この変数は、SHOW GLOBAL STATUS または SHOW SESSION STATUS を使用して読み取ることができますが、スコープは実質的にグローバルです。

  詳細は、セクション18.5.15「NDB API 統計のカウンタと変数」を参照してください。

- Ndb_cluster_node_id

  サーバーが MySQL Cluster ノードとして機能している場合、この変数の値はクラスタ内のノード ID です。

  サーバーが MySQL Cluster の一部でない場合、この変数の値は 0 です。

- Ndb_config_from_host

  サーバーが MySQL Cluster の一部である場合、この変数の値はサーバーの構成データの取得元であるクラスタ管理サーバーのホスト名または IP アドレスです。

  サーバーが MySQL Cluster の一部でない場合、この変数の値は空の文字列です。

- Ndb_config_from_port

  サーバーが MySQL Cluster の一部である場合、この変数の値はサーバーの構成データの取得元であるクラスタ管理サーバーへの接続に使用されているポート番号です。

  サーバーが MySQL Cluster の一部でない場合、この変数の値は 0 です。

- Ndb_conflict_fn_max

  この変数は、MySQL Cluster レプリケーションの競合解決で使用され、この mysqld の最後の起動以降、「もっとも大きいタイムスタンプが優先される」競合解決によって現在の SQL ノードに行が適用されなかった回数を示します。

  詳細は、セクション18.6.11「MySQL Cluster レプリケーションの競合解決」を参照してください。

- Ndb_conflict_fn_old

  この変数は、MySQL Cluster レプリケーションの競合解決で使用され、特定の mysqld の最後の再起動以降、「同じタイムスタンプが優先される」競合解決の結果として行が適用されなかった回数を示します。

  詳細は、セクション18.6.11「MySQL Cluster レプリケーションの競合解決」を参照してください。

- Ndb_conflict_fn_epoch

  この変数は、MySQL Cluster レプリケーションの競合解決で使用され、特定の mysqld の最後の再起動以降、NDB$EPOCH() の競合解決により競合が検出された行の数を示します。

  詳細は、セクション18.6.11「MySQL Cluster レプリケーションの競合解決」を参照してください。

- Ndb_conflict_fn_epoch_trans

  この変数は、MySQL Cluster レプリケーションの競合解決で使用され、特定の mysqld の最後の再起動以降、NDB$EPOCH_TRANS() の競合解決により競合が検出された行の数を示します。

  詳細は、セクション18.6.11「MySQL Cluster レプリケーションの競合解決」を参照してください。

- Ndb_conflict_trans_row_conflict_count

  このステータス変数は、MySQL Cluster レプリケーションの競合解決で使用され、特定の mysqld の最後の再起動以降、トランザクション競合関数によって直接競合していることが検出された行の数を示します。

  現在、MySQL Cluster でサポートされる唯一のトランザクション競合検出関数は NDB$EPOCH_TRANS() です。したがって、このステータス変数は実質的に Ndb_conflict_fn_epoch_trans と同じです。

  詳細は、セクション18.6.11「MySQL Cluster レプリケーションの競合解決」を参照してください。

- Ndb_conflict_trans_row_reject_count

  このステータス変数は、MySQL Cluster レプリケーションの競合解決で使用され、トランザクション競合検出関数によって競合していると判定されたために再編成された行の合計数を示します。これには、Ndb_conflict_trans_row_conflict_count だけでなく、競合するトランザクションに含まれる行や依存する行も含まれます。

  詳細は、セクション18.6.11「MySQL Cluster レプリケーションの競合解決」を参照してください。

- Ndb_conflict_trans_reject_count

  このステータス変数は、MySQL Cluster レプリケーションの競合解決で使用され、トランザクション競合検出関数によって競合が検出されたトランザクションの数を示します。

  詳細は、セクション18.6.11「MySQL Cluster レプリケーションの競合解決」を参照してください。

- Ndb_conflict_trans_detect_iter_count

  これは、MySQL Cluster レプリケーションの競合解決で使用され、エポックトランザクションをコミットするために必要な内部反復の回数を示します。Ndb_conflict_trans_conflict_commit_count より (わずかに) 大きいか、等しい値にしてください。

  詳細は、セクション18.6.11「MySQL Cluster レプリケーションの競合解決」を参照してください。

- Ndb_conflict_trans_conflict_commit_count

  これは、MySQL Cluster レプリケーションの競合解決で使用され、トランザクション競合処理が必要になったあとでコミットされたエポックトランザクションの数を示します。

  詳細は、セクション18.6.11「MySQL Cluster レプリケーションの競合解決」を参照してください。

- Ndb_execute_count

  操作によって行われた NDB カーネルへのラウンドトリップの回数を示します。

- Ndb_number_of_data_nodes

  サーバーが MySQL Cluster の一部である場合、この変数の値はクラスタ内のデータノードの数です。

  サーバーが MySQL Cluster の一部でない場合、この変数の値は 0 です。

- Ndb_pushed_queries_defined

  データノードでの分散処理のために NDB カーネルにプッシュダウンされた結合の合計数。EXPLAIN を使用してテストされたプッシュダウン可能な結合も、この数値の一因となります。

- Ndb_pushed_queries_dropped

  NDB カーネルにプッシュダウンされ、処理できなかった結合の数。

- Ndb_pushed_queries_executed

  NDB に正常にプッシュダウンされ、実行された結合の数。

- Ndb_pushed_reads

  プッシュダウンされた結合によって NDB カーネルから mysqld に返された行の数。NDB にプッシュダウンできる結合に対して EXPLAIN を実行しても、この数値に追加されません。

- Ndb_pruned_scan_count

  この変数には、MySQL Cluster の最後の起動以降、NDBCLUSTER によって実行された (NDBCLUSTER がパーティションプルーニングを使用できた) スキャンの回数のカウントが保持されます。

  スキーマ設計で、この変数を Ndb_scan_count と組み合わせて使用すると、単一のデータノードのみが関与するようにスキャンを単一のテーブルパーティションにプルーニングするサーバーの機能を最大限に生かすことができます。

- Ndb_scan_count

  この変数には、MySQL Cluster の最後の起動以降、NDBCLUSTER によって実行されたスキャンの合計回数のカウントが保持されます。

- Ndb_slave_last_conflict_epoch

  このスレーブで競合が検出された最近のエポック。この値を Ndb_slave_max_replicated_epoch と比較できます。Ndb_slave_max_replicated_epoch が Ndb_slave_last_conflict_epoch より大きい場合、競合はまだ検出されていません。

  この変数は MySQL Cluster NDB 7.4.1 で追加されました。

  詳細は、セクション18.6.11「MySQL Cluster レプリケーションの競合解決」を参照してください。

- Ndb_slave_max_replicated_epoch

  このスレーブで最近コミットされたエポック。MySQL Cluster NDB 7.4.1 以降では、この値を Ndb_slave_last_conflict_epoch と比較できます。この 2 つのうち Ndb_slave_max_replicated_epoch の方が大きい場合、競合はまだ検出されていません。

  この変数は、MySQL Cluster NDB 7.3.8 および MySQL Cluster NDB 7.4.1 で追加されました。

  詳細は、セクション18.6.11「MySQL Cluster レプリケーションの競合解決」を参照してください。

18.3.5 MySQL Cluster での高速インターコネクトの使用

1996 年に NDBCLUSTER の設計の開始前に、並列データベースの構築で発生する主な問題の 1 つがネットワーク内のノード間の通信であることは認識されていました。このため、NDBCLUSTER は当初から複数の異なるデータトランスポートメカニズムを使用できるように設計されました。このマニュアルでは、このメカニズムに対しトランスポータという用語を使用します。

MySQL Cluster のコードベースには、4 種類のトランスポータが用意されています。

- セクション18.3.2.8「MySQL クラスタの TCP/IP 接続」で説明している 100M ビット/秒またはギガビット Ethernet を使用した TCP/IP。
- ダイレクト (マシン間) TCP/IP。このトランスポータは前の項目で示したものと同じ TCP/IP プロトコルを使用しますが、ハードウェアを異なる方法でセットアップする必要があり、構成方法も異なります。このため、MySQL Cluster では別個のトランスポートメカニズムとみなされています。詳細は、セクション18.3.2.9「直接接続を使用する MySQL Cluster の TCP/IP 接続」を参照してください。
- 共有メモリー (SHM)。SHM の詳細は、セクション18.3.2.10「MySQL Cluster の共有メモリー接続」を参照してください。

  > **注記**
  >
  > SHM はあくまで実験的なものであり、正式にはサポートされません。

- この章の次のセクション (セクション18.3.2.11「MySQL Cluster での SCI トランスポート接続」) で説明するスケーラブルコヒーラントインタフェース (SCI)。

ほとんどのユーザーは現在、広く普及していることから、TCP/IP over Ethernet を採用しています。TCP/IP は、MySQL Cluster での使用に関してもっともよくテストされているトランスポータでもあります。

弊社は、あらゆるタイプのデータ転送にメリットがあるため、ndbd プロセスとの通信ができるだけ大きな「チャンク」で行われるように努力しています。

ユーザーが希望する場合は、クラスタのインターコネクトを使用してパフォーマンスをさらに向上させることもできます。これには 2 つの実現方法があります。このケースを処理できるようにカスタムトランスポータを設計するか、TCP/IP スタックをある程度バイパスするソケット実装を使用できます。弊社は、Dolphin Interconnect Solutions が開発した SCI (スケーラブルコヒーラントインタフェース) テクノロジを使用して、両方の手法を試しました。

18.3.5.1 SCI ソケットを使用する MySQL Cluster の構成

スケーラブルコヒーラントインタフェース (SCI) テクノロジを採用して、MySQL Cluster データノードと SQL ノード間の接続の速度とスループットを大幅に向上させることができます。SCI を使用するには、Dolphin SCI ネットワークカードを取り付けて、Dolphin から提供されるドライバおよびその他のソフトウェアを使用する必要があります。これらの入手方法については、Dolphin Interconnect Solutions を参照してください。SCI SuperSocket または SCI トランスポータは、32 ビットおよび 64 ビットの Linux、Solaris、Windows、およびその他のプラットフォームでサポートされます。SCI でサポートされるプラットフォームの詳細は、このセクションの後半で参照する Dolphin のドキュメントを参照してください。

必要な Dolphin のハードウェアおよびソフトウェアの入手後、通常の TCP/IP 通信用に構成された MySQL Cluster を調整して SCI を使用する方法に関する詳しい情報を Dolphin SCI online documentation から入手できます。

18.3.5.2 MySQL Cluster のインターコネクトとパフォーマンス

ndbd プロセスには、MySQL Cluster 内のデータへのアクセスに使用するいくつかの簡単な構造があります。弊社は、非常に簡単なベンチマークを作成して、各パフォーマンスと、さまざまなインターコネクトがパフォーマンスに与える影響をチェックしました。

4 つのアクセス方法があります。

- 主キーアクセス　これは、主キーを介したレコードへのアクセスです。もっとも簡単なケースでは、一度に 1 つのレコードだけにアクセスします。これにより、多数の TCP/IP メッセージを設定するためのコスト全体とコンテキストスイッチのための多くのコストが、この単一の要求によって発生します。1 つのバッチで複数の主キーアクセスを送信するケースでは、必要な TCP/IP メッセージとコンテキストスイッチの設定コストがこのアクセスによって共有されます。TCP/IP メッセージの宛先が異なる場合は、追加の TCP/IP メッセージを設定する必要があります。
- 一意キーアクセス　一意キーアクセスは主キーアクセスとほぼ同じですが、一意キーアクセスはインデックステーブルの読み取りとして実行され、そのあとでテーブルに対する主キーアクセスが実行されます。ただし、MySQL Server からは 1 つの要求だけが送信され、インデックステーブルの読み取りは ndbd によって処理されます。このような要求でも、バッチ化のメリットが得られます。

- フルテーブルスキャン　テーブルにルックアップ用のインデックスが存在しない場合は、フルテーブルスキャンが実行されます。これは ndbd プロセスに単一の要求として送信されます。その後、このプロセスはテーブルスキャンをクラスタすべての ndbd プロセスで実行される一連の並列スキャンに分割します。MySQL Cluster の今後のバージョンでは、SQL ノードがこのスキャンの一部をフィルタできるようになります。
- 順序付けされたインデックスを使用する範囲スキャン　順序付けされたインデックスを使用すると、フルテーブルスキャンと同じ方法でスキャンが実行されますが、MySQL サーバー (SQL ノード) によって送信されたクエリーの使用範囲内のレコードのみがスキャンされます。バインドされたインデックス属性にパーティション化キー内のすべての属性が含まれる場合は、すべてのパーティションが並列でスキャンされます。

簡単なバッチ化された主キーおよび一意キーアクセスのテストで MySQL が内部で開発したベンチマークを実施したところ、通信のパフォーマンスが問題にならないまれなケースを除き、SCI ソケットの使用によってパフォーマンスが TCP/IP に比べておよそ 100% 向上することがわかりました。これは、スキャンフィルタが処理時間の大部分を占めるときや、主キーアクセスの非常に大きなバッチが実行されたときに起きる可能性があります。その場合は、ndbd プロセス内の CPU 処理がオーバーヘッドのかなり大きな部分を占めています。

SCI ソケットの代わりに SCI トランスポータを使用することは、ndbd プロセス間の通信でのみメリットがあります。また、SCI トランスポータを使用すると ndbd プロセスがスリープ状態にならないため、SCI トランスポータの使用でメリットがあるのは、CPU をこのプロセス専用にできる場合だけです。ndbd プロセスが長期間にわたる実行 (Linux 2.6 ではプロセスを CPU にロックすることで実現可能です) によって優先度を失わない方法でこのプロセスの優先度を設定することも重要になります。このような構成が可能な場合、ndbd プロセスのパフォーマンスは SCI ソケットを使用する場合に比べて 10-70% 向上します。(更新の実行時 (あるいは並列スキャン操作時) は、この数値がより大きくなります。)

ほかにもコンピュータクラスタ用に最適化されたソケット実装がいくつか存在します (Myrinet、ギガビット Ethernet、Infiniband、VIA インタフェースなど)。ただし、現在のところ MySQL Cluster のテストで使用したのは SCI ソケットだけです。通常の TCP/IP を使用して SCI ソケットを MySQL Cluster 用に設定する方法については、セクション18.3.5.1「SCI ソケットを使用する MySQL Cluster の構成」を参照してください。

18.4 MySQL Cluster プログラム

MySQL Cluster を使用および管理するには、この章で説明する各種の専用のプログラムが必要となります。MySQL Cluster でのこれらのプログラムの目的、プログラムの使用方法、および各プログラムで使用可能な起動オプションについて説明します。

これらのプログラムには、MySQL Cluster データ、管理、および SQL ノードプロセス (ndbd、ndbmtd、ndb_mgmd、および mysqld)、および管理クライアント (ndb_mgm) が含まれます。

MySQL Cluster Auto-Installer を開始するために使用されるプログラム ndb_setup.py (MySQL Cluster NDB 7.3.1 で追加されました) についての情報も、このセクションに含まれています。セクション18.4.23「ndb_setup.py — MySQL Cluster のブラウザベース自動インストーラの開始」には、コマンド行クライアントに関する情報のみが含まれています。このプログラムから起動される GUI インストーラを使用して MySQL Cluster を構成および配備するための情報については、セクション18.2.1「MySQL Cluster Auto-Installer」を参照してください。

MySQL Cluster プロセスとしての mysqld の使用については、セクション18.5.4「MySQL Cluster での MySQL サーバーの使用法」を参照してください。

その他の NDB ユーティリティー、診断プログラム、およびプログラム例は、MySQL Cluster 配布に含まれています。これらには、ndb_restore、ndb_show_tables、および ndb_config が含まれます。これらのプログラムについてもこのセクションで説明します。

このセクションの最後の部分には、さまざまな MySQL Cluster プログラムのすべてに共通するオプションの表が含まれています。

18.4.1 ndbd — MySQL Cluster データノードデーモン

ndbd は、NDB Cluster ストレージエンジンを使用してテーブル内のすべてのデータを処理するために使用されるプロセスです。これは、分散型トランザクションの処理、ノードのリカバリ、ディスクでのチェックポイントの実行、オンラインバックアップ、および関連するタスクをデータノードが行うことができるようにするプロセスです。

MySQL Cluster では、一連の ndbd プロセスがデータを処理するために連携して動作します。これらのプロセスは、同じコンピュータ (ホスト) 上または別個のコンピュータ上で実行できます。データノードと Cluster ホストの通信は詳細に構成できます。

次の表には、MySQL Cluster データノードプログラム ndbd に固有のコマンドオプションが含まれています。追加説明が表のあとにあります。ほとんどの MySQL Cluster プログラム (ndbd を含む) に共通するオプションにつ

いては、セクション18.4.27「MySQL Cluster プログラムに共通するオプション — MySQL Cluster プログラムに共通するオプション」を参照してください。

表 18.77 この表は、ndbd プログラムのコマンド行オプションについて説明しています

形式	説明	追加または削除
--initial	ndbd の初期起動を実行します (ファイルシステムのクリーニングを含む)。このオプションを使用する前にドキュメントを参照してください	すべての MySQL 5.6 ベースリリース
--nostart、-n	ndbd をすぐに起動しません。ndbd は ndb_mgm からの起動するコマンドを待機します	すべての MySQL 5.6 ベースリリース
--daemon、-d	ndbd をデーモンとして開始します (デフォルト)。オーバーライドするには --nodaemon を指定します	すべての MySQL 5.6 ベースリリース
--nodaemon	ndbd をデーモンとして開始しません。テストのために提供されています	すべての MySQL 5.6 ベースリリース
--foreground	ndbd をフォアグラウンドで実行します。デバッグのために提供されています (--nodaemon の意味を含みます)	すべての MySQL 5.6 ベースリリース
--nowait-nodes=list	これらのデータノードが起動されるのを待機しません (ノード ID のカンマ区切りのリストを取ります)。--ndb-nodeid も使用する必要があります。	すべての MySQL 5.6 ベースリリース
--initial-start	部分的な初期起動を実行します (--nowait-nodes が必要です)	すべての MySQL 5.6 ベースリリース
--bind-address=name	ローカルバインドアドレス	すべての MySQL 5.6 ベースリリース
--install[=name]	データノードプロセスを Windows サービスとしてインストールするために使用します。Windows 以外のプラットフォームには該当しません。	すべての MySQL 5.6 ベースリリース
--remove[=name]	以前 Windows サービスとしてインストールされたデータノードプロセスを削除するために使用します。Windows 以外のプラットフォームには該当しません。	すべての MySQL 5.6 ベースリリース
--connect-retries=#	管理サーバーへの接続を試行する回数。無限に試行する場合は -1 を設定します	すべての MySQL 5.6 ベースリリース
--connect-delay=#	管理サーバーへの各接続試行間に待機する時間 (秒単位)	すべての MySQL 5.6 ベースリリース

> **注記**
>
> これらのすべてのオプションはこのプログラムのマルチスレッドバージョン (ndbmtd) にも適用され、「ndbd」 を 「ndbmtd」 に置き換えることができます (このセクションで後者が出てくる場合)。

- --bind-address

コマンド行形式	--bind-address=name
型	文字列
デフォルト	

ndbd が特定のネットワークインタフェース (ホスト名または IP アドレス) にバインドされます。このオプションにはデフォルト値はありません。

- --daemon、-d

コマンド行形式	--daemon
型	ブール
デフォルト	TRUE

ndbd または ndbmtd にデーモンプロセスとして実行するように指示します。これはデフォルトの動作です。--nodaemon を使用すると、プロセスがデーモンとして実行されなくなります。

Windows プラットフォームで ndbd または ndbmtd を実行している場合、このオプションは効果がありません。

- --nodaemon

コマンド行形式	--nodaemon
型	ブール
デフォルト	FALSE

ndbd または ndbmtd がデーモンプロセスとして実行されなくなります。このオプションは --daemon オプションをオーバーライドします。これは、バイナリをデバッグするときに、出力を画面にリダイレクトする場合に便利です。

Windows での ndbd および ndbmtd のデフォルト動作はフォアグラウンドでの実行であり、Windows プラットフォームではこのオプションは効果がなく、必要ありません。

- --foreground

コマンド行形式	--foreground
型	ブール
デフォルト	FALSE

主にデバッグのために、ndbd または ndbmtd がフォアグラウンドプロセスとして実行されます。このオプションは --nodaemon オプションの意味を含みます。

Windows プラットフォームで ndbd または ndbmtd を実行している場合、このオプションは効果がありません。

- --initial

コマンド行形式	--initial
型	ブール

デフォルト	FALSE

ndbd に初期起動を実行するように指示します。初期起動によって、以前の ndbd のインスタンスでリカバリのために作成されたファイルが消去されます。また、リカバリログファイルが再作成されます。一部のオペレーティングシステムでは、この処理にかなり長い時間がかかります。

--initial を指定した起動は、ndbd プロセスを非常に特殊な状況で起動するときにのみ使用します。これは、このオプションを指定すると、MySQL Cluster ファイルシステムからすべてのファイルが削除され、すべての Redo ログファイルが再作成されるためです。それらの状況を次に示します。

- ファイルの内容が変更されたソフトウェアアップグレードを実行する場合。
- 新しいバージョンの ndbd でノードを再起動する場合。
- 何かの理由でノードまたはシステムの再起動が繰り返し失敗する場合の最後の手段として。この場合、データファイルが破壊されるため、このノードはデータのリストアに使用できなくなります。

このオプションを使用すると、StartPartialTimeout および StartPartitionedTimeout 構成パラメータの効果がなくなります。

> **重要**
>
> このオプションは、次のタイプのファイルには影響しません。
>
> - 影響を受けるノードによってすでに作成されているバックアップファイル
> - MySQL Cluster ディスクデータファイル (セクション18.5.12「MySQL Cluster ディスクデータテーブル」を参照してください)。
>
> このオプションは、すでに実行されているデータノードから起動 (または再起動) されたばかりのデータノードによるデータのリカバリにも影響しません。このデータのリカバリは自動的に行われ、正常に実行されている MySQL Cluster でユーザーが操作を行う必要はありません。

クラスタを最初に起動するとき (つまり、データノードファイルが作成される前) にこのオプションを使用することはできますが、そうする必要はありません。

- --initial-start

コマンド行形式	--initial-start
型	ブール
デフォルト	FALSE

このオプションは、クラスタを部分的に初期起動するときに使用します。各ノードは、このオプションおよび --nowait-nodes を指定して起動してください。

データノードの ID が 2、3、4、および 5 である 4 ノードクラスタがあり、ノード 2、4、および 5 のみを使用して (つまり、ノード 3 を除外して) 部分的に初期起動を実行するとします。

```
shell> ndbd --ndb-nodeid=2 --nowait-nodes=3 --initial-start
shell> ndbd --ndb-nodeid=4 --nowait-nodes=3 --initial-start
shell> ndbd --ndb-nodeid=5 --nowait-nodes=3 --initial-start
```

このオプションを使用する場合は、--ndb-nodeid オプションを指定して起動するデータノードのノード ID も指定する必要があります。

> **重要**
>
> 複数の管理サーバーで構成されたクラスタをすべての管理サーバーがオンラインでなくても起動できるようにするために使用できる ndb_mgmd の --nowait-nodes オプションとこのオプションを混同しないでください。

- --nowait-nodes=node_id_1[, node_id_2[, ...]]

コマンド行形式	--nowait-nodes=list
型	文字列

デフォルト	

このオプションには、クラスタが起動前に待機しないデータノードのリストを指定します。

これは、パーティション化された状態のクラスタを起動するために使用できます。たとえば、4 ノードクラスタで実行されているデータノード (ノード 2、3、4、および 5) の半分のデータノードのみでクラスタを起動するには、--nowait-nodes=3,5 を指定して各 ndbd プロセスを開始します。この場合、クラスタはノード 2 およびノード 4 が接続するとすぐに起動し、ノード 3 およびノード 5 が接続するのを (接続していない場合) StartPartitionedTimeout ミリ秒間待機しません。

前の例と同じクラスタを 1 つの ndbd を除外して起動する場合 (たとえば、ノード 3 のホストマシンでハードウェアの障害が発生している場合) は、--nowait-nodes=3 を指定してノード 2、4、および 5 を 起動します。クラスタはノード 2、4、および 5 が接続するとすぐに起動し、ノード 3 が起動されるのを待機しません。

- --nostart、-n

コマンド行形式	--nostart
型	ブール
デフォルト	FALSE

ndbd に自動的に起動しないように指示します。このオプションを使用すると、ndbd は管理サーバーに接続してそこから構成データを取得し、通信オブジェクトを初期化します。ただし、管理サーバーによってそうするように明示的に要求されるまで、実行エンジンを実際に起動しません。これは、管理クライアントで適切な START コマンドを発行することによって実現されます (セクション18.5.2「MySQL Cluster 管理クライアントのコマンド」を参照してください)。

- --install[=name]

コマンド行形式	--install[=name]
プラットフォーム固有	Windows
型	文字列
デフォルト	ndbd

ndbd が Windows サービスとしてインストールされます。必要に応じて、サービス名を指定できます。設定しない場合、サービス名は ndbd にデフォルト設定されます。ほかの ndbd プログラムオプションは my.ini または my.cnf 構成ファイルに指定することが推奨されますが、--install と一緒に使用できます。ただし、そのような場合、Windows サービスのインストールが成功するには、--install オプションを最初に指定してから、ほかのオプションを指定する必要があります。

一般的に、このオプションを --initial オプションと一緒に使用することはお勧めしません。それにより、サービスが停止および開始されるたびにデータノードファイルシステムが消去および再作成されるためです。データノードの起動に影響するほかの ndbd オプション (--initial-start、--nostart、および --nowait-nodes を含む) を --install と一緒に使用する場合は、それを行うことを十分に理解し、起こり得る結果に対して十分に準備していることをしっかりと確認してください。

--install オプションは、Windows 以外のプラットフォームでは効果がありません。

- --remove[=name]

コマンド行形式	--remove[=name]
プラットフォーム固有	Windows
型	文字列
デフォルト	ndbd

以前に Windows サービスとしてインストールされた ndbd プロセスを削除します。必要に応じて、アンインストールするサービスの名前を指定できます。設定しない場合、サービス名は ndbd にデフォルト設定されます。

--remove オプションは、Windows 以外のプラットフォームでは効果がありません。

- --connect-retries=#

コマンド行形式	--connect-retries=#

型	数値
デフォルト	12
最小値	-1
最大値	65535

データノードが起動時に管理サーバーに接続を試みる回数を指定します。このオプションに -1 を設定すると、データノードが無限に接続を試みます。デフォルトは 12 回の試行です。各試行の間に待機する時間は、--connect-delay オプションによって制御されます。

- --connect-delay=#

コマンド行形式	--connect-delay=#
型	数値
デフォルト	5
最小値	0
最大値	3600

起動時に管理サーバーへの各接続試行の間に待機する時間を決定します (試行の間の時間は --connect-retries オプションによって制御されます)。デフォルトは 5 回の試行です。

このオプションは MySQL Cluster NDB 7.2.9 で追加されました。

ndbd は、config.ini 構成ファイルの DataDir で指定されているディレクトリに配置される一連のログファイルを生成します。

これらのログファイルを次に一覧します。node_id はノードの一意識別子です。node_id はノードの一意識別子を表します。たとえば、ndb_2_error.log はノード ID が 2 のデータノードによって生成されたエラーログです。

- ndb_node_id_error.log は、参照されている ndbd プロセスで発生したすべてのクラッシュの記録が含まれているファイルです。このファイルの各レコードには、簡単なエラー文字列、およびこのクラッシュのトレースファイルへの参照が含まれています。このファイルの一般的なエントリを次に示します。

```
Date/Time: Saturday 30 July 2004 - 00:20:01
Type of error: error
Message: Internal program error (failed ndbrequire)
Fault ID: 2341
Problem data: DbtupFixAlloc.cpp
Object of reference: DBTUP (Line: 173)
ProgramName: NDB Kernel
ProcessID: 14909
TraceFile: ndb_2_trace.log.2
***EOM***
```

データノードプロセスが予期せずに停止されたときに生成される可能性がある ndbd の終了コードおよびメッセージのリストは、Data Node Error Messagesにあります。

> **重要**
>
> このエラーログファイルの最後のエントリが最新のエントリであるとはかぎりません (その可能性もあまりありません)。このエラーログのエントリは、時系列順で一覧されず、ndb_node_id_trace.log.next ファイル (次を参照してください) で決定されるトレースファイルの順序に対応しています。エラーログのエントリは、このように循環的に上書きされ、順次的ではありません。

- ndb_node_id_trace.log.trace_id は、エラーが発生する直前に発生した事象を示すトレースファイルです。この情報は MySQL Cluster 開発チームが分析する場合に役に立ちます。

  古いファイルが上書きされる前に作成されるこれらのトレースファイルの数は構成できます。trace_id は連続する各トレースファイルで増分される数です。

- ndb_node_id_trace.log.next は、割り当てられる次のトレースファイル番号を追跡するファイルです。

- ndb_node_id_out.log は、ndbd プロセスによるデータ出力が含まれているファイルです。このファイルは、ndbd がデーモンとして開始された (デフォルトの動作) 場合にのみ作成されます。

- ndb_node_id.pid には、ndbd プロセスがデーモンとして開始された場合のプロセス ID が含まれています。これは、同じ識別子でノードが起動されるのを防ぐための ロックファイルとしても機能します。
- ndb_node_id_signal.log は、ndbd のデバッグバージョンでのみ使用されるファイルであり、ndbd プロセスのそれらのデータのすべての受信、送信、および内部メッセージをトレースできます。

NFS を使用してマウントしたディレクトリは使用しないことをお勧めします。一部の環境では、プロセスが終了しても .pid ファイルに対するロックが有効なままとなる問題が発生することがあるためです。

ndbd を開始する場合、管理サーバーのホスト名およびそれが待機するポートも指定する必要があることがあります。必要に応じて、プロセスが使用するノード ID を指定することもできます。

```
shell> ndbd --connect-string="nodeid=2;host=ndb_mgmd.mysql.com:1186"
```

これについての詳細は、セクション18.3.2.3「MySQL Cluster の接続文字列」を参照してください。セクション18.4.27「MySQL Cluster プログラムに共通するオプション — MySQL Cluster プログラムに共通するオプション」では、ndbd で使用できるその他のコマンド行オプションについて説明しています。データノード構成パラメータについては、セクション18.3.2.6「MySQL Cluster データノードの定義」を参照してください。

ndbd が開始されると、実際には 2 つのプロセスが開始されます。最初のプロセスは「エンジェルプロセス」と呼ばれ、その唯一の役割は実行プロセスが完了したときにそれを検出し、ndbd プロセスを再起動することです (そのように構成されている場合)。このため、UNIX の kill コマンドを使用して ndbd を強制終了しようとする場合は、両方のプロセスを強制終了する必要があります (エンジェルプロセスが最初)。ndbd プロセスを終了させるための推奨される方法は、管理クライアントを使用してそこからプロセスを停止することです。

実行プロセスは、データの読み取り、書き込み、スキャン、およびほかのすべてのアクティビティーに 1 つのスレッドを使用します。数千の同時アクションを容易に処理できるように、このスレッドは非同期に実装されます。また、監視スレッドは実行スレッドが無限ループでハングアップしないように管理します。スレッドのプールによってファイル I/O が処理され、各スレッドは 1 つのオープンファイルを処理できます。スレッドは ndbd プロセスのトランスポーターによるトランスポーター接続に使用することもできます。多数の操作 (更新を含む) を実行するマルチプロセッサーシステムの場合、ndbd プロセスは最大 2 つの CPU を使用できます (許可されている場合)。

多数の CPU を持つマシンの場合は、異なるノードグループに属する複数の ndbd プロセスを使用できます。ただし、そのような構成はまだ実験的と見なされ、MySQL 5.6 の本番設定ではサポートされません。セクション18.1.6「MySQL Cluster の既知の制限」を参照してください。

18.4.2 ndbinfo_select_all — ndbinfo テーブルからの選択

ndbinfo_select_all は、ndbinfo データベースの 1 つ以上のテーブルからすべての行およびカラムを選択するクライアントプログラムです。これは、MySQL Cluster NDB 7.2.2 から MySQL Cluster 配布に含められました。

このプログラムはすべての ndbinfo テーブルにアクセスできるわけではありません。ndbinfo_select_all は、counters、diskpagebuffer、logbuffers、logspaces、nodes、resources、threadblocks、threadstat、および transporters テーブルにアクセスできます。

ndbinfo_select_all を使用して 1 つ以上の ndbinfo テーブルから選択するには、プログラムを呼び出すときに次のようにテーブル名を指定する必要があります。

```
shell> ndbinfo_select_all table_name1  [table_name2] [...]
```

例:

```
shell> ndbinfo_select_all logbuffers logspaces
== logbuffers ==
node_id log_type      log_id  log_part      total   used    high
5       0     0       33554432        262144  0
6       0     0       33554432        262144  0
7       0     0       33554432        262144  0
8       0     0       33554432        262144  0
== logspaces ==
node_id log_type      log_id  log_part      total   used    high
5       0     0       268435456       0       0
5       0     1       268435456       0       0
5       0     2       268435456       0       0
5       0     3       268435456       0       0
6       0     0       268435456       0       0
6       0     1       268435456       0       0
6       0     2       268435456       0       0
6       0     3       268435456       0       0
7       0     0       268435456       0       0
7       0     1       268435456       0       0
```

```
7    0    0    2    268435456    0    0
7    0    0    3    268435456    0    0
8    0    0    0    268435456    0    0
8    0    0    1    268435456    0    0
8    0    0    2    268435456    0    0
8    0    0    3    268435456    0    0
shell>
```

次の表には、ndbinfo_select_all に固有のオプションが含まれています。追加説明が表のあとにあります。ほとんどの MySQL Cluster プログラム (ndbinfo_select_all を含む) に共通するオプションについては、セクション 18.4.27「MySQL Cluster プログラムに共通するオプション — MySQL Cluster プログラムに共通するオプション」を参照してください。

表 18.78 この表は、ndbinfo_select_all プログラムのコマンド行オプションについて説明しています

形式	説明	追加または削除
--delay=#	ループ間の遅延 (秒単位) を設定します。デフォルトは 5 です。	すべての MySQL 5.6 ベースリリース
--loops=#、-l	選択を実行する回数を設定します。デフォルトは 1 です。	すべての MySQL 5.6 ベースリリース
--database=db_name、-d	テーブルが存在するデータベースの名前。	すべての MySQL 5.6 ベースリリース
--parallelism=#、-p	並列性の度合いを設定します。	すべての MySQL 5.6 ベースリリース

- --delay=seconds

コマンド行形式	--delay=#
型	数値
デフォルト	5
最小値	0
最大値	MAX_INT

このオプションは、ループの各実行の間に待機する秒数を設定します。--loops に 0 または 1 が設定されている場合は効果がありません。

- --loops=number、-l number

コマンド行形式	--loops=#
型	数値
デフォルト	1
最小値	0
最大値	MAX_INT

このオプションは選択を実行する回数を設定します。各ループの時間間隔を設定するには、--delay を使用します。

18.4.3 ndbmtd — MySQL Cluster データノードデーモン (マルチスレッド)

ndbmtd は ndbd のマルチスレッドバージョンであり、NDBCLUSTER ストレージエンジンを使用してテーブル内のすべてのデータを処理するために使用するプロセスです。ndbmtd は複数の CPU コアを持つホストコンピュータで使用することが意図されています。特に記載されていない場合、ndbmtd は ndbd と同様に機能します。このため、このセクションでは、ndbmtd が ndbd と異なる点に重点を置き、シングルスレッドバージョンとマルチスレッドバージョンの両方のデータノードプロセスに該当する MySQL Cluster のデータノードの実行の詳細については、セクション18.4.1「ndbd — MySQL Cluster データノードデーモン」を参照してください。

ndbd で使用されるコマンド行オプションおよび構成パラメータは、ndbmtd にも当てはまります。これらのオプションおよびパラメータについては、それぞれセクション18.4.1「ndbd — MySQL Cluster データノードデーモン」およびセクション18.3.2.6「MySQL Cluster データノードの定義」を参照してください。

また、ndbmtd は ndbd とファイルシステム互換です。言い換えると、ndbd を実行しているデータノードを停止し、バイナリを ndbmtd に置き換えて、データ損失なしに再起動できます(ただし、これを行うときに、ndbmtd をマルチスレッドで実行する場合は、ノードを再起動する前に、MaxNoOfExecutionThreads に適切な値が設定されていることを確認する必要があります)。同様に、ノードを停止して、マルチスレッドバイナリの代わりに ndbd を開始することによって、ndbmtd バイナリを ndbd に置き換えることができます。これらの 2 つを切り替えるときに、--initial を使用してデータノードバイナリを開始する必要はありません。

ndbmtd を使用した場合は、ndbd を使用した場合と 2 つの重要な点で異なります。

1. ndbmtd はデフォルトではシングルスレッドモードで実行されるため (つまり、ndbd のように動作します)、マルチスレッドを使用するように構成する必要があります。これを行うには、config.ini ファイルの MaxNoOfExecutionThreads 構成パラメータまたは ThreadConfig 構成パラメータに適切な値を設定します。MaxNoOfExecutionThreads は簡単に使用できますが、ThreadConfig にはより柔軟性があります。これらの構成パラメータおよびその使用方法については、マルチスレッドの構成パラメータ (ndbmtd)を参照してください。

2. トレースファイルは ndbmtd プロセスで重大なエラーが発生したときに生成されますが、ndbd で障害が発生したときにこれらが生成される仕組みとは若干異なります。これらの相違については、次のいくつかの段落で説明します。

ndbd と同様に、ndbmtd は config.ini 構成ファイルの DataDir で指定されたディレクトリに配置される一連のログファイルを生成します。トレースファイルを除き、これらは ndbd で生成されるものと同様に生成され、同じ名前が付けられます。

重大なエラーが発生した場合、ndbmtd はエラーが発生する直前に発生した事象を示すトレースファイルを生成します。データノードの DataDir にあるこれらのファイルは、MySQL Cluster の開発チームおよびサポートチームが問題を分析するために役に立ちます。各 ndbmtd スレッドに対して 1 つのトレースファイルが生成されます。これらのファイルの名前には次のパターンがあります。

```
ndb_node_id_trace.log.trace_id_tthread_id,
```

このパターンで、node_id はクラスタ内のデータノード一意のノード ID を表し、trace_id はトレースシーケンス番号であり、thread_id はスレッド ID です。たとえば、ノード ID が 3 であり、MaxNoOfExecutionThreads が 4 である MySQL Cluster データノードとして実行されている ndbmtd プロセスで障害が発生した場合は、4 つのトレースファイルがデータノードのデータディレクトリに生成されます。このノードで障害が発生したのがはじめてであった場合、これらのファイルには ndb_3_trace.log.1_t1、ndb_3_trace.log.1_t2、ndb_3_trace.log.1_t3、および ndb_3_trace.log.1_t4 と名前が付けられます。内部的には、これらのトレースファイルは ndbd トレースファイルと同じ形式です。

データノードプロセスが予期せずに停止されたときに生成される ndbd の終了コードおよびメッセージは、ndbmtd でも使用されます。これらのリストについては、Data Node Error Messagesを参照してください。

> **注記**
>
> 同じ MySQL Cluster の異なるデータノードで ndbd および ndbmtd を同時に使用できます。ただし、そのような構成は十分にテストされていないため、現時点で本番設定でこれを行うことはお勧めできません。

18.4.4 ndb_mgmd — MySQL Cluster 管理サーバーデーモン

管理サーバーは、クラスタ構成ファイルを読み取り、この情報を要求したクラスタ内のすべてのノードにそれを配布するプロセスです。また、これはクラスタのアクティビティーに関するログを管理します。管理クライアントは、管理サーバーに接続してクラスタのステータスをチェックできます。

次の表には、MySQL Cluster 管理サーバープログラム ndb_mgmd に固有のオプションが含まれています。追加説明が表のあとにあります。ほとんどの MySQL Cluster プログラム (ndb_mgmd を含む) に共通するオプションについては、セクション18.4.27「MySQL Cluster プログラムに共通するオプション — MySQL Cluster プログラムに共通するオプション」を参照してください。

表 18.79 この表は、ndb_mgmd プログラムのコマンド行オプションについて説明しています

形式	説明	追加または削除
--config-file=file、 -f	クラスタ構成ファイルを指定します。NDB 6.4.0 以降では、構成	すべての MySQL 5.6 ベースリース

形式	説明	追加または削除
	キャッシュをオーバーライドする (ある場合) には --reload または --initial が必要となります	
--configdir=directory、 --config-dir=directory	クラスタ管理サーバーの構成キャッシュディレクトリを指定します	すべての MySQL 5.6 ベースリリース
--bind-address=ip_address	ローカルバインドアドレス	すべての MySQL 5.6 ベースリリース
--print-full-config、 -P	すべての構成を出力して終了します	すべての MySQL 5.6 ベースリリース
--daemon、 -d	ndb_mgmd をデーモンモードで実行します (デフォルト)	すべての MySQL 5.6 ベースリリース
--nodaemon	ndb_mgmd をデーモンとして実行しません	すべての MySQL 5.6 ベースリリース
--interactive	ndb_mgmd をインタラクティブモードで実行します (本番では正式にサポートされていません。テストのためのみです)	すべての MySQL 5.6 ベースリリース
--log-name=name	このノードに適用されるメッセージをクラスタログに書き込むときに使用される名前。	すべての MySQL 5.6 ベースリリース
--no-nodeid-checks	ノード ID チェックを提供しません	すべての MySQL 5.6 ベースリリース
--mycnf	my.cnf ファイルからクラスタ構成データを読み取ります	すべての MySQL 5.6 ベースリリース
--reload	管理サーバーが構成ファイルを構成キャッシュと比較します	すべての MySQL 5.6 ベースリリース
--initial	管理サーバーが構成キャッシュをバイパスして、構成データを構成ファイルからリロードします	すべての MySQL 5.6 ベースリリース
--nowait-nodes=list	この管理サーバーを起動するときに、これらの管理ノードを待機しません。--ndb-nodeid も使用する必要があります。	すべての MySQL 5.6 ベースリリース
--config-cache[=TRUE|FALSE]	管理サーバー構成キャッシュを有効にします。デフォルトは TRUE です。	すべての MySQL 5.6 ベースリリース
--install[=name]	管理サーバープロセスを Windows サービスとしてインストールするために使用します。Windows 以外のプラットフォームには該当しません。	すべての MySQL 5.6 ベースリリース
--remove[=name]	以前 Windows サービスとしてインストールされた管理サーバープロセスを削除するために使用します。必要に応じて、削除するサービスの名前を指定します。Windows 以外のプラットフォームには該当しません。	すべての MySQL 5.6 ベースリリース

- --bind-address=host[:port]

コマンド行形式	--bind-address=ip_address
型	文字列
デフォルト	[none]

このオプションを指定すると、管理クライアントによる管理サーバーへの接続が、指定したホスト名または IP アドレス (およびポート (指定された場合)) のクライアントに制限されます。その場合、ほかのアドレスから管理サーバーに接続しようとする管理クライアントは、次のエラーが発生して失敗します。Unable to setup port: host:port!

port を指定しない場合、管理クライアントはポート 1186 を使用することを試みます。

- --no-nodeid-checks

コマンド行形式	--no-nodeid-checks
型	ブール
デフォルト	FALSE

ノード ID のチェックを実行しません。

- --configdir=path

コマンド行形式	--configdir=directory --config-dir=directory
型	ファイル名
デフォルト	$INSTALLDIR/mysql-cluster

クラスタ管理サーバーの構成キャッシュディレクトリを指定します。--config-dir はこのオプションのエイリアスです。

- --config-cache

コマンド行形式	--config-cache[=TRUE|FALSE]
型	ブール
デフォルト	TRUE

このオプション (デフォルト値は 1 (または TRUE、ON)) は、管理サーバーが起動するたびに config.ini から構成が読み取られるように (セクション18.3.2「MySQL Cluster の構成ファイル」を参照してください) その構成キャッシュを無効にするために使用できます。これを行うには、次のいずれかのオプションを指定して ndb_mgmd プロセスを開始します。

- --config-cache=0
- --config-cache=FALSE
- --config-cache=OFF
- --skip-config-cache

一覧したいずれかのオプションの使用が有効となるのは、管理サーバーが開始されたときに格納されている構成がない場合のみです。管理サーバーが構成キャッシュファイルを見つけると、--config-cache オプションまたは --skip-config-cache オプションは無視されます。このため、構成キャッシュを無効にするには、管理サーバーを最初に起動するときにこのオプションを使用してください。それ以外の場合、つまり、構成キャッシュがすでに作成されている管理サーバーの構成キャッシュを無効にする場合は、管理サーバーを停止して、既存の構成キャッシュファイルを手動で削除してから、--skip-config-cache を指定して (または --config-cache に 0、OFF、または FALSE を設定して) 管理サーバーを再起動する必要があります。

構成キャッシュファイルは通常、インストールディレクトリの下の mysql-cluster という名前のディレクトリに作成されます (--configdir オプションを使用してこの場所がオーバーライドされていない場合)。管理サーバーが

構成データを更新するたびに、新しいキャッシュファイルが生成されます。このファイルは、次の形式を使用して作成順に連続した名前が付けられます。

```
ndb_node-id_config.bin.seq-number
```

node-id は管理サーバーのノード ID であり、seq-number は 1 から始まるシーケンス番号です。たとえば、管理サーバーのノード ID が 5 の場合、最初の 3 つの構成キャッシュファイルが作成されるときに ndb_5_config.bin.1、ndb_5_config.bin.2、および ndb_5_config.bin.3 という名前が付けられます。

実際にキャッシュを無効にせずに、構成キャッシュをパージまたはリロードする場合は、--skip-config-cache オプションの代わりに --reload オプションまたは --initial オプションのいずれかを指定して ndb_mgmd を開始してください。

構成キャッシュを再度有効にするには、構成キャッシュを無効にするために前に使用した --config-cache オプションまたは --skip-config-cache オプションを指定せずに管理サーバーを再起動します。

--skip-config-cache が使用された場合、ndb_mgmd は構成ディレクトリ (--configdir) をチェックせずに作成しようとします。(Bug#13428853)

- --config-file=filename、-f filename

コマンド行形式	--config-file=file
型	ファイル名
デフォルト	[none]

構成ファイルとして使用すべきファイルを管理サーバーに指示します。デフォルトでは、管理サーバーは ndb_mgmd 実行可能ファイルと同じディレクトリで config.ini という名前のファイルを探します。それ以外の場合は、ファイル名および場所を明示的に指定する必要があります。

このオプションにはデフォルト値はなく、--reload または --initial オプションを指定して ndb_mgmd が起動されたか、管理サーバーが構成キャッシュを見つけることができなかったために、管理サーバーが構成ファイルを読み取るように強制された場合を除き、無視されます。このオプションは、--config-cache=OFF を指定して ndb_mgmd が開始された場合にも読み取られます。詳細は、セクション18.3.2「MySQL Cluster の構成ファイル」を参照してください。

以前は、このオプションを --initial と一緒に使用すると、ファイルが見つからなくても構成キャッシュが削除されました。この問題は MySQL Cluster NDB 7.3.2 で解決されました。(Bug #1299289)

- --mycnf

コマンド行形式	--mycnf
型	ブール
デフォルト	FALSE

my.cnf ファイルから構成データを読み取ります。

- --daemon、-d

コマンド行形式	--daemon
型	ブール
デフォルト	TRUE

ndb_mgmd にデーモンプロセスとして開始するように指示します。これはデフォルトの動作です。

このオプションは ndb_mgmd を Windows プラットフォームで実行している場合は効果がありません。

- --interactive

コマンド行形式	--interactive
型	ブール

デフォルト	FALSE

ndb_mgmd をインタラクティブモードで開始します。つまり、管理サーバーが実行されるとすぐに ndb_mgm クライアントセッションが開始されます。このオプションはほかの MySQL Cluster ノードを起動しません。

- --initial

コマンド行形式	--initial
型	ブール
デフォルト	FALSE

構成データは、管理サーバーが開始されるたびにクラスタグローバル構成ファイルから読み取られるのではなく、内部的にキャッシュされます (セクション18.3.2「MySQL Cluster の構成ファイル」を参照してください)。--initial オプションを使用するとこの動作がオーバーライドされ、管理サーバーが既存のキャッシュファイルを削除し、クラスタ構成ファイルから構成データを再度読み取り、新しいキャッシュを作成するように強制されます。

これは、2 つの点で --reload オプションと異なります。まず、--reload を指定すると、サーバーが構成ファイルをキャッシュと照合し、ファイルの内容がキャッシュと異なる場合にのみデータをリロードすることが強制されます。2 番目に、--reload は既存のキャッシュファイルを削除しません。

--initial を指定して ndb_mgmd が呼び出されたけれどもグローバル構成ファイルが見つからない場合、管理サーバーは起動できません。

管理サーバーが起動されると、同じ MySQL Cluster 内の別の管理サーバーをチェックし、ほかの管理サーバーの構成データを使用することを試みます。ndb_mgmd は、それが実行されている唯一の管理サーバーである場合を除き、--initial を無視します。この動作は、複数の管理ノードを持つ MySQL Cluster でローリング再起動を実行するときにも影響します。詳細は、セクション18.5.5「MySQL Cluster のローリング再起動の実行」を参照してください。

以前は、このオプションを --config-file オプションと一緒に使用すると、ファイルが見つからなくても構成キャッシュが削除されました。MySQL Cluster NDB 7.3.2 以降では、構成ファイルが実際に見つかった場合にのみ、キャッシュがそのようにクリアされます。(Bug #1299289)

- --log-name=name

コマンド行形式	--log-name=name
型	文字列
デフォルト	MgmtSrvr

クラスタログでこのノードに使用される名前を指定します。

- --nodaemon

コマンド行形式	--nodaemon
型	ブール
デフォルト	FALSE

ndb_mgmd にデーモンプロセスとして開始しないように指示します。

Windows での ndb_mgmd のデフォルト動作はフォアグラウンドでの実行であるため、Windows プラットフォームではこのオプションは必要ありません。

- --print-full-config、-P

コマンド行形式	--print-full-config
型	ブール
デフォルト	FALSE

クラスタの構成に関する詳細情報を表示します。このオプションをコマンド行に指定すると、ndb_mgmd プロセスはクラスタ設定に関する情報 (クラスタ構成セクションの詳細なリスト、およびパラメータとその値を含む) を出力します。通常、--config-file (-f) オプションと一緒に使用します。

- --reload

コマンド行形式	--reload
型	ブール
デフォルト	FALSE

MySQL Cluster NDB 7.3 では、構成データは、管理サーバーが起動されるたびにクラスタグローバル構成ファイルから読み取られるのではなく、内部に格納されます (セクション18.3.2「MySQL Cluster の構成ファイル」を参照してください)。このオプションを使用すると、管理サーバーが内部データストアをクラスタ構成ファイルと照合し、構成ファイルがキャッシュと一致しないことが判明した場合は、構成をリロードすることを強制します。既存の構成キャッシュファイルは維持されますが使用されません。

これは、2 つの点で --initial オプションと異なります。最初に、--initial ではすべてのキャッシュファイルが削除されます。2 番目に、--initial は、グローバル構成ファイルを再度読み取って新しいキャッシュを作成することを管理サーバーに強制します。

管理サーバーがグローバル構成ファイルを見つけられない場合、--reload オプションは無視されます。

管理サーバーが起動されると、同じ MySQL Cluster 内の別の管理サーバーをチェックして、ほかの管理サーバーの構成データを使用することを試みます。ndb_mgmd は、それが実行されている唯一の管理サーバーである場合を除き、--reload を無視します。この動作は、複数の管理ノードを持つ MySQL Cluster でローリング再起動を実行するときにも影響します。詳細は、セクション18.5.5「MySQL Cluster のローリング再起動の実行」を参照してください。

- --nowait-nodes

コマンド行形式	--nowait-nodes=list
型	数値
デフォルト	
最小値	1
最大値	255

MySQL Cluster は、起動されると 2 つの管理ノードで構成され、通常、各管理サーバーは別の ndb_mgmd も動作しているかどうか、および別の管理サーバーの構成がそれ自体の構成と同じであるかどうかを確認します。ただし、クラスタを 1 つの管理ノードのみで起動すること (およびおそらく別の ndb_mgmd をあとで起動することを許可すること) が望ましい場合があります。このオプションを指定すると、このオプションに渡されたノード ID を持つ別の管理ノードを管理ノードがチェックせず、起動された管理ノードのみを使用するように構成されているかのようにクラスタが起動することを許可します。

説明のために、config.ini ファイルに次の部分があるとします (ここでは、この例に関連しないほとんどの構成パラメータを省略しています)。

```
[ndbd]
NodeId = 1
HostName = 192.168.0.101

[ndbd]
NodeId = 2
HostName = 192.168.0.102

[ndbd]
NodeId = 3
HostName = 192.168.0.103

[ndbd]
NodeId = 4
HostName = 192.168.0.104

[ndb_mgmd]
NodeId = 10
HostName = 192.168.0.150

[ndb_mgmd]
NodeId = 11
HostName = 192.168.0.151

[api]
NodeId = 20
```

```
HostName = 192.168.0.200

[api]
NodeId = 21
HostName = 192.168.0.201
```

ノード ID が 10 で、IP アドレスが 192.168.0.150 のホストで動作する管理サーバーのみを使用して、このクラスタを起動するとします。(たとえば、別の管理サーバーを実行する予定のホストコンピュータがハードウェア障害のために一時的に使用できず、それが修復されるのを待っていると想定してください)。このようにクラスタを起動するには、192.168.0.150 のマシンのコマンド行を使用して、次のコマンドを入力します。

```
shell> ndb_mgmd --ndb-nodeid=10 --nowait-nodes=11
```

前の例に示されているように、--nowait-nodes を使用する場合は、--ndb-nodeid オプションも使用して、この ndb_mgmd プロセスのノード ID を指定する必要があります。

その後、クラスタの各データノードを通常の方法で起動できます。最初の管理サーバーに加えて、あとでデータノードを再起動せずに 2 番目の管理サーバーを起動して使用する場合は、両方の管理サーバーを参照する接続文字列を次のように指定して、各データノードを起動する必要があります。

```
shell> ndbd -c 192.168.0.150,192.168.0.151
```

mysqld プロセスのうち、このクラスタに接続される MySQL Cluster SQL ノードとして起動するものに使用される接続文字列に関しても同様です。詳細は、セクション18.3.2.3「MySQL Cluster の接続文字列」を参照してください。

ndb_mgmd とともに使用すると、このオプションは別の管理ノードに関してのみ管理ノードの動作に影響します。ndbd または ndbmtd (全量より少ない数のデータノードでクラスタを起動できる) に使用される --nowait-nodes オプションと混同しないでください。このオプションをデータノードに使用した場合は、ほかのデータノードに関係する動作にのみ影響します。

複数の管理ノード ID は、カンマ区切りリストとしてこのオプションに渡すことができます。各ノード ID は、1 から 255 までである必要があります。実際には、同じ MySQL Cluster に 3 つ以上の管理サーバーを使用することは (または、そのようにする必要性があることは) きわめてまれです。ほとんどの場合、このオプションに渡す必要があるのは、クラスタの起動時に使用しない 1 つの管理サーバーの単一ノード ID のみです。

> **注記**
>
> 「欠けている」管理サーバーをあとで起動する場合は、クラスタですでに使用されている管理サーバーと構成が一致している必要があります。そうしないと、既存の管理サーバーによって実行される構成チェックで失敗して起動されません。

管理サーバーを起動するときに、接続文字列を必ず指定する必要があるわけではありません。ただし、複数の管理サーバーを使用している場合は、接続文字列を指定して、クラスタ内の各ノードにノード ID を明示的に指定してください。

接続文字列の使用方法については、セクション18.3.2.3「MySQL Cluster の接続文字列」を参照してください。セクション18.4.4「ndb_mgmd — MySQL Cluster 管理サーバーデーモン」では、ndb_mgmd のその他のオプションについて説明しています。

次のファイルは、起動ディレクトリにある ndb_mgmd によって作成または使用され、config.ini 構成ファイルに指定されている DataDir に配置されます。次のリストでは、node_id は一意のノード識別子です。

- config.ini はクラスタ全体の構成ファイルです。このファイルはユーザーが作成し、管理サーバーによって読み取られます。セクション18.3「MySQL Cluster の構成」では、このファイルをセットアップする方法について説明しています。
- ndb_node_id_cluster.log はクラスタイベントログファイルです。そのようなイベントの例としては、チェックポイントの開始と完了、ノードの起動イベント、ノードの障害、およびメモリー使用率のレベルが含まれます。クラスタイベントの完全なリストおよびその説明は、セクション18.5「MySQL Cluster の管理」にあります。

  クラスタログのサイズが百万バイトに達すると、ファイルがndb_node_id_cluster.log.seq_id に名前変更されます。ここで、seq_id はクラスタログファイルのシーケンス番号です(たとえば、シーケンス番号 1、2、3 を持つファイルがすでに存在する場合、次の口グファイルは番号 4 を使用して名前が付けられます)。
- ndb_node_id_out.log は、管理サーバーがデーモンとして実行されているときに、stdout および stderr のために使用されるファイルです。

- ndb_node_id.pid は、管理サーバーをデーモンとして実行しているときに使用されるプロセス ID ファイルです。

- --install[=name]

コマンド行形式	--install[=name]
プラットフォーム固有	Windows
型	文字列
デフォルト	ndb_mgmd

ndb_mgmd が Windows サービスとしてインストールされます。必要に応じて、サービスの名前を指定できます。設定しない場合、サービス名は ndb_mgmd にデフォルト設定されます。その他の ndb_mgmd プログラムオプションは my.ini または my.cnf 構成ファイルに指定することが推奨されますが、--install と一緒に使用できます。ただし、そのような場合、Windows サービスのインストールが成功するには、--install オプションを最初に指定してから、ほかのオプションを指定する必要があります。

このオプションを --initial オプションと一緒に使用することは一般的にお勧めしません。サービスが停止および開始されるたびに構成キャッシュが消去されて再作成されるためです。管理サーバーの起動に影響するほかの ndb_mgmd オプションを使用する場合にも注意すべきであり、それを行うことを十分に理解し、起こり得る結果に対して十分に準備していることをしっかりと確認してください。

--install オプションは、Windows 以外のプラットフォームでは効果がありません。

- --remove[=name]

コマンド行形式	--remove[=name]
プラットフォーム固有	Windows
型	文字列
デフォルト	ndb_mgmd

Windows サービスとして以前にインストールされた ndb_mgmd プロセスが削除されます。必要に応じて、アンインストールするサービスの名前を指定できます。設定しない場合、サービス名は ndb_mgmd にデフォルト設定されます。

--remove オプションは、Windows 以外のプラットフォームでは効果がありません。

18.4.5 ndb_mgm — MySQL Cluster 管理クライアント

ndb_mgm 管理クライアントプロセスは、クラスタを実行するために実際には必要ありません。その価値は、クラスタのステータスをチェックしたり、バックアップを開始したり、その他の管理機能を実行したりするための一連のコマンドを提供することにあります。管理クライアントは C API を使用して管理サーバーにアクセスします。上級ユーザーは、この API を使用して専用の管理プロセスをプログラムし、ndb_mgm によって実行されるタスクと同様のことを実行することもできます。

管理クライアントを開始するには、管理サーバーのホスト名およびポート番号を指定する必要があります。

```
shell> ndb_mgm [host_name [port_num]]
```

例:

```
shell> ndb_mgm ndb_mgmd.mysql.com 1186
```

デフォルトのホスト名およびポート番号は、それぞれ localhost および 1186 です。

次の表には、MySQL Cluster 管理クライアントプログラム ndb_mgm に固有のオプションが含まれています。追加説明が表のあとにあります。ほとんどの MySQL Cluster プログラム (ndb_mgm を含む) に共通するオプションについては、セクション18.4.27「MySQL Cluster プログラムに共通するオプション — MySQL Cluster プログラムに共通するオプション」を参照してください。

表 18.80 この表は、ndb_mgm プログラムのコマンド行オプションについて説明しています

形式	説明	追加または削除
--try-reconnect=#、 -t	ndb_mgmd への接続を試みる回数を指定します (0 = 無限)	すべての MySQL 5.6 ベースリリース

形式	説明	追加または削除
--execute=name、 -e	コマンドを実行して終了します	すべての MySQL 5.6 ベースリリース

- --execute=command、-e command

コマンド行形式	--execute=name

このオプションは、システムシェルから MySQL Cluster 管理クライアントにコマンドを送信するために使用できます。たとえば、次のコマンドはいずれも管理クライアントで SHOW を実行することと同等です。

```
shell> ndb_mgm -e "SHOW"

shell> ndb_mgm --execute="SHOW"
```

これは --execute または -e オプションが mysql コマンド行クライアントで動作する仕組みに似ています。セクション4.2.4「コマンド行でのオプションの使用」を参照してください。

> **注記**
>
> このオプションを使用して渡される管理クライアントコマンドにスペース文字が含まれている場合は、コマンドを引用符で囲む必要があります。一重引用符または二重引用符を使用できます。管理クライアントコマンドにスペース文字が含まれていない場合は、引用符を使用しなくてもかまいません。

- --try-reconnect=number

コマンド行形式	--try-reconnect=#
型	ブール
デフォルト	TRUE

管理サーバーへの接続が切断された場合、ノードは 5 秒ごとに再接続を試みます (成功するまで)。このオプションを使用することで、接続を中止して代わりにエラーを報告する前に接続を試行する回数を number 回に制限できます。

ndb_mgm の使用方法に関する追加情報については、セクション18.5.2「MySQL Cluster 管理クライアントのコマンド」を参照してください。

18.4.6 ndb_blob_tool — MySQL Cluster テーブルの BLOB および TEXT カラムのチェックおよび修復

このツールは、孤立した BLOB カラムパーツをチェックして NDB テーブルから削除するため、および孤立したパーツを一覧したファイルを生成するために使用できます。これは、BLOB または TEXT カラムが含まれている破損または損傷した NDB テーブルを診断および修復する場合に役に立つことがあります。

ndb_blob_tool の基本的な構文を次に示します。

```
ndb_blob_tool [options] table [column, ...]
```

--help オプションを使用する場合を除き、1 つ以上のオプション (--check-orphans、--delete-orphans、または --dump-file) を含めることによって、実行するアクションを指定する必要があります。これらのオプションを指定すると、ndb_blob_tool がそれぞれ孤立した BLOB パーツをチェックしたり、孤立した BLOB パーツを削除したり、孤立した BLOB パーツを一覧するダンプファイルを生成したりします。これについての詳細は、このセクションで後述します。

ndb_blob_tool を呼び出すときは、テーブルの名前も指定する必要があります。また、必要に応じて、テーブル名の後ろに、(カンマで区切った) そのテーブルの 1 つ以上の BLOB または TEXT カラムの名前を続けることができます。カラムをリストしない場合、このツールはテーブルのすべての BLOB および TEXT カラムを処理します。データベースを指定する必要がある場合は、--database (-d) オプションを使用します。

--verbose オプションは、ツールの進行状況に関する追加の情報を出力します。

次の表には、ndb_blob_tool に固有のオプションが含まれています。追加説明が表のあとにあります。ほとんどの MySQL Cluster プログラム (ndb_blob_tool を含む) に共通するオプションについては、セクション

18.4.27「MySQL Cluster プログラムに共通するオプション — MySQL Cluster プログラムに共通するオプション」を参照してください。

表 18.81 この表は、ndb_blob_tool プログラムのコマンド行オプションについて説明しています

形式	説明	追加または削除
--check-orphans	孤立した BLOB パーツをチェックします	すべての MySQL 5.6 ベースリリース
--database=db_name、-d	テーブルを探すデータベース。	すべての MySQL 5.6 ベースリリース
--delete-orphans	孤立した BLOB パーツを削除します	すべての MySQL 5.6 ベースリリース
--dump-file=file	指定したファイルに孤立したキーを書き込みます	すべての MySQL 5.6 ベースリリース
--verbose、-v	冗長出力	すべての MySQL 5.6 ベースリリース

- --check-orphans

コマンド行形式	--check-orphans
型	ブール
デフォルト	FALSE

MySQL Cluster テーブルの孤立した BLOB パーツをチェックします。

- --database=db_name、-d

コマンド行形式	--database=db_name
型	文字列
デフォルト	[none]

テーブルを探すデータベースを指定します。

- --delete-orphans

コマンド行形式	--delete-orphans
型	ブール
デフォルト	FALSE

MySQL Cluster テーブルから孤立した BLOB パーツを削除します。

- --dump-file=file

コマンド行形式	--dump-file=file
型	ファイル名
デフォルト	[none]

孤立した BLOB カラムパーツのリストを file に書き込みます。ファイルに書き込まれる情報には、各孤立した BLOB パーツのテーブルキーおよび BLOB パーツ番号が含まれます。

- --verbose

コマンド行形式	--verbose
型	ブール
デフォルト	FALSE

進行状況に関する追加情報をツールの出力に書き込みます。

例

最初に、次に示す CREATE TABLE ステートメントを使用して、test データベースに NDB テーブルを作成します。

```
USE test;

CREATE TABLE btest (
    c0 BIGINT UNSIGNED NOT NULL AUTO_INCREMENT PRIMARY KEY,
    c1 TEXT,
    c2 BLOB
)  ENGINE=NDB;
```

その後、これに似た一連のステートメントを使用して、このテーブルにいくつかの行を挿入します。

```
INSERT INTO btest VALUES (NULL, 'x', REPEAT('x', 1000));
```

このテーブルに対して --check-orphans を指定して ndb_blob_tool を実行すると、次の出力が生成されます。

```
shell> ndb_blob_tool --check-orphans --verbose -d test btest
connected
processing 2 blobs
processing blob #0 c1 NDB$BLOB_19_1
NDB$BLOB_19_1: nextResult: res=1
total parts: 0
orphan parts: 0
processing blob #1 c2 NDB$BLOB_19_2
NDB$BLOB_19_2: nextResult: res=0
NDB$BLOB_19_2: nextResult: res=0
NDB$BLOB_19_2: nextResult: res=0
NDB$BLOB_19_2: nextResult: res=0
NDB$BLOB_19_2: nextResult: res=0
NDB$BLOB_19_2: nextResult: res=0
NDB$BLOB_19_2: nextResult: res=0
NDB$BLOB_19_2: nextResult: res=0
NDB$BLOB_19_2: nextResult: res=0
NDB$BLOB_19_2: nextResult: res=0
NDB$BLOB_19_2: nextResult: res=1
total parts: 10
orphan parts: 0
disconnected

NDBT_ProgramExit: 0 - OK
```

c1 は TEXT カラムですが、カラム c1 に関連付けられている NDB BLOB カラムパーツはないことをこのツールは報告しています。これは、NDB テーブルでは、BLOB または TEXT カラム値の最初の 256 バイトのみがインラインで格納され、それを超過する部分 (ある場合) のみが別個に格納されるためです。したがって、これらのタイプのいずれかのカラムに 256 バイトを超える値がない場合、このカラムの BLOB カラムパーツは NDB によって作成されません。詳細は、セクション11.7「データ型のストレージ要件」を参照してください。

18.4.7 ndb_config — MySQL Cluster 構成情報の抽出

このツールは、いくつかのソース (MySQL Cluster 管理ノード、あるいは config.ini ファイルまたは my.cnf ファイル) のいずれかから、データノード、SQL ノード、および API ノードの現在の構成情報を抽出します。デフォルトでは、管理ノードが構成データのソースです。デフォルトをオーバーライドするには、--config-file オプションまたは --mycnf オプションを指定して ndb_config を実行します。--config_from_node=node_id にノード ID を指定することによって、データノードをソースとして使用することもできます。

ndb_config は、使用できるすべての構成パラメータ、およびそれらのデフォルト値、最大値、最小値、その他の情報のオフラインダンプを生成することもできます。このダンプはテキスト形式または XML 形式で生成できます。詳細は、このセクションで後述する --configinfo および --xml オプションの説明を参照してください)。

--nodes、--system、または --connections のいずれかのオプションを使用すると、結果をセクション (DB、SYSTEM、または CONNECTIONS) によってフィルタリングできます。

次の表には、ndb_config に固有のオプションが含まれています。追加説明が表のあとにあります。ほとんどの MySQL Cluster プログラム (ndb_config を含む) に共通するオプションについては、セクション18.4.27「MySQL Cluster プログラムに共通するオプション — MySQL Cluster プログラムに共通するオプション」を参照してください。

表 18.82 この表は、ndb_config プログラムのコマンド行オプションについて説明しています

形式	説明	追加または削除
--nodes	ノード情報 (DB セクション) のみを出力します。	すべての MySQL 5.6 ベースリリース
--connections	CONNECTIONS セクション情報のみを出力します。--nodes オプションまたは --system オプションとともに使用することはできません。	すべての MySQL 5.6 ベースリリース
--query=string、 -q	1 つ以上のクエリーオプション (属性)	すべての MySQL 5.6 ベースリリース
--host=name	ホストを指定します	すべての MySQL 5.6 ベースリリース
--type=name	ノードタイプを指定します	すべての MySQL 5.6 ベースリリース
--nodeid、 --id	この ID のノードの構成を取得します	すべての MySQL 5.6 ベースリリース
--fields=string、 -f	フィールド区切り文字	すべての MySQL 5.6 ベースリリース
--rows=string、 -r	行区切り文字	すべての MySQL 5.6 ベースリリース
--config-file=path	config.ini ファイルへのパスを設定します	すべての MySQL 5.6 ベースリリース
--mycnf	my.cnf ファイルから構成データを読み取ります	すべての MySQL 5.6 ベースリリース
-c	--ndb-connectstring の短縮形	すべての MySQL 5.6 ベースリリース
--configinfo	すべての NDB 構成パラメータに関する情報をデフォルト値、最大値、および最小値とともにテキスト形式でダンプします。XML 出力を取得するには、--xml とともに使用します。	すべての MySQL 5.6 ベースリリース
--configinfo --xml	すべての NDB 構成パラメータのダンプをデフォルト値、最大値、および最小値とともに XML 形式で取得するには、--configinfo に -xml を使用します。	すべての MySQL 5.6 ベースリリース
--system	SYSTEM セクション情報のみを出力します。--nodes オプションまたは --connections オプションとともに使用することはできません。	すべての MySQL 5.6 ベースリリース
--config_from_node=#	この ID のノード (データノードである必要があります) から構成データを取得します。	すべての MySQL 5.6 ベースリリース

- --usage、--help、または -?

コマンド行形式	--help --usage

ndb_config が使用可能なオプションのリストを出力してから終了します。

- --config_from_node=#

コマンド行形式	--config-from-node=#
型	数値
デフォルト	none
最小値	1
最大値	48

この ID を持つデータノードからクラスタの構成データを取得します。

この ID を持つノードがデータノードではない場合、ndb_config は エラーで失敗します(代わりに、管理ノードから構成データを取得するには、単純にこのオプションを省略します)。

- --version、-V

コマンド行形式	--version

ndb_config がバージョン情報文字列を出力してから終了します。

- --ndb-connectstring=connect_string、-c connect_string

コマンド行形式	--ndb-connectstring=connectstring --connect-string=connectstring
型	文字列
デフォルト	localhost:1186

管理サーバーに接続するために使用する接続文字列を指定します。この接続文字列の形式は、セクション 18.3.2.3「MySQL Cluster の接続文字列」で説明したものと同じであり、デフォルトは localhost:1186 です。

- --config-file=path-to-file

コマンド行形式	--config-file=path
型	ファイル名
デフォルト	

管理サーバーの構成ファイル (config.ini) へのパスを指定します。これには相対パスまたは絶対パスを指定できます。ndb_config が呼び出されるホストと別のホストに管理ノードがある場合は、絶対パスを使用する必要があります。

- --mycnf

コマンド行形式	--mycnf
型	ブール
デフォルト	FALSE

my.cnf ファイルから構成データを読み取ります。

- --query=query-options、-q query-options

コマンド行形式	--query=string
型	文字列

デフォルト	

これは、クエリーオプションのカンマ区切りのリストです。つまり、1 つ以上の返されるノード属性のリストです。これらには、id (ノード ID)、タイプ (ノードタイプ。つまり、ndbd、mysqld、または ndb_mgmd)、および値が取得される構成パラメータが含まれます。

たとえば、 --query=id,type,indexmemory,datamemory は、各ノードのノード ID、ノードタイプ、DataMemory、および IndexMemory を返します。

> **注記**
>
> 指定したパラメータが特定のタイプのノードに当てはまらない場合、対応する値には空の文字列が返されます。詳細はこのセクションで後述する例を参照してください。

- --host=hostname

コマンド行形式	--host=name
型	文字列
デフォルト	

構成情報を取得するノードのホスト名を指定します。

> **注記**
>
> ホスト名 localhost は通常、IP アドレス 127.0.0.1 に解決されますが、これはすべてのオペレーティングプラットフォームおよび構成でそうなるとはかぎりません。これは、config.ini に localhost が使用されて、localhost が別のアドレスに解決される (たとえば、一部のバージョンの SUSE Linux では、これは 127.0.0.2 です) 別のホストで ndb_config が実行されている場合、ndb_config --host=localhost が失敗することがあることを意味します。一般的に最適な結果を得るには、ホストに関連するすべての MySQL Cluster 構成値に数値 IP アドレスを使用するか、すべての MySQL Cluster ホストが localhost を同様に扱っていることを確認してください。

- --id=node_id

  --nodeid=node_id

コマンド行形式	--ndb-nodeid=#
型	数値
デフォルト	0

構成情報を取得するノードのノード ID を指定する場合は、これらのオプションのいずれかを使用できます。推奨される形式は --nodeid です。

- --nodes

コマンド行形式	--nodes
型	ブール
デフォルト	FALSE

DB セクションに定義されているパラメータの情報のみを出力するように、ndb_config に通知します。このオプションは、--connections または --system と一緒に使用できません。

- --connections

コマンド行形式	--connections
型	ブール
デフォルト	FALSE

CONNECTIONS の情報のみを出力するように、ndb_config に通知します。このオプションは、--nodes または --system と一緒に使用できません。

- --system

コマンド行形式	--system
型	ブール
デフォルト	FALSE

SYSTEM の情報のみを出力するように ndb_config に通知します。

このオプションは、--nodes オプションまたは --system オプションと一緒に使用できません。

- --type=node_type

コマンド行形式	--type=name
型	列挙
デフォルト	[none]
有効な値	ndbd mysqld ndb_mgmd

指定した node_type のノード (ndbd、mysqld、または ndb_mgmd) に該当する構成値のみが返されるように、結果をフィルタリングします。

- --fields=delimiter、-f delimiter

コマンド行形式	--fields=string
型	文字列
デフォルト	

結果のフィールドを区切るために使用される delimiter 文字列を指定します。デフォルトは 「,」 (カンマ) です。

> **注記**
>
> delimiter にスペースまたはエスケープ (改行文字の \n など) が含まれている場合は、引用符で囲む必要があります。

- --rows=separator、-r separator

コマンド行形式	--rows=string
型	文字列
デフォルト	

結果の行を区切るために使用される separator 文字列を指定します。デフォルトはスペース文字です。

> **注記**
>
> separator にスペースまたはエスケープ (改行文字の \n など) が含まれている場合は、引用符で囲む必要があります。

- --configinfo

  --configinfo オプションを指定すると、ndb_config が属する MySQL Cluster 配布でサポートされる次の情報を含む各 MySQL Cluster 構成パラメータのリストを ndb_config がダンプします。

  - 各パラメータの目的、効果、および使用方法の簡単な説明
  - パラメータを使用できる config.ini ファイルのセクション
  - パラメータのデータ型または単位
  - 該当する場合、パラメータのデフォルト値、最小値、および最大値
  - パラメータの目的、効果、および使用方法についての簡単な説明
  - MySQL Cluster のリリースバージョンおよびビルド情報

  デフォルトでは、この出力はテキスト形式です。この出力の一部を次に示します。

```
shell> ndb_config --configinfo

****** SYSTEM ******

Name (String)
Name of system (NDB Cluster)
MANDATORY

PrimaryMGMNode (Non-negative Integer)
Node id of Primary ndb_mgmd(MGM) node
Default: 0 (Min: 0, Max: 4294967039)

ConfigGenerationNumber (Non-negative Integer)
Configuration generation number
Default: 0 (Min: 0, Max: 4294967039)

****** DB ******

MaxNoOfSubscriptions (Non-negative Integer)
Max no of subscriptions (default 0 == MaxNoOfTables)
Default: 0 (Min: 0, Max: 4294967039)

MaxNoOfSubscribers (Non-negative Integer)
Max no of subscribers (default 0 == 2 * MaxNoOfTables)
Default: 0 (Min: 0, Max: 4294967039)

...
```

  --configinfo --xml

コマンド行形式	--configinfo --xml
型	ブール
デフォルト	false

  --xml オプションを追加することによって、ndb_config --configinfo の出力を XML として取得できます。出力結果の一部を次の例に示します。

```
shell> ndb_config --configinfo --xml

<configvariables protocolversion="1" ndbversionstring="5.6.22-ndb-7.3.9"
                 ndbversion="458758" ndbversionmajor="7" ndbversionminor="0"
                 ndbversionbuild="6">
 <section name="SYSTEM">
  <param name="Name" comment="Name of system (NDB Cluster)" type="string"
         mandatory="true"/>
  <param name="PrimaryMGMNode" comment="Node id of Primary ndb_mgmd(MGM) node"
         type="unsigned" default="0" min="0" max="4294967039"/>
  <param name="ConfigGenerationNumber" comment="Configuration generation number"
         type="unsigned" default="0" min="0" max="4294967039"/>
 </section>
 <section name="NDBD">
  <param name="MaxNoOfSubscriptions"
         comment="Max no of subscriptions (default 0 == MaxNoOfTables)"
         type="unsigned" default="0" min="0" max="4294967039"/>
  <param name="MaxNoOfSubscribers"
```

```
          comment="Max no of subscribers (default 0 == 2 * MaxNoOfTables)"
          type="unsigned" default="0" min="0" max="4294967039"/>

  ...

 </section>

 ...

</configvariables>
```

> **注記**
>
> 通常、ndb_config --configinfo --xml によって生成される XML 出力では 1 行に 1 要素の形式が使用されますが、上記の例および次の例では読みやすくするために空白を追加しています。ほとんどの XML プロセッサでは通常の処理として不要な空白が無視されるか、無視するように指示できるため、これによってこの出力を使用するアプリケーションに違いは生じないはずです。

この XML 出力は、特定のパラメータが変更されたときに、--initial オプションを使用してデータノードを再起動する必要があることも示しています。これは、対応する <param> 要素に initial="true" 属性が表示されることによって示されます。また、再起動タイプ (system または node) も示されます。これは、特定のパラメータでシステムの再起動が要求される場合に、対応する <param> 要素に restart="system" 属性が表示されることによって示されます。たとえば、Diskless パラメータの値セットを変更すると、次に示されているようにシステムの初期再起動が必要となります (restart 属性および initial 属性が見やすいように強調表示されています)。

```
<param name="Diskless" comment="Run wo/ disk" type="bool" default="false"
       restart="system" initial="true"/>
```

現在、初期再起動が必要ないパラメータに対応する <param> 要素の XML 出力には、initial 属性は表示されません。言い換えると、initial="false" がデフォルトであり、属性が表示されていない場合は値 false を想定してください。同様に、デフォルトの再起動タイプは node (つまり、クラスタのオンラインまたは「ローリング」再起動) ですが、restart 属性は再起動タイプが system (すべてのクラスタノードを同時にシャットダウンしてから再起動する必要があることを意味します) である場合にのみ表示されます。

> **重要**
>
> --xml オプションは --configinfo オプションを指定した場合にのみ使用できます。--configinfo を指定せずに --xml を使用すると、エラーで失敗します。

現在の構成データを取得するためにこのプログラムで使用されるオプションと異なり、ndb_config がコンパイルされた場合、--configinfo および --xml は MySQL Cluster ソースから取得される情報を使用します。このため、これらの 2 つのオプションの場合、実行されている MySQL Cluster に接続したり、config.ini ファイルまたは my.cnf ファイルにアクセスしたりする必要はありません。

ほかの ndb_config オプション (--query、--type など) を --configinfo または --xml と組み合わせることはサポートされていません。現在、そうしようとすると通常、--configinfo または --xml 以外のすべてのオプションが単に無視されます。ただし、この動作は保証されず、今後変更されることがあります。また、--configinfo オプションを指定して使用した場合、ndb_config は MySQL Cluster にアクセスせず、どのファイルも読み取らないため、--ndb-connectstring、--config-file などの追加オプションを --configinfo と一緒に指定しようとしても意味はありません。

例

1. クラスタのノード ID および各ノードのタイプを取得するには、次のようにします。

```
shell> ./ndb_config --query=id,type --fields=':' --rows='\n'
1:ndbd
2:ndbd
3:ndbd
4:ndbd
5:ndb_mgmd
6:mysqld
7:mysqld
8:mysqld
9:mysqld
```

この例では、--fields オプションを使用して ID および各ノードのタイプをコロン (:) で区切り、--rows オプションを使用して各ノードの値を新しい行に出力しています。

2. データノード、SQL ノード、および API ノードが管理サーバーに接続するために使用できる接続文字列を生成するには、次のようにします。

```
shell> ./ndb_config --config-file=usr/local/mysql/cluster-data/config.ini \
--query=hostname,portnumber --fields=: --rows=, --type=ndb_mgmd
192.168.0.179:1186
```

3. この ndb_config 呼び出しでは、データノードのみがチェックされ (--type オプションを使用しています)、各ノードの ID とホスト名の値、および DataMemory、IndexMemory、DataDir パラメータの値セットが表示されます。

```
shell> ./ndb_config --type=ndbd --query=id,host,datamemory,indexmemory,datadir -f ' : ' -r '\n'
1 : 192.168.0.193 : 83886080 : 18874368 : /usr/local/mysql/cluster-data
2 : 192.168.0.112 : 83886080 : 18874368 : /usr/local/mysql/cluster-data
3 : 192.168.0.176 : 83886080 : 18874368 : /usr/local/mysql/cluster-data
4 : 192.168.0.119 : 83886080 : 18874368 : /usr/local/mysql/cluster-data
```

この例では、短いオプション -f および -r をフィールド区切り文字および行区切り文字の設定にそれぞれ使用しました。

4. 特定の 1 つのホスト以外のホストを結果から除外するには、--host オプションを使用します。

```
shell> ./ndb_config --host=192.168.0.176 -f : -r '\n' -q id,type
3:ndbd
5:ndb_mgmd
```

また、この例では、短い形式 -q を使用して、問い合わせる属性を指定しています。

同様に、特定の ID のノードに結果を制限するには、--id オプションまたは --nodeid オプションを使用します。

18.4.8 ndb_cpcd — NDB 開発のためのテストの自動化

この名前のユーティリティーは以前、MySQL Cluster のテストおよびデバッグに使用された内部自動テストフレームワークの一部でした。オラクルによって提供される MySQL Cluster 配布には含まれなくなりました。

18.4.9 ndb_delete_all — NDB テーブルからのすべての行の削除

ndb_delete_all は、指定された NDB テーブルからすべての行を削除します。これは、DELETE または TRUNCATE TABLE よりも速いことがあります。

使用法

```
ndb_delete_all -c connect_string tbl_name -d db_name
```

これは、db_name という名前のデータベースの tbl_name という名前のテーブルからすべての行を削除しています。MySQL で TRUNCATE db_name.tbl_name を実行することとまったく同じです。

次の表には、ndb_delete_all に固有のオプションが含まれています。追加説明が表のあとにあります。ほとんどの MySQL Cluster プログラム (ndb_delete_all を含む) に共通するオプションについては、セクション 18.4.27「MySQL Cluster プログラムに共通するオプション — MySQL Cluster プログラムに共通するオプション」を参照してください。

表 18.83 この表は、ndb_delete_all プログラムのコマンド行オプションについて説明しています

形式	説明	追加または削除
--database=dbname、 -d	テーブルを探すデータベースの名前	すべての MySQL 5.6 ベースリリース
--transactional、 -t	単一トランザクションで削除を実行します (操作の数が足りなくなることがあります)	すべての MySQL 5.6 ベースリリース
--tupscan	TUP スキャンを実行します	すべての MySQL 5.6 ベースリリース
--diskscan	ディスクスキャンを実行します	すべての MySQL 5.6 ベースリリース

- --transactional、-t

このオプションを使用すると、削除操作が単一のトランザクションとして実行されます。

> **警告**
>
> 非常に大きいテーブルの場合は、このオプションを使用すると、クラスタで使用できる操作の数を超えることがあります。

18.4.10 ndb_desc — NDB テーブルの表示

ndb_desc は、1 つ以上の NDB テーブルの詳細な説明を出力します。

使用法

```
ndb_desc -c connect_string tbl_name -d db_name [-p]

ndb_desc -c connect_string index_name -d db_name -t tbl_name
```

出力例

MySQL テーブル作成およびレコード挿入ステートメント:

```
USE test;

CREATE TABLE fish (
    id INT(11) NOT NULL AUTO_INCREMENT,
    name VARCHAR(20) NOT NULL,
    length_mm INT(11) NOT NULL,
    weight_gm INT(11) NOT NULL,

    PRIMARY KEY pk (id),
    UNIQUE KEY uk (name)
) ENGINE=NDB;

INSERT INTO fish VALUES
    ('','guppy', 35, 2), ('','tuna', 2500, 150000),
    ('','shark', 3000, 110000), ('','manta ray', 1500, 50000),
    ('','grouper', 900, 125000), ('','puffer', 250, 2500);
```

ndb_desc の出力:

```
shell> ./ndb_desc -c localhost fish -d test -p
-- fish --
Version: 2
Fragment type: 9
K Value: 6
Min load factor: 78
Max load factor: 80
Temporary table: no
Number of attributes: 4
Number of primary keys: 1
Length of frm data: 311
Row Checksum: 1
Row GCI: 1
SingleUserMode: 0
ForceVarPart: 1
FragmentCount: 2
TableStatus: Retrieved
-- Attributes --
id Int PRIMARY KEY DISTRIBUTION KEY AT=FIXED ST=MEMORY AUTO_INCR
name Varchar(20;latin1_swedish_ci) NOT NULL AT=SHORT_VAR ST=MEMORY
length_mm Int NOT NULL AT=FIXED ST=MEMORY
weight_gm Int NOT NULL AT=FIXED ST=MEMORY

-- Indexes --
PRIMARY KEY(id) - UniqueHashIndex
PRIMARY(id) - OrderedIndex
uk$unique(name) - UniqueHashIndex
uk(name) - OrderedIndex

-- Per partition info --
Partition  Row count  Commit count  Frag fixed memory ...
0          2          2             32768             ...
1          4          4             32768             ...

... Frag varsized memory  Extent_space  Free extent_space
... 32768                 0             0
```

```
... 32768               0          0

NDBT_ProgramExit: 0 - OK
```

複数のテーブルに関する情報は、テーブル名をスペースで区切って ndb_desc を 1 回呼び出すことによって取得できます。すべてのテーブルは同じデータベースにある必要があります。

特定のインデックスに関する追加情報を取得するには、次に示すように ndb_desc の最初の引数としてインデックス名を指定して --table (短縮形: -t) オプションを使用します。

```
shell> ./ndb_desc uk -d test -t fish
-- uk --
Version: 3
Base table: fish
Number of attributes: 1
Logging: 0
Index type: OrderedIndex
Index status: Retrieved
-- Attributes --
name Varchar(20;latin1_swedish_ci) NOT NULL AT=SHORT_VAR ST=MEMORY
-- IndexTable 10/uk --
Version: 3
Fragment type: FragUndefined
K Value: 6
Min load factor: 78
Max load factor: 80
Temporary table: yes
Number of attributes: 2
Number of primary keys: 1
Length of frm data: 0
Row Checksum: 1
Row GCI: 1
SingleUserMode: 2
ForceVarPart: 0
FragmentCount: 4
ExtraRowGciBits: 0
ExtraRowAuthorBits: 0
TableStatus: Retrieved
-- Attributes --
name Varchar(20;latin1_swedish_ci) NOT NULL AT=SHORT_VAR ST=MEMORY
NDB$TNODE Unsigned [64] PRIMARY KEY DISTRIBUTION KEY AT=FIXED ST=MEMORY
-- Indexes --
PRIMARY KEY(NDB$TNODE) - UniqueHashIndex

NDBT_ProgramExit: 0 - OK
```

このようにインデックスを指定すると、--extra-partition-info および --extra-node-info オプションは効果がありません。

出力の Version カラムには、テーブルのスキーマオブジェクトバージョンが表示されます。この値の解釈については、NDB Schema Object Versionsを参照してください。

Extent_space および Free extent_space カラムは、ディスクにカラムがある NDB テーブルにのみ関係しています。インメモリーカラムのみを持つテーブルの場合、これらのカラムには常に値 0 が表示されます。

これらの使用方法を説明するために、前の例を変更します。最初に、必要なディスクデータオブジェクトを次のように作成する必要があります。

```
CREATE LOGFILE GROUP lg_1
    ADD UNDOFILE 'undo_1.log'
    INITIAL_SIZE 16M
    UNDO_BUFFER_SIZE 2M
    ENGINE NDB;

ALTER LOGFILE GROUP lg_1
    ADD UNDOFILE 'undo_2.log'
    INITIAL_SIZE 12M
    ENGINE NDB;

CREATE TABLESPACE ts_1
    ADD DATAFILE 'data_1.dat'
    USE LOGFILE GROUP lg_1
    INITIAL_SIZE 32M
    ENGINE NDB;

ALTER TABLESPACE ts_1
    ADD DATAFILE 'data_2.dat'
    INITIAL_SIZE 48M
```

```
  ENGINE NDB;
```

(上記のステートメントおよびそれらによって作成されるオブジェクトについては、セクション 18.5.12.1「MySQL Cluster ディスクデータオブジェクト」、セクション13.1.14「CREATE LOGFILE GROUP 構文」、およびセクション13.1.18「CREATE TABLESPACE 構文」を参照してください。)

カラムの 2 をディスクに格納するバージョンの fish テーブルを作成して移入できるようになりました (前のバージョンのテーブルが存在する場合はそれを最初に削除します)。

```
CREATE TABLE fish (
  id INT(11) NOT NULL AUTO_INCREMENT,
  name VARCHAR(20) NOT NULL,
  length_mm INT(11) NOT NULL,
  weight_gm INT(11) NOT NULL,

  PRIMARY KEY pk (id),
  UNIQUE KEY uk (name)
) TABLESPACE ts_1 STORAGE DISK
ENGINE=NDB;

INSERT INTO fish VALUES
  ('','guppy', 35, 2), ('','tuna', 2500, 150000),
  ('','shark', 3000, 110000), ('','manta ray', 1500, 50000),
  ('','grouper', 900, 125000), ('','puffer', 250, 2500);
```

ndb_desc をこのバージョンのテーブルに対して実行すると、次の出力が表示されます。

```
shell> ./ndb_desc -c localhost fish -d test -p
-- fish --
Version: 3
Fragment type: 9
K Value: 6
Min load factor: 78
Max load factor: 80
Temporary table: no
Number of attributes: 4
Number of primary keys: 1
Length of frm data: 321
Row Checksum: 1
Row GCI: 1
SingleUserMode: 0
ForceVarPart: 1
FragmentCount: 2
TableStatus: Retrieved
-- Attributes --
id Int PRIMARY KEY DISTRIBUTION KEY AT=FIXED ST=MEMORY AUTO_INCR
name Varchar(20;latin1_swedish_ci) NOT NULL AT=SHORT_VAR ST=MEMORY
length_mm Int NOT NULL AT=FIXED ST=DISK
weight_gm Int NOT NULL AT=FIXED ST=DISK

-- Indexes --
PRIMARY KEY(id) - UniqueHashIndex
PRIMARY(id) - OrderedIndex
uk$unique(name) - UniqueHashIndex
uk(name) - OrderedIndex
-- Per partition info --
Partition  Row count  Commit count  Frag fixed memory ...
0          2          2             32768             ...
1          4          4             32768             ...

... Frag varsized memory  Extent_space  Free extent_space
... 32768                 0             0
... 32768                 0             0

NDBT_ProgramExit: 0 - OK
```

これは、テーブルスペースからこのテーブルの各パーティションに 1048576 バイトが割り当てられ、そのうちの 1044440 バイトが追加の格納のための空き領域であることを意味します。言い換えると、このテーブルのディスクベースのカラムのデータを格納するために、現在パーティションごとに 1048576 - 1044440 = 4136 バイトが使用されています。Free extent_space として表示されるバイト数は、fish テーブルのディスク上カラムデータを格納するためにのみ使用できます。このため、INFORMATION_SCHEMA.FILES テーブルから選択したときに表示されません。

次の表には、ndb_desc に固有のオプションが含まれています。追加説明が表のあとにあります。ほとんどの MySQL Cluster プログラム (ndb_desc を含む) に共通するオプションについては、セクション18.4.27「MySQL Cluster プログラムに共通するオプション — MySQL Cluster プログラムに共通するオプション」を参照してください。

表 18.84 この表は、ndb_desc プログラムのコマンド行オプションについて説明しています

形式	説明	追加または削除
--blob-info、 -b	BLOB テーブルのパーティション情報を出力に含めます。-p オプションも使用する必要があります	すべての MySQL 5.6 ベースリリース
--database=dbname、 -d	テーブルが含まれているデータベースの名前	すべての MySQL 5.6 ベースリリース
--extra-node-info、 -n	パーティションとデータノードのマッピングを出力に含めます。-p オプションも使用する必要があります	すべての MySQL 5.6 ベースリリース
--extra-partition-info、 -p	パーティションに関する情報を表示します	すべての MySQL 5.6 ベースリリース
--retries=#、 -r	接続を再試行する回数 (1 秒おき)	すべての MySQL 5.6 ベースリリース
--table=tbl_name、 -t	インデックスを探すテーブルを指定します。このオプションを使用すると、-p および -n は効果がなくなり、無視されます。	すべての MySQL 5.6 ベースリリース
--unqualified、 -u	修飾されていないテーブル名を使用します	すべての MySQL 5.6 ベースリリース

- --blob-info、-b

  従属する BLOB および TEXT カラムに関する情報を含めます。

  このオプションを使用する場合は、--extra-partition-info (-p) オプションも使用する必要があります。

- --database=db_name, -d

  テーブルが見つかるはずのデータベースを指定します。

- --extra-node-info, -n

  テーブルパーティションおよびそれらが存在するデータノードのマッピングに関する情報を含めます。この情報は、配布認識メカニズムを検証するため、および MySQL Cluster に格納されているデータへのより効率的なアプリケーションアクセスをサポートするために役に立ちます。

  このオプションを使用する場合は、--extra-partition-info (-p) オプションも使用する必要があります。

- --extra-partition-info, -p

  テーブルのパーティションに関する追加情報を出力します。

- --retries=#, -r

  接続を停止する前に、この回数だけ接続を試みます。接続の試行は 1 秒おきに実行されます。

- --table=tbl_name, -t

  インデックスを探すテーブルを指定します。

- --unqualified, -u

  修飾されていないテーブル名を使用します。

18.4.11 ndb_drop_index — NDB テーブルからのインデックスの削除

ndb_drop_index は、指定されたインデックスを NDB テーブルから削除します。このユーティリティーは、NDB API アプリケーションを記述するための例としてのみ使用することをお勧めします。詳細は、このセクションで後述する「警告」を参照してください。

使用法

```
ndb_drop_index -c connect_string table_name index -d db_name
```

上記のステートメントは、database の table から index という名前のインデックスを削除します。

次の表には、ndb_drop_index に固有のオプションが含まれています。追加説明が表のあとにあります。ほとんどの MySQL Cluster プログラム (ndb_drop_index を含む) に共通するオプションについては、セクション 18.4.27「MySQL Cluster プログラムに共通するオプション — MySQL Cluster プログラムに共通するオプション」を参照してください。

表 18.85 この表は、ndb_drop_index プログラムのコマンド行オプションについて説明しています

形式	説明	追加または削除
--database=dbname、 -d	テーブルを探すデータベースの名前	すべての MySQL 5.6 ベースリリース

> **警告**
>
> NDB API を使用して Cluster テーブルインデックスに操作を実行しても、MySQL は認識できず、MySQL サーバーはそのテーブルを使用できません。このプログラムを使用してインデックスを削除し、SQL ノードからテーブルにアクセスしようとすると、次のようなエラーが発生します。

```
shell> ./ndb_drop_index -c localhost dogs ix -d ctest1
Dropping index dogs/idx...OK

NDBT_ProgramExit: 0 - OK

shell> ./mysql -u jon -p ctest1
Enter password: *******
Reading table information for completion of table and column names
You can turn off this feature to get a quicker startup with -A

Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 7 to server version: 5.6.22-ndb-7.3.9

Type 'help;' or '\h' for help. Type '\c' to clear the buffer.

mysql> SHOW TABLES;
+-----------------+
| Tables_in_ctest1 |
+-----------------+
| a               |
| bt1             |
| bt2             |
| dogs            |
| employees       |
| fish            |
+-----------------+
6 rows in set (0.00 sec)

mysql> SELECT * FROM dogs;
ERROR 1296 (HY000): Got error 4243 'Index not found' from NDBCLUSTER
```

そのような場合、MySQL でテーブルをふたたび使用できるようにする唯一の方法は、テーブルを削除して再作成することです。テーブルを削除するには、SQL ステートメント DROP TABLE または ndb_drop_table ユーティリティー (セクション18.4.12「ndb_drop_table — NDB テーブルの削除」を参照してください) を使用できます。

18.4.12 ndb_drop_table — NDB テーブルの削除

ndb_drop_table は指定された NDB テーブルを削除します(NDB 以外のストレージエンジンで作成されたテーブルに対してこれを使用しようとすると、次のエラーで失敗します: 723: No such table exists。)この操作は非常に速く、NDB テーブルに対して MySQL の DROP TABLE ステートメントを使用するよりも格段に速いことがあります。

使用法

```
ndb_drop_table -c connect_string tbl_name -d db_name
```

次の表には、ndb_drop_table に固有のオプションが含まれています。追加説明が表のあとにあります。ほとんどの MySQL Cluster プログラム (ndb_drop_table を含む) に共通するオプションについては、セクション 18.4.27「MySQL Cluster プログラムに共通するオプション — MySQL Cluster プログラムに共通するオプション」を参照してください。

表 18.86 この表は、ndb_drop_table プログラムのコマンド行オプションについて説明しています

形式	説明	追加または削除
--database=dbname、 -d	テーブルを探すデータベースの名前	すべての MySQL 5.6 ベースリリース

18.4.13 ndb_error_reporter — NDB エラーレポートユーティリティー

ndb_error_reporter は、クラスタのバグまたはその他の問題の診断に使用できるデータノードおよび管理ノードログファイルからアーカイブを作成します。MySQL Cluster でバグのレポートを提出するときは、このユーティリティーを使用することを強くお勧めします。

次の表には、MySQL Cluster プログラム ndb_error_reporter に固有のコマンドオプションが含まれています。追加説明が表のあとにあります。ほとんどの MySQL Cluster プログラム (ndb_error_reporter を含む) に共通するオプションについては、セクション18.4.27「MySQL Cluster プログラムに共通するオプション — MySQL Cluster プログラムに共通するオプション」を参照してください。

MySQL Cluster NDB 7.3.3 より前では、ndb_error_reporter は --help オプションをサポートしていませんでした (Bug #11756666、Bug #48606)。--connection-timeout --dry-scp オプションおよび --skip-nodegroup オプションもこのリリースで追加されました (Bug #16602002)。

表 18.87 この表は、ndb_error_reporter プログラムのコマンド行オプションについて説明しています

形式	説明	追加または削除
--connection-timeout=timeout	ノードに接続するときに待機する秒数 (これを経過するとタイムアウト)。	追加: NDB 7.3.3
--dry-scp	リモートホストで scp を無効にします。テストのためのみに使用します。	追加: NDB 7.3.3
--fs	エラーレポートにファイルシステムデータを含めます。大量のディスク領域が使用されることがあります	すべての MySQL 5.6 ベースリリース
--skip-nodegroup=nodegroup_id	この ID のノードグループ内のすべてのノードをスキップします。	追加: NDB 7.3.3

使用法

```
ndb_error_reporter path/to/config-file [username] [options]
```

このユーティリティーは管理ノードホストで使用することが意図されており、管理ホスト構成ファイル (通常、config.ini という名前です) へのパスが必要となります。必要に応じて、データノードログファイルをコピーするために、SSH を使用してクラスタのデータノードにアクセスできるユーザーの名前を指定できます。ndb_error_reporter は、実行されたディレクトリと同じディレクトリに作成されるアーカイブにそれらのすべてのファイルを含めます。アーカイブには ndb_error_report_YYYYMMDDHHMMSS.tar.bz2 という名前が付けられます。ここで YYYYMMDDHHMMSS は日時文字列です。

ndb_error_reporter は次に一覧されているオプションも受け入れます。

- --connection-timeout=timeout

コマンド行形式	--connection-timeout=timeout
導入	5.6.14-ndb-7.3.3
型	整数
デフォルト	0

ノードに接続を試みるときに、タイムアウトする前にこの秒数だけ待機します。

- --dry-scp

コマンド行形式	--dry-scp
導入	5.6.14-ndb-7.3.3
型	ブール
デフォルト	TRUE

リモートホストからの scp を使用せずに ndb_error_reporter を実行します。テストにのみ使用されます。

- --fs

コマンド行形式	--fs
型	ブール
デフォルト	FALSE

データノードファイルシステムを管理ホストにコピーし、それらをアーカイブに含めます。

データノードファイルシステムは圧縮されていても非常に大きいことがあるため、特にそうするように要請された場合を除き、このオプションを使用して作成したアーカイブをオラクルに送信しないでください。

- --skip-nodegroup=nodegroup_id

コマンド行形式	--connection-timeout=timeout
導入	5.6.14-ndb-7.3.3
型	整数
デフォルト	0

指定したノードグループ ID を持つノードグループに属するすべてのノードをスキップします。

18.4.14 ndb_index_stat — NDB インデックス統計ユーティリティー

ndb_index_stat は、NDB テーブルのインデックスに関するフラグメントごとの統計情報を表示します。これには、キャッシュバージョンと経過期間、パーティションごとのインデックスエントリの数、およびインデックスによるメモリー使用量が含まれます。

使用法

指定した NDB テーブルの基本的なインデックス統計を取得するには、最初の引数としてテーブル名を指定し、--database (-d) オプションを使用してこのテーブルが含まれているデータベース名をその直後に指定して、ndb_index_stat を次のように呼び出します。

```
ndb_index_stat table -d database
```

この例では、ndb_index_stat を使用して、test データベースの mytable という名前の NDB テーブルに関するそのような情報を取得しています。

```
shell> ndb_index_stat -d test mytable
table:City index:PRIMARY fragCount:2
sampleVersion:3 loadTime:1399585986 sampleCount:1994 keyBytes:7976
query cache: valid:1 sampleCount:1994 totalBytes:27916
times in ms: save: 7.133 sort: 1.974 sort per sample: 0.000

NDBT_ProgramExit: 0 - OK
```

sampleVersion は、統計データが取得されたキャッシュのバージョン番号です。--update オプションを指定して ndb_index_stat を実行すると、sampleVersion が増分されます。

loadTime はキャッシュが最後に更新された時間を示しています。これは UNIX エポックからの秒数として表されます。

sampleCount はパーティションごとに見つかったインデックスエントリの数です。エントリの合計数を見積もるには、これをフラグメントの数 (fragCount として表示されます) で乗算します。

sampleCount は SHOW INDEX または INFORMATION_SCHEMA.STATISTICS のカーディナリティーと似ています。ただし、後者の 2 つはテーブル全体の統計を示し、ndb_index_stat はフラグメントごとの平均を示します。

keyBytes はインデックスによって使用されるバイト数です。この例では、主キーは整数であり、各インデックスに 4 バイトが必要となるため、keyBytes はこの場合次のように計算できます。

```
keyBytes = sampleCount * (4 bytes per index) = 1994 * 4 = 7976
```

この情報は、INFORMATION_SCHEMA.COLUMNS の対応するカラム定義を使用して取得することもできます (これには、MySQL Server および MySQL クライアントアプリケーションが必要となります)。

totalBytes はテーブルのすべてのインデックスで使用される合計メモリーです (バイト単位)。

前述の例に示されている時間は、ndb_index_stat の各呼び出しに固有のものです。

--verbose オプションを指定すると、次のように追加出力が表示されます。

```
shell> ndb_index_stat -d test mytable --verbose
random seed 1337010518
connected
loop 1 of 1
table:mytable index:PRIMARY fragCount:4
sampleVersion:2 loadTime:1336751773 sampleCount:0 keyBytes:0
read stats
query cache created
query cache: valid:1 sampleCount:0 totalBytes:0
times in ms: save: 20.766 sort: 0.001
disconnected

NDBT_ProgramExit: 0 - OK

shell>
```

オプション

次の表には、MySQL Cluster ndb_index_stat ユーティリティーに固有のオプションが含まれています。詳しい説明は表のあとに一覧されています。ほとんどの MySQL Cluster プログラム (ndb_index_stat を含む) に共通するオプションについては、セクション18.4.27「MySQL Cluster プログラムに共通するオプション — MySQL Cluster プログラムに共通するオプション」を参照してください。

表 18.88 この表は、ndb_index_stat プログラムのコマンド行オプションについて説明しています

形式	説明	追加または削除
--database=name、 -d	テーブルが含まれているデータベースの名前。	すべての MySQL 5.6 ベースリリース
--delete	指定されたテーブルのインデックス統計を削除し、以前に構成された自動更新を停止します。	すべての MySQL 5.6 ベースリリース
--update	指定されたテーブルのインデックス統計を更新し、以前に構成された自動更新を再開します。	すべての MySQL 5.6 ベースリリース
--dump	クエリーキャッシュを出力します。	すべての MySQL 5.6 ベースリリース
--query=#	最初のキー属性 (符号なしの int である必要があります) に対して多数のランダム範囲クエリーを実行します。	すべての MySQL 5.6 ベースリリース
--sys-drop	NDB カーネルの統計テーブルおよびイベントを削除します (すべての統計が失われます)	すべての MySQL 5.6 ベースリリース
--sys-create	NDB カーネルにすべての統計テーブルおよびイベントを作成します (それらがまったく存在していなかった場合)	すべての MySQL 5.6 ベースリリース
--sys-create-if-not-exist	NDB カーネルにあらかじめ存在しない統計テーブルおよびイベントを作成します。	すべての MySQL 5.6 ベースリリース
--sys-create-if-not-valid	無効なものを削除したあとに、NDB カーネルにあらかじめ存在しない統	すべての MySQL 5.6 ベースリリース

形式	説明	追加または削除
	計テーブルまたはイベントを作成します。	
--sys-check	NDB システムのインデックス統計およびイベントテーブルが存在することを確認します。	すべての MySQL 5.6 ベースリリース
--sys-skip-tables	sys-* オプションをテーブルに適用しません。	すべての MySQL 5.6 ベースリリース
--sys-skip-events	sys-* オプションをイベントに適用しません。	すべての MySQL 5.6 ベースリリース
--verbose、 -v	冗長出力を有効にします	すべての MySQL 5.6 ベースリリース
--loops=#	指定されたコマンドを実行する回数を設定します。デフォルトは 0 です。	すべての MySQL 5.6 ベースリリース

ndb_index_stat の統計オプション　次のオプションはインデックス統計を生成するために使用します。これらは指定されたテーブルおよびデータベースを処理します。これらはシステムオプション (ndb_index_stat のシステムオプションを参照してください) と混在させることはできません。

- --database=name、-d name

コマンド行形式	--database=name
型	文字列
デフォルト	[none]
最小値	
最大値	

問い合わせるテーブルが含まれているデータベースの名前。

- --delete

コマンド行形式	--delete
型	ブール
デフォルト	false
最小値	
最大値	

指定したテーブルのインデックス統計を削除し、以前構成された自動更新を停止します。

- --update

コマンド行形式	--update
型	ブール
デフォルト	false
最小値	
最大値	

指定したテーブルのインデックス統計を更新し、以前構成された自動更新を再開します。

- --dump

コマンド行形式	--dump
型	ブール
デフォルト	false
最小値	

最大値	

クエリーキャッシュの内容をダンプします。

- --query=#

コマンド行形式	--query=#
型	数値
デフォルト	0
最小値	0
最大値	MAX_INT

最初のキー属性 (符号なしの int である必要があります) に対してランダム範囲クエリーを実行します。

ndb_index_stat のシステムオプション　次のオプションは、NDB カーネルの統計テーブルを生成および更新するために使用します。これらのオプションは、統計オプションと混在させることはできません (ndb_index_stat の統計オプションを参照してください)。

- --sys-drop

コマンド行形式	--sys-drop
型	ブール
デフォルト	false
最小値	
最大値	

NDB カーネルのすべての統計テーブルおよびイベントを削除します。これを実行すると、すべての統計が失われます。

- --sys-create

コマンド行形式	--sys-create
型	ブール
デフォルト	false
最小値	
最大値	

NDB カーネルにすべての統計テーブルおよびイベントを作成します。これはそれらがあらかじめ存在していなかった場合にのみ動作します。

- sys-create-if-not-exist

コマンド行形式	--sys-create-if-not-exist
型	ブール
デフォルト	false
最小値	
最大値	

このプログラムが呼び出されたときにあらかじめ存在していなかった NDB システム統計テーブルまたはイベント (あるいはその両方) を作成します。

- --sys-create-if-not-valid

コマンド行形式	--sys-create-if-not-valid
型	ブール
デフォルト	false
最小値	

最大値	

無効なものを削除したあとにあらかじめ存在していなかった NDB システムの統計テーブルまたはイベントを作成します。

- --sys-check

コマンド行形式	--sys-check
型	ブール
デフォルト	false
最小値	
最大値	

必要なすべてのシステム統計テーブルおよびイベントが NDB カーネルに存在することを検証します。

- --sys-skip-tables

コマンド行形式	--sys-skip-tables
型	ブール
デフォルト	false
最小値	
最大値	

--sys-* オプションを統計テーブルに適用しません。

- --sys-skip-events

コマンド行形式	--sys-skip-events
型	ブール
デフォルト	false
最小値	
最大値	

--sys-* オプションをイベントに適用しません。

- --verbose

コマンド行形式	--verbose
型	ブール
デフォルト	false
最小値	
最大値	

冗長出力を有効にします。

- --loops=#

コマンド行形式	--loops=#
型	数値
デフォルト	0
最小値	0
最大値	MAX_INT

コマンドをこの回数繰り返します (テストで使用するため)。

18.4.15 ndb_print_backup_file — NDB バックアップファイルの内容の出力

ndb_print_backup_file は、クラスタバックアップファイルから診断情報を取得します。

使用法

```
ndb_print_backup_file file_name
```

file_name はクラスタバックアップファイル名です。これには、クラスタバックアップディレクトリにある任意のファイル (.Data、.ctl、または .log ファイル) を指定できます。これらのファイルは、データノードのバックアップディレクトリのサブディレクトリ BACKUP-# にあります。ここで、# はバックアップのシーケンス番号です。クラスタバックアップファイルおよびその内容については、セクション18.5.3.1「MySQL Cluster バックアップの概念」を参照してください。

ndb_print_schema_file および ndb_print_sys_file と同様に (および管理サーバーホストで実行するか、管理サーバーに接続することが意図されているほかのほとんどの NDB ユーティリティーと異なり)、ndb_print_backup_file は、データノードファイルシステムに直接アクセスするため、クラスタデータノードで実行する必要があります。このユーティリティーは管理サーバーを使用しないため、管理サーバーが実行されていない場合、およびクラスタが完全にシャットダウンされている場合にも使用できます。

その他のオプション

なし。

18.4.16 ndb_print_file — NDB ディスクデータファイル内容の出力

ndb_print_file は、MySQL Cluster ディスクデータファイルから情報を取得します。

使用法

```
ndb_print_file [-v] [-q] file_name+
```

file_name は、MySQL Cluster ディスクデータファイル名です。複数のファイル名はスペースで区切ることによって受け入れられます。

ndb_print_schema_file および ndb_print_sys_file と同様に (および管理サーバーホストで実行すること、または管理サーバーに接続することが意図されているほかのほとんどの NDB ユーティリティーと異なり)、ndb_print_file は、データノードファイルシステムに直接アクセスするため、MySQL Cluster データノードで実行する必要があります。このユーティリティーは管理サーバーを使用しないため、管理サーバーが実行されていない場合、およびクラスタが完全にシャットダウンされている場合にも使用できます。

その他のオプション

ndb_print_file では、次のオプションがサポートされます。

- -v: 出力を冗長にします。
- -q: 出力を抑止します (抑止モード)。
- --help、-h、-?: 出力を抑止します (抑止モード)。

  このオプションは、MySQL Cluster NDB 7.3.7 より前は正常に動作しませんでした。(Bug #17069285)

詳細は、セクション18.5.12「MySQL Cluster ディスクデータテーブル」を参照してください。

18.4.17 ndb_print_schema_file — NDB スキーマファイル内容の出力

ndb_print_schema_file は、クラスタスキーマファイルから診断情報を取得します。

使用法

```
ndb_print_schema_file file_name
```

file_name はクラスタスキーマファイル名です。クラスタスキーマファイルについては、NDB Cluster Data Node File System Directoryを参照してください。

ndb_print_backup_file および ndb_print_sys_file と同様に (および管理サーバーのホストで実行すること、または管理サーバーに接続することが意図されているほかのほとんどの NDB ユーティリティーと異なり)、ndb_schema_backup_file は、データノードファイルシステムに直接アクセスするため、クラスタデータノードで実行する必要があります。このユーティリティーは管理サーバーを使用しないため、管理サーバーが実行されていない場合、およびクラスタが完全にシャットダウンされている場合にも使用できます。

その他のオプション

なし。

18.4.18 ndb_print_sys_file — NDB システムファイル内容の出力

ndb_print_sys_file は、MySQL Cluster システムファイルから診断情報を取得します。

使用法

```
ndb_print_sys_file file_name
```

file_name はクラスタシステムファイル (sysfile) 名です。クラスタシステムファイルはデータノードのデータディレクトリ (DataDir) にあり、このディレクトリからシステムファイルへのパスは ndb_#_fs/D#/DBDIH/P#.sysfile というパターンに一致します。それぞれの箇所で、# は数字を表しています (同じ数字であるとはかぎりません)。詳細は、NDB Cluster Data Node File System Directoryを参照してください。

ndb_print_backup_file および ndb_print_schema_file と同様に (および管理サーバーのホストで実行すること、または管理サーバーに接続することが意図されているほかのほとんどの NDB ユーティリティーと異なり)、ndb_print_backup_file は、データノードファイルシステムに直接アクセスするため、クラスタデータノードで実行する必要があります。このユーティリティーは管理サーバーを使用しないため、管理サーバーが実行されていない場合、およびクラスタが完全にシャットダウンされている場合にも使用できます。

その他のオプション

なし。

18.4.19 ndbd_redo_log_reader — クラスタ Redo ログ内容のチェックおよび出力

Redo ログファイルを読み取って、そのエラーをチェックする、または人間が読むことのできる形式で内容を出力する、あるいはその両方を行います。ndbd_redo_log_reader は、MySQL Cluster 開発者およびサポート担当者が問題をデバッグおよび診断するために使用することが主に意図されています。

このユーティリティーはまだ開発中であり、構文および動作は今後の MySQL Cluster リリースで変更されることがあります。

ndbd_redo_log_reader の C++ ソースファイルは、/storage/ndb/src/kernel/blocks/dblqh/redoLogReader ディレクトリにあります。

次の表には、MySQL Cluster プログラム ndbd_redo_log_reader に固有のオプションが含まれています。追加説明が表のあとにあります。ほとんどの MySQL Cluster プログラム (ndbd_redo_log_reader を含む) に共通するオプションについては、セクション18.4.27「MySQL Cluster プログラムに共通するオプション — MySQL Cluster プログラムに共通するオプション」を参照してください。

表 18.89 この表は、ndbd_redo_log_reader プログラムのコマンド行オプションについて説明しています

形式	説明	追加または削除
-noprint	レコードを出力しません	すべての MySQL 5.6 ベースリリース
-nocheck	レコードのエラーをチェックしません	すべての MySQL 5.6 ベースリリース
--help	使用方法の情報を出力します	追加: NDB 7.3.4

使用法

```
ndbd_redo_log_reader file_name [options]
```

file_name はクラスタ Redo ログファイルの名前です。Redo ログファイルはデータノードのデータディレクトリ (DataDir) の下の番号付きディレクトリにあり、このディレクトリから Redo ログファイルへのパスは ndb_#_fs/D#/LCP/#/T#F#.Data というパターンに一致します。それぞれの箇所で、# は数字を表しています (同じ数字であるとはかぎりません)。詳細は、NDB Cluster Data Node File System Directoryを参照してください。

読み取られるファイルの名前の後ろには、次に一覧する 1 つ以上のオプションが続くことがあります。

コマンド行形式	-noprint
型	ブール

デフォルト	FALSE

-noprint: ログファイルの内容を出力しません。

コマンド行形式	-nocheck
型	ブール
デフォルト	FALSE

-nocheck: ログファイルのエラーをチェックしません。

コマンド行形式	--help
導入	5.6.15-ndb-7.3.4

--help: 使用方法の情報を出力します。

MySQL Cluster NDB 7.3.4 で追加されました。(Bug #11749591、Bug #36805)

ndb_print_backup_file および ndb_print_schema_file と同様に (および管理サーバーのホストで実行すること、または管理サーバーに接続することが意図されているほとんどの NDB ユーティリティーと異なり)、ndbd_redo_log_reader は、データノードファイルシステムに直接アクセスするため、クラスタデータノードで実行する必要があります。このユーティリティーは管理サーバーを使用しないため、管理サーバーが実行されていない場合、およびクラスタが完全にシャットダウンされている場合にも使用できます。

18.4.20 ndb_restore — MySQL Cluster バックアップのリストア

クラスタリストアプログラムは、別個のコマンド行ユーティリティー ndb_restore として実装されており、通常は MySQL の bin ディレクトリにあります。このプログラムは、バックアップの結果として作成されたファイルを読み取り、格納されている情報をデータベースに挿入します。

ndb_restore は、バックアップを作成するために使用される START BACKUP コマンド (セクション 18.5.3.2「MySQL Cluster 管理クライアントを使用したバックアップの作成」を参照してください) によって作成されたバックアップファイルごとに、一度実行する必要があります。これは、バックアップが作成された時点の、クラスタ内のデータノード数と同じです。

> **注記**
>
> 複数のデータノードを並列でリストアしている場合を除き、ndb_restore を使用する前に、クラスタをシングルユーザーモードで実行することをお勧めします。詳細は、セクション18.5.8「MySQL Cluster のシングルユーザーモード」を参照してください。

次の表には、MySQL Cluster ネイティブバックアップリストアプログラム ndb_restore に固有のオプションが含まれています。追加説明が表のあとにあります。ほとんどの MySQL Cluster プログラム (ndb_restore を含む) に共通するオプションについては、セクション18.4.27「MySQL Cluster プログラムに共通するオプション — MySQL Cluster プログラムに共通するオプション」を参照してください。

表 18.90 この表は、ndb_restore プログラムのコマンド行オプションについて説明しています

形式	説明	追加または削除
--connect、 -c	--connectstring のエイリアス。	すべての MySQL 5.6 ベースリリース
--nodeid=#、 -n	この ID のノードからファイルをバックアップします	すべての MySQL 5.6 ベースリリース
--backupid=#、 -b	指定された ID のバックアップからリストアします	すべての MySQL 5.6 ベースリリース
--restore_data、 -r	NDB API を使用してテーブルデータおよびログを NDB Cluster にリストアします	すべての MySQL 5.6 ベースリリース
--restore_meta、 -m	NDB API を使用してメタデータを NDB Cluster にリストアします	すべての MySQL 5.6 ベースリリース

形式	説明	追加または削除
--no-upgrade、 -u	VAR データがまだサイズ変更されていない可変サイズ属性の配列タイプをアップグレードせず、カラム属性を変更しません	すべての MySQL 5.6 ベースリリース
--promote-attributes、 -A	バックアップからデータをリストアするときに、属性の昇格を許可します	すべての MySQL 5.6 ベースリリース
--preserve-trailing-spaces、 -P	固定幅文字列型を可変幅型に昇格するときに、末尾の空白 (パディングを含む) を維持することを許可します	すべての MySQL 5.6 ベースリリース
--no-restore-disk-objects、 -d	ディスクデータに関連するオブジェクトをリストアしません	すべての MySQL 5.6 ベースリリース
--restore_epoch、 -e	エポック情報をステータステーブルにリストアします。MySQL Cluster レプリケーションスレーブでレプリケーションを開始する場合に便利です。mysql.ndb_apply_status の ID 0 の行が更新/挿入されます。	すべての MySQL 5.6 ベースリリース
--skip-table-check、 -s	データのリストア中に、テーブル構造チェックをスキップします	すべての MySQL 5.6 ベースリリース
--parallelism=#、 -p	データのリストア中に使用する並列トランザクションの数	すべての MySQL 5.6 ベースリリース
--print	メタデータ、データ、およびログを stdout に出力します (--print_meta --print_data --print_log と同等です)	すべての MySQL 5.6 ベースリリース
--print_meta	メタデータを stdout に出力します	すべての MySQL 5.6 ベースリリース
--print_data	データを stdout に出力します	すべての MySQL 5.6 ベースリリース
--print_log	stdout に出力します	すべての MySQL 5.6 ベースリリース
--backup_path=path	バックアップファイルディレクトリへのパス	すべての MySQL 5.6 ベースリリース
--dont_ignore_systab_0、 -f	リストア中にシステムテーブルを無視しません。実験用であり、本番で使用するためのものではありません	すべての MySQL 5.6 ベースリリース
--ndb-nodegroup-map=map、 -z	NDBCLUSTER ストレージエンジンのためのノードグループマップ。構文: (source_nodegroup, destination_nodegroup) のリスト	すべての MySQL 5.6 ベースリリース
--fields-enclosed-by=char	フィールドが指定された文字で囲まれます	すべての MySQL 5.6 ベースリリース
--fields-terminated-by=char	フィールドが指定された文字で終了します	すべての MySQL 5.6 ベースリリース
--fields-optionally-enclosed-by	フィールドは、必要に応じて指定された文字で囲まれます	すべての MySQL 5.6 ベースリリース
--lines-terminated-by=char	指定された文字で行が終了します	すべての MySQL 5.6 ベースリリース
--hex	バイナリタイプを 16 進形式で出力します	すべての MySQL 5.6 ベースリリース

形式	説明	追加または削除
--tab=path、 -T	指定されたパス内の各テーブルの、タブ区切りの .txt ファイルを作成します	すべての MySQL 5.6 ベースリリース
--append	タブ区切りのファイルにデータを追加します	すべての MySQL 5.6 ベースリリース
--progress-frequency=#	指定された秒数おきに、リストアのステータスを出力します	すべての MySQL 5.6 ベースリリース
--no-binlog	mysqld が接続されていて、バイナリロギングを使用している場合、リストアされたデータのログを記録しません	すべての MySQL 5.6 ベースリリース
--verbose=#	出力の冗長性レベル	すべての MySQL 5.6 ベースリリース
--include-databases=db-list	1 つ以上のリストアするデータベースのリスト (指定されていないものを除外します)	すべての MySQL 5.6 ベースリリース
--exclude-databases=db-list	1 つ以上の除外するデータベースのリスト (指定されていないものを含めます)	すべての MySQL 5.6 ベースリリース
--include-tables=table-list	リストアする 1 つ以上のテーブルのリスト (同じデータベース内の指定されていないものを除外します)。各テーブル参照にはデータベース名が含まれている必要があります	すべての MySQL 5.6 ベースリリース
--exclude-tables=table-list	除外する 1 つ以上のテーブルのリスト (同じデータベース内の指定されていないものを含めます)。各テーブル参照にはデータベース名が含まれている必要があります	すべての MySQL 5.6 ベースリリース
--exclude-missing-columns	バックアップバージョンのテーブルから、データベース内のバージョンのテーブルにないカラムが無視されます。	すべての MySQL 5.6 ベースリリース
--exclude-missing-tables	バックアップから、データベースにないテーブルが無視されます。	追加: NDB 7.3.7
--disable-indexes	バックアップ内のインデックスが無視されます。データのリストアに必要となる時間が短くなることがあります。	すべての MySQL 5.6 ベースリリース
--rebuild-indexes	バックアップで見つかった順序付けされたインデックスをマルチスレッドで再構築します。使用されるスレッド数は、BuildIndexThreads パラメータを設定することによって決定されます。	すべての MySQL 5.6 ベースリリース
--skip-broken-objects	バックアップファイル内で欠けている BLOB テーブルが無視されます。	すべての MySQL 5.6 ベースリリース
--skip-unknown-objects	新しいバージョンの MySQL Cluster から作成されたバックアップを古いバージョンにリストアするときに、ndb_restore によって認識されないスキーマオブジェクトが無視されます。	すべての MySQL 5.6 ベースリリース
--rewrite-database=olddb,newdb	元の名前と異なる名前のデータベースにリストアします	すべての MySQL 5.6 ベースリリース
--lossy-conversions、	バックアップからデータをリストアするときに、カラム値の不可逆変換	すべての MySQL 5.6 ベースリリース

形式	説明	追加または削除
-L	(型の昇格または符号の変更) を許可します	
--restore-privilege-tables	以前 NDB に移動された MySQL 権限テーブルをリストアします。	すべての MySQL 5.6 ベースリリース
--exclude-intermediate-sql-tables[=TRUE|FALSE]	TRUE (デフォルト) の場合は、ALTER TABLE のコピー操作で残された中間テーブル (名前の前に「#sql-」が付けられています) がリストアされません。	追加: NDB 7.3.6

このユーティリティーの一般的なオプションを次に示します。

通常、MySQL Cluster のバックアップからリストアする場合、ndb_restore では --nodeid (短縮形: -n)、--backupid (短縮形: -b)、および --backup_path オプションが最低限必要となります。

```
ndb_restore [-c connection_string] -n node_id -b backup_id \
    [-m] -r --backup_path=/path/to/backup/files
```

-c オプションは、ndb_restore にクラスタ管理サーバーを探す場所を伝える接続文字列を指定するために使用します。(接続文字列については、セクション18.3.2.3「MySQL Cluster の接続文字列」を参照してください)。このオプションを使用しない場合、ndb_restore は localhost:1186 の管理サーバーに接続を試みます。このユーティリティーはクラスタ API ノードとして動作するため、クラスタ管理サーバーに接続するための空き接続「スロット」が必要となります。これは、クラスタ config.ini ファイル内にそれが使用できる [api] セクションまたは [mysqld] セクションが少なくとも 1 つ存在する必要があることを意味します。このため、MySQL サーバーまたはほかのアプリケーションに使用されていない空の [api] セクションまたは [mysqld] セクションを、config.ini 内に少なくとも 1 つ用意することをお勧めします (セクション18.3.2.7「MySQL Cluster 内の SQL ノードおよびその他の API ノードの定義」を参照してください)。

ndb_restore がクラスタに接続されていることを確認するには、ndb_mgm 管理クライアントで SHOW コマンドを使用します。システムシェルで次のようにすることでこれを実現することもできます。

```
shell> ndb_mgm -e "SHOW"
```

-n はバックアップが取得されたデータノードのノード ID を指定するために使用します。

ndb_restore リストアプログラムを最初に実行するときは、メタデータもリストアする必要があります。言い換えると、データベーステーブルを再作成する必要があります。これは、--restore_meta (-m) オプションを指定して実行することによって行うことができます。メタデータをリストアする必要があるのは単一データノードだけです。これで十分にクラスタ全体にリストアできます。バックアップのリストアを開始するときには、クラスタに空のデータベースがあるべきです(言い換えると、リストアを実行する前に、--initial を指定して ndbd を開始してください)。

テーブルメタデータをリストアしなくてもデータをリストアできます。これを行うときのデフォルト動作は、テーブルデータがテーブルスキーマと一致しない場合に ndb_restore がエラーで失敗することです。これは、--skip-table-check オプションまたは -s オプションを使用してオーバーライドできます。

ndb_restore を使用してデータをリストアするときのカラム定義内の不一致に関する制限の一部が緩和されています。それらのタイプの不一致のいずれかが発生しても、ndb_restore は以前のように エラーで停止しなくなり、代わりにデータを受け入れてターゲットテーブルに挿入し、これが行われているという警告をユーザーに発行します。この動作は、--skip-table-check オプションまたは --promote-attributes オプションが使用されているかどうかにかかわらず、実行されます。カラム定義でのこれらの違いには次のタイプがあります。

- 異なる COLUMN_FORMAT 設定 (FIXED、DYNAMIC、DEFAULT)
- 異なる STORAGE 設定 (MEMORY、DISK)
- 異なるデフォルト値
- 異なる配布キー設定

ndb_restore は、MySQL レプリケーションでサポートされるのとほぼ同様に、属性昇格を制限付きでサポートします。つまり、特定の型のカラムからバックアップされたデータは一般的に、「より大きい同様の」型を使用してカラムにリストアできます。たとえば、CHAR(20) カラムのデータは VARCHAR(20)、VARCHAR(30)、または CHAR(30) として宣言されているカラムにリストアでき、MEDIUMINT カラムのデータは、INT または BIGINT

型のカラムにリストアできます。属性昇格によって現在サポートされる型変換の表については、データ型が異なるカラムのレプリケーションを参照してください。

ndb_restore による属性昇格は、次のように明示的に有効にする必要があります。

1. バックアップをリストアするテーブルを準備します。ndb_restore を使用してオリジナルと異なる定義でテーブルを再作成することはできません。これは、テーブルを手動で作成するか、またはテーブルメタデータをリストアしたあとかつデータをリストアする前に昇格するカラムを ALTER TABLE を使用して変更する必要があることを意味します。

2. テーブルデータをリストアするときに、--promote-attributes オプション (短縮形: -A) を指定してndb_restore を呼び出します。このオプションを使用しない場合、属性昇格は行われず、リストア操作がエラーで失敗します。

MySQL Cluster NDB 7.3.3 より前は、文字データ型と TEXT または BLOB の間の変換が正しく処理されませんでした (Bug #17325051)。

MySQL Cluster NDB 7.3.7 より前は、TEXT から TINYTEXT への降格が正しく処理されませんでした (Bug #18875137)。

文字データ型と TEXT または BLOB の間で変換する場合は、文字型 (CHAR および VARCHAR) とバイナリ型 (BINARY および VARBINARY) の間の変換のみを同時に実行できます。たとえば、同じ ndb_restore 呼び出しで、VARCHAR カラムを TEXT に昇格するときに、INT カラムを BIGINT に昇格できません。

異なる文字セットが使用されている TEXT カラム間の変換はサポートされていません。MySQL Cluster NDB 7.3.7 以降では、これは明確に許可されません (Bug #18875137)。

ndb_restore を使用して文字型またはバイナリ型から TEXT または BLOB への変換を実行するときに、table_name$STnode_id という名前の 1 つ以上のステージングテーブルが作成および使用されることに気付くことがあります。これらのテーブルはその後必要ではなくなるため、通常はリストアが成功したあとに ndb_restore によって削除されます。

--lossy-conversions、-L

コマンド行形式	--lossy-conversions
型	ブール
デフォルト	FALSE (オプションを使用しない場合)

このオプションは、--promote-attributes オプションを補完することを意図しています。--lossy-conversions を使用するときは、バックアップからデータをリストアするときにカラム値の不可逆変換 (型降格または符号の変更) が許可されます。いくつかの例外はありますが、降格を制御するルールは MySQL レプリケーションの場合と同じです。属性降格によって現在サポートされる型変換については、データ型が異なるカラムのレプリケーションを参照してください。

ndb_restore は、不可逆変換中に実行されるデータの切り捨てを属性およびカラムごとに報告します。

--preserve-trailing-spaces オプション (短縮形: -R) を使用すると、固定幅の文字データ型を可変幅の同等の型に昇格するとき (つまり、CHAR カラム値を VARCHAR に、または BINARY カラム値を VARBINARY に昇格するとき) に、末尾の空白が維持されます。それ以外の場合は、新しいカラムに挿入されるときに、末尾の空白はそのようなカラム値から削除されます。

> **注記**
>
> CHAR カラムを VARCHAR に、および BINARY カラムを VARBINARY に昇格することはできますが、VARCHAR カラムを CHAR に、または VARBINARY カラムを BINARY に昇格することはできません。

-b オプションはバックアップの ID またはシーケンス番号を指定するために使用し、バックアップ完了時に表示される「Backup backup_id completed」メッセージで管理クライアントによって示される番号と同じ番号です。(セクション18.5.3.2「MySQL Cluster 管理クライアントを使用したバックアップの作成」を参照してください)。

> **重要**
>
> クラスタバックアップをリストアするときは、同じバックアップ ID を持つバックアップからすべてのデータノードをリストアする必要があります。別のバックアップのファイルを使用すると、クラスタがリストアされたとしても不整合性な状態で、要するに失敗する可能性があります。

--restore_epoch (短縮形: -e) は、エポック情報をクラスタレプリケーションステータステーブルに追加 (またはリストア) します。これは、MySQL Cluster レプリケーションスレーブでレプリケーションを開始する場合に便利です。このオプションを使用すると、id カラムが 0 である mysql.ndb_apply_status 内の行が更新されます (存在する場合)。存在しない場合はそのような行が挿入されます(セクション18.6.9「MySQL Cluster レプリケーションを使用した MySQL Cluster バックアップ」を参照してください)。

--restore_data

このオプションを指定すると、ndb_restore が NDB テーブルデータおよびログを出力します。

--restore_meta

このオプションを指定すると、ndb_restore が NDB テーブルメタデータを出力します。通常、このオプションを使用する必要があるのは、クラスタの最初のデータノードをリストアする場合のみです。追加データノードは最初のものからメタデータを取得できます。

--restore-privilege-tables

デフォルトでは、ndb_restore は配布された MySQL 権限テーブルをリストアしません。このオプションを指定すると、ndb_restore が権限テーブルをリストアします。

これが動作するのは、バックアップが取得される前に、権限テーブルが NDB に変換された場合のみです。詳細は、セクション18.5.14「MySQL Cluster の配布された MySQL 権限」を参照してください。

--backup_path

バックアップディレクトリへのパスが必要です。これは --backup_path オプションを使用して ndb_restore に指定し、リストアするバックアップのバックアップ ID に対応するサブディレクトリが含まれている必要があります。たとえば、データノードの DataDir が /var/lib/mysql-cluster である場合、バックアップディレクトリは /var/lib/mysql-cluster/BACKUP であり、ID 3 のバックアップのバックアップファイルは /var/lib/mysql-cluster/BACKUP/BACKUP-3 にあります。パスには、絶対パス、または ndb_restore 実行可能ファイルがあるディレクトリからの相対パスを指定でき、必要に応じて backup_path= を前に付けることができます。

作成されたデータベースとは異なる構成のデータベースにバックアップをリストアできます。たとえば、ノード ID 2 および 3 の 2 つのデータベースノードがあるクラスタで作成されたバックアップ ID 12 のバックアップを 4 ノードのクラスタに復旧するとします。その場合は、ndb_restore を 2 回実行する必要があります (バックアップを取得したクラスタのデータベースノードごとに 1 回)。ただし、ndb_restore はあるバージョンの MySQL で実行されているクラスタから作成されたバックアップを、別のバージョンの MySQL で実行されているクラスタに常に復旧できるとはかぎりません。詳細は、セクション18.2.8「MySQL Cluster NDB 7.3 のアップグレードとダウングレード」を参照してください。

> **重要**
>
> 新しいバージョンの MySQL Cluster から作成したバックアップを古いバージョンの ndb_restore を使用してリストアすることはできません。新しいバージョンの MySQL から作成したバックアップを古いクラスタにリストアすることはできますが、それを行うには新しいバージョンの MySQL Cluster の ndb_restore のコピーを使用する必要があります。
>
> たとえば、MySQL Cluster NDB 7.2.5 を実行しているクラスタから取得したクラスタバックアップを MySQL Cluster NDB 7.1.21 を実行しているクラスタにリストアするには、MySQL Cluster NDB 7.2.5 配布に付属する ndb_restore を使用する必要があります。

リストアをより高速にするために、十分な数のクラスタ接続が使用できる場合は、データを並列でリストアできます。より正確に言うと、複数のノードを並列でリストアする場合は、各並列 ndb_restore プロセスに使用できる [api] または [mysqld] セクションがクラスタ config.ini ファイルに存在する必要があります。ただし、データファイルは常にログの前に適用する必要があります。

--no-upgrade

ndb_restore を使用してバックアップをリストアする場合、古い固定長形式を使用して作成された VARCHAR カラムは、現在採用されている可変幅形式を使用してサイズ変更および再作成されます。この動作は、ndb_restore を実行するときに、--no-upgrade オプション (短縮形: -u) を使用してオーバーライドできます。

--print_data

--print_data オプションを指定すると、ndb_restore が出力を stdout にリダイレクトします。

TEXT および BLOB カラム値は常に切り捨てられます。MySQL Cluster NDB 7.3.7 以前では、そのような値は出力で最初の 240 バイトに切り捨てられ、MySQL Cluster NDB 7.3.8 以降では、256 バイトに切り捨てられます (Bug #14571512、Bug #65467)。これは現在 --print_data を使用してもオーバーライドできません。

データダンプを stdout またはファイルに生成する際に、いくつかの追加オプションを --print_data オプションとともに使用できます。これらは mysqldump で使用される一部のオプションに似ており、次のリストに示されています。

- --tab、-T

コマンド行形式	--tab=path

このオプションを使用すると、--print_data はテーブルごとに、それぞれ tbl_name.txt という名前が付けられたダンプファイルを作成します。ファイルを保存すべきディレクトリへのパスを引数が必要です。現在のディレクトリの場合は . を使用します。

- --fields-enclosed-by=string

コマンド行形式	--fields-enclosed-by=char
型	文字列
デフォルト	

各カラム値は、このオプションに渡された文字列によって囲まれます (データ型は関係ありません。次の項目を参照してください)。

- --fields-optionally-enclosed-by=string

コマンド行形式	--fields-optionally-enclosed-by
型	文字列
デフォルト	

このオプションに渡された文字列は、文字データが含まれているカラム値 (CHAR、VARCHAR、BINARY、TEXT、ENUM など) を囲むために使用されます。

- --fields-terminated-by=string

コマンド行形式	--fields-terminated-by=char
型	文字列
デフォルト	\t (tab)

このオプションに渡された文字列は、カラム値を区切るために使用されます。デフォルト値はタブ文字 (\t) です。

- --hex

コマンド行形式	--hex

このオプションを使用すると、すべてのバイナリ値は 16 進形式で出力されます。

- --fields-terminated-by=string

コマンド行形式	--fields-terminated-by=char
型	文字列
デフォルト	\t (tab)

このオプションは、出力の各行を終了するために使用される文字列を指定します。デフォルトは改行文字 (\n) です。

- --append

コマンド行形式	--append

--tab オプションおよび --print_data オプションとともに使用した場合、同じ名前を持つ既存のファイルにデータが付加されます。

> **注記**
>
> テーブルに明示的な主キーがない場合、--print_data オプションを使用して生成される出力には、テーブルの隠し主キーが含められます。

--print_meta

このオプションを指定すると、ndb_restore がすべてのメタデータをstdout に出力します。

--print_log

--print_log オプションを指定すると、ndb_restore がログを stdout に出力します。

--print

ndb_restore がすべてのデータ、メタデータ、およびログを stdout に出力します。--print_data、--print_meta、および --print_log オプションを一緒に使用することと同等です。

> **注記**
>
> --print または --print_* オプションのいずれかを使用した場合は、仮実行をするときに有効になります。これらのオプションを 1 つ以上含めると、出力が stdout にリダイレクトされます。そのような場合、ndb_restore はデータまたはメタデータを MySQL Cluster にリストアしようとしません。

--dont_ignore_systab_0

通常、テーブルデータおよびメタデータをリストアするときに、ndb_restore はバックアップに存在する NDB システムテーブルのコピーを無視します。--dont_ignore_systab_0 を指定すると、システムテーブルがリストアされます。このオプションは、実験および開発での使用のみが意図されており、本番環境には推奨されていません。

--ndb-nodegroup-map、-z

このオプションは、あるノードグループから取得されたバックアップを別のノードグループにリストアするために使用できます。引数は、source_node_group, target_node_group という形式のリストです。

--no-binlog

このオプションは、接続されている SQL ノードが ndb_restore によってリストアされるデータをバイナリログに書き込むことを阻止します。

--no-restore-disk-objects、-d

このオプションを指定すると、ndb_restore が MySQL Cluster ディスクデータオブジェクト (テーブルスペース、ログファイルグループなど) をリストアすることを阻止します。これらの詳細は、セクション18.5.12「MySQL Cluster ディスクデータテーブル」を参照してください。

--parallelism=#、-p

ndb_restore が使用を試みることができる並列トランザクションの最大数を決定します。デフォルトでは、これは 128 であり、最小値は 1、最大値は 1024 です。

--progress-frequency=N

バックアップの進行中に、ステータスレポートを N 秒ごとに出力します。0 (デフォルト) を指定すると、ステータスレポートは出力されません。最大値は 65535 です。

--verbose=#

出力の冗長性のレベルを設定します。最小値は 0 であり、最大値は 255 です。デフォルト値は 1 です。

次に示す構文を使用すると、選択したデータベースのみ、または単一データベースの選択したテーブルのみをリストアできます。

```
ndb_restore other_options db_name,[db_name[,...] | tbl_name[,tbl_name][,...]]
```

言い換えると、次のいずれかをリストアすることを指定できます。

- 1 つ以上のデータベースのすべてのテーブル
- 単一データベースの 1 つ以上のテーブル

--include-databases=db_name[,db_name][,...]

コマンド行形式	--include-databases=db-list
型	文字列
デフォルト	

--include-tables=db_name.tbl_name[,db_name.tbl_name][,...]

コマンド行形式	--include-tables=table-list
型	文字列
デフォルト	

特定のデータベースまたはテーブルのみをリストアするには、--include-databases オプションまたは --include-tables オプションをそれぞれ使用します。--include-databases は、リストアするデータベースのカンマ区切りリストを受け入れます。--include-tables はリストアするテーブルのカンマ区切りリスト (database.table という形式) を受け入れます。

--include-databases または --include-tables を使用すると、それらのオプションによって指定されたデータベースまたはテーブルのみがリストアされます。ほかのすべてのデータベースおよびテーブルは ndb_restore によって除外され、リストアされません。

次の表は、--include-* オプション (必要となる可能性があるほかのオプションは、わかりやすくするために省略しています) を使用した ndb_restore のいくつかの呼び出し、および MySQL Cluster バックアップからリストアする場合のそれらの効果を示しています。

使用するオプション	結果
--include-databases=db1	データベース db1 内のテーブルのみがリストアされます。ほかのすべてのデータベース内のすべてのテーブルは無視されます
--include-databases=db1,db2 (または --include-databases=db1 --include-databases=db2)	データベース db1 および db2 内のテーブルのみがリストアされます。ほかのすべてのデータベース内のすべてのテーブルは無視されます
--include-tables=db1.t1	データベース db1 内のテーブル t1 のみがリストアされます。db1 またはその他のデータベース内のほかのテーブルはリストアされません
--include-tables=db1.t2,db2.t1 (または --include-tables=db1.t2 --include-tables=db2.t1)	データベース db1 内のテーブル t2 およびデータベース db2 内のテーブル t1 のみがリストアされます。db1、db2、またはその他のデータベース内のほかのテーブルはリストアされません

これらの 2 つのオプションは一緒に使用することもできます。たとえば、次の場合は、データベース db1 および db2 内のすべてのテーブルに加えて、データベース db3 内のテーブル t1 および t2 がリストアされます (ほかのデータベースまたはテーブルはリストアされません)。

```
shell> ndb_restore [...] --include-databases=db1,db2 --include-tables=db3.t1,db3.t2
```

(この例でも、必要となる可能性があるほかのオプションを省略しています。)

--exclude-databases=db_name[,db_name][,...]

コマンド行形式	--exclude-databases=db-list
型	文字列
デフォルト	

--exclude-tables=db_name.tbl_name[,db_name.tbl_name][,...]

コマンド行形式	--exclude-tables=table-list
型	文字列

デフォルト	

1 つ以上のデータベースまたはテーブルがリストアされないようにするには、ndb_restore の --exclude-databases および --exclude-tables オプションを使用します。--exclude-databases はリストアすべきでない 1 つ以上のデータベースのカンマ区切りリストを受け入れます。--exclude-tables はリストアすべきでない 1 つ以上のテーブルのカンマ区切りリスト (database.table という形式を使用します) を受け入れます。

--exclude-databases または --exclude-tables を使用した場合は、これらのオプションによって指定されたデータベースまたはテーブルのみが除外されます。ほかのすべてのデータベースおよびテーブルは ndb_restore によってリストアされます。

次の表は、--exclude-* オプション (必要となる可能性があるほかのオプションは、わかりやすくするために省略しています) を使用したいくつかの ndb_restore 呼び出し、および MySQL Cluster バックアップからリストアする場合にこれらのオプションが持つ効果を示しています。

使用するオプション	結果
--exclude-databases=db1	db1 を除くすべてのデータベース内のすべてのテーブルがリストアされます。db1 内のテーブルはリストアされません
--exclude-databases=db1,db2 (または --exclude-databases=db1 --exclude-databases=db2)	db1 および db2 を除くすべてのデータベース内のすべてのテーブルがリストアされます。db1 または db2 内のテーブルはリストアされません
--exclude-tables=db1.t1	データベース db1 内の t1 を除くすべてのテーブルがリストアされます。db1 内のほかのすべてのテーブルはリストアされます。ほかのすべてのデータベース内のすべてのテーブルはリストアされます
--exclude-tables=db1.t2,db2.t1 (または --exclude-tables=db1.t2 --exclude-tables=db2.t1)	データベース db1 内の t2 を除くすべてのテーブル、およびデータベース db2 内の t1 を除くすべてのテーブルがリストアされます。db1 または db2 内のほかのテーブルはリストアされません。ほかのすべてのデータベース内のすべてのテーブルはリストアされます

これらの 2 つのオプションは一緒に使用できます。たとえば、次の場合は、データベース db1 と db2、およびデータベース db3 内のテーブル t1 と t2 を除くすべてのデータベース内のすべてのテーブルがリストアされません。

```
shell> ndb_restore [...] --exclude-databases=db1,db2 --exclude-tables=db3.t1,db3.t2
```

(この例でも、わかりやすく簡潔にするために、必要となる可能性があるほかのオプションを省略しています)。

--include-* オプションおよび --exclude-* オプションは、次のルールに従って一緒に使用できます。

- すべての --include-* および --exclude-* オプションのアクションは累積されます。
- すべての --include-* および --exclude-* オプションは、ndb_restore に渡された順序で右から左に評価されます。
- 競合するオプションがある場合は、最初 (もっとも右側) のオプションが優先されます。言い換えると、対象となるデータベースまたはテーブルと一致する最初のオプション (右から左に適用して) が「適用されます」。

たとえば、次の一連のオプションを指定すると、ndb_restore がデータベース db1 の db1.t1 を除くすべてのテーブルをリストアし、ほかのデータベースのほかのテーブルはリストアしません。

```
--include-databases=db1 --exclude-tables=db1.t1
```

ただし、前述のオプションの順序を逆にすると、データベース db1 のすべてのテーブルがリストアされます (db1.t1 は含まれますが、ほかのデータベースのテーブルは含まれません)。これは、もっとも右にある --include-databases オプションがデータベース db1 に対して最初に一致し、db1 または db1 内のテーブルと一致するその他のオプションより優先されるためです。

```
--exclude-tables=db1.t1 --include-databases=db1
```

--exclude-missing-columns

コマンド行形式	--exclude-missing-columns

--exclude-missing-columns オプションを使用すると、選択したテーブルカラムのみをリストアすることもできます。このオプションを使用するときは、ndb_restore は、バックアップで見つかったテーブルのバージョンと比較して、リストアされるテーブルで欠けているカラムを無視します。このオプションはリストアされるすべてのテーブルに適用されます。選択したテーブルまたはデータベースのみにこのオプションを適用したい場合は、それを実行する際に前の段落で説明したオプションの 1 つ以上を組み合わせて使用できます。つまり、残りのテーブルには、それらのオプションセットを補完的に使用してデータをリストアできます。

--exclude-missing-tables

コマンド行形式	--exclude-missing-tables
導入	5.6.21-ndb-7.3.7

MySQL Cluster NDB 7.3.7 以降では、このオプションを使用して選択したテーブルカラムのみをリストアすることもでき、その場合、ndb_restore はバックアップからターゲットデータベースにないテーブルを無視します。

--disable-indexes

コマンド行形式	--disable-indexes

ネイティブ NDB バックアップからのデータのリストア中に、インデックスのリストアを無効にします。あとで、--rebuild-indexes を使用してマルチスレッドでインデックスを構築することで、すべてのテーブルのインデックスを一度にリストアできます。非常に大きいテーブルの場合には、インデックスを同時に再構築するよりもこの方が速いはずです。

--rebuild-indexes

コマンド行形式	--rebuild-indexes

ndb_restore でこのオプションを使用すると、ネイティブ NDB バックアップのリストア中に、順序付けされたインデックスがマルチスレッドで再構築されます。このオプション付きの ndb_restore で順序付けされたインデックスを構築するために使用されるスレッドの数は、BuildIndexThreads データノード構成パラメータによって制御されます。

このオプションを使用する必要があるのは、ndb_restore の最初の実行の場合のみです。これにより、以降のノードをリストアするときに --rebuild-indexes をふたたび使用しなくても、すべての順序付けられたインデックスが再構築されます。このオプションはデータベースに新しい行を挿入する前に使用してください。そうしないと、インデックスを再構築しようとするときに、挿入される行があとで一意制約違反になることがあります。

一意インデックスの再構築では、Redo ロギングおよびローカルチェックポイント処理のディスク書き込み帯域幅が使用されます。この帯域幅が十分ではない場合、Redo バッファーオーバーロードエラーまたはログオーバーロードエラーになることがあります。そのような場合は、ndb_restore --rebuild-indexes を再度実行できます。この処理はエラーが発生した箇所から再開されます。これは、一時エラーが発生したときにも行うことができます。ndb_restore --rebuild-indexes の実行は無限に繰り返すことができます。DiskCheckpointSpeed の値を減らして Redo ロギングに追加のディスク帯域幅を与えることによって、そのようなエラーを停止できることがあります。

--skip-broken-objects

コマンド行形式	--skip-broken-objects

このオプションを指定すると、ndb_restore がネイティブ NDB バックアップを読み取るときに破損しているテーブルを無視して、残りのテーブル (破損していない) のリストアを続行します。現在のところ、--skip-broken-objects オプションは BLOB パーツテーブルが欠けている場合にのみ機能します。

--skip-unknown-objects

コマンド行形式	--skip-unknown-objects

このオプションを指定すると、ndb_restore がネイティブ NDB バックアップを読み取るときに、認識されないスキーマオブジェクトを無視します。これは、MySQL Cluster NDB 7.3 を実行しているクラスタから作成されたバックアップを MySQL Cluster NDB 7.2 を実行しているクラスタにリストアするために使用できます。

--rewrite-database=old_dbname,new_dbname

コマンド行形式	--rewrite-database=olddb,newdb
型	文字列

デフォルト	none

このオプションを指定すると、バックアップに使用された名前と異なる名前を持つデータベースにリストアできます。たとえば、バックアップが products という名前のデータベースから作成された場合は、このオプションを次のように使用して (必要となる可能性があるほかのオプションを省略しています)、含まれているデータを inventory という名前のデータベースにリストアできます。

```
shell> ndb_restore --rewrite-database=product,inventory
```

このオプションは、ndb_restore の単一呼び出しで複数回指定できます。このため、--rewrite-database=db1,db2 --rewrite-database=db3,db4 を使用して、db1 という名前のデータベースから db2 という名前のデータベースに、および db3 という名前のデータベースから db4 という名前のデータベースに同時にリストアできます。複数の --rewrite-database 発生箇所の間にほかの ndb_restore オプションを指定できます。

複数の --rewrite-database オプションで競合が発生した場合は、左から右に読んで最後に使用されている --rewrite-database オプションが有効となります。たとえば、--rewrite-database=db1,db2 --rewrite-database=db1,db3 が使用された場合、--rewrite-database=db1,db3 のみが有効となり、--rewrite-database=db1,db2 は無視されます。複数のデータベースから単一データベースにリストアすることもできるため、--rewrite-database=db1,db3 --rewrite-database=db2,db3 を指定すると、データベース db1 および db2 のすべてのテーブルおよびデータがデータベース db3 にリストアされます。

> **重要**
>
> --rewrite-database を使用して複数のバックアップデータベースから単一ターゲットデータベースにリストアする場合、テーブル名またはその他のオブジェクト名間の競合はチェックされず、行がリストアされる順序は保証されません。これは、そのような場合に行が上書きされて更新が失われることがあることを意味します。

--exclude-intermediate-sql-tables[=TRUE|FALSE]

コマンド行形式	--exclude-intermediate-sql-tables[=TRUE|FALSE]
導入	5.6.17-ndb-7.3.6
型	ブール
デフォルト	TRUE

ALTER TABLE のコピー操作を実行するときに、mysqld は中間テーブルを作成します (名前の前に #sql- が付けられます)。TRUE を指定すると、--exclude-intermediate-sql-tables オプションによって、ndb_restore がそのような操作で残された可能性があるテーブルをリストアしなくなります。このオプションはデフォルトでは TRUE です。

--exclude-intermediate-sql-tables オプションは MySQL Cluster NDB 7.3.6 で導入されました。(Bug #17882305)

エラー報告
ndb_restore は一時的および永続的エラーの両方を報告します。一時エラーの場合は、それらからリカバリできる場合があり、そのようなときには「Restore successful, but encountered temporary error, please look at configuration」と報告されます。

> **重要**
>
> ndb_restore を使用して循環レプリケーションに使用するために MySQL Cluster を初期化したあとに、レプリケーションスレーブとして動作する SQL ノード上のバイナリログは自動的に作成されないため、それらを作成させるように手動で操作する必要があります。バイナリログを作成させるには、START SLAVE を実行する前に、その SQL ノードで SHOW TABLES ステートメントを発行します。これは MySQL Cluster の既知の問題です。

18.4.21 ndb_select_all — NDB テーブルの行の出力

ndb_select_all は、NDB テーブルのすべての行を stdout に出力します。

使用法

```
ndb_select_all -c connect_string tbl_name -d db_name [> file_name]
```

次の表には、MySQL Cluster ネイティブバックアップリストアプログラム ndb_select_all に固有のオプションが含まれています。追加説明が表のあとにあります。ほとんどの MySQL Cluster プログラム (ndb_select_all を含

む) に共通するオプションについては、セクション18.4.27「MySQL Cluster プログラムに共通するオプション — MySQL Cluster プログラムに共通するオプション」を参照してください。

表 18.91 この表は、ndb_select_all プログラムのコマンド行オプションについて説明しています

形式	説明	追加または削除
--database=dbname、 -d	テーブルを探すデータベースの名前	すべての MySQL 5.6 ベースリリース
--parallelism=#、 -p	並列性の度合い	すべての MySQL 5.6 ベースリリース
--lock=#、 -l	ロックタイプ	すべての MySQL 5.6 ベースリリース
--order=index、 -o	指定された名前のインデックスに従って結果セットをソートします	すべての MySQL 5.6 ベースリリース
--descending、 -z	結果セットを降順にソートします (順序フラグが必要となります)	すべての MySQL 5.6 ベースリリース
--header、 -h	ヘッダーを出力します (出力でヘッダーを無効にするには 0|FALSE を設定します)	すべての MySQL 5.6 ベースリリース
--useHexFormat、 -x	数値を 16 進形式で出力します	すべての MySQL 5.6 ベースリリース
--delimiter=char、 -D	カラム区切り文字を設定します	すべての MySQL 5.6 ベースリリース
--disk	ディスク参照を出力します (インデックスが設定されていないカラムを持つディスクデータテーブルでのみ役に立ちます)	すべての MySQL 5.6 ベースリリース
--rowid	ROWID を出力します	すべての MySQL 5.6 ベースリリース
--gci	出力に GCI を含めます	すべての MySQL 5.6 ベースリリース
--gci64	出力に GCI および行工ポックを含めます	すべての MySQL 5.6 ベースリリース
--tup、 -t	TUP 順序でスキャンします	すべての MySQL 5.6 ベースリリース
--nodata	テーブルカラムデータを出力しません	すべての MySQL 5.6 ベースリリース

- --database=dbname、-d dbname

  テーブルが見つかるデータベースの名前。デフォルト値は TEST_DB です。

- parallelism=#、-p #

  並列の度合いを指定します。

- --lock=lock_type、-l lock_type

  テーブルを読み取るときにロックを適用します。lock_type に指定できる値を次に示します。

  - 0: 読み取りロック
  - 1: ホールド付きの読み取りロック
  - 2: 排他的読み取りロック

このオプションにはデフォルト値はありません。

- --order=index_name、-o index_name

  index_name という名前のインデックスの順序で出力します。これはカラム名ではなくインデックス名であり、インデックスは作成時に明示的に名前が付けられている必要があります。

- --descending、-z

  出力を降順でソートします。このオプションは、-o (--order) オプションを指定した場合にのみ使用できます。

- --header=FALSE

  出力からカラムヘッダーを除外します。

- --useHexFormat -x

  すべての数値を 16 進数形式で表示します。これは文字列または日時値に含まれている数値の出力には影響しません。

- --delimiter=character、-D character

  character をカラム区切り文字として使用します。この区切り文字で区切られるのはテーブルデータカラムのみです。

  デフォルトの区切り文字はタブ文字です。

- --disk

  出力にディスクリファレンスカラムを追加します。このカラムが空でないのは、インデックスが設定されていないカラムを持つディスクデータテーブルの場合だけです。

- --rowid

  行が保存されるフラグメントに関する情報を示す ROWID カラムを追加します。

- --gci

  各行が最後に更新されたグローバルチェックポイントを示す GCI カラムを出力に追加します。チェックポイントに関する詳細は、セクション18.1「MySQL Cluster の概要」およびセクション18.5.6.2「MySQL Cluster ログイベント」を参照してください。

- --gci64

  各行が最後に更新されたグローバルチェックポイントおよびこの更新が発生した工ポックの数を示す ROW $GCI64 カラムを出力に追加します。

- --tupscan、-t

  タプルの順序でテーブルをスキャンします。

- --nodata

  テーブルデータを省きます。

出力例

MySQL SELECT ステートメントからの出力:

```
mysql> SELECT * FROM ctest1.fish;
+----+-----------+
| id | name      |
+----+-----------+
|  3 | shark     |
|  6 | puffer    |
|  2 | tuna      |
|  4 | manta ray |
|  5 | grouper   |
|  1 | guppy     |
+----+-----------+
```

```
6 rows in set (0.04 sec)
```

同等の ndb_select_all 呼び出しからの出力:

```
shell> ./ndb_select_all -c localhost fish -d ctest1
id      name
3       [shark]
6       [puffer]
2       [tuna]
4       [manta ray]
5       [grouper]
1       [guppy]
6 rows returned

NDBT_ProgramExit: 0 - OK
```

ndb_select_all の出力では、すべての文字列値は角括弧 (「[...]」) で囲まれます。ほかの例として、次のように作成されて移入されたテーブルがあるとします。

```
CREATE TABLE dogs (
    id INT(11) NOT NULL AUTO_INCREMENT,
    name VARCHAR(25) NOT NULL,
    breed VARCHAR(50) NOT NULL,
    PRIMARY KEY pk (id),
    KEY ix (name)
)
TABLESPACE ts STORAGE DISK
ENGINE=NDBCLUSTER;

INSERT INTO dogs VALUES
    ('', 'Lassie', 'collie'),
    ('', 'Scooby-Doo', 'Great Dane'),
    ('', 'Rin-Tin-Tin', 'Alsatian'),
    ('', 'Rosscoe', 'Mutt');
```

ほかのいくつかの ndb_select_all オプションの使用方法を次に示します。

```
shell> ./ndb_select_all -d ctest1 dogs -o ix -z --gci --disk
GCI     id name          breed        DISK_REF
834461  2  [Scooby-Doo]  [Great Dane] [ m_file_no: 0 m_page: 98 m_page_idx: 0 ]
834878  4  [Rosscoe]     [Mutt]       [ m_file_no: 0 m_page: 98 m_page_idx: 16 ]
834463  3  [Rin-Tin-Tin] [Alsatian]   [ m_file_no: 0 m_page: 34 m_page_idx: 0 ]
835657  1  [Lassie]      [Collie]     [ m_file_no: 0 m_page: 66 m_page_idx: 0 ]
4 rows returned

NDBT_ProgramExit: 0 - OK
```

18.4.22 ndb_select_count — NDB テーブルの行数の出力

ndb_select_count は、1 つ以上の NDB テーブルの行数を出力します。単一テーブルでは、結果は MySQL ステートメント SELECT COUNT(*) FROM tbl_name を使用して取得される結果と同じになります。

使用法

```
ndb_select_count [-c connect_string] -ddb_name tbl_name[, tbl_name2[, ...]]
```

次の表には、MySQL Cluster ネイティブバックアップリストアプログラム ndb_select_count に固有のオプションが含まれています。追加説明が表のあとにあります。ほとんどの MySQL Cluster プログラム (ndb_select_count を含む) に共通するオプションについては、セクション18.4.27「MySQL Cluster プログラムに共通するオプション — MySQL Cluster プログラムに共通するオプション」を参照してください。

表 18.92 この表は、ndb_select_count プログラムのコマンド行オプションについて説明しています

形式	説明	追加または削除
--database=dbname、 -d	テーブルを探すデータベースの名前	すべての MySQL 5.6 ベースリリース
--parallelism=#、 -p	並列性の度合い	すべての MySQL 5.6 ベースリリース
--lock=#、 -l	ロックタイプ	すべての MySQL 5.6 ベースリリース

同じデータベース内の複数のテーブルの行数を取得するには、「出力例」に示されているように、このコマンドを呼び出すときにテーブル名をスペースで区切ってリストします。

出力例

```
shell> ./ndb_select_count -c localhost -d ctest1 fish dogs
6 records in table fish
4 records in table dogs

NDBT_ProgramExit: 0 - OK
```

18.4.23 ndb_setup.py — MySQL Cluster のブラウザベース自動インストーラの開始

ndb_setup.py は、MySQL Cluster Auto-Installer を開始し、デフォルト Web ブラウザでインストーラの「Start」ページを開きます。

このセクションでは、コマンド行ツールのみの使用方法およびプログラムオプションについて説明します。ndb_setup.py を呼び出すと開始される Auto-Installer GUI の使用方法については、セクション 18.2.1「MySQL Cluster Auto-Installer」を参照してください。

使用法

すべてのプラットフォーム:

```
ndb_setup.py [options]
```

または、Windows プラットフォームのみ:

```
setup.bat [options]
```

次の表には、MySQL Cluster のインストールおよび構成プログラム ndb_setup.py によってサポートされるすべてのオプションが含まれています。追加説明が表のあとにあります。

表 18.93 この表は、ndb_setup.py プログラムのコマンド行オプションについて説明しています

形式	説明	追加または削除
--browser-start-page=filename、-s	Web ブラウザを開始したときに開くページ。	すべての MySQL 5.6 ベースリリース
--ca-certs-file=filename、-a	サーバーに接続できるクライアント証明書のリストが含まれているファイル	すべての MySQL 5.6 ベースリリース
--cert-file=filename、-c	サーバーを識別する X509 証明書が含まれているファイル(デフォルト: cfg.pem)	すべての MySQL 5.6 ベースリリース
--debug-level=level、-d	Python ロギングモジュールのデバッグレベル。DEBUG、INFO、WARNING (デフォルト)、ERROR、または CRITICAL のいずれか。	すべての MySQL 5.6 ベースリリース
--help、-h	ヘルプメッセージを出力します	すべての MySQL 5.6 ベースリリース
--key-file=file、-k	プライベートキーが含まれているファイルを指定します (--cert-file に含まれていない場合)	すべての MySQL 5.6 ベースリリース
--no-browser、-n	ブラウザで開始ページを開かずに、単にツールを開始します	すべての MySQL 5.6 ベースリリース
--port=#、-p	Web サーバーによって使用されるポートを指定します	すべての MySQL 5.6 ベースリリース

形式	説明	追加または削除
--server-log-file=file、 o	要求のログをこのファイルに記録します。代わりに stderr への口ギングを強制するには '-' を使用します。	すべての MySQL 5.6 ベースリリース
--server-name=name、 -N	接続するサーバーの名前	すべての MySQL 5.6 ベースリリース
--use-https、 -S	セキュア (HTTPS) クライアントサーバー接続を使用します	すべての MySQL 5.6 ベースリリース

- --browser-start-page=file、-s

コマンド行形式	--browser-start-page=filename
型	文字列
デフォルト	index.html

ブラウザでインストールおよび構成の「Start」ページとして開くファイルを指定します。デフォルトは index.html です。

- --ca-certs-file=file、-a

コマンド行形式	--ca-certs-file=filename
型	ファイル名
デフォルト	[none]

サーバーに接続できるクライアント証明書のリストが含まれているファイルを指定します。デフォルトは空の文字列であり、クライアント認証が使用されないことを意味します。

- --cert-file=file、-c

コマンド行形式	--cert-file=filename
型	ファイル名
デフォルト	cfg.pem

サーバーを識別する X509 証明書が含まれているファイルを指定します。証明書は自己署名できます。デフォルトは cfg.pem です。

- --debug-level=level、-d

コマンド行形式	--debug-level=level
型	列挙
デフォルト	WARNING
有効な値	DEBUG INFO ERROR CRITICAL

Python 口ギングモジュールデバッグレベルを設定します。これは、DEBUG、INFO、WARNING、ERROR、または CRITICAL のいずれかです。WARNING がデフォルトです。

- --help、-h

コマンド行形式	--help

ヘルプメッセージを出力します。

- --key-file=file、-d

コマンド行形式	--key-file=file
型	ファイル名
デフォルト	[none]

X509 証明書ファイル (--cert-file) にプライベートキーが含まれていない場合、これが含まれているファイルを指定します。デフォルトは空の文字列であり、そのようなファイルが使用されないことを意味します。

- --no-browser、-n

コマンド行形式	--no-browser

インストールおよび構成ツールを開始しますが、ブラウザで「Start」ページを開きません。

- --port=#、-p

コマンド行形式	--port=#
型	数値
デフォルト	8081
最小値	1
最大値	65535

Web サーバーによって使用されるポートを設定します。デフォルトは 8081 です。

- --server-log-file=file、-o

コマンド行形式	--server-log-file=file o
型	ファイル名
デフォルト	ndb_setup.log
有効な値	- (stderr にログを記録します)

要求のログをこのファイルに記録します。デフォルトは ndb_setup.log です。ファイルではなく stderr にロギングするように指定するには、ファイル名に - (ダッシュ文字) を使用します。

- --server-name=host、-N

コマンド行形式	--server-name=name
型	文字列
デフォルト	localhost

ブラウザが接続するときに使用するホスト名または IP アドレスを指定します。デフォルトは localhost です。

- --use-https、-S

コマンド行形式	--use-https

ブラウザがサーバーに対してセキュア (HTTPS) 接続を使用します。

18.4.24 ndb_show_tables — NDB テーブルのリストの表示

ndb_show_tables は、クラスタ内のすべての NDB データベースオブジェクトのリストを表示します。デフォルトでは、これには、ユーザーが作成したテーブルと NDB システムテーブルのみではなく、NDB 固有のインデックス、内部トリガー、および MySQL Cluster ディスクデータオブジェクトも含まれます。

次の表には、MySQL Cluster ネイティブバックアップリストアプログラム ndb_show_tables に固有のオプションが含まれています。追加説明が表のあとにあります。ほとんどの MySQL Cluster プログラム (ndb_show_tables を含む) に共通するオプションについては、セクション18.4.27「MySQL Cluster プログラムに共通するオプション — MySQL Cluster プログラムに共通するオプション」を参照してください。

表 18.94 この表は、ndb_show_tables プログラムのコマンド行オプションについて説明しています

形式	説明	追加または削除
--database=string、 -d	テーブルを探すデータベースを指定します	すべての MySQL 5.6 ベースリリース
--loops=#、 -l	出力を繰り返す回数	すべての MySQL 5.6 ベースリリース
--type=#、 -t	このタイプのオブジェクトに出力を制限します	すべての MySQL 5.6 ベースリリース
--unqualified、 -u	テーブル名を修飾しません	すべての MySQL 5.6 ベースリリース
--parsable、 -p	MySQL LOAD DATA INFILE ステートメントに適した出力を返します	すべての MySQL 5.6 ベースリリース
--show-temp-status	テーブル一時フラグを表示します	すべての MySQL 5.6 ベースリリース

使用法

```
ndb_show_tables [-c connect_string]
```

- --database、-d

  テーブルが見つかるデータベースの名前を指定します。

- --loops、-l

  ユーティリティーを実行すべき回数を指定します。このオプションを指定しない場合はこれは 1 ですが、このオプションを使用する場合はそれに整数の引数を指定する必要があります。

- --parsable、-p

  このオプションを使用すると、出力が LOAD DATA INFILE での使用に適した形式になります。

- --show-temp-status

  指定した場合は、一時テーブルが表示されます。

- --type、-t

  次に示す整数型コードで指定された 1 種類のオブジェクトに出力を制限するために使用できます。

  - 1: システムテーブル
  - 2: ユーザーが作成したテーブル
  - 3: 一意のハッシュインデックス

  その他の値を指定すると、すべての NDB データベースオブジェクトが一覧されます (デフォルト)。

- --unqualified、-u

  これを指定すると、修飾されていないオブジェクト名が表示されます。

> **注記**
>
> MySQL からアクセスできるのは、ユーザーが作成した MySQL Cluster テーブルのみです。SYSTAB_0 などのシステムテーブルは mysqld からは見えません。ただし、ndb_select_all などの NDB API アプリケーションを使用してシステムテーブルの内容を調べることができます (セクション18.4.21「ndb_select_all — NDB テーブルの行の出力」を参照してください)。

18.4.25 ndb_size.pl — NDBCLUSTER サイズ要件エスティメータ

これは、NDBCLUSTER ストレージエンジンを使用するように変更された場合に、MySQL データベースによって要求される空き領域の大きさを見積もるために使用できる Perl スクリプトです。このセクションで説明したほかのユーティリティーとは異なり、MySQL Cluster にアクセスする必要はありません (実際、アクセスする理由がありません)。ただし、テストするデータベースがある MySQL サーバーにアクセスする必要はあります。

要件

- 実行されている MySQL サーバー。サーバーインスタンスが MySQL Cluster のサポートを提供する必要はありません。
- Perl の有効なインストール。
- DBI モジュール (まだ Perl インストールの一部でない場合は、CPAN から取得できます)(多くの Linux およびその他のオペレーティングシステム配布では、このライブラリの独自のパッケージが提供されています)。
- 必要な権限を持つ MySQL ユーザーアカウント。既存のアカウントを使用したくない場合は、GRANT USAGE ON db_name.* を使用して作成するだけでこの用途には十分です。ここで、db_name は検査するデータベースの名前です。

ndb_size.pl は、storage/ndb/tools 内の MySQL ソースでも見つかります。

次の表には、MySQL Cluster プログラム ndb_size.pl に固有のオプションが含まれています。追加説明が表のあとにあります。ほとんどの MySQL Cluster プログラム (ndb_size.pl を含む) に共通するオプションについては、セクション18.4.27「MySQL Cluster プログラムに共通するオプション — MySQL Cluster プログラムに共通するオプション」を参照してください。

表 18.95 この表は、ndb_size.pl プログラムのコマンド行オプションについて説明しています

形式	説明	追加または削除
--database=dbname	検査するデータベース。カンマ区切りのリストを指定できます。デフォルトは ALL です (サーバーで見つかったすべてのデータベースが使用されます)	すべての MySQL 5.6 ベースリリース
--hostname[:port]	ホストおよびオプションポートを host[:port] として指定します	すべての MySQL 5.6 ベースリリース
--socket=file	接続するソケットを指定します	すべての MySQL 5.6 ベースリリース
--user=string	MySQL ユーザー名を指定します	すべての MySQL 5.6 ベースリリース
--password=string	MySQL ユーザーのパスワードを指定します	すべての MySQL 5.6 ベースリリース
--format=string	出力形式 (テキストまたは HTML) を設定します	すべての MySQL 5.6 ベースリリース
--excludetables=tbl_list	テーブルのカンマ区切りリスト内のテーブルをスキップします	すべての MySQL 5.6 ベースリリース
--excludedbs=db_list	データベースのカンマ区切りリスト内のデータベースをスキップします	すべての MySQL 5.6 ベースリリース
--savequeries=file	データベースへのすべてのクエリーを指定したファイルに保存します	すべての MySQL 5.6 ベースリリース
--loadqueries=file	指定したファイルからすべてのクエリーをロードします。データベースに接続しません	すべての MySQL 5.6 ベースリリース
--real_table_name=table	一意インデックスサイズ計算を処理するテーブルを指定します	すべての MySQL 5.6 ベースリリース

使用法

```
perl ndb_size.pl [--database={db_name|ALL}] [--hostname=host[:port]] [--socket=socket] \
    [--user=user] [--password=password]  \
    [--help|-h] [--format={html|text}] \
```

```
    [--loadqueries=file_name] [--savequeries=file_name]
```

デフォルトでは、このユーティリティーはサーバー上のすべてのデータベースを分析しようとします。--database オプションを使用すると、単一データベースを指定できます。デフォルト動作を明示的に指定するには、データベースの名前に ALL を使用します。1 つ以上のデータベースを除外するには、--excludedbs オプションを使用して、スキップするデータベース名のカンマ区切りのリストを指定します。同様に、特定のテーブルをスキップするには、オプションの --excludetables オプションに続けて、それらの名前をカンマで区切ってリストします。ホスト名を指定するには、--hostname を使用します。デフォルトは localhost です。ホストに加えてポートを指定するには、--hostname の値に host:port 形式を使用します。デフォルトのポート番号は 3306 です。必要に応じて、ソケットも指定できます。デフォルトは /var/lib/mysql.sock です。MySQL のユーザー名およびパスワードは、表示されている対応するオプションで指定できます。出力の形式を制御するには、--format オプションを使用します。これには、値 html または text を指定でき、text がデフォルトです。テキスト出力の例を次に示します。

```
shell> ndb_size.pl --database=test --socket=/tmp/mysql.sock
ndb_size.pl report for database: 'test' (1 tables)
----------------------------------------------------
Connected to: DBI:mysql:host=localhost;mysql_socket=/tmp/mysql.sock

Including information for versions: 4.1, 5.0, 5.1

test.t1
-------

DataMemory for Columns (* means varsized DataMemory):
        Column Name           Type  Varsized   Key  4.1  5.0   5.1
     HIDDEN_NDB_PKEY         bigint           PRI    8    8     8
                c2     varchar(50)        Y       52   52    4*
                c1        int(11)                  4    4     4
                                                  --   --    --
Fixed Size Columns DM/Row                          64   64    12
  Varsize Columns DM/Row                            0    0     4

DataMemory for Indexes:
   Index Name              Type       4.1       5.0       5.1
     PRIMARY              BTREE        16        16        16
                                       --        --        --
      Total Index DM/Row                16        16        16

IndexMemory for Indexes:
             Index Name       4.1       5.0       5.1
               PRIMARY         33        16        16
                              --        --        --
          Indexes IM/Row       33        16        16

Summary (for THIS table):
                            4.1       5.0       5.1
    Fixed Overhead DM/Row    12        12        16
         NULL Bytes/Row       4         4         4
         DataMemory/Row      96        96        48
                (Includes overhead, bitmap and indexes)

  Varsize Overhead DM/Row     0         0         8
   Varsize NULL Bytes/Row     0         0         4
      Avg Varside DM/Row      0         0        16

               No. Rows       0         0         0

       Rows/32kb DM Page     340       340       680
Fixedsize DataMemory (KB)      0         0         0

Rows/32kb Varsize DM Page      0         0      2040
  Varsize DataMemory (KB)      0         0         0

        Rows/8kb IM Page     248       512       512
        IndexMemory (KB)       0         0         0

Parameter Minimum Requirements
------------------------------
* indicates greater than default

              Parameter     Default       4.1       5.0       5.1
         DataMemory (KB)       81920         0         0         0
       NoOfOrderedIndexes         128         1         1         1
            NoOfTables          128         1         1         1
         IndexMemory (KB)       18432         0         0         0
    NoOfUniqueHashIndexes          64         0         0         0
          NoOfAttributes         1000         3         3         3
           NoOfTriggers          768         5         5         5
```

デバッグのために、このスクリプトによって実行されるクエリーが含まれる Perl 配列は、--savequeries を使用してファイルに保存できます。スクリプト実行時に読み取られるそのような配列が含まれるファイルは、--loadqueries を使用して指定できます。これらのオプションにはデフォルト値はありません。

出力を HTML 形式で生成するには、次に示すように --format オプションを使用して出力をファイルにリダイレクトします。

```
shell> ndb_size.pl --database=test --socket=/tmp/mysql.sock --format=html > ndb_size.html
```

(リダイレクトしない場合、出力は stdout に送られます)。

このスクリプトの出力には次の情報が含まれています。

- 分析されるテーブルを収容するために必要な DataMemory、IndexMemory、MaxNoOfTables、MaxNoOfAttributes、MaxNoOfOrderedIndexes、MaxNoOfUniqueHashIndex および MaxNoOfTriggers 構成パラメータの最小値。
- データベースに定義されているすべてのテーブル、属性、順序付けされたインデックス、および一意のハッシュインデックスのメモリー要件。
- テーブルおよびテーブル行ごとに必要な IndexMemory および DataMemory。

18.4.26 ndb_waiter — MySQL Cluster が指定したステータスになるまで待機する

ndb_waiter は、クラスタが指定したステータスになるか、--timeout 制限を超えるまで、すべてのクラスタデータノードのステータスを繰り返し出力してから (100 ミリ秒ごと)、終了します。デフォルトでは、すべてのノードが起動されてクラスタに接続された状態である STARTED ステータスにクラスタがなるまで待機します。これは、--no-contact および --not-started オプションを使用してオーバーライドできます。

このユーティリティーによって報告されるノード状態を次に示します。

- NO_CONTACT: ノードに接続できません。
- UNKNOWN: ノードに接続できますが、ステータスがまだ不明です。通常、これはノードが管理サーバーから START または RESTART コマンドを受け取ったけれども、まだそこで実行されていないことを意味します。
- NOT_STARTED: ノードは停止されたけれども、クラスタにまだ接続されています。これは、管理クライアントの RESTART コマンドを使用してノードを再起動したときに表示されます。
- STARTING: ノードの ndbd プロセスが開始されたけれども、ノードはまだクラスタに接続されていません。
- STARTED:ノードは動作しており、クラスタに接続されています。
- SHUTTING_DOWN: ノードがシャットダウン中です。
- SINGLE USER MODE: これは、クラスタがシングルユーザーモードの場合に、すべてのクラスタデータノードに表示されます。

次の表には、MySQL Cluster ネイティブバックアップリストアプログラム ndb_waiter に固有のオプションが含まれています。追加説明が表のあとにあります。ほとんどの MySQL Cluster プログラム (ndb_waiter を含む) に共通するオプションについては、セクション18.4.27「MySQL Cluster プログラムに共通するオプション — MySQL Cluster プログラムに共通するオプション」を参照してください。

表 18.96 この表は、ndb_waiter プログラムのコマンド行オプションについて説明しています

形式	説明	追加または削除
--no-contact、 -n	クラスタが NO CONTACT 状態になるのを待機します	すべての MySQL 5.6 ベースリリース
--not-started	クラスタが NOT STARTED 状態になるのを待機します	すべての MySQL 5.6 ベースリリース
--single-user	クラスタがシングルユーザーモードになるのを待機します	すべての MySQL 5.6 ベースリリース
--timeout=#、 -t	クラスタが目的の状態になったかどうかにかかわらず、この秒数待機し	すべての MySQL 5.6 ベースリリース

形式	説明	追加または削除
	てから終了します。デフォルトは 2 分 (120 秒) です	
--nowait-nodes=list	待機しないノードのリスト。	すべての MySQL 5.6 ベースリリース
--wait-nodes=list、 -w	待機するノードのリスト。	すべての MySQL 5.6 ベースリリース

使用法

```
ndb_waiter [-c connect_string]
```

その他のオプション

- --no-contact、-n

  STARTED 状態を待機する代わりに、ndb_waiter はクラスタが NO_CONTACT ステータスになるまで実行を継続してから終了します。

- --not-started

  STARTED 状態を待機する代わりに、ndb_waiter はクラスタが NOT_STARTED ステータスになるまで実行を継続してから終了します。

- --timeout=seconds、-t seconds

  待機する時間。この秒数内に目的の状態に達しなかった場合、プログラムは終了します。デフォルトは 120 秒 (1200 レポートサイクル) です。

- --single-user

  プログラムはクラスタがシングルユーザーモードになるのを待機します。

- --nowait-nodes=list

  このオプションを使用した場合、ndb_waiter は ID がリストされているノードを待機しません。次に示すように、このリストはカンマ区切りであり、範囲はダッシュで示すことができます。

  ```
  shell> ndb_waiter --nowait-nodes=1,3,7-9
  ```

  > **重要**
  >
  > このオプションは --wait-nodes オプションと一緒に使用しないでください。

- --wait-nodes=list、-w list

  このオプションを使用した場合、ndb_waiter は ID がリストされているノードのみを待機します。次に示すように、このリストはカンマ区切りであり、範囲はダッシュで示すことができます。

  ```
  shell> ndb_waiter --wait-nodes=2,4-6,10
  ```

  > **重要**
  >
  > このオプションは --nowait-nodes オプションと一緒に使用しないでください。

出力例　4 ノードのクラスタで 2 つのノードをシャットダウンしてから手動で再度起動したときの ndb_waiter の出力を次に示します。重複するレポート (「...」 で示されています) は省略しています。

```
shell> ./ndb_waiter -c localhost

Connecting to mgmsrv at (localhost)
State node 1 STARTED
State node 2 NO_CONTACT
State node 3 STARTED
State node 4 NO_CONTACT
Waiting for cluster enter state STARTED

...
```

```
State node 1 STARTED
State node 2 UNKNOWN
State node 3 STARTED
State node 4 NO_CONTACT
Waiting for cluster enter state STARTED

...

State node 1 STARTED
State node 2 STARTING
State node 3 STARTED
State node 4 NO_CONTACT
Waiting for cluster enter state STARTED

...

State node 1 STARTED
State node 2 STARTING
State node 3 STARTED
State node 4 UNKNOWN
Waiting for cluster enter state STARTED

...

State node 1 STARTED
State node 2 STARTING
State node 3 STARTED
State node 4 STARTING
Waiting for cluster enter state STARTED

...

State node 1 STARTED
State node 2 STARTED
State node 3 STARTED
State node 4 STARTING
Waiting for cluster enter state STARTED

...

State node 1 STARTED
State node 2 STARTED
State node 3 STARTED
State node 4 STARTED
Waiting for cluster enter state STARTED

NDBT_ProgramExit: 0 - OK
```

> **注記**
>
> 接続文字列を指定しない場合、ndb_waiter は localhost の管理サーバーへの接続を試み、「Connecting to mgmsrv at (null)」を報告します。

18.4.27 MySQL Cluster プログラムに共通するオプション — MySQL Cluster プログラムに共通するオプション

すべての MySQL Cluster プログラムは、次の例外を除き、このセクションで説明しているオプションを受け入れます。

- mysqld
- ndb_print_backup_file
- ndb_print_schema_file
- ndb_print_sys_file

以前のバージョンの MySQL Cluster のユーザーは、これらのオプションの一部が相互に整合性を保つように、および mysqld と整合性を保つように変更されていることに留意してください。プログラムでサポートされるオプションのリストを表示するには、MySQL Cluster プログラム (ndb_print_backup_file、ndb_print_schema_file、および ndb_print_sys_file を除く) で --help オプションを使用できます。

次の表のオプションは、すべての MySQL Cluster 実行可能ファイル (このセクションで前述したものを除く) に共通しています。

表 18.97 この表は、すべての MySQL Cluster プログラムに共通するコマンド行オプションについて説明しています

形式	説明	追加または削除
--help、 --usage、 -?	ヘルプメッセージを表示して終了	すべての MySQL 5.6 ベースリリース
--ndb-connectstring=connectstring、 --connect-string=connectstring、 -c	ndb_mgmd に接続するための接続文字列を設定します。構文: [nodeid=<id>;][host=]<hostname>[:<port>]NDB_CONNECTSTRING または my.cnf に指定されているエントリをオーバーライドします。	すべての MySQL 5.6 ベースリリース
--ndb-mgmd-host=host[:port]	管理サーバーに接続するためのホスト (およびポート (必要な場合)) を設定します	すべての MySQL 5.6 ベースリリース
--ndb-nodeid=#	このノードのノード ID を設定します	すべての MySQL 5.6 ベースリリース
--ndb-optimized-node-selection	トランザクションのためのノードをより最適な方法で選択します	すべての MySQL 5.6 ベースリリース
--character-sets-dir=path	文字セットがインストールされているディレクトリ	すべての MySQL 5.6 ベースリリース
--debug=options	デバッグ呼び出しからの出力を有効にします。デバッグを有効にしてコンパイルされたバージョンにのみ使用できます	すべての MySQL 5.6 ベースリリース
--core-file	エラー時にコアを書き出します (デバッグ用ビルドでは TRUE にデフォルト設定されます)	すべての MySQL 5.6 ベースリリース
--version、 -V	バージョン情報を出力して終了します	すべての MySQL 5.6 ベースリリース

個別の MySQL Cluster プログラムに固有のオプションについては、セクション18.4「MySQL Cluster プログラム」を参照してください。

MySQL Cluster に関連する mysqld のオプションについては、セクション18.3.4.2「MySQL Cluster 用の MySQL Server オプション」を参照してください。

- --help、--usage、-?

コマンド行形式	--help --usage

使用可能なコマンドオプションの説明の短いリストを出力します。

- --character-sets-dir=name

コマンド行形式	--character-sets-dir=path
型	ファイル名
デフォルト	

文字セット情報を探す場所をプログラムに通知します。

- --ndb-connectstring=connect_string、--connect-string=connect_string、-c connect_string

コマンド行形式	--ndb-connectstring=connectstring --connect-string=connectstring

型	文字列
デフォルト	localhost:1186

このオプションには、次に示すようにアプリケーションが接続する管理サーバーを指定する MySQL Cluster 接続文字列を指定します。

```
shell> ndbd --ndb-connectstring="nodeid=2;host=ndb_mgmd.mysql.com:1186"
```

詳細は、セクション18.3.2.3「MySQL Cluster の接続文字列」を参照してください。

- --core-file

コマンド行形式	--core-file
型	ブール
デフォルト	FALSE

プログラムが停止した場合、コアファイルが出力されます。コアファイルの名前および場所はシステムによって異なります(Linux 上で実行されている MySQL Cluster プログラムノードの場合、デフォルトの場所はプログラムの作業ディレクトリであり、データノードの場合は、ノードの DataDir です)。一部のシステムでは、制限がある場合があります。たとえば、サーバーを起動する前に ulimit -c unlimited を実行する必要がある場合があります。詳細は、システムのドキュメントを参照してください。

MySQL Cluster が configure に --debug オプションを使用してビルドされた場合、--core-file がデフォルトで有効になります。通常のビルドの場合、デフォルトでは --core-file は無効にされています。

- --debug[=options]

コマンド行形式	--debug=options
型	文字列
デフォルト	d:t:O,/tmp/ndb_restore.trace

このオプションは、デバッグを有効にしてコンパイルしたバージョンにのみ使用できます。mysqld プロセスの場合と同様に、デバッグ呼び出しからの出力を有効にするために使用します。

- --ndb-mgmd-host=host[:port]

コマンド行形式	--ndb-mgmd-host=host[:port]
型	文字列
デフォルト	localhost:1186

プログラムに接続される単一の管理サーバーのホストとポート番号の設定に使用できます。プログラムが接続情報に複数の管理サーバーのノード ID または参照 (あるいはその両方) を必要とする場合は、--ndb-connectstring オプションを代わりに使用します。

- --ndb-nodeid=#

コマンド行形式	--ndb-nodeid=#
型	数値
デフォルト	0

このノードの MySQL Cluster ノード ID を設定します。許可される値の範囲は、ノードのタイプ (データ、管理、または API) および MySQL Cluster ソフトウェアのバージョンによって異なります。詳細は、セクション18.1.6.2「MySQL Cluster の制限と標準の MySQL の制限との違い」を参照してください。

- --ndb-optimized-node-selection

コマンド行形式	--ndb-optimized-node-selection
型	ブール
デフォルト	TRUE

トランザクションのためのノードの選択を最適化します。デフォルトで有効。

- --version、-V

コマンド行形式	--version

実行可能ファイルの MySQL Cluster バージョン番号を出力します。すべてのバージョンを一緒に使用できるとはかぎらないため、バージョン番号が重要になります。MySQL Cluster 起動処理では、使用されるバイナリのバージョンが同じクラスタ内で共存できるかどうかが検証されます。これは MySQL Cluster のオンライン (ローリング) ソフトウェアアップグレードまたはダウングレードを実行する場合にも重要です。

詳細は、セクション18.5.5「MySQL Cluster のローリング再起動の実行」を参照してください。

18.5 MySQL Cluster の管理

MySQL Cluster の管理には、多くのタスクが伴い、その最初のタスクは、MySQL Cluster の構成および起動です。これについては、セクション18.3「MySQL Cluster の構成」およびセクション18.4「MySQL Cluster プログラム」で説明されています。

次のいくつかのセクションでは、実行中の MySQL Cluster の管理について説明します。

MySQL Cluster の管理および配備に関するセキュリティー上の問題については、セクション18.5.11「MySQL Cluster のセキュリティー上の問題」を参照してください。

実行中の MySQL Cluster を積極的に管理する方法は、基本的に 2 つあります。それらのうちの 1 つ目は、管理クライアントに入力されたコマンドを使用するものです。これにより、クラスタのステータスをチェックしたり、ログレベルを変更したり、バックアップを開始および停止したり、ノードを停止および起動したりできます。2 つ目の方法には、クラスタログ ndb_node_id_cluster.log の内容の調査が伴います。通常、これは管理サーバーの DataDir ディレクトリで見つかりますが、この場所は LogDestination オプションを使用すればオーバーライドできます。(node_id は、アクティビティーのログが記録されるノードの一意の識別子を表すことを思い出してください。)クラスタログには、ndbd で生成されたイベントレポートが含まれます。クラスタログのエントリを Unix システムログに送信することもできます。

SHOW ENGINE NDB STATUS ステートメントを使用して、SQL ノードからクラスタ操作のいくつかの側面をモニターすることもできます。

ndbinfo データベースを使用すると、SQL インタフェースから MySQL Cluster の操作に関するより詳細な情報をリアルタイムで入手できます。詳細は、セクション18.5.10「ndbinfo MySQL Cluster 情報データベース」を参照してください。

NDB 統計カウンタでは、mysql クライアントを使用したモニタリングが改善されています。NDB カーネルに実装されたこれらのカウンタは、Ndb オブジェクトによって実行されたり、それらのオブジェクトに影響を与えたりする操作 (トランザクションの開始、終了、および中止、主キーおよび一意キー操作、テーブルスキャン、範囲スキャン、およびプルーニングスキャン、さまざまな操作が完了するまで待機しているブロックされたスレッド、MySQL Cluster によって送受信されたデータやイベントなど) に関連します。カウンタは、NDB API が呼び出されるたびに、またはデータノードでデータが送受信されるたびに NDB カーネルによってインクリメントされます。

mysqld では、NDB API 統計カウンタがシステムステータス変数として表示されます。これらは、すべての名前に共通するプリフィクス (Ndb_api_) から識別できます。これらの変数の値は、SHOW STATUS ステートメントの出力から mysql クライアントで読み取ることも、(INFORMATION_SCHEMA データベース内の) SESSION_STATUS テーブルまたは GLOBAL_STATUS テーブルでクエリーを実行することで読み取ることもできます。NDB テーブルに対して機能する SQL ステートメントの実行前後でステータス変数の値を比較すると、NDB API レベルで実行され、このステートメントに対応するアクションを確認できます。これは、MySQL Cluster のモニタリングおよびパフォーマンスチューニングの際に役立つことがあります。

MySQL Enterprise Monitor を使用して、MySQL Cluster 配備の一部である MySQL サーバーをモニターすることもできます。MySQL Enterprise Monitor 2.3 では、MySQL Cluster リソースに関する情報を提供するグラフのセットを含み、DataMemory 使用状況などのデータノードからの重要な情報に関するアラートのルールを定義する MySQL Cluster アドバイザが追加されました。MySQL Cluster に接続された任意の MySQL サーバーで ndbinfo を使用すると、この情報を MySQL Enterprise Monitor 2.3 以降で使用できるようになります。アドバイザは、クラスタ内の単一の MySQL サーバーに対して、またはモニタリングサービスの可用性レベルを高くするためにペアに対して実行できます。詳細は、MySQL Enterprise Monitor 2.3 マニュアルを参照してください。

MySQL Cluster Manager は、多数のノードを含む MySQL Cluster の起動、停止、再起動など、多くの、それがなければ複雑になる MySQL Cluster の管理タスクを簡単にする高度なコマンド行インタフェースを備えています。MySQL Cluster Manager クライアントでは、MySQL Cluster に関連する mysqld サーバーのオプションおよび変数に加えて、ほとんどのノード構成パラメータの値を取得および設定するためのコマンドもサポートし

ています。MySQL Cluster Manager 1.1 では、データノードのオンラインでの追加をサポートしています。詳細は、MySQL Cluster Manager 1.1 のユーザーマニュアルを参照してください。

18.5.1 MySQL Cluster の起動フェーズのサマリー

このセクションでは、MySQL Cluster データノードの起動時に含まれるステップについて簡単な概要を示します。『NDB Internals Guide』のNDB Cluster Start Phasesでは、さらに完全な情報を見つけることができます。

これらのフェーズは、管理クライアントで node_id STATUS コマンドからの出力にレポートされるものと同じです (セクション18.5.2「MySQL Cluster 管理クライアントのコマンド」を参照してください)。これらの起動フェーズは、ndbinfo.nodes テーブルの start_phase カラムにもレポートされます。

起動のタイプ　次のリストに示すように、さまざまな起動のタイプおよびモードがあります。

- 初期起動　すべてのデータノード上のクリーンなファイルシステムでクラスタが起動します。これは、まったくはじめてクラスタが起動されるとき、または --initial オプションを使用してすべてのデータノードが再起動されるときに発生します。

  > **注記**
  >
  > --initial を使用してノードを再起動すると、ディスクデータファイルが削除されません。

- システムの再起動　クラスタが起動し、データノードに格納されたデータを読み取ります。これは、クラスタが使用されたあとにシャットダウンされたとき、およびクラスタが操作を中止した時点から再開することが望ましいときに発生します。
- ノードの再起動　これは、クラスタ自体が実行しているときのクラスタノードのオンライン再起動です。
- ノードの初期再起動　これは、クリーンなファイルシステムでノードが再初期化および起動される点を除いて、ノードの再起動と同じです。

設定と初期化 (フェーズ -1)　起動する前に、各データノード (ndbd プロセス) が初期化されている必要があります。初期化は次のステップで構成されます。

1. ノード ID を取得します
2. 構成データをフェッチします
3. ノード間通信で使用されるポートを割り当てます
4. 構成ファイルから取得された設定に従って、メモリーを割り当てます

データノードまたは SQL ノードがはじめて管理ノードに接続すると、クラスタノード ID を予約します。ほかのノードによって同じノード ID が割り当てられないように、この ID は、ノードからクラスタへの接続が完了し、このノードが接続されたことが少なくとも 1 つの ndbd でレポートされるまで保持されます。このようなノード ID の保持は、該当するノードと ndb_mgmd 間の接続で保護されます。

各データノードが初期化されたら、クラスタの起動プロセスに進むことができます。このプロセス中にクラスタが進む段階を次に示します。

- フェーズ 0　NDBFS および NDBCNTR ブロックが起動します (NDB Kernel Blocksを参照してください)。--initial オプションを付けて起動されたデータノード上で、データノードファイルシステムがクリアされます。
- フェーズ 1　この段階では、残りのすべての NDB カーネルブロックが起動されます。MySQL Cluster 接続が設定され、ブロック間の通信が確立され、ハートビートが開始されます。ノードの再起動の場合は、API ノードの接続もチェックされます。

  > **注記**
  >
  > フェーズ 1 で 1 つ以上のノードがハングアップし、フェーズ 2 で残りの 1 つまたは複数のノードがハングアップするときは、多くの場合にネットワークの問題を示しています。このような問題が発生する原因の 1 つとして、複数のネットワークインタフェースを持つ 1 つ以上のクラスタホストが考えられます。このような状況が発生する問題のもう 1 つの一般的な原因は、クラスタノード間の通信に必要な TCP/IP ポートのブロックです。後者では、これは多くの場合に、ファイアウォールが誤って構成されているためです。

- フェーズ 2　NDBCNTR カーネルブロックは、既存のすべてのノードの状態をチェックします。マスターノードが選択され、クラスタスキーマファイルが初期化されます。

- フェーズ 3　DBLQH および DBTC カーネルブロックは、それらの間の通信を設定します。起動タイプが特定されます。これが再起動の場合、DBDIH ブロックは再起動を実行するための権限を取得します。
- フェーズ 4　初期起動またはノードの初期再起動の場合、Redo ログファイルが作成されます。これらのファイルの数は、NoOfFragmentLogFiles に等しいです。

  システムの再起動の場合:

  - 1 つまたは複数のスキーマを読み取ります。
  - ローカルチェックポイントからデータを読み取ります。
  - 最新のリストア可能なグローバルチェックポイントに達するまで、すべての Redo 情報を適用します。

  ノードの再起動の場合、Redo ログの末尾を検索します。
- フェーズ 5　このフェーズ中に、データノード起動のデータベース関連部分のほとんどが実行されます。初期起動またはシステムの再起動の場合、ローカルチェックポイントに続いて、グローバルチェックポイントが実行されます。このフェーズ中に、メモリー使用率の定期的なチェックが開始され、必要なノードのテイクオーバーが実行されます。
- フェーズ 6　このフェーズでは、ノードグループが定義および設定されます。
- フェーズ 7　アービトレータノードが選択され、動作が開始されます。次のバックアップ ID、およびバックアップディスクの書き込み速度が設定されます。この起動フェーズに到達したノードは、Started とマークされます。この時点で、API ノード (SQL ノードを含む) はクラスタに接続できます。
- フェーズ 8　これがシステムの再起動の場合、すべてのインデックスが (DBDIH によって) 再構築されます。
- フェーズ 9　ノード内部の起動変数がリセットされます。
- フェーズ 100 (廃止)　以前は、ノードの再起動またはノードの初期再起動のこの時点で、API ノードはノードに接続し、イベントの受信を開始できました。現在は、このフェーズが空になっています。
- フェーズ 101　ノードの再起動またはノードの初期再起動のこの時点で、クラスタに参加するノードにイベント配信が渡されます。新たに参加したノードは、そのプライマリデータをサブスクライバに配信する責任を引き受けます。このフェーズは、SUMA ハンドオーバーフェーズとも呼ばれます。

初期起動またはシステムの再起動の場合、このプロセスが完了すると、トランザクションの処理が有効になります。ノードの再起動またはノードの初期再起動の場合、起動プロセスが完了したことは、現在ノードがトランザクションコーディネータとして動作している可能性があることを意味します。

18.5.2 MySQL Cluster 管理クライアントのコマンド

主要な構成ファイルに加えて、管理クライアント ndb_mgm から使用可能なコマンド行インタフェースによって、クラスタを制御することもできます。これは、実行中のクラスタへのプライマリ管理インタフェースです。

イベントログのコマンドは、セクション18.5.6「MySQL Cluster で生成されたイベントレポート」に示しています。バックアップを作成するためのコマンドと、バックアップからリストアするためのコマンドは、セクション18.5.3「MySQL Cluster のオンラインバックアップ」で説明しています。

MySQL Cluster Manager での ndb_mgm の使用　MySQL Cluster Manager では、プロセスの起動と停止が処理され、プロセスの状態が内部で追跡されるため、MySQL Cluster Manager の制御下にある MySQL Cluster では、これらのタスクに、ndb_mgm を使用する必要がありません。ノードの起動または停止が伴う操作を実行する際には、MySQL Cluster 配布に付属する ndb_mgm コマンド行クライアントを使用しないことをお勧めします。これらには、START、STOP、RESTART、および SHUTDOWN が含まれますが、これだけに限定されません。詳細は、MySQL Cluster Manager Process Commandsを参照してください。

管理クライアントには、次のような基本的なコマンドが用意されています。次のリストでは、node_id はデータベースノード ID、またはクラスタのデータノードのすべてにコマンドを適用すべきことを示すキーワード ALL を示します。

- HELP

  使用可能なすべてのコマンドに関する情報を表示します。
- SHOW

  クラスタのステータスに関する情報を表示します。可能性のあるノードステータスの値には、UNKNOWN、NO_CONTACT、NOT_STARTED、STARTING、STARTED、SHUTTING_DOWN、および

RESTARTING が含まれます。このコマンドからの出力には、クラスタがシングルユーザーモード (ステータス SINGLE USER MODE) になるタイミングも示されます。

- node_id START

  node_id によって識別されるデータノード (またはすべてのデータノード) をオンラインにします。

  ALL START はすべてのデータノードでのみ動作し、管理ノードには影響を与えません。

  > **重要**
  >
  > このコマンドを使用してデータノードをオンラインにするには、ndbd --nostart または ndbd -n を使用してデータノードが起動されている必要があります。

- node_id STOP [-a] [-f]

  node_id によって識別されるデータノードまたは管理ノードを停止します。ALL STOP はすべてのデータノードを停止させるためにのみ機能し、管理ノードには影響を与えません。

  このコマンドの影響を受けるノードはクラスタから切断され、それに関連付けられた ndbd または ndb_mgmd プロセスは終了します。

  -a オプションを使用すると、保留中のトランザクションが完了するまで待機せずに、すぐにノードが停止されます。

  通常、結果として不完全なクラスタが生成される場合は、STOP に失敗します。-f オプションを使用すると、これをチェックせずに、強制的にノードがシャットダウンされます。このオプションが使用され、結果が不完全なクラスタである場合は、すぐにクラスタがシャットダウンします。

  > **警告**
  >
  > また、-a オプションを使用すると、ノードが停止されても不完全なクラスタが生成されないように、STOP の呼び出し時に本来実行される安全性チェックも無効になります。つまり、STOP コマンドで -a オプションを使用すると、クラスタは NDB に格納されたすべてのデータの完全なコピーを保持しなくなるため、強制的にシャットダウンされる可能性があることにより、このオプションを使用する際は、細心の注意を払うようにしてください。

- node_id RESTART [-n] [-i] [-a] [-f]

  node_id によって識別されるデータノード (またはすべてのデータノード) を再起動します。

  RESTART で -i オプションを使用すると、データノードが初期再起動を実行します。つまり、ノードのファイルシステムが削除され、再作成されます。この効果は、データノードのプロセスを停止してから、システムシェルから ndbd --initial を使用して再起動することで実現される効果と同じです。このオプションを使用しても、バックアップファイルおよびディスクデータファイルは削除されません。

  -n オプションを使用するとデータノードプロセスが再起動されますが、実際には、適切な START コマンドが発行されるまでデータノードはオンラインになりません。このオプションの効果は、データノードを停止してから、システムシェルから ndbd --nostart または ndbd -n を使用して再起動することで実現される効果と同じです。

  -a を使用すると、このノードに依存している現在のトランザクションがすべて中止されます。ノードがクラスタに再度参加するときは、GCP チェックが実行されません。

  通常、ノードをオフラインにしたことで不完全なクラスタが生成される場合は、RESTART が失敗します。-f オプションを使用すると、これをチェックせずに、強制的にノードが再起動されます。このオプションが使用され、結果が不完全なクラスタである場合は、クラスタ全体が再起動されます。

- node_id STATUS

  node_id によって識別されるデータノード (またはすべてのデータノード) のステータス情報を表示します。

  このコマンドからの出力には、クラスタがシングルユーザーモードになるタイミングも示されます。

- node_id REPORT report-type

  node_id によって識別されるデータノード、または ALL を使用するすべてのデータノードに対する report-type タイプのレポートを表示します。

現在、report-type には次の 2 つの値が許可されています。

- BackupStatus では、進行中のクラスタバックアップに関するステータスレポートが提供されます。
- MemoryUsage では、次の例で示すように、各データノードで使用されているデータメモリーおよびインデックスメモリーの量が表示されます。

```
ndb_mgm> ALL REPORT MEMORY

Node 1: Data usage is 5%(177 32K pages of total 3200)
Node 1: Index usage is 0%(108 8K pages of total 12832)
Node 2: Data usage is 5%(177 32K pages of total 3200)
Node 2: Index usage is 0%(108 8K pages of total 12832)
```

この情報は、ndbinfo.memoryusage テーブルから取得することもできます。

report-type は大文字と小文字が区別されず、「あいまい」です。MemoryUsage では、MEMORY (前述の例で示すとおり)、memory、または単に MEM (または mem) を使用できます。BackupStatus も同様に短縮できます。

- ENTER SINGLE USER MODE node_id

シングルユーザーモードを開始します。これにより、ノード ID node_id によって識別される MySQL サーバーにのみデータベースへのアクセスが許可されます。

現在、シングルユーザーモードで実行しているときは、データノードが MySQL Cluster に参加できません。(Bug #20395)

- EXIT SINGLE USER MODE

シングルユーザーモードを終了します。これにより、すべての SQL ノード (つまり、実行中のすべての mysqld プロセス) がデータベースにアクセスできます。

> **注記**
>
> シングルユーザーモードでないときでも、EXIT SINGLE USER MODE を使用できますが、この場合、コマンドの効果はありません。

- QUIT, EXIT

管理クライアントを終了します。

このコマンドは、クラスタに接続されているノードに影響を与えません。

- SHUTDOWN

すべてのクラスタデータノードおよび管理ノードをシャットダウンします。これが実行されたあとに管理クライアントを終了するには、EXIT または QUIT を使用します。

このコマンドは、クラスタに接続しているどの SQL ノードまたは API ノードもシャットダウンしません。

- CREATE NODEGROUP nodeid[, nodeid, ...]

新しい MySQL Cluster ノードグループを作成し、これにデータノードを参加させます。

このコマンドは、新しいデータノードをオンラインで MySQL Cluster に追加したあとに使用し、それらを新しいノードグループに参加させ、それによってクラスタへの完全な参加を開始します。このコマンドは唯一のパラメータとして、ノード ID をカンマで区切ったリストを取りますが、これらは、先ほど追加した、新しいノードグループに参加することになるノードの ID です。ノードの数は、すでにクラスタの一部になっている各ノードグループ内のノードの数と同じである必要があります (各 MySQL Cluster ノードグループは、同じ数のノードを持つ必要があります)。言い換えると、MySQL Cluster にそれぞれ 2 つのデータノードを持つ 2 つのノードグループが含まれている場合は、新しいノードグループも 2 つのデータノードを持つ必要があります。

このコマンドで作成された新しいノードグループのノードグループ ID は自動的に決定され、常に、クラスタ内で次に最大の未使用のノードグループ ID となり、手動で設定することはできません。

詳細は、セクション18.5.13「MySQL Cluster データノードのオンライン追加」を参照してください。

- DROP NODEGROUP nodegroup_id

指定された nodegroup_id を持つ MySQL Cluster ノードグループを削除します。

このコマンドを使用すると、MySQL Cluster からノードグループを削除できます。DROP NODEGROUP は唯一の引数として、削除するノードグループのノードグループ ID を取ります。

DROP NODEGROUP は、影響を受けるノードグループ内のデータグループをそのノードグループから削除するためにのみ機能します。データノードは停止されず、別のノードグループにも割り当てられず、クラスタの構成からも削除されません。ノードグループに属さないデータノードは、次のように (太字のテキストを使用して示すように)、ノードグループ ID の代わりに、no nodegroup を付けた管理クライアントの SHOW コマンドの出力に表示されます。

```
id=3    @10.100.2.67  (5.6.22-ndb-7.4.4, no nodegroup)
```

MySQL Cluster NDB 7.0.4 よりも前では、DROP NODEGROUP のあとに SHOW の出力が正しく更新されませんでした。(Bug #43413)

DROP NODEGROUP は、削除されるノードグループ内のすべてのデータノードに、テーブルデータやテーブル定義がまったく存在しない場合にのみ機能します。現在は、ndb_mgm または mysql クライアントを使用して、特定のデータノードやノードグループからすべてのデータを削除する方法がないため、これは次のような 2 つの場合にのみコマンドが成功することを意味します。

1. ndb_mgm クライアントで CREATE NODEGROUP を発行してから、mysql クライアントで任意の ALTER ONLINE TABLE ... REORGANIZE PARTITION ステートメントを発行するまでの間。
2. DROP TABLE を使用してすべての NDBCLUSTER テーブルを削除したあと。

   TRUNCATE TABLE はテーブルデータのみを削除するため、この目的では機能しません。テーブルのメタデータが削除される DROP TABLE ステートメントが発行されるまで、データノードには引き続き NDBCLUSTER テーブルの定義が格納されます。

DROP NODEGROUP の詳細は、セクション18.5.13「MySQL Cluster データノードのオンライン追加」を参照してください。

18.5.3 MySQL Cluster のオンラインバックアップ

次のいくつかのセクションでは、ndb_mgm 管理クライアントにあるその目的のための機能を使用して、MySQL Cluster バックアップを準備してから作成する方法について説明します。このタイプのバックアップと mysqldump を使用して作成されたバックアップを区別するために、これを「ネイティブ」 MySQL Cluster バックアップと呼ぶこともあります。(mysqldump を使用したバックアップの作成については、セクション 4.5.4「mysqldump — データベースバックアッププログラム」を参照してください。)MySQL Cluster バックアップのリストアは、MySQL Cluster 配布で提供される ndb_restore ユーティリティーを使用して実行されます。ndb_restore について、およびこれを MySQL Cluster バックアップのリストア時に使用する方法については、セクション18.4.20「ndb_restore — MySQL Cluster バックアップのリストア」を参照してください。

18.5.3.1 MySQL Cluster バックアップの概念

バックアップとは、特定の時点でのデータベースのスナップショットです。バックアップは次の 3 つの主な部分で構成されます。

- メタデータ　　すべてのデータベーステーブルの名前と定義
- テーブルレコード　　バックアップが作成された時点でデータベーステーブルに実際に格納されたデータ
- トランザクションログ　　データがデータベースに格納された方法と日時を示す順次レコード

これらの各部分は、バックアップに参加しているすべてのノード上に保存されます。バックアップ中に、各ノードはこれらの 3 つの部分をディスク上の 3 つのファイルに保存します。

- BACKUP-backup_id.node_id.ctl

  制御情報およびメタデータを含む制御ファイルです。各ノードは、このファイルの各ノード独自のバージョンに、(クラスタ内のすべてのテーブルの) 同じテーブル定義を保存します。

- BACKUP-backup_id-0.node_id.data

  フラグメントごとに保存され、テーブルレコードを含むデータファイルです。つまり、バックアップ中にノードによって異なるフラグメントを保存します。各ノードによって保存されたファイルは、レコードが属する

テーブルを示すヘッダーで始まります。レコードのリストのあとに、すべてのレコードのチェックサムを含むフッターが続きます。

- BACKUP-backup_id.node_id.log

  コミットされたトランザクションのレコードを含むログファイルです。このログには、バックアップに格納されたテーブル上のトランザクションのみが格納されます。ノードによって異なるデータベースのフラグメントをホストするため、バックアップに関与するノードはさまざまなレコードを保存します。

上記のリストでは、backup_id はバックアップ識別子を表し、node_id はファイルを作成するノードの一意の識別子を表します。

18.5.3.2 MySQL Cluster 管理クライアントを使用したバックアップの作成

バックアップを開始する前に、バックアップを実行するためにクラスタが適切に構成されていることを確認します。(セクション18.5.3.3「MySQL Cluster バックアップ用の構成」を参照してください。)

START BACKUP コマンドは、バックアップを作成するために使用されます。

```
START BACKUP [backup_id] [wait_option] [snapshot_option]

wait_option:
WAIT {STARTED | COMPLETED} | NOWAIT

snapshot_option:
SNAPSHOTSTART | SNAPSHOTEND
```

連続したバックアップは自動的に順次識別されるため、backup_id (1 以上の整数) はオプションです。これを省略すると、次に使用可能な値が使用されます。既存の backup_id 値が使用されると、 Backup failed: file already existsというエラーでバックアップに失敗します。これを使用する場合は、その他のオプションを使用する前に、START BACKUP の直後に backup_id を指定する必要があります。

wait_option を使用すると、次のリストに示すように、START BACKUP コマンドを発行したあとに、管理クライアントに制御が返されるタイミングを判断できます。

- NOWAIT を指定すると、すぐに次に示すようなプロンプトが管理クライアントに表示されます。

  ```
  ndb_mgm> START BACKUP NOWAIT
  ndb_mgm>
  ```

  この場合、バックアッププロセスから進行状況の情報が出力されている間でも、管理クライアントを使用できます。

- WAIT STARTED を指定すると、次に示すように、管理クライアントはバックアップが開始されるまで待機してから、ユーザーに制御を返します。

  ```
  ndb_mgm> START BACKUP WAIT STARTED
  Waiting for started, this may take several minutes
  Node 2: Backup 3 started from node 1
  ndb_mgm>
  ```

- WAIT COMPLETED を指定すると、管理クライアントはバックアッププロセスが完了するまで待機してから、ユーザーに制御を返します。

WAIT COMPLETED はデフォルトです。

snapshot_option を使用すると、START BACKUP が発行されたとき、またはそれが完了したときのクラスタの状態とバックアップが一致するかどうかを判断できます。SNAPSHOTSTART を使用すると、バックアップがバックアップを開始したときのクラスタの状態と一致します。SNAPSHOTEND を使用すると、バックアップはバックアップが終了したときのクラスタの状態を反映します。SNAPSHOTEND はデフォルトであり、以前の MySQL Cluster リリースで見られた動作と一致します。

> **注記**
>
> START BACKUP を付けて SNAPSHOTSTART を使用するときに、CompressedBackup パラメータが有効になっている場合、データファイルおよび制御ファイルのみが圧縮され、ログファイルは圧縮されません。

wait_option と snapshot_option の両方を使用する場合、それらはいずれの順序でも指定できます。たとえば、ID として 4 を持つ既存のバックアップが存在しないと仮定すると、次のコマンドはすべて有効です。

```
START BACKUP WAIT STARTED SNAPSHOTSTART
START BACKUP SNAPSHOTSTART WAIT STARTED
START BACKUP 4 WAIT COMPLETED SNAPSHOTSTART
START BACKUP SNAPSHOTEND WAIT COMPLETED
START BACKUP 4 NOWAIT SNAPSHOTSTART
```

バックアップを作成する手順は、次のステップで構成されます。

1. 管理クライアント (ndb_mgm) がまだ実行されていない場合は、起動します。

2. START BACKUP コマンドを実行します。これにより、次に示すように、バックアップの進行状況を示す複数行の出力が生成されます。

   ```
   ndb_mgm> START BACKUP
   Waiting for completed, this may take several minutes
   Node 2: Backup 1 started from node 1
   Node 2: Backup 1 started from node 1 completed
    StartGCP: 177 StopGCP: 180
    #Records: 7362 #LogRecords: 0
    Data: 453648 bytes Log: 0 bytes
   ndb_mgm>
   ```

3. バックアップが開始されると、次のメッセージが管理クライアントに表示されます。

   ```
   Backup backup_id started from node node_id
   ```

   backup_id は、特定のバックアップを表す一意の識別子です。ほかに構成されていない場合、この識別子はクラスタログに保存されます。node_id は、データノードとバックアップを調整する管理サーバーの識別子です。バックアッププロセスのこの時点で、クラスタはバックアップリクエストを受信して処理しています。バックアップが終了したことを意味するわけではありません。次に、このステートメントの例を示します。

   ```
   Node 2: Backup 1 started from node 1
   ```

4. 管理クライアントは、次のようなメッセージでバックアップが開始されたことを示します。

   ```
   Backup backup_id started from node node_id completed
   ```

   バックアップが開始されたことを示す通知の場合も同様に、backup_id は、この特定のバックアップを表す一意の識別子であり、node_id は、データノードとバックアップを調整する管理サーバーのノード ID です。この出力には、次に示すように、関連するグローバルチェックポイント、バックアップされたレコードの数、およびデータのサイズを含む追加情報が記載されます。

   ```
   Node 2: Backup 1 started from node 1 completed
    StartGCP: 177 StopGCP: 180
    #Records: 7362 #LogRecords: 0
    Data: 453648 bytes Log: 0 bytes
   ```

また、次の例に示すように、-e または --execute オプションを付けて ndb_mgm を呼び出して、システムシェルからバックアップを実行することもできます。

```
shell> ndb_mgm -e "START BACKUP 6 WAIT COMPLETED SNAPSHOTSTART"
```

この方法で START BACKUP を使用する際は、バックアップ ID を指定する必要があります。

クラスタのバックアップは、デフォルトで各データノード上の DataDir の BACKUP サブディレクトリに作成されます。これは、1 つ以上のデータノードごとに個別にオーバーライドすることも、BackupDataDir 構成パラメータを使用することで、config.ini ファイル内のすべてのクラスタデータノードに対してオーバーライドすることもできます。特定の backup_id を持つバックアップ用に作成されたバックアップファイルは、バックアップディレクトリ内の BACKUP-backup_id という名前のサブディレクトリに格納されます。

すでに進行中のバックアップを中止するには:

1. 管理クライアントを起動します。

2. 次のコマンドを実行します。

   ```
   ndb_mgm> ABORT BACKUP backup_id
   ```

   数値 backup_id は、バックアップが開始されたときに管理クライアントの応答 (「Backup backup_id started from node management_node_id」メッセージ内) に含まれていたバックアップの識別子です。

3. 管理クライアントは、「Abort of backup backup_id ordered」で中止リクエストを承認します。

> **注記**
>
> この時点で、管理クライアントはまだクラスタデータノードからこのリクエストへの応答を受信していないため、実際にはバックアップはまだ中止されていません。

4. バックアップが中止されると、管理クライアントは次に示すものと同様の方法で、このことをレポートします。

```
Node 1: Backup 3 started from 5 has been aborted.
  Error: 1321 - Backup aborted by user request: Permanent error: User defined error
Node 3: Backup 3 started from 5 has been aborted.
  Error: 1323 - 1323: Permanent error: Internal error
Node 2: Backup 3 started from 5 has been aborted.
  Error: 1323 - 1323: Permanent error: Internal error
Node 4: Backup 3 started from 5 has been aborted.
  Error: 1323 - 1323: Permanent error: Internal error
```

この例では、4 つのデータノードを持つクラスタのサンプル出力を示します。ここで、中止されるバックアップのシーケンス番号は 3 で、クラスタ管理クライアントが接続される管理ノードのノード ID は 5 です。バックアップの中止時にその役割を最初に完了したノードは、中止の理由がユーザーからのリクエストのためであったことをレポートします。(残りのノードは、不明な内部エラーが原因でバックアップが中止されたことをレポートします。)

> **注記**
>
> クラスタノードが特定の順序で ABORT BACKUP コマンドに応答する保証はありません。

「Backup backup_id started from node management_node_id has been aborted」というメッセージは、バックアップが終了し、このバックアップに関連するすべてのファイルがクラスタファイルシステムから削除されたことを示しています。

このコマンドを使用すると、システムシェルから進行中のバックアップを中止することもできます。

```
shell> ndb_mgm -e "ABORT BACKUP backup_id"
```

> **注記**
>
> ABORT BACKUP の発行時に、ID backup_id を持つバックアップが実行されていない場合は、管理クライアントが応答することも、クラスタログに無効な中止コマンドが送信されたことが表示されることもありません。

18.5.3.3 MySQL Cluster バックアップ用の構成

バックアップには、次の 5 つの構成パラメータが不可欠です。

- BackupDataBufferSize

  ディスクに書き込む前のデータをバッファーに入れる際に使用されるメモリーの量。

- BackupLogBufferSize

  ディスクに書き込む前のログレコードをバッファーに入れる際に使用されるメモリーの量。

- BackupMemory

  バックアップ用にデータノードに割り当てられた合計メモリー。これは、バックアップデータバッファーとバックアップログバッファー用に割り当てられたメモリーの合計になります。

- BackupWriteSize

  ディスクに書き込まれたブロックのデフォルトサイズ。これは、バックアップデータバッファーとバックアップログバッファーの両方に適用されます。

- BackupMaxWriteSize

  ディスクに書き込まれたブロックの最大サイズ。これは、バックアップデータバッファーとバックアップログバッファーの両方に適用されます。

これらのパラメータに関する詳細は、「バックアップパラメータ」で見つかります。

18.5.3.4 MySQL Cluster バックアップのトラブルシューティング

バックアップリクエストを発行した際にエラーコードが返される場合は、不十分なメモリーまたはディスク容量が原因である可能性がもっとも高いです。バックアップ用に十分なメモリーが割り当てられていることを確認するようにしてください。

> **重要**
>
> BackupDataBufferSize および BackupLogBufferSize を設定していて、それらの合計が 4M バイトよりも大きい場合は、BackupMemory も設定する必要があります。

バックアップ対象のハードドライブパーティション上に十分な容量があることも確認するようにしてください。

NDB では反復可能読み取りがサポートされていないため、リストアプロセスで問題が発生する可能性があります。バックアップは「ホット」プロセスですが、バックアップからの MySQL Cluster のリストアは 100% 「ホット」プロセスであるとはかぎりません。その理由は、リストアプロセス中にトランザクションを実行すると、リストアされたデータから反復不可能読み取りが発生するためです。つまり、リストアの進行中は、データの状態に一貫性がありません。

18.5.4 MySQL Cluster での MySQL サーバーの使用法

mysqld は従来の MySQL サーバーのプロセスです。MySQL Cluster で使用するには、mysqld は https://dev.mysql.com/downloads/ から入手可能なプリコンパイル済みバイナリに含まれるため、NDB ストレージエンジンのサポートを使用して構築する必要があります。ソースから MySQL を構築する場合は、-DWITH_NDBCLUSTER=1 オプションを付けて CMake を呼び出して、NDB のサポートを含める必要があります。

ソースからの MySQL Cluster のコンパイルについての詳細は、セクション18.2.2.3「Linux でのソースからの MySQL Cluster のビルド」およびセクション18.2.3.2「Windows でのソースからの MySQL Cluster のコンパイルとインストール」を参照してください。

(このセクションで説明したものに加えて、MySQL Cluster に関連する mysqld のオプションおよび変数については、セクション18.3.4「MySQL Cluster 用の MySQL Server オプションおよび変数」を参照してください。)

Cluster サポートを使用して mysqld バイナリを構築した場合、NDBCLUSTER ストレージエンジンはまだデフォルトで無効になっています。次の 2 つの指定可能なオプションのいずれかを使用すると、このエンジンを有効にできます。

- mysqld を起動する際に、コマンド行で起動オプションとして --ndbcluster を使用します。
- my.cnf ファイルの [mysqld] セクションに ndbcluster を含む行を挿入します。

NDBCLUSTER ストレージエンジンを有効にしてサーバーが実行されていることを確認する簡単な方法は、MySQL Monitor (mysql) で SHOW ENGINES ステートメントを発行することです。NDBCLUSTER の行に、Support 値として値 YES が表示されます。この行に NO が表示される場合や、このような行が出力に表示されない場合は、NDB 対応の MySQL バージョンが動作していません。この行に DISABLED が表示される場合は、先ほど説明した 2 つの方法のいずれかを使用して有効にする必要があります。

クラスタ構成データを読み取るには、MySQL サーバーに少なくとも次の 3 つの情報が必要です。

- MySQL サーバー自身のクラスタノード ID
- 管理サーバー (MGM ノード) のホスト名または IP アドレス
- 管理サーバーに接続できる TCP/IP ポートの数

ノード ID は動的に割り当てられるため、明示的に指定する必要は厳密にはありません。

接続文字列を指定するには、mysqld の起動時にコマンド行で、または my.cnf 内に、mysqld パラメータ ndb-connectstring を使用します。接続文字列には、管理サーバーを検出できるホスト名または IP アドレス、および使用される TCP/IP ポートが含まれます。

次の例では、ndb_mgmd.mysql.com は管理サーバーが存在するホストであり、管理サーバーはポート 1186 でクラスタメッセージを待機します。

```
shell> mysqld --ndbcluster --ndb-connectstring=ndb_mgmd.mysql.com:1186
```

接続文字列の詳細は、セクション18.3.2.3「MySQL Cluster の接続文字列」を参照してください。

この情報が指定されると、MySQL サーバーがクラスタへの完全な参加者となります。(この方法で実行している mysqld プロセスは、多くの場合 SQL ノードと呼んでいます。)それは、すべてのクラスタデータノードおよびそ

のステータスを完全に認識し、すべてのデータノードへの接続を確立します。この場合、任意のデータノードをトランザクションコーディネータとして使用し、ノードデータを読み取って更新できます。

mysql クライアントで SHOW PROCESSLIST を使用すると、MySQL サーバーがクラスタに接続されているかどうかを確認できます。MySQL サーバーがクラスタに接続されていて、PROCESS 権限を持っている場合は、出力の最初の行が次のように表示されます。

```
mysql> SHOW PROCESSLIST \G
*************************** 1. row ***************************
     Id: 1
   User: system user
   Host:
     db:
Command: Daemon
   Time: 1
  State: Waiting for event from ndbcluster
   Info: NULL
```

> **重要**
>
> MySQL Cluster に参加するには、--ndbcluster と --ndb-connectstring の両方のオプション (my.cnf の同等のオプション) を付けて、mysqld プロセスを起動する必要があります。--ndbcluster オプションのみを付けて mysqld が起動された場合や、それがクラスタに接続できない場合は、ストレージエンジンに関係なく、NDB テーブルを操作することも、新しいテーブルを作成することもできません。後者の制約は、SQL ノードがクラスタに接続されていない間に、NDB テーブルと同じ名前を持つテーブルが作成されることを防ぐ目的の安全対策です。mysqld プロセスが MySQL Cluster に参加していない間に、別のストレージエンジンを使用してテーブルを作成する必要がある場合は、--ndbcluster オプションを付けずに、サーバーを起動する必要があります。

18.5.5 MySQL Cluster のローリング再起動の実行

このセクションでは、MySQL Cluster インストールのローリング再起動を実行する方法について説明します。このように呼ばれている理由は、クラスタ自体は操作可能のままになるように、各ノードを順番に停止および起動 (または再起動) することが伴うためです。これは多くの場合に、ローリングアップグレードまたはローリングダウングレードの一部として実行されます。ここでは、クラスタの高可用性が必須であり、クラスタ全体の停止時間は許可されません。一般に、ここで提供しているアップグレードについての情報は、ダウングレードにも適用されます。

ローリング再起動が望ましい理由は、いくつかあります。次のいくつかの段落で、これらについて説明します。

構成の変更
クラスタへの SQL ノードの追加や、構成パラメータの新しい値への設定などのクラスタ構成の変更を行うため。

MySQL Cluster ソフトウェアのアップグレードまたはダウングレード　クラスタを新しいバージョンの MySQL Cluster ソフトウェアにアップグレード (または、古いバージョンにダウングレード) するため。通常、これは「ローリングアップグレード」 (古いバージョンの MySQL Cluster に戻す場合は、「ローリングダウングレード」) と呼ばれます。

ノードホスト上での変更　1 つ以上の MySQL Cluster ノードプロセスが実行されているハードウェアまたはオペレーティングシステムで変更を行うため。

システムのリセット (クラスタのリセット)
望ましくない状態に達したために、クラスタをリセットするため。このような場合は、多くの場合に 1 つ以上のデータノードのデータおよびメタデータを再ロードすることが望ましいと考えられます。これは、次の 3 つの方法のいずれかで実行できます。

- --initial オプションを付けて各データノードプロセス (ndbd、場合によっては ndbmtd) を起動します。これにより、強制的にデータノードのファイルシステムがクリアされ、その他のデータノードから MySQL Cluster のデータおよびメタデータがすべて再ロードされます。
- 再起動を実行する前に、ndb_mgm クライアントの BACKUP コマンドを使用してバックアップを作成します。アップグレード後に、ndb_restore を使用して、1 つまたは複数のノードをリストアします。

  詳細は、セクション18.5.3「MySQL Cluster のオンラインバックアップ」およびセクション18.4.20「ndb_restore — MySQL Cluster バックアップのリストア」を参照してください。
- アップグレード前に、mysqldump を使用してバックアップを作成します。その後、LOAD DATA INFILE を使用してダンプをリストアします。

リソースのリカバリ
ほかの MySQL Cluster テーブルで再利用するために、連続した INSERT および DELETE 操作によって以前にテーブルに割り当てられたメモリーを解放するため。

ローリング再起動を実行するプロセスは、次のように一般化できます。

1. すべてのクラスタ管理ノード (ndb_mgmd プロセス) を停止し、再構成し、再起動します。(複数の管理サーバーでのローリング再起動を参照してください。)
2. 各クラスタデータノード (ndbd プロセス) を順番に停止し、再構成し、再起動します。
3. 各クラスタ SQL ノード (mysqld プロセス) を順番に停止し、再構成し、再起動します。

特定のローリングアップグレードの実装の詳細は、行われる変更によって異なります。次に、プロセスを詳しく説明します。

<table>
<tr><th colspan="5">再起動タイプ:</th></tr>
<tr><th colspan="2">Cluster
構成の変更</th><th>Cluster
ソフトウェアの
アップグレードまたは
ダウングレード</th><th>ノードホスト上
での変更</th><th colspan="2">Cluster のリセット</th></tr>
<tr><td colspan="5">A. 管理ノード（ndb_mgmd）プロセス...</td></tr>
<tr><td colspan="2">1. ndb_mgmd
プロセスをすべて停止

2. グローバル構成ファイルを変更

3. ndb_mgmd プロセスをすべて
起動</td><td>1. ndb_mgmd
プロセスを
すべて停止
2 各 ndb_mgmd
バイナリを新しい
バージョンに置換
3. ndb_mgmd
プロセスを起動</td><td>1. ndb_mgmd
プロセスを
すべて停止
2. ハードウェア、
オペレーティング
システム、または
両方に必要な変更
を加える
3. ndb_mgmd
プロセスを
すべて起動</td><td colspan="2">(または)

1. ndb_mgmd
プロセスを
すべて停止

2. ndb_mgmd
プロセスを
すべて起動

すべての
ndb_mgmd
プロセスを
再起動
(オプション)</td></tr>
<tr><td colspan="5">B. 各データノード（ndbd）プロセスについて...</td></tr>
<tr><td colspan="2">(または)

1. ndbd を停止

2. ndbd を起動</td><td>ndbd を再起動</td><td>1. ndbd を停止
2 ハードウェア、
オペレーティング
システム、
または両方に
必要な変更を
加える
3. ndbd
を起動</td><td colspan="2">(または)

1. ndbd を停止

2. ndbd を起動

ndbd を
再起動</td></tr>
<tr><td colspan="5">C. 各 SQL ノード（mysqld）プロセスについて...</td></tr>
<tr><td colspan="2">(または)

1.mysqld を停止

2.mysqld を起動</td><td>mysqld を
再起動</td><td>1. mysqld を
停止
2. mysqld
バイナリ
を新しい
バージョンに
置換
3. mysqld
を起動</td><td>1. mysqld を
停止
2 ハードウェア、
オペレーティング
システム、
または両方に
必要な変更を
加える
3. mysqld
を起動</td><td colspan="2">(または)

1.mysqld を停止

2.mysqld を起動

mysqld を
再起動</td></tr>
</table>

上記の図で、停止および起動のステップは、シェルコマンド (ほとんどの Unix システムでの kill など) または管理クライアントの STOP コマンドを使用して、プロセスを完全に停止してから、必要に応じて ndbd または ndb_mgmd 実行可能ファイルを呼び出して、システムシェルから再起動する必要があることを示しています。Windows では、NET START および NET STOP コマンド、または Windows Service Manager を使用して、Windows サービスとしてインストールされているノードを起動および停止することもできます (セクション 18.2.3.4「Windows サービスとしての MySQL Cluster プロセスのインストール」を参照してください)。

再起動は、ndb_mgm 管理クライアントの RESTART コマンドを使用してプロセスを再起動できることを示しています (セクション18.5.2「MySQL Cluster 管理クライアントのコマンド」を参照してください)。

MySQL Cluster では、ノードをアップグレードする際に柔軟性のある順序がサポートされています。MySQL Cluster をアップグレードする場合、管理ノード、データノード、またはその両方をアップグレードする前に、API ノード (SQL ノードを含む) をアップグレードできます。言い換えると、API ノードおよび SQL ノードを任意の順序でアップグレードすることが許可されています。これには、次のような条件が課せられます。

- この機能は、オンラインアップグレードの一部として使用することのみを目的としています。さまざまな MySQL Cluster リリースのノードバイナリを混在させることは、本番環境設定で継続的に長期間使用することを意図されておらず、サポートもされていません。
- データノードをアップグレードする前に、すべての管理ノードをアップグレードする必要があります。このことは、クラスタの API ノードおよび SQL ノードをアップグレードする順序に関係なく当てはまります。
- すべての管理ノードおよびデータノードがアップグレードされるまで、「新しい」バージョンに固有の機能は使用しないでください。

  このことは、NDB エンジンバージョンの変更に加えて、適用される可能性のある任意の MySQL サーバーバージョンの変更にも適用されます。したがって、アップグレードを計画する際には、これを考慮に入れることを忘れないでください。(一般に、このことは MySQL Cluster のオンラインアップグレードに当てはまります。)

Bug #48528 および Bug #49163 も参照してください。

複数の管理サーバーでのローリング再起動　　複数の管理ノードを持つ MySQL Cluster のローリング再起動を実行すると、ndb_mgmd によって、その他の管理ノードが実行されているかどうかがチェックされ、実行されている場合は、そのノードの構成データの使用が試みられることに注意してください。これが発生することを回避し、ndb_mgmd にその構成ファイルを強制的に再読み取りさせるには、次のステップを実行します。

1. MySQL Cluster のすべての ndb_mgmd プロセスを停止します。
2. すべての config.ini ファイルを更新します。
3. 必要に応じて、--reload、--initial、またはその両方のオプションを付けて単一の ndb_mgmd を起動します。
4. 最初の ndb_mgmd を --initial オプションを付けて起動した場合は、残りの ndb_mgmd プロセスもすべて --initial を使用して起動する必要があります。

   最初の ndb_mgmd を起動するときに使用されたその他のオプションに関係なく、最初の ndb_mgmd プロセスのあとに、--reload を使用して残りのプロセスを再起動しないようにしてください。
5. 通常どおりに、データノードおよび API ノードのローリング再起動を完了します。

ローリング再起動を実行してクラスタの構成を更新する際に、ndbinfo.nodes テーブルの config_generation カラムを使用して、新しい構成で正常に再起動されたデータノードを追跡できます。セクション18.5.10.18「ndbinfo nodes テーブル」を参照してください。

18.5.6 MySQL Cluster で生成されたイベントレポート

このセクションでは、MySQL Cluster で提供されるイベントログのタイプ、およびログに記録されるイベントのタイプについて説明します。

MySQL Cluster では、次の 2 つのタイプのイベントログが提供されています。

- すべてのクラスタノードで生成されたイベントが含まれるクラスタログ。クラスタログは、単一の場所でクラスタ全体に関するロギング情報を提供するため、ほとんどの使用目的で推奨される口グです。

  デフォルトでは、クラスタログは ndb_mgm バイナリが存在する同じディレクトリ内の ndb_node_id_cluster.log (ここで、node_id は管理サーバーのノード ID です) というファイルに保存されます。

  クラスタのロギング情報は、DataDir および LogDestination 構成パラメータに設定された値によって指定されたファイルに保存することに加えて、またはその代わりに、stdout または syslog 機能に送信することもできま

す。これらのパラメータについての詳細は、セクション18.3.2.5「MySQL Cluster 管理サーバーの定義」を参照してください。

- ノードログは、各ノードにローカルです。

  ノードイベントのロギングで生成された出力は、ノードの DataDir 内の ndb_node_id_out.log ファイル (ここで、node_id はノードのノード ID です) に書き込まれます。ノードイベントログは、管理ノードとデータノードの両方で生成されます。

  ノードログは、アプリケーションの開発時またはアプリケーションコードのデバッグに使用されることのみを目的としています。

両方のタイプのイベントログを設定すると、イベントのさまざまなサブセットのログを記録できます。

レポート可能な各イベントは、次の 3 つの基準に従って区別できます。

- カテゴリ: これは、STARTUP、SHUTDOWN、STATISTICS、CHECKPOINT、NODERESTART、CONNECTION、ERROR、または INFO 値のいずれかを指定できます。
- 優先度: これは、1 から 15 までの数値で表されます。ここで、1 は「もっとも重要」、15 は「もっとも重要でない」ことを示します。
- 重大度レベル: これは、ALERT、CRITICAL、ERROR、WARNING、INFO、または DEBUG 値のいずれかを指定できます。

クラスタログとノードログのどちらもこれらのプロパティーでフィルタ処理できます。

クラスタログで使用される形式は、次に示すとおりです。

```
2007-01-26 19:35:55 [MgmSrvr] INFO     -- Node 1: Data usage is 2%(60 32K pages of total 2560)
2007-01-26 19:35:55 [MgmSrvr] INFO     -- Node 1: Index usage is 1%(24 8K pages of total 2336)
2007-01-26 19:35:55 [MgmSrvr] INFO     -- Node 1: Resource 0 min: 0 max: 639 curr: 0
2007-01-26 19:35:55 [MgmSrvr] INFO     -- Node 2: Data usage is 2%(76 32K pages of total 2560)
2007-01-26 19:35:55 [MgmSrvr] INFO     -- Node 2: Index usage is 1%(24 8K pages of total 2336)
2007-01-26 19:35:55 [MgmSrvr] INFO     -- Node 2: Resource 0 min: 0 max: 639 curr: 0
2007-01-26 19:35:55 [MgmSrvr] INFO     -- Node 3: Data usage is 2%(58 32K pages of total 2560)
2007-01-26 19:35:55 [MgmSrvr] INFO     -- Node 3: Index usage is 1%(25 8K pages of total 2336)
2007-01-26 19:35:55 [MgmSrvr] INFO     -- Node 3: Resource 0 min: 0 max: 639 curr: 0
2007-01-26 19:35:55 [MgmSrvr] INFO     -- Node 4: Data usage is 2%(74 32K pages of total 2560)
2007-01-26 19:35:55 [MgmSrvr] INFO     -- Node 4: Index usage is 1%(25 8K pages of total 2336)
2007-01-26 19:35:55 [MgmSrvr] INFO     -- Node 4: Resource 0 min: 0 max: 639 curr: 0
2007-01-26 19:39:42 [MgmSrvr] INFO     -- Node 4: Node 9 Connected
2007-01-26 19:39:42 [MgmSrvr] INFO     -- Node 1: Node 9 Connected
2007-01-26 19:39:42 [MgmSrvr] INFO     -- Node 1: Node 9: API 5.6.22-ndb-7.4.4
2007-01-26 19:39:42 [MgmSrvr] INFO     -- Node 2: Node 9 Connected
2007-01-26 19:39:42 [MgmSrvr] INFO     -- Node 2: Node 9: API 5.6.22-ndb-7.4.4
2007-01-26 19:39:42 [MgmSrvr] INFO     -- Node 3: Node 9 Connected
2007-01-26 19:39:42 [MgmSrvr] INFO     -- Node 3: Node 9: API 5.6.22-ndb-7.4.4
2007-01-26 19:39:42 [MgmSrvr] INFO     -- Node 4: Node 9: API 5.6.22-ndb-7.4.4
2007-01-26 19:59:22 [MgmSrvr] ALERT    -- Node 2: Node 7 Disconnected
2007-01-26 19:59:22 [MgmSrvr] ALERT    -- Node 2: Node 7 Disconnected
```

クラスタログの各行には、次の情報が含まれます。

- YYYY-MM-DD HH:MM:SS 形式のタイムスタンプ。
- ロギングを実行しているノードのタイプ。クラスタログでは、これは常に [MgmSrvr] です。
- イベントの重大度。
- イベントをレポートするノードの ID。
- イベントの説明。ログに表示されるもっとも一般的なイベントタイプは、クラスタ内のさまざまなノード間およびチェックポイントの発生時の接続と切断です。場合によっては、説明にステータス情報が含まれることがあります。

18.5.6.1 MySQL Cluster のロギング管理コマンド

ndb_mgm では、クラスタログに関連するいくつかの管理コマンドがサポートされています。次のリストでは、node_id はデータベースノード ID、またはクラスタのデータノードのすべてにコマンドを適用すべきことを示すキーワード ALL を示します。

- CLUSTERLOG ON

  クラスタログをオンにします。

- CLUSTERLOG OFF

  クラスタログをオフにします。

- CLUSTERLOG INFO

  クラスタログの設定に関する情報を提供します。

- node_id CLUSTERLOG category=threshold

  threshold 以下の優先度を持つ category のイベントをクラスタログに記録します。

- CLUSTERLOG FILTER severity_level

  指定された severity_level のイベントのクラスタロギングを切り換えます。

次の表では、クラスタログカテゴリのしきい値の (すべてのデータノードの) デフォルト設定について説明します。イベントの優先度が優先度のしきい値以下である場合は、そのイベントがクラスタログにレポートされます。

イベントはデータノードごとにレポートされ、しきい値はノードごとに異なる値に設定できます。

カテゴリ	デフォルトのしきい値 (すべてのデータノード)
STARTUP	7
SHUTDOWN	7
STATISTICS	7
CHECKPOINT	7
NODERESTART	7
CONNECTION	7
ERROR	15
INFO	7

STATISTICS カテゴリは、役に立つデータを大量に提供できます。詳細は、セクション18.5.6.3「MySQL Cluster 管理クライアントでの CLUSTERLOG STATISTICS の使用」を参照してください。

しきい値は、各カテゴリ内のイベントをフィルタ処理するために使用されます。たとえば、優先度が 3 の STARTUP イベントのログは、STARTUP のしきい値が 3 以上に設定されていなければ記録されません。しきい値が 3 の場合は、優先度が 3 以下のイベントのみが送信されます。

次の表は、イベントの重大度レベルを示しています。

> **注記**
>
> これらは、使用されることもマップされることもない LOG_EMERG と LOG_NOTICE を除く、Unix の syslog レベルに対応します。

重大度レベル値	重大度	説明
1	ALERT	すぐに修正するべき状況 (システムデータベースの破損など)
2	CRITICAL	クリティカルな状況 (デバイスエラーやリソース不足など)
3	ERROR	修正するべき状況 (構成エラーなど)
4	WARNING	エラーではないが、特別な処理が必要な可能性がある状況
5	INFO	情報メッセージ
6	DEBUG	NDBCLUSTER の開発で使用されるデバッグメッセージ

イベント重大度レベルは、CLUSTERLOG FILTER を使用することでオンとオフを切り替えることができます (上記を参照してください)。重大度レベルをオンにすると、優先度がカテゴリしきい値以下であるすべてのイベントがログに記録されます。重大度レベルをオフにすると、その重大度レベルに属するイベントはログに記録されません。

> **重要**
>
> クラスタログレベルは、ndb_mgmd ごとに、スクライバ単位で設定されます。つまり、複数の管理サーバーを持つ MySQL Cluster では、1 台の管理サーバーに接続された ndb_mgm のインスタンスで CLUSTERLOG コマンドを使用すると、その管理サーバーで生成されたログのみが影響を受け、その他のサーバーで生成されたログは影響を受けません。また、管理サーバーの 1 台が再起動されると、その管理サーバーで生成されたログのみが、再起動によって発生したログレベルのリセットによる影響を受けます。

18.5.6.2 MySQL Cluster ログイベント

イベントログでレポートされるイベントレポートの形式は、次のとおりです。

```
datetime [string] severity -- message
```

例:

```
09:19:30 2005-07-24 [NDB] INFO -- Node 4 Start phase 4 completed
```

このセクションでは、レポート可能なすべてのイベントについて、カテゴリおよび各カテゴリ内の重大度レベルの順に説明します。

イベントの説明では、GCP と LCP はそれぞれ「グローバルチェックポイント」および「ローカルチェックポイント」を表します。

CONNECTION イベント

これらのイベントは、クラスタノード間の接続に関連するものです。

イベント	優先度	重大度レベル	説明
Connected	8	INFO	データノードが接続されました
Disconnected	8	ALERT	データノードが切断されました
CommunicationClosed	8	INFO	SQL ノードまたはデータノードの接続が閉じられました
CommunicationOpened	8	INFO	SQL ノードまたはデータノードの接続が開かれました
ConnectedApiVersion	8	INFO	API バージョンを使用した接続

CHECKPOINT イベント

次に示すロギングメッセージは、チェックポイントに関連するものです。

イベント	優先度	重大度レベル	説明
GlobalCheckpointStarted	9	INFO	GCP の開始: Redo ログがディスクに書き込まれます
GlobalCheckpointCompleted	10	INFO	GCP が終了しました
LocalCheckpointStarted	7	INFO	LCP の開始: データがディスクに書き込まれます
LocalCheckpointCompleted	7	INFO	LCP が正常に完了しました
LCPStoppedInCalcKeepGci	0	ALERT	LCP が停止しました
LCPFragmentCompleted	11	INFO	フラグメント上の LCP が完了しました
UndoLogBlocked	7	INFO	Undo ロギングがブロックされました。バッファーのオーバーフローに近付いています
RedoStatus	7	INFO	Redo ステータス

STARTUP イベント

ノードまたはクラスタの起動や、その成功または失敗に対応して、次のイベントが生成されます。それらは、起動プロセスの進行状況に関する情報 (ロギングアクティビティーに関する情報を含む) も提供します。

イベント	優先度	重大度レベル	説明
NDBStartStarted	1	INFO	データノードの起動フェーズが開始されました (すべてのノードが起動します)
NDBStartCompleted	1	INFO	起動フェーズが完了しました、すべてのデータノード
STTORRYRecieved	15	INFO	再起動の完了後にブロックを受信しました
StartPhaseCompleted	4	INFO	データノードの起動フェーズ X が完了しました
CM_REGCONF	3	INFO	ノードが正常にクラスタに追加されました。ノード、管理ノード、および動的 ID が表示されます
CM_REGREF	8	INFO	ノードはクラスタへの追加を拒否されました。構成が間違っている、通信を確立できないなどの問題が原因で、クラスタに追加できません
FIND_NEIGHBOURS	8	INFO	隣接したデータノードを表示します
NDBStopStarted	1	INFO	データノードのシャットダウンが開始されました
NDBStopCompleted	1	INFO	データノードのシャットダウンが完了しました
NDBStopForced	1	ALERT	データノードが強制的にシャットダウンされました
NDBStopAborted	1	INFO	データノードを正常にシャットダウンできませんでした
StartREDOLog	4	INFO	新しい Redo ログが開始されました。GCI は X を保持し、最新のリストア可能な GCI は Y です
StartLog	10	INFO	新しいログが開始されました。ログ部分は X、開始 MB は Y、終了 MB は Z です
UNDORecordsExecuted	15	INFO	Undo レコードが実行されました
StartReport	4	INFO	レポートが開始されました
LogFileInitStatus	7	INFO	ログファイルの初期化ステータス
LogFileInitCompStatus	7	INFO	ログファイルの完了ステータス
StartReadLCP	10	INFO	ローカルチェックポイントの読み取りを開始します
ReadLCPComplete	10	INFO	ローカルチェックポイントの読み取りが完了しました
RunRedo	8	INFO	Redo ログが実行されています
RebuildIndex	10	INFO	インデックスを再構築しています

NODERESTART イベント

次のイベントは、ノードを再起動するときに生成され、ノードの再起動プロセスの成功または失敗に関連するものです。

イベント	優先度	重大度レベル	説明
NR_CopyDict	7	INFO	ディレクトリ情報のコピーが完了しました
NR_CopyDistr	7	INFO	ディストリビューション情報のコピーが完了しました
NR_CopyFragsStarted	7	INFO	フラグメントのコピーを開始しています
NR_CopyFragDone	10	INFO	フラグメントのコピーが完了しました
NR_CopyFragsCompleted	7	INFO	すべてのフラグメントのコピーが完了しました
NodeFailCompleted	8	ALERT	ノードの失敗フェーズが完了しました
NODE_FAILREP	8	ALERT	ノードが失敗したことをレポートします

イベント	優先度	重大度レベル	説明
ArbitState	6	INFO	アービトレータが見つかったかどうかをレポートします。アービトレータの検索時の可能性のある結果は、次に示す 7 つです。 • 管理サーバーがアービトレーションスレッドを再起動します [状態 =X] • アービトレータノード X を準備します [チケット =Y] • アービトレータノード X を受信します [チケット =Y] • アービトレータノード X を起動しました [チケット =Y] • アービトレータノード X が失われました - プロセスの失敗 [状態 =Y] • アービトレータノード X が失われました - プロセスの終了 [状態 =Y] • アービトレータノード X が失われました <エラーメッセージ> [状態 =Y]
ArbitResult	2	ALERT	アービトレータの結果をレポートします。アービトレータの試行の可能性のある結果は、次に示した 8 つです。 • アービトレーションチェックに失敗しました: 1/2 未満のノードが残っています • アービトレーションチェックに成功しました: ノードグループの大部分 • アービトレーションチェックに失敗しました: ノードグループが見つかりません • ネットワークのパーティション化: アービトレーションが必要です • アービトレーションに成功しました: ノード X からの肯定的応答 • アービトレーションに失敗しました: ノード X からの否定的応答 • ネットワークのパーティション化: 使用可能なアービトレータがありません • ネットワークのパーティション化: アービトレータが構成されていません
GCP_TakeoverStarted	7	INFO	GCP テイクオーバーが開始されました
GCP_TakeoverCompleted	7	INFO	GCP テイクオーバーが完了しました
LCP_TakeoverStarted	7	INFO	LCP テイクオーバーが開始されました
LCP_TakeoverCompleted	7	INFO	LCP テイクオーバーが完了しました (状態 = X)
ConnectCheckStarted	6	INFO	接続チェックが開始されました
ConnectCheckCompleted	6	INFO	接続チェックが完了しました
NodeFailRejected	6	ALERT	ノードの失敗フェーズに失敗しました

STATISTICS イベント

次のイベントは、統計的な特性を持っています。それらは、トランザクションやその他の操作の回数、各ノードによって送受信されたデータの量、メモリーの使用状況などの情報を提供します。

イベント	優先度	重大度レベル	説明
TransReportCounters	8	INFO	トランザクション、コミット、読み取り、単純な読み取り、書き込み、同時操作、属性情報、中止の数などのトランザクション統計をレポートします
OperationReportCounters	8	INFO	操作数
TableCreated	7	INFO	作成されたテーブルの数をレポートします
JobStatistic	9	INFO	内部ジョブスケジューリングの平均の統計
ThreadConfigLoop	9	INFO	スレッド構成ループの数
SendBytesStatistic	9	INFO	ノード X に送信された平均バイト数
ReceiveBytesStatistic	9	INFO	ノード X から受信された平均バイト数
MemoryUsage	5	INFO	データおよびインデックスメモリー使用状況 (80%、90%、および 100%)
MTSignalStatistics	9	INFO	マルチスレッドシグナル

SCHEMA イベント

これらのイベントは、MySQL Cluster スキーマ操作に関連するものです。

イベント	優先度	重大度	説明
CreateSchemaObject	8	INFO	スキーマオブジェクトが作成されました
AlterSchemaObject	8	INFO	スキーマオブジェクトが更新されました
DropSchemaObject	8	INFO	スキーマオブジェクトが削除されました

ERROR イベント

これらのイベントは、クラスタのエラーおよび警告に関連するものです。一般に、これらが 1 つ以上存在することは、重大な誤動作や障害が発生したことを示しています。

イベント	優先度	重大度	説明
TransporterError	2	ERROR	トランスポータエラー
TransporterWarning	8	WARNING	トランスポータ警告
MissedHeartbeat	8	WARNING	ノード X でハートビート番号 Y が失われました
DeadDueToHeartbeat	8	ALERT	ハートビートが失われたため、ノード X が「停止」を宣言しました
WarningEvent	2	WARNING	一般警告イベント
SubscriptionStatus	4	WARNING	サブスクリプションのステータスの変更

INFO イベント

これらのイベントは、クラスタの状態およびロギングやハートビート伝送などのクラスタのメンテナンスに関連するアクティビティーに関する一般情報を提供します。

イベント	優先度	重大度	説明
SentHeartbeat	12	INFO	ハートビートを送信しました
CreateLogBytes	11	INFO	ログの作成: ログ部分、ログファイル、MB 単位のサイズ
InfoEvent	2	INFO	一般情報イベント
EventBufferStatus	7	INFO	イベントバッファーステータス

SINGLEUSER イベント

これらのイベントは、シングルユーザーモードの開始と終了に関連するものです。

イベント	優先度	重大度	説明
SingleUser	7	INFO	シングルユーザーモードを開始または終了しています

BACKUP イベント

これらのイベントは、作成またはリストアされるバックアップに関する情報を提供します。

イベント	優先度	重大度	説明
BackupStarted	7	INFO	バックアップが開始されました
BackupStatus	7	INFO	バックアップのステータス
BackupCompleted	7	INFO	バックアップが完了しました
BackupFailedToStart	7	ALERT	バックアップの開始に失敗しました
BackupAborted	7	ALERT	バックアップがユーザーによって中止されました
RestoreStarted	7	INFO	バックアップからのリストアが開始されました
RestoreMetaData	7	INFO	メタデータをリストアしています
RestoreData	7	INFO	データをリストアしています
RestoreLog	7	INFO	ログファイルをリストアしています
RestoreCompleted	7	INFO	バックアップからのリストアが完了しました
SavedEvent	7	INFO	イベントが保存されました

18.5.6.3 MySQL Cluster 管理クライアントでの CLUSTERLOG STATISTICS の使用

NDB 管理クライアントの CLUSTERLOG STATISTICS コマンドは、その出力にいくつかの役に立つ統計を表示できます。クラスタの状態に関する情報を提供するカウンタは、トランザクションコーディネータ (TC) およびローカルクエリーハンドラ (LQH) によって 5 秒のレポート間隔で更新され、クラスタログに書き込まれます。

トランザクションコーディネータの統計　各トランザクションには、1 つのトランザクションコーディネータがあり、それは次の方法のいずれかによって選択されます。

- ラウンドロビン方式で
- 通信の近さによって
- (MySQL Cluster NDB 6.3.4 以降:) トランザクションの開始時にデータ配置のヒントを指定することによって

> **注記**
>
> ndb_optimized_node_selection システム変数を使用すると、特定の SQL ノードから起動されたトランザクションに使用される TC の選択方法を判断できます。

同じトランザクション内の操作ではすべて、同じトランザクションコーディネータが使用され、次のような統計がレポートされます。

- Trans count　これは、この TC をトランザクションコーディネータとして使用して、最後の期間で開始されたトランザクション数です。これらのトランザクションのいずれかは、レポート期間の最後でコミットされた、中止された、または未コミットのままの状態になっている可能性があります。

  > **注記**
  >
  > トランザクションは TC 間で移行しません。

- Commit count　これは、最後のレポート期間でコミットされた、この TC をトランザクションコーディネータとして使用したトランザクション数です。このレポート期間でコミットされた一部のトランザクションは、前のレポート期間で開始された可能性があるため、Commit count が Trans count よりも大きくなる可能性もあります。
- Read count　これは、最後のレポート期間で開始された、この TC をトランザクションコーディネータとして使用した主キーの読み取り操作 (単純な読み取りを含む) 数です。このカウントには、一意のインデックス操作の一部として実行される読み取りも含まれます。一意のインデックス読み取り操作では、2 つの主キー読み取り操作 (非表示の一意のインデックステーブルに対して 1 つ、および読み取りが行われるテーブルに対して 1 つ) が発生します。

- Simple read count　これは、最後のレポート期間で開始された、この TC をトランザクションコーディネータとして使用した単純な読み取り操作数です。これは Read count のサブセットです。Simple read count の値は Read count とは異なる時点でインクリメントされるため、Read count よりもわずかに遅れる可能性があります。そのため、その時間内に行われたすべての読み取りが実際に単純な読み取りであった場合でも、特定のレポート期間で Simple read count が Read count と等しくなくなることがあり得ます。

- Write count　これは、最後のレポート期間で開始された、この TC をトランザクションコーディネータとして使用した主キーの書き込み操作数です。これには、一意のインデックス操作の一部として実行される書き込みに加えて、すべての挿入、更新、書き込み、および削除も含まれます。

  > **注記**
  >
  > 一意のインデックスの更新操作では、インデックステーブルおよびベーステーブルで、複数の PK 読み取りおよび書き込み操作が発生する可能性があります。

- AttrInfoCount　これは、この TC をトランザクションコーディネータとして使用した主キー操作の最後のレポート期間で受信された 32 ビットのデータ語の数です。読み取りの場合、これはリクエストされたカラムの数に比例します。挿入および更新の場合、これは書き込まれたカラムの数、およびそれらのデータのサイズに比例します。削除操作の場合、これは通常ゼロです。

  一意のインデックス操作では、複数の PK 操作が発生するため、このカウントが増加します。ただし、ここでは、PK 操作自体を記述するために送信されたデータ語、および送信されたキー情報はカウントされません。スキャン用に読み取られるカラムを記述するため、または ScanFilters を記述するために送信された属性情報も、AttrInfoCount ではカウントされません。

- Concurrent Operations　これは、最後のレポート期間中に開始されたが完了しなかった、この TC をトランザクションコーディネータとして使用した主キーまたはスキャン操作数です。このカウンタは、操作が開始されるとインクリメントされ、操作が完了するとデクリメントされます。これは、トランザクションのコミット後に行われます。このカウンタは失敗した操作だけでなく、ダーティー読み取りおよび書き込みでもデクリメントされます。

  Concurrent Operations に指定可能な最大値は、TC ブロックでサポートできる操作の最大数です。現在、これは (2 * MaxNoOfConcurrentOperations) + 16 + MaxNoOfConcurrentTransactions です。(これらの構成パラメータについての詳細は、セクション18.3.2.6「MySQL Cluster データノードの定義」の「トランザクションパラメータ」セクションを参照してください。)

- Abort count　これは、最後のレポート期間中に中止された、この TC をトランザクションコーディネータとして使用したトランザクション数です。最後のレポート期間で中止された一部のトランザクションは、前のレポート期間で開始された可能性があるため、Abort count が Trans count よりも大きくなる可能性もあります。

- Scans　これは、最後のレポート期間中に開始された、この TC をトランザクションコーディネータとして使用したテーブルスキャン数です。これには、範囲スキャン (つまり、順序付きインデックススキャン) は含まれません。

- Range scans　これは、最後のレポート期間で開始された、この TC をトランザクションコーディネータとして使用した順序付きインデックススキャン数です。

ローカルクエリーハンドラの統計 (操作)　ローカルクエリーハンドラブロックごとに、1 つのクラスタイベント (つまり、データノードプロセスごとに 1 つずつ) があります。操作は、それらが操作しているデータが存在する LQH に記録されます。

> **注記**
>
> 単一のトランザクションが複数の LQH ブロックに格納されたデータを操作する場合もあります。

Operations 統計には、最後のレポート期間で、この LQH ブロックによって実行されたローカル操作数が表示され、すべてのタイプの読み取りおよび書き込み操作 (挿入、更新、書き込み、および削除の操作) が含まれます。これには、書き込みをレプリケートするために使用される操作も含まれます。たとえば、2 つのレプリカクラスタでは、プライマリレプリカへの書き込みがプライマリ LQH に記録され、バックアップへの書き込みがバックアップ LQH に記録されます。一意のキー操作では、複数のローカル操作が発生する可能性があります。ただし、これには、テーブルスキャンまたは順序付きインデックススキャンの結果として発生するローカル操作は含まれず、カウントもされません。

プロセススケジューラの統計　トランザクションコーディネータおよびローカルクエリーハンドラによってレポートされる統計に加えて、各 ndbd プロセスには、MySQL Cluster のパフォーマンスに関する役に立つメトリ

クスも提供するスケジューラが含まれています。このスケジューラは、無限ループ時に実行されます。各ループ中に、スケジューラは次のタスクを実行します。

1. ソケットからジョブバッファーに受信メッセージを読み込みます。
2. 時間指定のメッセージが実行されたかどうかをチェックします。実行された場合は、これらのメッセージもジョブバッファーに配置します。
3. ジョブバッファー内のメッセージを (ループ内で) 実行します。
4. ジョブバッファー内のメッセージを実行することで生成された配信されるメッセージを送信します。
5. 新しい受信メッセージを待機します。

プロセススケジューラの統計には、次の情報が含まれています。

- Mean Loop Counter　これは、上記のリストの 3 番目のステップで実行されたループの数です。TCP/IP バッファーの使用率が改善されると、この統計のサイズが増加します。これを使用すると、新しいデータノードプロセスを追加したときにパフォーマンスの変化をモニターできます。
- Mean send size と Mean receive size　これらの統計を使用すると、ノード間の書き込みと読み取りのそれぞれの効率性を測定できます。値はバイト単位で指定されます。値が大きいほど、送受信される 1 バイト当たりのコストが小さくなることを意味します。最大値は 64K です。

NDB 管理クライアントで次のコマンドを使用すると、クラスタログの統計をすべて記録できます。

```
ndb_mgm> ALL CLUSTERLOG STATISTICS=15
```

> **注記**
>
> STATISTICS のしきい値を 15 に設定すると、クラスタログが非常に詳細になり、MySQL Cluster 内のクラスタノードの数およびアクティビティーの量に正比例して、サイズが急速に増大します。

ロギングおよびレポートに関する MySQL Cluster 管理クライアントコマンドについての詳細は、セクション 18.5.6.1「MySQL Cluster の ロギング管理コマンド」を参照してください。

18.5.7 MySQL Cluster ログメッセージ

このセクションでは、さまざまなクラスタログイベントに対応してクラスタに書き込まれるメッセージに関する情報を提供します。NDB トランスポータのエラーについて、より具体的な追加情報も提供します。

18.5.7.1 MySQL Cluster: クラスタログ内のメッセージ

次の表には、もっとも一般的な NDB クラスタログメッセージを一覧表示します。クラスタログ、ログイベント、およびイベントタイプについては、セクション18.5.6「MySQL Cluster で生成されたイベントレポート」を参照してください。これらのログメッセージは、MGM API のログイベントタイプにも対応します。Cluster API の開発者が関心を持つような関連情報については、The Ndb_logevent_type Typeを参照してください。

ログメッセージ　Node mgm_node_id: Node data_node_id Connected 説明　ノード ID node_id を持つデータノードが管理サーバー (ノード mgm_node_id) に接続しました。	イベント名　Connected イベントタイプ　Connection 優先度　8 重大度　INFO
ログメッセージ　Node mgm_node_id: Node data_node_id Disconnected 説明　ノード ID data_node_id を持つデータノードが管理サーバー (ノード mgm_node_id) から切断しました。	イベント名　Disconnected イベントタイプ　Connection 優先度　8 重大度　ALERT
ログメッセージ　Node data_node_id: Communication to Node api_node_id closed 説明　ノード ID api_node_id を持つ API ノードまたは SQL ノードがデータノード data_node_id と通信しなくなりました。	イベント名　CommunicationClosed イベントタイプ　Connection 優先度　8 重大度　INFO

ログメッセージ　Node data_node_id: Communication to Node api_node_id opened 説明　ノード ID api_node_id を持つ API ノードまたは SQL ノードがデータノード data_node_id と通信しています。	イベント名　CommunicationOpened イベントタイプ　Connection 優先度　8 重大度　INFO
ログメッセージ　Node mgm_node_id: Node api_node_id: API version 説明　ノード ID api_node_id を持つ API ノードが NDB API バージョン version (一般に、MySQL バージョン番号と同じ) を使用して、管理ノード mgm_node_id に接続しました。	イベント名　ConnectedApiVersion イベントタイプ　Connection 優先度　8 重大度　INFO
ログメッセージ　Node node_id: Global checkpoint gci started 説明　ID gci を持つグローバルチェックポイントが開始されました。ノード node_id は、このグローバルチェックポイントを担当するマスターです。	イベント名　GlobalCheckpointStarted イベントタイプ　Checkpoint 優先度　9 重大度　INFO
ログメッセージ　Node node_id: Global checkpoint gci completed 説明　ID gci を持つグローバルチェックポイントが完了しました。ノード node_id は、このグローバルチェックポイントを担当したマスターでした。	イベント名　GlobalCheckpointCompleted イベントタイプ　Checkpoint 優先度　10 重大度　INFO
ログメッセージ　Node node_id: Local checkpoint lcp started. Keep GCI = current_gci oldest restorable GCI = old_gci 説明　シーケンス ID lcp を持つローカルチェックポイントがノード node_id で開始されました。使用可能な最新の GCI はインデックス current_gci を持ち、クラスタをリストアできるもっとも古い GCI はインデックス old_gci を持っています。	イベント名　LocalCheckpointStarted イベントタイプ　Checkpoint 優先度　7 重大度　INFO
ログメッセージ　Node node_id: Local checkpoint lcp completed 説明　ノード node_id 上でシーケンス ID lcp を持つローカルチェックポイントが完了しました。	イベント名　LocalCheckpointCompleted イベントタイプ　Checkpoint 優先度　8 重大度　INFO
ログメッセージ　Node node_id: Local Checkpoint stopped in CALCULATED_KEEP_GCI 説明　このノードは、使用可能な最新の GCI を特定できませんでした。	イベント名　LCPStoppedInCalcKeepGci イベントタイプ　Checkpoint 優先度　0 重大度　ALERT
ログメッセージ　Node node_id: Table ID = table_id, fragment ID = fragment_id has completed LCP on Node node_id maxGciStarted: started_gci maxGciCompleted: completed_gci 説明　ノード node_id 上のディスクに対して、テーブルフラグメントのチェックポイントが実行されました。進行中の GCI はインデックス started_gci を持ち、最近完了した GCI はインデックス completed_gci を持っています。	イベント名　LCPFragmentCompleted イベントタイプ　Checkpoint 優先度　11 重大度　INFO
ログメッセージ　Node node_id: ACC Blocked num_1 and TUP Blocked num_2 times last second 説明　ログバッファーがオーバーフローしそうなため、Undo ロギングがブロックされています。	イベント名　UndoLogBlocked イベントタイプ　Checkpoint 優先度　7 重大度　INFO

ログメッセージ　Node node_id: Start initiated version 説明　NDB バージョン version を実行しているデータノード node_id が起動プロセスを開始しています。	イベント名　NDBStartStarted イベントタイプ　StartUp 優先度　1 重大度　INFO
ログメッセージ　Node node_id: Started version 説明　NDB バージョン version を実行しているデータノード node_id が正常に起動されました。	イベント名　NDBStartCompleted イベントタイプ　StartUp 優先度　1 重大度　INFO
ログメッセージ　Node node_id: STTORRY received after restart finished 説明　ノードがクラスタの再起動が完了したことを示すシグナルを受信しました。	イベント名　STTORRYRecieved イベントタイプ　StartUp 優先度　15 重大度　INFO
ログメッセージ　Node node_id: Start phase phase completed (type) 説明　ノードが type 起動の起動フェーズ phase を完了しました。起動フェーズのリストについては、セクション 18.5.1「MySQL Cluster の起動フェーズのサマリー」を参照してください (type は initial、system、node、initial node、または <Unknown> のいずれかです)。	イベント名　StartPhaseCompleted イベントタイプ　StartUp 優先度　4 重大度　INFO
ログメッセージ　Node node_id: CM_REGCONF president = president_id, own Node = own_id, our dynamic id = dynamic_id 説明　ノード president_id が「プレジデント」として選択されました。own_id および dynamic_id は、常にレポートノードの ID (node_id) と同じであるべきです。	イベント名　CM_REGCONF イベントタイプ　StartUp 優先度　3 重大度　INFO
ログメッセージ　Node node_id: CM_REGREF from Node president_id to our Node node_id. Cause = cause 説明　レポートノード (ID node_id) がプレジデントとしてノード president_id を受け入れることができませんでした。問題の cause は、Busy、Election with wait = false、Not president、Election without selecting new candidate、または No such cause のいずれかとして指定されます。	イベント名　CM_REGREF イベントタイプ　StartUp 優先度　8 重大度　INFO
ログメッセージ　Node node_id: We are Node own_id with dynamic ID dynamic_id, our left neighbor is Node id_1, our right is Node id_2 説明　ノードがクラスタ内の隣接するノード (ノード id_1 とノード id_2) を検出しました。node_id、own_id、および dynamic_id は、常に同じであるべきです。そうでない場合は、クラスタノード構成の重大な誤りを示しています。	イベント名　FIND_NEIGHBOURS イベントタイプ　StartUp 優先度　8 重大度　INFO
ログメッセージ　Node node_id: type shutdown initiated 説明　ノードがシャットダウンシグナルを受信しました。シャットダウンの type は Cluster または Node です。	イベント名　NDBStopStarted イベントタイプ　StartUp 優先度　1 重大度　INFO
ログメッセージ　Node node_id: Node shutdown completed [, action] [Initiated by signal signal.] 説明　ノードがシャットダウンされました。このレポートには、action が含まれる場合があります。これが存在する場合	イベント名　NDBStopCompleted イベントタイプ　StartUp 優先度　1

は、restarting、no start、または initial のいずれかです。また、レポートには NDB プロトコル signal への参照が含まれる場合もあります。可能性のあるシグナルについては、Operations and Signalsを参照してください。	重大度　INFO
ログメッセージ　Node node_id: Forced node shutdown completed [, action]. [Occured during startphase start_phase.] [Initiated by signal.] [Caused by error error_code: 'error_message(error_classification). error_status'. [(extra info extra_code)]] 説明　ノードが強制的にシャットダウンされました。あとで実行された action (restarting、no start、または initial のいずれか) があれば、それもレポートされます。ノードの起動時にシャットダウンが発生した場合、レポートに、ノードの障害が発生した start_phase が含まれます。これが、ノードに signal が送信された結果であった場合は、その情報も提供されます (詳細は、Operations and Signalsを参照してください)。障害の原因が既知のエラーである場合は、それも含まれます。NDB のエラーメッセージおよび分類についての詳細は、NDB Cluster API Errorsを参照してください。	イベント名　NDBStopForced イベントタイプ　StartUp 優先度　1 重大度　ALERT
ログメッセージ　Node node_id: Node shutdown aborted 説明　ノードのシャットダウンプロセスがユーザーによって中止されました。	イベント名　NDBStopAborted イベントタイプ　StartUp 優先度　1 重大度　INFO
ログメッセージ　Node node_id: StartLog: [GCI Keep: keep_pos LastCompleted: last_pos NewestRestorable: restore_pos] 説明　これは、ノード起動時に参照されるグローバルチェックポイントをレポートします。keep_pos よりも前の Redo ログは削除されます。last_pos は、データノードが最後に関与したグローバルチェックポイントです。restore_pos は、すべてのデータノードをリストアするために実際に使用されるグローバルチェックポイントです。	イベント名　StartREDOLog イベントタイプ　StartUp 優先度　4 重大度　INFO
ログメッセージ　startup_message [Listed separately; see below.] 説明　さまざまな環境下でログに記録される可能性のある多くの起動メッセージがあります。	イベント名　StartReport イベントタイプ　StartUp 優先度　4 重大度　INFO
ログメッセージ　Node node_id: Node restart completed copy of dictionary information 説明　再起動されたノードへのデータディクショナリ情報のコピーが完了しました。	イベント名　NR_CopyDict イベントタイプ　NodeRestart 優先度　8 重大度　INFO
ログメッセージ　Node node_id: Node restart completed copy of distribution information 説明　再起動されたノードへのデータ配布情報のコピーが完了しました。	イベント名　NR_CopyDistr イベントタイプ　NodeRestart 優先度　8 重大度　INFO
ログメッセージ　Node node_id: Node restart starting to copy the fragments to Node node_id 説明　起動データノード node_id へのフラグメントのコピーが開始されました。	イベント名　NR_CopyFragsStarted イベントタイプ　NodeRestart 優先度　8 重大度　INFO

ログメッセージ　Node node_id: Table ID = table_id, fragment ID = fragment_id have been copied to Node node_id 説明　テーブル table_id からのフラグメント fragment_id がデータノード node_id にコピーされました。	イベント名　NR_CopyFragDone イベントタイプ　NodeRestart 優先度　10 重大度　INFO
ログメッセージ　Node node_id: Node restart completed copying the fragments to Node node_id 説明　再起動データノード node_id へのすべてのテーブルフラグメントのコピーが完了しました。	イベント名　NR_CopyFragsCompleted イベントタイプ　NodeRestart 優先度　8 重大度　INFO
ログメッセージ　次のいずれか: 1. Node node_id: Node node1_id completed failure of Node node2_id 2. All nodes completed failure of Node node_id 3. Node failure of node_idblock completed 説明　次のいずれか (上記の同じ番号のメッセージにそれぞれ対応します): 1. データノード node1_id がデータノード node2_id の障害を検出しました。 2. すべての (残りの) データノードがデータノード node_id の障害を検出しました。 3. データノード node_id の障害が blockNDB カーネルブロックで検出されました。ここで、ブロックは DBTC、DBDICT、DBDIH、または DBLQH のいずれかです。詳細は、NDB Kernel Blocksを参照してください。	イベント名　NodeFailCompleted イベントタイプ　NodeRestart 優先度　8 重大度　ALERT
ログメッセージ　Node mgm_node_id: Node data_node_id has failed. The Node state at failure was state_code 説明　データノードで障害が発生しました。障害発生時の状態は、アービトレーション状態コード state_code で記述されます。可能性のある状態コードの値は、include/kernel/signaldata/ArbitSignalData.hpp ファイルで見つかります。	イベント名　NODE_FAILREP イベントタイプ　NodeRestart 優先度　8 重大度　ALERT
ログメッセージ　President restarts arbitration thread [state=state_code] または Prepare arbitrator node node_id [ticket=ticket_id] または Receive arbitrator node node_id [ticket=ticket_id] または Started arbitrator node node_id [ticket=ticket_id] または Lost arbitrator node node_id - process failure [state=state_code] または Lost arbitrator node node_id - process exit [state=state_code] または Lost arbitrator node node_id - error_message [state=state_code] 説明　これは、クラスタ内のアービトレーションの現在の状態および進行状況に関するレポートです。node_id は、アービトレータとして選択された管理ノードまたは SQL ノードのノード ID です。state_code は、include/kernel/signaldata/ArbitSignalData.hpp で見つかるアービトレーション状態コードです。エラーが発生すると、ArbitSignalData.hpp でも定義されている error_message が表示されます。ticket_id は、アービトレータが選択された時にそれによって、その選択に参加したすべてのノードに渡された一意の識別子です。これは、アービトレーションをリクエストする各ノードが選択プロセスに参加したノードのいずれかであることを確認するために使用されます。	イベント名　ArbitState イベントタイプ　NodeRestart 優先度　6 重大度　INFO

ログメッセージ　Arbitration check lost - less than 1/2 nodes left または Arbitration check won - all node groups and more than 1/2 nodes left または Arbitration check won - node group majority または Arbitration check lost - missing node group または Network partitioning - arbitration required または Arbitration won - positive reply from node node_id または Arbitration lost - negative reply from node node_id または Network partitioning - no arbitrator available または Network partitioning - no arbitrator configured または Arbitration failure - error_message [state=state_code] 説明　このメッセージは、アービトレーションの結果についてレポートします。アービトレーションに失敗した場合、error_message およびアービトレーション state_code が表示されます。これらの両方の定義は、include/kernel/signaldata/ArbitSignalData.hpp で見つかります。	イベント名　ArbitResult イベントタイプ　NodeRestart 優先度　2 重大度　ALERT
ログメッセージ　Node node_id: GCP Take over started 説明　このノードは、次のグローバルチェックポイントに対する責任を負おうとしています (つまり、マスターノードになります)。	イベント名　GCP_TakeoverStarted イベントタイプ　NodeRestart 優先度　7 重大度　INFO
ログメッセージ　Node node_id: GCP Take over completed 説明　このノードはマスターになり、次のグローバルチェックポイントに対する責任を負いました。	イベント名　GCP_TakeoverCompleted イベントタイプ　NodeRestart 優先度　7 重大度　INFO
ログメッセージ　Node node_id: LCP Take over started 説明　このノードは、次のグローバルチェックポイントセットに対する責任を負おうとしています (つまり、マスターノードになります)。	イベント名　LCP_TakeoverStarted イベントタイプ　NodeRestart 優先度　7 重大度　INFO
ログメッセージ　Node node_id: LCP Take over completed 説明　このノードはマスターになり、次のグローバルチェックポイントセットに対する責任を負いました。	イベント名　LCP_TakeoverCompleted イベントタイプ　NodeRestart 優先度　7 重大度　INFO
ログメッセージ　Node node_id: Trans. Count = transactions, Commit Count = commits, Read Count = reads, Simple Read Count = simple_reads, Write Count = writes, AttrInfo Count = AttrInfo_objects, Concurrent Operations = concurrent_operations, Abort Count = aborts, Scans = scans, Range scans = range_scans 説明　このトランザクションアクティビティーのレポートは、約 10 秒に 1 回表示されます	イベント名　TransReportCounters イベントタイプ　Statistic 優先度　8 重大度　INFO
ログメッセージ　Node node_id: Operations=operations 説明　このノードで実行された操作数です (約 10 秒に 1 回表示されます)	イベント名　OperationReportCounters イベントタイプ　Statistic 優先度　8 重大度　INFO
ログメッセージ　Node node_id: Table with ID = table_id created 説明　示されたテーブル ID を持つテーブルが作成されました	イベント名　TableCreated イベントタイプ　Statistic 優先度　7

	重大度　INFO
ログメッセージ　Node node_id: Mean loop Counter in doJob last 8192 times = count 説明	イベント名　JobStatistic イベントタイプ　Statistic 優先度　9 重大度　INFO
ログメッセージ　Mean send size to Node = node_id last 4096 sends = bytes bytes 説明　このノードはノード node_id への送信あたり、平均で bytes バイト送信しています	イベント名　SendBytesStatistic イベントタイプ　Statistic 優先度　9 重大度　INFO
ログメッセージ　Mean receive size to Node = node_id last 4096 sends = bytes bytes 説明　このノードはノード node_id からデータを受信するたびに、平均で bytes バイトのデータを受信しています	イベント名　ReceiveBytesStatistic イベントタイプ　Statistic 優先度　9 重大度　INFO
ログメッセージ　Node node_id: Data usage is data_memory_percentage% (data_pages_used 32K pages of total data_pages_total) / Node node_id: Index usage is index_memory_percentage% (index_pages_used 8K pages of total index_pages_total) 説明　このレポートは、クラスタ管理クライアントで DUMP 1000 コマンドが発行されると生成されます。詳細は、MySQL NDB Cluster Internals ManualのDUMP 1000を参照してください	イベント名　MemoryUsage イベントタイプ　Statistic 優先度　5 重大度　INFO
ログメッセージ　Node node1_id: Transporter to node node2_id reported error error_code: error_message 説明　ノード node2_id との通信中に、トランスポータのエラーが発生しました。トランスポータのエラーコードおよびメッセージのリストについては、MySQL NDB Cluster Internals ManualのNDB Transporter Errorsを参照してください。	イベント名　TransporterError イベントタイプ　Error 優先度　2 重大度　ERROR
ログメッセージ　Node node1_id: Transporter to node node2_id reported error error_code: error_message 説明　ノード node2_id との通信中の潜在的なトランスポータ問題の警告。トランスポータのエラーコードおよびメッセージのリストについて詳しくは、NDB Transporter Errorsを参照してください。	イベント名　TransporterWarning イベントタイプ　Error 優先度　8 重大度　WARNING
ログメッセージ　Node node1_id: Node node2_id missed heartbeat heartbeat_id 説明　このノードはノード node2_id からのハートビートを受信できませんでした	イベント名　MissedHeartbeat イベントタイプ　Error 優先度　8 重大度　WARNING
ログメッセージ　Node node1_id: Node node2_id declared dead due to missed heartbeat 説明　このノードは、ノード node2_id からの少なくとも 3 つのハートビートを受信できなかったため、そのノードの「停止」を宣言しました	イベント名　DeadDueToHeartbeat イベントタイプ　Error 優先度　8 重大度　ALERT
ログメッセージ　Node node1_id: Node Sent Heartbeat to node = node2_id 説明　このノードはノード node2_id にハートビートを送信しました	イベント名　SentHeartbeat イベントタイプ　Info 優先度　12

	重大度　INFO
ログメッセージ　Node node_id: Event buffer status: used=bytes_used (percent_used%) alloc=bytes_allocated (percent_available%) max=bytes_available apply_gci=latest_restorable_GCI latest_gci=latest_GCI 説明　このレポートは、比較的短期間に多くの更新が適用されるときなど、イベントバッファーの使用率が高いときに表示されます。レポートには、使用済みのイベントバッファーメモリーのバイト数と割合、割り当てられているバイト数とまだ使用可能な割合、および最新とリストア可能な最新のグローバルチェックポイントが表示されます。	イベント名　EventBufferStatus イベントタイプ　Info 優先度　7 重大度　INFO
ログメッセージ　Node node_id: Entering single user mode, Node node_id: Entered single user mode Node API_node_id has exclusive access, Node node_id: Entering single user mode 説明　これらのレポートは、シングルユーザーモードの開始時および終了時にクラスタログに書き込まれます。API_node_id は、クラスタへの排他アクセス権を持っている API または SQL のノード ID です (詳細は、セクション18.5.8「MySQL Cluster のシングルユーザーモード」を参照してください)。「Unknown single user report API_node_id」というメッセージは、エラーが発生したことを示し、通常の操作では表示されません。	イベント名　SingleUser イベントタイプ　Info 優先度　7 重大度　INFO
ログメッセージ　Node node_id: Backup backup_id started from node mgm_node_id 説明　mgm_node_id を持つ管理ノードを使用して、バックアップが開始されました。このメッセージは、START BACKUP コマンドの発行時にクラスタ管理クライアントにも表示されます。詳細は、セクション18.5.3.2「MySQL Cluster 管理クライアントを使用したバックアップの作成」を参照してください。	イベント名　BackupStarted イベントタイプ　Backup 優先度　7 重大度　INFO
ログメッセージ　Node node_id: Backup backup_id started from node mgm_node_id completed. StartGCP: start_gcp StopGCP: stop_gcp #Records: records #LogRecords: log_records Data: data_bytes bytes Log: log_bytes bytes 説明　ID backup_id を持つバックアップが完了しました。詳細は、セクション18.5.3.2「MySQL Cluster 管理クライアントを使用したバックアップの作成」を参照してください	イベント名　BackupCompleted イベントタイプ　Backup 優先度　7 重大度　INFO
ログメッセージ　Node node_id: Backup request from mgm_node_id failed to start. Error: error_code 説明　バックアップの開始に失敗しました。エラーコードについては、MGM API Errorsを参照してください	イベント名　BackupFailedToStart イベントタイプ　Backup 優先度　7 重大度　ALERT
ログメッセージ　Node node_id: Backup backup_id started from mgm_node_id has been aborted. Error: error_code 説明　バックアップが開始後に終了しました。ユーザーの介入に原因がある可能性があります	イベント名　BackupAborted イベントタイプ　Backup 優先度　7 重大度　ALERT

18.5.7.2 MySQL Cluster: NDB トランスポータのエラー

このセクションでは、トランスポータエラーの発生時にクラスタログに書き込まれるエラーのコード、名前、およびメッセージを一覧表示します。

エラーコード	エラー名	エラーテキスト
0x00	TE_NO_ERROR	No error

エラーコード	エラー名	エラーテキスト
0x01	TE_ERROR_CLOSING_SOCKET	Error found during closing of socket
0x02	TE_ERROR_IN_SELECT_BEFORE_ACCEPT	Error found before accept.The transporter will retry
0x03	TE_INVALID_MESSAGE_LENGTH	Error found in message (invalid message length)
0x04	TE_INVALID_CHECKSUM	Error found in message (checksum)
0x05	TE_COULD_NOT_CREATE_SOCKET	Error found while creating socket(can't create socket)
0x06	TE_COULD_NOT_BIND_SOCKET	Error found while binding server socket
0x07	TE_LISTEN_FAILED	Error found while listening to server socket
0x08	TE_ACCEPT_RETURN_ERROR	Error found during accept(accept return error)
0x0b	TE_SHM_DISCONNECT	The remote node has disconnected
0x0c	TE_SHM_IPC_STAT	Unable to check shm segment
0x0d	TE_SHM_UNABLE_TO_CREATE_SEGMENT	Unable to create shm segment
0x0e	TE_SHM_UNABLE_TO_ATTACH_SEGMENT	Unable to attach shm segment
0x0f	TE_SHM_UNABLE_TO_REMOVE_SEGMENT	Unable to remove shm segment
0x10	TE_TOO_SMALL_SIGID	Sig ID too small
0x11	TE_TOO_LARGE_SIGID	Sig ID too large
0x12	TE_WAIT_STACK_FULL	Wait stack was full
0x13	TE_RECEIVE_BUFFER_FULL	Receive buffer was full
0x14	TE_SIGNAL_LOST_SEND_BUFFER_FULL	Send buffer was full,and trying to force send fails
0x15	TE_SIGNAL_LOST	Send failed for unknown reason(signal lost)
0x16	TE_SEND_BUFFER_FULL	The send buffer was full, but sleeping for a while solved
0x0017	TE_SCI_LINK_ERROR	There is no link from this node to the switch
0x18	TE_SCI_UNABLE_TO_START_SEQUENCE	Could not start a sequence, because system resources are exumed or no sequence has been created
0x19	TE_SCI_UNABLE_TO_REMOVE_SEQUENCE	Could not remove a sequence
0x1a	TE_SCI_UNABLE_TO_CREATE_SEQUENCE	Could not create a sequence, because system resources are exempted.Must reboot
0x1b	TE_SCI_UNRECOVERABLE_DATA_TFX_ERROR	Tried to send data on redundant link but failed
0x1c	TE_SCI_CANNOT_INIT_LOCALSEGMENT	Cannot initialize local segment
0x1d	TE_SCI_CANNOT_MAP_REMOTESEGMENT	Cannot map remote segment
0x1e	TE_SCI_UNABLE_TO_UNMAP_SEGMENT	Cannot free the resources used by this segment (step 1)
0x1f	TE_SCI_UNABLE_TO_REMOVE_SEGMENT	Cannot free the resources used by this segment (step 2)

エラーコード	エラー名	エラーテキスト
0x20	TE_SCI_UNABLE_TO_DISCONNECT_SEGMENT	Cannot disconnect from a remote segment
0x21	TE_SHM_IPC_PERMANENT	Shm ipc Permanent error
0x22	TE_SCI_UNABLE_TO_CLOSE_CHANNEL	Unable to close the sci channel and the resources allocated

18.5.8 MySQL Cluster のシングルユーザーモード

シングルユーザーモードにより、データベース管理者はデータベースシステムへのアクセスを、MySQL サーバー (SQL ノード) や ndb_restore のインスタンスなど単一の API ノードに制限できます。シングルユーザーモードに入ると、その他のすべての API ノードへの接続が正常に閉じられ、実行中のトランザクションがすべて中止されます。新しいトランザクションの開始は許可されません。

クラスタがシングルユーザーモードに入ると、指定された API ノードにのみデータベースへのアクセス権が付与されます。

ndb_mgm クライアントで ALL STATUS コマンドを使用すると、クラスタがシングルユーザーモードに入ったタイミングを確認できます。また、ndbinfo.nodes テーブルの status カラムをチェックすることもできます (詳細は、セクション18.5.10.18「ndbinfo nodes テーブル」を参照してください)。

例:

```
ndb_mgm> ENTER SINGLE USER MODE 5
```

このコマンドが実行され、クラスタがシングルユーザーモードに入ると、ノード ID が 5 の API ノードがクラスタで許可された唯一のユーザーになります。

前述のコマンドで指定されたノードは API ノードである必要があるため、その他のノードタイプを指定しようとしても拒否されます。

> **注記**
>
> 前述のコマンドが呼び出されると、指定されたノードで実行されているすべてのトランザクションが中止され、接続が閉じられるため、サーバーを再起動する必要があります。

EXIT SINGLE USER MODE コマンドは、クラスタのデータノードの状態をシングルユーザーモードから通常モードに変更します。接続を待機している (つまり、クラスタの準備ができ、使用可能になるまで待機している) MySQL サーバーなどの API ノードは、接続が再度許可されます。単一ユーザーノードとして指定された API ノードは、状態の変更中および変更後も引き続き実行されます (まだ接続されている場合)。

例:

```
ndb_mgm> EXIT SINGLE USER MODE
```

シングルユーザーモードでの実行時にノードの障害を処理する際には、次の 2 つの方法が推奨されています。

- 方法 1:
  1. シングルユーザーモードのトランザクションをすべて終了します
  2. EXIT SINGLE USER MODE コマンドを発行します
  3. クラスタのデータノードを再起動します
- 方法 2:

  シングルユーザーモードに入る前にデータベースノードを再起動します。

18.5.9 クイックリファレンス: MySQL Cluster の SQL ステートメント

このセクションでは、MySQL Cluster に接続された MySQL サーバーを管理およびモニターする際に役立つ可能性のあるいくつかの SQL ステートメントについて説明し、場合によってクラスタ自体に関する情報も提供します。

- SHOW ENGINE NDB STATUS、SHOW ENGINE NDBCLUSTER STATUS

  このステートメントの出力には、クラスタへのサーバーの接続、MySQL Cluster オブジェクトの作成と使用、および MySQL Cluster レプリケーションのバイナリロギングに関する情報が含まれます。

  使用例および詳細な情報については、セクション13.7.5.16「SHOW ENGINE 構文」を参照してください。

- SHOW ENGINES

  このステートメントを使用すると、MySQL サーバーでクラスタ化のサポートが有効になっているかどうか、およびその場合はアクティブであるかどうかを確認できます。

  詳細は、セクション13.7.5.17「SHOW ENGINES 構文」を参照してください。

  > **注記**
  >
  > このステートメントでは、LIKE 句がサポートされていません。ただし、次の項目で説明するように、LIKE を使用すれば、INFORMATION_SCHEMA.ENGINES テーブルに対するクエリーをフィルタ処理できます。

- SELECT * FROM INFORMATION_SCHEMA.ENGINES [WHERE ENGINE LIKE 'NDB%']

  これは、SHOW ENGINES と同等ですが、INFORMATION_SCHEMA の ENGINES テーブルを使用します。SHOW ENGINES ステートメントを使用する場合とは異なり、LIKE 句を使用して結果をフィルタ処理したり、特定のカラムを選択してスクリプトで役立つ可能性のある情報を取得したりできます。たとえば、次のクエリーは、NDB サポートを使用してサーバーが構築されたかどうか、およびその場合は有効になっているかどうかを表示します。

```
mysql> SELECT SUPPORT FROM INFORMATION_SCHEMA.ENGINES
    -> WHERE ENGINE LIKE 'NDB%';
+---------+
| support |
+---------+
| ENABLED |
+---------+
```

  詳細は、セクション21.6「INFORMATION_SCHEMA ENGINES テーブル」を参照してください。

- SHOW VARIABLES LIKE 'NDB%'

  このステートメントは、次に示すように、NDB ストレージエンジンに関するほとんどのサーバーシステム変数、およびそれらの値のリストを表示します。

```
mysql> SHOW VARIABLES LIKE 'NDB%';
+-----------------------------------+-------+
| Variable_name                     | Value |
+-----------------------------------+-------+
| ndb_autoincrement_prefetch_sz     | 32    |
| ndb_cache_check_time              | 0     |
| ndb_extra_logging                 | 0     |
| ndb_force_send                    | ON    |
| ndb_index_stat_cache_entries      | 32    |
| ndb_index_stat_enable             | OFF   |
| ndb_index_stat_update_freq        | 20    |
| ndb_report_thresh_binlog_epoch_slip | 3     |
| ndb_report_thresh_binlog_mem_usage | 10    |
| ndb_use_copying_alter_table       | OFF   |
| ndb_use_exact_count               | ON    |
| ndb_use_transactions              | ON    |
+-----------------------------------+-------+
```

  詳細は、セクション5.1.4「サーバーシステム変数」を参照してください。

- SELECT * FROM INFORMATION_SCHEMA.GLOBAL_VARIABLES WHERE VARIABLE_NAME LIKE 'NDB%';

  このステートメントは、前の項目で説明した SHOW コマンドと同等であり、次に示すように、ほぼ同じ出力を生成します。

```
mysql> SELECT * FROM INFORMATION_SCHEMA.GLOBAL_VARIABLES
    -> WHERE VARIABLE_NAME LIKE 'NDB%';
+-----------------------------------+----------------+
| VARIABLE_NAME                     | VARIABLE_VALUE |
+-----------------------------------+----------------+
| NDB_AUTOINCREMENT_PREFETCH_SZ     | 32             |
```

```
| NDB_CACHE_CHECK_TIME              | 0           |
| NDB_EXTRA_LOGGING                 | 0           |
| NDB_FORCE_SEND                    | ON          |
| NDB_INDEX_STAT_CACHE_ENTRIES      | 32          |
| NDB_INDEX_STAT_ENABLE             | OFF         |
| NDB_INDEX_STAT_UPDATE_FREQ        | 20          |
| NDB_REPORT_THRESH_BINLOG_EPOCH_SLIP | 3           |
| NDB_REPORT_THRESH_BINLOG_MEM_USAGE  | 10          |
| NDB_USE_COPYING_ALTER_TABLE       | OFF         |
| NDB_USE_EXACT_COUNT               | ON          |
| NDB_USE_TRANSACTIONS              | ON          |
+-----------------------------------+-------------+
```

SHOW コマンドを使用する場合とは異なり、個々のカラムを選択できます。例:

```
mysql> SELECT VARIABLE_VALUE
    ->   FROM INFORMATION_SCHEMA.GLOBAL_VARIABLES
    ->   WHERE VARIABLE_NAME = 'ndb_force_send';
+----------------+
| VARIABLE_VALUE |
+----------------+
| ON             |
+----------------+
```

詳細は、セクション21.9「INFORMATION_SCHEMA GLOBAL_VARIABLES および SESSION_VARIABLES テーブル」およびセクション5.1.4「サーバーシステム変数」を参照してください。

- SHOW STATUS LIKE 'NDB%'

  このステートメントは、MySQL サーバーがクラスタ SQL ノードとして動作しているかどうかをひと目でわかるように表示し、その場合は次に示すように、MySQL サーバーのクラスタノード ID、接続先のクラスタ管理サーバーのホスト名とポート、およびクラスタ内のデータノードの数を表示します。

```
mysql> SHOW STATUS LIKE 'NDB%';
+--------------------------+---------------+
| Variable_name            | Value         |
+--------------------------+---------------+
| Ndb_cluster_node_id      | 10            |
| Ndb_config_from_host     | 192.168.0.103 |
| Ndb_config_from_port     | 1186          |
| Ndb_number_of_data_nodes | 4             |
+--------------------------+---------------+
```

  MySQL サーバーがクラスタ化サポートを使用して構築されたが、クラスタに接続されていない場合は、このステートメントの出力のすべての行に、ゼロまたは空の文字列が格納されます。

```
mysql> SHOW STATUS LIKE 'NDB%';
+--------------------------+-------+
| Variable_name            | Value |
+--------------------------+-------+
| Ndb_cluster_node_id      | 0     |
| Ndb_config_from_host     |       |
| Ndb_config_from_port     | 0     |
| Ndb_number_of_data_nodes | 0     |
+--------------------------+-------+
```

  セクション13.7.5.36「SHOW STATUS 構文」も参照してください。

- SELECT * FROM INFORMATION_SCHEMA.GLOBAL_STATUS WHERE VARIABLE_NAME LIKE 'NDB%';

  このステートメントは、前の項目で説明した SHOW コマンドと同様の出力を生成します。ただし、SHOW STATUS を使用する場合とは異なり、SELECT を使用して、モニタリングおよび自動化のためにスクリプトで使用するための SQL の値を抽出できます。

  詳細は、セクション21.8「INFORMATION_SCHEMA GLOBAL_STATUS および SESSION_STATUS テーブル」を参照してください。

また、多くの MySQL Cluster 操作に関するリアルタイムのデータについて、ndbinfo 情報データベース内のテーブルをクエリーすることもできます。セクション18.5.10「ndbinfo MySQL Cluster 情報データベース」を参照してください。

18.5.10 ndbinfo MySQL Cluster 情報データベース

ndbinfo は、MySQL Cluster に固有の情報を格納するデータベースです。

このデータベースは多くのテーブルを格納し、各テーブルは、MySQL Cluster ノードのステータス、リソースの使用率、および操作に関するさまざまな種類のデータを提供します。次のいくつかのセクションで、これらの各テーブルに関する詳細な情報を見つけることができます。

ndbinfo は、MySQL Cluster サポートとともに MySQL サーバーに付属しているため、特別なコンパイルや構成のステップは必要ありません。MySQL サーバーがクラスタに接続すると、MySQL サーバーによってテーブルが作成されます。SHOW PLUGINS を使用すると、特定の MySQL サーバーインスタンスで ndbinfo サポートがアクティブになっていることを確認できます。ndbinfo サポートが有効になっている場合は、次に (強調表示したテキストで) 示すように、Name カラムに ndbinfo が含まれ、Status カラムに ACTIVE が含まれる行が表示されます。

```
mysql> SHOW PLUGINS;
+----------------------------------+--------+--------------------+---------+---------+
| Name                             | Status | Type               | Library | License |
+----------------------------------+--------+--------------------+---------+---------+
| binlog                           | ACTIVE | STORAGE ENGINE     | NULL    | GPL     |
| mysql_native_password            | ACTIVE | AUTHENTICATION     | NULL    | GPL     |
| mysql_old_password               | ACTIVE | AUTHENTICATION     | NULL    | GPL     |
| CSV                              | ACTIVE | STORAGE ENGINE     | NULL    | GPL     |
| MEMORY                           | ACTIVE | STORAGE ENGINE     | NULL    | GPL     |
| MRG_MYISAM                       | ACTIVE | STORAGE ENGINE     | NULL    | GPL     |
| MyISAM                           | ACTIVE | STORAGE ENGINE     | NULL    | GPL     |
| PERFORMANCE_SCHEMA               | ACTIVE | STORAGE ENGINE     | NULL    | GPL     |
| BLACKHOLE                        | ACTIVE | STORAGE ENGINE     | NULL    | GPL     |
| ARCHIVE                          | ACTIVE | STORAGE ENGINE     | NULL    | GPL     |
| ndbcluster                       | ACTIVE | STORAGE ENGINE     | NULL    | GPL     |
| ndbinfo                          | ACTIVE | STORAGE ENGINE     | NULL    | GPL     |
| ndb_transid_mysql_connection_map | ACTIVE | INFORMATION SCHEMA | NULL    | GPL     |
| InnoDB                           | ACTIVE | STORAGE ENGINE     | NULL    | GPL     |
| INNODB_TRX                       | ACTIVE | INFORMATION SCHEMA | NULL    | GPL     |
| INNODB_LOCKS                     | ACTIVE | INFORMATION SCHEMA | NULL    | GPL     |
| INNODB_LOCK_WAITS                | ACTIVE | INFORMATION SCHEMA | NULL    | GPL     |
| INNODB_CMP                       | ACTIVE | INFORMATION SCHEMA | NULL    | GPL     |
| INNODB_CMP_RESET                 | ACTIVE | INFORMATION SCHEMA | NULL    | GPL     |
| INNODB_CMPMEM                    | ACTIVE | INFORMATION SCHEMA | NULL    | GPL     |
| INNODB_CMPMEM_RESET              | ACTIVE | INFORMATION SCHEMA | NULL    | GPL     |
| partition                        | ACTIVE | STORAGE ENGINE     | NULL    | GPL     |
+----------------------------------+--------+--------------------+---------+---------+
22 rows in set (0.00 sec)
```

また、次に (強調表示されたテキストで) 示すように、Engine カラムに ndbinfo が含まれ、Support カラムに YES が含まれる行があるかどうかについて、SHOW ENGINES の出力をチェックすることで、これを実行することもできます。

```
mysql> SHOW ENGINES\G
*************************** 1. row ***************************
      Engine: ndbcluster
     Support: YES
     Comment: Clustered, fault-tolerant tables
Transactions: YES
          XA: NO
  Savepoints: NO
*************************** 2. row ***************************
      Engine: MRG_MYISAM
     Support: YES
     Comment: Collection of identical MyISAM tables
Transactions: NO
          XA: NO
  Savepoints: NO
*************************** 3. row ***************************
      Engine: ndbinfo
     Support: YES
     Comment: MySQL Cluster system information storage engine
Transactions: NO
          XA: NO
  Savepoints: NO
*************************** 4. row ***************************
      Engine: CSV
     Support: YES
     Comment: CSV storage engine
Transactions: NO
          XA: NO
  Savepoints: NO
*************************** 5. row ***************************
      Engine: MEMORY
     Support: YES
     Comment: Hash based, stored in memory, useful for temporary tables
Transactions: NO
          XA: NO
```

```
  Savepoints: NO
*************************** 6. row ***************************
      Engine: FEDERATED
     Support: NO
     Comment: Federated MySQL storage engine
Transactions: NULL
          XA: NULL
  Savepoints: NULL
*************************** 7. row ***************************
      Engine: ARCHIVE
     Support: YES
     Comment: Archive storage engine
Transactions: NO
          XA: NO
  Savepoints: NO
*************************** 8. row ***************************
      Engine: InnoDB
     Support: YES
     Comment: Supports transactions, row-level locking, and foreign keys
Transactions: YES
          XA: YES
  Savepoints: YES
*************************** 9. row ***************************
      Engine: MyISAM
     Support: DEFAULT
     Comment: Default engine as of MySQL 3.23 with great performance
Transactions: NO
          XA: NO
  Savepoints: NO
*************************** 10. row ***************************
      Engine: BLACKHOLE
     Support: YES
     Comment: /dev/null storage engine (anything you write to it disappears)
Transactions: NO
          XA: NO
  Savepoints: NO
10 rows in set (0.00 sec)
```

ndbinfo サポートが有効になっている場合は、mysql や別の MySQL クライアントで SQL ステートメントを使用して、ndbinfo にアクセスできます。たとえば、次に示すように、SHOW DATABASES の出力に ndbinfo が一覧表示されます。

```
mysql> SHOW DATABASES;
+--------------------+
| Database           |
+--------------------+
| information_schema |
| mysql              |
| ndbinfo            |
| test               |
+--------------------+
4 rows in set (0.00 sec)
```

--ndbcluster オプションを付けて mysqld プロセスが起動されなかった場合は、ndbinfo を使用できず、SHOW DATABASES によって表示されません。以前に mysqld が MySQL Cluster に接続されたが、(クラスタのシャットダウンやネットワーク接続の損失などのイベントが原因で) クラスタを使用できなくなった場合、ndbinfo およびそのテーブルは引き続き表示されますが、(blocks または config_params 以外の) テーブルにアクセスしようとしても、次のエラーで失敗します: Got error 157 'Connection to NDB failed' from NDBINFO

blocks および config_params テーブルを除いて、ndbinfo 「テーブル」と呼ばれるものは、実際には、MySQL サーバーから通常見えない内部 NDB テーブルから生成されるビューです。

ndbinfo テーブルはすべて読み取り専用であり、クエリーの実行時に要求に応じて生成されます。これらの多くはデータノードによって並列して生成されますが、その他は特定の SQL ノードに固有であるため、一貫性のあるスナップショットを提供する保証はありません。

さらに、ndbinfo テーブルでは、結合のプッシュダウンがサポートされていません。そのため、クエリーで WHERE 句を使用している場合でも、大きな ndbinfo テーブルの結合には、リクエスト元の API ノードに大量のデータを転送する必要があることがあります。

クエリーキャッシュに、ndbinfo テーブルが含まれていません。(Bug #59831)

その他のデータベースの場合と同様に、USE ステートメントで ndbinfo データベースを選択してから、SHOW TABLES ステートメントを発行すると、次のようなテーブルのリストを取得できます。

```
mysql> USE ndbinfo;
```

```
Database changed

mysql> SHOW TABLES;
+---------------------+
| Tables_in_ndbinfo   |
+---------------------+
| blocks              |
| cluster_operations  |
| cluster_transactions |
| config_params       |
| counters            |
| diskpagebuffer      |
| logbuffers          |
| logspaces           |
| memoryusage         |
| nodes               |
| resources           |
| server_operations   |
| server_transactions |
| threadblocks        |
| threadstat          |
| transporters        |
+---------------------+
16 rows in set (0.04 sec)
```

通常の想定どおりに、これらのテーブルに対して SELECT ステートメントを実行できます。

```
mysql> SELECT * FROM memoryusage;
+---------+--------------------+--------+------------+------------+-------------+
| node_id | memory_type        | used   | used_pages | total      | total_pages |
+---------+--------------------+--------+------------+------------+-------------+
|       5 | Data memory        | 753664 |         23 | 1073741824 |       32768 |
|       5 | Index memory       | 163840 |         20 | 1074003968 |      131104 |
|       5 | Long message buffer |   2304 |          9 |   67108864 |      262144 |
|       6 | Data memory        | 753664 |         23 | 1073741824 |       32768 |
|       6 | Index memory       | 163840 |         20 | 1074003968 |      131104 |
|       6 | Long message buffer |   2304 |          9 |   67108864 |      262144 |
+---------+--------------------+--------+------------+------------+-------------+
6 rows in set (0.02 sec)
```

memoryusage テーブルを使用する次の 2 つの SELECT ステートメントのような、さらに複雑なクエリーも実行できます。

```
mysql> SELECT SUM(used) as 'Data Memory Used, All Nodes'
    >     FROM memoryusage
    >     WHERE memory_type = 'Data memory';
+-----------------------------+
| Data Memory Used, All Nodes |
+-----------------------------+
|                        6460 |
+-----------------------------+
1 row in set (0.37 sec)

mysql> SELECT SUM(max) as 'Total IndexMemory Available'
    >     FROM memoryusage
    >     WHERE memory_type = 'Index memory';
+-----------------------------+
| Total IndexMemory Available |
+-----------------------------+
|                       25664 |
+-----------------------------+
1 row in set (0.33 sec)
```

ndbinfo テーブルおよびカラムの名前は、(ndbinfo データベース自体の名前と同様に) 大文字と小文字が区別されます。これらの識別子は小文字です。大文字と小文字を誤って使用すると、次の例で示すようなエラーが発生します。

```
mysql> SELECT * FROM nodes;
+---------+--------+---------+-------------+
| node_id | uptime | status  | start_phase |
+---------+--------+---------+-------------+
|       1 |  13602 | STARTED |           0 |
|       2 |     16 | STARTED |           0 |
+---------+--------+---------+-------------+
2 rows in set (0.04 sec)

mysql> SELECT * FROM Nodes;
ERROR 1146 (42S02): Table 'ndbinfo.Nodes' doesn't exist
```

mysqldump は、ndbinfo データベースを完全に無視し、どの出力からも除外します。このことは、--databases または --all-databases オプションを使用する場合でも当てはまります。

MySQL Cluster は、INFORMATION_SCHEMA 情報データベース内に、MySQL Cluster ディスクデータストレージ用に使用されるファイルに関する情報を格納する FILES テーブルなどのテーブルも保持しています。詳細は、セクション21.30「MySQL Cluster の INFORMATION_SCHEMA テーブル」を参照してください。

18.5.10.1 ndbinfo arbitrator_validity_detail テーブル

arbitrator_validity_detail テーブルは、クラスタ内の各データノードがアービトレータについて保持するビューを示します。これは、membership テーブルのサブセットです。

次の表には、arbitrator_validity_detail テーブルのカラムに関する情報を示します。この表には、各カラムの名前、データ型、および簡単な説明を示します。追加情報については、表のあとに示す注記で見つけることができます。

カラム名	型	説明
node_id	整数	このノードのノード ID
arbitrator	整数	アービトレータのノード ID
arb_ticket	文字列	アービトレーションを追跡するために使用される内部識別子
arb_connected	Yes または No	このノードがアービトレータに接続されているかどうか
arb_state	列挙 (テキストを参照)	アービトレーションの状態

このノード ID は、ndb_mgm -e "SHOW" でレポートされるものと同じです。

すべてのノードで arb_state の値が同じであるだけでなく、arbitrator と arb_ticket の値も同じです。可能性のある arb_state の値は ARBIT_NULL、ARBIT_INIT、ARBIT_FIND、ARBIT_PREP1、ARBIT_PREP2、ARBIT_START、ARBIT_RUN、ARBIT_CH および UNKNOWN です。

arb_connected は、現在のノードが arbitrator に接続されているかどうかを示します。

18.5.10.2 ndbinfo arbitrator_validity_summary テーブル

arbitrator_validity_summary テーブルは、クラスタのデータノードに関するアービトレータの複合ビューを示します。

次の表には、arbitrator_validity_summary テーブルのカラムに関する情報を示します。この表には、各カラムの名前、データ型、および簡単な説明を示します。追加情報については、表のあとに示す注記で見つけることができます。

カラム名	型	説明
arbitrator	整数	アービトレータのノード ID
arb_ticket	文字列	アービトレーションを追跡するために使用される内部識別子
arb_connected	Yes または No	このアービトレータがクラスタに接続されているかどうか
consensus_count	整数	このノードをアービトレータとして見ているデータノードの数

通常の操作では、このテーブルには、相当に長い期間でも 1 行しか含まれません。やや長い期間で複数行が含まれる場合は、一部のノードがアービトレータに接続されていないか、またはすべてのノードが接続されているが、同じアービトレータを承認していません。

arbitrator カラムは、アービトレータのノード ID を示します。

arb_ticket は、このアービトレータで使用される内部識別子です。

arb_connected は、このノードがアービトレータとしてクラスタに接続されているかどうかを示します。

18.5.10.3 ndbinfo blocks テーブル

blocks テーブルは、すべての NDB カーネルブロックの名前と内部 ID を単に格納する静的テーブルです (NDB Kernel Blocksを参照してください)。これは、人間が読める出力を生成するためにブロック番号をブロック名にマップする際に、その他の ndbinfo テーブル (大部分が実際にはビューです) で使用されます。

次の表は、blocks テーブルのカラムに関する情報を示します。この表には、各カラムの名前、データ型、および簡単な説明を示します。追加情報については、表のあとに示す注記で見つけることができます。

カラム名	型	説明
block_number	整数	ブロック番号
block_name	文字列	ブロック名

すべてのブロック名のリストを取得するには、SELECT block_name FROM ndbinfo.blocks を実行するだけです。これは静的テーブルですが、その内容はさまざまな MySQL Cluster リリース間で異なる可能性があります。

18.5.10.4 ndbinfo cluster_operations テーブル

cluster_operations テーブルは、ローカルデータ管理 (LQH) ブロック (The DBLQH Blockを参照してください) の観点から、MySQL Cluster のすべてのアクティビティー操作ごとの (ステートフルな主キー操作) ビューを示します。

次の表に、cluster_operations テーブルのカラムに関する情報を示します。この表には、各カラムの名前、データ型、および簡単な説明を示します。追加情報については、表のあとに示す注記で見つけることができます。

カラム名	型	説明
node_id	整数	レポート LQH ブロックのノード ID
block_instance	整数	LQH ブロックインスタンス
transid	整数	トランザクション ID
operation_type	文字列	操作のタイプ (可能性のある値についてはテキストを参照)
state	文字列	操作の状態 (可能性のある値についてはテキストを参照)
tableid	整数	テーブル ID
fragmentid	整数	フラグメント ID
client_node_id	整数	クライアントノード ID
client_block_ref	整数	クライアントのブロック参照
tc_node_id	整数	トランザクションコーディネータノード ID
tc_block_no	整数	トランザクションコーディネータブロック番号
tc_block_instance	整数	トランザクションコーディネータブロックインスタンス

トランザクション ID は、NDB API の getTransactionId() メソッドを使用して取得できる一意の 64 ビット数値です。(現在、MySQL サーバーは進行中のトランザクションの NDB API トランザクション ID を公開しません。)

operation_type カラムは、READ、READ-SH、READ-EX、INSERT、UPDATE、DELETE、WRITE、UNLOCK、REFRESH、SCAN、SCAN-SH、SCAN-EX、または <unknown> のいずれかの値を取ることができます。

state カラムは、ABORT_QUEUED、ABORT_STOPPED、COMMITTED、COMMIT_QUEUED、COMMIT_STOPPED、COPY_CLOSE_STO または WAIT_TUP_TO_ABORT のいずれかの値を取ることができます。(ndbinfo_show_hidden を有効にして MySQL サーバーが実行されている場合は、通常は非表示になっている ndb$dblqh_tcconnect_state テーブルから選択することで、この状態のリストを表示できます。)

ndb_show_tables の出力をチェックして、テーブル ID から NDB テーブルの名前を取得できます。

fragid は、ndb_desc --extra-partition-info (短縮形式 -p) の出力で見られるパーティション番号と同じです。

client_node_id および client_block_ref の client は MySQL Cluster の API ノードまたは SQL ノード (つまり、クラスタに接続されている NDB API クライアントまたは MySQL サーバー) を表します。

18.5.10.5 ndbinfo cluster_transactions テーブル

cluster_transactions テーブルは、MySQL Cluster 内で進行中のすべてのトランザクションに関する情報を示します。

次の表に、cluster_transactions テーブルのカラムに関する情報を示します。この表には、各カラムの名前、データ型、および簡単な説明を示します。追加情報については、表のあとに示す注記で見つけることができます。

カラム名	型	説明
node_id	整数	トランザクションコーディネータのノード ID
block_instance	整数	TC ブロックインスタンス
transid	整数	トランザクション ID
state	文字列	操作の状態 (可能性のある値についてはテキストを参照)
count_operations	整数	トランザクション内のステートフルな主キー操作 (DML 操作に加えて、ロックを伴う読み取りを含む) の数
outstanding_operations	整数	ローカルデータ管理ブロックでまだ実行されている操作
inactive_seconds	整数	API の待機に要した時間
client_node_id	整数	クライアントノード ID
client_block_ref	整数	クライアントのブロック参照

トランザクション ID は、NDB API の getTransactionId() メソッドを使用して取得できる一意の 64 ビット数値です。(現在、MySQL サーバーは進行中のトランザクションの NDB API トランザクション ID を公開しません。)

state カラムは、CS_ABORTING、CS_COMMITTING、CS_COMMIT_SENT、CS_COMPLETE_SENT、CS_COMPLETING、CS_CONのいずれかの値を持つ可能性があります。(ndbinfo_show_hidden を有効にして MySQL サーバーが実行されている場合は、通常は非表示になっている ndb$dbtc_apiconnect_state テーブルから選択することで、この状態のリストを表示できます。)

client_node_id および client_block_ref の client は MySQL Cluster の API ノードまたは SQL ノード (つまり、クラスタに接続されている NDB API クライアントまたは MySQL サーバー) を表します。

18.5.10.6 ndbinfo config_params テーブル

config_params テーブルは、すべての MySQL Cluster 構成パラメータの名前および内部 ID 番号を提供する静的テーブルです。

次の表に、config_params テーブルのカラムに関する情報を示します。この表には、各カラムの名前、データ型、および簡単な説明を示します。追加情報については、表のあとに示す注記で見つけることができます。

カラム名	型	説明
param_number	整数	パラメータの内部 ID 番号
param_name	文字列	パラメータの名前

これは静的テーブルですが、ソフトウェアリリースやクラスタハードウェアの構成やその他の要因の違いによって、サポートされているパラメータが異なる可能性があるため、その内容は MySQL Cluster インストール間で異なる可能性があります。

18.5.10.7 ndbinfo counters テーブル

counters テーブルは、特定のカーネルブロックおよびデータノードに対する読み取りや書き込みなどのイベントの現在までの合計数を示します。最近のノードの起動または再起動はカウントされません。ノードを起動または再起動すると、そのノード上のすべてのカウンタがリセットされます。すべてのカーネルブロックですべてのタイプのカウンタを使用しているとはかぎりません。

次の表に、counters テーブルのカラムに関する情報を示します。この表には、各カラムの名前、データ型、および簡単な説明を示します。追加情報については、表のあとに示す注記で見つけることができます。

カラム名	型	説明
node_id	整数	データノード ID
block_name	文字列	関連付けられた NDB カーネルブロックの名前 (NDB Kernel Blocksを参照してください)。
block_instance	整数	ブロックインスタンス
counter_id	整数	カウンタの内部 ID 番号。通常は 1 から 10 までの整数。
counter_name	文字列	カウンタの名前。各カウンタが関連付けられている個々のカウンタおよび NDB カーネルブロックの名前については、テキストを参照してください。
val	整数	カウンタの値

各カウンタは、特定の NDB カーネルブロックに関連付けられています。

OPERATIONS カウンタは、DBLQH (ローカルクエリーハンドラ) カーネルブロックに関連付けられています (The DBLQH Blockを参照してください)。主キーの読み取りは、主キーの更新と同様に、1 操作としてカウントされます。読み取りの場合、DBTC での操作ごとに DBLQH での操作が 1 回発生します。書き込みの場合、レプリカごとに 1 回の操作がカウントされます。

ATTRINFO、TRANSACTIONS、COMMITS、READS、LOCAL_READS、SIMPLE_READS、WRITES、LOCAL_WRITES、ABO および RANGE_SCANS カウンタは、DBTC (トランザクションコーディネータ) カーネルブロックに関連付けられています (The DBTC Blockを参照してください)。

LOCAL_WRITES および LOCAL_READS は、レコードのプライマリレプリカも保持するノードで、トランザクションコーディネータを使用する主キー操作です。

READS カウンタには、すべての読み取りが含まれます。LOCAL_READS には、このトランザクションコーディネータと同じノード上でのプライマリレプリカの読み取りのみが含まれます。SIMPLE_READS には、読み取り操作が特定のトランザクションに対する最初と最後の操作である読み取りのみが含まれるため、その他の読み取りとのトランザクション上の関係がありません。

ATTRINFO には、解釈済みプログラムがデータノードに送信される回数のカウントが保持されます。NDB カーネルの ATTRINFO メッセージについての詳細は、NDB Protocol Messagesを参照してください。

LOCAL_TABLE_SCANS_SENT、READS_RECEIVED、PRUNED_RANGE_SCANS_RECEIVED、RANGE_SCANS_RECEIVED および SCAN_ROWS_RETURNED カウンタは、DBSPJ (プッシュダウン結合を選択します) カーネルブロックに関連付けられています (The DBSPJ Blockを参照してください)。

このような問題のトラブルシューティングを行う際に、いくつかのカウンタによってトランスポータの過負荷および送信バッファーのサイズに関する情報が提供されます。LQH インスタンスごとに、次のリストに示す各カウンタのインスタンスが 1 つ存在します。

- LQHKEY_OVERLOAD: トランスポータの過負荷が原因で、LQH ブロックインスタンスで拒否された主キーリクエストの数
- LQHKEY_OVERLOAD_TC: TC ノードのトランスポータが過負荷状態になった LQHKEY_OVERLOAD のインスタンス数
- LQHKEY_OVERLOAD_READER: API リーダー (読み取り専用) ノードが過負荷状態になった LQHKEY_OVERLOAD のインスタンス数。
- LQHKEY_OVERLOAD_NODE_PEER: 次のバックアップデータノード (書き込み専用) が過負荷状態になった LQHKEY_OVERLOAD のインスタンス数
- LQHKEY_OVERLOAD_SUBSCRIBER: イベントサブスクライバ (書き込み専用) が過負荷状態になった LQHKEY_OVERLOAD のインスタンス数。
- LQHSCAN_SLOWDOWNS: スキャン中の API トランスポータの過負荷が原因で、フラグメントスキャンのバッチサイズが減少したインスタンスの数。

18.5.10.8 ndbinfo dict_obj_types テーブル

dict_obj_types テーブルは、NDB カーネルで使用される可能性のあるディクショナリオブジェクトのタイプを一覧表示した静的なテーブルです。これらは、NDB API の Object::Type で定義されるタイプと同じです。

次の表に、dict_obj_types テーブルのカラムに関する情報を示します。この表には、各カラムの名前、データ型、および簡単な説明を示します。

カラム名	型	説明
type_id	整数	このタイプのタイプ ID
type_name	文字列	このタイプの名前

このテーブルは MySQL Cluster NDB 7.4.1 で追加されました。

18.5.10.9 ndbinfo disk_write_speed_base テーブル

disk_write_speed_base テーブルは、LCP、バックアップ、およびリストア操作時のディスク書き込みの速度に関する基本情報を示します。

次の表に、disk_write_speed_base テーブルのカラムに関する情報を示します。この表には、各カラムの名前、データ型、および簡単な説明を示します。追加情報については、表のあとに示す注記で見つけることができます。

カラム名	型	説明
node_id	整数	このノードのノード ID
thr_no	整数	この LDM スレッドのスレッド ID
millis_ago	整数	このレポート期間が終了してからのミリ秒数
millis_passed	整数	このレポート期間内に経過したミリ秒数
backup_lcp_bytes_written	整数	この期間中にローカルチェックポイントおよびバックアッププロセスによってディスクに書き込まれたバイト数
redo_bytes_written	整数	この期間中に Redo ログに書き込まれたバイト数
target_disk_write_speed	整数	LDM スレッド (基本データ) ごとの実際のディスク書き込み速度

このテーブルは MySQL Cluster NDB 7.4.1 で追加されました。

18.5.10.10 ndbinfo disk_write_speed_aggregate テーブル

disk_write_speed_aggregate テーブルは、LCP、バックアップ、およびリストア操作時のディスク書き込みの速度に関して集計された情報を示します。

次の表に、MySQL Cluster NDB 7.4.3 以降での disk_write_speed_aggregate テーブルのカラムに関する情報を示します。MySQL Cluster NDB 7.4.2 以前の disk_write_speed_aggregate に関する情報は、このセクションの後半で見つけることができます。この表には、各カラムの名前、データ型、および簡単な説明を示します。追加情報については、表のあとに示す注記で見つけることができます。

カラム名	型	説明
node_id	整数	このノードのノード ID
thr_no	整数	この LDM スレッドのスレッド ID
backup_lcp_speed_last_sec	整数	過去 1 秒間にバックアップおよび LCP プロセスによってディスクに書き込まれたバイト数
redo_speed_last_sec	整数	過去 1 秒間に Redo ログに書き込まれたバイト数
backup_lcp_speed_last_10sec	整数	バックアップおよび LCP プロセスによってディスクに書き込まれた 1 秒当たりのバイト数 (過去 10 秒間の平均値)
redo_speed_last_10sec	整数	Redo ログにディスクに書き込まれた 1 秒当たりのバイト数 (過去 10 秒間の平均値)

カラム名	型	説明
std_dev_backup_lcp_speed_last_10sec	整数	バックアップおよび LCP プロセスによってディスクに書き込まれた 1 秒当たりのバイト数の標準偏差 (過去 10 秒間の平均値)
std_dev_redo_speed_last_10sec	整数	Redo ログにディスクに書き込まれた 1 秒当たりのバイト数の標準偏差 (過去 10 秒間の平均値)
backup_lcp_speed_last_60sec	整数	バックアップおよび LCP プロセスによってディスクに書き込まれた 1 秒当たりのバイト数 (過去 60 秒間の平均値)
redo_speed_last_60sec	整数	Redo ログにディスクに書き込まれた 1 秒当たりのバイト数 (過去 10 秒間の平均値)
std_dev_backup_lcp_speed_last_60sec	整数	バックアップおよび LCP プロセスによってディスクに書き込まれた 1 秒当たりのバイト数の標準偏差 (過去 60 秒間の平均値)
std_dev_redo_speed_last_60sec	整数	Redo ログにディスクに書き込まれた 1 秒当たりのバイト数の標準偏差 (過去 60 秒間の平均値)
slowdowns_due_to_io_lag	整数	Redo ログの I/O ラグが原因でディスクの書き込み速度が低下した秒数
slowdowns_due_to_high_cpu	整数	高い CPU 使用率が原因でディスクの書き込み速度が低下した秒数
disk_write_speed_set_to_min	整数	ディスクの書き込み速度が最小に設定された秒数
current_target_disk_write_speed	整数	LDM スレッドごとの実際のディスク書き込み速度 (集計値)

MySQL Cluster NDB 7.4.3 よりも前では、このテーブルの std_dev_backup_lcp_speed_last_10sec、std_dev_redo_speed_last_10sec、std_dev_backup_lcp_speed_last_60sec、および std_dev_redo_speed_last_60sec カラムに表示される値を取得する際に使用される標準偏差が正しく計算されませんでした。(Bug #74317、Bug #19795072)

MySQL Cluster NDB 7.4.3 よりも前では、このテーブルの一部のカラムには、値がキロバイトまたはその他のバイトの倍数単位で表示されていました。MySQL Cluster NDB 7.4.3 以降では、このようなカラムはすべて、値がバイト単位で表示されます。(Bug #74317、Bug #19795072) 次の表に、以前に実装された disk_write_speed_aggregate に関する情報を示します。

カラム名	型	説明
node_id	整数	このノードのノード ID
thr_no	整数	この LDM スレッドのスレッド ID
backup_lcp_speed_last_sec	整数	過去 1 秒間にバックアップおよび LCP プロセスによってディスクに書き込まれたキロバイト数
redo_speed_last_sec	整数	過去 1 秒間に Redo ログにディスクに書き込まれたキロバイト数
backup_lcp_speed_last_10sec	整数	バックアップおよび LCP プロセスによってディスクに書き込まれた 1 秒当たりのキロバイト数 (過去 10 秒間の平均値)
redo_speed_last_10sec	整数	Redo ログにディスクに書き込まれた 1 秒当たりのキロバイト数 (過去 10 秒間の平均値)
std_dev_backup_lcp_speed_last_10sec	整数	バックアップおよび LCP プロセスによってディスクに書き込まれた 1 秒当たりのキロバイト数の標準偏差 (過去 10 秒間の平均値)
std_dev_redo_speed_last_10sec	整数	Redo ログにディスクに書き込まれた 1 秒当たりのキロバイト数の標準偏差 (過去 10 秒間の平均値)

カラム名	型	説明
backup_lcp_speed_last_60sec	整数	バックアップおよび LCP プロセスによってディスクに書き込まれた 1 秒当たりのキロバイト数 (過去 60 秒間の平均値)
redo_speed_last_60sec	整数	Redo ログにディスクに書き込まれた 1 秒当たりのキロバイト数 (過去 10 秒間の平均値)
std_dev_backup_lcp_speed_last_60sec	整数	バックアップおよび LCP プロセスによってディスクに書き込まれた 1 秒当たりのキロバイト数の標準偏差 (過去 60 秒間の平均値)
std_dev_redo_speed_last_60sec	整数	Redo ログに書き込まれた 1 秒当たりのキロバイト数の標準偏差 (過去 60 秒間の平均値)
slowdowns_due_to_io_lag	整数	Redo ログの I/O ラグが原因でディスクの書き込み速度が低下した秒数
slowdowns_due_to_high_cpu	整数	高い CPU 使用率が原因でディスクの書き込み速度が低下した秒数
disk_write_speed_set_to_min	整数	ディスクの書き込み速度が最小に設定された秒数
current_target_disk_write_speed	整数	LDM スレッドごとの実際のディスク書き込み速度 (集計値)

disk_write_speed_aggregate テーブルは、MySQL Cluster NDB 7.4.1 で追加されました。

18.5.10.11 ndbinfo disk_write_speed_aggregate_node テーブル

disk_write_speed_aggregate_node テーブルは、LCP、バックアップ、およびリストア操作時のディスク書き込みの速度に関して集計された情報を示します。

次の表に、MySQL Cluster NDB 7.4.3 以降で実装された disk_write_speed_aggregate_node テーブルのカラムに関する情報を示します。MySQL Cluster NDB 7.4.2 以前の disk_write_speed_aggregate_node に関する情報は、このセクションの後半で見つけることができます。この表には、各カラムの名前、データ型、および簡単な説明を示します。追加情報については、表のあとに示す注記で見つけることができます。

カラム名	型	説明
node_id	整数	このノードのノード ID
backup_lcp_speed_last_sec	数値	過去 1 秒間にバックアップおよび LCP プロセスによってディスクに書き込まれたバイト数
redo_speed_last_sec	数値	過去 1 秒間に Redo ログに書き込まれたバイト数
backup_lcp_speed_last_10sec	数値	バックアップおよび LCP プロセスによってディスクに書き込まれた 1 秒当たりのバイト数 (過去 10 秒間の平均値)
redo_speed_last_10sec	数値	Redo ログにディスクに書き込まれた 1 秒当たりのバイト数 (過去 10 秒間の平均値)
backup_lcp_speed_last_60sec	数値	バックアップおよび LCP プロセスによってディスクに書き込まれた 1 秒当たりのバイト数 (過去 60 秒間の平均値)
redo_speed_last_60sec	数値	バックアップおよび LCP プロセスによってディスクに書き込まれた 1 秒当たりのバイト数 (過去 60 秒間の平均値)

MySQL Cluster NDB 7.4.3 よりも前では、このテーブルのカラムは、値がキロバイト単位で示されていました。MySQL Cluster NDB 7.4.3 以降では、カラムにはこのような値がすべてバイト単位で表示されます。(Bug #74317、Bug #19795072) 次の表に、MySQL Cluster NDB 7.4.2 以前に実装された disk_write_speed_aggregate に関する情報を示します。

カラム名	型	説明
node_id	整数	このノードのノード ID

カラム名	型	説明
backup_lcp_speed_last_sec	数値	過去 1 秒間にバックアップおよび LCP プロセスによってディスクに書き込まれたキロバイト数
redo_speed_last_sec	数値	過去 1 秒間に Redo ログにディスクに書き込まれたキロバイト数
backup_lcp_speed_last_10sec	数値	バックアップおよび LCP プロセスによってディスクに書き込まれた 1 秒当たりのキロバイト数 (過去 10 秒間の平均値)
redo_speed_last_10sec	数値	Redo ログにディスクに書き込まれた 1 秒当たりのキロバイト数 (過去 10 秒間の平均値)
backup_lcp_speed_last_60sec	数値	バックアップおよび LCP プロセスによってディスクに書き込まれた 1 秒当たりのキロバイト数 (過去 60 秒間の平均値)
redo_speed_last_60sec	数値	バックアップおよび LCP プロセスによってディスクに書き込まれた 1 秒当たりのキロバイト数 (過去 60 秒間の平均値)

disk_write_speed_aggregate_node テーブルは、MySQL Cluster NDB 7.4.1 で追加されました。

18.5.10.12 ndbinfo diskpagebuffer テーブル

diskpagebuffer テーブルは、MySQL Cluster ディスクデータテーブルによるディスクページバッファーの使用状況に関する統計を示します。

次の表に、diskpagebuffer テーブルのカラムに関する情報を示します。この表には、各カラムの名前、データ型、および簡単な説明を示します。追加情報については、表のあとに示す注記で見つけることができます。

カラム名	型	説明
node_id	整数	データノード ID
block_instance	整数	ブロックインスタンス
pages_written	整数	ディスクに書き込まれたページ数。
pages_written_lcp	整数	ローカルチェックポイントによって書き込まれたページ数。
pages_read	整数	ディスクから読み取られたページ数
log_waits	整数	ログがディスクに書き込まれるまで待機しているページ書き込みの数
page_requests_direct_return	整数	バッファー内で使用可能だったページに対するリクエストの数
page_requests_wait_queue	整数	ページがバッファー内で使用可能になるまで待機する必要があったリクエストの数
page_requests_wait_io	整数	ディスク上のページから読み取る必要があった (バッファー内のページを使用できなかった) リクエストの数

このテーブルを MySQL Cluster ディスクデータテーブルとともに使用すると、データをディスクからではなく、バッファーから読み取ることができるほど十分に DiskPageBufferMemory が大きいかどうかを確認できます。ディスクシークを最小にすることで、それらのテーブルのパフォーマンスの向上に役立つ可能性があります。

このようなクエリーを使用すると、読み取りの合計回数に対する DiskPageBufferMemory からの読み取りの比率を特定できます。この比率は、パーセンテージとして取得されます。

```
SELECT
  node_id,
  100 * page_requests_direct_return /
    (page_requests_direct_return + page_requests_wait_io)
      AS hit_ratio
FROM ndbinfo.diskpagebuffer;
```

このクエリーによる結果は、クラスタ内のデータノード (この例では、クラスタには 4 つのデータノードがあります) ごとに 1 行で、次に示すものと同様になります。

```
+---------+-----------+
| node_id | hit_ratio |
+---------+-----------+
|       5 |   97.6744 |
|       6 |   97.6879 |
|       7 |   98.1776 |
|       8 |   98.1343 |
+---------+-----------+
4 rows in set (0.00 sec)
```

hit_ratio の値が 100% に近付いていることは、バッファーからではなく、ディスクから行われている読み取りの数が非常にわずかしかないことを示します。つまり、ディスクデータの読み取りパフォーマンスが最適なレベルに近付いています。これらの値のいずれかが 95% 未満である場合は、config.ini ファイルで DiskPageBufferMemory の設定を大きくする必要があることを強く示しています。

> **注記**
>
> DiskPageBufferMemory の変更が有効になる前に、クラスタのすべてのデータノードのローリング再起動が必要です。

18.5.10.13 ndbinfo logbuffers テーブル

logbuffer テーブルは、MySQL Cluster ログバッファーの使用状況に関する情報を示します。

次の表に、logbuffers テーブルのカラムに関する情報を示します。この表には、各カラムの名前、データ型、および簡単な説明を示します。

カラム名	型	説明
node_id	整数	このデータノードの ID。
log_type	文字列	ログのタイプ。REDO または DD-UNDO のいずれか。
log_id	整数	ログ ID。
log_part	整数	ログパート番号。
total	整数	このログに使用可能な合計容量。
used	整数	このログによって使用された容量。

18.5.10.14 ndbinfo logspaces テーブル

このテーブルは、MySQL Cluster ログ容量の使用状況に関する情報を示します。

次の表に、logspaces テーブルのカラムに関する情報を示します。この表には、各カラムの名前、データ型、および簡単な説明を示します。

カラム名	型	説明
node_id	整数	このデータノードの ID。
log_type	文字列	ログのタイプ。REDO または DD-UNDO のいずれか。
log_id	整数	ログ ID。
log_part	整数	ログパート番号。
total	整数	このログに使用可能な合計容量。
used	整数	このログによって使用された容量。

18.5.10.15 ndbinfo membership テーブル

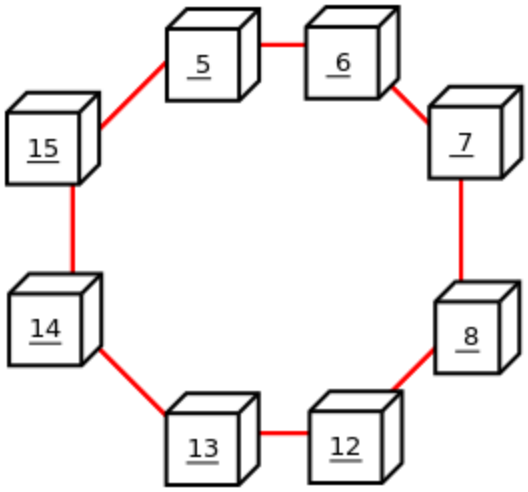
membership テーブルは、ノードグループのメンバーシップ、プレジデントノード、アービトレータ、アービトレータの後継ノード、アービトレータの接続状態、およびその他の情報を含む、各データノードがクラスタ内のほかのすべてのノードについて保持するビューを示します。

次の表に、membership テーブルのカラムに関する情報を示します。この表には、各カラムの名前、データ型、および簡単な説明を示します。追加情報については、表のあとに示す注記で見つけることができます。

カラム名	型	説明
node_id	整数	このノードのノード ID
group_id	整数	このノードが属するノードグループ
left node	整数	前のノードのノード ID
right_node	整数	次のノードのノード ID
president	整数	プレジデントのノード ID
successor	整数	プレジデントの後継ノードのノード ID
succession_order	整数	このノードがプレジデントの任務を継承する順序
Conf_HB_order	整数	-
arbitrator	整数	アービトレータのノード ID
arb_ticket	文字列	アービトレーションを追跡するために使用される内部識別子
arb_state	列挙 (テキストを参照)	アービトレーションの状態
arb_connected	Yes または No	このノードがアービトレータに接続されているかどうか
connected_rank1_arbs	ノード ID のリスト	接続されたランク 1 のアービトレータ
connected_rank2_arbs	ノード ID のリスト	接続されたランク 1 のアービトレータ

ノード ID およびノードグループ ID は、ndb_mgm -e "SHOW" でレポートされるものと同じです。

left_node および right_node は、次に示すように、時計文字板上の数字の順序と同様に、すべてのデータノードを、それらのノード ID の順序で円形に接続するモデルに関して定義されます。



この例では、円形に右回りの順序で、5、6、7、8、12、13、14、15 の番号が付けられた 8 つのデータノードがあります。「左」と「右」は、円の内側から判断します。ノード 5 の左側にあるノードはノード 15 であり、ノード 5 の右側にあるノードはノード 6 です。次のクエリーを実行し、その出力を調査することで、これらの関係をすべて確認できます。

```
mysql> SELECT node_id,left_node,right_node
    -> FROM ndbinfo.membership;
+---------+-----------+------------+
| node_id | left_node | right_node |
+---------+-----------+------------+
|       5 |        15 |          6 |
|       6 |         5 |          7 |
|       7 |         6 |          8 |
|       8 |         7 |         12 |
|      12 |         8 |         13 |
|      13 |        12 |         14 |
|      14 |        13 |         15 |
|      15 |        14 |          5 |
+---------+-----------+------------+
8 rows in set (0.00 sec)
```

「左」と「右」の指定は、イベントログでも同様に使用されます。

president ノードは、現在のノードから、アービトレータを設定する責任を持つものとして見られるノードです (NDB Cluster Start Phasesを参照してください)。プレジデントで障害が発生したり、切断されたりすると、現在のノードは、successor カラムに ID が示されたノードが新しいプレジデントになることを期待します。succession_order カラムは、現在のノードが自分自身で持っているとみなす後継キュー内の場所を示します。

通常の MySQL Cluster では、すべてのデータノードが、同じノードを president として、および同じノード (プレジデント以外) をその successor として見るべきです。さらに、現在のプレジデントは、自分自身を後継順序の 1 とみなし、successor ノードは、自分自身を 2 とみなす、というようにするべきです。

すべてのノードで同じ arb_state の値、および同じ arb_ticket の値を示すべきです。可能性のある arb_state の値は ARBIT_NULL、ARBIT_INIT、ARBIT_FIND、ARBIT_PREP1、ARBIT_PREP2、ARBIT_START、ARBIT_RUN、ARBIT_CH および UNKNOWN です。

arb_connected は、このノードの arbitrator として示されているノードに、このノードが接続されているかどうかを示します。

connected_rank1_arbs および connected_rank2_arbs の各カラムには、ArbitrationRank がそれぞれ 1 または 2 と等しい 0 個以上のアービトレータのリストが表示されます。

> **注記**
>
> 管理ノードと API ノードの両方に、アービトレータになる資格があります。

18.5.10.16 ndbinfo memoryusage テーブル

このテーブルをクエリーすると、ndb_mgm クライアントで ALL REPORT MemoryUsage コマンドによって提供されるものや、ALL DUMP 1000 によってログに記録されるものと同様の情報が提供されます。

次の表に、memoryusage テーブルのカラムに関する情報を示します。この表には、各カラムの名前、データ型、および簡単な説明を示します。追加情報については、表のあとに示す注記で見つけることができます。

カラム名	型	説明
node_id	整数	このデータノードのノード ID。
memory_type	文字列	Data memory または Index memory のいずれか、または (MySQL Cluster NDB 7.3.5 以降) Long message buffer。
used	整数	このデータノードでデータメモリーまたはインデックスメモリーに現在使用されているバイト数。
used_pages	整数	このデータノードでデータメモリーまたはインデックスメモリーに現在使用されているページ数 (テキストを参照してください)。
total	整数	このデータノードに使用可能なデータメモリーまたはインデックスメモリーの合計バイト数 (テキストを参照してください)。
total_pages	整数	このデータノード上のデータメモリーまたはインデックスメモリーに使用可能なメモリーページの合計数 (テキストを参照してください)。

total カラムは、特定のデータノード上の特定のリソース (データメモリーまたはインデックスメモリ) に使用可能なメモリーの合計量をバイト単位で表します。この数値は、config.ini ファイル内の対応する構成パラメータの設定にほぼ等しいです。

クラスタにノード ID 5 と 6 を持つ 2 つのデータノードがあり、config.ini ファイルに次が含まれると仮定します。

```
[ndbd default]
DataMemory = 1G
IndexMemory = 1G
```

さらに、LongMessageBuffer 構成パラメータの値をそのデフォルト (MySQL Cluster NDB 7.3.5 以降では 64M バイト) にすることが許可されていると仮定します。

次のクエリーでは、ほぼ同じ値が表示されます。

```
mysql> SELECT node_id, memory_type, total
    > FROM ndbinfo.memoryusage;
+---------+--------------------+------------+
| node_id | memory_type        | total      |
+---------+--------------------+------------+
|       5 | Data memory        | 1073741824 |
|       5 | Index memory       | 1074003968 |
|       5 | Long message buffer |   67108864 |
|       6 | Data memory        | 1073741824 |
|       6 | Index memory       | 1074003968 |
|       6 | Long message buffer |   67108864 |
+---------+--------------------+------------+
6 rows in set (0.00 sec)
```

この場合、インデックスメモリーの total カラムの値は、内部の丸め処理によって IndexMemory の値セットよりもわずかに大きくなります。

used_pages および total_pages カラムでは、リソースがページ単位で計測されます。これらのサイズは、DataMemory では 32K、IndexMemory では 8K です。長いメッセージバッファーメモリーの場合、ページサイズは 256 バイトです。

MySQL Cluster NDB 7.3.5 以降では、このテーブルに長いメッセージバッファーの情報が見つかることがあります。旧バージョンの MySQL Cluster では、データメモリーとインデックスメモリーのみが含まれていました。

18.5.10.17 ndbinfo memory_per_fragment テーブル

memory_per_fragment テーブルは、個々のフラグメントごとのメモリーの使用状況に関する情報を示します。

次の表に、memory_per_fragment テーブルのカラムに関する情報を示します。この表には、各カラムの名前、データ型、および簡単な説明を示します。追加情報については、表のあとに示す注記で見つけることができます。

カラム名	型	説明
fq_name	文字列	このフラグメントの名前
parent_fq_name	文字列	このフラグメントの親の名前
type	文字列	オブジェクトのタイプ (可能性のある値についてはテキストを参照してください)
table_id	整数	このテーブルのテーブル ID
node_id	整数	このノードのノード ID
block_instance	整数	カーネルブロックインスタンス ID
fragment_num	整数	フラグメント ID (番号)
fixed_elem_alloc_bytes	整数	固定サイズの要素に割り当てられたバイト数
fixed_elem_free_bytes	整数	固定サイズの要素に割り当てられたページ単位の残りの空きバイト数
fixed_elem_size_bytes	整数	固定サイズの各要素のバイト単位の長さ
fixed_elem_count	整数	固定サイズの要素の数
fixed_elem_free_rows	10 進数	固定サイズの要素用の空き行数
var_elem_alloc_bytes	整数	可変サイズの要素に割り当てられたバイト数
var_elem_free_bytes	整数	可変サイズの要素に割り当てられたページ単位の残りの空きバイト数
var_elem_count	整数	可変サイズの要素の数
hash_index_alloc_bytes	整数	ハッシュインデックスに割り当てられたバイト数

このテーブルの type カラムは、このフラグメントで使用されるディクショナリオブジェクトのタイプ (NDB API では Object::Type) を示し、次のリストに示す値のいずれかを取る可能性があります。

- System table

- User table
- Unique hash index
- Hash index
- Unique ordered index
- Ordered index
- Hash index trigger
- Subscription trigger
- Read only constraint
- Index trigger
- Reorganize trigger
- テーブルスペース
- Log file group
- データファイル
- Undo file
- Hash map
- Foreign key definition
- Foreign key parent trigger
- Foreign key child trigger
- Schema transaction

このリストは、mysql クライアントで SELECT * FROM ndbinfo.dict_obj_types を実行して取得することもできます。

このテーブルは MySQL Cluster NDB 7.4.1 で追加されました。

18.5.10.18 ndbinfo nodes テーブル

このテーブルには、データノードのステータスに関する情報が格納されます。このテーブルの対応する行には、クラスタ内で実行されているデータノードごとに、ノードのノード ID、ステータス、および稼働時間が示されます。起動中のノードごとに、現在の起動フェーズも示されます。

次の表に、nodes テーブルのカラムに関する情報を示します。この表には、各カラムの名前、データ型、および簡単な説明を示します。追加情報については、表のあとに示す注記で見つけることができます。

カラム名	型	説明
node_id	整数	クラスタ内のデータノードの一意のノード ID。
uptime	整数	ノードが最後に起動されてからの秒単位の時間。
status	文字列	データノードの現在のステータス。可能性のある値についてはテキストを参照してください。
start_phase	整数	データノードが起動中の場合、現在の起動フェーズ。
config_generation	整数	このデータノードで使用中のクラスタ構成ファイルのバージョン。

uptime カラムは、このノードが最後に起動または再起動されてから実行している時間を秒単位で示します。これは BIGINT 値です。この数値には、ノードを起動するために実際に必要な時間が含まれます。言い換えると、このカウンタは、ndbd または ndbmtd が最初に呼び出された瞬間に実行を開始します。したがって、ノードの起動が終了していない場合でも、uptime にゼロ以外の値が示される可能性があります。

status カラムは、ノードの現在のステータスを示します。これは、NOTHING、CMVMI、STARTING、STARTED、SINGLEUSER、STOPPING_1、STOPPING_2、STOPPING_3、

または STOPPING_4 のいずれかです。ステータスが STARTING の場合は、start_phase カラムで現在の起動フェーズを確認できます (このセクションの後半を参照してください)。クラスタがシングルユーザーモードになっているときは、すべてのデータノードの status カラムに SINGLEUSER が表示されます (セクション 18.5.8「MySQL Cluster のシングルユーザーモード」を参照してください)。STOPPING 状態のいずれかが表示されても、必ずしもノードがシャットダウン中であることを意味していませんが、それよりも新しい状態に移行していることを意味している可能性があります。たとえば、クラスタをシングルユーザーモードにしている場合、ステータスが SINGLEUSER に変更される前に、データノードの状態が一時的に STOPPING_2 としてレポートされることがあります。

start_phase カラムでは、ndb_mgm クライアントの node_id STATUS コマンドの出力で使用されるものと同じ範囲の値が使用されます (セクション18.5.2「MySQL Cluster 管理クライアントのコマンド」を参照してください)。ノードが現在起動中でない場合、このカラムは 0 を示します。MySQL Cluster の起動フェーズの説明を含むリストについては、セクション18.5.1「MySQL Cluster の起動フェーズのサマリー」を参照してください。

config_generation カラムは、各データノードで有効になっているクラスタ構成のバージョンを示します。これは、構成パラメータの変更を行うために、クラスタのローリング再起動を実行する際に役立つことがあります。たとえば、次の SELECT ステートメントの出力から、ノード 1、2、4 では最新バージョンのクラスタ構成 (6) が使用されているが、ノード 3 ではまだ使用されていないことを確認できます。

```
mysql> USE ndbinfo;
Database changed
mysql> SELECT * FROM nodes;
+---------+--------+---------+-------------+-------------------+
| node_id | uptime | status  | start_phase | config_generation |
+---------+--------+---------+-------------+-------------------+
|       1 |  10462 | STARTED |           0 |                 6 |
|       2 |  10460 | STARTED |           0 |                 6 |
|       3 |  10457 | STARTED |           0 |                 5 |
|       4 |  10455 | STARTED |           0 |                 6 |
+---------+--------+---------+-------------+-------------------+
2 rows in set (0.04 sec)
```

したがって、上記で示したケースでは、クラスタのローリング再起動を完了するために、ノード 3 を再起動するようにしてください。

このテーブルでは、停止されたノードは考慮されません。MySQL Cluster が 4 つのデータノード (ノード ID 1、2、3、4) を持ち、すべてのノードが正常に実行されていると仮定すると、このテーブルにはデータノードごとに 1 行ずつの 4 つの行 が格納されます。

```
mysql> USE ndbinfo;
Database changed
mysql> SELECT * FROM nodes;
+---------+--------+---------+-------------+-------------------+
| node_id | uptime | status  | start_phase | config_generation |
+---------+--------+---------+-------------+-------------------+
|       1 |  11776 | STARTED |           0 |                 6 |
|       2 |  11774 | STARTED |           0 |                 6 |
|       3 |  11771 | STARTED |           0 |                 6 |
|       4 |  11769 | STARTED |           0 |                 6 |
+---------+--------+---------+-------------+-------------------+
4 rows in set (0.04 sec)
```

ノードのいずれかをシャットダウンすると、この SELECT ステートメントの出力には、次に示すように、まだ実行中のノードのみが表示されます。

```
ndb_mgm> 2 STOP
Node 2: Node shutdown initiated
Node 2: Node shutdown completed.
Node 2 has shutdown.
```

```
mysql> SELECT * FROM nodes;
+---------+--------+---------+-------------+-------------------+
| node_id | uptime | status  | start_phase | config_generation |
+---------+--------+---------+-------------+-------------------+
|       1 |  11807 | STARTED |           0 |                 6 |
|       3 |  11802 | STARTED |           0 |                 6 |
|       4 |  11800 | STARTED |           0 |                 6 |
+---------+--------+---------+-------------+-------------------+
3 rows in set (0.02 sec)
```

18.5.10.19 ndbinfo resources テーブル

このテーブルは、データノードリソースの可用性および使用状況に関する情報を示します。

これらのリソースは、スーパープールと呼ばれることがあります。

次の表に、resources テーブルのカラムに関する情報を示します。この表には、各カラムの名前、データ型、および簡単な説明を示します。追加情報については、表のあとに示す注記で見つけることができます。

カラム名	型	説明
node_id	整数	このデータノードの一意のノード ID。
resource_name	文字列	リソースの名前。テキストを参照してください。
reserved	整数	このリソース用に予約された量。
used	整数	このリソースで実際に使用された量。
max	整数	ノードが最後に起動されてから使用されたこのリソースの最大量。

resource_name は、RESERVED、DISK_OPERATIONS、DISK_RECORDS、DATA_MEMORY、JOBBUFFER、FILE_BUFFERS、または TRANSPORTER_BUFFERS のいずれかになる可能性があります。

18.5.10.20 ndbinfo server_operations テーブル

server_operations テーブルは、現在の SQL ノード (MySQL サーバー) が現在関与している進行中のすべての NDB 操作を表すエントリを格納します。それは事実上、その他の SQL ノードおよび API ノードに対する操作が表示されない cluster_operations テーブルのサブセットです。

次の表に、server_operations テーブルのカラムに関する情報を示します。この表には、各カラムの名前、データ型、および簡単な説明を示します。追加情報については、表のあとに示す注記で見つけることができます。

カラム名	型	説明
mysql_connection_id	整数	MySQL Server 接続 ID
node_id	整数	ノード ID
block_instance	整数	ブロックインスタンス
transid	整数	トランザクション ID
operation_type	文字列	操作のタイプ (可能性のある値についてはテキストを参照)
state	文字列	操作の状態 (可能性のある値についてはテキストを参照)
tableid	整数	テーブル ID
fragmentid	整数	フラグメント ID
client_node_id	整数	クライアントノード ID
client_block_ref	整数	クライアントのブロック参照
tc_node_id	整数	トランザクションコーディネータノード ID
tc_block_no	整数	トランザクションコーディネータブロック番号
tc_block_instance	整数	トランザクションコーディネータブロックインスタンス

mysql_connection_id は、SHOW PROCESSLIST の出力に示された接続またはセッション ID と同じです。それは INFORMATION_SCHEMA のテーブル NDB_TRANSID_MYSQL_CONNECTION_MAP から取得されます。

トランザクション ID は、NDB API の getTransactionId() メソッドを使用して取得できる一意の 64 ビット数値です。(現在、MySQL サーバーは進行中のトランザクションの NDB API トランザクション ID を公開しません。)

operation_type カラムは、READ、READ-SH、READ-EX、INSERT、UPDATE、DELETE、WRITE、UNLOCK、REFRESH、SCAN、SCAN-SH、SCAN-EX、または <unknown> のいずれかの値を取ることができます。

state カラムは、ABORT_QUEUED、ABORT_STOPPED、COMMITTED、COMMIT_QUEUED、COMMIT_STOPPED、COPY_CLOSE または WAIT_TUP_TO_ABORT のいずれかの値を取ることができます。(ndbinfo_show_hidden を有効にして MySQL サーバーが実行されている場合は、通常は非表示になっている ndb$dblqh_tcconnect_state テーブルから選択することで、この状態のリストを表示できます。)

ndb_show_tables の出力をチェックして、テーブル ID から NDB テーブルの名前を取得できます。

fragid は、ndb_desc --extra-partition-info (短縮形式 -p) の出力で見られるパーティション番号と同じです。

client_node_id および client_block_ref の client は MySQL Cluster の API ノードまたは SQL ノード (つまり、クラスタに接続されている NDB API クライアントまたは MySQL サーバー) を表します。

18.5.10.21 ndbinfo server_transactions テーブル

server_transactions テーブルは、cluster_transactions テーブルのサブセットですが、このテーブルに関連する接続 ID が含まれている場合は、現在の SQL ノード (MySQL サーバー) が参加しているトランザクションのみが含まれます。

次の表に、server_transactions テーブルのカラムに関する情報を示します。この表には、各カラムの名前、データ型、および簡単な説明を示します。追加情報については、表のあとに示す注記で見つけることができます。

カラム名	型	説明
mysql_connection_id	整数	MySQL Server 接続 ID
node_id	整数	トランザクションコーディネータノード ID
block_instance	整数	トランザクションコーディネータブロックインスタンス
transid	整数	トランザクション ID
state	文字列	操作の状態 (可能性のある値についてはテキストを参照)
count_operations	整数	トランザクションでのステートフル操作の数
outstanding_operations	整数	ローカルデータ管理レイヤー (LQH ブロック) でまだ実行されている操作
inactive_seconds	整数	API の待機に要した時間
client_node_id	整数	クライアントノード ID
client_block_ref	整数	クライアントのブロック参照

mysql_connection_id は、SHOW PROCESSLIST の出力に示された接続またはセッション ID と同じです。それは INFORMATION_SCHEMA のテーブル NDB_TRANSID_MYSQL_CONNECTION_MAP から取得されます。

トランザクション ID は、NDB API の getTransactionId() メソッドを使用して取得できる一意の 64 ビット数値です。(現在、MySQL サーバーは進行中のトランザクションの NDB API トランザクション ID を公開しません。)

state カラム
は、CS_ABORTING、CS_COMMITTING、CS_COMMIT_SENT、CS_COMPLETE_SENT、CS_COMPLETING、CS_CONNEC
のいずれかの値を持つ可能性があります。(ndbinfo_show_hidden を有効にして MySQL サーバーが実行されている場合は、通常は非表示になっている ndb$dbtc_apiconnect_state テーブルから選択することで、この状態のリストを表示できます。)

client_node_id および client_block_ref の client は MySQL Cluster の API ノードまたは SQL ノード (つまり、クラスタに接続されている NDB API クライアントまたは MySQL サーバー) を表します。

18.5.10.22 ndbinfo threadblocks テーブル

threadblocks テーブルは、NDB カーネルブロックのデータノード、スレッド、およびインスタンスを関連付けます。

次の表に、threadblocks テーブルのカラムに関する情報を示します。この表には、各カラムの名前、データ型、および簡単な説明を示します。追加情報については、表のあとに示す注記で見つけることができます。

カラム名	型	説明
node_id	整数	ノード ID
thr_no	整数	スレッド ID
block_name	文字列	ブロック名
block_instance	整数	ブロックインスタンス番号

block_name は、ndbinfo.blocks テーブルから選択する場合に block_name カラムで見つかる値のいずれかです。特定の MySQL Cluster リリースで可能性のある値のリストは静的ですが、そのリストはリリース間で異なることがあります。

18.5.10.23 ndbinfo threadstat テーブル

threadstat テーブルは、NDB カーネルで実行されているスレッドの統計の概略のスナップショットを示します。

次の表に、threadstat テーブルのカラムに関する情報を示します。この表には、各カラムの名前、データ型、および簡単な説明を示します。追加情報については、表のあとに示す注記で見つけることができます。

カラム名	型	説明
node_id	整数	ノード ID
thr_no	整数	スレッド ID
thr_nm	文字列	スレッド名
c_loop	文字列	メインループ内のループ数
c_exec	文字列	実行されたシグナルの数
c_wait	文字列	追加入力を待機している回数
c_l_sent_prioa	整数	独自のノードに送信された優先度 A のシグナルの数
c_l_sent_priob	整数	独自のノードに送信された優先度 B のシグナルの数
c_r_sent_prioa	整数	リモートノードに送信された優先度 A のシグナルの数
c_r_sent_priob	整数	リモートノードに送信された優先度 B のシグナルの数
os_tid	整数	OS スレッド ID
os_now	整数	OS 時間 (ms)
os_ru_utime	整数	OS ユーザーの CPU 時間 (µs)
os_ru_stime	整数	OS システムの CPU 時間 (µs)
os_ru_minflt	整数	OS ページ再利用 (ソフトページフォルト)
os_ru_majflt	整数	OS ページフォルト (ハードページフォルト)
os_ru_nvcsw	整数	OS の自発的コンテキストスイッチ
os_ru_nivcsw	整数	OS の非自発的コンテキストスイッチ

os_time では、gettimeofday() システムコールが使用されます。

os_ru_utime、os_ru_stime、os_ru_minflt、os_ru_majflt、os_ru_nvcsw、および os_ru_nivcsw カラムの値は、getrusage() システムコールまたは同等のものを使用することで取得されます。

このテーブルには特定の時点で取得されたカウントが格納されるため、最適な結果を得るには、このテーブルでクエリーを定期的に実行し、その結果を 1 つまたは複数の中間テーブルに格納する必要があります。MySQL サーバーのイベントスケジューラを採用して、このようなモニタリングを自動化できます。詳細は、セクション 20.4「イベントスケジューラの使用」を参照してください。

18.5.10.24 ndbinfo transporters テーブル

このテーブルには、NDB トランスポータに関する情報が格納されます。

次の表に、transporters テーブルのカラムに関する情報を示します。この表には、各カラムの名前、データ型、および簡単な説明を示します。追加情報については、表のあとに示す注記で見つけることができます。

カラム名	型	説明
node_id	整数	クラスタ内のこのデータノードの一意のノード ID。
remote_node_id	整数	リモートデータノードのノード ID。
status	文字列	接続のステータス。

カラム名	型	説明
remote_address	文字列	リモートホストの名前または IP アドレス
bytes_sent	整数	この接続を使用して送信されたバイト数
bytes_received		この接続を使用して受信されたバイト数
connect_count		このトランスポータ上で確立された接続の回数
overloaded		このトランスポータが現在過負荷状態である場合は 1、それ以外の場合は 0
overload_count		このトランスポータが接続してから過負荷状態になった回数
slowdown		このトランスポータがスキャン速度低下状態になっている場合は 1、それ以外の場合は 0
slowdown_count		このトランスポータが接続してからスキャン速度低下状態になった回数

transporters テーブルには、クラスタ内で実行中のデータノードごとに、そのノードとクラスタ内のすべてのノード (自分自身を含む) との各接続のステータスを示す行が表示されます。この情報は、テーブルの status カラムに示されます。このカラムは、CONNECTING、CONNECTED、DISCONNECTING、または DISCONNECTED のいずれかの値になる可能性があります。

構成されているが、現在はクラスタに接続されていない API および管理ノードへの接続は、DISCONNECTED のステータスで示されます。node_id が現在接続されていないデータノードのものである場合、その行はこのテーブルに表示されません。(これは、ndbinfo.nodes テーブルの切断されたノードの省略に似ています。

remote_address は、remote_node_id カラムに ID が表示されないノードのホスト名またはアドレスです。このノードからの bytes_sent およびこのノードによる bytes_received はそれぞれ、接続が確立されてから、この接続を使用してノードによって送信されたバイト数と受信されたバイト数です。ステータスが CONNECTING または DISCONNECTED であるノードの場合、これらのカラムには常に 0 が表示されます。

connect_count、overloaded、overload_count、slowdown、および slowdown_count のカウンタは接続時にリセットされ、それらの値はリモートノードの切断後にも保持されます。bytes_send および bytes_received のカウンタも接続時にリセットされるため、それらの値は切断後にも (次の接続でリセットされるまで) 保持されます。

ndb_mgm クライアントの SHOW コマンドの出力に示されるように、2 つのデータノード、2 つの SQL ノード、および 1 つの管理ノードで構成される 5 ノードクラスタがあると仮定します。

```
ndb_mgm> SHOW
Connected to Management Server at: localhost:1186
Cluster Configuration
---------------------
[ndbd(NDB)]     2 node(s)
id=1    @10.100.10.1  (5.6.22-ndb-7.4.4, Nodegroup: 0, *)
id=2    @10.100.10.2  (5.6.22-ndb-7.4.4, Nodegroup: 0)

[ndb_mgmd(MGM)] 1 node(s)
id=10   @10.100.10.10  (5.6.22-ndb-7.4.4)

[mysqld(API)]   2 node(s)
id=20   @10.100.10.20  (5.6.22-ndb-7.4.4)
id=21   @10.100.10.21  (5.6.22-ndb-7.4.4)
```

すべてのデータノードが実行されていると仮定すると、次に示すように、transporters テーブルには 10 行 (1 番目のデータノード用に 5 行、2 番目のデータノード用に 5 行) が表示されます。

```
mysql> SELECT node_id, remote_node_id, status
    ->   FROM ndbinfo.transporters;
+---------+----------------+--------------+
| node_id | remote_node_id | status       |
+---------+----------------+--------------+
|       1 |              1 | DISCONNECTED |
|       1 |              2 | CONNECTED    |
|       1 |             10 | CONNECTED    |
|       1 |             20 | CONNECTED    |
|       1 |             21 | CONNECTED    |
|       2 |              1 | CONNECTED    |
|       2 |              2 | DISCONNECTED |
|       2 |             10 | CONNECTED    |
|       2 |             20 | CONNECTED    |
|       2 |             21 | CONNECTED    |
```

```
+---------+----------------+---------------+
10 rows in set (0.04 sec)
```

ndb_mgm クライアントで 2 STOP コマンドを使用して、このクラスタ内のデータノードのいずれかをシャットダウンしてから、(再度 mysql クライアントを使用して) 前のクエリーを繰り返すと、次に示すように、このテーブルには、5 行 (残りの管理ノードから別のノードへの接続 (そのデータノード自体と現在オフラインになっているデータノードの両方を含む) ごとに 1 行) のみが表示され、現在オフラインになっているデータノードへの残りの各接続のステータスを表す CONNECTING が表示されます。

```
mysql> SELECT node_id, remote_node_id, status
    ->   FROM ndbinfo.transporters;
+---------+----------------+---------------+
| node_id | remote_node_id | status        |
+---------+----------------+---------------+
|       1 |              1 | DISCONNECTED  |
|       1 |              2 | CONNECTING    |
|       1 |             10 | CONNECTED     |
|       1 |             20 | CONNECTED     |
|       1 |             21 | CONNECTED     |
+---------+----------------+---------------+
5 rows in set (0.02 sec)
```

18.5.11 MySQL Cluster のセキュリティー上の問題

このセクションでは、MySQL Cluster の設定時および実行時に考慮に入れるべきセキュリティー上の考慮事項について説明します。

このセクションで説明するトピックは、次のとおりです。

- MySQL Cluster とネットワークのセキュリティー上の問題
- MySQL Cluster のセキュアな実行に関する構成の問題
- MySQL Cluster と MySQL 権限システム
- MySQL Cluster に対する必要に応じた MySQL 標準のセキュリティー手順

18.5.11.1 MySQL Cluster のセキュリティーとネットワーク上の問題

このセクションでは、MySQL Cluster に関する基本的なネットワークセキュリティー上の問題について説明します。MySQL Cluster の「出荷時状態」はセキュアではないことを覚えておくことは、きわめて重要です。ユーザーやネットワーク管理者は、ネットワーク上のクラスタが危険にさらされる可能性がないように、適切なステップを実行する必要があります。

クラスタの通信プロトコルは本質的にセキュアではなく、クラスタ内のノード間の通信では暗号化や同様のセキュリティー対策が使用されません。ネットワークの速度および待機時間によって、クラスタの効率性が直接影響を受けるため、ノード間のネットワーク通信に SSL やその他の暗号化により、事実上通信速度が低下するので、そのようなスキームを採用することはお勧めしません。

MySQL Cluster への API ノードのアクセスを制御する際に、認証が使用されないことも事実です。暗号化と同様に、認証の要件を課すオーバーヘッドによって、クラスタのパフォーマンスが影響を受けることがあります。

さらに、クラスタへのアクセス時に、次のいずれに対するソース IP アドレスのチェックも実行されません。

- config.ini ファイル内の空の [mysqld] または [api] セクションによって作成される「空きスロット」を使用している SQL ノードまたは API ノード

  つまり、config.ini ファイル内に空の [mysqld] または [api] セクションがある場合、管理サーバーのホスト名 (または IP アドレス) とポートを認識しているどの API ノード (SQL ノードを含む) でも、制約なしでクラスタに接続し、そのデータにアクセスできます。(このことおよび関連する問題についての詳細は、セクション 18.5.11.2「MySQL Cluster と MySQL 権限」を参照してください。)

  > 注記
  >
  > config.ini ファイル内のすべての [mysqld] および [api] セクションに HostName パラメータを指定すると、クラスタへの SQL ノードおよび API ノードのアクセスに対していくらか制御できます。ただし、これは、これまで使用されていないホストからクラスタに API ノードを接続する場合、そのホスト名を含む [api] セクションを config.ini ファイルに追加する必要があることも意味します。

> 詳細は、HostName パラメータについてのこの章のほかの場所で説明しています。API ノードで HostName を使用する構成例について、セクション18.3.1「MySQL Cluster の簡易テストセットアップ」も参照してください。

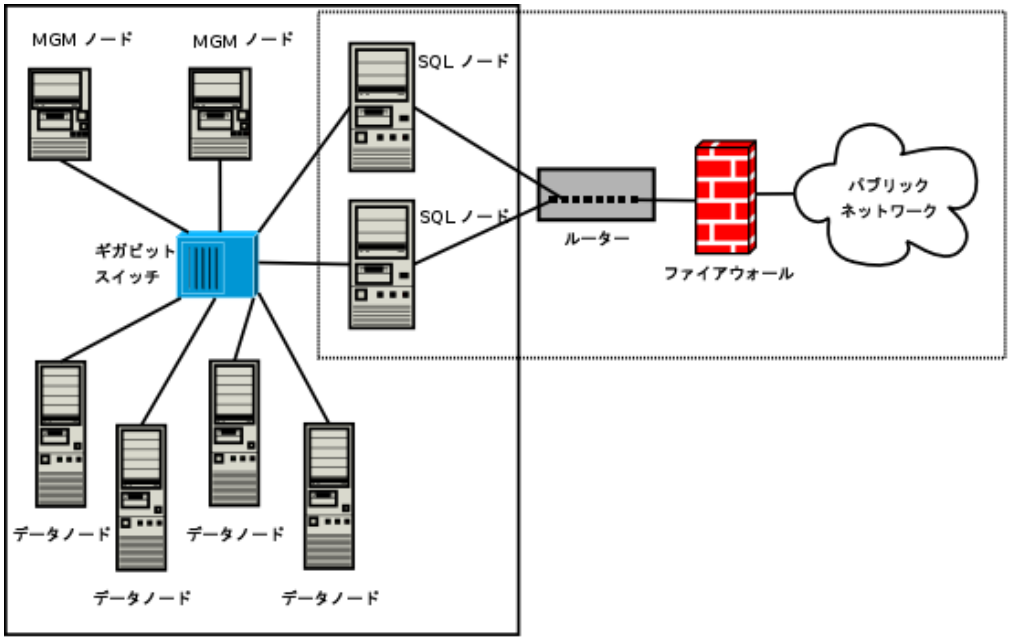
- 任意の ndb_mgm クライアント

  つまり、管理サーバーのホスト名 (または IP アドレス) とポート (標準ポートでない場合) が指定された任意のクラスタ管理クライアントがクラスタに接続し、任意の管理クライアントコマンドを実行できます。これには、ALL STOP や SHUTDOWN などのコマンドが含まれます。

このような理由により、ネットワークレベルでクラスタを保護する必要があります。もっとも安全なクラスタのネットワーク構成は、クラスタノード間の接続をその他のネットワーク通信から分離する構成です。これは、次の方法のいずれかによって実現できます。

1. どのパブリックネットワークからも物理的に分離されたネットワーク上にクラスタノードを保持します。このオプションは、信頼性がもっとも高いですが、実装するコストももっとも高くなります。

   次に、このように物理的に分離されたネットワークを使用した MySQL Cluster の設定例を示します。



   この設定には、クラスタ管理サーバーおよびデータノードの 1 つのプライベート (実線の四角形) と、SQL ノードが存在する 1 つのパブリック (点線の四角形) の 2 つのネットワークがあります。(ギガビットスイッチは最高のパフォーマンスが実現されるため、これを使用して接続された管理ノードとデータノードを示しています。)どちらのネットワークも、ハードウェアファイアウォール (ネットワークベースのファイアウォールと呼ばれることもあります) によって外部から保護されています。

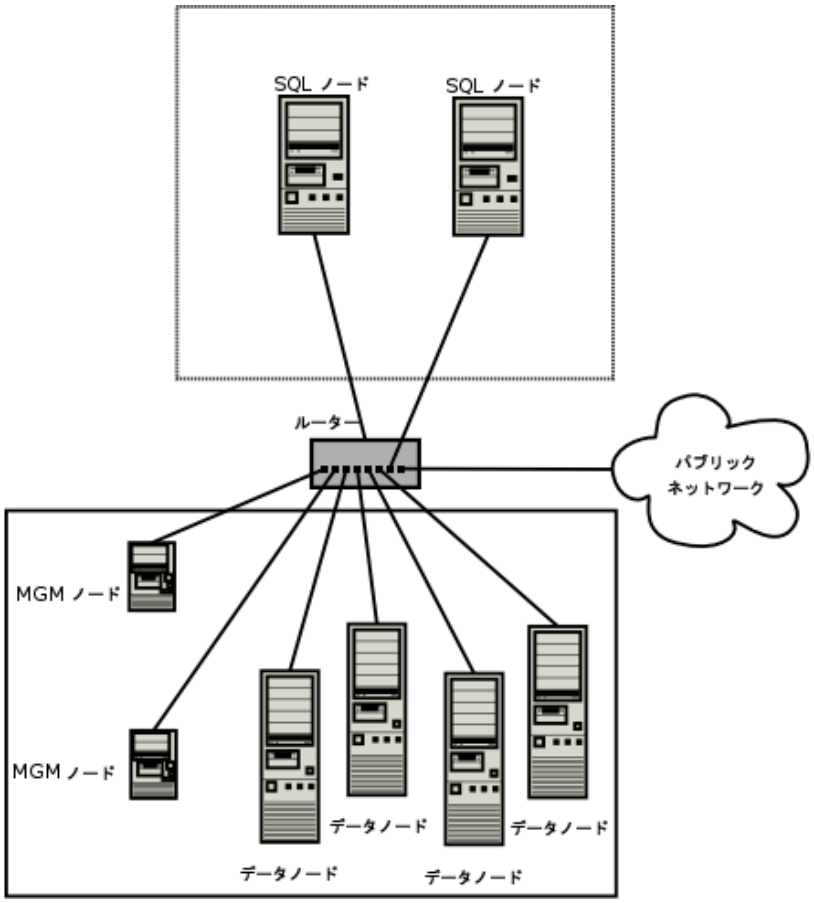
   SQL ノードでパケットの転送が許可されていなければ、パケットは SQL ノードを通過せずに、ネットワーク外部からクラスタの管理ノードまたはデータノードに到達できない (どのクラスタの内部通信も外部に到達できない) ため、このネットワーク設定がもっとも安全です。これは、当然、すべての SQL ノードをハッキングの試みに対して、セキュリティー保護する必要があることを意味します。

   > **重要**
   >
   > 潜在的なセキュリティーの脆弱性に関しては、SQL ノードとその他の MySQL サーバーとの間に相違はありません。MySQL サーバーをセキュリティー保護するために使用できる技法については、セクション6.1.3「攻撃者に対する MySQL のセキュアな状態の維持」を参照してください。

2. 1 つ以上のソフトウェアファイアウォール (ホストベースのファイアウォールと呼ばれることもあります) を使用して、アクセスする必要のないネットワークの部分からクラスタまで通過するパケットを制御できます。このタイプの設定では、それがなければローカルネットワークの外部からアクセスできてしまう可能性のあるクラスタ内のすべてのホストに、ソフトウェアファイアウォールをインストールする必要があります。

ホストベースのオプションは、実装するコストがもっとも低いですが、保護を提供するソフトウェアに完全に依存しているため、セキュアな状態を維持することがもっとも困難です。

次に、MySQL Cluster に対するこのタイプのネットワーク設定の例を示します。



このタイプのネットワーク設定を使用することは、MySQL Cluster ホストの 2 つのゾーンが存在することを意味します。各クラスタホストは、クラスタ内のその他のすべてのマシンと通信できる必要がありますが、SQL ノードをホストしているマシン (点線の四角形) のみ、外部との接続を許可でき、データノードおよび管理ノードを含むゾーン内のマシン (実線の四角形) は、クラスタに含まれないすべてのマシンから分離する必要があります。クラスタを使用するアプリケーションとそれらのアプリケーションのユーザーには、管理ノードおよびデータノードホストへの直接アクセスを許可しないでください。

これを実現するには、各クラスタホストコンピュータ上で実行されているノードのタイプに応じて、次の表に示す 1 つまたは複数のタイプにトラフィックを制限するソフトウェアファイアウォールを設定する必要があります。

アクセスされるノードのタイプ	許可するトラフィック
SQL ノードまたは API ノード	• それは、管理ノードまたはデータノードの IP アドレスから (任意の TCP または UDP ポートを使用して) 発信されています。 • それは、クラスタが存在し、アプリケーションで使用されているポート上にあるネットワーク内から発信されています。
データノードまたは管理ノード	• それは、管理ノードまたはデータノードの IP アドレスから (任意の TCP または UDP ポートを使用して) 発信されています。 • それは、SQL ノードまたは API ノードの IP アドレスから発信されています。

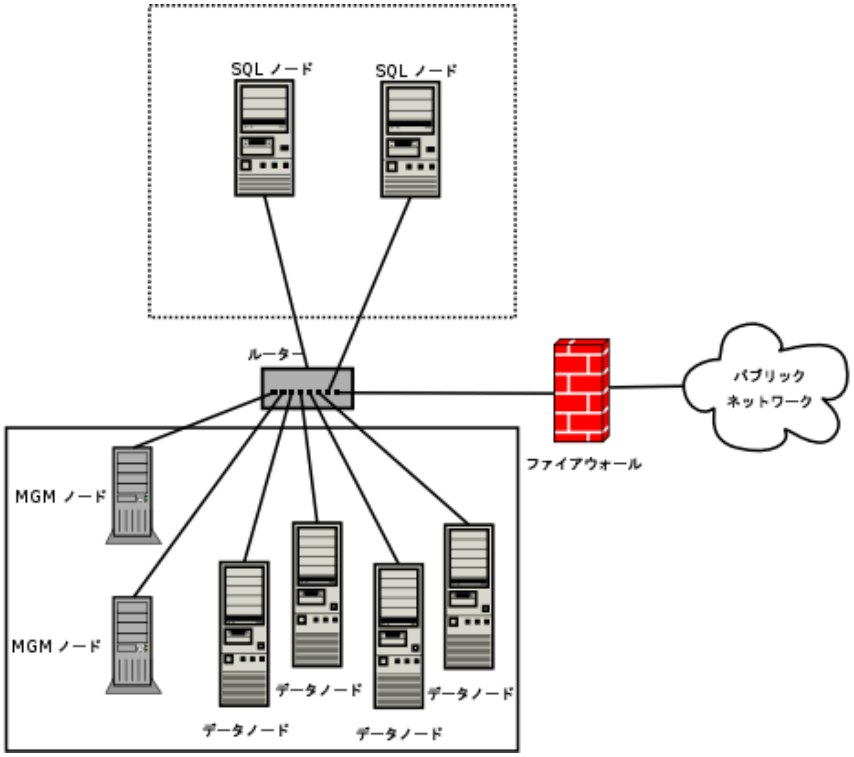
特定のノードタイプの、表に示した以外のトラフィックはすべて拒否されるべきです。

ファイアウォール構成の仕様は、ファイアウォールアプリケーションごとに異なるため、このマニュアルのスコープ外です。iptables は非常に一般的で、信頼性の高いファイアウォールアプリケーションであるため、多

くの場合に構成をより簡単にするためにフロントエンドとして APF とともに使用されます。このタイプまたは次の項目で説明する「混在」タイプの MySQL Cluster ネットワーク設定を実装するように選択した場合、採用するソフトウェアファイアウォールについて、ドキュメントを参照できます (するべきです)。

3. ハードウェアとソフトウェアの両方を使用して (つまり、ネットワークベースとホストベースの両方のファイアウォールを使用して) クラスタをセキュリティー保護することで、最初の 2 つの方法を組み合わせたものを採用することもできます。これは、セキュリティーレベルとコストの両方の点で、最初の 2 つのスキームの中間に位置するものです。このタイプのネットワーク設定では、クラスタがハードウェアファイアウォールの背後に配置されますが、受信パケットは SQL ノードに到達するために、すべてのクラスタホストに接続しているルータを通過することが許可されます。

   次に、ハードウェアとソフトウェアの両方のファイアウォールを組み合わせて使用することで実現できる MySQL Cluster クラスタのネットワーク配備の一例を示します。



   この場合、SQL ノードおよび API ノードへのトラフィックを除くすべての外部トラフィックを拒否し、アプリケーションで必要とするポート上でのみそれらのノードへのトラフィックを許可するように、ハードウェアファイアウォールにルールを設定できます。

どのネットワーク構成を使用する場合でも、クラスタのセキュリティーを維持するという観点からの目標は同じであることは忘れないでください。つまり、クラスタ内のノード間でもっとも効率的な通信を確保しながら、不要なトラフィックがクラスタに到達することを妨げます。

MySQL Cluster では、ノード間の通信を行うために多数のポートを開く必要があるため、推奨されるオプションは、分離されたネットワークを使用することです。これは、不要なトラフィックがクラスタに到達することを防ぐためのもっとも簡単な方法です。

> 注記
>
> リモートで (つまり、ローカルネットワーク外部から) MySQL Cluster を管理する場合、これを実行する推奨される方法は、ssh や別のセキュアなログインシェルを使用して、SQL ノードホストにアクセスすることです。これにより、このホストから管理クライアントを実行して、クラスタ独自のローカルネットワーク内から管理サーバーに安全にアクセスできます。
>
> ndb_mgm を使用して、クラスタが実行されているローカルネットワーク外部から直接クラスタを管理することは理論上可能ですが、推奨されていません。管理クライアントと管理サーバー間では認証も暗号化も行われないため、これはクラスタを管理するきわめてセキュアでない方法であり、遅かれ早かれ、ほぼ確実に危険にさらされます。

18.5.11.2 MySQL Cluster と MySQL 権限

このセクションでは、MySQL Cluster に関連する MySQL 権限システムのしくみ、および MySQL Cluster のセキュリティーを維持するためのこの関わりについて説明します。

MySQL Cluster のテーブルには、標準の MySQL 権限が適用されます。これには、データベース、テーブル、およびカラムのレベルで付与されるすべての MySQL 権限タイプ (SELECT 権限、UPDATE 権限、DELETE 権限など) が含まれます。その他の MySQL サーバーの場合と同様に、ユーザーおよび権限の情報は mysql システムデータベースに格納されます。NDB テーブル、このようなテーブルを含むデータベース、およびこのようなテーブル内のカラムに対する権限を付与および取り消すために使用される SQL ステートメントは、(その他の) MySQL ストレージエンジンを含むデータベースオブジェクトとの接続に使用される GRANT および REVOKE ステートメントとすべての点で同じです。CREATE USER および DROP USER ステートメントについても、同じことが当てはまります。

MySQL 付与テーブルは、デフォルトで MyISAM ストレージエンジンを使用することに留意しておくことが重要です。そのため、MySQL Cluster の SQL ノードとして動作している MySQL サーバー間で、通常これらのテーブルの複製や共有は行われません。言い換えると、デフォルトでは、ユーザーおよびそれらの権限を変更しても自動的に SQL ノード間に反映されません。必要に応じて、MySQL Cluster の SQL ノード間で MySQL ユーザーおよび権限の自動配布を有効化できます。詳細は、セクション18.5.14「MySQL Cluster の配布された MySQL 権限」を参照してください。

逆に、MySQL には権限を拒否する方法がないため (そもそも権限を取り消すことや、付与しないことはできますが、それ自体を拒否することはできません)、ある SQL ノード上の NDB テーブルを別の SQL ノードに対する権限を持つユーザーから保護する特別な手段がありません。このことは、ユーザー権限の自動配布を使用していない場合でも当てはまります。これを示す明確な例として、任意のデータベースオブジェクトに対して任意のアクションを実行できる MySQL の root アカウントが挙げられます。このアカウントを config.ini ファイルの空の [mysqld] または [api] セクションと組み合わせると、著しく危険になる可能性があります。その理由を理解するために、次のようなシナリオを検討します。

- config.ini ファイルには、少なくとも 1 つの空の [mysqld] または [api] セクションが含まれています。つまり、MySQL Cluster 管理サーバーで、MySQL サーバー (またはその他の API ノード) から MySQL Cluster へのアクセス元のホストのチェックが実行されません。
- ファイアウォールが存在しないか、またはファイアウォールがネットワーク外部にあるホストから MySQL Cluster へのアクセスに対する保護に失敗します。
- MySQL Cluster の管理サーバーのホスト名または IP アドレスは既知であるか、またはネットワーク外部から特定できます。

これらの条件に当てはまる場合は、だれでも、どこからでも --ndbcluster --ndb-connectstring=management_host を付けて MySQL Server を起動し、この MySQL Cluster にアクセスできます。MySQL root アカウントを使用すると、このユーザーは次のアクションを実行できます。

- SHOW DATABASES ステートメント (サーバー上のすべての NDB データベースのリストを取得する) または SHOW TABLES FROM some_ndb_database ステートメントなどのメタデータステートメントを実行して、特定のデータベース内のすべての NDB テーブルのリストを取得します。
- 検出されたテーブルのいずれかに対して、次のような正当な MySQL ステートメントを実行します。
  - 任意のテーブルからすべてのデータを読み取る SELECT * FROM some_table
  - テーブルからすべてのデータを削除する DELETE FROM some_table
  - テーブルスキーマを確認する DESCRIBE some_table または SHOW CREATE TABLE some_table
  - テーブルカラムに「不正」データを入力する UPDATE some_table SET column1 = some_value。これは実際に、単にすべてのデータを削除するよりも、はるかに大きな損害が発生する可能性があります

    より狡猾なバリエーションには、次のようなステートメントが含まれることがあります。

    ```
    UPDATE some_table SET an_int_column = an_int_column + 1
    ```

    または

    ```
    UPDATE some_table SET a_varchar_column = REVERSE(a_varchar_column)
    ```

    このような悪意のあるステートメントは、攻撃者の想像力でしか制限されません。

このような種類の攻撃を受けるおそれのない唯一のテーブルは、NDB 以外のストレージエンジンを使用して作成され、そのために「不正な」 SQL ノードから見えないテーブルです。

また、root としてログインできるユーザーは INFORMATION_SCHEMA データベースおよびそのテーブルにアクセスすることもできるため、データベース、テーブル、ストアドルーチン、スケジュール済みのイベント、およびメタデータが INFORMATION_SCHEMA に格納されているその他のデータベースオブジェクトに関する情報を取得できます。

また、配布された権限を使用していなければ、MySQL Cluster SQL ノードごとに異なる root アカウントのパスワードを使用することも非常に適切な方法です。

要するに、MySQL Cluster がローカルネットワーク外部から直接アクセスできる場合は、安全性を確保できません。

> **重要**
>
> MySQL の root アカウントのパスワードは空のままにしないでください。これは、MySQL を、MySQL Cluster SQL ノードとして実行している場合も、スタンドアロン (非クラスタ) MySQL サーバーとして実行している場合もまったく同じように当てはまり、MySQL インストールプロセスの一部として、MySQL サーバーを MySQL Cluster の SQL ノードとして構成する前に実行するようにしてください。

MySQL Cluster の権限の配布機能を使用する場合は、単に NDB ストレージエンジンを使用するように、mysql データベース内のシステムテーブルを手動で変換しないでください。代わりに、この目的のために提供されているストアドプロシージャーを使用してください。セクション18.5.14「MySQL Cluster の配布された MySQL 権限」を参照してください。

それ以外の場合、SQL ノード間で mysql システムテーブルを同期する必要がある場合は、標準の MySQL レプリケーションを使用してこれを実行することも、スクリプトを使用して MySQL サーバー間でテーブルエントリをコピーすることもできます。

サマリー　次に、MySQL Cluster に関して、MySQL 権限システムについて覚えておくべきもっとも重要な点を示します。

1. ある SQL ノード上で確立されたユーザーおよび権限は、クラスタ内のその他の SQL ノードで自動的に存在することも、有効になることもありません。逆に、クラスタ内のある SQL ノード上のユーザーまたは権限を削除しても、その他の SQL ノードからユーザーまたは権限は削除されません。

2. SQL スクリプトおよびそれに含まれる、MySQL Cluster 配布でこの目的で提供されているストアドプロシージャーを使用して、複数の MySQL ノード間で SQL ユーザーおよび権限を配布できます。

3. MySQL Cluster 内のある SQL ノードから、MySQL ユーザーに NDB テーブルに対する権限が付与されると、権限配布を使用していない場合でも、そのユーザーは、データの生成元の SQL ノードに関係なく、そのテーブル内のどのデータも「参照」できます。

18.5.11.3 MySQL Cluster と MySQL のセキュリティー手順

このセクションでは、実行中の MySQL Cluster に適用される MySQL 標準セキュリティー手順について説明します。

一般に、MySQL をセキュアに実行するための標準手順は、MySQL Cluster の一部として実行している MySQL サーバーにも適用されます。なによりもまず、常に MySQL サーバーを mysql システムユーザーとして実行するようにしてください。これは、標準 (非クラスタ) 環境で実行している MySQL と違いはありません。mysql システムアカウントは、一意かつ明確に定義されるべきです。幸運にも、これは新しい MySQL インストールのデフォルトの動作です。次に示すようなシステムコマンドを使用すると、mysqld プロセスがシステムユーザー mysql として実行されていることを確認できます。

```
shell> ps aux | grep mysql
root     10467  0.0  0.1   3616  1380 pts/3    S    11:53   0:00 \
  /bin/sh ./mysqld_safe --ndbcluster --ndb-connectstring=localhost:1186
mysql    10512  0.2  2.5  58528 26636 pts/3    Sl   11:53   0:00 \
  /usr/local/mysql/libexec/mysqld --basedir=/usr/local/mysql \
  --datadir=/usr/local/mysql/var --user=mysql --ndbcluster \
  --ndb-connectstring=localhost:1186 --pid-file=/usr/local/mysql/var/mothra.pid \
  --log-error=/usr/local/mysql/var/mothra.err
jon      10579  0.0  0.0   2736   688 pts/0    S+   11:54   0:00 grep mysql
```

mysqld プロセスが mysql 以外のユーザーとして実行されている場合は、それをすぐにシャットダウンしてから、mysql ユーザーとして再起動するようにしてください。このユーザーがシステム上に存在しない場合は、mysql ユーザーアカウントを作成して、このユーザーを mysql ユーザーグループに属するようにしてください。この場合、このシステム上の (mysqld の --datadir オプションを使用して設定された) MySQL データディレク

トリが mysql ユーザーによって所有されていること、および SQL ノードの my.cnf ファイルの [mysqld] セクションに user=mysql が含まれていることも確認するようにしてください。また、コマンド行で --user=mysql を付けて MySQL サーバープロセスを起動できますが、コマンド行オプションを使用することを忘れたために、意図せずに mysqld が別のユーザーとして実行されている可能性もあるため、my.cnf オプションを使用することをお勧めします。mysqld_safe 起動スクリプトを使用すると、MySQL が強制的に mysql ユーザーとして実行されます。

> **重要**
>
> 絶対に mysqld をシステムの root ユーザーとして実行しないでください。これを行うと、システム上の任意のファイルが MySQL によって読み取られる可能性があるため、攻撃者によって MySQL が危険にさらされます。

前のセクションで説明したように (セクション18.5.11.2「MySQL Cluster と MySQL 権限」を参照してください)、常に、MySQL サーバーを実行したらすぐに、root パスワードを設定するようにしてください。また、デフォルトでインストールされている匿名ユーザーアカウントを削除するようにしてください。次のステートメントを使用すると、これらのタスクを実現できます。

```
shell> mysql -u root

mysql> UPDATE mysql.user
    ->     SET Password=PASSWORD('secure_password')
    ->     WHERE User='root';

mysql> DELETE FROM mysql.user
    ->     WHERE User='';

mysql> FLUSH PRIVILEGES;
```

DELETE ステートメントを実行する際は、WHERE 句を省略しないように注意してください。そうしないと、すべての MySQL ユーザーが削除される危険性があります。mysql.user テーブルを変更したら、その変更が即座に有効になるように、すぐに FLUSH PRIVILEGES ステートメントを実行してください。FLUSH PRIVILEGES を実行しなければ、次回サーバーを再起動するまで、変更が有効になりません。

> **注記**
>
> ndb_show_tables、ndb_desc、ndb_select_all などの多くの MySQL Cluster ユーティリティーは、認証なしでも動作し、テーブル名、スキーマ、およびデータを表示できます。デフォルトで、これらは Unix スタイルのシステムにアクセス権 wxr-xr-x (755) でインストールされます。これは、mysql/bin ディレクトリにアクセスできる任意のユーザーが実行できることを意味します。
>
> これらのユーティリティーについての詳細は、セクション18.4「MySQL Cluster プログラム」を参照してください。

18.5.12 MySQL Cluster ディスクデータテーブル

NDB テーブルのインデックスなしカラムは RAM 内ではなく、ディスク上に格納できます。

MySQL Cluster ディスクデータ動作の実装の一部として、MySQL Cluster では、ノードのリカバリ時および再起動時に、非常に大量 (数テラバイト) のデータを効率的に処理できるように、多くの点が改善されました。これらには、起動ノードを非常に大きなデータセットと同期するための「no-steal」アルゴリズムが含まれます。詳細は、MySQL Cluster の開発者 Mikael Ronström と Jonas Oreland 共著のペーパー『Recovery Principles of MySQL Cluster 5.1』を参照してください。

いくつかの構成パラメータによって、MySQL Cluster ディスクデータのパフォーマンスが影響を受ける可能性があります。これらのパラメータとその効果については、MySQL Cluster ディスクデータの構成パラメータおよび MySQL Cluster ディスクデータストレージとGCP Stopエラーを参照してください。

また、データノードファイルシステムを Undo ログファイルおよびテーブルスペースデータファイルから分離すると (これはシンボリックリンクを使用することで実行できます)、ディスクデータストレージを使用する MySQL Cluster のパフォーマンスも大幅に改善できます。詳細は、セクション18.5.12.2「ディスクデータオブジェクトでのシンボリックリンクの使用」を参照してください。

18.5.12.1 MySQL Cluster ディスクデータオブジェクト

MySQL Cluster ディスクデータストレージは、いくつかのディスクデータオブジェクトを使用して実装されます。これらには、次のものが含まれます。

- テーブルスペースは、その他のディスクデータオブジェクト用のコンテナとして機能します。

- Undo ログファイルは、トランザクションをロールバックするために必要な情報を元に戻します。
- 1 つ以上の Undo ログファイルがログファイルグループに割り当てられ、ログファイルグループはテーブルスペースに割り当てられます。
- データファイルには、ディスクデータテーブルデータが格納されます。データファイルは、テーブルスペースに直接割り当てられます。

Undo ログファイルおよびデータファイルは、各データノードのファイルシステム内にある実際のファイルです。デフォルトでは、MySQL Cluster config.ini ファイルに指定された DataDir の ndb_node_id_fs に配置されます。ここで、node_id はデータノードのノード ID です。Undo ログファイルまたはデータファイルの作成時に、ファイル名の一部として絶対パスまたは相対パスを指定すると、これらを別の場所に配置できます。これらのファイルを作成するステートメントは、このセクションの後半で示します。

MySQL Cluster のテーブルスペースおよびログファイルグループは、ファイルとして実装されていません。

> **重要**
>
> すべてのディスクデータオブジェクトがファイルとして実装されるとはかぎりませんが、すべてが同じ名前空間を共有します。つまり、各ディスクデータオブジェクトは (単に、特定の型の各ディスクデータオブジェクトというだけでなく)、一意の名前が付けられている必要があります。たとえば、テーブルスペースとログファイルグループの両方に dd1 という名前を付けることはできません。

すべてのノード (管理ノードと SQL ノードを含む) を含む MySQL Cluster をすでに作成していると仮定すると、ディスク上に MySQL Cluster テーブルを作成するための基本ステップは次のとおりです。

1. ログファイルグループを作成し、それに 1 つ以上の Undo ログファイルを割り当てます (Undo ログファイルは、undofile と呼ばれることもあります)。

   > **注記**
   >
   > Undo ログファイルは、ディスクデータテーブルでのみ必要です。メモリー内にのみ格納される NDBCLUSTER テーブルでは使用されません。

2. テーブルスペースを作成し、そのテーブルスペースにログファイルグループおよび 1 つ以上のデータファイルを割り当てます。
3. データストレージ用に、このテーブルスペースを使用するディスクデータテーブルを作成します。

これらの各タスクは、次の例に示すように、mysql クライアントまたはその他の MySQL クライアントアプリケーションで SQL ステートメントを使用することで実現できます。

1. CREATE LOGFILE GROUP を使用して、lg_1 という名前のログファイルグループを作成します。このログファイルグループは、undo_1.log および undo_2.log という名前の 2 つの Undo ログファイルで構成されます。初期サイズは、それぞれ 16M バイトと 12M バイトです。(Undo ログファイルのデフォルト初期サイズは 128M バイトです。)オプションで、ログファイルグループの Undo バッファーのサイズを指定したり、デフォルト値の 8M バイトになることを許可したりすることもできます。この例では、Undo バッファーのサイズを 2 M バイトに設定しています。ログファイルグループは、Undo ログファイルとともに作成する必要があります。そのため、次の CREATE LOGFILE GROUP ステートメントで undo_1.log を lg_1 に追加しています。

   ```
   CREATE LOGFILE GROUP lg_1
     ADD UNDOFILE 'undo_1.log'
     INITIAL_SIZE 16M
     UNDO_BUFFER_SIZE 2M
     ENGINE NDBCLUSTER;
   ```

   ログファイルグループに undo_2.log を追加するには、次の ALTER LOGFILE GROUP ステートメントを使用します。

   ```
   ALTER LOGFILE GROUP lg_1
     ADD UNDOFILE 'undo_2.log'
     INITIAL_SIZE 12M
     ENGINE NDBCLUSTER;
   ```

   いくつかの注意事項:

   - ここで使用される .log ファイル拡張子は必須ではありません。これは単に、ログファイルを簡単に認識できるようにするために使用しています。

- すべての CREATE LOGFILE GROUP および ALTER LOGFILE GROUP ステートメントには ENGINE 句を含める必要があります。MySQL Cluster NDB 7.3 以降では、この句に許可される値は NDBCLUSTER と NDB だけです。

  > **重要**
  >
  > 特定の時点で同じ MySQL Cluster には、最大でも 1 つのログファイルグループしか存在できません。

- ADD UNDOFILE 'filename' を使用して、ログファイルグループに Undo ログファイルを追加すると、クラスタ内の各データノードの DataDir 内にある ndb_node_id_fs ディレクトリに、filename という名前のファイルが作成されます。ここで、node_id はデータノードのノード ID です。各 Undo ログファイルのサイズは、SQL ステートメントで指定されます。たとえば、MySQL Cluster に 4 つのデータノードが含まれている場合、ここで示した ALTER LOGFILE GROUP ステートメントは 4 つの Undo ログファイル (4 つの各データノードのデータディレクトリに 1 つずつ) を作成します。これらの各ファイルの名前は undo_2.log であり、各ファイルのサイズは 12M バイトです。

- UNDO_BUFFER_SIZE は、使用可能なシステムメモリーの量によって制限されます。

- CREATE LOGFILE GROUP ステートメントの詳細は、セクション13.1.14「CREATE LOGFILE GROUP 構文」を参照してください。ALTER LOGFILE GROUP の詳細は、セクション13.1.3「ALTER LOGFILE GROUP 構文」を参照してください。

2. この時点で、データを格納するために MySQL Cluster ディスクデータテーブルで使用されるファイルを格納するテーブルスペースを作成できます。テーブルスペースは、特定のログファイルのグループにも関連付けられます。新しいテーブルスペースを作成する際は、Undo ログギングで使用されるログファイルグループを指定する必要があります。データファイルも指定する必要があります。テーブルスペースが作成されたら、追加のデータファイルをテーブルスペースに追加できます。テーブルスペースからデータファイルを削除することもできます (データファイルを削除する例については、このセクションの後半で説明しています)。

   ログファイルグループとして lg_1 を使用する ts_1 という名前のテーブルスペースを作成すると仮定します。このテーブルスペースには、data_1.dat および data_2.dat という名前で、初期サイズがそれぞれ 32 M バイトおよび 48M バイトの 2 つのデータファイルが格納されます。(INITIAL_SIZE のデフォルトの値は 128M バイトです。)これは、次に示すように、2 つの SQL ステートメントを使用することで実行できます。

   ```
   CREATE TABLESPACE ts_1
      ADD DATAFILE 'data_1.dat'
      USE LOGFILE GROUP lg_1
      INITIAL_SIZE 32M
      ENGINE NDBCLUSTER;

   ALTER TABLESPACE ts_1
      ADD DATAFILE 'data_2.dat'
      INITIAL_SIZE 48M
      ENGINE NDBCLUSTER;
   ```

   CREATE TABLESPACE ステートメントは、データファイル data_1.dat を含むテーブルスペース ts_1 を作成し、ts_1 をログファイルグループ lg_1 に関連付けます。ALTER TABLESPACE は、2 つめのデータファイル (data_2.dat) を追加します。

   いくつかの注意事項:

   - Undo ログファイルの例で使用した .log ファイル拡張子の場合と同様に、.dat ファイル拡張子にも特別な意味はありません。単に、データファイルを簡単に認識できるように使用されています。

   - ADD DATAFILE 'filename' を使用して、テーブルスペースにデータファイルを追加すると、クラスタ内の各データノードの DataDir 内にある ndb_node_id_fs ディレクトリに、filename という名前のファイルが作成されます。ここで、node_id はデータノードのノード ID です。各データファイルのサイズは、SQL ステートメントで指定します。たとえば、MySQL Cluster に 4 つのデータノードが含まれている場合、ここで示した ALTER TABLESPACE ステートメントは、4 つの各データノードのデータディレクトリに 1 つずつ、4 つのデータファイルを作成します。これらの各ファイルの名前は data_2.dat であり、各ファイルのサイズは 48M バイトです。

   - すべての CREATE TABLESPACE および ALTER TABLESPACE ステートメントには、ENGINE 句を含める必要があります。テーブルスペースには、テーブルスペースと同じストレージエンジンを使用しているテーブルのみを作成できます。MySQL Cluster NDB 7.3 では、この句に許可される値は NDBCLUSTER と NDB だけです。

- CREATE TABLESPACE および ALTER TABLESPACE ステートメントについての詳細は、セクション13.1.18「CREATE TABLESPACE 構文」およびセクション13.1.8「ALTER TABLESPACE 構文」を参照してください。

3. この時点で、ディスク上のテーブルスペース ts_1 内にインデックスなしカラムを格納するテーブルを作成できます。

```
CREATE TABLE dt_1 (
    member_id INT UNSIGNED NOT NULL AUTO_INCREMENT PRIMARY KEY,
    last_name VARCHAR(50) NOT NULL,
    first_name VARCHAR(50) NOT NULL,
    dob DATE NOT NULL,
    joined DATE NOT NULL,
    INDEX(last_name, first_name)
    )
    TABLESPACE ts_1 STORAGE DISK
    ENGINE NDBCLUSTER;
```

TABLESPACE ... STORAGE DISK オプションは、ディスクデータストレージ用にテーブルスペース ts_1 を使用するように NDBCLUSTER ストレージエンジンに指示します。

> **注記**
>
> CREATE TABLE または ALTER TABLE ステートメントでカラムの定義の一部として STORAGE 句を使用すると、個々のカラムをディスク上とメモリー内のどちらに格納するかを指定することもできます。STORAGE DISK を指定するとカラムはディスク上に格納され、STORAGE MEMORY を指定するとインメモリーストレージが使用されます。詳細は、セクション13.1.17「CREATE TABLE 構文」を参照してください。

次のようにテーブル ts_1 が作成されたら、その他の MySQL テーブルの場合と同様に、INSERT、SELECT、UPDATE、および DELETE ステートメントを実行できます。

ここで定義されたテーブル dt_1 の場合、dob および joined カラムのみがディスク上に格納されます。この理由は、id、last_name、および first_name カラムにインデックスがあるため、これらのカラムに属するデータが RAM 内に格納されるためです。MySQL Cluster NDB 7.3 および MySQL Cluster NDB 7.4 では、インデックスなしカラムのみをディスク上に保持でき、インデックスおよびインデックス付きカラムのデータは引き続き、メモリー内に格納されます。インデックスの使用と RAM の保存のこのようなトレードオフは、ディスクデータテーブルを設計する際に留意する必要があります。

パフォーマンスに関する注意　データノードファイルシステムから分離された物理ディスク上にディスクデータファイルが保持されている場合、ディスクデータストレージを使用しているクラスタのパフォーマンスが大幅に改善されます。顕著な利点を引き出すには、クラスタ内のデータノードごとに、これを実行する必要があります。

ADD UNDOFILE および ADD DATAFILE では、ファイルシステムの完全パスおよび相対パスを使用できます。相対パスは、データノードのデータディレクトリに相対的に計算されます。シンボリックリンクを使用することもできます。詳細および例については、セクション18.5.12.2「ディスクデータオブジェクトでのシンボリックリンクの使用」を参照してください。

これらを使用しているログファイルグループ、テーブルスペース、およびディスクデータテーブルは、特定の順序で作成する必要があります。これらのオブジェクトを削除する際にも、同じことが当てはまります。

- ログファイルグループは、テーブルスペースで使用されているかぎり、削除できません。
- テーブルスペースは、データファイルを格納しているかぎり、削除できません。
- テーブルスペースを使用しているテーブルが残っているかぎり、テーブルスペースからどのデータファイルも削除できません。
- ファイルが作成されたテーブルスペースとは別のテーブルスペースに関連付けられて作成されたファイルは、削除できません。(Bug #20053)

たとえば、このセクションでこれまで作成してきたすべてのオブジェクトを削除するには、次のステートメントを使用します。

```
mysql> DROP TABLE dt_1;

mysql> ALTER TABLESPACE ts_1
```

```
  -> DROP DATAFILE 'data_2.dat'
  -> ENGINE NDBCLUSTER;

mysql> ALTER TABLESPACE ts_1
  -> DROP DATAFILE 'data_1.dat'
  -> ENGINE NDBCLUSTER;

mysql> DROP TABLESPACE ts_1
  -> ENGINE NDBCLUSTER;

mysql> DROP LOGFILE GROUP lg_1
  -> ENGINE NDBCLUSTER;
```

これらのステートメントは、ここで示した順序で実行する必要があります。ただし、2 つの ALTER TABLESPACE ... DROP DATAFILE ステートメントは、どちらの順序でも実行できます。

INFORMATION_SCHEMA データベースの FILES テーブルをクエリーすると、ディスクデータテーブルによって使用されているデータファイルに関する情報を取得できます。追加の「NULL 行」には、Undo ログファイルに関する追加情報が示されます。詳細および例については、セクション21.30.1「INFORMATION_SCHEMA FILES テーブル」を参照してください。

ndb_desc ユーティリティーを使用すると、ディスクデータテーブルまたはテーブルパーティションごとに、割り当て済みおよび空きディスク容量に関する情報を表示することもできます。詳細は、セクション18.4.10「ndb_desc — NDB テーブルの表示」を参照してください。

18.5.12.2 ディスクデータオブジェクトでのシンボリックリンクの使用

データノードファイルシステムを Undo ログファイルおよびテーブルスペースデータファイルから分離し、これらを別々のディスク上に配置すると、ディスクデータストレージを使用する MySQL Cluster のパフォーマンスを大幅に改善できます。旧バージョンの MySQL Cluster では、これに対する直接的なサポートはありませんでしたが、このセクションで説明するようにシンボリックリンクを使用すれば、このような分離を実現できました。MySQL Cluster では、このためにシンボリックリンクを使用する必要性をなくすデータノード構成パラメータ FileSystemPathDD、FileSystemPathDataFiles、および FileSystemPathUndoFiles がサポートされています。これらのパラメータについての詳細は、ディスクデータのファイルシステムパラメータを参照してください。

クラスタ内の各データノードは、config.ini ファイルに定義されたデータノードの DataDir にある ndb_node_id_fs という名前のディレクトリに、ファイルシステムを作成します。この例では、各データノードホストが /data0、/data1、および /data2 というエイリアスが付けられた 3 台のディスクを持ち、クラスタの config.ini に次のエントリが含まれていると仮定します。

```
[ndbd default]
DataDir= /data0
```

ここでの目的は、各データノードホスト上で、/data1 にすべてのディスクデータログファイル、/data2 にすべてのディスクデータデータファイルを配置することです。

> **注記**
>
> この例では、クラスタのデータノードホストではすべて、Linux オペレーティングシステムが使用されていると仮定します。その他のプラットフォームでは、ここで示したコマンドをオペレーティングシステムのコマンドで置き換える必要がある場合があります。

これを実現するには、次のステップを実行します。

- データノードファイルシステムで、ほかのドライブを指しているシンボリックリンクを作成します。

```
shell> cd /data0/ndb_2_fs
shell> ls
D1  D10  D11  D2  D8  D9  LCP
shell> ln -s /data0 dnlogs
shell> ln -s /data1 dndata
```

この時点で、次の 2 つのシンボリックリンクが表示されます。

```
shell> ls -l --hide=D*
lrwxrwxrwx 1 user group   30 2007-03-19 13:58 dndata -> /data1
lrwxrwxrwx 1 user group   30 2007-03-19 13:59 dnlogs -> /data2
```

ここでは、ノード ID が 2 のデータノードの場合のみを示していますが、これを各データノードで実行する必要があります。

- ここで、mysql クライアントで、次に示すようにシンボリックリンクを使用してログファイルグループおよびテーブルスペースを作成します。

```
mysql> CREATE LOGFILE GROUP lg1
    ->    ADD UNDOFILE 'dnlogs/undo1.log'
    ->    INITIAL_SIZE 150M
    ->    UNDO_BUFFER_SIZE = 1M
    ->    ENGINE=NDBCLUSTER;

mysql> CREATE TABLESPACE ts1
    ->    ADD DATAFILE 'dndata/data1.log'
    ->    USE LOGFILE GROUP lg1
    ->    INITIAL_SIZE 1G
    ->    ENGINE=NDBCLUSTER;
```

次に示すように、ファイルが正しく作成および配置されたことを確認します。

```
shell> cd /data1
shell> ls -l
total 2099304
-rw-rw-r--  1 user group 157286400 2007-03-19 14:02 undo1.dat

shell> cd /data2
shell> ls -l
total 2099304
-rw-rw-r--  1 user group 1073741824 2007-03-19 14:02 data1.dat
```

- 1 つのホスト上で複数のデータノードを実行している場合は、それらがディスクデータファイルに同じ領域の使用を試みないように注意する必要があります。各データノードファイルシステムにシンボリックリンクを作成すると、これを簡単にできます。両方のデータノードファイルシステムに /data0 を使用しているが、/data1 上に両方のノードのディスクデータファイルを配置する必要があるとします。この場合、次に示すものと同様の操作を実行できます。

```
shell> cd /data0
shell> ln -s /data1/dn2 ndb_2_fs/dd
shell> ln -s /data1/dn3 ndb_3_fs/dd
shell> ls -l --hide=D* ndb_2_fs
lrwxrwxrwx 1 user group   30 2007-03-19 14:22 dd -> /data1/dn2
shell> ls -l --hide=D* ndb_3_fs
lrwxrwxrwx 1 user group   30 2007-03-19 14:22 dd -> /data1/dn3
```

- この時点で、次に示すように、シンボリックリンクを使用してログファイルグループおよびテーブルスペースを作成できます。

```
mysql> CREATE LOGFILE GROUP lg1
    ->    ADD UNDOFILE 'dd/undo1.log'
    ->    INITIAL_SIZE 150M
    ->    UNDO_BUFFER_SIZE = 1M
    ->    ENGINE=NDBCLUSTER;

mysql> CREATE TABLESPACE ts1
    ->    ADD DATAFILE 'dd/data1.log'
    ->    USE LOGFILE GROUP lg1
    ->    INITIAL_SIZE 1G
    ->    ENGINE=NDBCLUSTER;
```

次に示すように、ファイルが正しく作成および配置されたことを確認します。

```
shell> cd /data1
shell> ls
dn2       dn3
shell> ls dn2
undo1.log       data1.log
shell> ls dn3
undo1.log       data1.log
```

18.5.12.3 MySQL Cluster ディスクデータストレージの要件

ディスクテータストレージの要件には、次の項目が適用されます。

- ディスクデータテーブルの可変長カラムは、固定の容量を占有します。これは、行ごとに、そのカラムで可能性のある最大値を格納するために必要な容量に等しくなります。

  このような値の計算に関する一般的な情報については、セクション11.7「データ型のストレージ要件」を参照してください。

INFORMATION_SCHEMA.FILES テーブルをクエリーして、データファイルおよび Undo ログファイルで使用可能な推定容量を取得できます。詳細および例については、セクション21.30.1「INFORMATION_SCHEMA FILES テーブル」を参照してください。

> **注記**
>
> OPTIMIZE TABLE ステートメントを実行しても、ディスクデータテーブルは影響を受けません。

- ディスクデータテーブルでは、TEXT または BLOB カラムの最初の 256 バイトがメモリー内に格納され、その残りだけがディスク上に格納されます。
- ディスクデータテーブル内の各行では、ディスク上に格納されたデータを指すためにメモリー内の 8 バイトが使用されます。つまり、場合によっては、インメモリーカラムをディスクベースの形式に変換すれば、実際にメモリーの使用率を改善できます。たとえば、CHAR(4) カラムをメモリーベースの形式からディスクベースの形式に変換すると、1 行当たりに使用される DataMemory の量が 4 バイトから 8 バイトに増加します。

> **重要**
>
> --initial オプションを付けてクラスタを起動しても、ディスクデータファイルは削除されません。クラスタの初期再起動を実行する前に、これらを手動で削除する必要があります。

DiskPageBufferMemory のサイズが十分にあることを確認して、ディスクシークの回数を最小限に抑えると、ディスクデータテーブルのパフォーマンスを改善できます。diskpagebuffer テーブルをクエリーすると、このパラメータの値を大きくする必要があるかどうかを判断する際に役立ちます。

18.5.13 MySQL Cluster データノードのオンライン追加

このセクションでは、「オンライン」で (つまり、プロセスの一部として、クラスタを完全にシャットダウンしてから再起動する必要なく) MySQL Cluster データノードを追加する方法について説明します。

> **重要**
>
> 現在、新しいノードグループの一部として、MySQL Cluster に新しいデータノードを追加する必要があります。さらに、レプリカの数 (または 1 ノードグループ当たりのノード数) はオンラインで変更できません。

18.5.13.1 MySQL Cluster データノードのオンライン追加: 一般的な問題

このセクションでは、MySQL Cluster ノードをオンラインで追加する際の動作および現在の制限事項に関する一般的な情報を示します。

データの再配布　新しいノードをオンラインで追加する機能には、ALTER ONLINE TABLE ... REORGANIZE PARTITION ステートメントを使用して、すべてのデータノード (新しいものを含む) 間で配布されるように NDBCLUSTER テーブルデータおよびインデックスを再編成する手段も含まれます。インメモリーとディスクデータの両方のテーブルの再編成がサポートされています。現在、この再配布には一意のインデックスが含まれていません (順序付きインデックスのみが再配布されます)。MySQL Cluster NDB 7.3.3 より前では、この方法を使用して BLOB テーブルデータも再配布されません (Bug #13714148)。

新しいデータノードが追加される前にすでに存在していた NDBCLUSTER テーブルの再配布は、自動的に実行されませんが、mysql または別の MySQL クライアントアプリケーションで単純な SQL ステートメントを使用して実現できます。ただし、新しいノードグループが追加されたあとに作成されたテーブルに追加されたすべてのデータおよびインデックスは、すべてのクラスタデータノード (新しいノードグループの一部として追加されたものを含む) 間で自動的に配布されます。

部分的な起動　新しいデータノードがすべて起動されていなくても、新しいノードグループを追加できます。また、新しいノードグループを機能低下状態のクラスタ (つまり、部分的にしか起動されていないクラスタや、1 つ以上のデータノードが実行されていないクラスタ) に追加することもできます。後者の場合、新しいノードグループを追加する前に、クラスタで十分な数のノードが実行可能になっている必要があります。

進行中の操作への影響　新しいノードグループを作成または追加したり、テーブルを再編成したりしても、MySQL Cluster データを使用する通常の DML 操作は妨げられません。ただし、テーブルの再編成と同時に DDL を実行することはできません。つまり、ALTER TABLE ... REORGANIZE PARTITION ステートメントの実

行中は、ほかの DDL ステートメントを発行できません。さらに、ALTER TABLE ... REORGANIZE PARTITION の実行 (またはその他の DDL ステートメントの実行) 中は、クラスタデータノードを再起動できません。

エラー処理　ノードグループの作成時およびテーブルの再編成時のデータノードのエラーは、次の表に示すように処理されます。

エラーが発生するタイミング:	エラーが発生する場所:		
	「古い」データノード	「新しい」データノード	システム

ノードグループの作成	• マスター以外のノードでエラーが発生した場合:　ノードグループの作成は常にロールフォワードされます。
	• マスターでエラーが発生した場合:
	  • 内部コミットポイントに達した場合:　ノードグループの作成はロールフォワードされます。
	  • 内部コミットポイントにまだ達していない場合:　ノードグループの作成はロールバックされます。
		• マスター以外のノードでエラーが発生した場合:　ノードグループの作成は常にロールフォワードされます。
		• マスターでエラーが発生した場合:
		  • 内部コミットポイントに達した場合:　ノードグループの作成はロールフォワードされます。
		  • 内部コミットポイントにまだ達していない場合:　ノードグループの作成はロールバックされます。
			• CREATE NODEGROUP の実行が内部コミットポイントに達した場合:　再起動時に、クラスタは新しいノードグループを含みます。それ以外の場合は、含みません。
			• CREATE NODEGROUP の実行が内部コミットポイントにまだ達していない場合:　再起動時に、クラスタは新しいノードグループを含みません。

テーブルの再編成	• マスター以外のノードでエラーが発生した場合:　テーブルの再編成は常にロールフォワードされます。
	• マスターでエラーが発生した場合:
	  • 内部コミットポイントに達した場合:　テーブルの再編成はロールフォワードされます。
	  • 内部コミットポイントにまだ達していない場合:　テーブルの再編成はロールバックされます。
		• マスター以外のノードでエラーが発生した場合:　テーブルの再編成は常にロールフォワードされます。
		• マスターでエラーが発生した場合:
		  • 内部コミットポイントに達した場合:　テーブルの再編成はロールフォワードされます。
		  • 内部コミットポイントにまだ達していない場合:　テーブルの再編成はロールバックされます。
			• ALTER ONLINE TABLE テーブル の REORGANIZE PARTITION ステートメントの実行が内部コミットポイントに達した場合:　クラスタの再起動時に、table に属するデータおよびインデックスが「新しい」データノードを使用して配布されます。
			• ALTER ONLINE TABLE テーブル の REORGANIZE PARTITION ステートメントの実行が内部コミットポイントにまだ達していない場合:　クラスタの再起動時に、table に属するデータおよびインデックスが「古い」データノードのみを使用して配布されます。

ノードグループの削除　ndb_mgm クライアントは、DROP NODEGROUP コマンドをサポートしていますが、ノードグループ内のデータノードにどのデータも含まれていない場合にのみ削除できます。現在、特定のデータノードまたはノードグループを「空にする」方法がないため、このコマンドは次の 2 つのケースでしか機能しません。

1. ndb_mgm クライアントで CREATE NODEGROUP を発行してから、mysql クライアントで任意の ALTER ONLINE TABLE ... REORGANIZE PARTITION ステートメントを発行するまでの間。
2. DROP TABLE を使用してすべての NDBCLUSTER テーブルを削除したあと。

データノードには引き続きテーブルの定義が格納されるため、この目的では TRUNCATE TABLE は機能しません。

18.5.13.2 MySQL Cluster データノードのオンライン追加: 基本的な手順

このセクションでは、新しいデータノードを MySQL Cluster に追加するために必要な基本的なステップを一覧表示します。この手順は、データノードのプロセスに ndbd バイナリと ndbmtd バイナリのどちらを使用する場合でも適用されます。詳細な例については、セクション18.5.13.3「MySQL Cluster データノードのオンライン追加: 詳細な例」を参照してください。

すでに MySQL Cluster が実行されていると仮定すると、データノードをオンラインで追加するには次のステップが必要です。

1. 追加対象のノードに対応する新しい [ndbd] セクションを追加して、クラスタ構成 config.ini ファイルを編集します。クラスタで複数の管理サーバーが使用されている場合は、管理サーバーで使用されるすべての config.ini ファイルに、これらの変更を加える必要があります。

   config.ini ファイルに追加される新しいデータノードのノード ID が、既存のノードで使用されるノード ID と重複しないように注意する必要があります。動的に割り当てられたノード ID を使用している API ノードがあり、それらの ID が新しいノードに使用するノード ID と一致する場合は、この手順の後半で説明するように、このような API ノードを強制的に「移行」できます。

2. すべての MySQL Cluster 管理サーバーのローリング再起動を実行します。

   > **重要**
   >
   > 新しい構成を強制的に読み取るには、すべての管理サーバーを --reload または --initial オプションを付けて再起動する必要があります。

3. 既存のすべての MySQL Cluster データノードのローリング再起動を実行します。既存のデータノードを再起動するときは、--initial を使用する必要がありません (通常は望ましくもありません)。

   新しいデータノードに割り当てるいずれかのノード ID と一致する動的に割り当てられた ID を持つ API ノードを使用している場合は、このステップでデータノードプロセスのいずれかを再起動する前に、すべての API ノード (SQL ノードを含む) を再起動する必要があります。これにより、事前に明示的に割り当てられなかったノード ID を持つ API ノードは、このようなノード ID を放棄し、新しい ID を取得します。

4. MySQL Cluster に接続されているすべての SQL ノードまたは API ノードのローリング再起動を実行します。

5. 新しいデータノードを起動します。

   新しいデータノードは、任意の順序で起動できます。またそれらは、既存のすべてのデータノードのローリング再起動が完了してから、次のステップに進むまでの間に起動されるかぎり、同時に起動することもできます。

6. MySQL Cluster 管理クライアントで CREATE NODEGROUP コマンドを 1 回以上実行して、新しいデータノードが属する 1 つまたは複数の新しいノードグループを作成します。

7. NDBCLUSTER テーブルごとに、mysql クライアントで ALTER ONLINE TABLE ... REORGANIZE PARTITION ステートメントを発行して、すべてのデータノード (新しいものを含む) 間でクラスタのデータを再配布します。

   > **注記**
   >
   > これは、新しいノードグループを追加する時点ですでに存在しているテーブルに対してのみ実行する必要があります。新しいノードグループを追加したあとに作成されたテーブル内のデータは、自動的に再配布されません。ただし、新しいノードを追加する前に存在していた特定のテーブル tbl に追加されたデータは、そのテーブルが ALTER ONLINE TABLE tbl REORGANIZE PARTITION を使用して再編成されるまで、新しいノードを使用して再配布されません。

8. NDBCLUSTER テーブルごとに、mysql クライアントで OPTIMIZE TABLE ステートメントを発行して、「古い」ノード上の解放された領域を再利用します。

目的のノードをすべて追加してから、複数の CREATE NODEGROUP コマンドを連続して発行し、新しいノードグループをクラスタに追加できます。

18.5.13.3 MySQL Cluster データノードのオンライン追加: 詳細な例

このセクションでは、MySQL Cluster データノードをオンラインで追加する方法について説明する詳細な例を示します。まず、単一のノードグループ内に 2 つのデータノードを持つ MySQL Cluster から開始し、2 つのノードグループ内に 4 つのデータノードを持つクラスタで終了します。

構成の開始　説明のために、最小限の構成とし、次の情報のみを含む config.ini ファイルをクラスタで使用すると仮定します。

```
[ndbd default]
DataMemory = 100M
IndexMemory = 100M
NoOfReplicas = 2
DataDir = /usr/local/mysql/var/mysql-cluster

[ndbd]
Id = 1
HostName = 192.168.0.1

[ndbd]
Id = 2
HostName = 192.168.0.2

[mgm]
HostName = 192.168.0.10
Id = 10

[api]
Id=20
HostName = 192.168.0.20

[api]
Id=21
HostName = 192.168.0.21
```

> **注記**
>
> データノード ID とその他のノード間のシーケンスにギャップを残しています。これにより、あとで新たに追加されるデータノードに、まだ使用されていないノード ID を簡単に割り当てることができます。

また、適切なコマンド行または my.cnf オプションを使用して、クラスタをすでに起動しており、管理クライアントで SHOW を実行すると、次に示すものと同様の出力が生成されると仮定します。

```
-- NDB Cluster -- Management Client --
ndb_mgm> SHOW
Connected to Management Server at: 192.168.0.10:1186
Cluster Configuration
---------------------
[ndbd(NDB)]     2 node(s)
id=1    @192.168.0.1  (5.6.22-ndb-7.4.4, Nodegroup: 0, *)
id=2    @192.168.0.2  (5.6.22-ndb-7.4.4, Nodegroup: 0)

[ndb_mgmd(MGM)] 1 node(s)
id=10   @192.168.0.10  (5.6.22-ndb-7.4.4)

[mysqld(API)]   2 node(s)
id=20   @192.168.0.20  (5.6.22-ndb-7.4.4)
id=21   @192.168.0.21  (5.6.22-ndb-7.4.4)
```

最後に、次に示すように作成された単一の NDBCLUSTER テーブルがクラスタに含まれていると仮定します。

```
USE n;

CREATE TABLE ips (
    id BIGINT NOT NULL AUTO_INCREMENT PRIMARY KEY,
    country_code CHAR(2) NOT NULL,
    type CHAR(4) NOT NULL,
    ip_address varchar(15) NOT NULL,
    addresses BIGINT UNSIGNED DEFAULT NULL,
    date BIGINT UNSIGNED DEFAULT NULL
)   ENGINE NDBCLUSTER;
```

このセクションの後半で示すメモリーの使用率および関連情報は、このテーブルに約 50000 行挿入したあとに生成されました。

> **注記**
>
> この例では、データノードプロセスに、シングルスレッド ndbd を使用することを示します。ただし、MySQL Cluster NDB 7.0.4 以降では、マルチスレッドの ndbmtd を使用している場合でも、以降のステップの ndbd が示されている箇所をすべて ndbmtd に置き換えることによって、この例を適用できます。(Bug #43108)

ステップ 1: 構成ファイルの更新　テキストエディタでクラスタのグローバル構成ファイルを開き、2 つの新しいデータノードに対応する [ndbd] セクションを追加します。(これらのデータノードに ID 3 と 4 を付与し、それらがそれぞれ 192.168.0.3 と 192.168.0.4 のアドレスにあるホストマシンで実行されると仮定します。)新しいセクションを追加したら、config.ini ファイルの内容が次に示すように見えます。ここで、ファイルに追加した部分は太字で示しています。

```
[ndbd default]
DataMemory = 100M
IndexMemory = 100M
NoOfReplicas = 2
DataDir = /usr/local/mysql/var/mysql-cluster

[ndbd]
Id = 1
HostName = 192.168.0.1

[ndbd]
Id = 2
HostName = 192.168.0.2

[ndbd]
Id = 3
HostName = 192.168.0.3

[ndbd]
Id = 4
HostName = 192.168.0.4

[mgm]
HostName = 192.168.0.10
Id = 10

[api]
Id=20
HostName = 192.168.0.20

[api]
Id=21
HostName = 192.168.0.21
```

必要な変更が完了したら、ファイルを保存します。

ステップ 2: 管理サーバーの再起動　クラスタ管理サーバーを再起動するには、次のように別々のコマンドを発行して、管理サーバーを停止してから再起動する必要があります。

1. 次に示すように管理クライアントの STOP コマンドを使用して、管理サーバーを停止します。

   ```
   ndb_mgm> 10 STOP
   Node 10 has shut down.
   Disconnecting to allow Management Server to shutdown

   shell>
   ```

2. 管理サーバーをシャットダウンすると管理クライアントが終了するため、システムシェルから管理サーバーを起動する必要があります。単純にするために、config.ini は管理サーバーバイナリと同じディレクトリにあると仮定していますが、実際には、構成ファイルの正確なパスを指定する必要があります。また、管理サーバーがその構成キャッシュからではなく、ファイルから新しい構成を読み取るように、--reload または --initial オプションも指定する必要があります。シェルの現在のディレクトリも管理サーバーバイナリが配置されているディレクトリと同じである場合は、次に示すように管理サーバーを呼び出すことができます。

   ```
   shell> ndb_mgmd -f config.ini --reload
   2008-12-08 17:29:23 [MgmSrvr] INFO     -- NDB Cluster Management Server. 5.6.22-ndb-7.4.4
   2008-12-08 17:29:23 [MgmSrvr] INFO     -- Reading cluster configuration from 'config.ini'
   ```

ndb_mgm プロセスが再起動されたあとに、管理クライアントで SHOW の出力をチェックすると、次に示すように表示されます。

```
-- NDB Cluster -- Management Client --
ndb_mgm> SHOW
```

```
Connected to Management Server at: 192.168.0.10:1186
Cluster Configuration
---------------------
[ndbd(NDB)]     2 node(s)
id=1    @192.168.0.1  (5.6.22-ndb-7.4.4, Nodegroup: 0, *)
id=2    @192.168.0.2  (5.6.22-ndb-7.4.4, Nodegroup: 0)
id=3 (not connected, accepting connect from 192.168.0.3)
id=4 (not connected, accepting connect from 192.168.0.4)

[ndb_mgmd(MGM)] 1 node(s)
id=10   @192.168.0.10  (5.6.22-ndb-7.4.4)

[mysqld(API)]   2 node(s)
id=20   @192.168.0.20  (5.6.22-ndb-7.4.4)
id=21   @192.168.0.21  (5.6.22-ndb-7.4.4)
```

ステップ 3: 既存のデータノードのローリング再起動の実行　次に示すように RESTART コマンドを使用すると、このステップを完全にクラスタ管理クライアント内で実現できます。

```
ndb_mgm> 1 RESTART
Node 1: Node shutdown initiated
Node 1: Node shutdown completed, restarting, no start.
Node 1 is being restarted

ndb_mgm> Node 1: Start initiated (version 7.4.4)
Node 1: Started (version 7.1.35)

ndb_mgm> 2 RESTART
Node 2: Node shutdown initiated
Node 2: Node shutdown completed, restarting, no start.
Node 2 is being restarted

ndb_mgm> Node 2: Start initiated (version 7.4.4)

ndb_mgm> Node 2: Started (version 7.4.4)
```

> **重要**
>
> 各 X RESTART コマンドを発行したら、管理クライアントによって「Node X: Started (version ...)」というレポートが出力されるまで待機してから、先に進みます。

mysql クライアントで ndbinfo.nodes テーブルをチェックすると、既存のデータノードがすべて更新済みの構成を使用して再起動されたことを確認できます。

ステップ 4: すべてのクラスタ API ノードのローリング再起動の実行　mysqladmin shutdown に続いて、mysqld_safe (または別の起動スクリプト) を使用して、クラスタ内の SQL ノードとして機能している各 MySQL サーバーをシャットダウンし、再起動します。これは、次に示すものと同様になります。ここで、password は特定の MySQL サーバーインスタンスの MySQL root パスワードです。

```
shell> mysqladmin -uroot -ppassword shutdown
081208 20:19:56 mysqld_safe mysqld from pid file
/usr/local/mysql/var/tonfisk.pid ended
shell> mysqld_safe --ndbcluster --ndb-connectstring=192.168.0.10 &
081208 20:20:06 mysqld_safe Logging to '/usr/local/mysql/var/tonfisk.err'.
081208 20:20:06 mysqld_safe Starting mysqld daemon with databases
from /usr/local/mysql/var
```

当然、正確な入力および出力は、MySQL がシステムにインストールされている方法や場所、および起動する際に選択したオプション (およびこれらのオプションの一部またはすべてを my.cnf ファイルに指定しているかどうか) によって異なります。

ステップ 5: 新しいデータノードの初期起動の実行　次に示すように、新しいデータノードの各ホスト上のシステムシェルから --initial オプションを使用して、データノードを起動します。

```
shell> ndbd -c 192.168.0.10 --initial
```

> **注記**
>
> 既存のデータノードを再起動する場合とは異なり、新しいデータノードは同時に起動できます。あるデータノードの起動が完了するまで待機してから、別のデータノードを起動する必要はありません。

新しいデータノードの両方が起動されるまで待機してから、次のステップに進みます。新しいデータノードが起動されたら、管理クライアントの SHOW コマンドの出力で、(次に太字で示されているように) それらがどのノードグループにも属していないことを確認できます。

```
ndb_mgm> SHOW
Connected to Management Server at: 192.168.0.10:1186
Cluster Configuration
---------------------
[ndbd(NDB)]     2 node(s)
id=1    @192.168.0.1  (5.6.22-ndb-7.4.4, Nodegroup: 0, *)
id=2    @192.168.0.2  (5.6.22-ndb-7.4.4, Nodegroup: 0)
id=3    @192.168.0.3  (5.6.22-ndb-7.4.4, no nodegroup)
id=4    @192.168.0.4  (5.6.22-ndb-7.4.4, no nodegroup)

[ndb_mgmd(MGM)] 1 node(s)
id=10   @192.168.0.10  (5.6.22-ndb-7.4.4)

[mysqld(API)]   2 node(s)
id=20   @192.168.0.20  (5.6.22-ndb-7.4.4)
id=21   @192.168.0.21  (5.6.22-ndb-7.4.4)
```

ステップ 6: 新しいノードグループの作成　　これは、クラスタ管理クライアントで CREATE NODEGROUP コマンドを発行することで実行できます。次に示すように、このコマンドは、引数として、新しいノードグループに含むデータノードのノード ID をカンマで区切ったリストを取ります。

```
ndb_mgm> CREATE NODEGROUP 3,4
Nodegroup 1 created
```

再度 SHOW を発行すると、(同様に太字に示されているように) データノード 3 と 4 が新しいノードグループに参加したことを確認できます。

```
ndb_mgm> SHOW
Connected to Management Server at: 192.168.0.10:1186
Cluster Configuration
---------------------
[ndbd(NDB)]     2 node(s)
id=1    @192.168.0.1  (5.6.22-ndb-7.4.4, Nodegroup: 0, *)
id=2    @192.168.0.2  (5.6.22-ndb-7.4.4, Nodegroup: 0)
id=3    @192.168.0.3  (5.6.22-ndb-7.4.4, Nodegroup: 1)
id=4    @192.168.0.4  (5.6.22-ndb-7.4.4, Nodegroup: 1)

[ndb_mgmd(MGM)] 1 node(s)
id=10   @192.168.0.10  (5.6.22-ndb-7.4.4)

[mysqld(API)]   2 node(s)
id=20   @192.168.0.20  (5.6.22-ndb-7.4.4)
id=21   @192.168.0.21  (5.6.22-ndb-7.4.4)
```

ステップ 7: クラスタデータの再配布　　管理クライアントで適切な REPORT コマンドを発行することで確認できるように、ノードグループが作成されたときに、既存のデータおよびインデックスは自動的に新しいノードグループのデータノードに配布されません。

```
ndb_mgm> ALL REPORT MEMORY

Node 1: Data usage is 5%(177 32K pages of total 3200)
Node 1: Index usage is 0%(108 8K pages of total 12832)
Node 2: Data usage is 5%(177 32K pages of total 3200)
Node 2: Index usage is 0%(108 8K pages of total 12832)
Node 3: Data usage is 0%(0 32K pages of total 3200)
Node 3: Index usage is 0%(0 8K pages of total 12832)
Node 4: Data usage is 0%(0 32K pages of total 3200)
Node 4: Index usage is 0%(0 8K pages of total 12832)
```

-p オプションを付けて ndb_desc を使用すると、出力にパーティション化の情報が含まれるため、テーブルではまだ 2 つのパーティションしか使用されていないことを (ここに太字のテキストで示されている、出力の「Per partition info」セクションで) 確認できます。

```
shell> ndb_desc -c 192.168.0.10 -d n ips -p
-- ips --
Version: 1
Fragment type: 9
K Value: 6
Min load factor: 78
Max load factor: 80
Temporary table: no
Number of attributes: 6
Number of primary keys: 1
Length of frm data: 340
Row Checksum: 1
Row GCI: 1
SingleUserMode: 0
```

```
ForceVarPart: 1
FragmentCount: 2
TableStatus: Retrieved
-- Attributes --
id Bigint PRIMARY KEY DISTRIBUTION KEY AT=FIXED ST=MEMORY AUTO_INCR
country_code Char(2;latin1_swedish_ci) NOT NULL AT=FIXED ST=MEMORY
type Char(4;latin1_swedish_ci) NOT NULL AT=FIXED ST=MEMORY
ip_address Varchar(15;latin1_swedish_ci) NOT NULL AT=SHORT_VAR ST=MEMORY
addresses Bigunsigned NULL AT=FIXED ST=MEMORY
date Bigunsigned NULL AT=FIXED ST=MEMORY

-- Indexes --
PRIMARY KEY(id) - UniqueHashIndex
PRIMARY(id) - OrderedIndex

-- Per partition info --
Partition   Row count   Commit count  Frag fixed memory   Frag varsized memory
0           26086       26086         1572864             557056
1           26329       26329         1605632             557056

NDBT_ProgramExit: 0 - OK
```

NDBCLUSTER テーブルごとに、mysql クライアントで ALTER ONLINE TABLE ... REORGANIZE PARTITION ステートメントを実行して、すべてのデータノード間でデータが再配布されるようにできます。ALTER ONLINE TABLE ips REORGANIZE PARTITION ステートメントを発行したあとに、ndb_desc を使用すると、次に示すように (太字の出力の関連部分によって)、このテーブルのテータが 4 つのパーティションを使用して格納されるようになったことを確認できます。

```
shell> ndb_desc -c 192.168.0.10 -d n ips -p
-- ips --
Version: 16777217
Fragment type: 9
K Value: 6
Min load factor: 78
Max load factor: 80
Temporary table: no
Number of attributes: 6
Number of primary keys: 1
Length of frm data: 341
Row Checksum: 1
Row GCI: 1
SingleUserMode: 0
ForceVarPart: 1
FragmentCount: 4
TableStatus: Retrieved
-- Attributes --
id Bigint PRIMARY KEY DISTRIBUTION KEY AT=FIXED ST=MEMORY AUTO_INCR
country_code Char(2;latin1_swedish_ci) NOT NULL AT=FIXED ST=MEMORY
type Char(4;latin1_swedish_ci) NOT NULL AT=FIXED ST=MEMORY
ip_address Varchar(15;latin1_swedish_ci) NOT NULL AT=SHORT_VAR ST=MEMORY
addresses Bigunsigned NULL AT=FIXED ST=MEMORY
date Bigunsigned NULL AT=FIXED ST=MEMORY

-- Indexes --
PRIMARY KEY(id) - UniqueHashIndex
PRIMARY(id) - OrderedIndex

-- Per partition info --
Partition   Row count   Commit count  Frag fixed memory   Frag varsized memory
0           12981       52296         1572864             557056
1           13236       52515         1605632             557056
2           13105       13105         819200              294912
3           13093       13093         819200              294912

NDBT_ProgramExit: 0 - OK
```

> **注記**
>
> 通常、ALTER [ONLINE] TABLE table_name REORGANIZE PARTITION は、パーティション識別子のリストおよびパーティション定義のセットとともに使用して、すでに明示的にパーティション化されているテーブルに新しいパーティション化スキームを作成します。この点で、新しい MySQL Cluster ノードグループにデータを再配布するここでの使い方は例外です。このように使用する場合、TABLE キーワードのあとにテーブルの名前のみを使用し、その他のキーワードや識別子は REORGANIZE PARTITION のあとに続けません。
>
> 詳細は、セクション13.1.7「ALTER TABLE 構文」を参照してください。

さらに、無駄になっている領域が再利用されるように、テーブルごとに、ALTER ONLINE TABLE ステートメントのあとに OPTIMIZE TABLE を続けて指定するべきです。INFORMATION_SCHEMA.TABLES テーブルに対して次のクエリーを使用すると、すべての NDBCLUSTER テーブルのリストを取得できます。

```
SELECT TABLE_SCHEMA, TABLE_NAME
  FROM INFORMATION_SCHEMA.TABLES
  WHERE ENGINE = 'NDBCLUSTER';
```

> **注記**
>
> MySQL Cluster テーブルでは、テーブルを作成するために使用された CREATE TABLE ステートメント (または、別のストレージエンジンから既存のテーブルを変換するために使用された ALTER TABLE ステートメント) の ENGINE オプションで NDB が使用されたか、または NDBCLUSTER が使用されたかに関係なく、INFORMATION_SCHEMA.TABLES.ENGINE の値は常に NDBCLUSTER です。

これらのステートメントを実行したあとは、次に示すように ALL REPORT MEMORY の出力で、データおよびインデックスがすべてのデータノード間で再配布されたことを確認できます。

```
ndb_mgm> ALL REPORT MEMORY

Node 1: Data usage is 5%(176 32K pages of total 3200)
Node 1: Index usage is 0%(76 8K pages of total 12832)
Node 2: Data usage is 5%(176 32K pages of total 3200)
Node 2: Index usage is 0%(76 8K pages of total 12832)
Node 3: Data usage is 2%(80 32K pages of total 3200)
Node 3: Index usage is 0%(51 8K pages of total 12832)
Node 4: Data usage is 2%(80 32K pages of total 3200)
Node 4: Index usage is 0%(50 8K pages of total 12832)
```

> **注記**
>
> NDBCLUSTER テーブルに対して同時に 1 回の DDL 操作しか実行できないため、各 ALTER ONLINE TABLE ... REORGANIZE PARTITION ステートメントが完了するまで待機してから、次のステートメントを発行する必要があります。

新しいデータノードが追加されたあとに作成された NDBCLUSTER テーブルに対しては、ALTER ONLINE TABLE ... REORGANIZE PARTITION ステートメントを発行する必要はありません。このようなテーブルに追加されたデータは、すべてのデータノード間で自動的に配布されます。ただし、新しいノードが追加される前に存在していた NDBCLUSTER テーブルでは、それらのテーブルが ALTER ONLINE TABLE ... REORGANIZE PARTITION を使用して再編成されるまで、既存のデータも新規のデータも新しいノードを使用して配布されません。

ローリング再起動を使用しない代替の手順　はじめてクラスタを起動するときに、余分なデータノードを構成するが、それらを起動しないことによって、ローリング再起動の必要性を回避できます。前のように、まず 1 つのノードグループ内の 2 つのデータノード (ノード 1 と 2) から開始し、あとでノード 3 と 4 で構成される 2 つめのノードグループを追加することで、クラスタを 4 つのデータノードまで拡張すると仮定します。

```
[ndbd default]
DataMemory = 100M
IndexMemory = 100M
NoOfReplicas = 2
DataDir = /usr/local/mysql/var/mysql-cluster

[ndbd]
Id = 1
HostName = 192.168.0.1

[ndbd]
Id = 2
HostName = 192.168.0.2

[ndbd]
Id = 3
HostName = 192.168.0.3
Nodegroup = 65536

[ndbd]
Id = 4
HostName = 192.168.0.4
Nodegroup = 65536

[mgm]
HostName = 192.168.0.10
Id = 10
```

```
[api]
Id=20
HostName = 192.168.0.20

[api]
Id=21
HostName = 192.168.0.21
```

あとでオンラインになるデータノード (ノード 3 と 4) を NodeGroup = 65536 で構成できます。この場合、ノード 1 と 2 はそれぞれ次のように起動できます。

```
shell> ndbd -c 192.168.0.10 --initial
```

NodeGroup = 65536 で構成されたデータノードは、StartNoNodeGroupTimeout データノード構成パラメータの設定によって指定された期間待機してから、--nowait-nodes=3,4 を使用してノード 1 と 2 を起動したかのように、管理サーバーによって扱われます。デフォルトでは、これは 15 秒 (15000 ミリ秒) です。

> **注記**
>
> StartNoNodegroupTimeout はクラスタ内のすべてのデータノードで同じである必要があります。そのため、これは個々のデータノードではなく、常に config.ini ファイルの [ndbd default] セクションに設定するようにしてください。

2 つめのノードグループを追加する準備ができたら、次の追加ステップを実行する必要があるだけです。

1. データノード 3 と 4 を起動し、新しいノードごとにデータノードのプロセスを 1 回ずつ呼び出します。

   ```
   shell> ndbd -c 192.168.0.10 --initial
   ```

2. 管理クライアントで適切な CREATE NODEGROUP コマンドを発行します。

   ```
   ndb_mgm> CREATE NODEGROUP 3,4
   ```

3. mysql クライアントで、既存の NDBCLUSTER テーブルごとに ALTER ONLINE TABLE ... REORGANIZE PARTITION および OPTIMIZE TABLE ステートメントを発行します。(このセクションの別の場所で述べたように、これが完了するまで、既存の MySQL Cluster テーブルはデータ配布に新しいノードを使用できません。)

18.5.14 MySQL Cluster の配布された MySQL 権限

MySQL Cluster では、MySQL Cluster 内のすべての SQL ノード間での MySQL ユーザーおよび権限の配布がサポートされています。このサポートは、デフォルトで無効になっています。有効にするには、このセクションで概要を示した手順に従うようにしてください。

通常は、mysql データベース内の各 MySQL サーバーのユーザー権限テーブルで、MyISAM ストレージエンジンが使用されている必要があります。つまり、ある SQL ノードで作成されたユーザーアカウントおよびそれに関連付けられた権限は、クラスタのその他の SQL ノードで使用できません。SQL ファイル ndb_dist_priv.sql は、MySQL Cluster NDB 7.3 以降の配布で提供されています。このファイルは、MySQL インストールディレクトリ内の share ディレクトリで見つけることができます。

配布された権限を有効にする最初のステップは、このスクリプトを SQL ノードとして機能する MySQL サーバー (これは、ターゲット SQL ノードまたは MySQL サーバーと呼ばれます) にロードすることです。これは、ターゲット SQL ノードで、その MySQL インストールディレクトリに移動したあとに、システムシェルから次のコマンドを実行することで実現できます (ここで、options は、この SQL ノードに接続するために必要となる任意の追加オプションを表します)。

```
shell> mysql options -uroot < share/ndb_dist_priv.sql
```

ndb_dist_priv.sql をインポートすると、ターゲット SQL ノード上の mysql データベースに、いくつかのストアドルーチン (6 つのストアドプロシージャーと 1 つのストアドファンクション) が作成されます。mysql クライアント内の SQL ノードに (MySQL root ユーザーとして) 接続すると、これらが次に示すように作成されたことを確認できます。

```
mysql> SELECT ROUTINE_NAME, ROUTINE_SCHEMA, ROUTINE_TYPE
    ->     FROM INFORMATION_SCHEMA.ROUTINES
    ->     WHERE ROUTINE_NAME LIKE 'mysql_cluster%'
    ->     ORDER BY ROUTINE_TYPE;
+-----------------------------------------+----------------+--------------+
| ROUTINE_NAME                            | ROUTINE_SCHEMA | ROUTINE_TYPE |
+-----------------------------------------+----------------+--------------+
| mysql_cluster_privileges_are_distributed | mysql          | FUNCTION     |
| mysql_cluster_backup_privileges          | mysql          | PROCEDURE    |
```

```
| mysql_cluster_move_grant_tables            | mysql      | PROCEDURE    |
| mysql_cluster_move_privileges              | mysql      | PROCEDURE    |
| mysql_cluster_restore_local_privileges     | mysql      | PROCEDURE    |
| mysql_cluster_restore_privileges           | mysql      | PROCEDURE    |
| mysql_cluster_restore_privileges_from_local | mysql      | PROCEDURE    |
+--------------------------------------------+----------------+--------------+

7 rows in set (0.01 sec)
```

mysql_cluster_move_privileges という名前のストアドプロシージャーは、既存の権限テーブルのバックアップコピーを作成してから、それらを NDB に変換します。

mysql_cluster_move_privileges は、2 つのステップでバックアップおよび変換を実行します。1 つめのステップは、mysql_cluster_backup_privileges を呼び出すことで、これにより、mysql データベースに 2 セットのコピーが作成されます。

- MyISAM ストレージエンジンを使用するローカルコピーのセット。これらの名前は、元の権限テーブル名に _backup というサフィクスを追加することで生成されます。
- NDBCLUSTER ストレージエンジンを使用する配布されたコピーのセット。これらのテーブルは、元のテーブル名の前に ndb_ を付け、_backup を追加することによって、名前が付けられます。

コピーが作成されると、mysql_cluster_move_privileges によって、mysql システムテーブルを NDB に変換する ALTER TABLE ... ENGINE = NDB ステートメントを含む mysql_cluster_move_grant_tables が呼び出されます。

通常は、mysql_cluster_backup_privileges または mysql_cluster_move_grant_tables を手動で呼び出すべきではありません。これらのストアドプロシージャーは、mysql_cluster_move_privileges によって使用されるためにのみ提供されています。

元の権限テーブルは自動的にバックアップされますが、影響を受けるすべての SQL ノードの既存の権限テーブルのバックアップを手動で作成してから続行することは、常に適切な方法です。これは、次に示すものと同様の方法で mysqldump を使用して実行できます。

```
shell> mysqldump options -uroot \
   mysql host user db tables_priv columns_priv procs_priv proxies_priv > backup_file
```

変換を実行するには、mysql クライアントを (再度 MySQL root ユーザーとして) 使用して、ターゲット SQL ノードに接続する必要があります。次のように、ストアドプロシージャーを呼び出します。

```
mysql> CALL mysql.mysql_cluster_move_privileges();
Query OK, 0 rows affected (22.32 sec)
```

このプロシージャーの実行には、権限テーブル内の行数に応じて、ある程度の時間がかかる可能性があります。権限テーブルの一部が空になっている場合は、1 つ以上の No data - zero rows fetched, selected, or processed という警告が mysql_cluster_move_privileges が戻るときに表示されることがあります。このような場合は、安全に警告を無視できます。変換が正常に実行されたことを確認するには、次に示すようにストアドファンクション mysql_cluster_privileges_are_distributed を使用できます。

```
mysql> SELECT CONCAT(
    ->   'Conversion ',
    ->   IF(mysql.mysql_cluster_privileges_are_distributed(), 'succeeded', 'failed'),
    ->   '.')
    ->   AS Result;
+----------------------+
| Result               |
+----------------------+
| Conversion succeeded. |
+----------------------+
1 row in set (0.00 sec)
```

mysql_cluster_privileges_are_distributed は、配布された権限テーブルが存在するかどうかをチェックし、すべての権限テーブルが配布されている場合は 1 を返し、それ以外の場合は 0 を返します。

次のようなクエリーを使用すると、バックアップが作成されたことを確認できます。

```
mysql> SELECT TABLE_NAME, ENGINE FROM INFORMATION_SCHEMA.TABLES
    ->   WHERE TABLE_SCHEMA = 'mysql' AND TABLE_NAME LIKE '%backup'
    ->   ORDER BY ENGINE;
+------------------------+------------+
| TABLE_NAME             | ENGINE     |
+------------------------+------------+
| host_backup            | MyISAM     |
| db_backup              | MyISAM     |
| columns_priv_backup    | MyISAM     |
```

```
| user_backup             | MyISAM     |
| tables_priv_backup      | MyISAM     |
| proxies_priv_backup     | MyISAM     |
| procs_priv_backup       | MyISAM     |
| ndb_user_backup         | ndbcluster |
| ndb_tables_priv_backup  | ndbcluster |
| ndb_proxies_priv_backup | ndbcluster |
| ndb_procs_priv_backup   | ndbcluster |
| ndb_host_backup         | ndbcluster |
| ndb_db_backup           | ndbcluster |
| ndb_columns_priv_backup | ndbcluster |
+-------------------------+------------+
14 rows in set (0.00 sec)
```

配布された権限への変換が実行されると、任意の SQL ノードで MySQL ユーザーアカウントの作成、削除、その権限の更新が行われるたびに、クラスタに接続されているその他のすべての MySQL サーバーで、その変更がすぐに有効になります。権限が配布されると、新たにクラスタに接続するすべての MySQL サーバーが自動的に配布に参加します。

> **注記**
>
> mysql_cluster_move_privileges が実行された時点で SQL ノードにクライアントが接続されていた場合は、これらのクライアントが権限の変更を確認できるように、それらの SQL ノードで FLUSH PRIVILEGES を実行するか、クライアントを切断してから再接続する必要があることがあります。

すべての MySQL ユーザー権限は、接続されているすべての MySQL サーバーに配布されます。これには、ビューおよびストアドルーチンに関連付けられた任意の権限も含まれます。ただし、ビューおよびストアドルーチン自体の配布は、現在サポートされていません。

mysql_cluster_move_privileges の実行中に、SQL ノードがクラスタから切断される場合は、クラスタに再接続したあとに、DROP TABLE IF EXISTS mysql.user mysql.db mysql.tables_priv mysql.columns_priv mysql.procs_priv のようなステートメントを使用して、その権限テーブルを削除する必要があります。これにより、SQL ノードは独自のローカルバージョンの権限テーブルではなく、共有権限テーブルを使用します。これは、新しい SQL ノードをはじめてクラスタに接続するときには必要ありません。

クラスタ全体の初期再起動が行われる (すべてのデータノードがシャットダウンされてから、--initial を付けて再起動される) と、共有権限テーブルが失われます。これが発生した場合は、mysql_cluster_move_privileges によって作成されたバックアップから、または mysqldump を使用して作成されたダンプファイルから元のターゲット SQL ノードを使用して、それらをリストアできます。新しい MySQL サーバーを使用してリストアを実行する必要がある場合は、はじめてクラスタに接続するときに、--skip-grant-tables を付けて起動してください。それ以降は、権限テーブルをローカルでリストアしてから再度 mysql_cluster_move_privileges を使用して、それらを配布できます。テーブルをリストアして配布したあとは、--skip-grant-tables オプションを付けずに、この MySQL サーバーを再起動してください。

ndb_mgm クライアントで START BACKUP を使用して作成されたバックアップから ndb_restore --restore-privilege-tables を使用して、配布されたテーブルをリストアすることもできます。(mysql_cluster_move_privileges で作成された MyISAM テーブルは、START BACKUP コマンドでバックアップされません。) ndb_restore は、デフォルトで権限テーブルをリストアしません。--restore-privilege-tables オプションを使用すれば、これを実行できます。

2 つのプロシージャーのいずれかを使用すると、SQL ノードのローカル権限をリストアできます。mysql_cluster_restore_privileges は次のように機能します。

1. mysql.ndb_*_backup テーブルのコピーを使用できる場合は、これらからシステムテーブルのリストアを試みます。
2. それ以外の場合は、(ndb_ プリフィクスが付けられていない) *_backup という名前のローカルバックアップから、システムテーブルのリストアを試みます。

もう 1 つの mysql_cluster_restore_local_privileges という名前のプロシージャーは、ndb_* バックアップをチェックせずに、ローカルバックアップのみからシステムテーブルをリストアします。

mysql_cluster_restore_privileges または mysql_cluster_restore_local_privileges で再作成されたシステムテーブルでは、MySQL サーバーのデフォルトストレージエンジンが使用されます。また、それらはどのような方法でも共有も配布も行われず、MySQL Cluster の NDB ストエージエンジンが使用されません。

追加のストアドプロシージャー mysql_cluster_restore_privileges_from_local は、mysql_cluster_restore_privileges および mysql_cluster_restore_local_privileges で使用されるために提供されています。直接呼び出すべきではありません。

> **重要**
>
> NDB API や ClusterJ アプリケーションを含む MySQL Cluster データに直接アクセスするアプリケーションは、MySQL 権限システムの対象外です。つまり、付与テーブルを配布したら、ほかの NDB テーブルの場合と同様に、このようなアプリケーションからそれらに自由にアクセスできます。特に、NDB API および ClusterJ アプリケーションは制約なしで、ユーザー名、ホスト名、パスワードハッシュ、および配布された付与テーブルのその他の内容の読み取りと書き込みを行うことができることに留意してください。

18.5.15 NDB API 統計のカウンタと変数

Ndb オブジェクトによって実行されるアクションや、これらのオブジェクトに影響を与えるアクションに関する多くのタイプの統計カウンタを使用できます。このようなアクションには、トランザクションの開始と終了 (または中止)、主キーおよび一意のキーの操作、テーブルスキャン、範囲スキャン、およびプルーニングスキャン、さまざまな操作が完了するまで待機している間にブロックされたスレッド、NDBCLUSTER によって送受信されたデータおよびイベントが含まれます。NDB API が呼び出されたり、データノードによってデータが送受信されたりするたびに、NDB カーネル内部のカウンタが増分されます。mysqld では、これらのカウンタがシステムステータス変数として表示されます。これらの値は、SHOW STATUS の出力で、または INFORMATION_SCHEMA.SESSION_STATUS や INFORMATION_SCHEMA.GLOBAL_STATUS テーブルをクエリーして読み取ることができます。NDB テーブルを操作するステートメントの前後で値を比較すると、API レベルで実行された対応するアクション、およびステートメントを実行するコストを確認できます。

次の SHOW STATUS ステートメントを使用すると、これらのステータス変数をすべて一覧表示できます。

```
mysql> SHOW STATUS LIKE 'ndb_api%';
+-----------------------------------------------+----------+
| Variable_name                                 | Value    |
+-----------------------------------------------+----------+
| Ndb_api_wait_exec_complete_count_session      | 0        |
| Ndb_api_wait_scan_result_count_session        | 0        |
| Ndb_api_wait_meta_request_count_session       | 0        |
| Ndb_api_wait_nanos_count_session              | 0        |
| Ndb_api_bytes_sent_count_session              | 0        |
| Ndb_api_bytes_received_count_session          | 0        |
| Ndb_api_trans_start_count_session             | 0        |
| Ndb_api_trans_commit_count_session            | 0        |
| Ndb_api_trans_abort_count_session             | 0        |
| Ndb_api_trans_close_count_session             | 0        |
| Ndb_api_pk_op_count_session                   | 0        |
| Ndb_api_uk_op_count_session                   | 0        |
| Ndb_api_table_scan_count_session              | 0        |
| Ndb_api_range_scan_count_session              | 0        |
| Ndb_api_pruned_scan_count_session             | 0        |
| Ndb_api_scan_batch_count_session              | 0        |
| Ndb_api_read_row_count_session                | 0        |
| Ndb_api_trans_local_read_row_count_session    | 0        |
| Ndb_api_event_data_count_injector             | 0        |
| Ndb_api_event_nondata_count_injector          | 0        |
| Ndb_api_event_bytes_count_injector            | 0        |
| Ndb_api_wait_exec_complete_count_slave        | 0        |
| Ndb_api_wait_scan_result_count_slave          | 0        |
| Ndb_api_wait_meta_request_count_slave         | 0        |
| Ndb_api_wait_nanos_count_slave                | 0        |
| Ndb_api_bytes_sent_count_slave                | 0        |
| Ndb_api_bytes_received_count_slave            | 0        |
| Ndb_api_trans_start_count_slave               | 0        |
| Ndb_api_trans_commit_count_slave              | 0        |
| Ndb_api_trans_abort_count_slave               | 0        |
| Ndb_api_trans_close_count_slave               | 0        |
| Ndb_api_pk_op_count_slave                     | 0        |
| Ndb_api_uk_op_count_slave                     | 0        |
| Ndb_api_table_scan_count_slave                | 0        |
| Ndb_api_range_scan_count_slave                | 0        |
| Ndb_api_pruned_scan_count_slave               | 0        |
| Ndb_api_scan_batch_count_slave                | 0        |
| Ndb_api_read_row_count_slave                  | 0        |
| Ndb_api_trans_local_read_row_count_slave      | 0        |
| Ndb_api_wait_exec_complete_count              | 2        |
| Ndb_api_wait_scan_result_count                | 3        |
| Ndb_api_wait_meta_request_count               | 27       |
| Ndb_api_wait_nanos_count                      | 45612023 |
| Ndb_api_bytes_sent_count                      | 992      |
| Ndb_api_bytes_received_count                  | 9640     |
```

```
| Ndb_api_trans_start_count                | 2       |
| Ndb_api_trans_commit_count               | 1       |
| Ndb_api_trans_abort_count                | 0       |
| Ndb_api_trans_close_count                | 2       |
| Ndb_api_pk_op_count                      | 1       |
| Ndb_api_uk_op_count                      | 0       |
| Ndb_api_table_scan_count                 | 1       |
| Ndb_api_range_scan_count                 | 0       |
| Ndb_api_pruned_scan_count                | 0       |
| Ndb_api_scan_batch_count                 | 0       |
| Ndb_api_read_row_count                   | 1       |
| Ndb_api_trans_local_read_row_count       | 1       |
| Ndb_api_event_data_count                 | 0       |
| Ndb_api_event_nondata_count              | 0       |
| Ndb_api_event_bytes_count                | 0       |
+------------------------------------------+---------+
60 rows in set (0.02 sec)
```

これらのステータス変数は、次に示すように、INFORMATION_SCHEMA データベースの SESSION_STATUS および GLOBAL_STATUS テーブルから使用することもできます。

```
mysql> SELECT * FROM INFORMATION_SCHEMA.SESSION_STATUS
    -> WHERE VARIABLE_NAME LIKE 'ndb_api%';
+------------------------------------------+----------------+
| VARIABLE_NAME                            | VARIABLE_VALUE |
+------------------------------------------+----------------+
| NDB_API_WAIT_EXEC_COMPLETE_COUNT_SESSION | 2              |
| NDB_API_WAIT_SCAN_RESULT_COUNT_SESSION   | 0              |
| NDB_API_WAIT_META_REQUEST_COUNT_SESSION  | 1              |
| NDB_API_WAIT_NANOS_COUNT_SESSION         | 8144375        |
| NDB_API_BYTES_SENT_COUNT_SESSION         | 68             |
| NDB_API_BYTES_RECEIVED_COUNT_SESSION     | 84             |
| NDB_API_TRANS_START_COUNT_SESSION        | 1              |
| NDB_API_TRANS_COMMIT_COUNT_SESSION       | 1              |
| NDB_API_TRANS_ABORT_COUNT_SESSION        | 0              |
| NDB_API_TRANS_CLOSE_COUNT_SESSION        | 1              |
| NDB_API_PK_OP_COUNT_SESSION              | 1              |
| NDB_API_UK_OP_COUNT_SESSION              | 0              |
| NDB_API_TABLE_SCAN_COUNT_SESSION         | 0              |
| NDB_API_RANGE_SCAN_COUNT_SESSION         | 0              |
| NDB_API_PRUNED_SCAN_COUNT_SESSION        | 0              |
| NDB_API_SCAN_BATCH_COUNT_SESSION         | 0              |
| NDB_API_READ_ROW_COUNT_SESSION           | 1              |
| NDB_API_TRANS_LOCAL_READ_ROW_COUNT_SESSION | 1            |
| NDB_API_EVENT_DATA_COUNT_INJECTOR        | 0              |
| NDB_API_EVENT_NONDATA_COUNT_INJECTOR     | 0              |
| NDB_API_EVENT_BYTES_COUNT_INJECTOR       | 0              |
| NDB_API_WAIT_EXEC_COMPLETE_COUNT_SLAVE   | 0              |
| NDB_API_WAIT_SCAN_RESULT_COUNT_SLAVE     | 0              |
| NDB_API_WAIT_META_REQUEST_COUNT_SLAVE    | 0              |
| NDB_API_WAIT_NANOS_COUNT_SLAVE           | 0              |
| NDB_API_BYTES_SENT_COUNT_SLAVE           | 0              |
| NDB_API_BYTES_RECEIVED_COUNT_SLAVE       | 0              |
| NDB_API_TRANS_START_COUNT_SLAVE          | 0              |
| NDB_API_TRANS_COMMIT_COUNT_SLAVE         | 0              |
| NDB_API_TRANS_ABORT_COUNT_SLAVE          | 0              |
| NDB_API_TRANS_CLOSE_COUNT_SLAVE          | 0              |
| NDB_API_PK_OP_COUNT_SLAVE                | 0              |
| NDB_API_UK_OP_COUNT_SLAVE                | 0              |
| NDB_API_TABLE_SCAN_COUNT_SLAVE           | 0              |
| NDB_API_RANGE_SCAN_COUNT_SLAVE           | 0              |
| NDB_API_PRUNED_SCAN_COUNT_SLAVE          | 0              |
| NDB_API_SCAN_BATCH_COUNT_SLAVE           | 0              |
| NDB_API_READ_ROW_COUNT_SLAVE             | 0              |
| NDB_API_TRANS_LOCAL_READ_ROW_COUNT_SLAVE | 0              |
| NDB_API_WAIT_EXEC_COMPLETE_COUNT         | 4              |
| NDB_API_WAIT_SCAN_RESULT_COUNT           | 3              |
| NDB_API_WAIT_META_REQUEST_COUNT          | 28             |
| NDB_API_WAIT_NANOS_COUNT                 | 53756398       |
| NDB_API_BYTES_SENT_COUNT                 | 1060           |
| NDB_API_BYTES_RECEIVED_COUNT             | 9724           |
| NDB_API_TRANS_START_COUNT                | 3              |
| NDB_API_TRANS_COMMIT_COUNT               | 2              |
| NDB_API_TRANS_ABORT_COUNT                | 0              |
| NDB_API_TRANS_CLOSE_COUNT                | 3              |
| NDB_API_PK_OP_COUNT                      | 2              |
| NDB_API_UK_OP_COUNT                      | 0              |
| NDB_API_TABLE_SCAN_COUNT                 | 1              |
| NDB_API_RANGE_SCAN_COUNT                 | 0              |
| NDB_API_PRUNED_SCAN_COUNT                | 0              |
```

```
| NDB_API_SCAN_BATCH_COUNT                | 0           |
| NDB_API_READ_ROW_COUNT                  | 2           |
| NDB_API_TRANS_LOCAL_READ_ROW_COUNT      | 2           |
| NDB_API_EVENT_DATA_COUNT                | 0           |
| NDB_API_EVENT_NONDATA_COUNT             | 0           |
| NDB_API_EVENT_BYTES_COUNT               | 0           |
+-----------------------------------------+-------------+
60 rows in set (0.00 sec)

mysql> SELECT * FROM INFORMATION_SCHEMA.GLOBAL_STATUS
    ->     WHERE VARIABLE_NAME LIKE 'ndb_api%';
+-----------------------------------------+----------------+
| VARIABLE_NAME                           | VARIABLE_VALUE |
+-----------------------------------------+----------------+
| NDB_API_WAIT_EXEC_COMPLETE_COUNT_SESSION  | 2            |
| NDB_API_WAIT_SCAN_RESULT_COUNT_SESSION    | 0            |
| NDB_API_WAIT_META_REQUEST_COUNT_SESSION   | 1            |
| NDB_API_WAIT_NANOS_COUNT_SESSION          | 8144375      |
| NDB_API_BYTES_SENT_COUNT_SESSION          | 68           |
| NDB_API_BYTES_RECEIVED_COUNT_SESSION      | 84           |
| NDB_API_TRANS_START_COUNT_SESSION         | 1            |
| NDB_API_TRANS_COMMIT_COUNT_SESSION        | 1            |
| NDB_API_TRANS_ABORT_COUNT_SESSION         | 0            |
| NDB_API_TRANS_CLOSE_COUNT_SESSION         | 1            |
| NDB_API_PK_OP_COUNT_SESSION               | 1            |
| NDB_API_UK_OP_COUNT_SESSION               | 0            |
| NDB_API_TABLE_SCAN_COUNT_SESSION          | 0            |
| NDB_API_RANGE_SCAN_COUNT_SESSION          | 0            |
| NDB_API_PRUNED_SCAN_COUNT_SESSION         | 0            |
| NDB_API_SCAN_BATCH_COUNT_SESSION          | 0            |
| NDB_API_READ_ROW_COUNT_SESSION            | 1            |
| NDB_API_TRANS_LOCAL_READ_ROW_COUNT_SESSION | 1           |
| NDB_API_EVENT_DATA_COUNT_INJECTOR         | 0            |
| NDB_API_EVENT_NONDATA_COUNT_INJECTOR      | 0            |
| NDB_API_EVENT_BYTES_COUNT_INJECTOR        | 0            |
| NDB_API_WAIT_EXEC_COMPLETE_COUNT_SLAVE    | 0            |
| NDB_API_WAIT_SCAN_RESULT_COUNT_SLAVE      | 0            |
| NDB_API_WAIT_META_REQUEST_COUNT_SLAVE     | 0            |
| NDB_API_WAIT_NANOS_COUNT_SLAVE            | 0            |
| NDB_API_BYTES_SENT_COUNT_SLAVE            | 0            |
| NDB_API_BYTES_RECEIVED_COUNT_SLAVE        | 0            |
| NDB_API_TRANS_START_COUNT_SLAVE           | 0            |
| NDB_API_TRANS_COMMIT_COUNT_SLAVE          | 0            |
| NDB_API_TRANS_ABORT_COUNT_SLAVE           | 0            |
| NDB_API_TRANS_CLOSE_COUNT_SLAVE           | 0            |
| NDB_API_PK_OP_COUNT_SLAVE                 | 0            |
| NDB_API_UK_OP_COUNT_SLAVE                 | 0            |
| NDB_API_TABLE_SCAN_COUNT_SLAVE            | 0            |
| NDB_API_RANGE_SCAN_COUNT_SLAVE            | 0            |
| NDB_API_PRUNED_SCAN_COUNT_SLAVE           | 0            |
| NDB_API_SCAN_BATCH_COUNT_SLAVE            | 0            |
| NDB_API_READ_ROW_COUNT_SLAVE              | 0            |
| NDB_API_TRANS_LOCAL_READ_ROW_COUNT_SLAVE  | 0            |
| NDB_API_WAIT_EXEC_COMPLETE_COUNT          | 4            |
| NDB_API_WAIT_SCAN_RESULT_COUNT            | 3            |
| NDB_API_WAIT_META_REQUEST_COUNT           | 28           |
| NDB_API_WAIT_NANOS_COUNT                  | 53756398     |
| NDB_API_BYTES_SENT_COUNT                  | 1060         |
| NDB_API_BYTES_RECEIVED_COUNT              | 9724         |
| NDB_API_TRANS_START_COUNT                 | 3            |
| NDB_API_TRANS_COMMIT_COUNT                | 2            |
| NDB_API_TRANS_ABORT_COUNT                 | 0            |
| NDB_API_TRANS_CLOSE_COUNT                 | 3            |
| NDB_API_PK_OP_COUNT                       | 2            |
| NDB_API_UK_OP_COUNT                       | 0            |
| NDB_API_TABLE_SCAN_COUNT                  | 1            |
| NDB_API_RANGE_SCAN_COUNT                  | 0            |
| NDB_API_PRUNED_SCAN_COUNT                 | 0            |
| NDB_API_SCAN_BATCH_COUNT                  | 0            |
| NDB_API_READ_ROW_COUNT                    | 2            |
| NDB_API_TRANS_LOCAL_READ_ROW_COUNT        | 2            |
| NDB_API_EVENT_DATA_COUNT                  | 0            |
| NDB_API_EVENT_NONDATA_COUNT               | 0            |
| NDB_API_EVENT_BYTES_COUNT                 | 0            |
+-----------------------------------------+----------------+
60 rows in set (0.00 sec)
```

各 Ndb オブジェクトは、それぞれに独自のカウンタを持っています。NDB API アプリケーションは、最適化やモニタリングで使用するためにカウンタの値を読み取ることができます。同時に複数の Ndb オブジェクトを使用

するマルチスレッドクライアントの場合、特定の Ndb_cluster_connection に属するすべての Ndb オブジェクトから、カウンタの集計ビューを取得することもできます。

4 セットのカウンタが表示されます。1 セットは、現在のセッションにのみ適用されます。その他の 3 セットはグローバルです。これは、mysql クライアントでは、これらの値をセッションとグローバルのどちらのステータス変数としても取得できることに関係ありません。つまり、SHOW STATUS を使用して SESSION または GLOBAL キーワードを指定しても、NDB API 統計のステータス変数にレポートされる値は影響を受けず、SESSION_STATUS と GLOBAL_STATUS テーブルのどちらの同等のカラムから値を取得しているかに関係なく、これらの各変数の値は同じです。

- セッションカウンタ (セッションに固有)

  セッションカウンタは、現在のセッションで (のみ) 使用される Ndb オブジェクトに関連します。このようなオブジェクトをほかの MySQL クライアントが使用しても、これらのカウンタは影響を受けません。

  標準の MySQL セッション変数との混同を最小限にするために、これらの NDB API セッションに対応する変数は、先頭に下線を付けた 「_session 変数」と呼んでいます。

- スレーブカウンタ (グローバル)

  このカウンタセットは、レプリケーションスレーブ SQL スレッド (存在する場合) で使用される Ndb オブジェクトに関連します。この mysqld がレプリケーションスレーブとして機能していない場合や、NDB テーブルを使用していない場合、これらのカウントはすべて 0 になります。

  関連するステータス変数は、(先頭に下線を付けた) 「_slave 変数」と呼んでいます。

- インジェクタカウンタ (グローバル)

  インジェクタカウンタは、バイナリログインジェクタスレッドがクラスタイベントを待機するために使用する Ndb オブジェクトに関連します。バイナリログを書き込まないときでも、MySQL Cluster に接続された mysqld プロセスは、スキーマの変更などの一部のイベントを待機し続けます。

  NDB API インジェクタカウンタに対応するステータス変数は、(先頭に下線を付けた) 「_injector 変数」と呼んでいます。

- サーバー (グローバル) カウンタ (グローバル)

  このカウンタセットは、この mysqld によって現在使用されているすべての Ndb オブジェクトに関連します。これには、すべての MySQL クライアントアプリケーション、スレーブ SQL スレッド (存在する場合)、binlog インジェクタ、および NDB ユーティリティースレッドが含まれます。

  これらのカウンタに対応するステータス変数は、「グローバル変数」または 「mysqld レベルの変数」と呼んでいます。

(一般的なプリフィクス Ndb_api とともに) 変数名の部分文字列 session、slave、または injector で追加でフィルタ処理すると、特定の変数セットの値を取得できます。_session 変数の場合は、これを次に示すように実行できます。

```
mysql> SHOW STATUS LIKE 'ndb_api%session';
+-----------------------------------------------+---------+
| Variable_name                                 | Value   |
+-----------------------------------------------+---------+
| Ndb_api_wait_exec_complete_count_session      | 2       |
| Ndb_api_wait_scan_result_count_session        | 0       |
| Ndb_api_wait_meta_request_count_session       | 1       |
| Ndb_api_wait_nanos_count_session              | 8144375 |
| Ndb_api_bytes_sent_count_session              | 68      |
| Ndb_api_bytes_received_count_session          | 84      |
| Ndb_api_trans_start_count_session             | 1       |
| Ndb_api_trans_commit_count_session            | 1       |
| Ndb_api_trans_abort_count_session             | 0       |
| Ndb_api_trans_close_count_session             | 1       |
| Ndb_api_pk_op_count_session                   | 1       |
| Ndb_api_uk_op_count_session                   | 0       |
| Ndb_api_table_scan_count_session              | 0       |
| Ndb_api_range_scan_count_session              | 0       |
| Ndb_api_pruned_scan_count_session             | 0       |
| Ndb_api_scan_batch_count_session              | 0       |
| Ndb_api_read_row_count_session                | 1       |
| Ndb_api_trans_local_read_row_count_session    | 1       |
+-----------------------------------------------+---------+
18 rows in set (0.50 sec)
```

NDB API mysqld レベルのステータス変数のリストを取得するには、次のように、ndb_api で始まり、_count で終わる変数名でフィルタ処理します。

```
mysql> SELECT * FROM INFORMATION_SCHEMA.SESSION_STATUS
    ->     WHERE VARIABLE_NAME LIKE 'ndb_api%count';
+--------------------------------------+----------------+
| VARIABLE_NAME                        | VARIABLE_VALUE |
+--------------------------------------+----------------+
| NDB_API_WAIT_EXEC_COMPLETE_COUNT     | 4              |
| NDB_API_WAIT_SCAN_RESULT_COUNT       | 3              |
| NDB_API_WAIT_META_REQUEST_COUNT      | 28             |
| NDB_API_WAIT_NANOS_COUNT             | 53756398       |
| NDB_API_BYTES_SENT_COUNT             | 1060           |
| NDB_API_BYTES_RECEIVED_COUNT         | 9724           |
| NDB_API_TRANS_START_COUNT            | 3              |
| NDB_API_TRANS_COMMIT_COUNT           | 2              |
| NDB_API_TRANS_ABORT_COUNT            | 0              |
| NDB_API_TRANS_CLOSE_COUNT            | 3              |
| NDB_API_PK_OP_COUNT                  | 2              |
| NDB_API_UK_OP_COUNT                  | 0              |
| NDB_API_TABLE_SCAN_COUNT             | 1              |
| NDB_API_RANGE_SCAN_COUNT             | 0              |
| NDB_API_PRUNED_SCAN_COUNT            | 0              |
| NDB_API_SCAN_BATCH_COUNT             | 0              |
| NDB_API_READ_ROW_COUNT               | 2              |
| NDB_API_TRANS_LOCAL_READ_ROW_COUNT   | 2              |
| NDB_API_EVENT_DATA_COUNT             | 0              |
| NDB_API_EVENT_NONDATA_COUNT          | 0              |
| NDB_API_EVENT_BYTES_COUNT            | 0              |
+--------------------------------------+----------------+
21 rows in set (0.09 sec)
```

4 セットすべてのステータス変数に、すべてのカウンタが反映されるとはかぎりません。イベントカウンタ DataEventsRecvdCount、NondataEventsRecvdCount、および EventBytesRecvdCount の場合は、_injector と mysqld レベルの NDB API ステータス変数のみを使用できます。

```
mysql> SHOW STATUS LIKE 'ndb_api%event%';
+---------------------------------------+-------+
| Variable_name                         | Value |
+---------------------------------------+-------+
| Ndb_api_event_data_count_injector     | 0     |
| Ndb_api_event_nondata_count_injector  | 0     |
| Ndb_api_event_bytes_count_injector    | 0     |
| Ndb_api_event_data_count              | 0     |
| Ndb_api_event_nondata_count           | 0     |
| Ndb_api_event_bytes_count             | 0     |
+---------------------------------------+-------+
6 rows in set (0.00 sec)
```

次に示すように、その他の NDB API カウンタには、_injector ステータス変数が実装されていません。

```
mysql> SHOW STATUS LIKE 'ndb_api%injector%';
+---------------------------------------+-------+
| Variable_name                         | Value |
+---------------------------------------+-------+
| Ndb_api_event_data_count_injector     | 0     |
| Ndb_api_event_nondata_count_injector  | 0     |
| Ndb_api_event_bytes_count_injector    | 0     |
+---------------------------------------+-------+
3 rows in set (0.00 sec)
```

ステータス変数の名前は、対応するカウンタの名前に簡単に関連付けることができます。次の表に、各 NDB API 統計カウンタを、説明およびこのカウンタに対応する MySQL サーバーステータス変数の名前とともに一覧表示しています。

カウンタ名	説明
	ステータス変数 (統計タイプ別): • セッション • スレーブ • インジェクタ • サーバー
WaitExecCompleteCount	操作の実行が完了するまで待機する間にスレッドがブロックされた回数。すべての execute() 呼び出しと、クライアントから見えない BLOB および自動インクリメント操作の暗黙的な実行が含まれます。
	• Ndb_api_wait_exec_complete_count_session • Ndb_api_wait_exec_complete_count_slave • [none] • Ndb_api_wait_exec_complete_count
WaitScanResultCount	追加の結果やスキャンが閉じるまで待機するなど、スキャンベースの信号を待機する間にスレッドがブロックされた回数。
	• Ndb_api_wait_scan_result_count_session • Ndb_api_wait_scan_result_count_slave • [none] • Ndb_api_wait_scan_result_count
WaitMetaRequestCount	メタベースの信号を待機する間にスレッドがブロックされた回数。これは、DDL 操作またはエポックが開始される (または終了する) まで待機しているときに発生する可能性があります。
	• Ndb_api_wait_meta_request_count_session • Ndb_api_wait_meta_request_count_slave • [none] • Ndb_api_wait_meta_request_count
WaitNanosCount	データノードからの何らかのタイプの信号の待機にかかった合計時間 (ナノ秒)。
	• Ndb_api_wait_nanos_count_session • Ndb_api_wait_nanos_count_slave • [none] • Ndb_api_wait_nanos_count
BytesSentCount	データノードに送信されたデータ量 (バイト単位)。
	• Ndb_api_bytes_sent_count_session • Ndb_api_bytes_sent_count_slave • [none] • Ndb_api_bytes_sent_count
BytesRecvdCount	データノードから受信されたデータ量 (バイト単位)。
	• Ndb_api_bytes_received_count_session • Ndb_api_bytes_received_count_slave • [none]

カウンタ名	説明
	ステータス変数 (統計タイプ別): • セッション • スレーブ • インジェクタ • サーバー
	• Ndb_api_bytes_received_count
TransStartCount	開始されたトランザクションの数。
	• Ndb_api_trans_start_count_session • Ndb_api_trans_start_count_slave • [none] • Ndb_api_trans_start_count
TransCommitCount	コミットされたトランザクションの数。
	• Ndb_api_trans_commit_count_session • Ndb_api_trans_commit_count_slave • [none] • Ndb_api_trans_commit_count
TransAbortCount	中止されたトランザクションの数。
	• Ndb_api_trans_abort_count_session • Ndb_api_trans_abort_count_slave • [none] • Ndb_api_trans_abort_count
TransCloseCount	中止されたトランザクションの数。(この値は、TransCommitCount と TransAbortCount との合計より大きい場合があります。)
	• Ndb_api_trans_close_count_session • Ndb_api_trans_close_count_slave • [none] • Ndb_api_trans_close_count
PkOpCount	主キーに基づいた操作または主キーを使用した操作の数。このカウントには、BLOB 部分テーブル操作、暗黙的な口ック解除操作、自動インクリメント操作、および通常 MySQL クライアントから見える主キー操作が含まれます。
	• Ndb_api_pk_op_count_session • Ndb_api_pk_op_count_slave • [none] • Ndb_api_pk_op_count
UkOpCount	一意のキーに基づいた操作または一意のキーを使用した操作の数。
	• Ndb_api_uk_op_count_session • Ndb_api_uk_op_count_slave • [none]

カウンタ名	説明
	ステータス変数 (統計タイプ別): • セッション • スレーブ • インジェクタ • サーバー
	• Ndb_api_uk_op_count
TableScanCount	開始されたテーブルスキャンの数。これには、内部テーブルのスキャンが含まれます。
	• Ndb_api_table_scan_count_session • Ndb_api_table_scan_count_slave • [none] • Ndb_api_table_scan_count
RangeScanCount	開始された範囲スキャンの数。
	• Ndb_api_range_scan_count_session • Ndb_api_range_scan_count_slave • [none] • Ndb_api_range_scan_count
PrunedScanCount	単一パーティションにプルーニングされたスキャンの数。
	• Ndb_api_pruned_scan_count_session • Ndb_api_pruned_scan_count_slave • [none] • Ndb_api_pruned_scan_count
ScanBatchCount	受信された行のバッチの数。(このコンテキストで、バッチとは単一フラグメントからのスキャン結果のセットです。)
	• Ndb_api_scan_batch_count_session • Ndb_api_scan_batch_count_slave • [none] • Ndb_api_scan_batch_count
ReadRowCount	読み取られた行の合計数。主キー、一意のキー、またはスキャン操作を使用して読み取られた行が含まれます。
	• Ndb_api_read_row_count_session • Ndb_api_read_row_count_slave • [none] • Ndb_api_read_row_count
TransLocalReadRowCount	トランザクションが実行された同じデータノードから読み取られた行数。
	• Ndb_api_trans_local_read_row_count_session • Ndb_api_trans_local_read_row_count_slave • [none]

カウンタ名	説明
	ステータス変数 (統計タイプ別): • セッション • スレーブ • インジェクタ • サーバー
	• Ndb_api_trans_local_read_row_count
DataEventsRecvdCount	受信された行変更イベントの数。
	• [none] • [none] • Ndb_api_event_data_count_injector • Ndb_api_event_data_count
NondataEventsRecvdCount	受信された行変更イベント以外のイベントの数。
	• [none] • [none] • Ndb_api_event_nondata_count_injector • Ndb_api_event_nondata_count
EventBytesRecvdCount	受信されたイベントのバイト数。
	• [none] • [none] • Ndb_api_event_bytes_count_injector • Ndb_api_event_bytes_count

コミットされたトランザクションのすべてのカウント、つまり TransCommitCount カウンタのステータス変数を確認するには、次のように、SHOW STATUS の結果を部分文字列 trans_commit_count でフィルタ処理できます。

```
mysql> SHOW STATUS LIKE '%trans_commit_count%';
+-------------------------------------+-------+
| Variable_name                       | Value |
+-------------------------------------+-------+
| Ndb_api_trans_commit_count_session | 1     |
| Ndb_api_trans_commit_count_slave   | 0     |
| Ndb_api_trans_commit_count         | 2     |
+-------------------------------------+-------+
3 rows in set (0.00 sec)
```

ここから、現在の mysql クライアントセッションで 1 つのトランザクションがコミットされ、この mysqld で、それが最後に再起動されてから、2 つのトランザクションがコミットされたことを判断できます。

ステートメントを実行した直前と直後で、対応する _session ステータス変数の値を比較すると、特定の SQL ステートメントによって、さまざまな NDB API カウンタがどのように増分されているかを確認できます。この例では、SHOW STATUS から初期値を取得したあとに、test データベースに、単一のカラムを持つ t という名前の NDB テーブルを作成します。

```
mysql> SHOW STATUS LIKE 'ndb_api%session%';
+-----------------------------------------+--------+
| Variable_name                           | Value  |
+-----------------------------------------+--------+
| Ndb_api_wait_exec_complete_count_session   | 2      |
| Ndb_api_wait_scan_result_count_session     | 0      |
| Ndb_api_wait_meta_request_count_session    | 3      |
| Ndb_api_wait_nanos_count_session           | 820705 |
| Ndb_api_bytes_sent_count_session           | 132    |
| Ndb_api_bytes_received_count_session       | 372    |
```

```
| Ndb_api_trans_start_count_session         | 1     |
| Ndb_api_trans_commit_count_session        | 1     |
| Ndb_api_trans_abort_count_session         | 0     |
| Ndb_api_trans_close_count_session         | 1     |
| Ndb_api_pk_op_count_session               | 1     |
| Ndb_api_uk_op_count_session               | 0     |
| Ndb_api_table_scan_count_session          | 0     |
| Ndb_api_range_scan_count_session          | 0     |
| Ndb_api_pruned_scan_count_session         | 0     |
| Ndb_api_scan_batch_count_session          | 0     |
| Ndb_api_read_row_count_session            | 1     |
| Ndb_api_trans_local_read_row_count_session | 1     |
+-------------------------------------------+-------+
18 rows in set (0.00 sec)

mysql> USE test;
Database changed
mysql> CREATE TABLE t (c INT) ENGINE NDBCLUSTER;
Query OK, 0 rows affected (0.85 sec)
```

この時点で、次に示す (出力で変更された行を強調表示しています) ように、新しい SHOW STATUS ステートメントを実行し、変更を確認できます。

```
mysql> SHOW STATUS LIKE 'ndb_api%session%';
+-------------------------------------------+-----------+
| Variable_name                             | Value     |
+-------------------------------------------+-----------+
| Ndb_api_wait_exec_complete_count_session  | 8         |
| Ndb_api_wait_scan_result_count_session    | 0         |
| Ndb_api_wait_meta_request_count_session   | 17        |
| Ndb_api_wait_nanos_count_session          | 706871709 |
| Ndb_api_bytes_sent_count_session          | 2376      |
| Ndb_api_bytes_received_count_session      | 3844      |
| Ndb_api_trans_start_count_session         | 4         |
| Ndb_api_trans_commit_count_session        | 4         |
| Ndb_api_trans_abort_count_session         | 0         |
| Ndb_api_trans_close_count_session         | 4         |
| Ndb_api_pk_op_count_session               | 6         |
| Ndb_api_uk_op_count_session               | 0         |
| Ndb_api_table_scan_count_session          | 0         |
| Ndb_api_range_scan_count_session          | 0         |
| Ndb_api_pruned_scan_count_session         | 0         |
| Ndb_api_scan_batch_count_session          | 0         |
| Ndb_api_read_row_count_session            | 2         |
| Ndb_api_trans_local_read_row_count_session | 1         |
+-------------------------------------------+-----------+
18 rows in set (0.00 sec)
```

同様に、t に行を挿入することで発生した NDB API 統計カウンタの変更も確認できます。次に示すように、行を挿入してから、前の例で使用したものと同じ SHOW STATUS ステートメントを実行します。

```
mysql> INSERT INTO t VALUES (100);
Query OK, 1 row affected (0.00 sec)

mysql> SHOW STATUS LIKE 'ndb_api%session%';
+-------------------------------------------+-----------+
| Variable_name                             | Value     |
+-------------------------------------------+-----------+
| Ndb_api_wait_exec_complete_count_session  | 11        |
| Ndb_api_wait_scan_result_count_session    | 6         |
| Ndb_api_wait_meta_request_count_session   | 20        |
| Ndb_api_wait_nanos_count_session          | 707370418 |
| Ndb_api_bytes_sent_count_session          | 2724      |
| Ndb_api_bytes_received_count_session      | 4116      |
| Ndb_api_trans_start_count_session         | 7         |
| Ndb_api_trans_commit_count_session        | 6         |
| Ndb_api_trans_abort_count_session         | 0         |
| Ndb_api_trans_close_count_session         | 7         |
| Ndb_api_pk_op_count_session               | 8         |
| Ndb_api_uk_op_count_session               | 0         |
| Ndb_api_table_scan_count_session          | 1         |
| Ndb_api_range_scan_count_session          | 0         |
| Ndb_api_pruned_scan_count_session         | 0         |
| Ndb_api_scan_batch_count_session          | 0         |
| Ndb_api_read_row_count_session            | 3         |
| Ndb_api_trans_local_read_row_count_session | 2         |
+-------------------------------------------+-----------+
18 rows in set (0.00 sec)
```

これらの結果から、いくつかのことを観察できます。

- 明示的な主キーなしで t を作成しましたが、その際に 5 つの主キー操作が実行されました (Ndb_api_pk_op_count_session の「前」と「後」の値の差、つまり 6 から 1 を引く)。これは、NDB ストレージエンジンを使用しているすべてのテーブルの機能である非表示の主キーの作成を反映しています。

- Ndb_api_wait_nanos_count_session の連続した値を比較すると、CREATE TABLE ステートメントを実装している NDB API 操作が、INSERT によって実行された操作 (707370418 - 706871709 = 498709 ナノ秒、つまり約 0.0005 秒) よりも大幅に長い時間 (706871709 - 820705 = 706051004 ナノ秒、つまり約 0.7 秒)、データノードからの応答を待機したことを確認できます。mysql クライアントでこれらのステートメントについてレポートされた実行時間は、これらの数値に大まかに関連しています。

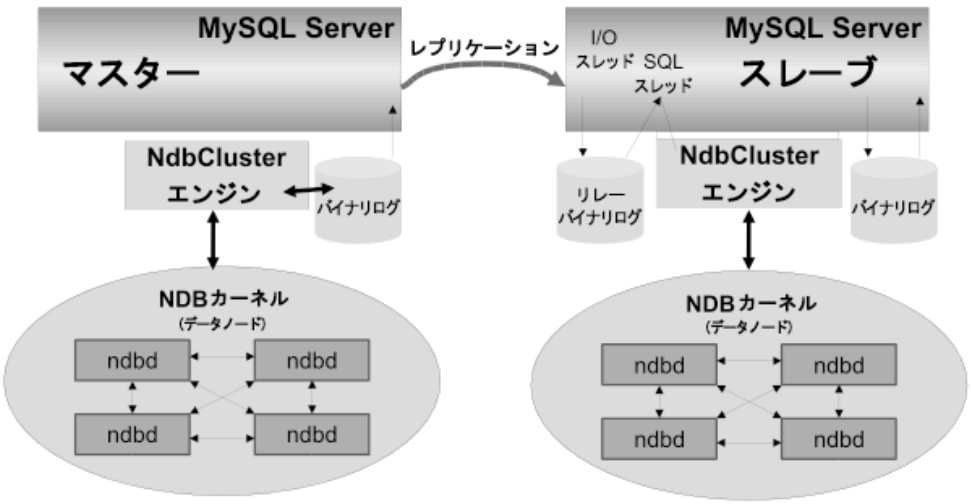
  十分な (ナノ秒) 時間分解能を備えていないプラットフォームでは、SQL ステートメントが非常に高速で実行されたために、WaitNanosCount NDB API カウンタの値の小さな変更が、Ndb_api_wait_nanos_count_session、Ndb_api_wait_nanos_count_slave、または Ndb_api_wait_nanos_count の値に表れないことがあります。

- Ndb_api_read_row_count_session と Ndb_api_trans_local_read_row_count_session の値が増加したことを反映して、INSERT ステートメントによって ReadRowCount と TransLocalReadRowCount の両方の NDB API 統計カウンタが増分されています。

18.6 MySQL Cluster レプリケーション

MySQL Cluster は、通常は単に「レプリケーション」と呼ばれる非同期レプリケーションをサポートしています。このセクションでは、MySQL Cluster として動作する、あるグループのコンピュータが、別のコンピュータまたはコンピュータのグループに複製する構成を、セットアップしたり管理したりする方法について説明します。標準の MySQL レプリケーションに関してはこのマニュアルで別途説明しています。(第17章「レプリケーション」を参照してください。)

通常 (非クラスタ) のレプリケーションは、「マスター」サーバーと 「スレーブ」サーバー、複製される操作とデータの元になるマスター、およびその受け手であるスレーブによって行われています。概念的にはレプリケーションは非常に似ていますが、MySQL Cluster では、実際には複雑さが増す可能性があります。2 つの完全なクラスタ間のレプリケーションを含む多くの異なる構成を対象にするために、MySQL Cluster が拡張される場合があるためです。MySQL Cluster 自身は、クラスタリングの機能を NDB ストレージエンジンに依存していますが、複製されたテーブルのスレーブのコピーにストレージエンジンとして NDB を使用する必要はありません (NDB から別のストレージエンジンへのレプリケーションを参照してください)。ただし、可用性を最大にするため、ある MySQL Cluster から別の MySQL Cluster に複製できます (複製することが望ましい)。また、ここで説明するのはこのシナリオであり、次の図で示すとおりです。



このシナリオでは、レプリケーションプロセスは、マスタークラスタの連続的な状態のログが取られ、スレーブクラスタに保存されるというプロセスです。このプロセスは NDB バイナリログインジェクタスレッドとして知られる特別なスレッドによって実行されます。このスレッドは各 MySQL サーバーで動作し、バイナリログ (binlog) を作成します。このスレッドは、バイナリログを作成するクラスタのすべての変更 (MySQL Server を介して影響を受けた変更だけではありません) を、シリアライゼーションの正しい順序でバイナリログに挿入します。ここでは、MySQL のレプリケーションマスターとレプリケーションスレーブのサーバーをレプリケーションサーバーまたはレプリケーションノードと呼び、そのサーバー間のデータフローまたは通信回線をレプリケーションチャネルと呼びます。

MySQL Cluster および MySQL Cluster レプリケーションを使用したポイントインタイムリカバリの実行の情報については、セクション18.6.9.2「MySQL Cluster レプリケーションを使用したポイントインタイムリカバリ」を参照してください。

NDB API _slave ステータス変数　NDB API カウンタは、MySQL Cluster のレプリケーションスレーブで高度なモニタリング監視機能を提供できます。これらは NDB 統計の _slave ステータス変数として実装されます。この内容が見られるのは、SHOW STATUS の出力の中、または MySQL Cluster レプリケーションでスレーブとして動作している MySQL Server に接続された mysql クライアントセッションの SESSION_STATUS または GLOBAL_STATUS テーブルに対してクエリーを発行した結果の中です。複製された NDB テーブルに作用するステートメントの実行の前後でこれらのステータス変数の値を比較することで、NDB API レベルでスレーブによって行われる、対応したアクションを監視できます。これにより、MySQL Cluster レプリケーションのモニタリングまたはトラブルシューティングのときに役に立つ場合があります。追加情報については、セクション18.5.15「NDB API 統計のカウンタと変数」を参照してください。

NDB テーブルから非 NDB テーブルへのレプリケーション　マスターとして動作する MySQL Cluster から、スレーブ mysqld で InnoDB または MyISAM などのほかの MySQL ストレージエンジンを使用するテーブルに NDB テーブルを複製できます。これには多くの条件が適用されます。詳細については、NDB から別のストレージエンジンへのレプリケーションおよびNDB から非トランザクションストレージエンジンへのレプリケーションを参照してください。

18.6.1 MySQL Cluster レプリケーション: 略語と記号

このセクションを通じて、マスターとスレーブのクラスタ、およびクラスタまたはクラスタノードで実行されるプロセスとコマンドを参照する場合に、次の略語と記号を使用します。

記号または略語	説明 (... を参照)
M	(プライマリ) レプリケーションマスターとしての役割を果たすクラスタ
S	(プライマリ) レプリケーションスレーブとしての役割を果たすクラスタ
shellM>	マスタークラスタで発行されるシェルコマンド
mysqlM>	マスタークラスタの SQL ノードとして動作する単一の MySQL サーバーで発行された MySQL クライアントコマンド
mysqlM*>	レプリケーションマスターのクラスタに参加しているすべての SQL ノードで発行される MySQL クライアントのコマンド
shellS>	スレーブクラスタで発行されるシェルコマンド
mysqlS>	スレーブクラスタで SQL ノードとして動作している単一の MySQL サーバーで発行される MySQLクライアントのコマンド
mysqlS*>	レプリケーションスレーブのクラスタに参加しているすべての SQL ノードで発行される MySQL クライアントのコマンド
C	プライマリレプリケーションチャネル
C'	セカンダリレプリケーションチャネル
M'	セカンダリレプリケーションマスター
S'	セカンダリレプリケーションスレーブ

18.6.2 MySQL Cluster レプリケーションの一般要件

レプリケーションチャネルには、レプリケーションサーバーとして動作する 2 台の MySQL サーバー (マスターとスレーブ用に各 1 台) が必要です。たとえば、2 つのレプリケーションチャネルを持つレプリケーションを設定する場合 (冗長性の確保に予備のチャネルを用意)、合計で 4 つのレプリケーションノード (クラスタごとに 2 つのノード) になります。

このセクション以降で説明する MySQL Cluster のレプリケーションは、行ベースのレプリケーションに依存しています。つまり、レプリケーションマスターの MySQL サーバーは、セクション18.6.6「MySQL Cluster レプリケーションの起動 (レプリケーションチャネルが 1 つ)」で説明のとおり、--binlog-format=ROW または --binlog-format=MIXED で動作している必要があります。行ベースのレプリケーションの一般的な情報は、セクション17.1.2「レプリケーション形式」を参照してください。

> **重要**
>
> --binlog-format=STATEMENTで MySQL Cluster レプリケーションを使用しようとすると、マスターの ndb_binlog_index テーブルおよびスレーブの ndb_apply_status テーブ

> ルの epoch カラムが更新されないため、レプリケーションは正常に動作しません (セクション18.6.4「MySQL Cluster レプリケーションスキーマとテーブル」を参照してください)。その代わり、レプリケーションマスターとして動作する MySQL サーバーでの更新のみがスレーブに伝播され、マスタークラスタのほかの SQL ノードからの更新は複製されません。
>
> MySQL Cluster NDB 7.3 の --binlog-format オプションのデフォルト値は MIXED です。

どちらかのクラスタのレプリケーションに使用される各 MySQL サーバーは、どちらかのクラスタに参加している MySQL レプリケーションサーバーの中で一意に識別される必要があります (マスターおよびスレーブの両方のクラスタにあるレプリケーションサーバーは同じ ID を共有できません)。これは、--server-id=id オプションを使用して各 SQL ノードを起動して、実行できます。ここで、id は一意の整数です。必須ではありませんが、ここでは、すべての MySQL Cluster のバイナリは同じリリースバージョンのバイナリであるものとします。

MySQL レプリケーションでは通常、使用するレプリケーションプロトコルおよびサーバーがサポートする SQL 機能セットの両方のバージョンに関与する両方の MySQL サーバー (mysqld プロセス) に互換性がある必要があります (セクション17.4.2「MySQL バージョン間のレプリケーション互換性」を参照してください)。これは、MySQL Cluster のバイナリと MySQL Server 5.6 のディストリビューションのバイナリでの違いとして、MySQL Cluster のレプリケーションでは、両方の mysqld バイナリを MySQL Cluster ディストリビューションのバイナリにするという追加の要件があるためです。mysqld サーバーの互換性を確保するもっとも単純で容易な方法は、マスターとスレーブのすべての mysqld バイナリに、同じ MySQL Cluster のディストリビューションを使用することです。

スレーブのサーバーまたはクラスタはマスターのレプリケーション専用であり、ほかのデータは保存されていないものとします。

> 注記
>
> ステートメントベースのレプリケーションを使用すると MySQL Cluster を複製できます。ただし、この場合、次の制限が適用されます。
>
> - マスターとして動作しているクラスタでのデータ行へのすべての更新は、単一の MySQL サーバーに対して行う必要があります。
> - 複数の同時 MySQL レプリケーションプロセスを使用してクラスタを複製することはできません。
> - SQL レベルで行われた変更のみが複製されます。
>
> これらは、行ベースのレプリケーションと対照的に、ステートメントベースのレプリケーションのその他の制限事項に追加されます。2 つのレプリケーション形式の間の違いに関する詳細な情報については、セクション17.1.2.1「ステートメントベースおよび行ベースレプリケーションのメリットとデメリット」を参照してください。

18.6.3 MySQL Cluster レプリケーションの既知の問題

このセクションでは、MySQL Cluster NDB 7.3 でレプリケーションを使用する場合の既知の問題や課題について説明します。

マススレーブの接続の喪失　接続の喪失は、レプリケーションマスターの SQL ノードとレプリケーションスレーブの SQL ノードとの間、またはレプリケーションマスターの SQL ノードとマスタークラスタのデータノードとの間のいずれかで発生する可能性があります。後者の場合にこの発生は、物理的な接続が喪失した結果 (たとえば、ネットワークケーブルの破損など) だけではなく、データノードイベントのバッファーのオーバーフローによる可能性もあります。SQL ノードは、応答が遅すぎると、クラスタでドロップされる可能性があります (これは、MaxBufferedEpochs および TimeBetweenEpochs 構成パラメータを調整することで、ある程度まで制御が可能です)。これが起こると、新しいデータをレプリケーションマスターのバイナリログに記録せずに、マスタークラスタに挿入できます。このため、高可用性を保証するには、スレーブクラスタとマスター間の同期の維持に必要な場合、セカンダリレプリケーションチャネルへのフェイルオーバー、バックアップレプリケーションチャネルの維持、プライマリチャネルのモニターが非常に重要です。MySQL Cluster は、自身でこのようなモニタリングを実行するようには設計されていないため、外部のアプリケーションが必要です。

レプリケーションマスターは、マスタークラスタに接続したり、再接続をしたりするときに「ギャップ」イベントを発行します。(ギャップイベントは一種の「インシデントイベント」であり、データベースの内容に影響を与えるが、一連の変更として容易に表現できないインシデントが発生したことを示します。インシデントの例は、サーバーのクラッシュ、データベースの再同期、(複数の) ソフトウェアの更新、(複数の) ハードウェアの更新などです。)スレーブは、レプリケーションログ内のギャップを検出すると、エラーメッセージを出して停止します。このメッセージは SHOW SLAVE STATUS の出力に表示され、SQL スレッドがレプリケーションスト

リームに登録されたインシデントによって停止し、手動による介入が必要なことを示します。このような状況での対処の詳細については、セクション18.6.8「MySQL Cluster レプリケーションを使用したフェイルオーバーの実装」を参照してください。

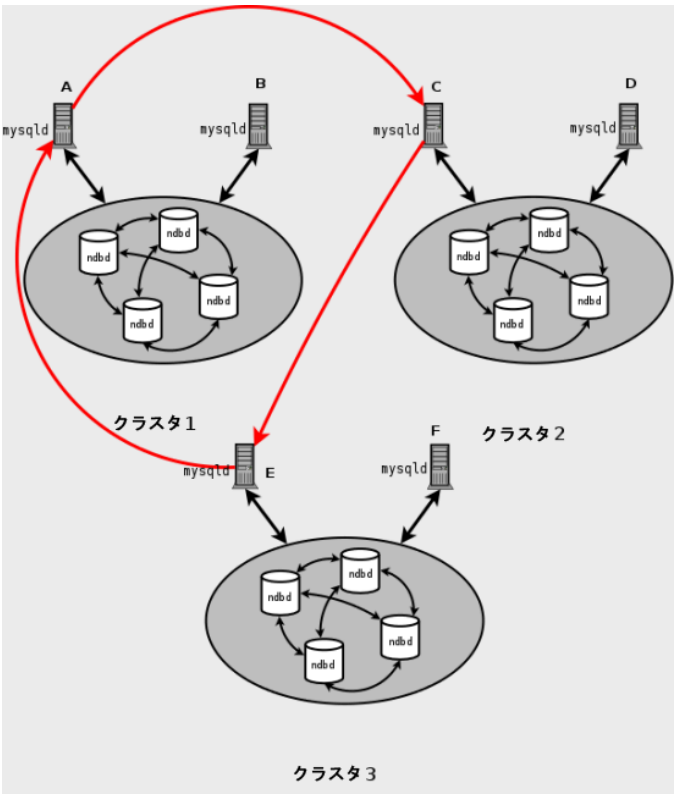
> **重要**
>
> MySQL Cluster は自身でレプリケーションのステータスをモニターしたり、フェイルオーバーの準備をしたりするようには設計されていないため、高可用性がスレーブのサーバーまたはクラスタの要件である場合、複数のレプリケーションラインをセットアップし、プライマリのレプリケーションラインでマスターの mysqld をモニターし、必要に応じてセカンダリラインへのフェイルオーバーの準備をする必要があります。これは、手動や、場合によってはサードパーティーのアプリケーションによって行う必要があります。この種のセットアップの実装に関する情報については、セクション18.6.7「2 つのレプリケーションチャネルを使用する MySQL Cluster レプリケーション」および セクション18.6.8「MySQL Cluster レプリケーションを使用したフェイルオーバーの実装」を参照してください。
>
> ただし、スタンドアロンの MySQL サーバーから MySQL Cluster に複製している場合は、通常 1 つのチャネルで十分です。

循環レプリケーション　　MySQL Cluster レプリケーションは、次の例で示すように、循環レプリケーションをサポートしています。レプリケーションのセットアップは、番号が 1、2、3 の 3 つの MySQL Cluster で構成され、クラスタ 1 はクラスタ 2 のレプリケーションマスターとして動作し、クラスタ 2 はクラスタ 3 のマスターとして動作し、クラスタ 3 はクラスタ 1 のマスターとして動作して循環を完成します。各 MySQL Cluster は 2 つの SQL ノードを持ち、SQL ノード A と B はクラスタ 1 に属し、SQL ノード C と D はクラスタ 2 に属し、SQL ノード E と F はクラスタ 3 に属しています。

これらのクラスタを使用する循環レプリケーションは、次の条件を満たすかぎり、サポートされます。

- すべてのマスターとスレーブの SQL ノードは同じ
- レプリケーションのマスターおよびスレーブとして動作するすべての SQL ノードが --log-slave-updates オプションを使用して起動される

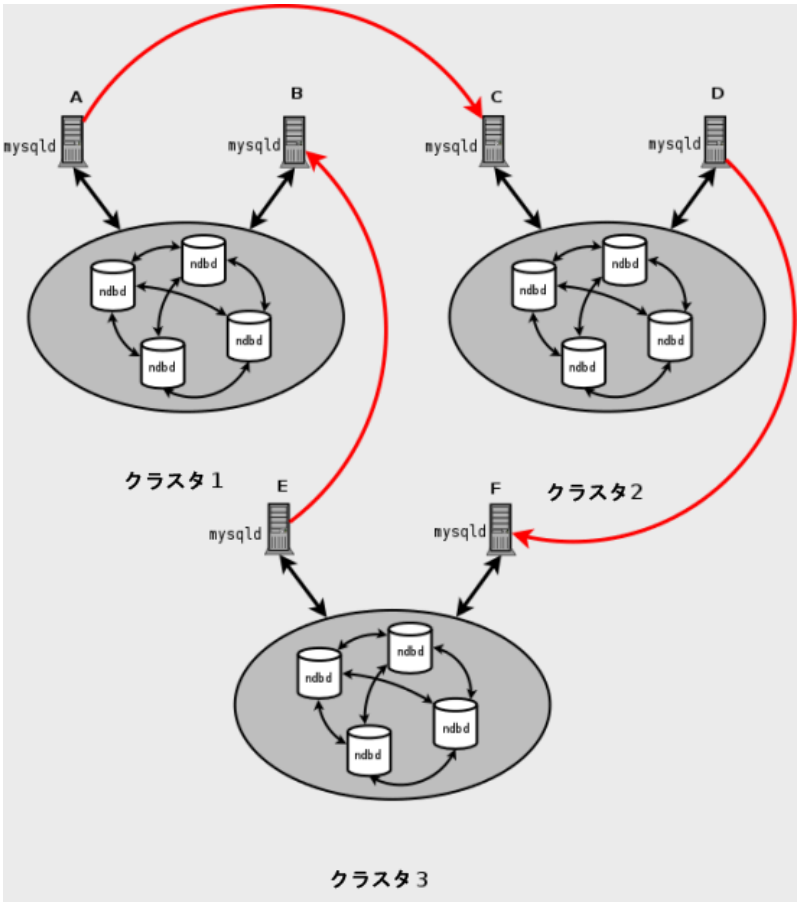
このタイプの循環レプリケーションのセットアップは、次の図に示すとおりです。



このシナリオでは、クラスタ 1 の SQL ノード A はクラスタ 2 の SQL ノード C に複製し、SQL ノード C はクラスタ 3 の SQL ノード E に複製し、SQL ノード E は SQL ノード A に複製します。言い換えると、レプリケー

ションライン (図中の赤の矢印) は、レプリケーションのマスターおよびスレーブとして使用されるすべての SQL ノードを直接接続します。

ここで示すように、すべてのマスター SQL ノードが必ずしもスレーブではない循環レプリケーションをセットアップすることも可能です。



この場合、各クラスタの異なる SQL ノードはレプリケーションのマスターおよびスレーブとして使用されます。ただし、SQL ノードを --log-slave-updates を使用して起動する必要はありません。レプリケーションのライン (これも図中の赤の矢印) が連続的でない MySQL Cluster のこのタイプの循環レプリケーションスキームは、可能性はありますが、まだ完全にはテストされていないため、実験的なものだと考えるようにしてください。

> 注記
>
> NDB ストレージエンジンはIDEMPOTENT 実行モードを使用します。このモードでは、このモードでない場合に MySQL Cluster の循環レプリケーションを中断する重複キーなどのエラーを抑止します。これは、グローバル slave_exec_mode システム変数を IDEMPOTENT に設定することと同等です。これは、MySQL Cluster を使用する場合のマルチマスターレプリケーションにも必要です。(Bug #31609)
>
> 多重モードは MySQL Cluster を使用する場合のマルチマスターレプリケーションにも必須ですが (Bug #31609)、MySQL Cluster のレプリケーションで slave_exec_mode を設定する必要はありません。MySQL Cluster でこの設定を自動的に行い、この変数の明示的な設定への試行を無視するためです。

MySQL Cluster レプリケーションと主キー　ノード障害が発生した場合、重複行が挿入される可能性があるため、主キーがない NDB テーブルのレプリケーションでエラーが発生する可能性があります。このため、複製されるすべての NDB テーブルに主キーを持つことを強くお勧めします。

MySQL Cluster レプリケーションと一意キー　古いバージョンの MySQL Cluster では、NDB テーブルの一意キーのカラムの値を更新する操作が行われると、複製時に重複キーエラーになる場合がありました。NDB テーブル間のレプリケーションに関するこの問題は、すべてのテーブル行の更新が実行されるまで一意キーのチェックを保留することで解決されます。

この方法で制約を保留することは、現在 NDB でのみサポートされています。このため、NDB から別のストレージエンジン (MyISAM や InnoDB など) に複製する場合の一意キーの更新は、まだサポートされていません。

一意キーの更新のチェックを保留しないレプリケーションが NDB テーブル (t など) を使用して説明が可能な場合に発生する問題は、ここで示すように、マスターで発生し、移入されます (および一意キーの更新の保留をサポートしていないスレーブに複製されます)。

```
CREATE TABLE t (
    p INT PRIMARY KEY,
    c INT,
    UNIQUE KEY u (c)
)  ENGINE NDB;

INSERT INTO t
    VALUES (1,1), (2,2), (3,3), (4,4), (5,5);
```

t での次の UPDATE ステートメントはマスターで成功します。これは、影響を受ける行が ORDER BY オプションで指定される順番で処理され、テーブル全体に実行されるためです。

```
UPDATE t SET c = c - 1 ORDER BY p;
```

しかしスレーブでは、同じステートメントが重複キーエラーなどの制約違反で失敗しました。これは、行の更新の順序付けが、テーブル全体に対して行われたのではなく、一回に 1 つのパーティションに対して行われたためです。

> **注記**
>
> 各 NDB テーブルは、作成時に、キーで無条件にパーティション化されます。詳細については、セクション19.2.5「KEY パーティショニング」を参照してください。

GTID は未サポート　グローバルトランザクション ID を使用するレプリケーションは、NDB ストレージエンジンと互換性がなく、サポートされていません。GTID を有効にすると、MySQL Cluster レプリケーションが失敗する可能性があります。

--initial での再起動　--initial オプションを使用してクラスタを再起動すると、GCI と エポック番号の順序が 0 からやり直されます。(これは一般的に MySQL Cluster に当てはまり、MySQL Cluster を含むレプリケーションシナリオに限定されているわけではありません。)この場合、レプリケーションに関与する MySQL サーバーは再起動されます。この後、RESET MASTER および RESET SLAVE ステートメントを使用して、無効な ndb_binlog_index および ndb_apply_status テーブルをそれぞれクリアしてください。

NDB から別のストレージエンジンへのレプリケーション　マスターの NDB テーブルを、異なるストレージエンジンを使用するスレーブのテーブルに複製できますが、次に記載した制限事項を考慮してください。

- マルチマスターと循環レプリケーションはサポートされていません (これを機能させるには、マスターとスレーブの両方のテーブルで NDB ストレージエンジンを使用する必要があります)。
- スレーブのテーブルにバイナリロギングを実行していないストレージエンジンを使用するには、特別な処理が必要です。
- スレーブのテーブルに非トランザクションのストレージエンジンを使用する場合も、特別な処理が必要です。
- マスターの mysqld は --ndb-log-update-as-write=0 または --ndb-log-update-as-write=OFF で起動する必要があります。

次のいくつかのパラグラフでは、ここで説明したそれぞれの問題の追加情報を示します。

NDB をほかのストレージエンジンに複製するときにサポートされていないマルチマスター　NDB からの異なるストレージエンジンへのレプリケーションの場合、2 つのデータベースの関係は、単純なマスタースレーブ関係である必要があります。つまり、循環レプリケーションまたはマスターマスターレプリケーションは MySQL Cluster とほかのストレージエンジン間でサポートされていません。

また、NDB および別のストレージエンジン間で複製する場合、複数のレプリケーションチャネルを構成できません。(ただし、MySQL Cluster データベースは複数のスレーブ MySQL Cluster データベースに同時に複製できます。)マスターが NDB テーブルを使用している場合、複数の MySQL Server がすべての変更のバイナリログを維持することは、引き続き可能です。ただし、スレーブがマスターを変えた場合 (フェイルオーバー)、新しいマスタースレーブ関係はスレーブで明示的に定義される必要があります。

バイナリロギングを実行していないスレーブのストレージエンジンへの NDB の複製　独自のバイナリロギングを処理しないストレージエンジンを使用するスレーブに MySQL Cluster から複製を試みると、レプリケーションプロセスは次のエラーにより停止します: Binary logging not possible ... Statement cannot be written atomically since more than one engine involved and at least one engine is self-logging (エラー 1595)。次の方法のいずれかで、この問題を回避できます。

- スレーブのバイナリロギングをオフにします　これを実現するには、sql_log_bin = 0 を設定します。
- mysql.ndb_apply_status テーブルに使用されるストレージエンジンを変更します　このテーブルが独自のバイナリロギングを処理しないエンジンを使用することになるため、競合も解消できます。これを行うには、スレーブで ALTER TABLE mysql.ndb_apply_status ENGINE=MyISAM などのステートメントを発行します。スレーブで非 NDB ストレージエンジンを使用する場合、この実行は安全です。これは、複数のスレーブ SQL ノードの同期の維持について気にする必要がないためです。
- スレーブでの mysql.ndb_apply_status テーブルへの変更を除外します　これを行うには、スレーブの SQL ノードを --replicate-ignore-table=mysql.ndb_apply_status で起動します。レプリケーションでほかのテーブルを無視する必要がある場合、代わりに適切な --replicate-wild-ignore-table オプションを使用することをお勧めします。

> **重要**
>
> mysql.ndb_apply_status のレプリケーションまたはバイナリロギングを無効にしたり、ある MySQL Cluster から別の MySQL Cluster に複製するときに、このテーブルに使用されるストレージエンジンを変更したりしないでください。詳細は、MySQL Cluster 間のレプリケーションで使用する、レプリケーションおよびバイナリロギングのフィルタリングルールを参照してください。

NDB から非トランザクションストレージエンジンへのレプリケーション　MyISAM などの非トランザクションストレージエンジンに NDB から複製する場合、INSERT ... ON DUPLICATE KEY UPDATE ステートメントを複製するときに、不要な重複キーエラーが発生することがあります。これらを抑止するには、強制的に更新を (更新としてではなく) 書き込みとして ログを取るようにする --ndb-log-update-as-write=0 を使用します。

MySQL Cluster 間のレプリケーションで使用する、レプリケーションおよびバイナリロギングのフィルタリングルール　--replicate-do-*、--replicate-ignore-*、--binlog-do-db、または --binlog-ignore-db のいずれかのオプションを使用して、複製されるデータベースまたはテーブルを除外する場合、mysql.ndb_apply_status のレプリケーションまたはバイナリロギングをブロックしないように注意する必要があります。このことは、MySQL Cluster 間のレプリケーションが適切に動作するために必要です。特に、次の点に留意しておく必要があります。

1. --replicate-do-db=db_name を使用すると (およびほかの --replicate-do-* または --replicate-ignore-* オプションを使用しない)、データベース db_name にあるテーブルだけが複製されます。この場合、--replicate-do-db=mysql、--binlog-do-db=mysql、または --replicate-do-table=mysql.ndb_apply_status も使用して、確実に mysql.ndb_apply_status をスレーブに移入する必要があります。

   --binlog-do-db=db_name を使用すると (およびほかの --binlog-do-db オプションを使用しない)、データベース db_name にあるテーブルへの変更のみがバイナリログに書き込まれます。この場合、--replicate-do-db=mysql、--binlog-do-db=mysql、または --replicate-do-table=mysql.ndb_apply_status も使用して、確実に mysql.ndb_apply_status をスレーブに移入する必要があります。

2. --replicate-ignore-db=mysql を使用すると、mysql データベースにあるテーブルは複製されません。この場合、--replicate-do-table=mysql.ndb_apply_status も使用して、確実に mysql.ndb_apply_status を複製する必要があります。

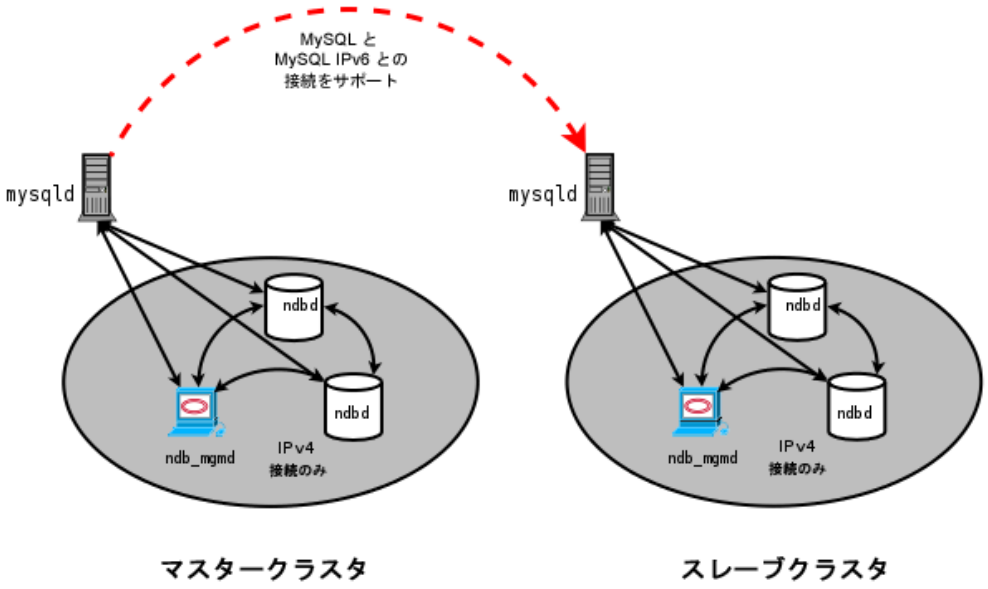
   --binlog-ignore-db=mysql を使用すると、mysql データベースにあるテーブルへの変更はバイナリログに書き込まれません。この場合、--replicate-do-table=mysql.ndb_apply_status も使用して、確実に mysql.ndb_apply_status を複製する必要があります。

各レプリケーションルールには次のことも必要になります。

1. 独自の --replicate-do-* または --replicate-ignore-* オプション。複数のルールはレプリケーションの 1 つのフィルタリングオプションで表現できません。これらのルールについての情報は、セクション17.1.4「レプリケーションおよびバイナリロギングのオプションと変数」を参照してください。
2. 独自の --binlog-do-db または --binlog-ignore-db オプション。複数のルールはバイナリログの 1 つのフィルタリングオプションで表現できません。これらのルールについての情報は、セクション5.2.4「バイナリログ」を参照してください。

NDB 以外のストレージエンジンを使用するスレーブに MySQL Cluster を複製する場合、このセクションの別のところで説明するように、直前で説明した考慮事項は当てはまりません。

MySQL Cluster レプリケーションと IPv6　現在、NDB API と MGM API は IPv6 をサポートしていません。ただし、MySQL Server (MySQL Cluster で SQL ノードとして動作する MySQL Server を含む) は IPv6 を使用してほかの MySQL Server に接続できます。つまり、次の図の点線の矢印で示すように、マスターとスレーブの SQL ノードを接続するために、IPv6 を使用して MySQL Cluster 間を複製できます。



ただし、MySQL Cluster 内で発生するすべての接続 (前の図の実線の矢印で示した接続) は IPv4 を使用する必要があります。言い換えると、すべての MySQL Cluster のデータノード、管理サーバー、および管理クライアントは、IPv4 を使用して互いにアクセスできる必要があります。また、SQL ノードは IPv4 を使用してクラスタと通信する必要があります。

現在、NDB と MGM の API では IPv6 をサポートしていないため、これらの API を使用して書かれたアプリケーションは、IPv4 を使用してすべての接続を作ることも必要です。

属性の昇格と降格　MySQL Cluster レプリケーションでは、属性の昇格と降格がサポートされています。後者の実装では、不可逆型と可逆型の変換は区別され、スレーブでのこれらの変換の使用は、slave_type_conversions グローバルシステム変数を設定することで制御できます。

MySQL Cluster の属性の昇格と降格についての詳細は、行ベースレプリケーション: 属性の昇格と降格を参照してください。

18.6.4 MySQL Cluster レプリケーションスキーマとテーブル

MySQL Cluster でのレプリケーションでは、複製されるクラスタとレプリケーションスレーブ (スレーブは単一のサーバーである場合やクラスタである場合があります) の両方で SQL ノードとしての役割を果たす各 MySQL Server インスタンス上の mysql データベースの数多くの専用のテーブルを使用します。これらのテーブルは mysql_install_db スクリプトによって MySQL のインストールプロセス中に作成され、バイナリログのインデックスデータを保存するテーブルを含みます。ndb_binlog_index テーブルは各 MySQL サーバーに対してローカルであり、クラスタ化には使用されず、MyISAM ストレージエンジンを使用します。すなわち、これは各 mysqld で個別に作成され、マスタークラスタに参加する必要があります。(ただし、バイナリログ自身には、複製されるクラスタ内にあるすべての MySQL サーバーからの更新が含まれます。)このテーブルは次のように定義されます。

```
CREATE TABLE `ndb_binlog_index` (
  `Position` BIGINT(20) UNSIGNED NOT NULL,
  `File` VARCHAR(255) NOT NULL,
  `epoch` BIGINT(20) UNSIGNED NOT NULL,
  `inserts` INT(10) UNSIGNED NOT NULL,
  `updates` INT(10) UNSIGNED NOT NULL,
  `deletes` INT(10) UNSIGNED NOT NULL,
  `schemaops` INT(10) UNSIGNED NOT NULL,
  `orig_server_id` INT(10) UNSIGNED NOT NULL,
  `orig_epoch` BIGINT(20) UNSIGNED NOT NULL,
  `gci` INT(10) UNSIGNED NOT NULL,
  `next_position` bigint(20) unsigned NOT NULL,
  `next_file` varchar(255) NOT NULL,
  PRIMARY KEY (`epoch`,`orig_server_id`,`orig_epoch`)
) ENGINE=MyISAM DEFAULT CHARSET=latin1;
```

このテーブルのサイズは、バイナリログファイル当たりのエポック数およびバイナリログファイル数に依存します。一般的に、バイナリログファイル当たりのエポック数は、エポックごとに作成されるバイナリログの量とバ

イナリログファイルのサイズに依存し、エポックが小さくなるとファイル当たりのエポックが増えます。空のエポックは、--ndb-log-empty-epochs オプションが OFF のときでも、ndb_binlog_index テーブルに挿入される結果となり、ファイル当たりのエントリ数はファイルが使用中である時間の長さに依存する点に注意してください。すなわち、次のとおりです。

```
[number of epochs per file] = [time spent per file] / TimeBetweenEpochs
```

ビジーな MySQL Cluster は定期的にバイナリログに書き込み、おそらく、ビジーでない MySQL Cluster よりも頻繁にバイナリログファイルを交替します。すなわち、--ndb-log-empty-epochs=ON である「ビジーでない」 MySQL Cluster は、大量のアクティビティーを持つ MySQL Cluster に比べて、実際にはファイル当たりの ndb_binlog_index 行数をかなり多く持つことができます。

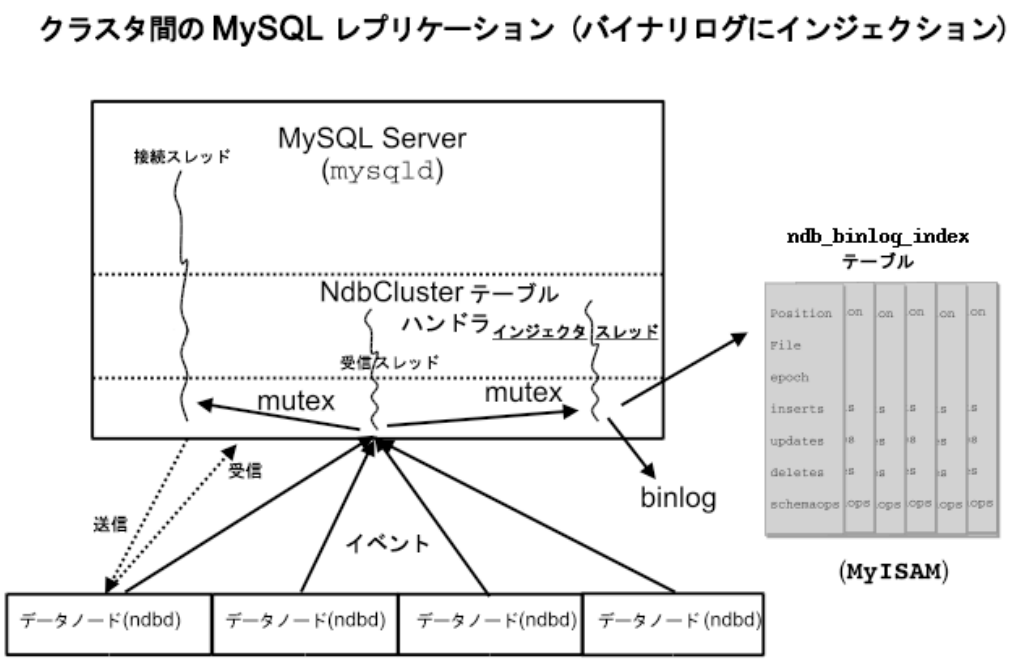
mysqld が --ndb-log-orig オプションで起動されると、orig_server_id と orig_epoch のカラムに、それぞれ、イベントが発生したサーバーの ID と、イベントが発生元のサーバーで行なったエポックが格納されます。これは、複数のマスターを使用する MySQL Cluster のレプリケーションのセットアップに役立ちます。マルチマスターのセットアップでスレーブに適用されるいちばん高いエポックにいちばん近いバイナリログの位置を検出するために使用される SELECT ステートメントは (セクション18.6.10「MySQL Cluster レプリケーション: マルチマスターと循環レプリケーション」を参照してください)、これら 2 つのカラム (インデックス化されていません) を使用します。これにより、特にマスターが --ndb-log-empty-epochs=ON で実行しているときに、クエリーはテーブルスキャンを実行しなければならないため、フェイルオーバーを試みるときにパフォーマンスの問題につながる可能性があります。ここで示すように、これらのカラムにインデックスを追加することで、マルチマスターのフェイルオーバー時間を改善できます。

```
ALTER TABLE mysql.ndb_binlog_index
  ADD INDEX orig_lookup USING BTREE (orig_server_id, orig_epoch);
```

このインデックスを追加しても、1 つのマスターから 1 つのスレーブに複製する場合はメリットがありません。これは、このような場合にバイナリログの位置の取得に使用するクエリーは orig_server_id または orig_epoch を利用しないためです。

next_position および next_file カラムの使用についての詳細は、セクション18.6.8「MySQL Cluster レプリケーションを使用したフェイルオーバーの実装」を参照してください。

次の図は、MySQL Cluster レプリケーションのマスターサーバー、そのバイナリログのインジェクタスレッド、および mysql.ndb_binlog_index テーブルの関係を示します。



ndb_apply_status という名前の追加テーブルは、マスターからスレーブに複製された操作の記録を取るために使用されます。ndb_binlog_index の場合と異なり、このテーブルのデータは (スレーブ) クラスタのどの SQL ノードにも特定されていないため、ここで示すように、ndb_apply_status は NDBCLUSTER ストレージエンジンを使用できます。

```
CREATE TABLE `ndb_apply_status` (
    `server_id`   INT(10) UNSIGNED NOT NULL,
    `epoch`       BIGINT(20) UNSIGNED NOT NULL,
    `log_name`    VARCHAR(255) CHARACTER SET latin1 COLLATE latin1_bin NOT NULL,
    `start_pos`   BIGINT(20) UNSIGNED NOT NULL,
    `end_pos`     BIGINT(20) UNSIGNED NOT NULL,
    PRIMARY KEY (`server_id`) USING HASH
) ENGINE=NDBCLUSTER   DEFAULT CHARSET=latin1;
```

ndb_apply_status テーブルはスレーブでのみ移入されます。このことは、マスターでは、このテーブルに行は含まれないことを意味しています。このため、ここで ndb_apply_status に割り当てられる DataMemory または IndexMemory を考慮する必要がありません。

このテーブルはマスターで発生したデータから移入されるため、複製は許可されます。ただし、レプリケーションのフィルタリングルールまたはバイナリログのフィルタリングルールによって、意図せずにスレーブが ndb_apply_status を更新しなかったり、マスターがバイナリログに書き込まなかったりすると、クラスタ間のレプリケーションが正しく動作しない場合があります。このようなフィルタリングルールに起因する潜在的な問題の詳細は、MySQL Cluster 間のレプリケーションで使用する、レプリケーションおよびバイナリロギングのフィルタリングルールを参照してください。

ndb_binlog_index と ndb_apply_status のテーブルは、ユーザーによって明示的に複製されることはないため、mysql データベースに作成されます。通常、これらのテーブルのどちらかを作成または維持するには、ユーザーの介入は必要ありません。ndb_binlog_index と ndb_apply_status の両方とも NDB バイナリログ (binlog) のインジェクタスレッドによって維持されるためです。これにより、マスターの mysqld プロセスは、NDB ストレージエンジンによって実行された変更への更新が維持されます。NDB binlog インジェクタスレッドは NDB ストレージエンジンから直接イベントを受け取ります。NDB インジェクタは、クラスタ内のすべてのデータイベントを取得する役割を担い、データを変更、挿入、または削除するすべてのイベントが ndb_binlog_index テーブルに記録されたかを確認します。スレーブの I/O スレッドは、マスターのバイナリログからスレーブのリレーログにイベントを転送します。

ただし、レプリケーション用に MySQL Cluster を準備する初期の段階で、これらのテーブルの存在と完全性を確認することをお勧めします。マスターで mysql.ndb_binlog_index テーブルに直接クエリーを発行することで、バイナリログに記録されたイベントデータを参照できます。これは、レプリケーションマスターまたはスレーブの MySQL サーバーのどちらかで SHOW BINLOG EVENTS ステートメントを使用して実現することもできます。(セクション13.7.5.3「SHOW BINLOG EVENTS 構文」を参照してください。)

SHOW ENGINE NDB STATUS の出力から有効な情報を取得することもできます。

ndb_schema テーブルは、NDB テーブルに行われたスキーマの変更を追跡するために使用されます。ここで示すように定義されます。

```
CREATE TABLE ndb_schema (
    `db` VARBINARY(63) NOT NULL,
    `name` VARBINARY(63) NOT NULL,
    `slock` BINARY(32) NOT NULL,
    `query` BLOB NOT NULL,
    `node_id` INT UNSIGNED NOT NULL,
    `epoch` BIGINT UNSIGNED NOT NULL,
    `id` INT UNSIGNED NOT NULL,
    `version` INT UNSIGNED NOT NULL,
    `type` INT UNSIGNED NOT NULL,
    PRIMARY KEY USING HASH (db,name)
) ENGINE=NDB   DEFAULT CHARSET=latin1;
```

このセクションで前に述べた 2 つのテーブルとは異なり、ndb_schema テーブルは MySQL SHOW ステートメントでも、INFORMATION_SCHEMA テーブルでも参照できません。ただし、ここで示すように、ndb_show_tables の出力で参照できます。

```
shell> ndb_show_tables -t 2
id   type              state   logging database    schema   name
4    UserTable         Online  Yes     mysql       def      ndb_apply_status
5    UserTable         Online  Yes     ndbworld    def      City
6    UserTable         Online  Yes     ndbworld    def      Country
3    UserTable         Online  Yes     mysql       def      NDB$BLOB_2_3
7    UserTable         Online  Yes     ndbworld    def      CountryLanguage
2    UserTable         Online  Yes     mysql       def      ndb_schema

NDBT_ProgramExit: 0 - OK
```

ここで示すように、mysql およびほかの MySQL クライアントアプリケーションでこのテーブルの SELECT を発行することも可能です。

```
mysql> SELECT * FROM mysql.ndb_schema WHERE name='City' \G
*************************** 1. row ***************************
    db: ndbworld
  name: City
 slock:
 query: alter table City engine=ndb
node_id: 4
 epoch: 0
    id: 0
version: 0
  type: 7
1 row in set (0.00 sec)
```

これは、アプリケーションのデバッグで有効となる場合があります。

> **注記**
>
> NDB テーブルでスキーマの変更を行う場合、アプリケーションは ALTER TABLE ステートメントを発行した MySQL クライアント接続でこのステートメントが戻るまで待ってから、更新されたテーブル定義の使用を試みます。

ndb_apply_status テーブルまたは ndb_schema テーブルがスレーブに存在しない場合、ndb_restore は存在しないテーブル (または複数のテーブル) を再作成します (Bug #14612)。

MySQL Cluster レプリケーションの競合の解決には、追加の mysql.ndb_replication テーブルが必要です。現在、このテーブルは手動で作成する必要があります。これを行う方法については、セクション18.6.11「MySQL Cluster レプリケーションの競合解決」を参照してください。

18.6.5 レプリケーションのための MySQL Cluster の準備

レプリケーションのための MySQL Cluster の準備は、次のステップで構成されます。

1. すべての MySQL サーバーに対してバージョンの互換性を確認します (セクション18.6.2「MySQL Cluster レプリケーションの一般要件」を参照してください)。

2. マスタークラスタで適切な権限を持つスレーブのアカウントを作成します。

   ```
   mysqlM> GRANT REPLICATION SLAVE
       -> ON *.* TO 'slave_user'@'slave_host'
       -> IDENTIFIED BY 'slave_password';
   ```

   前のステートメントで、slave_user はスレーブアカウントのユーザー名、slave_host はレプリケーションスレーブのホスト名または IP アドレス、slave_password はこのアカウントに割り当てるパスワードです。

   たとえば、名前が 「myslave」 のスレーブのユーザーアカウントを作成するには、名前が 「rep-slave」 のホストからログインし、パスワード 「53cr37」 を使用して、次の GRANT ステートメントを使用します。

   ```
   mysqlM> GRANT REPLICATION SLAVE
       -> ON *.* TO 'myslave'@'rep-slave'
       -> IDENTIFIED BY '53cr37';
   ```

   セキュリティー上、レプリケーションスレーブのアカウントには (ほかの目的に使用しない) 一意のユーザーアカウントを使用することをお勧めします。

3. スレーブがマスターを使用するように構成します。MySQL Monitor を使用すると、CHANGE MASTER TO ステートメントを使用してこれを実現できます。

   ```
   mysqlS> CHANGE MASTER TO
       -> MASTER_HOST='master_host',
       -> MASTER_PORT=master_port,
       -> MASTER_USER='slave_user',
       -> MASTER_PASSWORD='slave_password';
   ```

   前のステートメントでは、master_host はレプリケーションマスターのホスト名または IP アドレス、master_port はスレーブがマスターに接続するために使用されるポート、slave_user はマスターでスレーブ用にセットアップされたユーザー名、slave_password は前のステップでそのユーザーアカウント用に設定されたパスワードです。

   たとえば、ホスト名が 「rep-master」 の MySQL サーバーから、前のステップで作成されたレプリケーションスレーブのアカウントを使用して複製するようにスレーブに伝えるには、次のステートメントを使用します。

```
mysqlS> CHANGE MASTER TO
    -> MASTER_HOST='rep-master',
    -> MASTER_PORT=3306,
    -> MASTER_USER='myslave',
    -> MASTER_PASSWORD='53cr37';
```

このステートメントで使用できるオプションの完全なリストについては、セクション13.4.2.1「CHANGE MASTER TO 構文」を参照してください。

レプリケーションのバックアップ機能を提供するには、レプリケーションプロセスを起動する前に、--ndb-connectstring オプションをスレーブの my.cnf ファイルに追加することも必要です。詳細は、セクション18.6.9「MySQL Cluster レプリケーションを使用した MySQL Cluster バックアップ」を参照してください。

レプリケーションスレーブに my.cnf で設定できる追加オプションは、セクション17.1.4「レプリケーションおよびバイナリロギングのオプションと変数」を参照してください。

4. マスタークラスタがすでに使用中である場合、マスターのバックアップを作成し、それをスレーブにロードして、スレーブがマスターと同期を取る時間を削減できます。スレーブも MySQL Cluster を実行している場合、セクション18.6.9「MySQL Cluster レプリケーションを使用した MySQL Cluster バックアップ」 で説明されたバックアップとリストアの手順を使用して、これを実現できます。

   ```
   ndb-connectstring=management_host[:port]
   ```

   レプリケーションスレーブで MySQL Cluster を使用していない場合、レプリケーションマスターでこのコマンドを使用してバックアップを作成できます。

   ```
   shellM> mysqldump --master-data=1
   ```

   次に、ダンプファイルをスレーブにコピーして、その結果得られるデータダンプをスレーブにインポートします。この後、ここに示すように mysql クライアントを使用してデータをダンプファイルからスレーブのデータベースにインポートできます。ここで、dump_file はマスターで mysqldump を使用して生成されたファイルの名前であり、db_name は複製されるデータベースの名前です。

   ```
   shellS> mysql -u root -p db_name < dump_file
   ```

   mysqldump で使用するオプションの完全なリストは、セクション4.5.4「mysqldump — データベースバックアッププログラム」を参照してください。

   > **注記**
   >
   > この方法でスレーブにデータをコピーする場合、すべてのデータがロードされる前に、スレーブがマスターに接続を試みてレプリケーションを開始しないようにするため、コマンド行で --skip-slave-start オプションを使用してスレーブが起動されたことを確認する必要があります。または、スレーブの my.cnf ファイルに skip-slave-start 含める必要があります。データのロードが完了したら、次の 2 つのセクションで説明する追加ステップに従います。

5. レプリケーションマスターとして動作している各 MySQL サーバーが、一意のサーバー ID で構成され、バイナリロギングが有効化された状態で行フォーマットを使用して構成されていることを確認します。(セクション17.1.2「レプリケーション形式」を参照してください。)これらのオプションは、マスターサーバーの my.cnf ファイルで、またはマスターの mysqld プロセスを起動するときにコマンド行で設定できます。後者のオプションに関する情報については、セクション18.6.6「MySQL Cluster レプリケーションの起動 (レプリケーションチャネルが 1 つ)」を参照してください。

18.6.6 MySQL Cluster レプリケーションの起動 (レプリケーションチャネルが 1 つ)

このセクションでは、1 つのレプリケーションチャネルを使用する MySQL Cluster レプリケーションを起動するための手順の概要について説明します。

1. このコマンドを発行して MySQL レプリケーションのマスターサーバーを起動します。

   ```
   shellM> mysqld --ndbcluster --server-id=id \
         --log-bin &
   ```

   前のステートメントで、id はこのサーバーの一意の ID です (セクション18.6.2「MySQL Cluster レプリケーションの一般要件」を参照してください)。これは、適切なロギング形式を使用するバイナリロギングを有効にして、サーバーの mysqld プロセスを起動します。

> **注記**
>
> --binlog-format=MIXED でマスターを起動することもできます。この場合、クラスタ間で複製するときに行ベースのレプリケーションが自動的に使用されます。STATEMENT ベースのバイナリロギングは MySQL Cluster レプリケーションではサポートされていません (セクション18.6.2「MySQL Cluster レプリケーションの一般要件」を参照してください)。

2. ここで示すように、MySQL レプリケーションのスレーブサーバーを起動します。

```
shellS> mysqld --ndbcluster --server-id=id &
```

ここで示すコマンドで、id はスレーブサーバーの一意の ID です。レプリケーションスレーブでロギングを有効にする必要はありません。

> **注記**
>
> レプリケーションをすぐに開始したくない場合、このコマンドで --skip-slave-start オプションを使用するか、スレーブサーバーの my.cnf ファイルに skip-slave-start を含める必要があります。このオプションを使用すると、以下のステップ 4 で説明するように、適切な START SLAVE ステートメントが発行されるまで、レプリケーションの開始が遅延されます。

3. マスターサーバーのレプリケーションバイナリログとスレーブサーバーとの同期を取る必要があります。これまでにマスターでバイナリロギングが実行していなかった場合、次のステートメントをスレーブで実行します。

```
mysqlS> CHANGE MASTER TO
    -> MASTER_LOG_FILE='',
    -> MASTER_LOG_POS=4;
```

これは、ログの開始ポイントからマスターのバイナリログを読み取るようにスレーブに指示します。ほかの方法で、つまり、バックアップを使用してマスターからデータをロードする場合、MASTER_LOG_FILE および MASTER_LOG_POS に使用する適切な値を取得する方法について、セクション18.6.8「MySQL Cluster レプリケーションを使用したフェイルオーバーの実装」を参照してください。

4. 最後に、レプリケーションスレーブで mysql クライアントからこのコマンドを発行して、レプリケーションの適用を開始するようにスレーブに指示する必要があります。

```
mysqlS> START SLAVE;
```

これにより、レプリケーションデータのマスターからスレーブへの転送も開始します。

次のセクションで説明する手順に類似した方法で、2 つのレプリケーションチャネルを使用することも可能です。この方法と 1 つのレプリケーションチャネルを使用する方法との違いはセクション18.6.7「2 つのレプリケーションチャネルを使用する MySQL Cluster レプリケーション」で説明しています。

バッチ更新を有効にして、クラスタのレプリケーションのパフォーマンスを向上することも可能です。スレーブの mysqld プロセスで slave_allow_batching システム変数を設定することで、これを実現できます。通常、更新は受け取るとすぐに適用されます。しかし、バッチを使用すると、更新は 32K バイトがまとめて適用されるため、特に個々の更新が比較的小さい場合、スループットは向上しますが、CPU 利用率は低下する結果となる場合があります。

> **注記**
>
> スレーブのバッチ処理はエポックごとに機能します。複数のトランザクションに属する更新は同じバッチの一部として送ることができます。
>
> すべての未処理の更新は、その更新の合計が 32K バイト未満であっても、エポックの最後に達したときに適用されます。

バッチ処理は、実行時にオンとオフを切り替えることができます。実行時にこれを有効にするため、次の 2 つのステートメントのどちらかを使用できます。

```
SET GLOBAL slave_allow_batching = 1;
SET GLOBAL slave_allow_batching = ON;
```

特定のバッチによって問題が起きる場合 (作用により正しく複製されるように思えないステートメントなど)、スレーブのバッチ処理は次のステートメントのどちらかを使用して無効化できます。

```
SET GLOBAL slave_allow_batching = 0;
SET GLOBAL slave_allow_batching = OFF;
```

現在スレーブのバッチ処理が適切な SHOW VARIABLES ステートメントによって使用されているかどうかを、次のように確認できます。

```
mysql> SHOW VARIABLES LIKE 'slave%';
+---------------------------+-------+
| Variable_name             | Value |
+---------------------------+-------+
| slave_allow_batching      | ON    |
| slave_compressed_protocol | OFF   |
| slave_load_tmpdir         | /tmp  |
| slave_net_timeout         | 3600  |
| slave_skip_errors         | OFF   |
| slave_transaction_retries | 10    |
+---------------------------+-------+
6 rows in set (0.00 sec)
```

18.6.7 2 つのレプリケーションチャネルを使用する MySQL Cluster レプリケーション

さらに完成された例のシナリオでは、2 つのレプリケーションチャネルを使用することで冗長性を提供し、これによって、1 つのレプリケーションチャネルで発生する可能性のある障害を防ぐことを想定しています。これには合計 4 台のレプリケーションサーバーが必要で、2 台のマスターサーバーをマスタークラスタに、2 台のスレーブサーバーをスレーブクラスタに使用します。以後の説明では、一意の識別子が次のように割り当てられているものとします。

サーバー ID	説明
1	マスター - プライマリレプリケーションチャネル (M)
2	マスター - セカンダリレプリケーションチャネル (M')
3	スレーブ - プライマリレプリケーションチャネル (S)
4	スレーブ - セカンダリレプリケーションチャネル (S')

2 つのチャネルを使用するレプリケーションの設定は、1 つのレプリケーションチャネルの設定と根本的に異なるわけではありません。最初に、プライマリおよびセカンダリのレプリケーションマスターに対する mysqld プロセスを起動し、次にプライマリおよびセカンダリのスレーブに対するプロセスを起動する必要があります。次に、各スレーブで START SLAVE ステートメントを発行すると、レプリケーションプロセスを開始できます。コマンドと、そのコマンドの発行に必要な順序を次に示します。

1. プライマリレプリケーションマスターを起動します。

   ```
   shellM> mysqld --ndbcluster --server-id=1 \
               --log-bin &
   ```

2. セカンダリレプリケーションマスターを起動します。

   ```
   shellM'> mysqld --ndbcluster --server-id=2 \
               --log-bin &
   ```

3. プライマリレプリケーションのスレーブサーバーを起動します。

   ```
   shellS> mysqld --ndbcluster --server-id=3 \
               --skip-slave-start &
   ```

4. セカンダリレプリケーションスレーブを起動します。

   ```
   shellS'> mysqld --ndbcluster --server-id=4 \
                --skip-slave-start &
   ```

5. 最後に、ここで示すように、プライマリスレーブで START SLAVE ステートメントを実行して、プライマリチャネルでレプリケーションを開始します。

   ```
   mysqlS> START SLAVE;
   ```

> **警告**
>
> この時点では、プライマリチャネルだけが起動されます。セカンダリレプリケーションチャネルは、セクション18.6.8「MySQL Cluster レプリケーションを使用したフェイルオーバーの実装」で説明しているように、プライマリレプリケーションチャネルで障害が起きた場合にのみ起動されます。複数のレプリケーションチャネルを同時に実行すると、不要な重複レコードがレプリケーションスレーブで作成される可能性があります。

前に述べたとおり、レプリケーションスレーブでバイナリロギングを有効にする必要はありません。

18.6.8 MySQL Cluster レプリケーションを使用したフェイルオーバーの実装

プライマリクラスタのレプリケーションプロセスが失敗した場合、セカンダリレプリケーションチャネルに切り替えることができます。次に、この実現に必要なステップについて説明します。

1. 最新のグローバルチェックポイント (GCP) の時間を取得します。すなわち、スレーブクラスタの ndb_apply_status テーブルから最新の エポックを指定する必要があり、これは次のクエリーを使用して特定できます。

```
mysqlS'> SELECT @latest:=MAX(epoch)
    ->       FROM mysql.ndb_apply_status;
```

2. ステップ 1 で示したクエリーから得た情報を使用して、マスタークラスタの ndb_binlog_index テーブルから対応するレコードを取得します。

   MySQL Cluster NDB 7.3 では、次のクエリーを使用すると、マスターの ndb_binlog_index テーブルから必要なレコードを取得できます。

```
mysqlM'> SELECT
    ->    @file:=SUBSTRING_INDEX(next_file, '/', -1),
    ->    @pos:=next_position
    -> FROM mysql.ndb_binlog_index
    -> WHERE epoch = @latest
    -> ORDER BY epoch ASC LIMIT 1;
```

   これらは、プライマリレプリケーションチャネルでの障害発生後に、マスターに保存されたレコードです。ここでは、ユーザー変数 @latest を使用して、ステップ 1 で取得した値を表します。もちろん、ある mysqld インスタンスが、ほかのサーバーインスタンスに設定されたユーザー変数には直接アクセスできません。これらの値は、手動で 2 番目のクエリーに、またはアプリケーションコードに「プラグインする」必要があります。

   > **重要**
   >
   > START SLAVE を実行する前に、スレーブ mysqld が --slave-skip-errors=ddl_exist_errorsで起動されていることを確認します。そうしないと、レプリケーションが重複 DDL エラーで停止する可能性があります。

3. これで、セカンダリスレーブサーバーで次のクエリーを実行して、セカンダリチャネルの同期を取ることができるようになります。

```
mysqlS'> CHANGE MASTER TO
    ->    MASTER_LOG_FILE='@file',
    ->    MASTER_LOG_POS=@pos;
```

   再度ユーザー変数 (この場合は @file と @pos) を使用して、ステップ 2 で取得され、ステップ 3 で適用される値を表します。実際には、これらの値を手動で挿入するか、関与する両方のサーバーにアクセスできるアプリケーションコードを使用して挿入する必要があります。

   > **注記**
   >
   > @file は '/var/log/mysql/replication-master-bin.00001' などの文字列値であるため、SQL またはアプリケーションコードで使用するときは、引用符で囲む必要があります。ただし、@pos で表される値は引用符で囲む必要はありません。通常、MySQL は文字列を数字に変換しようとしますが、この場合は例外です。

4. これで、セカンダリスレーブの mysqld で適切なコマンドを発行して、セカンダリチャネルでレプリケーションを開始できるようになりました。

```
mysqlS'> START SLAVE;
```

セカンダリレプリケーションチャネルがアクティブになったら、プライマリの不具合を調べて、修復できます。これを行うために必要な的確なアクションは、プライマリチャネルで失敗した原因によって異なります。

> **警告**
>
> セカンダリレプリケーションチャネルは、プライマリレプリケーションチャネルの停止時や停止した場合にのみ、起動されます。複数のレプリケーションチャネルを同時に実行すると、不要な重複レコードがレプリケーションスレーブで作成される可能性があります。

障害が 1 台のサーバーに限定される場合、M から S' に、または M' から S に、(論理的には) 複製できますが、まだ検証されていません。

18.6.9 MySQL Cluster レプリケーションを使用した MySQL Cluster バックアップ

このセクションでは、バックアップの作成と、バックアップからの MySQL Cluster レプリケーションを使用したリストアについて説明します。レプリケーションサーバーは、前に説明したとおりにすでに構成されたものとします (セクション18.6.5「レプリケーションのための MySQL Cluster の準備」およびその直後のセクションを参照してください)。これがすでに行われている場合、バックアップを作成してそのバックアップからリストアする手順は次のとおりです。

1. バックアップを開始するには、2 つの異なる方法があります。

   - 方法 A　この方法では、レプリケーションプロセスを開始する前に、クラスタのバックアッププロセスがマスターサーバーで有効になっている必要があります。これを行うには、my.cnf file の [mysql_cluster] セクションに次の行を加えます。ここで、management_host はマスタークラスタに対する NDB 管理サーバーの IP アドレスまたはホスト名であり、port は管理サーバーのポート番号です。

     ```
     ndb-connectstring=management_host[:port]
     ```

     > **注記**
     >
     > ポート番号は、デフォルトのポート (1186) が使用されていない場合にのみ、指定する必要があります。MySQL Cluster でのポートおよびポートの割り当てに関する詳細は、セクション18.2.4「MySQL Cluster の初期構成」を参照してください。

     この場合、バックアップは、このステートメントをレプリケーションマスターで実行すると起動できます。

     ```
     shellM> ndb_mgm -e "START BACKUP"
     ```

   - 方法 B　my.cnf ファイルで、管理ホストを検出する場所が指定されていない場合、START BACKUP コマンドの一部としてこの情報を NDB 管理クライアントに渡すことで、バックアッププロセスを起動できます。ここで示すように、これを実行できます。ここで、management_host と port は管理サーバーのホスト名とポート番号です。

     ```
     shellM> ndb_mgm management_host:port -e "START BACKUP"
     ```

     前に述べたようなシナリオの場合 (セクション18.6.5「レプリケーションのための MySQL Cluster の準備」を参照してください)、これは次のように実行されます。

     ```
     shellM> ndb_mgm rep-master:1186 -e "START BACKUP"
     ```

2. オンラインにされているスレーブにクラスタのバックアップファイルをコピーします。マスタークラスタの ndbd プロセスを動作している各システムには、そのシステム上にクラスタのバックアップファイルが配置され、これらのすべてのファイルがスレーブにコピーされ、正常なリストアを確実に行う必要があります。バックアップファイルは、スレーブの管理ホストが存在するコンピュータ上のどのディレクトリにもコピーできますが、MySQL および NDB バイナリがそのディレクトリの読み取り権限を持っている場合にかぎります。このケースでは、このファイルがディレクトリ /var/BACKUPS/BACKUP-1 にコピーされたものとします。

   スレーブクラスタがマスターと同じ ndbd プロセス (データノード) 番号を持つ必要はありませんが、この番号を同じ番号にすることを強くお勧めします。レプリケーションプロセスの不適切な起動を防ぐには、スレーブを --skip-slave-start オプションで起動する必要があります。

3. マスタークラスタに存在し、スレーブに複製するデータベースをスレーブクラスタに作成します。

> **重要**
>
> 複製する各データベースに対応する CREATE DATABASE (または CREATE SCHEMA) ステートメントは、スレーブクラスタの各 SQL ノードで実行します。

4. MySQL Monitor でこのステートメントを使用してスレーブクラスタをリセットします。

```
mysqlS> RESET SLAVE;
```

リストアプロセスを実行する前に、スレーブの ndb_apply_status テーブルにレコードが含まれていないことを確認することが重要です。これを行うには、スレーブでこの SQL ステートメントを実行します。

```
mysqlS> DELETE FROM mysql.ndb_apply_status;
```

5. これで、各バックアップファイルに対して順番に ndb_restore コマンドを使用して、レプリケーションスレーブでクラスタのリストアプロセスを起動できます。このプロセスの最初に、クラスタのメタデータをリストアするための -m オプションを加える必要があります。

```
shellS> ndb_restore -c slave_host:port -n node-id \
      -b backup-id -m -r dir
```

dir は、バックアップファイルがレプリケーションスレーブに置かれたディレクトリへのパスです。残りのバックアップファイルに対応する ndb_restore コマンドに、-m オプションを使用しないでください。

バックアップファイルがディレクトリ /var/BACKUPS/BACKUP-1 にコピーされた場所である 4 つのデータノード (セクション18.6「MySQL Cluster レプリケーション」 の図に表示されているとおりです) を使用してマスタークラスタからリストアする場合、スレーブで実行されるコマンドの正しいシーケンスは、次のようになります。

```
shellS> ndb_restore -c rep-slave:1186 -n 2 -b 1 -m \
      -r ./var/BACKUPS/BACKUP-1
shellS> ndb_restore -c rep-slave:1186 -n 3 -b 1 \
      -r ./var/BACKUPS/BACKUP-1
shellS> ndb_restore -c rep-slave:1186 -n 4 -b 1 \
      -r ./var/BACKUPS/BACKUP-1
shellS> ndb_restore -c rep-slave:1186 -n 5 -b 1 -e \
      -r ./var/BACKUPS/BACKUP-1
```

> **重要**
>
> エポックがスレーブの mysql.ndb_apply_status に書き込まれるには、この例の ndb_restore の最後の起動に -e (または --restore_epoch) オプションが必要です。この情報がないと、スレーブはマスターと適切に同期を取れなくなります。(セクション18.4.20「ndb_restore — MySQL Cluster バックアップのリストア」を参照してください。)

6. ここで、スレーブの ndb_apply_status テーブルから最新のエポックを取得する必要があります (セクション18.6.8「MySQL Cluster レプリケーションを使用したフェイルオーバーの実装」で説明したとおりです)。

```
mysqlS> SELECT @latest:=MAX(epoch)
      FROM mysql.ndb_apply_status;
```

7. 前のステップで取得したエポック値として @latest を使用すると、ここで示したクエリーを使用して、マスターの mysql.ndb_binlog_index テーブルから正しいバイナリログファイル @file の正しい起動位置 @pos を取得できます。

```
mysqlM> SELECT
    ->     @file:=SUBSTRING_INDEX(File, '/', -1),
    ->     @pos:=Position
    -> FROM mysql.ndb_binlog_index
    -> WHERE epoch > @latest
    -> ORDER BY epoch ASC LIMIT 1;
```

現在レプリケーショントラフィックがない場合、マスターで SHOW MASTER STATUS を実行して、File カラムに表示されるすべてのファイルに対していちばん大きな値のサフィクスを持つ名前のファイルの Position カラムの値を使用すると、この情報を取得できます。ただしこの場合、これを指定し、その内容を次のステップで手動で指定するか、スクリプトで出力を解析して指定する必要があります。

8. 前のステップで取得した値を使用すると、スレーブの mysql クライアントで適切な CHANGE MASTER TO ステートメントを発行できるようになります。

```
mysqlS> CHANGE MASTER TO
    ->     MASTER_LOG_FILE='@file',
    ->     MASTER_LOG_POS=@pos;
```

9. スレーブはどのバイナリログファイルのどの点から読み取りを開始するかがマスターから「わかる」ようになるため、スレーブはこの標準 MySQL ステートメントで複製を開始できます。

```
mysqlS> START SLAVE;
```

セカンダリレプリケーションチャネルでバックアップとリストアを実行するために必要なことは、必要に応じて、プライマリのマスターとスレーブのレプリケーションサーバーのホスト名と ID に代わって、セカンダリのマスターとスレーブのホスト名と ID を使用し、前述のステートメントを実行するステップを繰り返すだけです。

クラスタのバックアップの実行とバックアップからのクラスタのリストアに関する詳細は、セクション 18.5.3「MySQL Cluster のオンラインバックアップ」を参照してください。

18.6.9.1 MySQL Cluster レプリケーション: レプリケーションスレーブとマスターのバイナリログとの同期の自動化

前のセクションで説明した多くのプロセスを自動化できます (セクション18.6.9「MySQL Cluster レプリケーションを使用した MySQL Cluster バックアップ」参照してください)。次の Perl スクリプト reset-slave.pl は、これを行う方法の実例として役に立ちます。

```
#!/user/bin/perl -w

#  file: reset-slave.pl

#  Copyright ©2005 MySQL AB

#  This program is free software; you can redistribute it and/or modify
#  it under the terms of the GNU General Public License as published by
#  the Free Software Foundation; either version 2 of the License, or
#  (at your option) any later version.

#  This program is distributed in the hope that it will be useful,
#  but WITHOUT ANY WARRANTY; without even the implied warranty of
#  MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
#  GNU General Public License for more details.

#  You should have received a copy of the GNU General Public License
#  along with this program; if not, write to:
#  Free Software Foundation, Inc.
#  59 Temple Place, Suite 330
#  Boston, MA 02111-1307 USA
#
#  Version 1.1


######################## Includes #################################

use DBI;

######################## Globals ##################################

my  $m_host='';
my  $m_port='';
my  $m_user='';
my  $m_pass='';
my  $s_host='';
my  $s_port='';
my  $s_user='';
my  $s_pass='';
my  $dbhM='';
my  $dbhS='';

######################## Sub Prototypes ###########################

sub CollectCommandPromptInfo;
sub ConnectToDatabases;
sub DisconnectFromDatabases;
sub GetSlaveEpoch;
sub GetMasterInfo;
sub UpdateSlave;
```

```
######################### Program Main ############################

CollectCommandPromptInfo;
ConnectToDatabases;
GetSlaveEpoch;
GetMasterInfo;
UpdateSlave;
DisconnectFromDatabases;

################## Collect Command Prompt Info ##################

sub CollectCommandPromptInfo
{
  ### Check that user has supplied correct number of command line args
  die "Usage:\n
      reset-slave >master MySQL host< >master MySQL port< \n
                  >master user< >master pass< >slave MySQL host< \n
                  >slave MySQL port< >slave user< >slave pass< \n
      All 8 arguments must be passed. Use BLANK for NULL passwords\n"
      unless @ARGV == 8;

  $m_host  =  $ARGV[0];
  $m_port  =  $ARGV[1];
  $m_user  =  $ARGV[2];
  $m_pass  =  $ARGV[3];
  $s_host  =  $ARGV[4];
  $s_port  =  $ARGV[5];
  $s_user  =  $ARGV[6];
  $s_pass  =  $ARGV[7];

  if ($m_pass eq "BLANK") { $m_pass = '';}
  if ($s_pass eq "BLANK") { $s_pass = '';}
}

###############  Make connections to both databases #############

sub ConnectToDatabases
{
  ### Connect to both master and slave cluster databases

  ### Connect to master
  $dbhM
    = DBI->connect(
    "dbi:mysql:database=mysql;host=$m_host;port=$m_port",
    "$m_user", "$m_pass")
      or die "Can't connect to Master Cluster MySQL process!
              Error: $DBI::errstr\n";

  ### Connect to slave
  $dbhS
    = DBI->connect(
         "dbi:mysql:database=mysql;host=$s_host",
         "$s_user", "$s_pass")
    or die "Can't connect to Slave Cluster MySQL process!
            Error: $DBI::errstr\n";
}

################  Disconnect from both databases ################

sub DisconnectFromDatabases
{
  ### Disconnect from master

  $dbhM->disconnect
  or warn " Disconnection failed: $DBI::errstr\n";

  ### Disconnect from slave

  $dbhS->disconnect
  or warn " Disconnection failed: $DBI::errstr\n";
}

######################  Find the last good GCI ##################

sub GetSlaveEpoch
{
  $sth = $dbhS->prepare("SELECT MAX(epoch)
                         FROM mysql.ndb_apply_status;")
      or die "Error while preparing to select epoch from slave: ",
             $dbhS->errstr;
```

```
  $sth->execute
     or die "Selecting epoch from slave error: ", $sth->errstr;

  $sth->bind_col (1, \$epoch);
  $sth->fetch;
  print "\tSlave Epoch =  $epoch\n";
  $sth->finish;
}

#######  Find the position of the last GCI in the binary log ########

sub GetMasterInfo
{
  $sth = $dbhM->prepare("SELECT
                          SUBSTRING_INDEX(File, '/', -1), Position
                        FROM mysql.ndb_binlog_index
                        WHERE epoch > $epoch
                        ORDER BY epoch ASC LIMIT 1;")
     or die "Prepare to select from master error: ", $dbhM->errstr;

  $sth->execute
     or die "Selecting from master error: ", $sth->errstr;

  $sth->bind_col (1, \$binlog);
  $sth->bind_col (2, \$binpos);
  $sth->fetch;
  print "\tMaster binary log =  $binlog\n";
  print "\tMaster binary log position =  $binpos\n";
  $sth->finish;
}

##########  Set the slave to process from that location #########

sub UpdateSlave
{
  $sth = $dbhS->prepare("CHANGE MASTER TO
                        MASTER_LOG_FILE='$binlog',
                        MASTER_LOG_POS=$binpos;")
     or die "Prepare to CHANGE MASTER error: ", $dbhS->errstr;

  $sth->execute
      or die "CHANGE MASTER on slave error: ", $sth->errstr;
  $sth->finish;
  print "\tSlave has been updated. You may now start the slave.\n";
}

# end reset-slave.pl
```

18.6.9.2 MySQL Cluster レプリケーションを使用したポイントインタイムリカバリ

ポイントインタイムリカバリ (つまり、特定の時点よりあとに行われたデータ変更のリカバリ) は、サーバーをバックアップが行われた時点の状態に戻す完全バックアップのリストア後に実行されます。MySQL Cluster と MySQL Cluster レプリケーションを使用した MySQL Cluster テーブルのポイントインタイムリカバリの実行は、ネイティブの NDB データバックアップ (ndb_mgm クライアントで CREATE BACKUP を発行して取得されます) を使用し、ndb_binlog_index テーブル (mysqldump を使用して作られたダンプ) をリストアすることで実行できます。

MySQL Cluster のポイントインタイムリカバリを実行するには、ここで示すステップに従う必要があります。

1. ndb_mgm クライアントで START BACKUP コマンドを使用して、クラスタ内のすべての NDB データベースをバックアップします (セクション18.5.3「MySQL Cluster のオンラインバックアップ」を参照してください)。

2. その後の時点でクラスタをリストアする前に、mysql.ndb_binlog_index テーブルのバックアップを作成します。このタスクに mysqldump を使用することが、おそらくもっとも簡単です。このときに、バイナリログファイルもバックアップします。

   このバックアップは、必要に応じて定期的 (等間隔で 1 時間ごとなど) に更新してください。

3. (重大な障害または エラーが発生します。)

4. 最新の既知の良好なバックアップを探します。

5. データノードのファイルシステムをクリアします (ndbd --initial または ndbmtd --initial を使用します)。

> **注記**
>
> MySQL Cluster ディスクデータテーブルスペースおよびログファイルは --initial では削除されません。これらを手動で削除する必要があります。

6. mysql.ndb_binlog_index テーブルで DROP TABLE または TRUNCATE TABLE を使用します。
7. ndb_restore を実行して、すべてのデータをリストアします。ndb_apply_status テーブルが正しく移入されるように、ndb_restore を実行するときに --restore_epoch を加える必要があります。(詳細については、セクション18.4.20「ndb_restore — MySQL Cluster バックアップのリストア」を参照してください。)
8. 必要に応じて、mysqldump の出力から ndb_binlog_index テーブルをリストアしたり、バックアップからバイナリログファイルをリストアしたりします。
9. 最後に適用されたエポック (すなわち、ndb_apply_status テーブルの epoch カラムの最大値) をユーザー変数 @LATEST_EPOCH として検索します (重要)。

```
SELECT @LATEST_EPOCH:=MAX(epoch)
  FROM mysql.ndb_apply_status;
```

10. ndb_binlog_index テーブルの @LATEST_EPOCH に対応する、最新のバイナリログファイル (@FIRST_FILE) と位置 (Position カラム値) をこのファイル内で検索します。

```
SELECT Position, @FIRST_FILE:=File
  FROM mysql.ndb_binlog_index
  WHERE epoch > @LATEST_EPOCH ORDER BY epoch ASC LIMIT 1;
```

11. mysqlbinlog を使用して、障害が発生した時点まで、特定のファイルと位置からバイナリログイベントを再現します。(セクション4.6.8「mysqlbinlog — バイナリログファイルを処理するためのユーティリティー」を参照してください。)

バイナリログ、レプリケーション、および増分リカバリに関する詳細は、セクション7.5「バイナリログを使用したポイントインタイム (増分) リカバリ」も参照してください。

18.6.10 MySQL Cluster レプリケーション: マルチマスターと循環レプリケーション

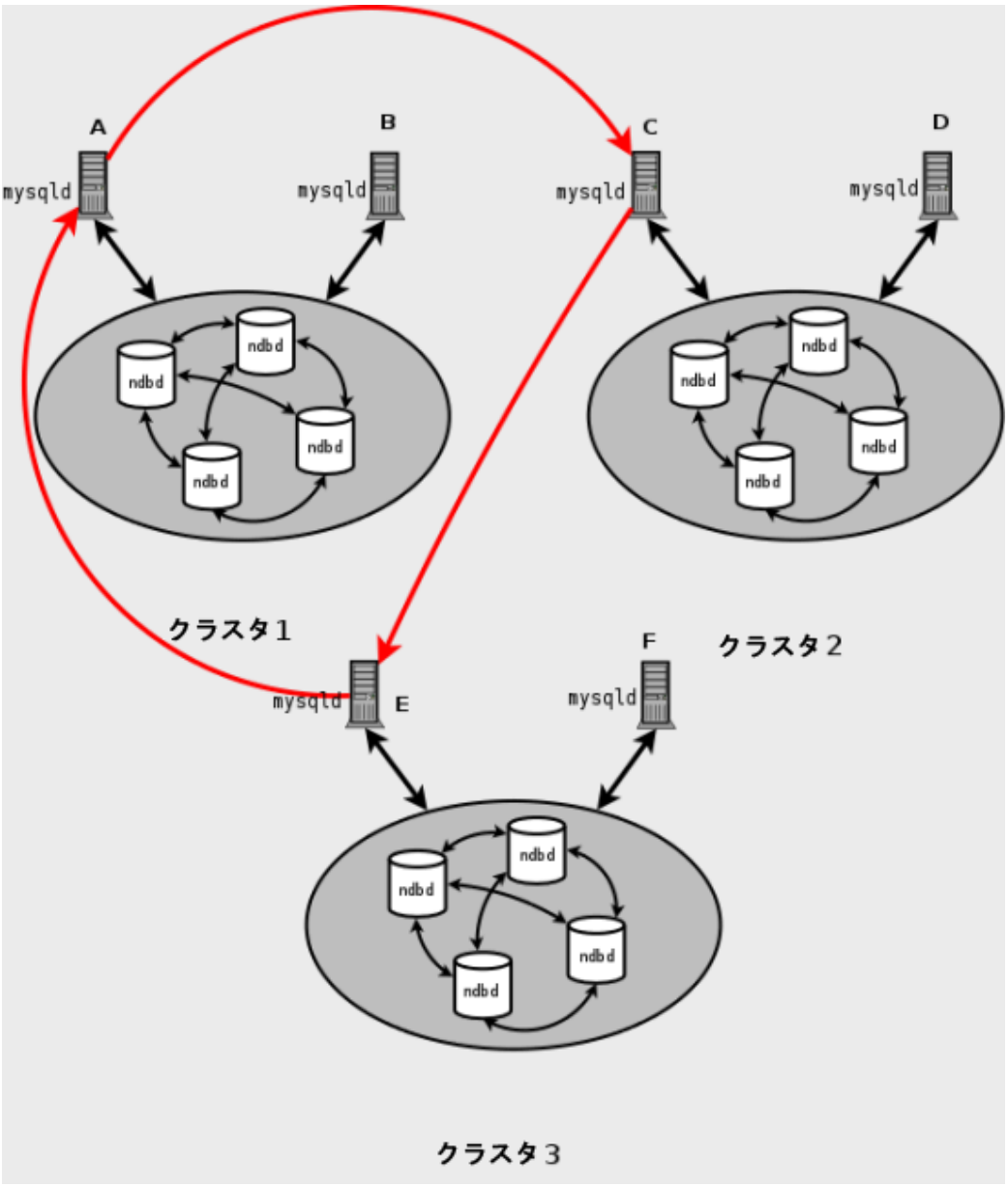
マルチマスターレプリケーションで MySQL Cluster を使用することは可能です (多くの MySQL Cluster 間の循環レプリケーションを含みます)。

循環レプリケーションの例　次のいくつかのパラグラフでは、次のようなレプリケーションセットアップの例について検討します。このセットアップは、番号が 1、2、3 の 3 つの MySQL Cluster で構成され、クラスタ 1 はクラスタ 2 のレプリケーションマスターとして動作し、クラスタ 2 はクラスタ 3 のマスターとして動作し、クラスタ 3 はクラスタ 1 のマスターとして動作します。各クラスタは 2 つの SQL ノードを持ち、SQL ノード A と B はクラスタ 1 に属し、SQL ノード C と D はクラスタ 2 に属し、SQL ノード E と F はクラスタ 3 に属しています。

これらのクラスタを使用する循環レプリケーションは、次の条件を満たすかぎり、サポートされます。

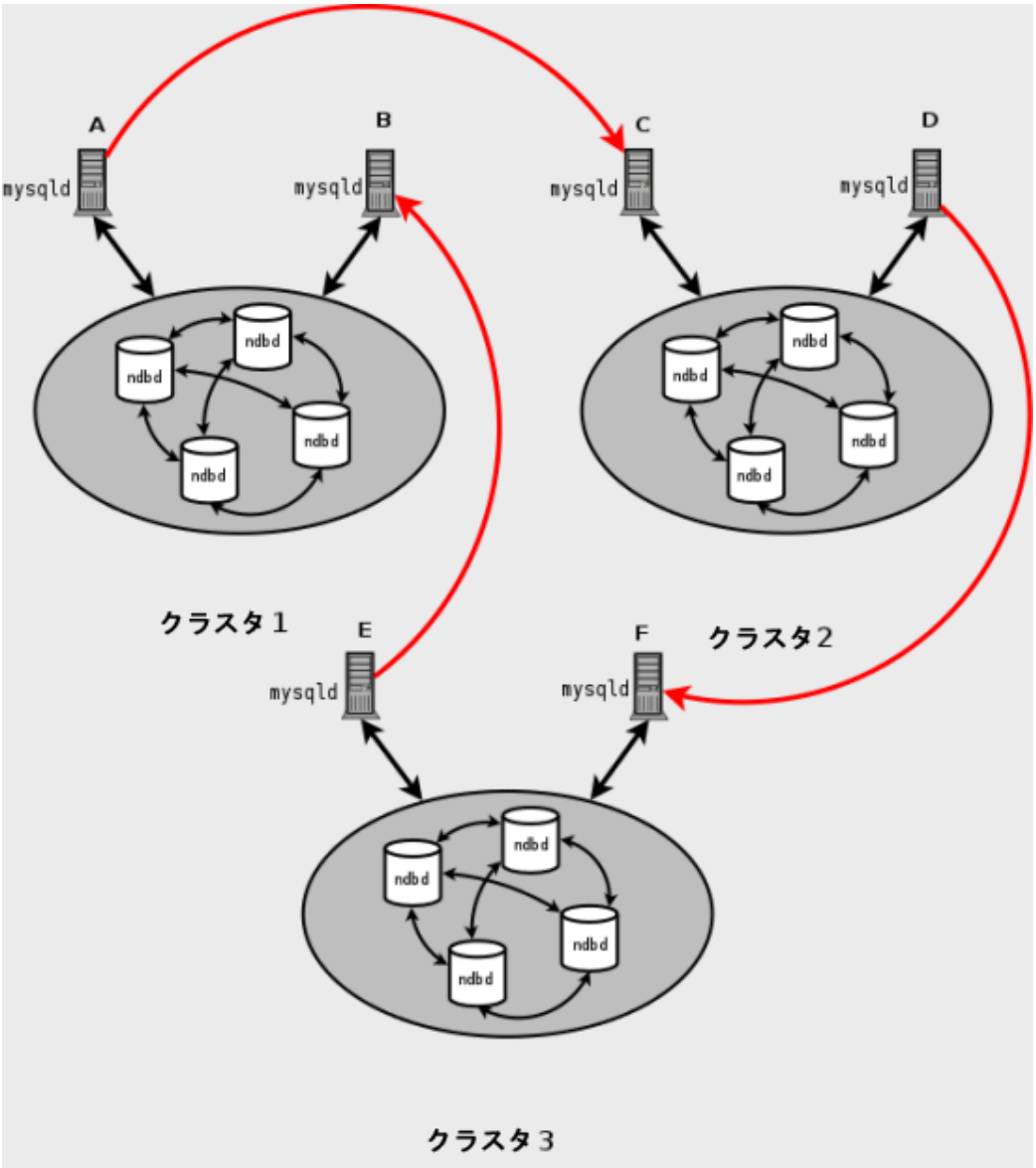
- すべてのマスターとスレーブの SQL ノードは同じ
- レプリケーションのマスターおよびスレーブとして動作するすべての SQL ノードが --log-slave-updates オプションを使用して起動される

このタイプの循環レプリケーションのセットアップは、次の図に示すとおりです。



このシナリオでは、クラスタ 1 の SQL ノード A はクラスタ 2 の SQL ノード C に複製し、SQL ノード C はクラスタ 3 の SQL ノード E に複製し、SQL ノード E は SQL ノード A に複製します。言い換えると、レプリケーションライン (図中の赤の矢印) は、レプリケーションのマスターおよびスレーブとして使用されるすべての SQL ノードを直接接続します。

ここで示すように、すべてのマスター SQL ノードが必ずしもスレーブというわけではない場合に、循環レプリケーションをセットアップすることも可能です。

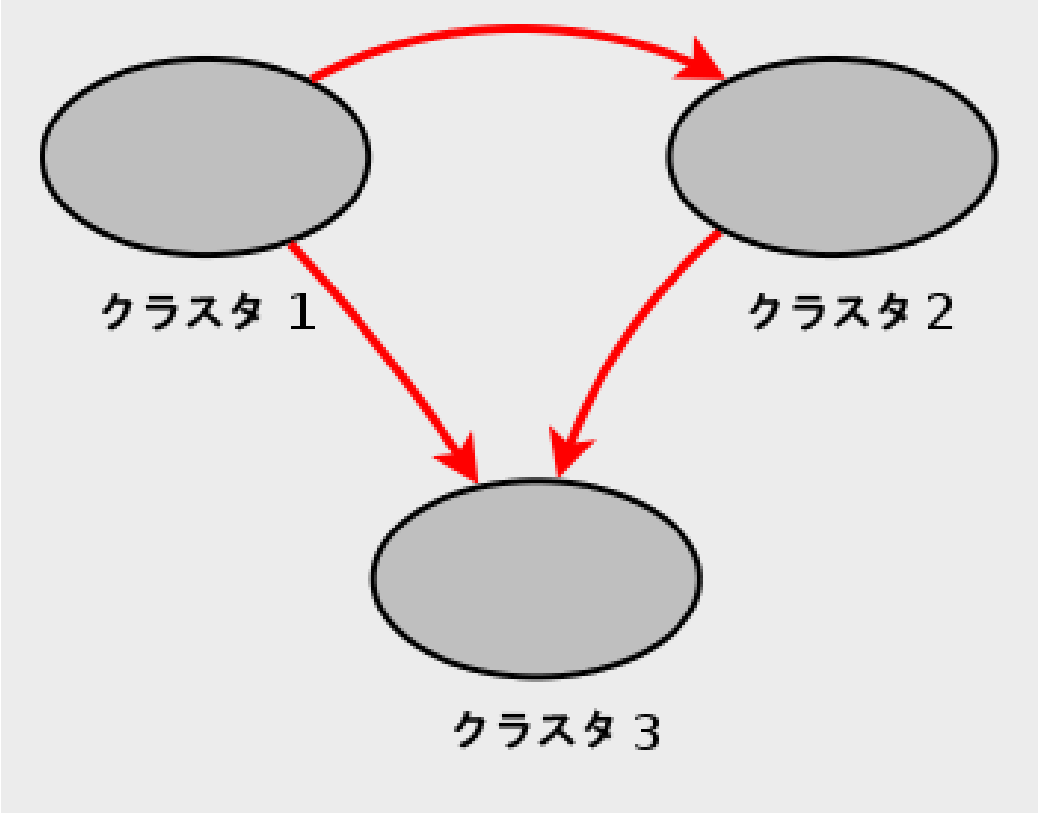


この場合、各クラスタの異なる SQL ノードはレプリケーションのマスターおよびスレーブとして使用されます。ただし、SQL ノードを --log-slave-updates を使用して起動する必要はありません。レプリケーションのライン (これも図中の赤の矢印) が連続的でない MySQL Cluster のこのタイプの循環レプリケーションスキームは、可能性はありますが、まだ完全にはテストされていないため、実験的なものだと考えるようにしてください。

NDB ネイティブのバックアップとリストアを使用したスレーブ MySQL Cluster の初期化　循環レプリケーションをセットアップする場合、バックアップを作成する MySQL Cluster で管理クライアントの BACKUP コマンドを使用してから、ndb_restore を使用して別の MySQL Cluster でこのバックアップを適用すると、スレーブクラスタを初期化できます。ただし、これによってバイナリログが、レプリケーションスレーブとして動作する 2 番目の MySQL Cluster の SQL ノードに、自動的に作成されることはありません。バイナリログが作成されるようにするには、対象の SQL ノードで SHOW TABLES ステートメントを発行する必要があります。これは、START SLAVE を実行する前に行なってください。

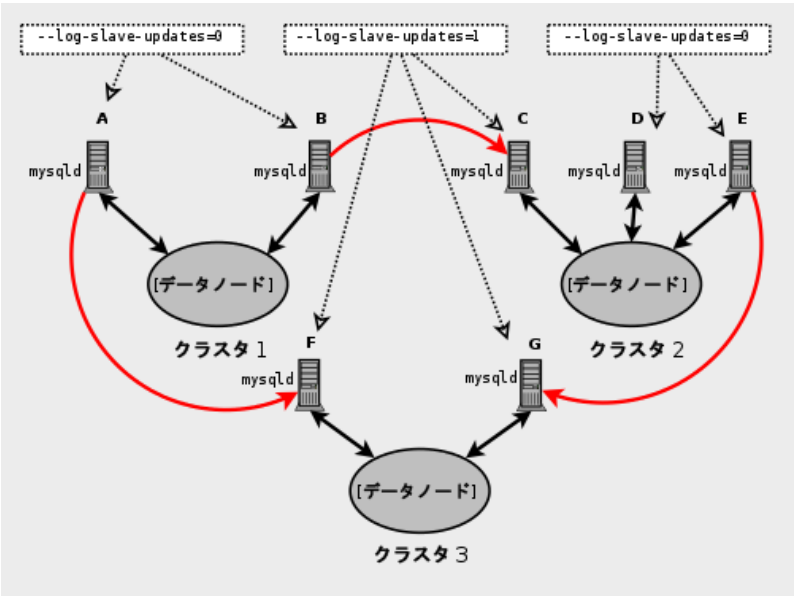
これは、今後のリリースで対応する予定の既知の問題です。

マルチマスターのフェイルオーバーの例　このセクションでは、3 つの MySQL Cluster のサーバー ID が 1、2、および 3 であるマルチマスターの MySQL Cluster レプリケーションセットアップでのフェイルオーバーについて説明します。このシナリオでは、クラスタ 1 はクラスタ 2 および 3 に複製し、クラスタ 2 もクラスタ 3 に複製します。この関係はここで示すとおりです。



つまり、データはクラスタ 1 からクラスタ 3 へ異なる 2 つのルートを介して (直接、およびクラスタ 2 経由で) 複製されます。

マルチマスターレプリケーションに参加するすべての MySQL サーバーが必ずしもマスターとスレーブの両方の役割を果たさなければならないわけではなく、特定の MySQL Cluster が異なるレプリケーションチャネルの異なる SQL ノードを使用する場合があります。このようなケースを次に示します。



レプリケーションスレーブとして動作している MySQL サーバーは、--log-slave-updates オプションを使用して実行する必要があります。また、どの mysqld プロセスでこのオプションが必要か、前の図に示されています。

> **注記**
>
> --log-slave-updates オプションを使用しても、レプリケーションスレーブとして動作していないサーバーには作用しません。

複製しているクラスタの 1 つが停止した場合、フェイルオーバーの必要性が生じます。この例では、クラスタ 1 のサービスが失われ、そのためにクラスタ 3 がクラスタ 1 の更新の 2 つのソースを失うケースを検討します。MySQL Cluster 間のレプリケーションが非同期であるため、クラスタ 1 から直接生じるクラスタ 3 の更新が、クラスタ 2 を経由して受け取った更新よりも新しいという保証はありません。クラスタ 1 からの更新に関して、クラスタ 3 が確実にクラスタ 2 に追いつくことで、これに対処できます。すなわち、MySQL サーバーに関して、未処理の更新を MySQL サーバー C からサーバー F に複製する必要があります。

サーバー C で、次のクエリーを実行します。

```
mysqlC> SELECT @latest:=MAX(epoch)
    ->    FROM mysql.ndb_apply_status
    ->    WHERE server_id=1;

mysqlC> SELECT
    ->    @file:=SUBSTRING_INDEX(File, '/', -1),
    ->    @pos:=Position
    ->    FROM mysql.ndb_binlog_index
    ->    WHERE orig_epoch >= @latest
    ->    AND orig_server_id = 1
    ->    ORDER BY epoch ASC LIMIT 1;
```

> **注記**
>
> 適切なインデックスを ndb_binlog_index テーブルに追加することで、このクエリーのパフォーマンスを向上できるため、フェイルオーバー時間が大幅に短縮される可能性があります。詳細については、セクション18.6.4「MySQL Cluster レプリケーションスキーマとテーブル」を参照してください。

@file および @pos の値を手動でサーバー C からサーバー F にコピーをします (またはアプリケーションで同様に実行させます)。次にサーバー F で、次の CHANGE MASTER TO ステートメントを実行します。

```
mysqlF> CHANGE MASTER TO
    ->    MASTER_HOST = 'serverC'
    ->    MASTER_LOG_FILE='@file',
    ->    MASTER_LOG_POS=@pos;
```

この実行後、MySQL サーバー F で START SLAVE ステートメントを発行でき、サーバー B から生じた、不足している更新がサーバー F に複製されます。

CHANGE MASTER TO ステートメントは IGNORE_SERVER_IDS オプションもサポートします。このオプションは、カンマ区切りのサーバーの ID を使用し、対応するサーバーから発生したイベントが無視されます。詳細については、セクション13.4.2.1「CHANGE MASTER TO 構文」およびセクション13.7.5.35「SHOW SLAVE STATUS 構文」を参照してください。

18.6.11 MySQL Cluster レプリケーションの競合解決

複数のマスターが関与するレプリケーションセットアップを使用する場合 (循環レプリケーションを含みます)、異なるマスターが、異なるデータを持つスレーブ上の同じ行を更新しようとする可能性があります。MySQL Cluster レプリケーションでの競合解決は、特定のマスター上の更新をスレーブで適用すべきかどうかを判断するために使用されるユーザー定義の解決カラムを許可することで、このような競合を解決できます。

MySQL Cluster でサポートされる競合解決のいくつかのタイプ (NDB$OLD()、NDB$MAX()、NDB$MAX_DELETE_WIN()) は、このユーザー定義カラムを 「timestamp」 カラムとして実装しています (ただし、このセクションの後半で説明するように、このタイプは TIMESTAMP になり得ません)。このタイプの競合解決は、常に、トランザクションごとではなく、行ごとに適用されます。エポックベースの競合解決関数 NDB$EPOCH() および NDB$EPOCH_TRANS() は、エポックが複製された順番を比較します (このため、これらの関数はトランザクション型です)。このセクションの後半で説明するように、競合が発生したときにスレーブで解決カラム値を比較するために、さまざまな方法を使用できます。使用する方法はテーブルごとに設定できます。

更新を適用するかどうかを判断するときに解決関数で適切な選択を行えるように、解決カラムに適切な値で正しく移入されることを確認することは、アプリケーションの責任になります。

要件　競合解決の準備は、マスターとスレーブの両方で行う必要があります。これらのタスクは次のリストで説明します。

- バイナリログを書き込むマスターで、どのカラム (すべてのカラム、または更新されたカラムだけ) を送るか指定する必要があります。MySQL Server でこれを行うには、全体的には mysqld 起動オプション --ndb-log-updated-only (このセクションの後半で説明します) を適用し、テーブル単位には mysql.ndb_replication テーブルのエントリによって実行します (ndb_replication システムテーブルを参照してください)。

> **注記**
>
> 非常に大きなカラム (TEXT や BLOB カラムなど) を持つテーブルを複製する場合、--ndb-log-updated-only はマスターとスレーブのバイナリログのサイズを減らしたり、max_allowed_packet を超えたことによって起こる可能性のあるレプリケーション障害を回避したりする場合にも役立つことがあります。
>
> この問題に関する詳細は、セクション17.4.1.20「レプリケーションと max_allowed_packet」を参照してください。

- スレーブでは、どのタイプの競合解決 (「最後のタイムスタンプが優先」、「同じタイムスタンプが優先」、「プライマリが優先」、「プライマリが優先し、トランザクションを完結する」、または何もしない) を適用するかを指定する必要があります。これは、mysql.ndb_replication システムテーブルを使用して、テーブルごとに行われます (ndb_replication システムテーブルを参照してください)。
- MySQL Cluster NDB 7.4.1 以降は読み取り競合検出をサポートしています。すなわち、あるクラスタの特定の行の読み取りと、別のクラスタの同じ行の更新または削除との間の競合を検出します。これには、スレーブで ndb_log_exclusive_reads を 1 に設定することで取得される排他的読み取りロックが必要です。競合している読み取りによって読み取られたすべての行のログが、例外テーブルに記録されます。詳細は、読み取り競合の検出と解決を参照してください。

タイムスタンプベースの競合解決に関数 NDB$OLD()、NDB$MAX()、および NDB$MAX_DELETE_WIN() を使用する場合、更新を「タイムスタンプ」カラムとして指定するために使用されるカラムを参照するのが一般的です。ただし、このカラムのデータ型は TIMESTAMP にしないでください。このデータ型は INT (INTEGER) または BIGINT にしてください。また、「timestamp」 カラムは UNSIGNED および NOT NULL にしてください。

このセクションの後半で説明する NDB$EPOCH() および NDB$EPOCH_TRANS() 関数は、プライマリおよびセカンダリ MySQL Cluster で適用されるレプリケーションエポックの相対順序を比較することで機能し、タイプスタンプを利用しません。

マスターカラムの制御　「前」のイメージおよび「後」のイメージ (すなわち、更新が適用される前と後のテーブルの状態) に関して、更新の操作を確認できます。一般的に主キーでテーブルを更新する場合、「前」のイメージはそれほど問題はありません。ただし、更新ごとにレプリケーションスレーブで更新された値を使用するかどうかを指定する必要がある場合は、両方のイメージがマスターのバイナリログに書き込まれることを確認する必要があります。このセクションの後半で説明するように、これは mysqld の --ndb-log-update-as-write オプションで実行されます。

> **重要**
>
> 行全体のロギングを行うか、更新されたカラムだけのロギングを行うかは、MySQL サーバーが起動されたときに決まり、オンラインでは変更できません。異なるロギングオプションを使用して、mysqld を再起動するか、新しい mysqld インスタンスを起動する必要があります。

すべてまたは一部の行のロギング (--ndb-log-updated-only オプション)

コマンド行形式	--ndb-log-updated-only
システム変数	ndb_log_updated_only
スコープ	グローバル
動的	はい
型	ブール
デフォルト	ON

競合解決のために、行のログを取る基本的な方法は 2 つあり、その方法は mysqld の --ndb-log-updated-only オプションを設定すると指定されます。

- 行全体のログを取得します
- 更新されたカラムデータ (すなわち、値が設定されたカラムデータ) だけのログを取ります。この値が実際に変更されたかどうかには関係しません。これはデフォルトの動作です。

通常、更新されたカラムのみのログを取るだけで十分 (しかも効率的) です。ただし、すべての行のログを取る必要がある場合、--ndb-log-updated-only を 0 または OFF に設定すると、これを実行できます。

--ndb-log-update-as-write オプション: 変更されたデータを更新としてログを取得

コマンド行形式	--ndb-log-update-as-write
システム変数	ndb_log_update_as_write
スコープ	グローバル
動的	はい
型	ブール
デフォルト	ON

MySQL Server の --ndb-log-update-as-write オプションの設定では、ロギングが「前」のイメージがある状態で実行されるか、ない状態で実行されるかを指定します。競合解決は MySQL Server の更新ハンドラで行われるため、更新は更新であって書き込みではないようにマスターでのロギングを制御する必要があります。すなわち、更新は、新しい行の書き込みではなく、(更新が既存の行を置き換える場合でも) 既存の行の変更として処理されます。このオプションはデフォルトではオンです。つまり、更新は書き込みとして処理されます。(つまりデフォルトでは、更新は update_row イベントとしてではなく、write_row イベントとしてバイナリログに書き込まれます。)

オプションをオフにするには、マスターの mysqld を --ndb-log-update-as-write=0 または --ndb-log-update-as-write=OFF で起動します。NDB テーブルから異なるストレージエンジンを使用するテーブルに複製する場合は、これを行う必要があります。詳細は、NDB から別のストレージエンジンへのレプリケーションおよびNDB から非トランザクションストレージエンジンへのレプリケーションを参照してください。

競合解決の制御　通常、競合解決は競合が発生する可能性のあるサーバーで有効になっています。ロギング方法の選択と同様に、mysql.ndb_replication テーブル内のエントリによって有効化されます。

ndb_replication システムテーブル　競合解決を有効にするには、競合解決のタイプ、および使用する方法によってマスター、スレーブ、またはその両方の mysql システムデータベースに ndb_replication テーブルを作成する必要があります。このテーブルは、ロギングと競合解決関数をテーブル単位に制御するために使用され、レプリケーションに関与する 1 つの行単位テーブルを持ちます。ndb_replication が作成され、競合を解決すべきサーバー上の制御情報が格納されます。スレーブでローカルにデータを変更することもできる単純なマスタースレーブのセットアップでは、一般的にこれはスレーブになります。より複雑なマスターマスター (2 方向) レプリケーションスキーマでは、これは関与するすべてのマスターになります。mysql.ndb_replication の各行は複製されるテーブルに対応し、対象テーブルに対するログの取得方法と競合の解決方法 (すなわち、もし競合が発生した場合に、どの競合解決関数を使用するか) を指定します。mysql.ndb_replication テーブルの定義は次のとおりです。

```
CREATE TABLE mysql.ndb_replication  (
    db VARBINARY(63),
    table_name VARBINARY(63),
    server_id INT UNSIGNED,
    binlog_type INT UNSIGNED,
    conflict_fn VARBINARY(128),
    PRIMARY KEY USING HASH (db, table_name, server_id)
)  ENGINE=NDB
PARTITION BY KEY(db,table_name);
```

このテーブルのカラムは、次のいくつかのパラグラフで説明します。

db　複製されるテーブルを含むデータベースの名前です。ワイルドカード _ と % のどちらか、またはその両方を、データベース名の一部として使用できます。マッチングは、LIKE 演算子に対して実装されたマッチングに似ています。

table_name　複製されるテーブルの名前です。テーブル名に、ワイルドカード _ と % のどちらか、またはその両方を含めることができます。マッチングは、LIKE 演算子に対して実装されたマッチングに似ています。

server_id　テーブルが存在する MySQL インスタンス (SQL ノード) の一意のサーバー ID です。

binlog_type　使用されるバイナリロギングのタイプです。これは、次の表に示すように指定されます。

値	内部値	説明
0	NBT_DEFAULT	サーバーのデフォルトを使用します
1	NBT_NO_LOGGING	バイナリログにこのテーブルの記録を行いません
2	NBT_UPDATED_ONLY	更新された属性のみが記録されます
3	NBT_FULL	更新されない場合でも、行全体を記録します (MySQL サーバーのデフォルトの動作)

値	内部値	説明
4	NBT_USE_UPDATE	(NBT_UPDATED_ONLY_USE_UPDATE および NBT_FULL_USE_UPDATE の値の生成のみを行い、単独では使用しません)
5	[Not used]	---
6	NBT_UPDATED_ONLY_USE_UPDATE (NBT_UPDATED_ONLY | NBT_USE_UPDATE に相当)	値が変更されていない場合でも、更新された属性を使用します
7	NBT_FULL_USE_UPDATE (NBT_FULL | NBT_USE_UPDATE に相当)	値が変更されていない場合でも、行全体を使用します

conflict_fn　適用される競合解決関数です。この関数は、次のリストに示されたいずれかの関数として指定する必要があります。

- NDB$OLD(column_name)
- NDB$MAX(column_name)
- NDB$MAX_DELETE_WIN(column_name)
- NDB$EPOCH() および NDB$EPOCH_TRANS()
- NDB$EPOCH_TRANS()
- NULL

これらの関数は、次のいくつかのパラグラフで説明します。

NDB$OLD(column_name)　column_name の値がマスターとスレーブで同じである場合、更新が適用されます。同じでないと、更新はスレーブに適用されず、例外がログに書き込まれます。これは、次の擬似コードで説明します。

```
if (master_old_column_value == slave_current_column_value)
  apply_update();
else
  log_exception();
```

この関数は「同じ値が優先」競合解決に使用されます。このタイプの競合解決では、誤ったマスターから更新がスレーブに適用されません。

> **重要**
>
> マスターの「前」のイメージのカラム値がこの関数で使用されます。

NDB$MAX(column_name)　マスターからの特定の行の 「タイムスタンプ」カラム値がスレーブでの値より高い場合は、適用されます。高くない場合は、スレーブで適用されません。これは、次の擬似コードで説明します。

```
if (master_new_column_value > slave_current_column_value)
  apply_update();
```

この関数は「もっとも大きいタイムスタンプが優先」競合解決に使用できます。このタイプの競合解決では、競合が発生した場合、最後に更新された行のバージョンが存続するバージョンになります。

> **重要**
>
> マスターの「後」のイメージからのカラム値がこの関数で使用されます。

NDB$MAX_DELETE_WIN(column_name)　これは NDB$MAX() のバリエーションです。タイムスタンプは削除操作に使用できないため、NDB$MAX() を使用する削除は、実際には NDB$OLD として処理されます。ただしユースケースによっては、これは最適ではありません。NDB$MAX_DELETE_WIN() の場合、マスターから既存の行を追加または更新する特定行の 「タイムスタンプ」カラム値が、スレーブでの値より大きい場合に適用されます。ただし、削除操作は常に値が高いものとして処理されます。これは、次の擬似コードで説明します。

```
if ( (master_new_column_value > slave_current_column_value)
       ||
     operation.type == "delete")
  apply_update();
```

この関数は「もっとも大きいタイムスタンプ、削除が優先」競合解決に使用できます。このタイプの競合解決では、競合が発生した場合、削除された行のバージョン、または (そうでなければ) 最後に更新された行のバージョンが存続するバージョンになります。

> **注記**
>
> NDB$MAX() と同様に、マスターの「後」のイメージからのカラム値が、この関数で使用される値です。

NDB$EPOCH() および NDB$EPOCH_TRANS()　NDB$EPOCH() 関数は、複製されたエポックが、スレーブで発生した変更に関連してスレーブの MySQL Cluster で適用される順番を追跡します。この相対的な順番は、スレーブで発生した変更が、ローカルに発生した変更と同時に起こったかどうか、そのために競合が発生する可能性があるかどうかを判断するために使用されます。

NDB$EPOCH() の説明で従う内容のほとんどは、NDB$EPOCH_TRANS() にも適用されます。例外は本文に記載されています。

NDB$EPOCH() は非対象であり、2 つのクラスタの循環レプリケーション構成の一方の MySQL Cluster で動作します (「アクティブアクティブ」レプリケーションと呼ぶこともあります)。ここで、プライマリとして動作するクラスタと、セカンダリとして動作するもう一方のクラスタについて言及します。プライマリのスレーブは競合の検出と処理を担います。一方、セカンダリのスレーブは競合の検出または処理には関与しません。

プライマリのスレーブが競合を検出すると、競合を相殺するためにスレーブ自身のバイナリログにイベントを挿入します。これにより、最終的にセカンダリの MySQL Cluster はプライマリに合うように再調整され、プライマリとセカンダリの不一致が回避されます。この補正と再調整のメカニズムには、プライマリの MySQL Cluster がセカンダリとの競合に常に勝る必要があります。つまり、競合が発生した場合、セカンダリからの変更ではなく、プライマリの変更が常に使用される必要があります。この「プライマリが常に勝つ」ルールには次の意味が込められています。

- プライマリでいったんコミットされると、データを変更する操作は永続し、競合の検出と解決によって取り消されたり、ロールバックされたりしません。
- プライマリから読み取られたデータには一貫性があります。プライマリでコミットされた変更 (ローカルまたはスレーブから) は、あとで取り消せません。
- セカンダリでデータを変更する操作は、競合が発生しているとプライマリが判断した場合、あとで取り消される可能性があります。
- セカンダリで読み取られた各行は、常に自己矛盾がなく、各行は、セカンダリでコミットされた状態、またはプライマリでコミットされた状態のいずれかを常に反映しています。
- セカンダリで読み取られた一連の行は、任意の時点で必ずしも一貫しているわけではありません。NDB$EPOCH_TRANS() の場合、これは一時的な状態であり、NDB$EPOCH() の場合は、永続的な状態となることがあります。
- 十分な期間、競合が発生しないと、セカンダリの MySQL Cluster のすべてのデータは (最終的に) プライマリのデータに一致します。

NDB$EPOCH() および NDB$EPOCH_TRANS() では、競合を検出するためにユーザースキーマを修正したり、アプリケーションを変更したりする必要はありません。ただし、システム全体が指定された制限内で動作することを検証するために、使用するスキーマ、および使用するアクセスパターンを考慮する必要があります。

NDB$EPOCH() および NDB$EPOCH_TRANS() 関数はオプションのパラメータを取ることができます。これはエポックの下位 32 ビットを表すために使用されるビット数であり、次の値以上に設定してください。

```
CEIL( LOG2( TimeBetweenGlobalCheckpoints / TimeBetweenEpochs ), 1)
```

これらの構成パラメータがデフォルト値 (それぞれ、2000 および 100 ミリ秒) である場合、値は 5 バイトになり、デフォルト値 (6) で十分です。ただし、ほかの値が TimeBetweenGlobalCheckpoints、TimeBetweenEpochs、またはその両方に使用される場合を除きます。値が小さすぎると誤判定となる可能性があります。一方、値が大きすぎると、データベース内に無駄なスペースが増える可能性があります。

このセクションの別のところで説明するように、例外テーブルが同じ例外テーブルのスキーマルールに従って定義された場合、NDB$EPOCH() と NDB$EPOCH_TRANS() の両方は競合している行のエントリを関連する例外テーブルに挿入します (NDB$OLD(column_name)を参照してください)。例外テーブルを使用するテーブルを作成する前に、例外テーブルを作成する必要があります。

NDB$EPOCH() および NDB$EPOCH_TRANS() は、このセクションで説明したほかの競合検出関数と同様に、mysql.ndb_replication テーブルに関連するエントリを含めることで起動されます (ndb_replication システムテーブルを参照してください)。このシナリオでのプライマリとセカンダリの MySQL Cluster のロールは、mysql.ndb_replication テーブルのエントリによってすべて決定されます。

NDB$EPOCH() および NDB$EPOCH_TRANS() で使用される競合検出アルゴリズムは非同期であるため、プライマリスレーブとセカンダリスレーブの server_id エントリで異なる値を使用する必要があります。

MySQL Cluster NDB 7.3.6 より前では、DELETE 操作間の競合は UPDATE 操作の競合と同様に処理され、同じエポック内の競合は競合していると見なされていました。MySQL Cluster NDB 7.3.6 以降では、DELETE 操作間の競合だけでは NDB$EPOCH() または NDB$EPOCH_TRANS() を使用して競合を起動するには十分でなく、エポック内の相対的な配置は重要ではありません。(Bug "18454499)

NDB$EPOCH() および NDB$EPOCH_TRANS() ステータス変数　NDB$EPOCH() および NDB$EPOCH_TRANS() の競合検出をモニターするために、いくつかのステータス変数を使用できます。このスレーブが Ndb_conflict_fn_epoch システムステータス変数の現在の値で最後に再起動されたあとに NDB$EPOCH() によって競合中であると検出された行数がわかります。

Ndb_conflict_fn_epoch_trans は NDB$EPOCH_TRANS() によって競合中であると直接検出された行数を示します。実際に再構成された行数 (行のメンバーシップ、またはほかの競合する行と同じトランザクションの依存関係によって影響を受ける行を含みます) は、Ndb_conflict_trans_row_reject_count によって取得されます。

詳細については、セクション18.3.4.4「MySQL Cluster のステータス変数」を参照してください。

NDB$EPOCH() の制約　NDB$EPOCH() を使用して競合検出を実行する場合、現在、次の制約が適用されます。

- TimeBetweenEpochs (デフォルト: 100 ミリ秒) に比例する精度で、競合が MySQL Cluster のエポック境界を使用して検出されます。最小競合ウィンドウは、両方のクラスタの同じデータへの並列更新が常に競合を報告する最小時間です。これは、常にゼロでない時間であり、2 * (latency + queueing + TimeBetweenEpochs) にほぼ比例します。これは、TimeBetweenEpochs にデフォルトを仮定し、またクラスタ間の待機時間 (およびキューイング遅延) を無視して、最小競合ウィンドウが約 200 ミリ秒であることを意味します。この最小ウィンドウは、予期されたアプリケーション「競争」パターンを調べるときに考慮してください。

- NDB$EPOCH() および NDB$EPOCH_TRANS() 関数を使用するテーブルには、追加ストレージが必要です。関数に渡される値によって、1 行当たり 1 ビットから 32 ビットのスペースが必要です。

- 削除操作の競合は、プライマリとセカンダリの間の相違につながる可能性があります。行が両方のクラスタ上で同時に削除される場合、競合は検出できますが、行が削除されるために競合は記録されません。すなわち、後続の再編成操作の伝播中にさらなる競合は検出されず、不一致になる可能性があります。

  削除は外部シリアライズするか、1 つのクラスタだけにルーティングしてください。また、行の削除の間の競合を追跡できるように、このような削除および削除に続く挿入でトランザクションごとに個々の行を更新してください。これには、アプリケーションの変更が必要となる場合があります。

- 競合の検出に NDB$EPOCH() または NDB$EPOCH_TRANS() を使用する場合、循環「アクティブアクティブ」構成の MySQL Cluster が 2 つの場合のみがサポートされています。

- BLOB または TEXT カラムを持つテーブルは、現在 NDB$EPOCH() または NDB$EPOCH_TRANS() ではサポートされていません。

NDB$EPOCH_TRANS()　NDB$EPOCH_TRANS() は NDB$EPOCH() 関数を拡張したものです。「プライマリがすべてに優先」ルール (NDB$EPOCH() および NDB$EPOCH_TRANS()を参照してください) を使用する同じ方法で競合が検出され、処理されますが、競合が発生した同じトランザクションで更新されたその他の行も競合していると見なす、という条件が追加されます。つまり、NDB$EPOCH() がセカンダリの競合している各行を再編成するのに対して、NDB$EPOCH_TRANS() は競合しているトランザクションを再編成します。

また、競合しているトランザクションへの依存が検出されたトランザクションも競合していると見なされ、これらの依存関係はセカンダリクラスタのバイナリログの内容によって判断されます。バイナリログにはデータ変更操作だけ (挿入、更新、削除) が含まれるため、重複するデータの変更だけがトランザクション間の依存関係の判断に使用されます。

NDB$EPOCH_TRANS() は NDB$EPOCH() と同じ条件と制約に従い、さらにバージョン 2 のバイナリログ行イベントが使用されます (--log-bin-use-v1-row-events は 0 に等しい)。これにより、バイナリログ内のイベント当たり 2 バイトのストレージオーバーヘッドが加わります。また、すべてのトランザクション ID をセカンダリのバイナリログに記録する必要があり (--ndb-log-transaction-id オプション)、このため、可変のオーバーヘッド (行当たり最大 13 バイト) がさらに加わります。

NDB$EPOCH() および NDB$EPOCH_TRANS()を参照してください。

NULL　対応するテーブルに競合解決を使用しないことを示します。

ステータス情報　サーバーステータス変数 Ndb_conflict_fn_max は、mysqld が最後に起動されてから、「もっとも大きいタイムスタンプが優先」競合解決によって現在の SQL ノードに行が適用されなかった回数のカウントを示します。

指定された mysqld が最後に再起動されたあとに「同じタイムスタンプが優先」競合解決の結果として行が挿入されなかった回数は、グローバルステータス変数 Ndb_conflict_fn_old によって取得されます。Ndb_conflict_fn_old をインクリメントすること以外に、このセクションの後半の説明のように、使用されなかった行の主キーは 例外テーブルに挿入されます。

競合解決の例外テーブル　NDB$OLD() 競合解決関数を使用するには、このタイプの競合解決が使用される各 NDB テーブルに対応する例外テーブルも作成する必要があります。これは、NDB$EPOCH() または NDB$EPOCH_TRANS() を使用する場合にも当てはまります。このテーブルの名前は、競合解決が適用されるテーブルの名前に文字列 $EX を付加した名前です。(たとえば、元のテーブルの名前が mytable である場合、対応する例外テーブルの名前は mytable$EX になります。)MySQL Cluster NDB 7.4.1 より前では、このテーブルは次のように作成されます。

```
CREATE TABLE original_table$EX  (
    server_id INT UNSIGNED,
    master_server_id INT UNSIGNED,
    master_epoch BIGINT UNSIGNED,
    count INT UNSIGNED,

    original_table_pk_columns,

    [additional_columns,]

    PRIMARY KEY(server_id, master_server_id, master_epoch, count)
) ENGINE=NDB;
```

MySQL Cluster NDB 7.4.1 以降では、例外のタイプ、原因、および元のトランザクションに関する情報を提供するオプションカラムを含む、拡張した例外テーブル定義をサポートしています。これらのバージョンでは、例外テーブルを作成する構文は次のとおりです。

```
CREATE TABLE original_table$EX  (
    [NDB$]server_id INT UNSIGNED,
    [NDB$]master_server_id INT UNSIGNED,
    [NDB$]master_epoch BIGINT UNSIGNED,
    [NDB$]count INT UNSIGNED,

    [NDB$OP_TYPE ENUM('WRITE_ROW','UPDATE_ROW', 'DELETE_ROW',
      'REFRESH_ROW', 'READ_ROW') NOT NULL,]
    [NDB$CFT_CAUSE ENUM('ROW_DOES_NOT_EXIST', 'ROW_ALREADY_EXISTS',
      'DATA_IN_CONFLICT', 'TRANS_IN_CONFLICT') NOT NULL,]
    [NDB$ORIG_TRANSID BIGINT UNSIGNED NOT NULL,]

    original_table_pk_columns,

    [orig_table_column|orig_table_column$OLD|orig_table_column$NEW,]

    [additional_columns,]

    PRIMARY KEY([NDB$]server_id, [NDB$]master_server_id, [NDB$]master_epoch, [NDB$]count)
) ENGINE=NDB;
```

最初の 4 つのカラムが必須です。最初の 4 つのカラムの名前、および元のテーブルの主キーカラムに一致するカラムの名前は重要ではありません。ただし、わかりやすさと一貫性のため、server_id、master_server_id、master_epoch、および count のカラムについてはここで示した名前を使用し、元のテーブルの主キーのカラムに一致するカラムについては元のテーブルと同じ名前を使用することをお勧めします。

MySQL Cluster NDB 7.4.1 以降では、例外テーブルがこのセクションの後半で説明した複数のオプションカラム NDB$OP_TYPE、NDB$CFT_CAUSE、または NDB$ORIG_TRANSID を使用している場合、各必須カラムもプリフィクス NDB$ を使用して名前を付ける必要があります。必要に応じて、オプションカラムを定義しない場合でも NDB$ プリフィクスを使用して必須カラムに名前を付けることができます。ただしこの場合、4 つすべての必須カラムにプリフィクスを使用して名前を付ける必要があります。

このカラムに続き、元のテーブルの主キーを構成するカラムは、元のテーブルの主キーの定義に使用される順番でコピーをしてください。元のテーブルの主キーカラムを複製するカラムのデータ型は、元のカラムと同じ (または大きい) データ型にしてください。MySQL Cluster NDB 7.3 以前では、例外テーブルの主キーはカラム対カラム

で再作成する必要があります。MySQL Cluster NDB 7.4.1 以降では、主キーカラムのサブセットを使用することも可能です。

使用する MySQL Cluster のバージョンにかかわらず、例外テーブルは NDB ストレージエンジンを使用する必要があります。(例外テーブルで NDB$OLD() を使用する例は、このセクションの後半で示します。)

オプションで、コピーされる主キーカラムのあとに追加カラムを定義できますが、その前に追加カラムを定義することはできません。このような追加カラムは NOT NULL にはできません。MySQL Cluster NDB 7.4.1 以降では、事前定義の 3 つの追加オプションカラム NDB$OP_TYPE、NDB$CFT_CAUSE、および NDB$ORIG_TRANSID (次のいくつかのパラグラフで説明します) をサポートしています。

NDB$OP_TYPE: このカラムは、競合の原因となる操作の種類を取得するために使用できます。このカラムを使用する場合、ここで示すように定義します。

```
NDB$OP_TYPE ENUM('WRITE_ROW', 'UPDATE_ROW', 'DELETE_ROW',
  'REFRESH_ROW', 'READ_ROW') NOT NULL
```

WRITE_ROW、UPDATE_ROW、および DELETE_ROW 操作タイプは、ユーザー起動の操作を表します。REFRESH_ROW 操作は、競合を検出したクラスタから元のクラスタに戻されたトランザクションを相殺するときに、競合解決によって生成される操作です。READ_ROW 操作は、排他的行ロックを使用して定義される、ユーザー起動の読み取り追跡操作です。

NDB$CFT_CAUSE: 登録された競合の原因を示すオプションカラム NDB$CFT_CAUSE を定義できます。このカラムは、使用する場合、ここで示すように定義されます。

```
NDB$CFT_CAUSE ENUM('ROW_DOES_NOT_EXIST', 'ROW_ALREADY_EXISTS',
  'DATA_IN_CONFLICT', 'TRANS_IN_CONFLICT') NOT NULL
```

ROW_DOES_NOT_EXIST は UPDATE_ROW および WRITE_ROW 操作の原因として報告できます。ROW_ALREADY_EXISTS は WRITE_ROW イベントに対して報告できます。DATA_IN_CONFLICT は、行ベースの競合関数が競合を検出した場合に報告されます。TRANS_IN_CONFLICT は、トランザクション競合関数がトランザクション全体に属するすべての操作を拒否する場合に報告されます。

NDB$ORIG_TRANSID: NDB$ORIG_TRANSID カラムは、使用する場合、元のトランザクションの ID を格納します。このカラムは次のように定義されます。

```
NDB$ORIG_TRANSID BIGINT UNSIGNED NOT NULL
```

NDB$ORIG_TRANSID は NDB によって生成される 64 ビットの値です。この値は、同じまたは異なる例外テーブルから同じ競合トランザクションに属する複数の例外テーブルのエントリを相互に関連付けるために使用されます。

MySQL Cluster NDB 7.4.1 以降では、更新および削除の操作 (すなわち、DELETE_ROW イベントを含む操作です) で、元のテーブルの主キーの一部ではない追加参照カラムの名前を colname$OLD または colname$NEW. colname$OLD 参照の古い値にできます。colname$NEW は挿入および更新の操作 (言い換えると、WRITE_ROW イベント、UPDATE_ROW イベント、または両方のタイプのイベントを使用する操作です) で新しい値の参照に使用できます。競合中の操作は特定の非主キー参照カラムに値は指定されませんが、例外テーブルの行には NULL またはそのカラムに対して定義されたデフォルト値のいずれかが格納されます。

> **重要**
>
> mysql.ndb_replication テーブルは、データテーブルがレプリケーション用にセットアップされたときに読み取られるため、複製されるテーブルに対応する行は、複製されるテーブルが作成される前に mysql.ndb_replication に挿入する必要があります。

例

この例では、セクション18.6.5「レプリケーションのための MySQL Cluster の準備」およびセクション18.6.6「MySQL Cluster レプリケーションの起動 (レプリケーションチャネルが 1 つ)」で説明したとおり、すでに MySQL Cluster レプリケーションセットアップが正常に機能しているものとします。

NDB$MAX() の例　「タイムスタンプ」 としてカラム mycol を使用して、テーブル test.t1 で「もっとも大きいタイムスタンプが優先」競合解決を有効にするものとします。これは、次のステップで実行できます。

1. --ndb-log-update-as-write=OFF でマスターの mysqld を起動したことを確認します。
2. マスターで、INSERT ステートメントを実行します。

```
INSERT INTO mysql.ndb_replication
```

```
  VALUES ('test', 't1', 0, NULL, 'NDB$MAX(mycol)');
```

server_id に 0 を挿入すると、このテーブルにアクセスするすべての SQL ノードが競合解決を使用します。特定の mysqld だけで競合解決を使用する場合は、実際のサーバー ID を使用します。

binlog_type カラムに NULL を挿入すると、0 (NBT_DEFAULT) の挿入と同じ効果があり、サーバーのデフォルトが使用されます。

3. test.t1 テーブルを作成します。

```
CREATE TABLE test.t1 (
  columns
  mycol INT UNSIGNED,
  columns
) ENGINE=NDB;
```

これで、このテーブルに更新が行われると、競合解決が適用され、mycol のもっとも大きい値を持つ行のバージョンがスレーブに書き込まれます。

> **注記**
>
> ほかの binlog_type オプション (NBT_UPDATED_ONLY_USE_UPDATE など) は、コマンド行オプションを使用するのではなく、ndb_replication テーブルを使用してマスターでロギングを制御するために使用してください。

NDB$OLD() の例　NDB テーブル (ここで定義されたテーブルなど) が複製中であり、このテーブルへの更新に「同じタイムスタンプが優先」競合解決を有効にするものとします。

```
CREATE TABLE test.t2  (
  a INT UNSIGNED NOT NULL,
  b CHAR(25) NOT NULL,
  columns,
  mycol INT UNSIGNED NOT NULL,
  columns,
  PRIMARY KEY pk (a, b)
)  ENGINE=NDB;
```

ここで示す順番で、次のステップが必要です。

1. まず (test.t2 を作成する前に)、ここで示すように mysql.ndb_replication テーブルに行を挿入する必要があります。

```
INSERT INTO mysql.ndb_replication
  VALUES ('test', 't2', 0, NULL, 'NDB$OLD(mycol)');
```

binlog_type カラムに可能な値は、このセクションの前半に示しています。値 'NDB$OLD(mycol)' を conflict_fn カラムに挿入してください。

2. test.t2 に対応する例外テーブルを作成します。ここで示すテーブル作成ステートメントはすべての必須カラムを含みます。これらの必須カラムのあとと、テーブルの主キー定義の前に、追加カラムを宣言する必要があります。

```
CREATE TABLE test.t2$EX  (
  server_id SMALLINT UNSIGNED,
  master_server_id INT UNSIGNED,
  master_epoch BIGINT UNSIGNED,
  count BIGINT UNSIGNED,
  a INT UNSIGNED NOT NULL,
  b CHAR(25) NOT NULL,

  [additional_columns,]

  PRIMARY KEY(server_id, master_server_id, master_epoch, count)
)  ENGINE=NDB;
```

MySQL Cluster NDB 7.4.1 以降では、特定の競合に対するタイプ、原因、および元のトランザクション ID に関する情報のカラムを加えることができます。元のテーブルのすべての主キーカラムに一致するカラムを提供する必要もありません。これらのバージョンでは、次のように例外テーブルを作成できます。

```
CREATE TABLE test.t2$EX  (
  NDB$server_id SMALLINT UNSIGNED,
  NDB$master_server_id INT UNSIGNED,
  NDB$master_epoch BIGINT UNSIGNED,
  NDB$count BIGINT UNSIGNED,
```

```
    a INT UNSIGNED NOT NULL,

    NDB$OP_TYPE ENUM('WRITE_ROW','UPDATE_ROW', 'DELETE_ROW',
      'REFRESH_ROW', 'READ_ROW') NOT NULL,
    NDB$CFT_CAUSE ENUM('ROW_DOES_NOT_EXIST', 'ROW_ALREADY_EXISTS',
      'DATA_IN_CONFLICT', 'TRANS_IN_CONFLICT') NOT NULL,
    NDB$ORIG_TRANSID BIGINT UNSIGNED NOT NULL,

    [additional_columns,]

    PRIMARY KEY(NDB$server_id, NDB$master_server_id, NDB$master_epoch, NDB$count)
)   ENGINE=NDB;
```

少なくとも 1 つのカラム NDB$OP_TYPE、NDB$CFT_CAUSE、または NDB$ORIG_TRANSID をテーブル定義に含めたため、4 つの必須カラムに NDB$ プリフィクスが必要です。

3. 前に示したように、テーブル test.t2 を作成します。

NDB$OLD() を使用して競合解決を実行するテーブルごとに、これらのステップに従う必要があります。このようなテーブルごとに、mysql.ndb_replication に対応する行があり、複製されているテーブルと同じデータベースに例外テーブルがある必要があります。

読み取り競合の検出と解決　MySQL Cluster NDB 7.4.1 以降では、読み取り操作の追跡をサポートしています。これにより、あるクラスタの特定の行の読み取りと、別のクラスタの同じ行の更新または削除との間で競合を管理することが、循環レプリケーションセットアップで可能になります。この例では、employee および department テーブルを使用してシナリオをモデル化します。このシナリオでは、ある従業員がある部門から別の部門にマスタークラスタ (以降、クラスタ A と呼びます) 上で移動し、一方、スレーブクラスタ (以降、B と呼びます) は従業員の前の部門の従業員数をインターリーブされたトランザクションで更新します。

データテーブルは次の SQL ステートメントで作成されました。

```
# Employee table
CREATE TABLE employee (
    id INT PRIMARY KEY,
    name VARCHAR(2000),
    dept INT NOT NULL
)   ENGINE=NDB;

# Department table
CREATE TABLE department (
    id INT PRIMARY KEY,
    name VARCHAR(2000),
    members INT
)   ENGINE=NDB;
```

2 つのテーブルの内容は、次の SELECT ステートメントの出力 (一部) に示される行を含みます。

```
mysql> SELECT id, name, dept FROM employee;
+--------------+------+
| id   | name   | dept |
+------+--------+------+
...
| 998  |  Mike  | 3    |
| 999  |  Joe   | 3    |
| 1000 |  Mary  | 3    |
...
+------+--------+------+

mysql> SELECT id, name, members FROM department;
+-----+-------------+---------+
| id  | name        | members |
+-----+-------------+---------+
...
| 3   | Old project | 24      |
...
+-----+-------------+---------+
```

4 つの必須カラム (これらはこのテーブルの主キーに使用されます)、操作タイプと原因用のオプションカラム、および元のテーブルの主キーカラムを含んだ、ここで示した SQL ステートメントで作成された例外テーブルをすでに使用しているものとします。

```
CREATE TABLE employee$EX  (
    NDB$server_id INT UNSIGNED,
    NDB$master_server_id INT UNSIGNED,
    NDB$master_epoch BIGINT UNSIGNED,
    NDB$count INT UNSIGNED,
```

```
  NDB$OP_TYPE ENUM( 'WRITE_ROW','UPDATE_ROW', 'DELETE_ROW',
              'REFRESH_ROW','READ_ROW') NOT NULL,
  NDB$CFT_CAUSE ENUM( 'ROW_DOES_NOT_EXIST',
                'ROW_ALREADY_EXISTS',
                'DATA_IN_CONFLICT',
                'TRANS_IN_CONFLICT') NOT NULL,

  id INT NOT NULL,

  PRIMARY KEY(NDB$server_id, NDB$master_server_id, NDB$master_epoch, NDB$count)
)  ENGINE=NDB;
```

2 つのクラスタ上で同時に 2 つのトランザクションが発生するものとします。クラスタ A では、新しい部門を作成してから、従業員番号 999 をその部門に移動します。次の SQL ステートメントを使用します。

```
BEGIN;
  INSERT INTO department VALUES (4, "New project", 1);
  UPDATE employee SET dept = 4 WHERE id = 999;
COMMIT;
```

同時にクラスタ B では、次に示すように、別のトランザクションが employee から読み取ります。

```
BEGIN;
  SELECT name FROM employee WHERE id = 999;
  UPDATE department SET members = members - 1  WHERE id = 3;
commit;
```

競合しているトランザクションは、通常は競合解決メカニズムで検出されません。これは、競合が読み取り (SELECT) と更新操作の間で発生しているためです。MySQL Cluster NDB 7.4.1 以降では、スレーブクラスタで SET ndb_log_exclusive_reads = 1 を実行することで、この問題を回避できます。この方法で排他的読み取りロックを取得すると、マスターでの行の読み取りに、スレーブクラスタで競合解決が必要であることを示すフラグが付きます。これらのトランザクションのロギングの前にこの方法で排他的読み取りを有効にすると、クラスタ B での読み取りが追跡され、解決のためにクラスタ A に送られます。従業員の行の競合が検出され、クラスタ B でのトランザクションは中止されます。

競合は、ここで示すように (クラスタ A にある) 例外テーブルに READ_ROW 操作として登録されます (操作タイプの説明については、競合解決の例外テーブルを参照してください)。

```
mysql> SELECT id, NDB$OP_TYPE, NDB$CFT_CAUSE FROM employee$EX;
+-------+------------+------------------+
| id    | NDB$OP_TYPE | NDB$CFT_CAUSE    |
+-------+------------+------------------+
...
| 999   | READ_ROW    | TRANS_IN_CONFLICT |
+-------+------------+------------------+
```

読み取り操作で検出された、存在するどの行にもフラグが付けられます。すなわち、ここで示すように、クラスタ A での更新と、同時に発生したトランザクションで同じテーブルからのクラスタ B での複数行の読み取りとの間で競合の影響を調べることで、例外テーブルに同じ競合に起因する複数の行の口グが取られる場合があります。ここで、クラスタ A で実行されるトランザクションを示します。

```
BEGIN;
  INSERT INTO department VALUES (4, "New project", 0);
  UPDATE employee SET dept = 4 WHERE dept = 3;
  SELECT COUNT(*) INTO @count FROM employee WHERE dept = 4;
  UPDATE department SET members = @count WHERE id = 4;
COMMIT;
```

同時に、ここで示すステートメントを含むトランザクションがクラスタ B で実行されます。

```
SET ndb_log_exclusive_reads = 1;  # Must be set if not already enabled
...
BEGIN;
  SELECT COUNT(*) INTO @count FROM employee WHERE dept = 3 FOR UPDATE;
  UPDATE department SET members = @count WHERE id = 3;
COMMIT;
```

この場合、ここで示すように、2 番目のトランザクションの SELECT の中の WHERE 条件に一致する 3 つすべての行が読み取られ、例外テーブルでフラグが付けられます。

```
mysql> SELECT id, NDB$OP_TYPE, NDB$CFT_CAUSE FROM employee$EX;
+-------+------------+------------------+
| id    | NDB$OP_TYPE | NDB$CFT_CAUSE    |
+-------+------------+------------------+
```

```
...
| 998   | READ_ROW    | TRANS_IN_CONFLICT |
| 999   | READ_ROW    | TRANS_IN_CONFLICT |
| 1000  | READ_ROW    | TRANS_IN_CONFLICT |
...
+-------+-------------+-------------------+
```

読み取りの追跡は、存在する行だけに基づいて行われます。特定条件の追跡に基づく読み取りは、検出された行だけと競合し、インターリーブされたトランザクションで挿入された行とは競合しません。これは、MySQL Cluster の 1 つのインスタンスで排他的行ロックが実行される方法に類似しています。

18.7 MySQL Cluster リリースノート

MySQL Cluster リリースの変更点は、このリファレンスマニュアルとは別個に文書化されています。MySQL Cluster NDB 7.3 の各リリースの変更点に関するリリースノートは、『MySQL Cluster NDB 7.3 リリースノート』にあります。MySQL Cluster NDB 7.4 リリースについては、『MySQL Cluster NDB 7.4 リリースノート』にあります。「MySQL Cluster リリースノート」で MySQL Cluster の古いバージョンのリリースノートを入手することもできます。

第 19 章 パーティション化

目次

19.1 MySQL のパーティショニングの概要 2413
19.2 パーティショニングタイプ 2415
19.2.1 RANGE パーティショニング 2416
19.2.2 LIST パーティショニング 2420
19.2.3 COLUMNS パーティショニング 2422
19.2.4 HASH パーティショニング 2428
19.2.5 KEY パーティショニング 2431
19.2.6 サブパーティショニング 2432
19.2.7 MySQL パーティショニングによる NULL の扱い 2435
19.3 パーティション管理 2439
19.3.1 RANGE および LIST パーティションの管理 2439
19.3.2 HASH および KEY パーティションの管理 2444
19.3.3 パーティションとサブパーティションをテーブルと交換する 2445
19.3.4 パーティションの保守 2450
19.3.5 パーティションに関する情報を取得する 2451
19.4 パーティションプルーニング 2453
19.5 パーティション選択 2455
19.6 パーティショニングの制約と制限 2460
19.6.1 パーティショニングキー、主キー、および一意キー 2465
19.6.2 ストレージエンジンに関連するパーティショニング制限 2468
19.6.3 関数に関連するパーティショニング制限 2469
19.6.4 パーティショニングとロック 2470

この章では、MySQL でのユーザー定義パーティショニングの実装について説明します。使用している MySQL サーバーがパーティショニングをサポートしているかどうかを判別するには、次のように SHOW PLUGINS ステートメントの出力を確認します。

> **注記**
>
> MySQL の以前のバージョンには、have_partitioning 変数がありましたが、非推奨となり、MySQL 5.6.1 で削除されました。

```
mysql> SHOW PLUGINS;
+------------+----------+----------------+---------+---------+
| Name       | Status   | Type           | Library | License |
+------------+----------+----------------+---------+---------+
| binlog     | ACTIVE   | STORAGE ENGINE | NULL    | GPL     |
| partition  | ACTIVE   | STORAGE ENGINE | NULL    | GPL     |
| ARCHIVE    | ACTIVE   | STORAGE ENGINE | NULL    | GPL     |
| BLACKHOLE  | ACTIVE   | STORAGE ENGINE | NULL    | GPL     |
| CSV        | ACTIVE   | STORAGE ENGINE | NULL    | GPL     |
| FEDERATED  | DISABLED | STORAGE ENGINE | NULL    | GPL     |
| MEMORY     | ACTIVE   | STORAGE ENGINE | NULL    | GPL     |
| InnoDB     | ACTIVE   | STORAGE ENGINE | NULL    | GPL     |
| MRG_MYISAM | ACTIVE   | STORAGE ENGINE | NULL    | GPL     |
| MyISAM     | ACTIVE   | STORAGE ENGINE | NULL    | GPL     |
| ndbcluster | DISABLED | STORAGE ENGINE | NULL    | GPL     |
+------------+----------+----------------+---------+---------+
11 rows in set (0.00 sec)
```

次のようなクエリーを使用して INFORMATION_SCHEMA.PLUGINS テーブルを確認することもできます。

```
mysql> SELECT
    ->     PLUGIN_NAME as Name,
    ->     PLUGIN_VERSION as Version,
    ->     PLUGIN_STATUS as Status
    -> FROM INFORMATION_SCHEMA.PLUGINS
    -> WHERE PLUGIN_TYPE='STORAGE ENGINE';
+--------------------+---------+--------+
| Name               | Version | Status |
+--------------------+---------+--------+
| binlog             | 1.0     | ACTIVE |
| CSV                | 1.0     | ACTIVE |
| MEMORY             | 1.0     | ACTIVE |
```

```
| MRG_MYISAM         | 1.0     | ACTIVE |
| MyISAM             | 1.0     | ACTIVE |
| PERFORMANCE_SCHEMA | 0.1     | ACTIVE |
| BLACKHOLE          | 1.0     | ACTIVE |
| ARCHIVE            | 3.0     | ACTIVE |
| InnoDB             | 5.6     | ACTIVE |
| partition          | 1.0     | ACTIVE |
+--------------------+---------+--------+
10 rows in set (0.00 sec)
```

どちらの場合も、出力で partition プラグインの Status カラムに値 ACTIVE が示されていない (それぞれの例で太字で示されています) 場合、使用しているバージョンの MySQL にはパーティショニングサポートが組み込まれていません。

オラクルによって提供されている MySQL 5.6 Community バイナリには、パーティショニングのサポートが含まれています。商用の MySQL サーバーバイナリで提供されているパーティショニングサポートについては、第25章「MySQL Enterprise Edition」を参照してください。

ソースから MySQL 5.6 をコンパイルする場合にパーティショニングを有効にするには、-DWITH_PARTITION_STORAGE_ENGINE オプションを指定してビルドを構成する必要があります。詳細は、セクション2.9「ソースから MySQL をインストールする」を参照してください。

使用している MySQL バイナリにパーティショニングサポートが組み込まれている場合は、それを有効にするために何も行う必要はありません (たとえば、my.cnf ファイルに特別なエントリを指定する必要はありません)。

パーティショニングサポートを無効にするには、--skip-partition オプションを指定して MySQL サーバーを起動します。その場合は、have_partitioning の値が DISABLED になります。パーティショニングサポートを無効にすると、既存のパーティション化されたテーブルの表示、およびそれらの削除は行うことができますが (ただし、これを行うことはお勧めしません)、それ以外にそれらを操作したり、データにアクセスしたりすることはできません。

パーティショニングの概要およびパーティショニングの概念については、セクション19.1「MySQL のパーティショニングの概要」を参照してください。

MySQL では、いくつかのタイプのパーティショニングおよびサブパーティショニングをサポートしています。セクション19.2「パーティショニングタイプ」およびセクション19.2.6「サブパーティショニング」を参照してください。

セクション19.3「パーティション管理」では、既存のパーティション化されたテーブルでパーティションを追加、削除および変更する方法について説明しています。

セクション19.3.4「パーティションの保守」では、パーティション化されたテーブルで使用するテーブル保守コマンドについて説明しています。

INFORMATION_SCHEMA データベースの PARTITIONS テーブルには、パーティションおよびパーティション化されたテーブルに関する情報があります。詳細は、セクション21.13「INFORMATION_SCHEMA PARTITIONS テーブル」を参照してください。このテーブルに対するクエリーのいくつかの例については、セクション19.2.7「MySQL パーティショニングによる NULL の扱い」を参照してください。

MySQL 5.6 のパーティショニングの既知の問題については、セクション19.6「パーティショニングの制約と制限」を参照してください。

また、パーティション化されたテーブルを使用して作業を行うときに、次のリソースが役に立つことがあります。

追加のリソース　MySQL のユーザー定義パーティショニングに関するその他の情報ソースには、次のものがあります。

- MySQL パーティショニングフォーラム

  これは、MySQL パーティショニングテクノロジに興味があり実験している人のための公式ディスカッションフォーラムです。MySQL 開発者などからの発表や更新が掲載されています。パーティショニングの開発チームおよびドキュメントチームのメンバーによってモニターされています。

- Mikael Ronström のブログ

  MySQL パーティショニングの設計者、およびリード開発者である Mikael Ronström が、MySQL パーティショニングおよび MySQL Cluster に関する記事を頻繁に掲載しています。

- PlanetMySQL

MySQL を使用している人が関心を持つと思われる MySQL 関連ブログをまとめた MySQL ニュースサイト。MySQL パーティショニングを使用している人が更新しているブログへのリンクをここでチェックしたり、自分のブログを追加したりすることをお勧めします。

MySQL 5.6 バイナリは、https://dev.mysql.com/downloads/mysql/5.6.htmlから入手できます。ただし、最新のパーティショニングバグ修正および機能追加が必要な場合は、Bazaar リポジトリからソースを取得できます。パーティショニングを有効にするには、-DWITH_PARTITION_STORAGE_ENGINE オプションを指定してビルドを構成する必要があります。MySQL のビルドについては、セクション2.9「ソースから MySQL をインストールする」を参照してください。パーティションを有効にした MySQL 5.6 ビルドのコンパイルで問題が発生した場合は、「MySQL Partitioning Forum」をチェックして、問題の解決策がそこにない場合は支援を要請してください。

19.1 MySQL のパーティショニングの概要

このセクションでは、MySQL 5.6 のパーティショニングの概念について説明します。

パーティショニングの制約および機能制限については、セクション19.6「パーティショニングの制約と制限」を参照してください。

SQL 標準では、データ保存の物理的な仕様に関するガイダンスはあまり提供されていません。SQL 言語自体が、それが動作するスキーマ、テーブル、行、またはカラムの基盤となるデータ構造やメディアと独立して動作するように意図されています。それにもかかわらず、ほとんどの高度なデータベース管理システムでは、ファイルシステム、ハードウェア、またはその両方について、特定のデータを格納するために使用される物理的な場所を判別する方法が開発されてきました。MySQL では、InnoDB ストレージエンジンによってテーブルスペースの認識がサポートされてきており、MySQL サーバーは、パーティショニングが導入される前から、異なるデータベースの格納に異なる物理的なディレクトリを使用するように構成できました (これを行う方法については、セクション8.11.3.1「シンボリックリンクの使用」を参照してください)。

パーティショニングはこの認識をさらに一歩進めて、必要に応じて多くの部分を設定できるルールに従って、個々のテーブルの部分をファイルシステムに配分できるようにしています。それにより、テーブルの異なる部分が別個のテーブルとして別個の場所に格納されます。データを分割するためにユーザーが選択するルールはパーティショニング関数と呼ばれ、MySQL では法、範囲セットまたは値リストに対する単純な照合、内部ハッシュ関数、または線形ハッシュ関数が使用されます。関数は、ユーザーが指定したパーティショニングタイプに従って選択され、ユーザーが指定した式の値をパラメータとして取ります。この式には、使用されるパーティショニングのタイプに応じて、カラム値、1 つ以上のカラム値を操作する関数、または 1 つ以上のカラム値のセットを指定できます。

RANGE、LIST、および [LINEAR] HASH パーティショニングの場合、パーティショニングカラムの値はパーティショニング関数に渡され、特定のレコードを格納すべきパーティションの番号を表す整数値が返されます。この関数は非定数および非ランダムである必要があります。クエリーを含めることはできませんが、式が NULL または次のような整数 intval を返すかぎり、MySQL で有効な SQL 式を使用できます。

```
-MAXVALUE <= intval <= MAXVALUE
```

(MAXVALUE は対象となる整数型の上限を表すために使用されます。-MAXVALUE は下限を表します。)

[LINEAR] KEY、RANGE COLUMNS、および LIST COLUMNS パーティショニングの場合、パーティショニング式は 1 つ以上のカラムのリストから構成されます。

[LINEAR] KEY パーティショニングの場合、パーティショニング関数は MySQL によって提供されます。

許可されるパーティショニングカラムタイプおよびパーティショニング関数については、セクション19.2「パーティショニングタイプ」、およびパーティショニング構文の説明および追加例を示しているセクション13.1.17「CREATE TABLE 構文」を参照してください。パーティショニング関数の制約については、セクション19.6.3「関数に関連するパーティショニング制限」を参照してください。

これは「水平パーティショニング」と呼ばれます。つまり、テーブル内の異なる行を異なる物理パーティションに割り当てることができます。MySQL 5.6 は、テーブルの異なるカラムが異なる物理パーティションに割り当てられる、「垂直パーティショニング」はサポートしていません。現時点で、垂直パーティショニングを MySQL 5.6 に導入する計画はありません。

MySQL サーバーバイナリがユーザー定義パーティショニングをサポートしているかどうかの判断については、第19章「パーティション化」を参照してください。

パーティション化されたテーブルを作成する場合、MySQL サーバーによってサポートされているほとんどのストレージエンジンを使用できます。MySQL パーティショニングエンジンは別のレイヤーで実行されており、それらのいずれともやり取りできます。MySQL 5.6 では、同じパーティション化されたテーブルのすべてのパーティションは同じストレージエンジンを使用する必要があります。たとえば、あるパーティションに MyISAM、別の

ものに InnoDB を使用することはできません。ただし、同じ MySQL サーバーまたは同じデータベース上の異なるパーティション化されたテーブルに、異なるストレージエンジンを使用することはできます。

MySQL パーティショニングは、MERGE、CSV、または FEDERATED ストレージエンジンで使用できません。

KEY または LINEAR KEY によるパーティショニングは NDB で使用できますが、ほかのタイプのユーザー定義パーティショニングはこのストレージエンジンを使用するテーブルでサポートされません。また、ユーザー定義パーティショニングを使用する NDB テーブルには明示的な主キーが必要であり、テーブルのパーティショニング式で参照されるカラムは主キーの一部である必要があります。ただし、ユーザーパーティション化された NDB テーブルを作成または変更するために使用される CREATE TABLE または ALTER TABLE ステートメントの PARTITION BY KEY 句または PARTITION BY LINEAR KEY 句にカラムが 1 つもリストされていない場合は、テーブルに明示的な主キーは必要ありません。詳細は、セクション18.1.6.1「MySQL Cluster での SQL 構文の不適合」を参照してください。

パーティション化されたテーブルに特定のストレージエンジンを使用するために必要なのは、パーティション化されていないテーブルの場合と同様に、[STORAGE] ENGINE オプションを使用することだけです。ただし、[STORAGE] ENGINE (およびほかのテーブルオプション) は、CREATE TABLE ステートメントでパーティショニングオプションを使用する前に指定する必要があります。次の例は、ハッシュによって 6 個のパーティションにパーティション化され、InnoDB ストレージエンジンを使用するテーブルの作成方法を示しています。

```
CREATE TABLE ti (id INT, amount DECIMAL(7,2), tr_date DATE)
    ENGINE=INNODB
    PARTITION BY HASH( MONTH(tr_date) )
    PARTITIONS 6;
```

各 PARTITION 句に [STORAGE] ENGINE オプションを含めることはできますが、MySQL 5.6 ではこれは効果がありません。

> **重要**
>
> パーティショニングはテーブルのすべてのデータおよびインデックスに適用されます。データだけにパーティション化しインデックスは行わないことはできず (その逆も不可)、テーブルの一部のみをパーティション化することもできません。

各パーティションのデータおよびインデックスを特定のディレクトリに割り当てるには、パーティション化されたテーブルを作成するために使用する CREATE TABLE ステートメントの PARTITION 句に DATA DIRECTORY および INDEX DIRECTORY オプションを使用します。

Windows 上の MyISAM テーブルの個別のパーティションまたはサブパーティションでは、DATA DIRECTORY および INDEX DIRECTORY はサポートされません。これらは InnoDB テーブルの個別のパーティションおよびサブパーティションでサポートされます (すべてのプラットフォームと同様)。

また、MAX_ROWS および MIN_ROWS は、各パーティションに格納できる行のそれぞれ最大数および最小数を決定するために使用できます。MAX_ROWS オプションは、MySQL Cluster テーブルを追加パーティションで作成し、ハッシュインデックスにより大きなストレージを確保するために役立つことがあります。詳細は、DataMemory データノード構成パラメータのドキュメント、および セクション18.1.2「MySQL Cluster のノード、ノードグループ、レプリカ、およびパーティション」を参照してください。

パーティショニングのいくつかの利点を次に示します。

- パーティショニングを使用すると、単一ディスクまたはファイルシステムパーティションに保持できるデータより多くのデータを 1 つのテーブルに格納できます。
- 有効性を失っているデータは、多くの場合、そのデータのみが含まれているパーティションを削除することによって、パーティション化されたテーブルから簡単に削除できます。反対に、新しいデータを追加する処理は、そのデータだけを格納するための 1 つ以上の新しいパーティションを追加することによって、非常に便利になることがあります。
- 指定された WHERE 句を満たすデータを 1 つ以上のパーティションのみに格納できることによって、検索からほかのパーティションが自動的に除外され、一部のクエリーが大幅に最適化されることがあります。パーティションはパーティション化されたテーブルが作成されたあとに変更できるため、使用頻度の高いクエリー (パーティショニングスキームが最初に設定されたときはあまり使用されていなかった) を改善するためにデータを再編成できます。一致しないパーティション (およびそれらに含まれている行) を除外するこの機能は、よくパーティションプルーニングと呼ばれます。詳細は、セクション19.4「パーティションプルーニング」を参照してください。

  また、MySQL 5.6 ではクエリーで明示的なパーティション選択がサポートされます。たとえば、SELECT * FROM t PARTITION (p0,p1) WHERE c < 5 は、パーティション p0 および p1 の WHERE 条件に一致する行のみを選択します。この場合、MySQL はテーブル t のほかのパーティションをチェックしません。これにより、

検査するパーティションが事前にわかっているときにクエリー速度が大幅に向上することがあります。パーティション選択は、データ変更ステートメント (DELETE、INSERT、REPLACE、UPDATE、LOAD DATA、および LOAD XML) でもサポートされます。詳細および例については、これらのステートメントの説明を参照してください。

通常はパーティショニングに関連付けられるその他の利点を次に示します。これらの機能は MySQL パーティショニングに現在実装されていませんが、オラクルの優先事項リストの上位にあります。

- SUM()、COUNT() などの集約関数を含むクエリーを簡単に並列化できます。そのようなクエリーの簡単な例として、SELECT salesperson_id, COUNT(orders) as order_total FROM sales GROUP BY salesperson_id; を挙げられます。「並列化」することで、クエリーを各パーティションで同時に実行してから、すべてのパーティションで取得される結果を単に集約するだけで最終的な結果が得られることを意味します。
- 複数のディスクにデータシークを分散することによって、クエリーのスループットが向上します。

MySQL パーティショニング開発は継続されているため、このセクションおよび章が更新されているかどうかを頻繁にチェックしてください。

19.2 パーティショニングタイプ

このセクションでは、MySQL 5.6 で使用できるパーティショニングのタイプについて説明します。これらには、次に一覧したタイプが含まれます。

- RANGE パーティショニング　このタイプのパーティショニングは、指定された範囲に含まれるカラム値に基づいて、行をパーティションに割り当てます。セクション19.2.1「RANGE パーティショニング」を参照してください。このタイプを拡張した RANGE COLUMNS については、セクション19.2.3.1「RANGE COLUMNS パーティショニング」を参照してください。
- LIST パーティショニング　RANGE によるパーティショニングに似ていますが、別個の値のセットのいずれかに一致するカラムに基づいて、パーティションが選択されます。セクション19.2.2「LIST パーティショニング」を参照してください。このタイプを拡張した LIST COLUMNS については、セクション19.2.3.2「LIST COLUMNS パーティショニング」を参照してください。
- HASH パーティショニング　このタイプのパーティショニングでは、テーブルに挿入される行内のカラム値を操作するユーザー定義式によって返される値に基づいて、パーティションが選択されます。関数は、負ではない整数値を返す MySQL の有効な式で構成できます。このタイプを拡張した LINEAR HASH も使用できます。セクション19.2.4「HASH パーティショニング」を参照してください。
- KEY パーティショニング　このタイプのパーティショニングは、HASH によるパーティショニングに似ていますが、評価される 1 つ以上のカラムのみを指定し、MySQL サーバーが独自のハッシュ関数を提供します。MySQL によって提供されるハッシュ関数ではカラムデータ型に関係なく整数結果が保証されるため、これらのカラムに整数以外の値が含まれていてもかまいません。このタイプを拡張した LINEAR KEY も使用できます。セクション19.2.5「KEY パーティショニング」を参照してください。

データベースパーティショニングの非常に一般的な使用方法は、日付によってデータを分けることです。一部のデータベースシステムは、MySQL 5.6 では実装されていない、明示的な日付パーティショニングをサポートしています。ただし、MySQL で、DATE、TIME、または DATETIME カラムに基づいて、またはそのようなカラムを使用する式に基づいて、パーティショニングスキームを作成することは難しくありません。

KEY または LINEAR KEY でパーティショニングする場合は、DATE、TIME、または DATETIME カラムを、カラム値の変更を実行しないパーティショニングカラムとして使用できます。たとえば、次のテーブル作成ステートメントは MySQL で完全に有効です。

```
CREATE TABLE members (
    firstname VARCHAR(25) NOT NULL,
    lastname VARCHAR(25) NOT NULL,
    username VARCHAR(16) NOT NULL,
    email VARCHAR(35),
    joined DATE NOT NULL
)
PARTITION BY KEY(joined)
PARTITIONS 6;
```

MySQL 5.6 では、RANGE COLUMNS および LIST COLUMNS パーティショニングを使用して、DATE または DATETIME カラムをパーティショニングカラムとして使用することもできます。

ただし、MySQL のほかのパーティショニングタイプでは、整数値または NULL を返すパーティショニング式が必要です。RANGE、LIST、HASH、または LINEAR HASH による日付ベースパーティショニングを使用する場合

は、次のように単純に DATE、TIME、または DATETIME カラムを操作してそのような値を返す関数を使用できます。

```
CREATE TABLE members (
    firstname VARCHAR(25) NOT NULL,
    lastname VARCHAR(25) NOT NULL,
    username VARCHAR(16) NOT NULL,
    email VARCHAR(35),
    joined DATE NOT NULL
)
PARTITION BY RANGE( YEAR(joined) ) (
    PARTITION p0 VALUES LESS THAN (1960),
    PARTITION p1 VALUES LESS THAN (1970),
    PARTITION p2 VALUES LESS THAN (1980),
    PARTITION p3 VALUES LESS THAN (1990),
    PARTITION p4 VALUES LESS THAN MAXVALUE
);
```

日付を使用したパーティショニングの追加例は、この章の次のセクションにあります。

- セクション19.2.1「RANGE パーティショニング」
- セクション19.2.4「HASH パーティショニング」
- セクション19.2.4.1「LINEAR HASH パーティショニング」

日付ベースパーティショニングのより複雑な例については、次のセクションを参照してください。

- セクション19.4「パーティションプルーニング」
- セクション19.2.6「サブパーティショニング」

MySQL パーティショニングは、TO_DAYS()、YEAR()、および TO_SECONDS() 関数で使用するために最適化されています。ただし、整数または NULL を返すほかの日時関数 (WEEKDAY()、DAYOFYEAR()、MONTH() など) を使用できます。そのような関数の詳細については、セクション12.7「日付および時間関数」を参照してください。

使用するパーティショニングのタイプにかかわらず、パーティションには作成時に常に 0 から始まる番号が順番に自動的に付けられることを覚えておくことが重要です。新しい行がパーティション化されたテーブルに挿入されるときは、これらのパーティション番号が正しいパーティションを識別するために使用されます。たとえば、テーブルで 4 つのパーティションが使用される場合、これらのパーティションには 0、1、2、および 3 という番号が付けられます。RANGE および LIST パーティショニングタイプの場合は、各パーティション番号のパーティションが定義されている必要があります。HASH パーティショニングの場合は、使用されるユーザー関数が 0 より大きい整数値を返す必要があります。KEY パーティショニングの場合は、MySQL サーバーが内部で使用しているハッシュ関数によって、この問題が自動的に対処されます。

パーティションの名前は通常、ほかの MySQL 識別子 (テーブル名、データベース名など) を制御するルールに従っています。ただし、パーティション名では大文字/小文字が区別されません。たとえば、次の CREATE TABLE ステートメントは、示されているように失敗します。

```
mysql> CREATE TABLE t2 (val INT)
    -> PARTITION BY LIST(val)(
    ->     PARTITION mypart VALUES IN (1,3,5),
    ->     PARTITION MyPart VALUES IN (2,4,6)
    -> );
ERROR 1488 (HY000): Duplicate partition name mypart
```

失敗は、MySQL がパーティション名 mypart と MyPart の違いを認識できないために発生します。

テーブルのパーティション番号を指定するときは、先行ゼロなしのゼロ以外の正の整数リテラルとして表現する必要があり、0.8E+01 や 6-2 などの式であってはいけません (これが整数値に評価されるとしても)。小数は許可されません。

以降のセクションでは、各パーティションタイプの作成に使用できるすべての形式の構文を提供しているわけではありません。この情報については、セクション13.1.17「CREATE TABLE 構文」も参照してみてください。

19.2.1 RANGE パーティショニング

範囲によってパーティション化されるテーブルは、各パーティションに含まれる行のパーティショニング式値が指定された範囲に収まるようにパーティション化されます。範囲は、連続しているけれども重複しないものであるべきで、VALUES LESS THAN 演算子を使用して定義されます。次のいくつかの例では、20 のビデオ店で構成されるチェーン (1 から 20 までの番号が付けられている) の従業員レコードを保持する、次のようなテーブルを作成していると想定してください。

```
CREATE TABLE employees (
    id INT NOT NULL,
    fname VARCHAR(30),
    lname VARCHAR(30),
    hired DATE NOT NULL DEFAULT '1970-01-01',
    separated DATE NOT NULL DEFAULT '9999-12-31',
    job_code INT NOT NULL,
    store_id INT NOT NULL
);
```

> **注記**
>
> ここで使用する employees テーブルには主キーまたは一意キーがありません。これらの例はここでの説明のためのもので、実際のテーブルは主キー、一意キー、またはその両方を備えている可能性がきわめて高く、パーティショニングカラムに利用できる選択肢はこれらのキー (ある場合) に使用されるカラムに依存します。これらの事項については、セクション19.6.1「パーティショニングキー、主キー、および一意キー」を参照してください。

このテーブルは、必要に応じていくつかの方法で、範囲によるパーティション化を実行できます。1 つの方法は、store_id カラムを使用することです。たとえば、次のように PARTITION BY RANGE 句を追加することで、テーブルを 4 つのパーティションに分割することを決定できます。

```
CREATE TABLE employees (
    id INT NOT NULL,
    fname VARCHAR(30),
    lname VARCHAR(30),
    hired DATE NOT NULL DEFAULT '1970-01-01',
    separated DATE NOT NULL DEFAULT '9999-12-31',
    job_code INT NOT NULL,
    store_id INT NOT NULL
)
PARTITION BY RANGE (store_id) (
    PARTITION p0 VALUES LESS THAN (6),
    PARTITION p1 VALUES LESS THAN (11),
    PARTITION p2 VALUES LESS THAN (16),
    PARTITION p3 VALUES LESS THAN (21)
);
```

このパーティショニングスキームでは、店舗 1 から店舗 5 で働いている従業員に対応するすべての行がパーティション p0 に格納され、店舗 6 から店舗 10 の従業員がパーティション p1 に格納されます (以下同様)。各パーティションは、もっとも小さい値からもっとも大きい値の順で定義されます。これは PARTITION BY RANGE 構文の要件です。これは、C または Java の一連の if ... elseif ... ステートメントに似ていると考えることができます。

データ (72, 'Michael', 'Widenius', '1998-06-25', NULL, 13) が含まれている新しい行がパーティション p2 に挿入されることは簡単に判断できますが、このチェーンに 21 番目の店舗が追加されたらどうなるでしょうか。このスキームでは、store_id が 20 よりも大きい行に対応するルールがなく、サーバーはどこに置くべきかがわからないため、エラーになります。これが発生しないようにするには、「すべての状況に対応する」 VALUES LESS THAN 句を CREATE TABLE ステートメントで使用して、明示的に指定されている最大値を超えるすべての値に備えます。

```
CREATE TABLE employees (
    id INT NOT NULL,
    fname VARCHAR(30),
    lname VARCHAR(30),
    hired DATE NOT NULL DEFAULT '1970-01-01',
    separated DATE NOT NULL DEFAULT '9999-12-31',
    job_code INT NOT NULL,
    store_id INT NOT NULL
)
PARTITION BY RANGE (store_id) (
    PARTITION p0 VALUES LESS THAN (6),
    PARTITION p1 VALUES LESS THAN (11),
    PARTITION p2 VALUES LESS THAN (16),
    PARTITION p3 VALUES LESS THAN MAXVALUE
);
```

> **注記**
>
> 一致する値が見つからないときにエラーを回避するための別の方法は、INSERT ステートメントの一部として IGNORE キーワードを使用することです。例については、セクション19.2.2「LIST パーティショニング」を参照してください。また、IGNORE の一般的な情報については、セクション13.2.5「INSERT 構文」を参照してください。

MAXVALUE は、可能な最大整数値よりも確実に大きな整数値を表します (数学用語では、上限です)。これによって、store_id カラム値が 16 (定義されている最大値) 以上である行は、パーティション p3 に格納されます。将来のある時点で、店舗の数が 25、30、またはそれ以上に増加したときは、ALTER TABLE ステートメントを使用して、店舗 21-25、26-30 などのための新しいパーティションを追加できます (これを行う方法の詳細については、セクション19.3「パーティション管理」を参照してください)。

同様に、従業員ジョブコードに基づいて (つまり、job_code カラム値の範囲に基づいて) テーブルをパーティション化できます。たとえば、正規 (インストア) 従業員に 2 桁のジョブコード、オフィスおよびサポート従業員に 3 桁のコード、および管理職に 4 桁のコードが使用されると想定すると、次のステートメントを使用してパーティション化されたテーブルを作成できます。

```
CREATE TABLE employees (
    id INT NOT NULL,
    fname VARCHAR(30),
    lname VARCHAR(30),
    hired DATE NOT NULL DEFAULT '1970-01-01',
    separated DATE NOT NULL DEFAULT '9999-12-31',
    job_code INT NOT NULL,
    store_id INT NOT NULL
)
PARTITION BY RANGE (job_code) (
    PARTITION p0 VALUES LESS THAN (100),
    PARTITION p1 VALUES LESS THAN (1000),
    PARTITION p2 VALUES LESS THAN (10000)
);
```

この例では、インストア従業員に関連するすべての行はパーティション p0、オフィスおよびサポートスタッフに関連するものは p1、および管理職に関連するものはパーティション p2 に格納されます。

VALUES LESS THAN 句に式を使用することもできます。ただし、MySQL が式の戻り値を LESS THAN (<) 比較の一部として評価できる必要があります。

店舗番号に従ってテーブルデータを分割するのではなく、代わりに 2 つの DATE カラムのうちの 1 つに基づく式を使用できます。たとえば、各従業員が会社を退職した年 (つまり、YEAR(separated) の値) に基づいてパーティション化するとします。そのようなパーティショニングスキームを実装する CREATE TABLE ステートメントの例を次に示します。

```
CREATE TABLE employees (
    id INT NOT NULL,
    fname VARCHAR(30),
    lname VARCHAR(30),
    hired DATE NOT NULL DEFAULT '1970-01-01',
    separated DATE NOT NULL DEFAULT '9999-12-31',
    job_code INT,
    store_id INT
)
PARTITION BY RANGE ( YEAR(separated) ) (
    PARTITION p0 VALUES LESS THAN (1991),
    PARTITION p1 VALUES LESS THAN (1996),
    PARTITION p2 VALUES LESS THAN (2001),
    PARTITION p3 VALUES LESS THAN MAXVALUE
);
```

このスキームでは、1991 年より前に退職したすべての従業員の場合、行はパーティション p0 に格納されます。1991 年から 1995 年までに退職した人は p1、1996 年から 2000 年までに退職した人は p2、および 2000 年よりあとに退職した従業員は p3 に格納されます。

次の例に示すように、UNIX_TIMESTAMP() 関数を使用して、TIMESTAMP カラムの値に基づいて、RANGE によってテーブルをパーティション化することもできます。

```
CREATE TABLE quarterly_report_status (
    report_id INT NOT NULL,
    report_status VARCHAR(20) NOT NULL,
    report_updated TIMESTAMP NOT NULL DEFAULT CURRENT_TIMESTAMP ON UPDATE CURRENT_TIMESTAMP
)
PARTITION BY RANGE ( UNIX_TIMESTAMP(report_updated) ) (
    PARTITION p0 VALUES LESS THAN ( UNIX_TIMESTAMP('2008-01-01 00:00:00') ),
    PARTITION p1 VALUES LESS THAN ( UNIX_TIMESTAMP('2008-04-01 00:00:00') ),
    PARTITION p2 VALUES LESS THAN ( UNIX_TIMESTAMP('2008-07-01 00:00:00') ),
    PARTITION p3 VALUES LESS THAN ( UNIX_TIMESTAMP('2008-10-01 00:00:00') ),
    PARTITION p4 VALUES LESS THAN ( UNIX_TIMESTAMP('2009-01-01 00:00:00') ),
    PARTITION p5 VALUES LESS THAN ( UNIX_TIMESTAMP('2009-04-01 00:00:00') ),
    PARTITION p6 VALUES LESS THAN ( UNIX_TIMESTAMP('2009-07-01 00:00:00') ),
    PARTITION p7 VALUES LESS THAN ( UNIX_TIMESTAMP('2009-10-01 00:00:00') ),
```

```
    PARTITION p8 VALUES LESS THAN ( UNIX_TIMESTAMP('2010-01-01 00:00:00') ),
    PARTITION p9 VALUES LESS THAN (MAXVALUE)
);
```

TIMESTAMP 値を含むほかの式は許可されません(Bug #42849 を参照してください)。

次の条件の 1 つ以上が true のときは、RANGE パーティショニングが特に役立ちます。

- 「古い」データを削除したい、またはする必要がある。直前のパーティショニングスキームを使用している場合は、単純に ALTER TABLE employees DROP PARTITION p0; を使用して 1991 年より前に会社を退職した従業員に関連するすべての行を削除できます。(詳細は、セクション13.1.7「ALTER TABLE 構文」およびセクション19.3「パーティション管理」を参照してください)。テーブルに多数の行がある場合、これは DELETE FROM employees WHERE YEAR(separated) <= 1990; などの DELETE クエリーを実行するよりもはるかに効率的な場合があります。
- 日付または時間値、または何らかのほかの一連値から生じる値が含まれるカラムを使用したい。
- テーブルのパーティショニングに使用されるカラムに直接依存するクエリーを頻繁に実行する。たとえば、EXPLAIN PARTITIONS SELECT COUNT(*) FROM employees WHERE separated BETWEEN '2000-01-01' AND '2000-12-31' GROUP BY store_id; などのクエリーを実行する場合、MySQL はパーティション p2 のみをスキャンする必要であることをすばやく判断できます (残りのパーティションに WHERE 句を満たすレコードが含まれるはずがないため)。これがどのように実現されるかについての詳細は、セクション19.4「パーティションプルーニング」を参照してください。

このタイプのパーティショニングのバリアントが RANGE COLUMNS パーティショニングです。RANGE COLUMNS によるパーティショニングでは、複数のカラムを使用してパーティショニング範囲を定義できます (パーティション内での行の配置、およびパーティションプルーニングを実行するときに特定のパーティションの包含または除外を判断する際に適用されます)。詳細は、セクション19.2.3.1「RANGE COLUMNS パーティショニング」を参照してください。

時間間隔に基づくパーティショニングスキーム　　MySQL 5.6 で時間の範囲または間隔に基づいてパーティショニングスキームを実装する場合は、2 つの方法があります。

1. 次のように、RANGE によってテーブルをパーティション化し、パーティショニング式に DATE、TIME、または DATETIME カラムを操作して整数値を返す関数を使用します。

   ```
   CREATE TABLE members (
       firstname VARCHAR(25) NOT NULL,
       lastname VARCHAR(25) NOT NULL,
       username VARCHAR(16) NOT NULL,
       email VARCHAR(35),
       joined DATE NOT NULL
   )
   PARTITION BY RANGE( YEAR(joined) ) (
       PARTITION p0 VALUES LESS THAN (1960),
       PARTITION p1 VALUES LESS THAN (1970),
       PARTITION p2 VALUES LESS THAN (1980),
       PARTITION p3 VALUES LESS THAN (1990),
       PARTITION p4 VALUES LESS THAN MAXVALUE
   );
   ```

   MySQL 5.6 では、次の例に示すように UNIX_TIMESTAMP() 関数を使用して、TIMESTAMP カラムの値に基づいて RANGE によってテーブルをパーティション化することもできます。

   ```
   CREATE TABLE quarterly_report_status (
       report_id INT NOT NULL,
       report_status VARCHAR(20) NOT NULL,
       report_updated TIMESTAMP NOT NULL DEFAULT CURRENT_TIMESTAMP ON UPDATE CURRENT_TIMESTAMP
   )
   PARTITION BY RANGE ( UNIX_TIMESTAMP(report_updated) ) (
       PARTITION p0 VALUES LESS THAN ( UNIX_TIMESTAMP('2008-01-01 00:00:00') ),
       PARTITION p1 VALUES LESS THAN ( UNIX_TIMESTAMP('2008-04-01 00:00:00') ),
       PARTITION p2 VALUES LESS THAN ( UNIX_TIMESTAMP('2008-07-01 00:00:00') ),
       PARTITION p3 VALUES LESS THAN ( UNIX_TIMESTAMP('2008-10-01 00:00:00') ),
       PARTITION p4 VALUES LESS THAN ( UNIX_TIMESTAMP('2009-01-01 00:00:00') ),
       PARTITION p5 VALUES LESS THAN ( UNIX_TIMESTAMP('2009-04-01 00:00:00') ),
       PARTITION p6 VALUES LESS THAN ( UNIX_TIMESTAMP('2009-07-01 00:00:00') ),
       PARTITION p7 VALUES LESS THAN ( UNIX_TIMESTAMP('2009-10-01 00:00:00') ),
       PARTITION p8 VALUES LESS THAN ( UNIX_TIMESTAMP('2010-01-01 00:00:00') ),
       PARTITION p9 VALUES LESS THAN (MAXVALUE)
   );
   ```

   MySQL 5.6 では、TIMESTAMP 値を含むほかの式は許可されません。(Bug #42849 を参照してください)。

> **注記**
>
> MySQL 5.6 では、LIST によってパーティション化されるテーブルのパーティショニング式として UNIX_TIMESTAMP(timestamp_column) を使用することもできます。ただし、このようにするのは通常は実用的ではありません。

2. DATE または DATETIME カラムをパーティショニングカラムとして使用して、RANGE COLUMNS によってテーブルをパーティション化します。たとえば、次のように joined カラムを直接使用して members テーブルを定義できます。

```
CREATE TABLE members (
    firstname VARCHAR(25) NOT NULL,
    lastname VARCHAR(25) NOT NULL,
    username VARCHAR(16) NOT NULL,
    email VARCHAR(35),
    joined DATE NOT NULL
)
PARTITION BY RANGE COLUMNS(joined) (
    PARTITION p0 VALUES LESS THAN ('1960-01-01'),
    PARTITION p1 VALUES LESS THAN ('1970-01-01'),
    PARTITION p2 VALUES LESS THAN ('1980-01-01'),
    PARTITION p3 VALUES LESS THAN ('1990-01-01'),
    PARTITION p4 VALUES LESS THAN MAXVALUE
);
```

> **注記**
>
> DATE または DATETIME 以外の日付または時間型を使用してパーティショニングカラムを使用することは、RANGE COLUMNS ではサポートされません。

19.2.2 LIST パーティショニング

MySQL の LIST パーティショニングは、多くの点で RANGE パーティショニングに似ています。RANGE によるパーティショニングと同様に、各パーティションを明示的に定義する必要があります。2 つのタイプのパーティショニングの主な違いは、LIST パーティショニングでは、各パーティションが、連続する値の範囲のセットのいずれかではなく、値リストのセットのいずれかに含まれるカラム値のメンバーシップに基づいて定義および選択されることです。これを行うには、PARTITION BY LIST (expr) を使用します。ここで、expr はカラム値またはカラム値に基づく式で、整数値を返し、VALUES IN (value_list) で各パーティションを定義します。ここで、value_list はカンマで区切られた整数のリストです。

> **注記**
>
> MySQL 5.6 では、LIST によってパーティション化するときに、整数 (および NULL も可。セクション19.2.7「MySQL パーティショニングによる NULL の扱い」を参照してください) のリストに対してのみ照合できます。
>
> ただし、LIST COLUMN パーティショニングを使用するときは、ほかのカラムタイプを値リストで使用できます (これについては、このセクションで後述します)。

範囲で定義されるパーティションの場合と異なり、リストパーティションは特定の順序で宣言する必要はありません。構文についての詳細は、セクション13.1.17「CREATE TABLE 構文」を参照してください。

以降の例では、パーティション化するテーブルの基本定義が、次に示す CREATE TABLE ステートメントによって提供されることを想定します。

```
CREATE TABLE employees (
    id INT NOT NULL,
    fname VARCHAR(30),
    lname VARCHAR(30),
    hired DATE NOT NULL DEFAULT '1970-01-01',
    separated DATE NOT NULL DEFAULT '9999-12-31',
    job_code INT,
    store_id INT
);
```

(これは、セクション19.2.1「RANGE パーティショニング」の例の出発点として使用したものと同じテーブルです。)

次の表に示すように、20 のビデオ店があり、それらが 4 つのフランチャイズに分類されていると想定します。

地域	店舗 ID 番号
北	3、5、6、9、17
東	1、2、10、11、19、20
西	4、12、13、14、18
中央	7、8、15、16

同じ地域に属する店舗の行が同じパーティションに格納されるようにこのテーブルをパーティション化するには、次のような CREATE TABLE ステートメントを使用できます。

```
CREATE TABLE employees (
    id INT NOT NULL,
    fname VARCHAR(30),
    lname VARCHAR(30),
    hired DATE NOT NULL DEFAULT '1970-01-01',
    separated DATE NOT NULL DEFAULT '9999-12-31',
    job_code INT,
    store_id INT
)
PARTITION BY LIST(store_id) (
    PARTITION pNorth VALUES IN (3,5,6,9,17),
    PARTITION pEast VALUES IN (1,2,10,11,19,20),
    PARTITION pWest VALUES IN (4,12,13,14,18),
    PARTITION pCentral VALUES IN (7,8,15,16)
);
```

これにより、特定の地域に関連する従業員レコードをテーブルで簡単に追加または削除できるようになります。たとえば、西地域の全店舗が別の会社に売却されたとします。MySQL 5.6 では、その地域の店舗で働いていた従業員に関連するすべての行をクエリー ALTER TABLE employees TRUNCATE PARTITION pWest を使用して削除できます。これは、同等の DELETE ステートメント DELETE FROM employees WHERE store_id IN (4,12,13,14,18); よりもはるかに効率的に実行できます(ALTER TABLE employees DROP PARTITION pWest を使用してもこれらのすべての行が削除されますが、テーブルの定義からパーティション pWest も削除されるため、ALTER TABLE ... ADD PARTITION ステートメントを使用してテーブルの元のパーティショニングスキームをリストアする必要があります)。

RANGE パーティショニングと同様に、LIST パーティショニングとハッシュまたはキーによるパーティショニングを組み合わせることによって、複合パーティショニング (サブパーティショニング) を生成できます。セクション19.2.6「サブパーティショニング」を参照してください。

RANGE パーティショニングの場合と異なり、MAXVALUE などの「すべての状況に対応する」ものはありません。パーティショニング式で予期されるすべての値を PARTITION ... VALUES IN (...) 句で指定してください。一致しないパーティショニングカラム値が含まれている INSERT ステートメントは、次の例に示すように、エラーで失敗します。

```
mysql> CREATE TABLE h2 (
    ->   c1 INT,
    ->   c2 INT
    -> )
    -> PARTITION BY LIST(c1) (
    ->   PARTITION p0 VALUES IN (1, 4, 7),
    ->   PARTITION p1 VALUES IN (2, 5, 8)
    -> );
Query OK, 0 rows affected (0.11 sec)

mysql> INSERT INTO h2 VALUES (3, 5);
ERROR 1525 (HY000): Table has no partition for value 3
```

単一 INSERT ステートメントを使用して複数の行を挿入するときの動作は、テーブルがトランザクションストレージエンジンを使用するかどうかによって変わります。InnoDB テーブルの場合、ステートメントは単一トランザクションと見なされ、一致しない値があるときはステートメントが完全に失敗し、行は挿入されません。MyISAM などの非トランザクションストレージエンジンを使用するテーブルの場合、一致しない値が含まれている行の前にある行は挿入されますが、あとにある行はされません。

このタイプのエラーは、IGNORE キーワードを使用することで無視させることができます。そうした場合、一致しないパーティショニングカラム値が含まれる行は挿入されませんが、一致する値を持つ行は挿入されてエラーが報告されません。

```
mysql> TRUNCATE h2;
Query OK, 1 row affected (0.00 sec)

mysql> SELECT * FROM h2;
Empty set (0.00 sec)
```

```
mysql> INSERT IGNORE INTO h2 VALUES (2, 5), (6, 10), (7, 5), (3, 1), (1, 9);
Query OK, 3 rows affected (0.00 sec)
Records: 5  Duplicates: 2  Warnings: 0

mysql> SELECT * FROM h2;
+------+------+
| c1   | c2   |
+------+------+
|   7 |   5 |
|   1 |   9 |
|   2 |   5 |
+------+------+
3 rows in set (0.00 sec)
```

MySQL 5.6 は LIST COLUMNS パーティショニングのサポートを提供します。これは、LIST パーティショニングのバリアントで、パーティショニングカラムに整数型以外の型のカラムを使用したり、複数のカラムをパーティショニングキーとして使用したりできます。詳細は、セクション19.2.3.2「LIST COLUMNS パーティショニング」を参照してください。

19.2.3 COLUMNS パーティショニング

次の 2 つのセクションでは、RANGE および LIST パーティショニングのバリアントである COLUMNS パーティショニングについて説明します。COLUMNS パーティショニングでは、パーティショニングキーに複数のカラムを使用できます。これらのすべてのカラムが、パーティションに行を配置するため、およびパーティションプルーニングでどのパーティションで一致する行をチェックするかを判断するという両方の目的のために考慮されます。

また、RANGE COLUMNS パーティショニングおよび LIST COLUMNS パーティショニングの両方が、値範囲またはリストメンバーの定義のために整数以外のカラムの使用をサポートします。許可されるデータ型を次のリストに示します。

- すべての整数型: TINYINT、SMALLINT、MEDIUMINT、INT (INTEGER)、および BIGINT(これは、RANGE および LIST によるパーティショニングと同じです)。

  ほかの数値データ型 (DECIMAL、FLOAT など) はパーティショニングカラムとしてサポートされません。

- DATE および DATETIME。

  日付または時間に関連するほかのデータ型を使用するカラムは、パーティショニングカラムとしてサポートされません。

- 次の文字列型: CHAR、VARCHAR、BINARY、および VARBINARY。

  TEXT カラムおよび BLOB カラムはパーティショニングカラムとしてサポートされません。

次の 2 つのセクションでの RANGE COLUMNS および LIST COLUMNS パーティショニングの説明では、MySQL 5.1 以降でサポートされる範囲およびリストに基づくパーティショニングをすでに理解していることを想定しています。これらについての詳細は、セクション19.2.1「RANGE パーティショニング」およびセクション19.2.2「LIST パーティショニング」をそれぞれ参照してください。

19.2.3.1 RANGE COLUMNS パーティショニング

RANGE COLUMNS パーティショニングは RANGE パーティショニングに似ていますが、複数のカラム値に基づく範囲を使用してパーティションを定義できます。また、整数型以外の型のカラムを使用して範囲を定義できます。

RANGE COLUMNS パーティショニングは、次の点で RANGE パーティショニングと大きく異なります。

- RANGE COLUMNS は式を受け入れません (カラムの名前のみ)。

- RANGE COLUMNS は 1 つ以上のカラムのリストを受け入れます。

  RANGE COLUMNS パーティションは、スカラー値の比較ではなく、タプル (カラム値のリスト) の比較に基づきます。RANGE COLUMNS パーティションでの行の配置も、タプルの比較に基づきます。これについては、このセクションで後述します。

- RANGE COLUMNS パーティショニングカラムは整数カラムに制限されません。文字列、DATE、および DATETIME カラムもパーティショニングカラムとして使用できます。(詳細は、セクション19.2.3「COLUMNS パーティショニング」を参照してください)。

RANGE COLUMNS によってパーティション化されたテーブルを作成するための基本構文を次に示します。

```
CREATE TABLE table_name
PARTITIONED BY RANGE COLUMNS(column_list) (
    PARTITION partition_name VALUES LESS THAN (value_list)[,
    PARTITION partition_name VALUES LESS THAN (value_list)][,
    ...]
)

column_list:
    column_name[, column_name][, ...]

value_list:
    value[, value][, ...]
```

> **注記**
>
> パーティション化されたテーブルを作成するときに使用できる CREATE TABLE オプションをすべて示しているわけではありません。詳細は、セクション13.1.17「CREATE TABLE 構文」を参照してください。

前述の構文で、column_list は 1 つ以上のカラムのリスト (パーティショニングカラムリストと呼ばれることもあります)、value_list は値のリスト (つまり、パーティション定義値リストです) です。value_list は各パーティション定義に指定する必要があり、各 value_list には column_list のカラムと同じ数の値が必要です。一般的に、COLUMNS 句に N 個のカラムを使用する場合は、各 VALUES LESS THAN 句にも N 個の値のリストを指定する必要があります。

パーティショニングカラムリストおよび各パーティションを定義する値リスト内の要素は、同じ順序で指定する必要があります。また、値リスト内の各要素は、カラムリスト内の対応する要素と同じデータ型である必要があります。ただし、パーティショニングカラムリストおよび値リスト内のカラム名の順序は、CREATE TABLE ステートメントの主要部内のテーブルカラム定義の順序と同じである必要はありません。RANGE によってパーティション化されるテーブルと同様に、MAXVALUE を使用して、指定されたカラムに挿入される正当な値より確実に大きな値を表すことができます。これらの点をすべて説明するために役立つ CREATE TABLE ステートメントの例を次に示します。

```
mysql> CREATE TABLE rcx (
    ->     a INT,
    ->     b INT,
    ->     c CHAR(3),
    ->     d INT
    -> )
    -> PARTITION BY RANGE COLUMNS(a,d,c) (
    ->     PARTITION p0 VALUES LESS THAN (5,10,'ggg'),
    ->     PARTITION p1 VALUES LESS THAN (10,20,'mmmm'),
    ->     PARTITION p2 VALUES LESS THAN (15,30,'sss'),
    ->     PARTITION p3 VALUES LESS THAN (MAXVALUE,MAXVALUE,MAXVALUE)
    -> );
Query OK, 0 rows affected (0.15 sec)
```

テーブル rcx にはカラム a、b、c、および d が含まれています。COLUMNS 句に指定されたパーティショニングカラムリストには、これらのカラムのうちの 3 つが a、d、および c の順に使用されています。パーティションを定義するために使用される各値リストには、同じ順序で 3 つの値が含まれます。つまり、各値リストタプルの形式は、カラム a、d、および c が (この順序で) 使用するデータ型に対応する、(INT, INT, CHAR(3)) です。

パーティションに行がどのように配置されるかは、挿入される行のタプル (COLUMNS 句内のカラムリストに一致) とVALUES LESS THAN 句に使用されるタプル (テーブルのパーティションを定義) を比較することによって判断されます。スカラー値ではなくタプル (つまり、値のリストまたはセット) を比較するため、RANGE COLUMNS パーティションで使用される VALUES LESS THAN のセマンティクスは、単純な RANGE パーティションの場合とは若干異なります。RANGE パーティショニングでは、VALUES LESS THAN 内の制限値と等しい式値を生成する行は、対応するパーティションに配置されません。ただし、RANGE COLUMNS パーティショニングを使用するときは、パーティショニングカラムリストの最初の要素が VALUES LESS THAN 値リスト内の最初の要素と値が等しい行が、対応するパーティションに配置されることがあります。

次のステートメントによって作成されるパーティション化された RANGE テーブルを検討します。

```
CREATE TABLE r1 (
    a INT,
    b INT
)
PARTITION BY RANGE (a)  (
    PARTITION p0 VALUES LESS THAN (5),
    PARTITION p1 VALUES LESS THAN (MAXVALUE)
);
```

各行の a のカラム値が 5 である 3 つの行をこのテーブルに挿入する場合、各行の a カラム値が 5 以上であるため、3 行がすべてパーティション p1 に格納されます。これは、INFORMATION_SCHEMA.PARTITIONS テーブルに対して適切なクエリーを実行することによって確認できます。

```
mysql> INSERT INTO r1 VALUES (5,10), (5,11), (5,12);
Query OK, 3 rows affected (0.00 sec)
Records: 3  Duplicates: 0  Warnings: 0

mysql> SELECT PARTITION_NAME,TABLE_ROWS
    ->     FROM INFORMATION_SCHEMA.PARTITIONS
    ->     WHERE TABLE_NAME = 'r1';
+----------------+------------+
| PARTITION_NAME | TABLE_ROWS |
+----------------+------------+
| p0             |          0 |
| p1             |          3 |
+----------------+------------+
2 rows in set (0.00 sec)
```

ここで、次のように作成される、カラム a および b の両方が COLUMNS 句で参照される、RANGE COLUMNS パーティショニングを使用する同様のテーブル rc1 を検討します。

```
CREATE TABLE rc1 (
    a INT,
    b INT
)
PARTITION BY RANGE COLUMNS(a, b) (
    PARTITION p0 VALUES LESS THAN (5, 12),
    PARTITION p3 VALUES LESS THAN (MAXVALUE, MAXVALUE)
);
```

r1 に挿入したのとまったく同じ行を rc1 に挿入した場合、行の配分はかなり異なります。

```
mysql> INSERT INTO rc1 VALUES (5,10), (5,11), (5,12);
Query OK, 3 rows affected (0.00 sec)
Records: 3  Duplicates: 0  Warnings: 0

mysql> SELECT PARTITION_NAME,TABLE_ROWS
    ->     FROM INFORMATION_SCHEMA.PARTITIONS
    ->     WHERE TABLE_NAME = 'rc1';
+--------------+----------------+------------+
| TABLE_SCHEMA | PARTITION_NAME | TABLE_ROWS |
+--------------+----------------+------------+
| p            | p0             |          2 |
| p            | p1             |          1 |
+--------------+----------------+------------+
2 rows in set (0.00 sec)
```

これは、スカラー値ではなく行を比較しているためです。挿入された行値と、テーブル rc1 のパーティション p0 を定義するために使用された VALUES THAN LESS THAN 句の行制限値とを次のように比較できます。

```
mysql> SELECT (5,10) < (5,12), (5,11) < (5,12), (5,12) < (5,12);
+-----------------+-----------------+-----------------+
| (5,10) < (5,12) | (5,11) < (5,12) | (5,12) < (5,12) |
+-----------------+-----------------+-----------------+
|               1 |               1 |               0 |
+-----------------+-----------------+-----------------+
1 row in set (0.00 sec)
```

2 つのタプル (5,10) および (5,11) は (5,12) より小さいと評価されるため、パーティション p0 に格納されています。5 は 5 以上、12 は 12 以上であるため、(5,12) は (5,12) 以上と見なされ、パーティション p1 に格納されています。

前の例の SELECT ステートメントは、次のように明示的な行コンストラクタを使用して記述することもできました。

```
SELECT ROW(5,10) < ROW(5,12), ROW(5,11) < ROW(5,12), ROW(5,12) < ROW(5,12);
```

MySQL で行コンストラクタを使用する方法についての詳細は、セクション13.2.10.5「行サブクエリー」を参照してください。

単一パーティショニングカラムのみを使用して RANGE COLUMNS によってパーティション化されたテーブルの場合、パーティションへの行の格納は RANGE によってパーティション化された同等のテーブルの場合と同じです。次の CREATE TABLE ステートメントでは、1 つのパーティショニングカラムを使用して RANGE COLUMNS によってパーティション化されるテーブルが作成されます。

```
CREATE TABLE rx (
    a INT,
    b INT
)
PARTITION BY RANGE COLUMNS (a)  (
    PARTITION p0 VALUES LESS THAN (5),
    PARTITION p1 VALUES LESS THAN (MAXVALUE)
);
```

このテーブルに行 (5,10)、(5,11)、および (5,12) を挿入する場合、それらの配置は、前に作成して移入したテーブル r の場合と同じです。

```
mysql> INSERT INTO rx VALUES (5,10), (5,11), (5,12);
Query OK, 3 rows affected (0.00 sec)
Records: 3  Duplicates: 0  Warnings: 0

mysql> SELECT PARTITION_NAME,TABLE_ROWS
    ->     FROM INFORMATION_SCHEMA.PARTITIONS
    ->     WHERE TABLE_NAME = 'rx';
+--------------+----------------+------------+
| TABLE_SCHEMA | PARTITION_NAME | TABLE_ROWS |
+--------------+----------------+------------+
| p            | p0             |          0 |
| p            | p1             |          3 |
+--------------+----------------+------------+
2 rows in set (0.00 sec)
```

1 つ以上のカラムの制限値が連続するパーティション定義で繰り返される、RANGE COLUMNS によってパーティション化されるテーブルを作成することもできます。これを行うには、パーティションを定義するために使用されるカラム値のタプルが厳密にしだいに増加する必要があります。たとえば、次の各 CREATE TABLE ステートメントは有効です。

```
CREATE TABLE rc2 (
    a INT,
    b INT
)
PARTITION BY RANGE COLUMNS(a,b) (
    PARTITION p0 VALUES LESS THAN (0,10),
    PARTITION p1 VALUES LESS THAN (10,20),
    PARTITION p2 VALUES LESS THAN (10,30),
    PARTITION p3 VALUES LESS THAN (MAXVALUE,MAXVALUE)
 );

CREATE TABLE rc3 (
    a INT,
    b INT
)
PARTITION BY RANGE COLUMNS(a,b) (
    PARTITION p0 VALUES LESS THAN (0,10),
    PARTITION p1 VALUES LESS THAN (10,20),
    PARTITION p2 VALUES LESS THAN (10,30),
    PARTITION p3 VALUES LESS THAN (10,35),
    PARTITION p4 VALUES LESS THAN (20,40),
    PARTITION p5 VALUES LESS THAN (MAXVALUE,MAXVALUE)
 );
```

次のステートメントも成功しますが、カラム b の制限値がパーティション p0 で 25 およびパーティション p1 で 20、カラム c の制限値がパーティション p1 で 100 およびパーティション p2 で 50 であるため、一見したところではそうでないように見えるかもしれません。

```
CREATE TABLE rc4 (
    a INT,
    b INT,
    c INT
)
PARTITION BY RANGE COLUMNS(a,b,c) (
    PARTITION p0 VALUES LESS THAN (0,25,50),
    PARTITION p1 VALUES LESS THAN (10,20,100),
    PARTITION p2 VALUES LESS THAN (10,30,50)
    PARTITION p3 VALUES LESS THAN (MAXVALUE,MAXVALUE,MAXVALUE)
 );
```

RANGE COLUMNS によってパーティション化されるテーブルを設計するときは、次のように mysql クライアントを使用して目的のタプルを比較することで、いつでも連続するパーティション定義をテストできます。

```
mysql> SELECT (0,25,50) < (10,20,100), (10,20,100) < (10,30,50);
+-----------------------+------------------------+
```

```
| (0,25,50) < (10,20,100) | (10,20,100) < (10,30,50) |
+-------------------------+--------------------------+
|                       1 |                        1 |
+-------------------------+--------------------------+
1 row in set (0.00 sec)
```

CREATE TABLE ステートメントに含まれるパーティション定義が、厳密にしだいに増加しない順序である場合は、次の例に示すようにエラーで失敗します。

```
mysql> CREATE TABLE rcf (
    ->     a INT,
    ->     b INT,
    ->     c INT
    -> )
    -> PARTITION BY RANGE COLUMNS(a,b,c) (
    ->     PARTITION p0 VALUES LESS THAN (0,25,50),
    ->     PARTITION p1 VALUES LESS THAN (20,20,100),
    ->     PARTITION p2 VALUES LESS THAN (10,30,50),
    ->     PARTITION p3 VALUES LESS THAN (MAXVALUE,MAXVALUE,MAXVALUE)
    ->  );
ERROR 1493 (HY000): VALUES LESS THAN value must be strictly increasing for each partition
```

そのようなエラーが発生するときは、それらのカラムリストの「小なり」比較を作成することで、無効なパーティション定義を推定できます。この場合、問題はパーティション p2 の定義にあります。次に示すように、それを定義するために使用されるタプルが、パーティション p3 を定義するために使用されるタプル以上であるためです。

```
mysql> SELECT (0,25,50) < (20,20,100), (20,20,100) < (10,30,50);
+-------------------------+--------------------------+
| (0,25,50) < (20,20,100) | (20,20,100) < (10,30,50) |
+-------------------------+--------------------------+
|                       1 |                        0 |
+-------------------------+--------------------------+
1 row in set (0.00 sec)
```

RANGE COLUMNS を使用するときは、複数の VALUES LESS THAN 句内の同じカラムに MAXVALUE を指定することもできます。ただし、それ以外の場合は、連続するパーティション定義内の個々のカラムの制限値はしだいに増加するべきであり、MAXVALUE をすべてのカラム値の上限として使用するパーティションは 1 つだけ定義するべきであり、このパーティション定義は PARTITION ... VALUES LESS THAN 句のリストの最後に指定するべきです。また、複数のパーティション定義の最初のカラムの制限値として MAXVALUE を使用することはできません。

前述したように、RANGE COLUMNS パーティショニングでは、整数以外のカラムをパーティショニングカラムとして使用することもできます(これらの完全な一覧については、セクション19.2.3「COLUMNS パーティショニング」を参照してください)。次のステートメントを使用して作成された employees という名前のテーブル (パーティション化されていません) を検討します。

```
CREATE TABLE employees (
    id INT NOT NULL,
    fname VARCHAR(30),
    lname VARCHAR(30),
    hired DATE NOT NULL DEFAULT '1970-01-01',
    separated DATE NOT NULL DEFAULT '9999-12-31',
    job_code INT NOT NULL,
    store_id INT NOT NULL
);
```

RANGE COLUMNS パーティショニングを使用して、次のように従業員の姓に基づいて各行を 4 つのパーティションのいずれかに格納する、このテーブルのバージョンを作成できます。

```
CREATE TABLE employees_by_lname (
    id INT NOT NULL,
    fname VARCHAR(30),
    lname VARCHAR(30),
    hired DATE NOT NULL DEFAULT '1970-01-01',
    separated DATE NOT NULL DEFAULT '9999-12-31',
    job_code INT NOT NULL,
    store_id INT NOT NULL
)
PARTITION BY RANGE COLUMNS (lname)  (
    PARTITION p0 VALUES LESS THAN ('g'),
    PARTITION p1 VALUES LESS THAN ('m'),
    PARTITION p2 VALUES LESS THAN ('t'),
    PARTITION p3 VALUES LESS THAN (MAXVALUE)
);
```

または、次の ALTER TABLE ステートメントを実行することで、前に作成した employees テーブルをこのスキームを使用してパーティション化できます。

```
ALTER TABLE employees PARTITION BY RANGE COLUMNS (lname)  (
    PARTITION p0 VALUES LESS THAN ('g'),
    PARTITION p1 VALUES LESS THAN ('m'),
    PARTITION p2 VALUES LESS THAN ('t'),
    PARTITION p3 VALUES LESS THAN (MAXVALUE)
);
```

> **注記**
>
> 文字セットおよび照合順序が異なるとソート順序が異なるため、文字列カラムをパーティショニングカラムとして使用するときに、使用している文字セットおよび照合順序が、指定された行が RANGE COLUMNS によってパーティション化されるテーブルのどのパーティションに格納されるかに影響することがあります。また、そのようなテーブルが作成されたあとに指定されたデータベース、テーブル、またはカラムの文字セットまたは照合順序を変更すると、行がどのように配分されるかが変わることがあります。たとえば、大文字/小文字が区別される照合順序を使用するときは、'and' は 'Andersen' の前に置かれますが、大文字/小文字が区別されない照合順序を使用するときは、その逆が true になります。
>
> MySQL が文字セットおよび照合順序をどのように扱うかについては、セクション 10.1「文字セットのサポート」を参照してください。

同様に、employees テーブルを、各行がいくつかの 10 年間 (その間に対応する従業員が雇用された) ベースのパーティションのいずれかに格納されるように、次のような ALTER TABLE ステートメントを使用してパーティション化できます。

```
ALTER TABLE employees PARTITION BY RANGE COLUMNS (hired)  (
    PARTITION p0 VALUES LESS THAN ('1970-01-01'),
    PARTITION p1 VALUES LESS THAN ('1980-01-01'),
    PARTITION p2 VALUES LESS THAN ('1990-01-01'),
    PARTITION p3 VALUES LESS THAN ('2000-01-01'),
    PARTITION p4 VALUES LESS THAN ('2010-01-01'),
    PARTITION p5 VALUES LESS THAN (MAXVALUE)
);
```

PARTITION BY RANGE COLUMNS 構文の詳細については、セクション13.1.17「CREATE TABLE 構文」を参照してください。

19.2.3.2 LIST COLUMNS パーティショニング

MySQL 5.6 は LIST COLUMNS パーティショニングのサポートを提供します。これは LIST パーティショニングのバリアントで、複数のカラムをパーティションキーとして使用でき、整数型以外のデータ型のカラムをパーティショニングカラムとして使用できます。文字列型、DATE、および DATETIME カラムを使用できます (COLUMNS パーティショニングカラムに許可されるデータ型の詳細については、セクション19.2.3「COLUMNS パーティショニング」を参照してください)。

ある会社の顧客が 12 の都市に存在し、販売およびマーケティングのために、それらを次の表に示すように 3 つの都市で構成される 4 つの地域に分類すると想定します。

地域	都市
1	Oskarshamn、Högsby、Mönsterås
2	Vimmerby、Hultsfred、Västervik
3	Nässjö、Eksjö、Vetlanda
4	Uppvidinge、Alvesta、Växjo

LIST COLUMNS パーティショニングでは、ここで示すように、顧客が所在する都市の名前に基づいてこれらの地域に対応する 4 つのパーティションのいずれかに行を割り当てる、顧客データのテーブルを作成できます。

```
CREATE TABLE customers_1 (
    first_name VARCHAR(25),
    last_name VARCHAR(25),
    street_1 VARCHAR(30),
    street_2 VARCHAR(30),
    city VARCHAR(15),
    renewal DATE
```

```
)
PARTITION BY LIST COLUMNS(city) (
    PARTITION pRegion_1 VALUES IN('Oskarshamn', 'Högsby', 'Mönsterås'),
    PARTITION pRegion_2 VALUES IN('Vimmerby', 'Hultsfred', 'Västervik'),
    PARTITION pRegion_3 VALUES IN('Nässjö', 'Eksjö', 'Vetlanda'),
    PARTITION pRegion_4 VALUES IN('Uppvidinge', 'Alvesta', 'Växjo')
);
```

RANGE COLUMNS によるパーティショニングのように、COLUMNS() 句で式を使用してカラム値を整数に変換する必要はありません(実際、カラム名ではなく式を使用することは COLUMNS() では許可されません)。

DATE および DATETIME カラムを使用することもでき、次の例では、前に示した customers_1 テーブルと同じ名前およびカラムを使用していますが、renewal カラムに基づく LIST COLUMNS パーティショニングを使用して、顧客のアカウントの更新がスケジュールされている 2010 年 2 月の週に応じて、4 つのパーティションのいずれかに行が格納されることを示しています。

```
CREATE TABLE customers_2 (
    first_name VARCHAR(25),
    last_name VARCHAR(25),
    street_1 VARCHAR(30),
    street_2 VARCHAR(30),
    city VARCHAR(15),
    renewal DATE
)
PARTITION BY LIST COLUMNS(renewal) (
    PARTITION pWeek_1 VALUES IN('2010-02-01', '2010-02-02', '2010-02-03',
        '2010-02-04', '2010-02-05', '2010-02-06', '2010-02-07'),
    PARTITION pWeek_2 VALUES IN('2010-02-08', '2010-02-09', '2010-02-10',
        '2010-02-11', '2010-02-12', '2010-02-13', '2010-02-14'),
    PARTITION pWeek_3 VALUES IN('2010-02-15', '2010-02-16', '2010-02-17',
        '2010-02-18', '2010-02-19', '2010-02-20', '2010-02-21'),
    PARTITION pWeek_4 VALUES IN('2010-02-22', '2010-02-23', '2010-02-24',
        '2010-02-25', '2010-02-26', '2010-02-27', '2010-02-28')
);
```

これは機能しますが、関係する日付の数が非常に多くなってきた場合に、定義および保守が面倒になります。そのような場合は通常、RANGE または RANGE COLUMNS パーティショニングを代わりに使用するほうが現実的です。この場合、パーティショニングキーとして使用するカラムは DATE カラムであるため、次に示すように RANGE COLUMNS パーティショニングを使用します。

```
CREATE TABLE customers_3 (
    first_name VARCHAR(25),
    last_name VARCHAR(25),
    street_1 VARCHAR(30),
    street_2 VARCHAR(30),
    city VARCHAR(15),
    renewal DATE
)
PARTITION BY RANGE COLUMNS(renewal) (
    PARTITION pWeek_1 VALUES LESS THAN('2010-02-09'),
    PARTITION pWeek_2 VALUES LESS THAN('2010-02-15'),
    PARTITION pWeek_3 VALUES LESS THAN('2010-02-22'),
    PARTITION pWeek_4 VALUES LESS THAN('2010-03-01')
);
```

詳細は、セクション19.2.3.1「RANGE COLUMNS パーティショニング」を参照してください。

また (RANGE COLUMNS パーティショニングと同様に)、COLUMNS() 句で複数のカラムを使用できます。

PARTITION BY LIST COLUMNS() 構文についての詳細は、セクション13.1.17「CREATE TABLE 構文」を参照してください。

19.2.4 HASH パーティショニング

HASH によるパーティショニングは、事前に決められた数のパーティションにデータを均等に配分するために主に使用されます。RANGE または LIST パーティショニングでは、指定されたカラム値またはカラム値セットがどのパーティションに格納されるかを明示的に指定する必要があります。HASH パーティショニングでは MySQL がこれを自動的に行うため、必要なことは、ハッシュされるカラム値またはカラム値に基づく式、およびパーティション化されたテーブルがいくつのパーティションに分割されるかを指定することだけです。

HASH パーティショニングを使用してテーブルをパーティション化する場合は、CREATE TABLE ステートメントに PARTITION BY HASH (expr) 句を付加する必要があります。ここで、expr は整数を返す式です。これには、型が MySQL の整数型のいずれかであるカラムの名前を単純に指定できます。また、これのあとにはほとんどの

場合 PARTITIONS num 句を続けます。ここで、num はテーブルがいくつのパーティションに分割されるかを表す正の整数です。

たとえば、次のステートメントは store_id カラムにハッシュを使用し、4 つのパーティションに分割されたテーブルを作成します。

```
CREATE TABLE employees (
    id INT NOT NULL,
    fname VARCHAR(30),
    lname VARCHAR(30),
    hired DATE NOT NULL DEFAULT '1970-01-01',
    separated DATE NOT NULL DEFAULT '9999-12-31',
    job_code INT,
    store_id INT
)
PARTITION BY HASH(store_id)
PARTITIONS 4;
```

PARTITIONS 句を含めない場合、パーティションの数はデフォルトで 1 となります。

PARTITIONS キーワードを使用する場合、そのあとに数を指定しないと構文エラーになります。

整数を返す SQL 式を expr に使用することもできます。たとえば、従業員が雇用された年度に基づいてパーティション化するとします。これは、次のように行うことができます。

```
CREATE TABLE employees (
    id INT NOT NULL,
    fname VARCHAR(30),
    lname VARCHAR(30),
    hired DATE NOT NULL DEFAULT '1970-01-01',
    separated DATE NOT NULL DEFAULT '9999-12-31',
    job_code INT,
    store_id INT
)
PARTITION BY HASH( YEAR(hired) )
PARTITIONS 4;
```

expr は、定数以外のランダムではない整数値 (つまり、変化するけれども決定論的であるべき) を返す必要があり、セクション19.6「パーティショニングの制約と制限」で説明されている禁止された構造体を含んでいてはいけません。また、この式は行が挿入または更新 (または場合によっては削除) されるたびに評価されるべきです。これは、非常に複雑な式がパフォーマンスの問題を起こすことがあることを意味します (特に、一度に多くの行に影響する操作 (バッチ挿入など) を実行するとき)。

もっとも効率的なハッシュ関数は、単一テーブルカラムに実行され、その値がカラム値に対して比例的に増加または減少するもので、これによってパーティションの範囲を「プルーニング」できます。つまり、式がそのベースのカラムの値に対してより密接に変化するほど、MySQL は式を HASH パーティショニングにより効率的に使用できます。

たとえば、date_col が DATE 型のカラムである場合、式 TO_DAYS(date_col) は date_col の値に正比例すると表現されます。date_col の値が変わるたびに、式の値が一定の方法で変化するためです。date_col に対する式 YEAR(date_col) の変化は、TO_DAYS(date_col) ほど比例的ではありません。date_col のあらゆる変化に対して YEAR(date_col) が同等に変化するとはかぎらないためです。それでも、YEAR(date_col) はハッシュ関数の良い候補の 1 つです。date_col の一部と正比例し、date_col の変化によって YEAR(date_col) で比例的でない変化が発生することがないためです。

比較のために、型が INT である int_col という名前のカラムがあるとします。式 POW(5-int_col,3) + 6 を検討してみてください。これは、int_col の値が変化したときに、式の値に比例的に変化することが保証されないため、ハッシュ関数の良い候選択肢ではありません。int_col の値が一定量で変化したときに、式の値の変化量が大きくなる可能性があります。たとえば、int_col が 5 から 6 に変化すると、式の値が -1 に変化しますが、int_col の値が 6 から 7 に変化すると、式の値が -7 に変化します。

つまり、カラム値と式の値のグラフが、等式 y=cx (ここで、c はゼロでない何らかの定数) によって描かれるような直線に近くなるほど、その式はハッシュにより適切になります。これは、式が非直線的であるほど、パーティションに対するデータの配分が不均衡になる傾向があることに関係しています。

理論上は、複数のカラム値を使用する式をプルーニングすることもできますが、そのような式のどれが適しているかを判断するのがかなり難しく、時間がかかることがあります。このため、複数のカラムを含むハッシュ式を使用することはあまり推奨されていません。

PARTITION BY HASH が使用された場合、MySQL はユーザー関数の結果の法に基づいて、num パーティションのうちのどのパーティションを使用するかを判断します。つまり、式 expr の場合、レコードが格納されるパー

ティションは、パーティション番号 N です (ここで、N = MOD(expr, num))。テーブル t1 が次のように 4 つのパーティションを持つように定義されているとします。

```
CREATE TABLE t1 (col1 INT, col2 CHAR(5), col3 DATE)
    PARTITION BY HASH( YEAR(col3) )
    PARTITIONS 4;
```

t1 に col3 値が '2005-09-15' であるレコードを挿入した場合、それが格納されるパーティションは次のように判断されます。

```
MOD(YEAR('2005-09-01'),4)
=  MOD(2005,4)
=  1
```

MySQL 5.6 は、パーティション化されたテーブルに挿入される新しい行の配置を判断するために、より複雑なアルゴリズムを使用する線形ハッシュと呼ばれる、HASH パーティショニングのバリアントもサポートします。このアルゴリズムについては、セクション19.2.4.1「LINEAR HASH パーティショニング」を参照してください。

ユーザー関数は、レコードが挿入または更新されるたびに評価されます。状況によっては、レコードが削除されるときにも評価されることがあります。

> **注記**
>
> パーティション化されるテーブルに UNIQUE キーがある場合、HASH ユーザー関数または KEY の column_list に引数として指定するカラムは、そのキーの一部である必要があります。

19.2.4.1 LINEAR HASH パーティショニング

MySQL は線形ハッシュもサポートしています。通常のハッシュと異なるところは、線形ハッシュは線形二乗アルゴリズムを使用し、通常のハッシュはハッシュ関数の値の法を使用することです。

構文的には、リニアハッシュパーティショニングと通常のハッシュの唯一の違いは、次に示すように、PARTITION BY 句に LINEAR キーワードが追加されていることです。

```
CREATE TABLE employees (
    id INT NOT NULL,
    fname VARCHAR(30),
    lname VARCHAR(30),
    hired DATE NOT NULL DEFAULT '1970-01-01',
    separated DATE NOT NULL DEFAULT '9999-12-31',
    job_code INT,
    store_id INT
)
PARTITION BY LINEAR HASH( YEAR(hired) )
PARTITIONS 4;
```

式 expr の場合、線形ハッシュが使用されるときにレコードが格納されるパーティションは、num パーティションのうちのパーティション番号 N です。ここで、N は次のアルゴリズムに従って導出されます。

1. num よりも大きい次の 2 の累乗を見つけます。この値を V と呼ぶことにします。これは次のように計算できます。

   ```
   V = POWER(2, CEILING(LOG(2, num)))
   ```

   (num が 13 であるとします。その場合、LOG(2,13) は 3.7004397181411 です。CEILING(3.7004397181411) は 4、V = POWER(2,4) は 16 です。)

2. N = F(column_list) & (V - 1) を設定します。

3. N >= num の間:

   - V = CEIL(V / 2) を設定します
   - N = N & (V - 1) を設定します

線形ハッシュパーティショニングを使用し、6 個のパーティションを持つテーブル t1 を次のステートメントを使用して作成するとします。

```
CREATE TABLE t1 (col1 INT, col2 CHAR(5), col3 DATE)
    PARTITION BY LINEAR HASH( YEAR(col3) )
    PARTITIONS 6;
```

col3 カラムの値が '2003-04-14' および '1998-10-19' である 2 つのレコードを t1 に挿入するとします。これらの 1 番目のパーティション番号は次のように決定されます。

```
V = POWER(2, CEILING( LOG(2,6) )) = 8
N = YEAR('2003-04-14') & (8 - 1)
  = 2003 & 7
  = 3

(3 >= 6 is FALSE: record stored in partition #3)
```

2 番目のレコードが格納されるパーティションの番号は、次のように計算されます。

```
V = 8
N = YEAR('1998-10-19') & (8-1)
  = 1998 & 7
  = 6

(6 >= 6 is TRUE: additional step required)

N = 6 & CEILING(8 / 2)
  = 6 & 3
  = 2

(2 >= 6 is FALSE: record stored in partition #2)
```

線形ハッシュによるパーティショニングの利点は、パーティションの追加、削除、マージ、および分割の速度が向上することです。これは、非常に大量 (テラバイト) のデータが含まれるテーブルを扱うときに利点になることがあります。欠点は、通常のハッシュパーティショニングを使用して獲得される配分と比べて、データがパーティションに均等に配分される可能性が低いことです。

19.2.5 KEY パーティショニング

キーによるパーティショニングはハッシュによるパーティショニングと似ていますが、ハッシュパーティショニングはユーザー定義の式を使用し、キーパーティショニング用のハッシュ関数は MySQL サーバーによって提供されます。MySQL Cluster はこのために MD5() を使用します。ほかのストレージエンジンを使用するテーブルの場合、サーバーは PASSWORD() と同じアルゴリズムに基づく独自の内部ハッシュ関数を使用します。

CREATE TABLE ... PARTITION BY KEY の構文規則は、ハッシュによってパーティション化されたテーブルを作成する場合のものと似ています。主な違いを次に示します。

- HASH ではなく KEY が使用されます。
- KEY は、0 個以上のカラム名のリストのみを取ります。パーティショニングキーとして使用されるカラムは、テーブルの主キーの一部またはすべてを構成している必要があります (テーブルにそれがある場合)。パーティショニングキーとしてカラム名を指定しない場合は、テーブルの主キーが使用されます (ある場合)。たとえば、次の CREATE TABLE ステートメントは MySQL 5.6 で有効です。

  ```
  CREATE TABLE k1 (
      id INT NOT NULL PRIMARY KEY,
      name VARCHAR(20)
  )
  PARTITION BY KEY()
  PARTITIONS 2;
  ```

  主キーはないけれども一意キーはある場合は、パーティショニングキーに一意キーが使用されます。

  ```
  CREATE TABLE k1 (
      id INT NOT NULL,
      name VARCHAR(20),
      UNIQUE KEY (id)
  )
  PARTITION BY KEY()
  PARTITIONS 2;
  ```

  ただし、一意キーカラムが NOT NULL として定義されていない場合、前のステートメントは失敗します。

  どちらの場合も、パーティショニングキーは id カラムです。ただし、SHOW CREATE TABLE の出力や INFORMATION_SCHEMA.PARTITIONS テーブルの PARTITION_EXPRESSION カラムには表示されません。

  ほかのパーティショニングタイプの場合と異なり、KEY によるパーティショニングに使用されるカラムは、整数または NULL 値に制限されません。たとえば、次の CREATE TABLE ステートメントは有効です。

  ```
  CREATE TABLE tm1 (
      s1 CHAR(32) PRIMARY KEY
  ```

```
)
PARTITION BY KEY(s1)
PARTITIONS 10;
```

ほかのパーティショニングタイプが指定された場合、前のステートメントは有効でなくなります(この場合、s1 はテーブルの主キーであるため、単純に PARTITION BY KEY() を使用することも有効であり、PARTITION BY KEY(s1) と同じ効果があります)。

この問題の詳細については、セクション19.6「パーティショニングの制約と制限」を参照してください。

> **注記**
>
> NDB ストレージエンジンを使用するテーブルは、テーブルの主キーをパーティショニングキーとして使用して、KEY によって暗黙的にパーティション化されます。MySQL Cluster テーブルに明示的な主キーがない場合は、NDB ストレージエンジンによって各 MySQL Cluster テーブルに生成される「隠し」主キーが、パーティショニングキーとして使用されます。
>
> NDB テーブルに明示的なパーティショニングスキームを定義する場合は、テーブルに明示的な主キーが必要であり、パーティショニング式に使用されるカラムがこのキーの一部である必要があります。ただし、テーブルが「空」のパーティショニング式を使用する (つまり、カラム参照なしの PARTITION BY KEY()) 場合、明示的な主キーは必要ありません。
>
> このパーティショニングは、ndb_desc ユーティリティー (-p オプション付き) を使用して確認できます。

> **重要**
>
> キーによってパーティション化されたテーブルの場合は、ALTER TABLE DROP PRIMARY KEY を実行できません。それを実行すると次のエラーが生成されます: ERROR 1466 (HY000): Field in list of fields for partition function not found in table。これは、KEY によってパーティション化された MySQL Cluster テーブルの場合は問題ではありません。そのような場合は、「隠し」主キーをテーブルの新しいパーティショニングキーとして使用してテーブルが再編成されます。第18章「MySQL Cluster NDB 7.3 および MySQL Cluster NDB 7.4」を参照してください。

リニアキーによってテーブルをパーティション化することもできます。次に、単純な例を示します。

```
CREATE TABLE tk (
    col1 INT NOT NULL,
    col2 CHAR(5),
    col3 DATE
)
PARTITION BY LINEAR KEY (col1)
PARTITIONS 3;
```

LINEAR を使用することは、HASH パーティショニングの場合と同じ効果が KEY パーティショニングにあり、パーティショニング番号は法演算ではなく二乗アルゴリズムを使用して導出されます。このアルゴリズムの説明およびその影響については、セクション19.2.4.1「LINEAR HASH パーティショニング」を参照してください。

19.2.6 サブパーティショニング

サブパーティショニング (複合パーティショニングとも呼ばれます) は、パーティション化されたテーブルの各パーティションをさらに分割することです。次の CREATE TABLE ステートメントを検討します。

```
CREATE TABLE ts (id INT, purchased DATE)
    PARTITION BY RANGE( YEAR(purchased) )
    SUBPARTITION BY HASH( TO_DAYS(purchased) )
    SUBPARTITIONS 2 (
        PARTITION p0 VALUES LESS THAN (1990),
        PARTITION p1 VALUES LESS THAN (2000),
        PARTITION p2 VALUES LESS THAN MAXVALUE
    );
```

テーブル ts には 3 つの RANGE パーティションがあります。これらの各パーティション (p0、p1、および p2) は、さらに 2 つのサブパーティションに分割されます。実際には、テーブル全体が 3 * 2 = 6 パーティションに分割されます。ただし、PARTITION BY RANGE 句のアクションによって、これらの最初の 2 つには purchased カラムで値が 1990 より小さいレコードのみが格納されます。

MySQL 5.6 では、RANGE または LIST によってパーティション化されたテーブルをサブパーティション化できます。サブパーティショニングには、HASH または KEY パーティショニングを使用できます。これは、複合パーティショニングとも呼ばれます。

> **注記**
>
> SUBPARTITION BY HASH および SUBPARTITION BY KEY は通常それぞれ、PARTITION BY HASH および PARTITION BY KEY と同じ構文規則に従います。これの例外は、SUBPARTITION BY KEY は現在 (PARTITION BY KEY と異なり) デフォルトカラムをサポートしないことで、この目的に使用されるカラムを指定する必要があります (テーブルに明示的な主キーがある場合でも)。これは既知の問題であり、対処中です。詳細および例については サブパーティションに関する問題を参照してください。

SUBPARTITION 句を使用して個々のサブパーティションのオプションを指定することによって、サブパーティションを明示的に定義することもできます。たとえば、前の例と同じテーブル ts をより冗長な形式で作成するには、次のようにします。

```
CREATE TABLE ts (id INT, purchased DATE)
    PARTITION BY RANGE( YEAR(purchased) )
    SUBPARTITION BY HASH( TO_DAYS(purchased) ) (
        PARTITION p0 VALUES LESS THAN (1990) (
            SUBPARTITION s0,
            SUBPARTITION s1
        ),
        PARTITION p1 VALUES LESS THAN (2000) (
            SUBPARTITION s2,
            SUBPARTITION s3
        ),
        PARTITION p2 VALUES LESS THAN MAXVALUE (
            SUBPARTITION s4,
            SUBPARTITION s5
        )
    );
```

構文に関するいくつかの注意事項を次に示します。

- 各パーティションには、同じ数のサブパーティションが必要です。
- パーティション化されたテーブルのパーティションに SUBPARTITION を使用してサブパーティションを明示的に定義する場合は、それらのすべてを定義する必要があります。言い換えると、次のステートメントは失敗します。

  ```
  CREATE TABLE ts (id INT, purchased DATE)
      PARTITION BY RANGE( YEAR(purchased) )
      SUBPARTITION BY HASH( TO_DAYS(purchased) ) (
          PARTITION p0 VALUES LESS THAN (1990) (
              SUBPARTITION s0,
              SUBPARTITION s1
          ),
          PARTITION p1 VALUES LESS THAN (2000),
          PARTITION p2 VALUES LESS THAN MAXVALUE (
              SUBPARTITION s2,
              SUBPARTITION s3
          )
      );
  ```

  このステートメントは、SUBPARTITIONS 2 句を含んでいた場合でも失敗します。

- 各 SUBPARTITION 句には、(少なくとも) サブパーティションの名前が含まれている必要があります。それ以外は、サブパーティションに適切なオプションを設定するか、またはそのオプションのデフォルト設定を想定します。
- サブパーティション名はテーブル全体で一意である必要があります。たとえば、次の CREATE TABLE ステートメントは MySQL 5.6 で有効です。

  ```
  CREATE TABLE ts (id INT, purchased DATE)
      PARTITION BY RANGE( YEAR(purchased) )
      SUBPARTITION BY HASH( TO_DAYS(purchased) ) (
          PARTITION p0 VALUES LESS THAN (1990) (
              SUBPARTITION s0,
              SUBPARTITION s1
          ),
          PARTITION p1 VALUES LESS THAN (2000) (
              SUBPARTITION s2,
  ```

```
        SUBPARTITION s3
    ),
    PARTITION p2 VALUES LESS THAN MAXVALUE (
        SUBPARTITION s4,
        SUBPARTITION s5
    )
);
```

サブパーティションは、非常に大きいテーブルで、データおよびインデックスを多数のディスクに分散するために使用できます。/disk0、/disk1、/disk2 などとしてマウントされた 6 個のディスクがあるとします。ここで次の例を検討します。

```
CREATE TABLE ts (id INT, purchased DATE)
  PARTITION BY RANGE( YEAR(purchased) )
  SUBPARTITION BY HASH( TO_DAYS(purchased) ) (
    PARTITION p0 VALUES LESS THAN (1990) (
      SUBPARTITION s0
        DATA DIRECTORY = '/disk0/data'
        INDEX DIRECTORY = '/disk0/idx',
      SUBPARTITION s1
        DATA DIRECTORY = '/disk1/data'
        INDEX DIRECTORY = '/disk1/idx'
    ),
    PARTITION p1 VALUES LESS THAN (2000) (
      SUBPARTITION s2
        DATA DIRECTORY = '/disk2/data'
        INDEX DIRECTORY = '/disk2/idx',
      SUBPARTITION s3
        DATA DIRECTORY = '/disk3/data'
        INDEX DIRECTORY = '/disk3/idx'
    ),
    PARTITION p2 VALUES LESS THAN MAXVALUE (
      SUBPARTITION s4
        DATA DIRECTORY = '/disk4/data'
        INDEX DIRECTORY = '/disk4/idx',
      SUBPARTITION s5
        DATA DIRECTORY = '/disk5/data'
        INDEX DIRECTORY = '/disk5/idx'
    )
  );
```

この場合、各 RANGE のデータおよびインデックスに個別のディスクが使用されます。ほかにも多数のバリエーションが考えられます。別の例を次に示します。

```
CREATE TABLE ts (id INT, purchased DATE)
  PARTITION BY RANGE(YEAR(purchased))
  SUBPARTITION BY HASH( TO_DAYS(purchased) ) (
    PARTITION p0 VALUES LESS THAN (1990) (
      SUBPARTITION s0a
        DATA DIRECTORY = '/disk0'
        INDEX DIRECTORY = '/disk1',
      SUBPARTITION s0b
        DATA DIRECTORY = '/disk2'
        INDEX DIRECTORY = '/disk3'
    ),
    PARTITION p1 VALUES LESS THAN (2000) (
      SUBPARTITION s1a
        DATA DIRECTORY = '/disk4/data'
        INDEX DIRECTORY = '/disk4/idx',
      SUBPARTITION s1b
        DATA DIRECTORY = '/disk5/data'
        INDEX DIRECTORY = '/disk5/idx'
    ),
    PARTITION p2 VALUES LESS THAN MAXVALUE (
      SUBPARTITION s2a,
      SUBPARTITION s2b
    )
  );
```

この場合は、次のように格納されます。

- purchased 日付が 1990 年より前の行には大きな領域が使用されるため、4 つに分割され、パーティション p0 を構成する 2 つの各サブパーティション (s0a および s0b) のデータおよびインデックスに個別のディスクが専用に割り当てられます。言い換えると、次のようになります。
  - サブパーティション s0a のデータは /disk0 に格納されます。
  - サブパーティション s0a のインデックスは、/disk1 に格納されます。

  - サブパーティション s0b のデータは、/disk2 に格納されます。
  - サブパーティション s0b のインデックスは、/disk3 に格納されます。
- 1990 年から 1999 年までの日付 (パーティション p1) が含まれている行は、1990 年より前の行ほどの領域を必要としません。これらは、p0 に格納されるレガシーレコードの場合の 4 つのディスクではなく、2 つのディスク (/disk4 および /disk5) に分割されます。
  - p1 の最初のサブパーティション (s1a) に属するデータおよびインデックスは、/disk4 (データは /disk4/data、およびインデックスは /disk4/idx) に格納されます。
  - p1 の 2 番目のサブパーティション (s1b) に属するデータおよびインデックスは、/disk5 (データは /disk5/data、およびインデックスは /disk5/idx) に格納されます。
- 2000 年から現在までの日付 (パーティション p2) を反映する行には、前の 2 つの範囲で必要とされたほどの領域は使用されません。現在のところ、これらのすべてをデフォルトの場所に格納しても問題ありません。

  将来、2000 年から始まる 10 年間の購入数が、デフォルトの場所では十分な領域を提供できないほど増えたときには、ALTER TABLE ... REORGANIZE PARTITION ステートメントを使用して対応する行を移動できます。これを行う方法については、セクション19.3「パーティション管理」を参照してください。

NO_DIR_IN_CREATE サーバー SQL モードが有効である場合、DATA DIRECTORY および INDEX DIRECTORY オプションはパーティション定義で許可されません。MySQL 5.6 では、これらのオプションはサブパーティションを定義するときにも許可されません (Bug #42954)。

19.2.7 MySQL パーティショニングによる NULL の扱い

MySQL のパーティショニングには、パーティショニング式の値 (カラム値またはユーザー定義式の値にかかわらず) として NULL を拒否する手段はありません。式の値として NULL を使用することは許可されていますが (そうでない場合は整数を返す必要がある)、NULL は数値でないことを認識することは重要です。MySQL のパーティショニング実装は、ORDER BY のように、NULL でない値より小さい値として NULL を扱います。

これは、NULL の扱いは各タイプのパーティショニングで異なり、これに準備していない場合は予期しない動作になる可能性があることを意味します。このような状況があるので、このセクションでは、各 MySQL パーティショニングタイプが、行をどのパーティションに格納するべきかを判断するときに NULL 値をどのように扱うかを説明し、それぞれの例を示します。

RANGE パーティショニングでの NULL の扱い　パーティションを判断するために使用されるカラム値が NULL である行を、RANGE によってパーティション化されたテーブルに挿入した場合、行はもっとも低いパーティションに挿入されます。p という名前のデータベースに、次のように作成された 2 つのテーブルがあるとします。

```
mysql> CREATE TABLE t1 (
    ->     c1 INT,
    ->     c2 VARCHAR(20)
    -> )
    -> PARTITION BY RANGE(c1) (
    ->     PARTITION p0 VALUES LESS THAN (0),
    ->     PARTITION p1 VALUES LESS THAN (10),
    ->     PARTITION p2 VALUES LESS THAN MAXVALUE
    -> );
Query OK, 0 rows affected (0.09 sec)

mysql> CREATE TABLE t2 (
    ->     c1 INT,
    ->     c2 VARCHAR(20)
    -> )
    -> PARTITION BY RANGE(c1) (
    ->     PARTITION p0 VALUES LESS THAN (-5),
    ->     PARTITION p1 VALUES LESS THAN (0),
    ->     PARTITION p2 VALUES LESS THAN (10),
    ->     PARTITION p3 VALUES LESS THAN MAXVALUE
    -> );
Query OK, 0 rows affected (0.09 sec)
```

これらの 2 つの CREATE TABLE ステートメントによって作成されたパーティションについては、次のクエリーを INFORMATION_SCHEMA データベース内の PARTITIONS テーブルに対して使用することで確認できます。

```
mysql> SELECT TABLE_NAME, PARTITION_NAME, TABLE_ROWS, AVG_ROW_LENGTH, DATA_LENGTH
    >   FROM INFORMATION_SCHEMA.PARTITIONS
    >   WHERE TABLE_SCHEMA = 'p' AND TABLE_NAME LIKE 't_';
```

```
+------------+----------------+------------+----------------+-------------+
| TABLE_NAME | PARTITION_NAME | TABLE_ROWS | AVG_ROW_LENGTH | DATA_LENGTH |
+------------+----------------+------------+----------------+-------------+
| t1         | p0             |          0 |              0 |           0 |
| t1         | p1             |          0 |              0 |           0 |
| t1         | p2             |          0 |              0 |           0 |
| t2         | p0             |          0 |              0 |           0 |
| t2         | p1             |          0 |              0 |           0 |
| t2         | p2             |          0 |              0 |           0 |
| t2         | p3             |          0 |              0 |           0 |
+------------+----------------+------------+----------------+-------------+
7 rows in set (0.00 sec)
```

(このテーブルについての詳細は、セクション21.13「INFORMATION_SCHEMA PARTITIONS テーブル」を参照してください。)ここで、これらの各テーブルのパーティショニングキーとして使用されるカラムに NULL が含まれる単一行を移入し、2 つの SELECT ステートメントを使用してこれらの行が挿入されたことを確認します。

```
mysql> INSERT INTO t1 VALUES (NULL, 'mothra');
Query OK, 1 row affected (0.00 sec)

mysql> INSERT INTO t2 VALUES (NULL, 'mothra');
Query OK, 1 row affected (0.00 sec)

mysql> SELECT * FROM t1;
+------+--------+
| id   | name   |
+------+--------+
| NULL | mothra |
+------+--------+
1 row in set (0.00 sec)

mysql> SELECT * FROM t2;
+------+--------+
| id   | name   |
+------+--------+
| NULL | mothra |
+------+--------+
1 row in set (0.00 sec)
```

挿入された行を格納するためにどのパーティションが使用されたかについては、前のクエリーを INFORMATION_SCHEMA.PARTITIONS に対して再実行して出力を検査することで確認できます。

```
mysql> SELECT TABLE_NAME, PARTITION_NAME, TABLE_ROWS, AVG_ROW_LENGTH, DATA_LENGTH
    >   FROM INFORMATION_SCHEMA.PARTITIONS
    >   WHERE TABLE_SCHEMA = 'p' AND TABLE_NAME LIKE 't_';
+------------+----------------+------------+----------------+-------------+
| TABLE_NAME | PARTITION_NAME | TABLE_ROWS | AVG_ROW_LENGTH | DATA_LENGTH |
+------------+----------------+------------+----------------+-------------+
| t1         | p0             |          1 |             20 |          20 |
| t1         | p1             |          0 |              0 |           0 |
| t1         | p2             |          0 |              0 |           0 |
| t2         | p0             |          1 |             20 |          20 |
| t2         | p1             |          0 |              0 |           0 |
| t2         | p2             |          0 |              0 |           0 |
| t2         | p3             |          0 |              0 |           0 |
+------------+----------------+------------+----------------+-------------+
7 rows in set (0.01 sec)
```

これらの行が各テーブルのもっとも低いパーティションに格納されたことについては、これらのパーティションを削除してから SELECT ステートメントを再実行することで確認できます。

```
mysql> ALTER TABLE t1 DROP PARTITION p0;
Query OK, 0 rows affected (0.16 sec)

mysql> ALTER TABLE t2 DROP PARTITION p0;
Query OK, 0 rows affected (0.16 sec)

mysql> SELECT * FROM t1;
Empty set (0.00 sec)

mysql> SELECT * FROM t2;
Empty set (0.00 sec)
```

(ALTER TABLE ... DROP PARTITION の詳細については、セクション13.1.7「ALTER TABLE 構文」を参照してください。)

SQL 関数を使用するパーティショニング式の場合も、NULL はこのように扱われます。次のような CREATE TABLE ステートメントを使用してテーブルを定義するとします。

```
CREATE TABLE tndate (
    id INT,
    dt DATE
)
PARTITION BY RANGE( YEAR(dt) ) (
    PARTITION p0 VALUES LESS THAN (1990),
    PARTITION p1 VALUES LESS THAN (2000),
    PARTITION p2 VALUES LESS THAN MAXVALUE
);
```

ほかの MySQL 関数と同様に、YEAR(NULL) は NULL を返します。dt カラム値が NULL である行は、パーティショニング式がほかの値より小さい値に評価されたかのように扱われ、パーティション p0 に挿入されます。

LIST パーティショニングでの NULL の扱い　LIST によってパーティション化されたテーブルで NULL 値が許可されるのは、NULL が含まれている値リストを使用していずれかのパーティションが定義されている場合のみです。これとは逆に、LIST によってパーティション化されたテーブルが、値リストで NULL を明示的に使用していない場合は、次の例のようにパーティショニング式で NULL 値に評価される行を拒否します。

```
mysql> CREATE TABLE ts1 (
    ->     c1 INT,
    ->     c2 VARCHAR(20)
    -> )
    -> PARTITION BY LIST(c1) (
    ->     PARTITION p0 VALUES IN (0, 3, 6),
    ->     PARTITION p1 VALUES IN (1, 4, 7),
    ->     PARTITION p2 VALUES IN (2, 5, 8)
    -> );
Query OK, 0 rows affected (0.01 sec)

mysql> INSERT INTO ts1 VALUES (9, 'mothra');
ERROR 1504 (HY000): Table has no partition for value 9

mysql> INSERT INTO ts1 VALUES (NULL, 'mothra');
ERROR 1504 (HY000): Table has no partition for value NULL
```

ts1 に挿入できるのは、c1 値が 0 以上 8 以下の行のみです。NULL は、数値 9 と同様にこの範囲を外れます。NULL が含まれる値リストを持つテーブル ts2 およびts3 は次のように作成できます。

```
mysql> CREATE TABLE ts2 (
    ->     c1 INT,
    ->     c2 VARCHAR(20)
    -> )
    -> PARTITION BY LIST(c1) (
    ->     PARTITION p0 VALUES IN (0, 3, 6),
    ->     PARTITION p1 VALUES IN (1, 4, 7),
    ->     PARTITION p2 VALUES IN (2, 5, 8),
    ->     PARTITION p3 VALUES IN (NULL)
    -> );
Query OK, 0 rows affected (0.01 sec)

mysql> CREATE TABLE ts3 (
    ->     c1 INT,
    ->     c2 VARCHAR(20)
    -> )
    -> PARTITION BY LIST(c1) (
    ->     PARTITION p0 VALUES IN (0, 3, 6),
    ->     PARTITION p1 VALUES IN (1, 4, 7, NULL),
    ->     PARTITION p2 VALUES IN (2, 5, 8)
    -> );
Query OK, 0 rows affected (0.01 sec)
```

パーティショニングの値リストを定義するときに、NULL をほかの値と同様に扱うことができます (そうすべきです)。たとえば、VALUES IN (NULL) および VALUES IN (1, 4, 7, NULL) は両方とも有効であり、VALUES IN (1, NULL, 4, 7)、VALUES IN (NULL, 1, 4, 7) なども同様です。カラム c1 が NULL である行をテーブル ts2 および ts3 にそれぞれ挿入できます。

```
mysql> INSERT INTO ts2 VALUES (NULL, 'mothra');
Query OK, 1 row affected (0.00 sec)

mysql> INSERT INTO ts3 VALUES (NULL, 'mothra');
Query OK, 1 row affected (0.00 sec)
```

INFORMATION_SCHEMA.PARTITIONS に対して適切なクエリーを発行することによって、先ほど挿入した行を格納するためにどのパーティションが使用されたかを確認できます (前の例と同様に、パーティション化されたテーブルが p データベースに作成されたことを想定しています)。

```
mysql> SELECT TABLE_NAME, PARTITION_NAME, TABLE_ROWS, AVG_ROW_LENGTH, DATA_LENGTH
```

```
    >   FROM INFORMATION_SCHEMA.PARTITIONS
    >   WHERE TABLE_SCHEMA = 'p' AND TABLE_NAME LIKE 'ts_';
+------------+----------------+------------+----------------+-------------+
| TABLE_NAME | PARTITION_NAME | TABLE_ROWS | AVG_ROW_LENGTH | DATA_LENGTH |
+------------+----------------+------------+----------------+-------------+
| ts2        | p0             |          0 |              0 |           0 |
| ts2        | p1             |          0 |              0 |           0 |
| ts2        | p2             |          0 |              0 |           0 |
| ts2        | p3             |          1 |             20 |          20 |
| ts3        | p0             |          0 |              0 |           0 |
| ts3        | p1             |          1 |             20 |          20 |
| ts3        | p2             |          0 |              0 |           0 |
+------------+----------------+------------+----------------+-------------+
7 rows in set (0.01 sec)
```

このセクションですでに示したように、行を格納するためにどのパーティションが使用されたかについては、それらのパーティションを削除してから SELECT を実行することで確認できます。

HASH および KEY パーティショニングでの NULL の扱い　HASH または KEY によってパーティション化されたテーブルの場合、NULL の扱いは少し異なります。これらの場合、NULL 値を返すパーティショニング式は、戻り値がゼロであったかのように扱われます。この動作については、HASH によってパーティション化されたテーブルを作成して該当する値が含まれるレコードを挿入することで、ファイルシステムにどのような影響があるかを検査することで確認できます。次のステートメントを使用して作成されたテーブル th (これも p データベース内) があるとします。

```
mysql> CREATE TABLE th (
    ->     c1 INT,
    ->     c2 VARCHAR(20)
    -> )
    -> PARTITION BY HASH(c1)
    -> PARTITIONS 2;
Query OK, 0 rows affected (0.00 sec)
```

このテーブルに属するパーティションは、次のクエリーを使用して表示できます。

```
mysql> SELECT TABLE_NAME,PARTITION_NAME,TABLE_ROWS,AVG_ROW_LENGTH,DATA_LENGTH
    >   FROM INFORMATION_SCHEMA.PARTITIONS
    >   WHERE TABLE_SCHEMA = 'p' AND TABLE_NAME ='th';
+------------+----------------+------------+----------------+-------------+
| TABLE_NAME | PARTITION_NAME | TABLE_ROWS | AVG_ROW_LENGTH | DATA_LENGTH |
+------------+----------------+------------+----------------+-------------+
| th         | p0             |          0 |              0 |           0 |
| th         | p1             |          0 |              0 |           0 |
+------------+----------------+------------+----------------+-------------+
2 rows in set (0.00 sec)
```

各パーティションの TABLE_ROWS は 0 です。ここで次に示すように、c1 カラム値が NULL および 0 である 2 つの行を th に挿入し、それらの行が挿入されたことを確認します。

```
mysql> INSERT INTO th VALUES (NULL, 'mothra'), (0, 'gigan');
Query OK, 1 row affected (0.00 sec)

mysql> SELECT * FROM th;
+------+---------+
| c1   | c2      |
+------+---------+
| NULL | mothra  |
+------+---------+
|    0 | gigan   |
+------+---------+
2 rows in set (0.01 sec)
```

任意の整数 N について、NULL MOD N の値は常に NULL であることを思い出してください。HASH または KEY によってパーティション化されたテーブルの場合、この結果は正しいパーティションを判別するために 0 として扱われます。INFORMATION_SCHEMA.PARTITIONS テーブルを再度確認すると、両方の行がパーティション p0 に挿入されたことがわかります。

```
mysql> SELECT TABLE_NAME, PARTITION_NAME, TABLE_ROWS, AVG_ROW_LENGTH, DATA_LENGTH
    >   FROM INFORMATION_SCHEMA.PARTITIONS
    >   WHERE TABLE_SCHEMA = 'p' AND TABLE_NAME ='th';
+------------+----------------+------------+----------------+-------------+
| TABLE_NAME | PARTITION_NAME | TABLE_ROWS | AVG_ROW_LENGTH | DATA_LENGTH |
+------------+----------------+------------+----------------+-------------+
| th         | p0             |          2 |             20 |          20 |
| th         | p1             |          0 |              0 |           0 |
+------------+----------------+------------+----------------+-------------+
2 rows in set (0.00 sec)
```

テーブルの定義で PARTITION BY HASH の代わりに PARTITION BY KEY を使用してこの例を繰り返すと、このタイプのパーティショニングでも NULL が 0 のように扱われることを簡単に確認できます。

19.3 パーティション管理

MySQL 5.6 は、パーティション化されたテーブルを変更するためにいくつかの方法を提供しています。既存のパーティションを追加、削除、再定義、マージ、または分割できます。これらのすべてのアクションは、ALTER TABLE ステートメントのパーティショニング拡張を使用して実行できます。パーティション化されたテーブルおよびパーティションに関する情報を取得する方法もあります。以降のセクションでは次のトピックについて説明します。

- RANGE または LIST によってパーティション化されたテーブルのパーティション管理については、セクション19.3.1「RANGE および LIST パーティションの管理」を参照してください。
- HASH および KEY パーティションの管理については、セクション19.3.2「HASH および KEY パーティションの管理」を参照してください。
- パーティション化されたテーブルおよびパーティションに関する情報を取得するために MySQL 5.6 で提供されるメカニズムについては、セクション19.3.5「パーティションに関する情報を取得する」を参照してください。
- パーティションの保守操作の実行については、セクション19.3.4「パーティションの保守」を参照してください。

> **注記**
>
> MySQL 5.6 では、パーティション化されたテーブルのすべてのパーティションに同じ数のサブパーティションが必要であり、テーブルが作成されたあとにサブパーティショニングを変更することはできません。

テーブルのパーティショニングスキームを変更するために必要なことは、partition_options 句付きで ALTER TABLE ステートメントを使用することだけです。この句の構文は、パーティション化されたテーブルを作成する CREATE TABLE で使用されるものと同じで、必ずキーワード PARTITION BY で始まります。次の CREATE TABLE ステートメントを使用して範囲によってパーティション化されたテーブルがあるとします。

```
CREATE TABLE trb3 (id INT, name VARCHAR(50), purchased DATE)
    PARTITION BY RANGE( YEAR(purchased) ) (
        PARTITION p0 VALUES LESS THAN (1990),
        PARTITION p1 VALUES LESS THAN (1995),
        PARTITION p2 VALUES LESS THAN (2000),
        PARTITION p3 VALUES LESS THAN (2005)
    );
```

このテーブルをキーによるパーティション化でパーティション化し直して、キーをベースとする id カラム値を使用する 2 つのパーティションに分割するために、次のステートメントを使用できます。

```
ALTER TABLE trb3 PARTITION BY KEY(id) PARTITIONS 2;
```

これは、テーブルを削除してから CREATE TABLE trb3 PARTITION BY KEY(id) PARTITIONS 2; を使用して再作成する場合と同じ効果を、テーブルの構造に対して持ちます。

ALTER TABLE ... ENGINE = ... は、テーブルによって使用されるストレージエンジンのみを変更し、テーブルのパーティショニングスキームはそのままにします。テーブルのパーティショニングを削除するには、ALTER TABLE ... REMOVE PARTITIONING を使用します。セクション13.1.7「ALTER TABLE 構文」を参照してください。

> **重要**
>
> ALTER TABLE ステートメントに使用できるのは、単一の PARTITION BY、ADD PARTITION、DROP PARTITION、REORGANIZE PARTITION、または COALESCE PARTITION 句のみです。たとえば、あるパーティションを削除して、テーブルの残りのパーティションを再編成する場合は、2 つの別々の ALTER TABLE ステートメントでそうする必要があります (DROP PARTITION を使用するものと REORGANIZE PARTITIONS を使用するもの)。

MySQL 5.6 では、ALTER TABLE ... TRUNCATE PARTITION を使用して、選択した 1 つ以上のパーティションからすべての行を削除できます。

19.3.1 RANGE および LIST パーティションの管理

RANGE および LIST パーティションは、パーティションの追加と削除がどのように処理されるかに関してはよく似ています。このため、このセクションでは両方のタイプのパーティショニングの管理について説明します。ハッシュまたはキーによってパーティション化されたテーブルの管理については、セクション19.3.2「HASH および KEY パーティションの管理」を参照してください。RANGE または LIST パーティションの削除は追加よりも単純なので、これを最初に説明します。

RANGE または LIST によってパーティション化されたテーブルからパーティションを削除する操作は、DROP PARTITION 句付きで ALTER TABLE ステートメントを使用することで実行できます。非常に基本的な例ですが、次の CREATE TABLE および INSERT ステートメントを使用して、範囲によってパーティション化されるテーブルをすでに作成済みで、10 件のレコードを移入しているとします。

```
mysql> CREATE TABLE tr (id INT, name VARCHAR(50), purchased DATE)
    ->     PARTITION BY RANGE( YEAR(purchased) ) (
    ->         PARTITION p0 VALUES LESS THAN (1990),
    ->         PARTITION p1 VALUES LESS THAN (1995),
    ->         PARTITION p2 VALUES LESS THAN (2000),
    ->         PARTITION p3 VALUES LESS THAN (2005)
    ->     );
Query OK, 0 rows affected (0.01 sec)

mysql> INSERT INTO tr VALUES
    ->     (1, 'desk organiser', '2003-10-15'),
    ->     (2, 'CD player', '1993-11-05'),
    ->     (3, 'TV set', '1996-03-10'),
    ->     (4, 'bookcase', '1982-01-10'),
    ->     (5, 'exercise bike', '2004-05-09'),
    ->     (6, 'sofa', '1987-06-05'),
    ->     (7, 'popcorn maker', '2001-11-22'),
    ->     (8, 'aquarium', '1992-08-04'),
    ->     (9, 'study desk', '1984-09-16'),
    ->     (10, 'lava lamp', '1998-12-25');
Query OK, 10 rows affected (0.01 sec)
```

パーティション p2 に挿入されているはずの項目を以下のように確認できます。

```
mysql> SELECT * FROM tr
    -> WHERE purchased BETWEEN '1995-01-01' AND '1999-12-31';
+------+-----------+------------+
| id   | name      | purchased  |
+------+-----------+------------+
|    3 | TV set    | 1996-03-10 |
|   10 | lava lamp | 1998-12-25 |
+------+-----------+------------+
2 rows in set (0.00 sec)
```

p2 という名前のパーティションを削除するには、次のコマンドを実行します。

```
mysql> ALTER TABLE tr DROP PARTITION p2;
Query OK, 0 rows affected (0.03 sec)
```

> **注記**
>
> NDB ストレージエンジンは ALTER TABLE ... DROP PARTITION をサポートしません。ただし、この章で説明されている ALTER TABLE へのほかのパーティショニング関連拡張はサポートしています。

パーティションを削除すると、そのパーティションに格納されていたすべてのデータも削除されることを覚えておくことは非常に重要です。前の SELECT クエリーを再実行することで、これが本当であることを確認できます。

```
mysql> SELECT * FROM tr WHERE purchased
    -> BETWEEN '1995-01-01' AND '1999-12-31';
Empty set (0.00 sec)
```

このため、テーブルに対して ALTER TABLE ... DROP PARTITION を実行するには、そのテーブルの DROP 権限が必要です。

テーブル定義およびそのパーティショニングスキームを保持したまま、すべてのパーティションからすべてのデータを削除する場合は、TRUNCATE TABLE ステートメントを使用してください。(セクション13.1.33「TRUNCATE TABLE 構文」を参照してください)。

データを失うことなくテーブルのパーティショニングを変更する場合は、代わりに ALTER TABLE ... REORGANIZE PARTITION を使用してください。REORGANIZE PARTITION については、後続の説明またはセクション13.1.7「ALTER TABLE 構文」を参照してください。

ここで SHOW CREATE TABLE ステートメントを実行すると、テーブルのパーティショニング構成がどのように変更されたかを確認できます。

```
mysql> SHOW CREATE TABLE tr\G
*************************** 1. row ***************************
       Table: tr
Create Table: CREATE TABLE `tr` (
  `id` int(11) default NULL,
  `name` varchar(50) default NULL,
  `purchased` date default NULL
) ENGINE=MyISAM DEFAULT CHARSET=latin1
PARTITION BY RANGE ( YEAR(purchased) ) (
  PARTITION p0 VALUES LESS THAN (1990) ENGINE = MyISAM,
  PARTITION p1 VALUES LESS THAN (1995) ENGINE = MyISAM,
  PARTITION p3 VALUES LESS THAN (2005) ENGINE = MyISAM
)
1 row in set (0.01 sec)
```

purchased カラム値が '1995-01-01' から '2004-12-31' まで (両端を含む) の新しい行を変更済みテーブルに挿入すると、それらの行はパーティション p3 に格納されます。このことを次のようにして確認できます。

```
mysql> INSERT INTO tr VALUES (11, 'pencil holder', '1995-07-12');
Query OK, 1 row affected (0.00 sec)

mysql> SELECT * FROM tr WHERE purchased
    -> BETWEEN '1995-01-01' AND '2004-12-31';
+------+---------------+------------+
| id   | name          | purchased  |
+------+---------------+------------+
|   11 | pencil holder  | 1995-07-12 |
|    1 | desk organiser | 2003-10-15 |
|    5 | exercise bike  | 2004-05-09 |
|    7 | popcorn maker  | 2001-11-22 |
+------+---------------+------------+
4 rows in set (0.00 sec)

mysql> ALTER TABLE tr DROP PARTITION p3;
Query OK, 0 rows affected (0.03 sec)

mysql> SELECT * FROM tr WHERE purchased
    -> BETWEEN '1995-01-01' AND '2004-12-31';
Empty set (0.00 sec)
```

ALTER TABLE ... DROP PARTITION の結果としてテーブルから削除された行数は、同等の DELETE クエリーとは異なり、サーバーから報告されません。

LIST パーティションを削除する場合は、RANGE パーティションの削除に使用するものとまったく同じ ALTER TABLE ... DROP PARTITION 構文を使用します。ただし、この操作が持つ影響について、このテーブルをあとで使用する際に重要な違いが 1 つあります。このテーブルには、削除したパーティションを定義する値リストに含まれていた値を持つ行を挿入できなくなります。(例については、セクション19.2.2「LIST パーティショニング」を参照してください)。

すでにパーティション化されたテーブルに新しい範囲またはリストパーティションを追加するには、ALTER TABLE ... ADD PARTITION ステートメントを使用します。RANGE によってパーティション化されたテーブルの場合は、これを使用して、既存のパーティションのリストの最後に新しい範囲を追加できます。次のように定義された、組織のメンバーシップデータが含まれるパーティション化されたテーブルがあるとします。

```
CREATE TABLE members (
    id INT,
    fname VARCHAR(25),
    lname VARCHAR(25),
    dob DATE
)
PARTITION BY RANGE( YEAR(dob) ) (
    PARTITION p0 VALUES LESS THAN (1970),
    PARTITION p1 VALUES LESS THAN (1980),
    PARTITION p2 VALUES LESS THAN (1990)
);
```

さらに、メンバーの最少年齢は 16 歳であるとします。カレンダーが 2005 年の終わりに近づいて、1990 年に生まれたメンバー (さらに、来年以降はそれよりあとのメンバー) をまもなく受け入れることに気付きます。次のように members テーブルを変更することで、1990 年から 1999 年までに生まれた新しいメンバーを受け入れることができます。

```
ALTER TABLE members ADD PARTITION (PARTITION p3 VALUES LESS THAN (2000));
```

範囲によってパーティション化されたテーブルで ADD PARTITION を使用するときは、パーティションリストの上端にのみ新しいパーティションを追加できます。この方法で新しいパーティションを既存のパーティションの間または前に追加しようとすると、次のようにエラーになります。

```
mysql> ALTER TABLE members
    >     ADD PARTITION (
    >     PARTITION n VALUES LESS THAN (1960));
ERROR 1463 (HY000): VALUES LESS THAN value must be strictly »
  increasing for each partition
```

この問題は、次のように最初のパーティションを 2 つに再編成し、それらの間の範囲を分割することで回避できます。

```
ALTER TABLE members
    REORGANIZE PARTITION p0 INTO (
        PARTITION n0 VALUES LESS THAN (1960),
        PARTITION n1 VALUES LESS THAN (1970)
);
```

SHOW CREATE TABLE を使用することで、ALTER TABLE ステートメントによって意図した効果が得られたことを確認できます。

```
mysql> SHOW CREATE TABLE members\G
*************************** 1. row ***************************
       Table: members
Create Table: CREATE TABLE `members` (
  `id` int(11) DEFAULT NULL,
  `fname` varchar(25) DEFAULT NULL,
  `lname` varchar(25) DEFAULT NULL,
  `dob` date DEFAULT NULL
) ENGINE=InnoDB DEFAULT CHARSET=latin1
/*!50100 PARTITION BY RANGE ( YEAR(dob))
(PARTITION n0 VALUES LESS THAN (1960) ENGINE = InnoDB,
 PARTITION n1 VALUES LESS THAN (1970) ENGINE = InnoDB,
 PARTITION p1 VALUES LESS THAN (1980) ENGINE = InnoDB,
 PARTITION p2 VALUES LESS THAN (1990) ENGINE = InnoDB,
 PARTITION p3 VALUES LESS THAN (2000) ENGINE = InnoDB) */
1 row in set (0.00 sec)
```

セクション13.1.7.1「ALTER TABLE パーティション操作」も参照してください。

ALTER TABLE ... ADD PARTITION を使用して、LIST によってパーティション化されたテーブルに新しいパーティションを追加することもできます。次の CREATE TABLE ステートメントを使用してテーブル tt が定義されているとします。

```
CREATE TABLE tt (
    id INT,
    data INT
)
PARTITION BY LIST(data) (
    PARTITION p0 VALUES IN (5, 10, 15),
    PARTITION p1 VALUES IN (6, 12, 18)
);
```

data カラム値が 7、14、および 21 である行を格納する新しいパーティションを次のように追加できます。

```
ALTER TABLE tt ADD PARTITION (PARTITION p2 VALUES IN (7, 14, 21));
```

既存のパーティションの値リストにすでに含まれている値を含む新しい LIST パーティションは、追加できません。これを試みると エラーになります。

```
mysql> ALTER TABLE tt ADD PARTITION
    >     (PARTITION np VALUES IN (4, 8, 12));
ERROR 1465 (HY000): Multiple definition of same constant »
                in list partitioning
```

data カラム値が 12 である行がパーティション p1 にすでに割り当てられているため、値リストに 12 が含まれる新しいパーティションをテーブル tt に作成することはできません。これを実現するために、p1 を削除し、np を追加してから、定義を変更した新しい p1 を追加できます。ただし、すでに説明したように、これによって p1 に格納されていたすべてのデータが失われるので、これが実際にやりたいことでないことが多いです。別の解決策になる可能性があるのが、CREATE TABLE ... SELECT ... を使用して、新しいパーティショニング付きでテーブルのコピーを作成し、データをそこにコピーしてから、古いテーブルを削除して新しいテーブルを名前変更することですが、これは大量のデータを扱うときに非常に時間がかかる可能性があります。高可用性が要求される状況では実行できない可能性もあります。

次のように単一 ALTER TABLE ... ADD PARTITION ステートメントで複数のパーティションを追加できます。

```
CREATE TABLE employees (
  id INT NOT NULL,
  fname VARCHAR(50) NOT NULL,
  lname VARCHAR(50) NOT NULL,
  hired DATE NOT NULL
)
PARTITION BY RANGE( YEAR(hired) ) (
  PARTITION p1 VALUES LESS THAN (1991),
  PARTITION p2 VALUES LESS THAN (1996),
  PARTITION p3 VALUES LESS THAN (2001),
  PARTITION p4 VALUES LESS THAN (2005)
);

ALTER TABLE employees ADD PARTITION (
    PARTITION p5 VALUES LESS THAN (2010),
    PARTITION p6 VALUES LESS THAN MAXVALUE
);
```

ありがたいことに、MySQL のパーティショニング実装は、データを失うことなくパーティショニングを再定義する方法を提供しています。まず、RANGE パーティショニングを使用するいくつかの簡単な例を見てみましょう。次のように定義された members テーブルを思い出してください。

```
mysql> SHOW CREATE TABLE members\G
*************************** 1. row ***************************
       Table: members
Create Table: CREATE TABLE `members` (
  `id` int(11) default NULL,
  `fname` varchar(25) default NULL,
  `lname` varchar(25) default NULL,
  `dob` date default NULL
) ENGINE=MyISAM DEFAULT CHARSET=latin1
PARTITION BY RANGE ( YEAR(dob) ) (
  PARTITION p0 VALUES LESS THAN (1970) ENGINE = MyISAM,
  PARTITION p1 VALUES LESS THAN (1980) ENGINE = MyISAM,
  PARTITION p2 VALUES LESS THAN (1990) ENGINE = MyISAM.
  PARTITION p3 VALUES LESS THAN (2000) ENGINE = MyISAM
)
```

1960 年より前に生まれたメンバーを表すすべての行を別のパーティションに移動するとします。すでに説明したように、これは ALTER TABLE ... ADD PARTITION を使用して行うことはできません。ただし、ALTER TABLE への別のパーティション関連拡張を使用して、これを行うことができます。

```
ALTER TABLE members REORGANIZE PARTITION p0 INTO (
    PARTITION s0 VALUES LESS THAN (1960),
    PARTITION s1 VALUES LESS THAN (1970)
);
```

実際には、このコマンドはパーティション p0 を 2 つの新しいパーティション s0 および s1 に分割します。さらに、p0 に格納されていたデータを 2つの PARTITION ... VALUES ... 句に示されているルールに従って新しいパーティションに移動する結果、s0 には YEAR(dob) が 1960 より小さいレコードのみが含まれ、s1 には YEAR(dob) が 1960 以上で 1970 より小さい行が含まれます。

REORGANIZE PARTITION 句を使用して、隣接するパーティションをマージすることもできます。次のように members テーブルを以前のパーティショニングに戻すことができます。

```
ALTER TABLE members REORGANIZE PARTITION s0,s1 INTO (
    PARTITION p0 VALUES LESS THAN (1970)
);
```

REORGANIZE PARTITION を使用してパーティションを分割またはマージしてもデータは失われません。上記のステートメントを実行すると、MySQL はパーティション s0 および s1 に格納されていたすべてのレコードをパーティション p0 に移動します。

REORGANIZE PARTITION の一般的な構文を次に示します。

```
ALTER TABLE tbl_name
    REORGANIZE PARTITION partition_list
    INTO (partition_definitions);
```

ここで、tbl_name はパーティション化されたテーブルの名前、partition_list は変更する 1 つ以上の既存のパーティションの名前のカンマ区切りのリストです。partition_definitions は、CREATE TABLE で使用される partition_definitions リスト (セクション13.1.17「CREATE TABLE 構文」を参照してください) と同じ規則に従

う、新しいパーティション定義のカンマ区切りのリストです。REORGANIZE PARTITION を使用するときは、複数のパーティションを 1 つにマージする、または 1 つのパーティションを多数に分割する以外のこともできます。たとえば、次のように members テーブルの 4 つのパーティションすべてを 2 つに再編成できます。

```
ALTER TABLE members REORGANIZE PARTITION p0,p1,p2,p3 INTO (
    PARTITION m0 VALUES LESS THAN (1980),
    PARTITION m1 VALUES LESS THAN (2000)
);
```

LIST によってパーティション化されたテーブルで REORGANIZE PARTITION を使用することもできます。リストによってパーティション化された tt テーブルに新しいパーティションを追加する操作が、既存のパーティションのいずれかの値リストにすでに存在する値が新しいパーティションに含まれていることが原因で失敗する問題に戻ります。これは、競合しない値のみが含まれるパーティションを追加してから、新しいパーティションと既存のものを再編成して既存のものに格納されていた値が新しいものに移動するようにすることで、対処できます。

```
ALTER TABLE tt ADD PARTITION (PARTITION np VALUES IN (4, 8));
ALTER TABLE tt REORGANIZE PARTITION p1,np INTO (
    PARTITION p1 VALUES IN (6, 18),
    PARTITION np VALUES in (4, 8, 12)
);
```

RANGE または LIST によってパーティション化されたテーブルをパーティション化し直すために ALTER TABLE ... REORGANIZE PARTITION を使用するときに注意すべき、いくつかの重要点を次に示します。

- 新しいパーティショニングスキームを決定するために使用される PARTITION 句には、CREATE TABLE ステートメントで使用されるものと同じ規則が適用されます。

  もっとも重要なことは、新しいパーティショニングスキームには、重複する範囲 (RANGE によってパーティション化されたテーブルに適用される) または値のセット (LIST によってパーティション化されたテーブルを再編成するとき) を指定できないことを覚えておくべきです。

- partition_definitions リストのパーティションの組み合わせは、partition_list に指定されたパーティションの組み合わせの範囲または値セット全体と同じであるべきです。

  たとえば、このセクションの例として使用されている members テーブルでは、パーティション p1 および p2 が 1980 年から 1999 年までを範囲としています。このため、これら 2 つのパーティションを再編成する場合は、全体として同じ年範囲を含むべきです。

- RANGE によってパーティション化されたテーブルの場合は、隣接するパーティションのみを再編成できます。範囲パーティションを飛ばすことはできません。

  たとえば、このセクションの例として使用されている members テーブルを、ALTER TABLE members REORGANIZE PARTITION p0,p2 INTO ... で始まるステートメントを使用して再編成することはできません。p0 は 1970 年より前の年を範囲とし、p2 は 1990 年から 1999 年まで (両端を含む) を範囲としていて、この 2 つは隣接するパーティションではないためです。

- REORGANIZE PARTITION を使用して、テーブルのパーティショニングタイプを変更することはできません。つまり、たとえば、RANGE パーティションを HASH パーティションに変更することはできません (逆も不可)。このコマンドを使用してパーティショニング式またはカラムを変更することもできません。これらの作業をテーブルを削除および再作成することなく実現するために、ALTER TABLE ... PARTITION BY ... を使用できます。例:

  ```
  ALTER TABLE members
    PARTITION BY HASH( YEAR(dob) )
    PARTITIONS 8;
  ```

19.3.2 HASH および KEY パーティションの管理

ハッシュまたはキーによってパーティション化されたテーブルは、パーティショニングセットアップで変更に関して互いによく似ていますが、範囲またはリストによってパーティション化されたテーブルとはいくつかの点で異なります。このため、このセクションではハッシュまたはキーによってパーティション化されたテーブルの変更についてのみ取り上げます。範囲またはリストによってパーティション化されたテーブルのパーティションを追加および削除することについては、セクション19.3.1「RANGE および LIST パーティションの管理」を参照してください。

HASH または KEY によってパーティション化されたテーブルから、RANGE または LIST によってパーティション化されたテーブルと同じ方法でパーティションを削除することはできません。ただし、ALTER TABLE ... COALESCE PARTITION ステートメントを使用して HASH または KEY のパーティションをマージすることはで

きます。クライアントに関するデータが含まれている、12 個のパーティションに分割されたテーブルがあるとします。clients テーブルは次のように定義されています。

```
CREATE TABLE clients (
    id INT,
    fname VARCHAR(30),
    lname VARCHAR(30),
    signed DATE
)
PARTITION BY HASH( MONTH(signed) )
PARTITIONS 12;
```

パーティションの数を 12 個から 8 個に減らす場合は、次の ALTER TABLE コマンドを実行します。

```
mysql> ALTER TABLE clients COALESCE PARTITION 4;
Query OK, 0 rows affected (0.02 sec)
```

COALESCE は、HASH、KEY、LINEAR HASH、または LINEAR KEY によってパーティション化されたテーブルで同等に適切に動作します。次の例は前の例と似ていますが、テーブルが LINEAR KEY によってパーティション化されている点のみが異なります。

```
mysql> CREATE TABLE clients_lk (
    ->     id INT,
    ->     fname VARCHAR(30),
    ->     lname VARCHAR(30),
    ->     signed DATE
    -> )
    -> PARTITION BY LINEAR KEY(signed)
    -> PARTITIONS 12;
Query OK, 0 rows affected (0.03 sec)

mysql> ALTER TABLE clients_lk COALESCE PARTITION 4;
Query OK, 0 rows affected (0.06 sec)
Records: 0  Duplicates: 0  Warnings: 0
```

COALESCE PARTITION のあとの数値は、残りにマージするパーティションの数です。つまり、テーブルから削除するパーティションの数です。

テーブルに含まれているものより多くのパーティションを削除しようとすると、次のような工ラーになります。

```
mysql> ALTER TABLE clients COALESCE PARTITION 18;
ERROR 1478 (HY000): Cannot remove all partitions, use DROP TABLE instead
```

clients テーブルのパーティションの数を 12 個から 18 個に増やす場合は、次のように ALTER TABLE ... ADD PARTITION を使用します。

```
ALTER TABLE clients ADD PARTITION PARTITIONS 6;
```

19.3.3 パーティションとサブパーティションをテーブルと交換する

MySQL 5.6 では、ALTER TABLE pt EXCHANGE PARTITION p WITH TABLE nt を使用して、テーブルパーティションまたはサブパーティションをテーブルと交換できます。ここで、pt はパーティション化されたテーブル、p はパーティション化されていないテーブル nt と交換する pt のパーティションまたはサブパーティションです (次の記述が true である場合)。

1. テーブル nt 自体はパーティション化されていない。
2. テーブル nt は一時テーブルではない。
3. テーブル pt および nt の構造はそれ以外の点で同じである。
4. テーブル nt は外部キー参照を含まず、ほかのどのテーブルも nt を参照する外部キーを持たない。
5. nt 内に p のパーティション定義の境界の外に存在する行がない。

ALTER TABLE ステートメントに通常必要な ALTER、INSERT、および CREATE 権限に加えて、ALTER TABLE ... EXCHANGE PARTITION を実行するための DROP 権限が必要です。

ALTER TABLE ... EXCHANGE PARTITION の次の影響も考慮してください。

- ALTER TABLE ... EXCHANGE PARTITION を実行しても、パーティション化されたテーブルまたは交換されるテーブルに対するトリガーは呼び出されません。

- 交換されるテーブル内の AUTO_INCREMENT カラムがリセットされます。
- IGNORE キーワードは、ALTER TABLE ... EXCHANGE PARTITION と一緒に使用された場合、効果を持つません。

ALTER TABLE ... EXCHANGE PARTITION ステートメントの完全な構文を次に示します。ここで、pt はパーティション化されたテーブル、p は交換されるパーティションまたはサブパーティション、nt は p と交換されるパーティション化されていないテーブルです。

```
ALTER TABLE pt
   EXCHANGE PARTITION p
   WITH TABLE nt;
```

単一 ALTER TABLE EXCHANGE PARTITION ステートメントでは、1 つのパーティションまたはサブパーティションのみを 1 つのパーティション化されていないテーブルのみと交換できます。複数のパーティションまたはサブパーティションを交換するには、複数の ALTER TABLE EXCHANGE PARTITION ステートメントを使用してください。EXCHANGE PARTITION は、ほかの ALTER TABLE オプションと組み合わせることはできません。パーティション化されたテーブルによって使用されるパーティショニングおよび (該当する場合) サブパーティショニングには、MySQL 5.6 でサポートされる任意のタイプを選択できます。

パーティションをパーティション化されていないテーブルと交換する

次の SQL ステートメントを使用して、パーティション化されたテーブル e が作成および移入されているとします。

```
CREATE TABLE e (
   id INT NOT NULL,
   fname VARCHAR(30),
   lname VARCHAR(30)
)
   PARTITION BY RANGE (id) (
      PARTITION p0 VALUES LESS THAN (50),
      PARTITION p1 VALUES LESS THAN (100),
      PARTITION p2 VALUES LESS THAN (150),
      PARTITION p3 VALUES LESS THAN (MAXVALUE)
);

INSERT INTO e VALUES
   (1669, "Jim", "Smith"),
   (337, "Mary", "Jones"),
   (16, "Frank", "White"),
   (2005, "Linda", "Black");
```

ここで、e2 という名前の、e のパーティション化されていないコピーを作成します。これは、mysql クライアントを使用して次のように行うことができます。

```
mysql> CREATE TABLE e2 LIKE e;
Query OK, 0 rows affected (1.34 sec)

mysql> ALTER TABLE e2 REMOVE PARTITIONING;
Query OK, 0 rows affected (0.90 sec)
Records: 0  Duplicates: 0  Warnings: 0
```

テーブル e のどのパーティションに行が含まれるかは、次のように INFORMATION_SCHEMA.PARTITIONS テーブルを照会することで確認できます。

```
mysql> SELECT PARTITION_NAME, TABLE_ROWS
   ->     FROM INFORMATION_SCHEMA.PARTITIONS
   ->     WHERE TABLE_NAME = 'e';
+----------------+------------+
| PARTITION_NAME | TABLE_ROWS |
+----------------+------------+
| p0             |          1 |
| p1             |          0 |
| p2             |          0 |
| p3             |          3 |
+----------------+------------+
4 rows in set (0.00 sec)
```

> **注記**
>
> パーティション化された InnoDB テーブルの場合、INFORMATION_SCHEMA.PARTITIONS テーブルの TABLE_ROWS カラムに示される行数は、SQL 最適化で使用される見積もり値であり、常に正確とはかぎりません。

テーブル e 内のパーティション p0 をテーブルe2 と交換するには、次のような ALTER TABLE ステートメントを使用できます。

```
mysql> ALTER TABLE e EXCHANGE PARTITION p0 WITH TABLE e2;
Query OK, 0 rows affected (0.28 sec)
```

より正確に言うと、ここで発行したステートメントによって、パーティションで見つかる行がテーブルで見つかるものと交換されます。これがどのように行われたかは、前のように INFORMATION_SCHEMA.PARTITIONS テーブルを照会することで観察できます。パーティション p0 で以前は見つかったテーブル行が存在しなくなっています。

```
mysql> SELECT PARTITION_NAME, TABLE_ROWS
    ->     FROM INFORMATION_SCHEMA.PARTITIONS
    ->     WHERE TABLE_NAME = 'e';
+----------------+------------+
| PARTITION_NAME | TABLE_ROWS |
+----------------+------------+
| p0             |          0 |
| p1             |          0 |
| p2             |          0 |
| p3             |          3 |
+----------------+------------+
4 rows in set (0.00 sec)
```

テーブル e2 を照会すると、「見つからない」行がそこで見つかります。

```
mysql> SELECT * FROM e2;
+----+-------+-------+
| id | fname | lname |
+----+-------+-------+
| 16 | Frank | White |
+----+-------+-------+
1 row in set (0.00 sec)
```

パーティションと交換されるテーブルは、必ずしも空である必要はありません。これを実証するために、まず新しい行をテーブル e に挿入してから、この行がパーティション p0 に格納されていることを確認します (50 より小さい id カラム値を選択し、これをあとで PARTITIONS テーブルを照会することで確認します)。

```
mysql> INSERT INTO e VALUES (41, "Michael", "Green");
Query OK, 1 row affected (0.05 sec)

mysql> SELECT PARTITION_NAME, TABLE_ROWS
    ->     FROM INFORMATION_SCHEMA.PARTITIONS
    ->     WHERE TABLE_NAME = 'e';
+----------------+------------+
| PARTITION_NAME | TABLE_ROWS |
+----------------+------------+
| p0             |          1 |
| p1             |          0 |
| p2             |          0 |
| p3             |          3 |
+----------------+------------+
4 rows in set (0.00 sec)
```

ここで、前と同じ ALTER TABLE ステートメントを使用して、ふたたびパーティション p0 をテーブル e2 と交換します。

```
mysql> ALTER TABLE e EXCHANGE PARTITION p0 WITH TABLE e2;
Query OK, 0 rows affected (0.28 sec)
```

次のクエリーの出力は、ALTER TABLE ステートメントを発行する前に、パーティション p0 に格納されていたテーブル行およびテーブル e2 に格納されていたテーブル行の配置が切り替わったことを示しています。

```
mysql> SELECT * FROM e;
+------+-------+-------+
| id   | fname | lname |
+------+-------+-------+
|   16 | Frank | White |
| 1669 | Jim   | Smith |
|  337 | Mary  | Jones |
| 2005 | Linda | Black |
+------+-------+-------+
4 rows in set (0.00 sec)

mysql> SELECT PARTITION_NAME, TABLE_ROWS
    ->     FROM INFORMATION_SCHEMA.PARTITIONS
```

```
    ->     WHERE TABLE_NAME = 'e';
+----------------+------------+
| PARTITION_NAME | TABLE_ROWS |
+----------------+------------+
| p0             |          1 |
| p1             |          0 |
| p2             |          0 |
| p3             |          3 |
+----------------+------------+
4 rows in set (0.00 sec)

mysql> SELECT * FROM e2;
+----+---------+-------+
| id | fname   | lname |
+----+---------+-------+
| 41 | Michael | Green |
+----+---------+-------+
1 row in set (0.00 sec)
```

一致しない行

ALTER TABLE ... EXCHANGE PARTITION ステートメントを発行する前にパーティション化されていないテーブルで見つかる行は、それらがターゲットパーティションに格納されるために必要な条件を満たしている必要があり、そうでない場合はステートメントが失敗することを覚えておいてください。これがどのように発生するかを確認するために、まずテーブルe のパーティション p0 のパーティション定義の境界外の行を、e2 に挿入します。たとえば、id カラム値が大きすぎる行を挿入してから、テーブルをパーティションとふたたび交換してみてください。

```
mysql> INSERT INTO e2 VALUES (51, "Ellen", "McDonald");
Query OK, 1 row affected (0.08 sec)

mysql> ALTER TABLE e EXCHANGE PARTITION p0 WITH TABLE e2;
ERROR 1707 (HY000): Found row that does not match the partition
```

IGNORE キーワードは受け入れられますが、次に示すように EXCHANGE PARTITION で使用されるときは効果がありません。

```
mysql> ALTER IGNORE TABLE e EXCHANGE PARTITION p0 WITH TABLE e2;
ERROR 1707 (HY000): Found row that does not match the partition
```

サブパーティションをパーティション化されていないテーブルと交換する

ALTER TABLE ... EXCHANGE PARTITION ステートメントを使用して、サブパーティション化されたテーブルのサブパーティション (セクション19.2.6「サブパーティショニング」を参照してください) をパーティション化されていないテーブルと交換することもできます。次の例では、まず RANGE によってパーティション化され、KEY によってサブパーティション化されたテーブル es を作成し、テーブル e と同様にこのテーブルに移入してから、このテーブルの空のパーティション化されていないコピー es2 を作成します。

```
mysql> CREATE TABLE es (
    ->     id INT NOT NULL,
    ->     fname VARCHAR(30),
    ->     lname VARCHAR(30)
    -> )
    ->     PARTITION BY RANGE (id)
    ->     SUBPARTITION BY KEY (lname)
    ->     SUBPARTITIONS 2 (
    ->         PARTITION p0 VALUES LESS THAN (50),
    ->         PARTITION p1 VALUES LESS THAN (100),
    ->         PARTITION p2 VALUES LESS THAN (150),
    ->         PARTITION p3 VALUES LESS THAN (MAXVALUE)
    ->     );
Query OK, 0 rows affected (2.76 sec)

mysql> INSERT INTO es VALUES
    ->     (1669, "Jim", "Smith"),
    ->     (337, "Mary", "Jones"),
    ->     (16, "Frank", "White"),
    ->     (2005, "Linda", "Black");
Query OK, 4 rows affected (0.04 sec)
Records: 4  Duplicates: 0  Warnings: 0

mysql> CREATE TABLE es2 LIKE es;
Query OK, 0 rows affected (1.27 sec)

mysql> ALTER TABLE es2 REMOVE PARTITIONING;
Query OK, 0 rows affected (0.70 sec)
```

```
Records: 0  Duplicates: 0  Warnings: 0
```

テーブル es を作成するときにサブパーティションの名前を明示的に指定しなかったけれども、PARTITIONS テーブルから選択するときに、次のように INFORMATION_SCHEMA からそのテーブルの SUBPARTITION_NAME を取り込むことで、それらに生成された名前を取得できます。

```
mysql> SELECT PARTITION_NAME, SUBPARTITION_NAME, TABLE_ROWS
    ->     FROM INFORMATION_SCHEMA.PARTITIONS
    ->     WHERE TABLE_NAME = 'es';
+----------------+-------------------+------------+
| PARTITION_NAME | SUBPARTITION_NAME | TABLE_ROWS |
+----------------+-------------------+------------+
| p0             | p0sp0             |          1 |
| p0             | p0sp1             |          0 |
| p1             | p1sp0             |          0 |
| p1             | p1sp1             |          0 |
| p2             | p2sp0             |          0 |
| p2             | p2sp1             |          0 |
| p3             | p3sp0             |          3 |
| p3             | p3sp1             |          0 |
+----------------+-------------------+------------+
8 rows in set (0.00 sec)
```

次の ALTER TABLE ステートメントは、テーブル es のサブパーティション p3sp0 をパーティション化されていないテーブル es2 と交換します。

```
mysql> ALTER TABLE es EXCHANGE PARTITION p3sp0 WITH TABLE es2;
Query OK, 0 rows affected (0.29 sec)
```

次のクエリーを発行することで、それらの行が交換されたことを確認できます。

```
mysql> SELECT PARTITION_NAME, SUBPARTITION_NAME, TABLE_ROWS
    ->     FROM INFORMATION_SCHEMA.PARTITIONS
    ->     WHERE TABLE_NAME = 'es';
+----------------+-------------------+------------+
| PARTITION_NAME | SUBPARTITION_NAME | TABLE_ROWS |
+----------------+-------------------+------------+
| p0             | p0sp0             |          1 |
| p0             | p0sp1             |          0 |
| p1             | p1sp0             |          0 |
| p1             | p1sp1             |          0 |
| p2             | p2sp0             |          0 |
| p2             | p2sp1             |          0 |
| p3             | p3sp0             |          0 |
| p3             | p3sp1             |          0 |
+----------------+-------------------+------------+
8 rows in set (0.00 sec)

mysql> SELECT * FROM es2;
+------+-------+-------+
| id   | fname | lname |
+------+-------+-------+
| 1669 | Jim   | Smith |
|  337 | Mary  | Jones |
| 2005 | Linda | Black |
+------+-------+-------+
3 rows in set (0.00 sec)
```

テーブルがサブパーティション化されている場合、次に示すように、パーティション化されていないテーブルと交換できるのは、テーブルのパーティション全体ではなくサブパーティションのみです。

```
mysql> ALTER TABLE es EXCHANGE PARTITION p3 WITH TABLE es2;
ERROR 1704 (HY000): Subpartitioned table, use subpartition instead of partition
```

MySQL によって使用されるテーブル構造の比較は非常に厳密です。カラムの数、順序、名前、および型、さらにパーティション化されたテーブルとパーティション化されていないテーブルのインデックスが、正確に一致する必要があります。また、両方のテーブルが同じストレージエンジンを使用している必要があります。

```
mysql> CREATE TABLE es3 LIKE e;
Query OK, 0 rows affected (1.31 sec)

mysql> ALTER TABLE es3 REMOVE PARTITIONING;
Query OK, 0 rows affected (0.53 sec)
Records: 0  Duplicates: 0  Warnings: 0

mysql> SHOW CREATE TABLE es3\G
*************************** 1. row ***************************
```

```
      Table: es3
Create Table: CREATE TABLE `es3` (
  `id` int(11) NOT NULL,
  `fname` varchar(30) DEFAULT NULL,
  `lname` varchar(30) DEFAULT NULL
) ENGINE=InnoDB DEFAULT CHARSET=latin1
1 row in set (0.00 sec)

mysql> ALTER TABLE es3 ENGINE = MyISAM;
Query OK, 0 rows affected (0.15 sec)
Records: 0  Duplicates: 0  Warnings: 0

mysql> ALTER TABLE es EXCHANGE PARTITION p3sp0 WITH TABLE es3;
ERROR 1497 (HY000): The mix of handlers in the partitions is not allowed in this version of MySQL
```

19.3.4 パーティションの保守

いくつかのテーブルおよびパーティション保守タスクは、MySQL 5.6 のパーティション化されたテーブルでそのような目的を実現するための SQL ステートメントを使用して実行できます。

パーティション化されたテーブルのテーブル保守は、パーティション化されたテーブルでサポートされる CHECK TABLE、OPTIMIZE TABLE、ANALYZE TABLE、および REPAIR TABLE ステートメントを使用して実現できます。

次のリストで説明しているように、ALTER TABLE へのいくつかの拡張を使用して、1 つ以上のパーティションに対してこのタイプの操作を直接実行できます。

- パーティションの再構築　パーティションを再構築します。これは、パーティションに格納されているすべてのレコードを削除してからそれらを再度挿入することと同じ効果があります。これはデフラグに役立つことがあります。

  例:

  ```
  ALTER TABLE t1 REBUILD PARTITION p0, p1;
  ```

- パーティションの最適化　パーティションから多数の行を削除した場合、または可変長行を持つ (つまり、VARCHAR、BLOB、または TEXT カラムを持つ) パーティション化されたテーブルに多くの変更を行なった場合は、ALTER TABLE ... OPTIMIZE PARTITION を使用して、未使用領域を解放したりパーティションデータファイルをデフラグしたりできます。

  例:

  ```
  ALTER TABLE t1 OPTIMIZE PARTITION p0, p1;
  ```

  指定されたパーティションに OPTIMIZE PARTITION を使用することは、そのパーティションに CHECK PARTITION、ANALYZE PARTITION、および REPAIR PARTITION を実行することと同等です。

  一部の MySQL ストレージエンジン (InnoDB を含む) は、パーティションごとの最適化をサポートしません。これらの場合は、ALTER TABLE ... OPTIMIZE PARTITION がテーブル全体を再構築されます。MySQL 5.6.9 以降では、そのようなテーブルでこのステートメントを実行すると、テーブル全体が再構築および分析され、該当する警告が発行されます (Bug #11751825、Bug #42822)。この問題を回避するには、代わりに ALTER TABLE ... REBUILD PARTITION および ALTER TABLE ... ANALYZE PARTITION を使用してください。

- パーティションの分析　これは、パーティションのキー分布を読み取って格納します。

  例:

  ```
  ALTER TABLE t1 ANALYZE PARTITION p3;
  ```

- パーティションの修復　これは、破損したパーティションを修復します。

  例:

  ```
  ALTER TABLE t1 REPAIR PARTITION p0,p1;
  ```

- パーティションのチェック　パーティション化されていないテーブルで CHECK TABLE を使用できるのとほぼ同様に、パーティションのエラーをチェックできます。

  例:

  ```
  ALTER TABLE trb3 CHECK PARTITION p1;
  ```

このコマンドは、テーブル t1 のパーティション p1 のデータまたはインデックスが破損しているかどうかを通知します。その場合は、ALTER TABLE ... REPAIR PARTITION を使用してパーティションを修復してください。

上記のリストの各ステートメントでは、パーティション名のリストの代わりにキーワード ALL もサポートされます。ALL を使用すると、テーブル内のすべてのパーティションにステートメントが作用します。

パーティション化されたテーブルでは mysqlcheck および myisamchk の使用はサポートされません。

MySQL 5.6 では、ALTER TABLE ... TRUNCATE PARTITION を使用してパーティションを切り捨てることもできます。このステートメントは、TRUNCATE TABLE がテーブルからすべての行を削除するのとほぼ同様に、1 つ以上のパーティションからすべての行を削除するために使用できます。

ALTER TABLE ... TRUNCATE PARTITION ALL はテーブル内のすべてのパーティションを切り捨てます。

ANALYZE、CHECK、OPTIMIZE、REBUILD、REPAIR、および TRUNCATE 操作は、サブパーティションではサポートされません。

19.3.5 パーティションに関する情報を取得する

このセクションでは、既存のパーティションに関する情報を取得する方法 (いくつかの方法が可能) について説明します。そのような情報を取得する方法には次のものが含まれます。

- SHOW CREATE TABLE ステートメントを使用して、パーティション化されたテーブルの作成に使用されたパーティショニング句を表示する。
- SHOW TABLE STATUS ステートメントを使用して、テーブルがパーティション化されているかどうかを判別する。
- INFORMATION_SCHEMA.PARTITIONS テーブルを照会する。
- EXPLAIN PARTITIONS SELECT ステートメントを使用して、指定された SELECT によってどのパーティションが使用されているかを確認する。

この章のほかの場所でも説明しているように、SHOW CREATE TABLE の出力にはパーティション化されたテーブルの作成に使用された PARTITION BY 句が含まれます。例:

```
mysql> SHOW CREATE TABLE trb3\G
*************************** 1. row ***************************
       Table: trb3
Create Table: CREATE TABLE `trb3` (
  `id` int(11) default NULL,
  `name` varchar(50) default NULL,
  `purchased` date default NULL
) ENGINE=MyISAM DEFAULT CHARSET=latin1
PARTITION BY RANGE (YEAR(purchased)) (
  PARTITION p0 VALUES LESS THAN (1990) ENGINE = MyISAM,
  PARTITION p1 VALUES LESS THAN (1995) ENGINE = MyISAM,
  PARTITION p2 VALUES LESS THAN (2000) ENGINE = MyISAM,
  PARTITION p3 VALUES LESS THAN (2005) ENGINE = MyISAM
)
1 row in set (0.00 sec)
```

パーティション化されたテーブルに対する SHOW TABLE STATUS の出力は、Create_options カラムに文字列 partitioned が含まれることを除いて、パーティション化されていないテーブルの場合と同じです。Engine カラムには、テーブルのすべてのパーティションによって使用されるストレージエンジンの名前が含まれます。(このステートメントについての詳細は、セクション13.7.5.37「SHOW TABLE STATUS 構文」を参照してください)。

パーティションに関する情報は、PARTITIONS テーブルを含む INFORMATION_SCHEMA からも取得できます。セクション21.13「INFORMATION_SCHEMA PARTITIONS テーブル」を参照してください。

指定された SELECT クエリーでパーティション化されたテーブルのどのパーティションが使用されるかは、EXPLAIN PARTITIONS を使用して判別できます。PARTITIONS キーワードは、partitions カラム (どのパーティションからのレコードがクエリーで照合されるかをリストする) を EXPLAIN の出力に追加します。

テーブル trb1 が次のように作成されて移入されているとします。

```
CREATE TABLE trb1 (id INT, name VARCHAR(50), purchased DATE)
   PARTITION BY RANGE(id)
```

```
    (
        PARTITION p0 VALUES LESS THAN (3),
        PARTITION p1 VALUES LESS THAN (7),
        PARTITION p2 VALUES LESS THAN (9),
        PARTITION p3 VALUES LESS THAN (11)
    );

INSERT INTO trb1 VALUES
    (1, 'desk organiser', '2003-10-15'),
    (2, 'CD player', '1993-11-05'),
    (3, 'TV set', '1996-03-10'),
    (4, 'bookcase', '1982-01-10'),
    (5, 'exercise bike', '2004-05-09'),
    (6, 'sofa', '1987-06-05'),
    (7, 'popcorn maker', '2001-11-22'),
    (8, 'aquarium', '1992-08-04'),
    (9, 'study desk', '1984-09-16'),
    (10, 'lava lamp', '1998-12-25');
```

SELECT * FROM trb1; などのクエリーでどのパーティションが使用されるかを次のように確認できます。

```
mysql> EXPLAIN PARTITIONS SELECT * FROM trb1\G
*************************** 1. row ***************************
           id: 1
  select_type: SIMPLE
        table: trb1
   partitions: p0,p1,p2,p3
         type: ALL
possible_keys: NULL
          key: NULL
      key_len: NULL
          ref: NULL
         rows: 10
        Extra: Using filesort
```

この場合、4 つのパーティションがすべて検索されます。ただし、次のようにパーティショニングキーを使用する制限条件をクエリーに追加すると、一致する値が含まれているパーティションのみがスキャンされることがわかります。

```
mysql> EXPLAIN PARTITIONS SELECT * FROM trb1 WHERE id < 5\G
*************************** 1. row ***************************
           id: 1
  select_type: SIMPLE
        table: trb1
   partitions: p0,p1
         type: ALL
possible_keys: NULL
          key: NULL
      key_len: NULL
          ref: NULL
         rows: 10
        Extra: Using where
```

EXPLAIN PARTITIONS では、普通の EXPLAIN SELECT ステートメントと同様に、使用されているキーおよび使用可能なキーに関する情報が表示されます。

```
mysql> ALTER TABLE trb1 ADD PRIMARY KEY (id);
Query OK, 10 rows affected (0.03 sec)
Records: 10  Duplicates: 0  Warnings: 0

mysql> EXPLAIN PARTITIONS SELECT * FROM trb1 WHERE id < 5\G
*************************** 1. row ***************************
           id: 1
  select_type: SIMPLE
        table: trb1
   partitions: p0,p1
         type: range
possible_keys: PRIMARY
          key: PRIMARY
      key_len: 4
          ref: NULL
         rows: 7
        Extra: Using where
```

EXPLAIN PARTITIONS に関する次の制約および制限に注目してください。

- EXTENDED および PARTITIONS キーワードは、同一 EXPLAIN ... SELECT ステートメント内で一緒に使用できません。そうしようとすると構文エラーになります。

- クエリーを検査するためにパーティション化されていないテーブルで EXPLAIN PARTITIONS を使用すると、エラーは発生しませんが、partitions カラムの値は常に NULL となります。

EXPLAIN PARTITIONS 出力の rows カラムには、テーブルの合計行数が表示されます。

セクション13.8.2「EXPLAIN 構文」も参照してください。

19.4 パーティションプルーニング

このセクションでは、パーティションプルーニングと呼ばれる最適化について説明します。パーティションプルーニングの背後の中心概念は比較的単純で、「一致する値がないパーティションはスキャンしない」と表現できます。次のステートメントによって定義されたパーティション化されたテーブル t1 があるとします。

```
CREATE TABLE t1 (
    fname VARCHAR(50) NOT NULL,
    lname VARCHAR(50) NOT NULL,
    region_code TINYINT UNSIGNED NOT NULL,
    dob DATE NOT NULL
)
PARTITION BY RANGE( region_code ) (
    PARTITION p0 VALUES LESS THAN (64),
    PARTITION p1 VALUES LESS THAN (128),
    PARTITION p2 VALUES LESS THAN (192),
    PARTITION p3 VALUES LESS THAN MAXVALUE
);
```

次のような SELECT ステートメントから結果を取得するとします。

```
SELECT fname, lname, region_code, dob
    FROM t1
    WHERE region_code > 125 AND region_code < 130;
```

返すべき行がパーティション p0 にも p3 にもないことは簡単にわかります。つまり、パーティション p1 および p2 のみを検索して一致する行を見つける必要があります。そうすることにより、一致する行を検索するために消費する時間と労力が、テーブル内のすべてのパーティションのスキャンに必要なものより少なくて済みます。必要のないパーティションをこのように「省く」ことをプルーニングといいます。オプティマイザがこのクエリーの実行でパーティションプルーニングを使用できるときは、同じカラム定義およびデータが含まれているパーティション化されていないテーブルで同じクエリーを実行するよりも、速度が大幅に向上することがあります。

WHERE 条件を次の 2 つのケースのいずれかにまとめられるとき、オプティマイザはプルーニングを実行できます。

- partition_column = constant
- partition_column IN (constant1, constant2, ..., constantN)

最初のケースで、オプティマイザは指定された値についてパーティショニング式を単純に評価し、どのパーティションに値が含まれるかを判別して、このパーティションのみをスキャンします。多くの場合、この等号を別の算術比較 (<、>、<=、>=、および <> を含む) に置き換えることができます。WHERE 句で BETWEEN を使用するクエリーも、パーティションプルーニングの利点を活用できます。このセクションの後続の例を参照してください。

2 番目のケースで、オプティマイザはリスト内の各値についてパーティショニング式を評価して、一致するパーティションのリストを作成してから、このパーティションリストのパーティションのみをスキャンします。

MySQL は、パーティションプルーニングを SELECT、DELETE、および UPDATE ステートメントに適用できます。INSERT ステートメントは現在のところ、プルーニングできません。

短い範囲にもプルーニングを適用できます (オプティマイザが同等の値リストに変換できるもの)。たとえば、前の例では、WHERE 句を WHERE region_code IN (126, 127, 128, 129) に変換できます。これにより、オプティマイザは、リスト内の最初の 3 つの値はパーティション p1 に見つかり、残りの 3 つの値はパーティション p2 に見つかり、ほかのパーティションには関連する値は含まれないため一致する行を検索する必要がないと判断できます。

MySQL 5.6 のオプティマイザは、RANGE COLUMNS または LIST COLUMNS パーティショニングを使用するテーブルの複数のカラムで、前述のタイプの比較を使用する WHERE 条件のプルーニングを実行することもできます。

このタイプの最適化は、パーティショニング式が同一性や範囲で構成されていてそれを同一性のセットにまとめられるとき、またはパーティショニング式が増減する関係を表すときに適用できます。パーティショニング式が YEAR() または TO_DAYS() 関数を使用するとき、DATE カラムまたは DATETIME カラムでパーティション化

されるテーブルにプルーニングを適用することもできます。また、MySQL 5.6 では、パーティショニング式が TO_SECONDS() 関数を使用するとき、そのようなテーブルにプルーニングを適用できます。

テーブル t2 が次のように定義されて、DATE カラムでパーティション化されるとします。

```
CREATE TABLE t2 (
    fname VARCHAR(50) NOT NULL,
    lname VARCHAR(50) NOT NULL,
    region_code TINYINT UNSIGNED NOT NULL,
    dob DATE NOT NULL
)
PARTITION BY RANGE( YEAR(dob) ) (
    PARTITION d0 VALUES LESS THAN (1970),
    PARTITION d1 VALUES LESS THAN (1975),
    PARTITION d2 VALUES LESS THAN (1980),
    PARTITION d3 VALUES LESS THAN (1985),
    PARTITION d4 VALUES LESS THAN (1990),
    PARTITION d5 VALUES LESS THAN (2000),
    PARTITION d6 VALUES LESS THAN (2005),
    PARTITION d7 VALUES LESS THAN MAXVALUE
);
```

t2 を使用する次のステートメントでは、パーティションプルーニングを使用できます。

```
SELECT * FROM t2 WHERE dob = '1982-06-23';

UPDATE t2 SET region_code = 8 WHERE dob BETWEEN '1991-02-15' AND '1997-04-25';

DELETE FROM t2 WHERE dob >= '1984-06-21' AND dob <= '1999-06-21'
```

最後のステートメントの場合、オプティマイザは次のようにも動作できます。

1. 範囲の下限が含まれるパーティションを見つけます。

   YEAR('1984-06-21') は値 1984 を返し、それはパーティション d3 に見つかります。

2. 範囲の上限が含まれるパーティションを見つけます。

   YEAR('1999-06-21') は 1999 と評価され、それはパーティション d5 に見つかります。

3. これらの 2 つのパーティションおよびそれらの間にある可能性のあるパーティションのみをスキャンします。

   この場合、これはパーティション d3、d4、および d5 のみがスキャンされることを意味します。残りのパーティションは安全に無視できます (そして無視されます)。

   > **重要**
   >
   > パーティション化されたテーブルに対するステートメントの WHERE 条件で参照される無効な DATE および DATETIME 値は、NULL として扱われます。これは、SELECT * FROM partitioned_table WHERE date_column < '2008-12-00' などのクエリーは値を返さないことを意味します (Bug #40972 を参照してください)。

ここまで、RANGE パーティショニングを使用する例のみを見てきましたが、プルーニングはほかのパーティショニングタイプにも適用できます。

次に示すテーブル t3 のように、パーティショニング式が増加または減少している LIST によってパーティション化されたテーブルがあるとします。(この例では、簡単にするために region_code カラムが値 1 から 10 まで (両端を含む) に制限されると想定します)。

```
CREATE TABLE t3 (
    fname VARCHAR(50) NOT NULL,
    lname VARCHAR(50) NOT NULL,
    region_code TINYINT UNSIGNED NOT NULL,
    dob DATE NOT NULL
)
PARTITION BY LIST(region_code) (
    PARTITION r0 VALUES IN (1, 3),
    PARTITION r1 VALUES IN (2, 5, 8),
    PARTITION r2 VALUES IN (4, 9),
    PARTITION r3 VALUES IN (6, 7, 10)
);
```

SELECT * FROM t3 WHERE region_code BETWEEN 1 AND 3 などのステートメントの場合、オプティマイザはどのパーティションで値 1、2、および 3 が見つかるかを判別して (r0 および r1)、残りのもの (r2 および r3) をスキップします。

HASH または [LINEAR] KEY によってパーティション化されたテーブルの場合も、パーティショニング式で使用されるカラムに対して WHERE 句が単純な = 関係を使用しているときは、パーティションプルーニングを適用できます。次のように作成されたテーブルがあるとします。

```
CREATE TABLE t4 (
    fname VARCHAR(50) NOT NULL,
    lname VARCHAR(50) NOT NULL,
    region_code TINYINT UNSIGNED NOT NULL,
    dob DATE NOT NULL
)
PARTITION BY KEY(region_code)
PARTITIONS 8;
```

カラム値を定数と比較するステートメントはプルーニングできます。

```
UPDATE t4 WHERE region_code = 7;
```

プルーニングは短い範囲にも適用できます。オプティマイザがそのような条件を IN 関係に変換できるためです。たとえば、前に定義したものと同じテーブル t4 を使用して、次のようなクエリーをプルーニングできます。

```
SELECT * FROM t4 WHERE region_code > 2 AND region_code < 6;

SELECT * FROM t4 WHERE region_code BETWEEN 3 AND 5;
```

どちらの場合も、WHERE 句はオプティマイザによって WHERE region_code IN (3, 4, 5) に変換されます。

> **重要**
>
> この最適化は、範囲サイズがパーティションの数より小さい場合にのみ使用されます。次のステートメントがあるとします。
>
> ```
> DELETE FROM t4 WHERE region_code BETWEEN 4 AND 12;
> ```
>
> WHERE 句の範囲は 9 個の値 (4、5、6、7、8、9、10、11、12) ですが、t4 のパーティションは 8 個だけです。これはこの DELETE をプルーニングできないことを意味します。

HASH または [LINEAR] KEY によってパーティション化されたテーブルの場合、プルーニングを使用できるのは整数カラムに対してのみです。たとえば、dob は DATE カラムであるため、次のステートメントにはプルーニングを使用できません。

```
SELECT * FROM t4 WHERE dob >= '2001-04-14' AND dob <= '2005-10-15';
```

ただし、このテーブルが年値を INT カラムに格納する場合は、WHERE year_col >= 2001 AND year_col <= 2005 を持つクエリーをプルーニングできます。

MySQL 5.6.8 以降では、自動パーティショニングを提供するストレージエンジン (MySQL Cluster によって使用される NDB ストレージエンジン (現在のところ、MySQL 5.6 ではサポートされません) など) を使用するすべてのテーブルで、パーティションプルーニングは無効です(Bug #14672885)。MySQL 5.6.10 以降では、明示的にパーティション化されている場合は、そのようなテーブルをプルーニングできます(Bug #14827952)。

19.5 パーティション選択

MySQL 5.6 は、ステートメントを実行するときに、指定された WHERE 条件と一致する行をチェックすべきパーティションおよびサブパーティションの明示的選択をサポートします。パーティション選択は、特定のパーティションのみで一致がチェックされる点でパーティションプルーニングと似ていますが、2 つの重要な点で異なります。

1. チェックされるパーティションは、パーティションプルーニングと異なり (自動)、ステートメントの発行者が指定します。
2. パーティションプルーニングはクエリーのみに適用されますが、明示的なパーティション選択はクエリーおよびいくつかの DML ステートメントの両方でサポートされます。

明示的なパーティション選択をサポートする SQL ステートメントを次に一覧します。

- SELECT
- DELETE

- INSERT
- REPLACE
- UPDATE
- LOAD DATA。
- LOAD XML。

このセクションの残りの部分では、上記に一覧したステートメントに一般的に適用される明示的パーティション選択について説明し、いくつかの例を示します。

明示的パーティション選択は、PARTITION オプションを使用して実装されます。サポートされるすべてのステートメントについて、このオプションは次のような構文を使用します。

```
    PARTITION (partition_names)

    partition_names:
        partition_name, ...
```

このオプションは常に、パーティションが属するテーブルの名前の後ろに続けます。partition_names は、使用されるパーティションまたはサブパーティションのカンマ区切りのリストです。このリスト内の各名前は、指定されたテーブルの既存のパーティションまたはサブパーティションの名前である必要があります。パーティションまたはサブパーティションが見つからない場合、ステートメントはエラー (partition 'partition_name' doesn't exist) で失敗します。partition_names に指定するパーティションまたはサブパーティションは、任意の順序でリストでき、重複していてもかまいません。

PARTITION オプションを使用すると、リストされたパーティションおよびサブパーティションのみで一致する行がチェックされます。このオプションを SELECT ステートメントで使用すると、指定したパーティションに属する行を判別できます。次のようなステートメントを使用して作成および移入された、employees という名前のパーティション化されたテーブルがあるとします。

```
SET @@SQL_MODE = '';

CREATE TABLE employees  (
    id INT NOT NULL AUTO_INCREMENT PRIMARY KEY,
    fname VARCHAR(25) NOT NULL,
    lname VARCHAR(25) NOT NULL,
    store_id INT NOT NULL,
    department_id INT NOT NULL
)
    PARTITION BY RANGE(id)  (
        PARTITION p0 VALUES LESS THAN (5),
        PARTITION p1 VALUES LESS THAN (10),
        PARTITION p2 VALUES LESS THAN (15),
        PARTITION p3 VALUES LESS THAN MAXVALUE
);

INSERT INTO employees VALUES
    ('', 'Bob', 'Taylor', 3, 2), ('', 'Frank', 'Williams', 1, 2),
    ('', 'Ellen', 'Johnson', 3, 4), ('', 'Jim', 'Smith', 2, 4),
    ('', 'Mary', 'Jones', 1, 1), ('', 'Linda', 'Black', 2, 3),
    ('', 'Ed', 'Jones', 2, 1), ('', 'June', 'Wilson', 3, 1),
    ('', 'Andy', 'Smith', 1, 3), ('', 'Lou', 'Waters', 2, 4),
    ('', 'Jill', 'Stone', 1, 4), ('', 'Roger', 'White', 3, 2),
    ('', 'Howard', 'Andrews', 1, 2), ('', 'Fred', 'Goldberg', 3, 3),
    ('', 'Barbara', 'Brown', 2, 3), ('', 'Alice', 'Rogers', 2, 2),
    ('', 'Mark', 'Morgan', 3, 3), ('', 'Karen', 'Cole', 3, 2);
```

どの行がパーティション p1 に格納されているかは次のように確認できます。

```
mysql> SELECT * FROM employees PARTITION (p1);
+----+-------+--------+----------+---------------+
| id | fname | lname  | store_id | department_id |
+----+-------+--------+----------+---------------+
|  5 | Mary  | Jones  |        1 |             1 |
|  6 | Linda | Black  |        2 |             3 |
|  7 | Ed    | Jones  |        2 |             1 |
|  8 | June  | Wilson |        3 |             1 |
|  9 | Andy  | Smith  |        1 |             3 |
+----+-------+--------+----------+---------------+
5 rows in set (0.00 sec)
```

この結果は、クエリー SELECT * FROM employees WHERE id BETWEEN 5 AND 9 によって取得されるものと同じです。

複数のパーティションからの行を取得するには、それらの名前をカンマ区切りのリストとして指定します。たとえば、SELECT * FROM employees PARTITION (p1, p2) はパーティション p1 および p2 からのすべての行を返し、残りのパーティションからの行を除外します。

パーティション化されたテーブルに対する有効なクエリーは、PARTITION オプションを使用して、結果を 1 つ以上の目的のパーティションに制限するように書き直すことができます。WHERE 条件、ORDER BY オプション、LIMIT オプションなどを使用できます。集約関数を HAVING および GROUP BY オプション付きで使用することもできます。次の各クエリーは、前に定義した employees テーブルで実行するときに、有効な結果を生成します。

```
mysql> SELECT * FROM employees PARTITION (p0, p2)
    ->     WHERE lname LIKE 'S%';
+----+-------+-------+----------+---------------+
| id | fname | lname | store_id | department_id |
+----+-------+-------+----------+---------------+
|  4 | Jim   | Smith |        2 |             4 |
| 11 | Jill  | Stone |        1 |             4 |
+----+-------+-------+----------+---------------+
2 rows in set (0.00 sec)

mysql> SELECT id, CONCAT(fname, ' ', lname) AS name
    ->     FROM employees PARTITION (p0) ORDER BY lname;
+----+----------------+
| id | name           |
+----+----------------+
|  3 | Ellen Johnson  |
|  4 | Jim Smith      |
|  1 | Bob Taylor     |
|  2 | Frank Williams |
+----+----------------+
4 rows in set (0.06 sec)

mysql> SELECT store_id, COUNT(department_id) AS c
    ->     FROM employees PARTITION (p1,p2,p3)
    ->     GROUP BY store_id HAVING c > 4;
+---+----------+
| c | store_id |
+---+----------+
| 5 |        2 |
| 5 |        3 |
+---+----------+
2 rows in set (0.00 sec)
```

パーティション選択を使用するステートメントは、MySQL 5.6 でサポートされるパーティショニングタイプを使用するテーブルで使用できます。テーブルが [LINEAR] HASH または [LINEAR] KEY パーティショニングを使用して作成されているけれども、パーティションの名前が指定されていない場合は、MySQL はパーティションに p0、p1、p2、...、pN-1 という名前を自動的に付けます。ここで、N はパーティションの数です。明示的に名前が付けられていないサブパーティションの場合、MySQL は各パーティション pX 内のサブパーティションに pXsp0、pXsp1、pXsp2、...、pXspM-1 という名前を自動的に割り当てます。ここで、M はサブパーティションの数です。このテーブルで SELECT (または明示的パーティション選択が許可されるほかの SQL ステートメント) を実行するときは、次のようにこれらの生成された名前を PARTITION オプションで使用できます。

```
mysql> CREATE TABLE employees_sub  (
    ->     id INT NOT NULL AUTO_INCREMENT,
    ->     fname VARCHAR(25) NOT NULL,
    ->     lname VARCHAR(25) NOT NULL,
    ->     store_id INT NOT NULL,
    ->     department_id INT NOT NULL,
    ->     PRIMARY KEY pk (id, lname)
    -> )
    ->     PARTITION BY RANGE(id)
    ->     SUBPARTITION BY KEY (lname)
    ->     SUBPARTITIONS 2 (
    ->         PARTITION p0 VALUES LESS THAN (5),
    ->         PARTITION p1 VALUES LESS THAN (10),
    ->         PARTITION p2 VALUES LESS THAN (15),
    ->         PARTITION p3 VALUES LESS THAN MAXVALUE
    -> );
Query OK, 0 rows affected (1.14 sec)

mysql> INSERT INTO employees_sub   # re-use data in employees table
    ->     SELECT * FROM employees;
Query OK, 18 rows affected (0.09 sec)
Records: 18  Duplicates: 0  Warnings: 0

mysql> SELECT id, CONCAT(fname, ' ', lname) AS name
    ->     FROM employees_sub PARTITION (p2sp1);
```

```
+----+---------------+
| id | name          |
+----+---------------+
| 10 | Lou Waters    |
| 14 | Fred Goldberg |
+----+---------------+
2 rows in set (0.00 sec)
```

次のように PARTITION オプションを INSERT ... SELECT ステートメントの SELECT 部分に使用することもできます。

```
mysql> CREATE TABLE employees_copy LIKE employees;
Query OK, 0 rows affected (0.28 sec)

mysql> INSERT INTO employees_copy
    ->     SELECT * FROM employees PARTITION (p2);
Query OK, 5 rows affected (0.04 sec)
Records: 5  Duplicates: 0  Warnings: 0

mysql> SELECT * FROM employees_copy;
+----+---------+----------+----------+---------------+
| id | fname   | lname    | store_id | department_id |
+----+---------+----------+----------+---------------+
| 10 | Lou     | Waters   |        2 |             4 |
| 11 | Jill    | Stone    |        1 |             4 |
| 12 | Roger   | White    |        3 |             2 |
| 13 | Howard  | Andrews  |        1 |             2 |
| 14 | Fred    | Goldberg |        3 |             3 |
+----+---------+----------+----------+---------------+
5 rows in set (0.00 sec)
```

パーティション選択は結合と一緒に使用することもできます。次のステートメントを使用して 2 つのテーブルを作成して移入するとします。

```
CREATE TABLE stores (
    id INT NOT NULL AUTO_INCREMENT PRIMARY KEY,
    city VARCHAR(30) NOT NULL
)
    PARTITION BY HASH(id)
    PARTITIONS 2;

INSERT INTO stores VALUES
    (", 'Nambucca'), (", 'Uranga'),
    (", 'Bellingen'), (", 'Grafton');

CREATE TABLE departments  (
    id INT NOT NULL AUTO_INCREMENT PRIMARY KEY,
    name VARCHAR(30) NOT NULL
)
    PARTITION BY KEY(id)
    PARTITIONS 2;

INSERT INTO departments VALUES
    (", 'Sales'), (", 'Customer Service'),
    (", 'Delivery'), (", 'Accounting');
```

任意またはすべてのテーブルからのパーティション (またはサブパーティション、あるいはその両方) を結合で明示的に選択できます(指定されたテーブルからのパーティションを選択するために使用する PARTITION オプションは、テーブル名の直後、かつほかのすべてのオプション (テーブルエイリアスを含む) の前に指定します)。たとえば、次のクエリーは都市 Nambucca および Bellingen (stores テーブルのパーティション p0) のいずれかの店舗の販売部門または配送部門 (departments テーブルのパーティション p1) で働いているすべての従業員の、名前、従業員 ID、部門、および都市を取得します。

```
mysql> SELECT
    ->     e.id AS 'Employee ID', CONCAT(e.fname, ' ', e.lname) AS Name,
    ->     s.city AS City, d.name AS department
    -> FROM employees AS e
    ->     JOIN stores PARTITION (p1) AS s ON e.store_id=s.id
    ->     JOIN departments PARTITION (p0) AS d ON e.department_id=d.id
    -> ORDER BY e.lname;
+-------------+---------------+-----------+------------+
| Employee ID | Name          | City      | department |
+-------------+---------------+-----------+------------+
|          14 | Fred Goldberg | Bellingen | Delivery   |
|           5 | Mary Jones    | Nambucca  | Sales      |
|          17 | Mark Morgan   | Bellingen | Delivery   |
|           9 | Andy Smith    | Nambucca  | Delivery   |
|           8 | June Wilson   | Bellingen | Sales      |
```

```
+------------+---------------+-----------+------------+
5 rows in set (0.00 sec)
```

MySQL での結合の一般情報については、セクション13.2.9.2「JOIN 構文」を参照してください。

DELETE ステートメントで PARTITION オプションを使用すると、オプションにリストされているパーティション (およびサブパーティション (ある場合)) でのみ削除される行がチェックされます。次のようにほかのパーティションは無視されます。

```
mysql> SELECT * FROM employees WHERE fname LIKE 'j%';
+----+-------+--------+----------+---------------+
| id | fname | lname  | store_id | department_id |
+----+-------+--------+----------+---------------+
|  4 | Jim   | Smith  |        2 |             4 |
|  8 | June  | Wilson |        3 |             1 |
| 11 | Jill  | Stone  |        1 |             4 |
+----+-------+--------+----------+---------------+
3 rows in set (0.00 sec)

mysql> DELETE FROM employees PARTITION (p0, p1)
    ->     WHERE fname LIKE 'j%';
Query OK, 2 rows affected (0.09 sec)

mysql> SELECT * FROM employees WHERE fname LIKE 'j%';
+----+-------+-------+----------+---------------+
| id | fname | lname | store_id | department_id |
+----+-------+-------+----------+---------------+
| 11 | Jill  | Stone |        1 |             4 |
+----+-------+-------+----------+---------------+
1 row in set (0.00 sec)
```

WHERE 条件と一致するパーティション p0 および p1 内の 2 つの行のみが削除されました。SELECT を 2 回目に実行したときの結果から確認できるように、WHERE 条件に一致する 1 行がテーブルに残っていますが、別のパーティション (p2) にあります。

明示的パーティション選択を使用する UPDATE ステートメントも同様に動作します。次のステートメントを実行することによって確認できるように、PARTITION オプションによって参照されるパーティション内の行のみが、更新される行を判別するときに考慮されます。

```
mysql> UPDATE employees PARTITION (p0)
    ->     SET store_id = 2 WHERE fname = 'Jill';
Query OK, 0 rows affected (0.00 sec)
Rows matched: 0  Changed: 0  Warnings: 0

mysql> SELECT * FROM employees WHERE fname = 'Jill';
+----+-------+-------+----------+---------------+
| id | fname | lname | store_id | department_id |
+----+-------+-------+----------+---------------+
| 11 | Jill  | Stone |        1 |             4 |
+----+-------+-------+----------+---------------+
1 row in set (0.00 sec)

mysql> UPDATE employees PARTITION (p2)
    ->     SET store_id = 2 WHERE fname = 'Jill';
Query OK, 1 row affected (0.09 sec)
Rows matched: 1  Changed: 1  Warnings: 0

mysql> SELECT * FROM employees WHERE fname = 'Jill';
+----+-------+-------+----------+---------------+
| id | fname | lname | store_id | department_id |
+----+-------+-------+----------+---------------+
| 11 | Jill  | Stone |        2 |             4 |
+----+-------+-------+----------+---------------+
1 row in set (0.00 sec)
```

同様に、DELETE 付きで PARTITION を使用すると、パーティションリストに指定されたパーティション内の行のみが削除をチェックされます。

行を挿入するステートメントの動作は、適切なパーティションが見つからないとステートメントが失敗する点が異なります。これは、次のように INSERT および REPLACE ステートメントの両方に当てはまります。

```
mysql> INSERT INTO employees PARTITION (p2) VALUES (20, 'Jan', 'Jones', 1, 3);
ERROR 1729 (HY000): Found a row not matching the given partition set
mysql> INSERT INTO employees PARTITION (p3) VALUES (20, 'Jan', 'Jones', 1, 3);
Query OK, 1 row affected (0.07 sec)

mysql> REPLACE INTO employees PARTITION (p0) VALUES (20, 'Jan', 'Jones', 3, 2);
```

```
ERROR 1729 (HY000): Found a row not matching the given partition set

mysql> REPLACE INTO employees PARTITION (p3) VALUES (20, 'Jan', 'Jones', 3, 2);
Query OK, 2 rows affected (0.09 sec)
```

InnoDB ストレージエンジンを使用するパーティション化されたテーブルに複数の行を書き込むステートメントの場合、VALUES に続くリスト内の行を partition_names リストに指定されたパーティションのいずれかに書き込むことができないときは、ステートメント全体が失敗し、一行も書き込まれません。これについては、次の例 (employees テーブルを再使用) の INSERT ステートメントで示されています。

```
mysql> ALTER TABLE employees
    ->     REORGANIZE PARTITION p3 INTO (
    ->         PARTITION p3 VALUES LESS THAN (20),
    ->         PARTITION p4 VALUES LESS THAN (25),
    ->         PARTITION p5 VALUES LESS THAN MAXVALUE
    ->     );
Query OK, 6 rows affected (2.09 sec)
Records: 6  Duplicates: 0  Warnings: 0

mysql> SHOW CREATE TABLE employees\G
*************************** 1. row ***************************
       Table: employees
Create Table: CREATE TABLE `employees` (
  `id` int(11) NOT NULL AUTO_INCREMENT,
  `fname` varchar(25) NOT NULL,
  `lname` varchar(25) NOT NULL,
  `store_id` int(11) NOT NULL,
  `department_id` int(11) NOT NULL,
  PRIMARY KEY (`id`)
) ENGINE=InnoDB AUTO_INCREMENT=27 DEFAULT CHARSET=latin1
/*!50100 PARTITION BY RANGE (id)
(PARTITION p0 VALUES LESS THAN (5) ENGINE = InnoDB,
 PARTITION p1 VALUES LESS THAN (10) ENGINE = InnoDB,
 PARTITION p2 VALUES LESS THAN (15) ENGINE = InnoDB,
 PARTITION p3 VALUES LESS THAN (20) ENGINE = InnoDB,
 PARTITION p4 VALUES LESS THAN (25) ENGINE = InnoDB,
 PARTITION p5 VALUES LESS THAN MAXVALUE ENGINE = InnoDB) */
1 row in set (0.00 sec)

mysql> INSERT INTO employees PARTITION (p3, p4) VALUES
    ->     (24, 'Tim', 'Greene', 3, 1),  (26, 'Linda', 'Mills', 2, 1);
ERROR 1729 (HY000): Found a row not matching the given partition set

mysql> INSERT INTO employees PARTITION (p3, p4. p5) VALUES
    ->     (24, 'Tim', 'Greene', 3, 1),  (26, 'Linda', 'Mills', 2, 1);
Query OK, 2 rows affected (0.06 sec)
Records: 2  Duplicates: 0  Warnings: 0
```

前述のことは、複数の行を書き込む INSERT および REPLACE ステートメントの両方に当てはまります。

MySQL 5.6.10 以降では、自動パーティショニングを提供するストレージエンジン (NDB など) を使用するテーブルの場合、パーティション選択は無効になります。(Bug #14827952)。

19.6 パーティショニングの制約と制限

このセクションでは、MySQL パーティショニングサポートでの現在の制約と制限について説明します。

禁止されている構造体　次の構造体はパーティショニング式で許可されません。

- ストアドプロシージャー、ストアドファンクション、UDF、またはプラグイン。
- 宣言された変数またはユーザー変数。

パーティショニング式で許可される SQL 関数のリストについては、セクション19.6.3「関数に関連するパーティショニング制限」を参照してください。

算術および論理演算子　算術演算子 +、-、および * の使用は、パーティショニング式で許可されます。ただし、結果は整数値または NULL である必要があります (この章のほかの場所で説明しているように、[LINEAR] KEY パーティショニングの場合を除きます。詳細は、セクション19.2「パーティショニングタイプ」を参照してください)。

DIV 演算子もサポートされますが、/ 演算子は許可されません。(Bug #30188、Bug #33182)。

ビット演算子 |、&、^、<<、>>、および ~ はパーティショニング式では許可されません。

HANDLER ステートメント　　MySQL 5.6 では、HANDLER ステートメントはパーティション化されたテーブルでサポートされません。

サーバー SQL モード　　ユーザー定義パーティショニングを使用するテーブルは、それらが作成された時点で有効だった SQL モードを保持しません。セクション5.1.7「サーバー SQL モード」で説明したように、多くの MySQL 関数および演算子の結果はサーバー SQL モードに従って変更されることがあります。このため、パーティション化されたテーブルの作成後の任意の時点に SQL モードを変更すると、そのようなテーブルの動作が大きく変わることがあり、データの破損または損失が発生しやすくなることがあります。これらの理由により、パーティション化されたテーブルを作成したあとにサーバー SQL モードを決して変更しないことが強く推奨されています。

例　　次の例は、サーバー SQL モードを変更したことによる、パーティション化されたテーブルの動作の変化をいくつか示しています。

1. エラー処理　　次のように、パーティショニング式が column DIV 0、column MOD 0 などのいずれかであるパーティション化されたテーブルを作成するとします。

```
mysql> CREATE TABLE tn (c1 INT)
    ->     PARTITION BY LIST(1 DIV c1) (
    ->       PARTITION p0 VALUES IN (NULL),
    ->       PARTITION p1 VALUES IN (1)
    -> );
Query OK, 0 rows affected (0.05 sec)
```

ゼロで除算した結果に対する MySQL のデフォルト動作は、エラーを発生させずに NULL を返すことです。

```
mysql> SELECT @@sql_mode;
+------------+
| @@sql_mode |
+------------+
|            |
+------------+
1 row in set (0.00 sec)


mysql> INSERT INTO tn VALUES (NULL), (0), (1);
Query OK, 3 rows affected (0.00 sec)
Records: 3  Duplicates: 0  Warnings: 0
```

ただし、ゼロによる除算をエラーとして扱い、厳密なエラー処理を適用するようにサーバー SQL モードを変更すると、次のように同じ INSERT ステートメントが失敗します。

```
mysql> SET sql_mode='STRICT_ALL_TABLES,ERROR_FOR_DIVISION_BY_ZERO';
Query OK, 0 rows affected (0.00 sec)

mysql> INSERT INTO tn VALUES (NULL), (0), (1);
ERROR 1365 (22012): Division by 0
```

2. テーブルアクセス可能性　　サーバー SQL モードを変更することによって、パーティション化されたテーブルが使用できなくなることがあります。次の CREATE TABLE ステートメントは、NO_UNSIGNED_SUBTRACTION モードが有効である場合にのみ、正常に実行できます。

```
mysql> SELECT @@sql_mode;
+------------+
| @@sql_mode |
+------------+
|            |
+------------+
1 row in set (0.00 sec)

mysql> CREATE TABLE tu (c1 BIGINT UNSIGNED)
    ->   PARTITION BY RANGE(c1 - 10) (
    ->     PARTITION p0 VALUES LESS THAN (-5),
    ->     PARTITION p1 VALUES LESS THAN (0),
    ->     PARTITION p2 VALUES LESS THAN (5),
    ->     PARTITION p3 VALUES LESS THAN (10),
    ->     PARTITION p4 VALUES LESS THAN (MAXVALUE)
    -> );
ERROR 1563 (HY000): Partition constant is out of partition function domain

mysql> SET sql_mode='NO_UNSIGNED_SUBTRACTION';
Query OK, 0 rows affected (0.00 sec)

mysql> SELECT @@sql_mode;
+------------------------+
| @@sql_mode             |
```

```
+-------------------------+
| NO_UNSIGNED_SUBTRACTION |
+-------------------------+
1 row in set (0.00 sec)

mysql> CREATE TABLE tu (c1 BIGINT UNSIGNED)
    ->   PARTITION BY RANGE(c1 - 10) (
    ->     PARTITION p0 VALUES LESS THAN (-5),
    ->     PARTITION p1 VALUES LESS THAN (0),
    ->     PARTITION p2 VALUES LESS THAN (5),
    ->     PARTITION p3 VALUES LESS THAN (10),
    ->     PARTITION p4 VALUES LESS THAN (MAXVALUE)
    -> );
Query OK, 0 rows affected (0.05 sec)
```

tu を作成したあとに NO_UNSIGNED_SUBTRACTION サーバー SQL モードを削除すると、このテーブルにアクセスできなくなる可能性があります。

```
mysql> SET sql_mode='';
Query OK, 0 rows affected (0.00 sec)

mysql> SELECT * FROM tu;
ERROR 1563 (HY000): Partition constant is out of partition function domain
mysql> INSERT INTO tu VALUES (20);
ERROR 1563 (HY000): Partition constant is out of partition function domain
```

サーバー SQL モードは、パーティション化されたテーブルのレプリケーションにも影響します。マスターとスレーブで SQL モードが異なると、パーティショニング式が違って評価されることがあります。これにより、指定されたテーブルのマスターのコピーとスレーブのコピーでパーティション間のデータ配分が違ってくることがあり、マスターで成功するパーティション化されたテーブルへの挿入がスレーブで失敗することさえあります。最適な結果を得るために、マスターとスレーブとで常に同じサーバー SQL モードを使用してください。

パフォーマンス考慮事項　パーティショニング操作のパフォーマンスへの影響の一部を次のリストに示します。

- ファイルシステム操作　パーティショニングおよび再パーティショニング操作 (PARTITION BY ...、REORGANIZE PARTITIONS、または REMOVE PARTITIONING を指定した ALTER TABLE など) は、それらの実装のファイルシステム操作に依存します。これは、これらの操作の速度が、ファイルシステムのタイプと特性、ディスク速度、スワップ領域、オペレーティングシステムによるファイル処理効率、ファイル処理に関連する MySQL サーバーのオプションと変数などの要因に影響されることを意味します。特に、large_files_support が有効になっていて、open_files_limit が適切に設定されていることを確認してください。MyISAM ストレージエンジンを使用するパーティション化されたテーブルの場合、myisam_max_sort_file_size を増やすとパフォーマンスが向上することがあります。innodb_file_per_table を有効にすることで、InnoDB テーブルを使用するパーティショニングおよび再パーティショニング操作の効率が向上することがあります。

  パーティションの最大数も参照してください。

- テーブルロック　テーブルに対してパーティショニング操作を実行する処理は、テーブルに対して書き込みロックを設定します。そのようなテーブルからの読み取りは比較的影響を受けません。保留中の INSERT および UPDATE 操作は、パーティショニング操作が完了するとすぐに実行されます。

- ストレージエンジン　パーティショニング操作、クエリー、および更新操作は通常、InnoDB または NDB テーブルより MyISAM テーブルで高速である傾向があります。

- インデックス、パーティションプルーニング　パーティション化されていないテーブルと同様に、インデックスを適切に使用することで、パーティション化されたテーブルに対する照会速度が大幅に向上することがあります。また、パーティション化されたテーブルおよびこれらのテーブルに対するクエリーをパーティションプルーニングの利点を活用するように設計することで、パフォーマンスが劇的に向上することがあります。詳細は、セクション19.4「パーティションプルーニング」を参照してください。

  インデックスコンディションプッシュダウンは、パーティション化されたテーブルではサポートされません。セクション8.2.1.6「インデックスコンディションプッシュダウンの最適化」を参照してください。

- LOAD DATA のパフォーマンス　MySQL 5.6 では、LOAD DATA はパフォーマンスを向上させるためにバッファリングを使用します。これを実現するために、バッファーがパーティションごとに 130K バイトメモリーを使用することを認識してください。

パーティションの最大数
MySQL 5.6.7 より前は、NDB ストレージエンジンを使用しないテーブルで可能な最大パーティション数は 1024 でした。MySQL 5.6.7 以降は、この制限は 8192 パーティションに増えています。MySQL Server バージョンにかかわらず、この最大数にはサブパーティションが含まれます。

NDB ストレージエンジンを使用するテーブルのユーザー定義パーティションの最大数は、使用されている MySQL Cluster ソフトウェアのバージョン、データノードの数、およびその他の要因に応じて決まります。詳細は、NDB とユーザー定義のパーティション化を参照してください。

多数のパーティション (ただし、最大数より少ない) を持つテーブルを作成するときに、Got error ... from storage engine: Out of resources when opening fileなどのエラーメッセージが表示される場合は、open_files_limit システム変数の値を増やすことによってこの問題に対処できることがあります。ただし、これはオペレーティングシステムによって異なるため、すべてのプラットフォームで可能または推奨されるとはかぎりません。詳細は、セクションB.5.2.18「'File' が見つかりません、および同様のエラー」を参照してください。場合によっては、多数の (数百の) パーティションを使用することがほかの問題のために推奨されないこともあり、より多くのパーティションを使用することが自動的に良い結果となるとはかぎりません。

ファイルシステム操作も参照してください。

クエリーキャッシュがサポートされない
クエリーキャッシュはパーティション化されたテーブルではサポートされません。MySQL 5.6.5 以降は、クエリーキャッシュはパーティション化されたテーブルを使用するクエリーで自動的に無効になり、そのようなクエリーで有効にすることはできません。(Bug #53775)

パーティションごとのキーキャッシュ
MySQL 5.6 では、CACHE INDEX および LOAD INDEX INTO CACHE ステートメントを使用することで、キーキャッシュがパーティション化された MyISAM テーブルでサポートされます。キーキャッシュは 1 つ、複数、またはすべてのパーティションに定義でき、1 つ、複数、またはすべてのパーティションのインデックスをキーキャッシュにプリロードできます。

パーティション化された InnoDB テーブルで外部キーがサポートされない
InnoDB ストレージエンジンを使用するパーティション化されたテーブルでは、外部キーはサポートされません。これは具体的には、次の 2 つの記述が true であることを意味します。

1. ユーザー定義パーティショニングを使用する InnoDB テーブルの定義には、外部キー参照を含めることはできません。定義に外部キー参照が含まれる InnoDB テーブルはパーティション化できません。

2. InnoDB テーブル定義に、ユーザーパーティション化されたテーブルへの外部キー参照を含めることはできません。ユーザー定義パーティショニングを持つ InnoDB テーブルに、外部キーによって参照されるカラムを含めることはできません。

上記の制約のスコープには、InnoDB ストレージエンジンを使用するすべてのテーブルが含まれます。結果のテーブルがこれらの制約に違反する CREATE TABLE および ALTER TABLE ステートメントは許可されません。

ALTER TABLE ... ORDER BY　パーティション化されたテーブルに ALTER TABLE ... ORDER BY column ステートメントを実行すると、各パーティション内でのみ行が並べ替えられます。

主キーを変更することによる REPLACE ステートメントへの影響　テーブルの主キーを変更することが望ましい場合があります (セクション19.6.1「パーティショニングキー、主キー、および一意キー」を参照してください)。REPLACE ステートメントを使用するアプリケーションでこれを行うと、これらのステートメントの結果が大きく変わることがあることを認識してください。詳細および例については、セクション13.2.8「REPLACE 構文」を参照してください。

FULLTEXT インデックス
パーティション化されたテーブルは、InnoDB または MyISAM ストレージエンジンを使用するパーティション化されたテーブルでも、FULLTEXT インデックスまたは検索をサポートしません。

空間カラム　POINT、GEOMETRY などの空間データ型を持つカラムは、パーティション化されたテーブルで使用できません。

一時テーブル
一時テーブルはパーティション化できません (Bug #17497)。

ログテーブル　ログテーブルをパーティション化することはできません。そのようなテーブルに ALTER TABLE ... PARTITION BY ... ステートメントを実行すると、エラーで失敗します。

パーティショニングキーのデータ型
パーティショニングキーは、整数カラム、または整数に解決される式である必要があります。ENUM カラムを使用する式は使用できません。カラムまたは式値は NULL でもかまいません(セクション19.2.7「MySQL パーティショニングによる NULL の扱い」を参照してください)。

この制約には 2 つの例外があります。

1. [LINEAR] KEY によってパーティショニングするときは、TEXT または BLOB 以外の有効な MySQL データ型のカラムをパーティショニングキーとして使用できます。MySQL の内部キーハッシュ関数によって、これらの型から正しいデータ型が生成されるためです。たとえば、次の 2 つの CREATE TABLE ステートメントは有効です。

```
CREATE TABLE tkc (c1 CHAR)
PARTITION BY KEY(c1)
PARTITIONS 4;

CREATE TABLE tke
    ( c1 ENUM('red', 'orange', 'yellow', 'green', 'blue', 'indigo', 'violet') )
PARTITION BY LINEAR KEY(c1)
PARTITIONS 6;
```

2. RANGE COLUMNS または LIST COLUMNS によってパーティショニングするときは、文字列、DATE、および DATETIME カラムを使用できます。たとえば、次の各 CREATE TABLE ステートメントは有効です。

```
CREATE TABLE rc (c1 INT, c2 DATE)
PARTITION BY RANGE COLUMNS(c2) (
    PARTITION p0 VALUES LESS THAN('1990-01-01'),
    PARTITION p1 VALUES LESS THAN('1995-01-01'),
    PARTITION p2 VALUES LESS THAN('2000-01-01'),
    PARTITION p3 VALUES LESS THAN('2005-01-01'),
    PARTITION p4 VALUES LESS THAN(MAXVALUE)
);

CREATE TABLE lc (c1 INT, c2 CHAR(1))
PARTITION BY LIST COLUMNS(c2) (
    PARTITION p0 VALUES IN('a', 'd', 'g', 'j', 'm', 'p', 's', 'v', 'y'),
    PARTITION p1 VALUES IN('b', 'e', 'h', 'k', 'n', 'q', 't', 'w', 'z'),
    PARTITION p2 VALUES IN('c', 'f', 'i', 'l', 'o', 'r', 'u', 'x', NULL)
);
```

上記の例外は、BLOB または TEXT カラム型には該当しません。

サブクエリー
パーティショニングキーはサブクエリーにできません (そのサブクエリーが整数値または NULL に解決される場合でも)。

サブパーティションに関する問題
サブパーティションは HASH または KEY パーティショニングを使用する必要があります。サブパーティション化できるのは RANGE および LIST パーティションのみです。HASH および KEY パーティションはサブパーティション化できません。

現在のところ、SUBPARTITION BY KEY にはサブパーティショニングカラムを明示的に指定する必要がありますが、PARTITION BY KEY の場合は省略できます (その場合、テーブルの主キーカラムがデフォルトで使用されます)、次のステートメントによって作成されたテーブルがあるとします。

```
CREATE TABLE ts (
    id INT NOT NULL AUTO_INCREMENT PRIMARY KEY,
    name VARCHAR(30)
);
```

次のようなステートメントを使用することで、KEY によってパーティション化された、同じカラムを持つテーブルを作成できます。

```
CREATE TABLE ts (
    id INT NOT NULL AUTO_INCREMENT PRIMARY KEY,
    name VARCHAR(30)
)
PARTITION BY KEY()
PARTITIONS 4;
```

前のステートメントは、次のように記述されているかのように扱われます (テーブルの主キーカラムがパーティショニングカラムとして使用されます)。

```
CREATE TABLE ts (
    id INT NOT NULL AUTO_INCREMENT PRIMARY KEY,
    name VARCHAR(30)
)
PARTITION BY KEY(id)
PARTITIONS 4;
```

ただし、次のステートメントは、デフォルトカラムをサブパーティショニングカラムとして使用するサブパーティション化されたテーブルを作成しようとするため失敗します。このステートメントが成功するには次のようにカラムを指定する必要があります。

```
mysql> CREATE TABLE ts (
    ->     id INT NOT NULL AUTO_INCREMENT PRIMARY KEY,
    ->     name VARCHAR(30)
    -> )
    -> PARTITION BY RANGE(id)
    -> SUBPARTITION BY KEY()
    -> SUBPARTITIONS 4
    -> (
    ->     PARTITION p0 VALUES LESS THAN (100),
    ->     PARTITION p1 VALUES LESS THAN (MAXVALUE)
    -> );
ERROR 1064 (42000): You have an error in your SQL syntax; check the manual that
corresponds to your MySQL server version for the right syntax to use near ')

mysql> CREATE TABLE ts (
    ->     id INT NOT NULL AUTO_INCREMENT PRIMARY KEY,
    ->     name VARCHAR(30)
    -> )
    -> PARTITION BY RANGE(id)
    -> SUBPARTITION BY KEY(id)
    -> SUBPARTITIONS 4
    -> (
    ->     PARTITION p0 VALUES LESS THAN (100),
    ->     PARTITION p1 VALUES LESS THAN (MAXVALUE)
    -> );
Query OK, 0 rows affected (0.07 sec)
```

これは既知の問題です (Bug #51470 を参照してください)。

DELAYED オプションがサポートされない　INSERT DELAYED を使用してパーティション化されたテーブルに行を挿入することはサポートされません。これを試みると エラーで失敗します。

DATA DIRECTORY および INDEX DIRECTORY オプション　DATA DIRECTORY および INDEX DIRECTORY は、パーティション化されたテーブルで使用するときに次の制限が適用されます。

- テーブルレベル DATA DIRECTORY および INDEX DIRECTORY オプションは無視されます (Bug #32091 を参照してください)。
- Windows では、DATA DIRECTORY および INDEX DIRECTORY オプションは、MyISAM テーブルの個々のパーティションまたはサブパーティションでサポートされません (Bug #30459)。ただし、InnoDB テーブルの個々のパーティションまたはサブパーティションには DATA DIRECTORY を使用できます。

パーティション化されたテーブルを修復および再構築する　ステートメント CHECK TABLE、OPTIMIZE TABLE、ANALYZE TABLE、および REPAIR TABLE がパーティション化されたテーブルでサポートされます。

また、ALTER TABLE ... REBUILD PARTITION を使用することで、パーティション化されたテーブルの 1 つ以上のパーティションを再構築できます。ALTER TABLE ... REORGANIZE PARTITION でもパーティションが再構築されます。これら 2 つのステートメントの詳細については、セクション13.1.7「ALTER TABLE 構文」を参照してください。

mysqlcheck、myisamchk、および myisampack はパーティション化されたテーブルでサポートされません。

FOR EXPORT オプション (FLUSH TABLES)　FLUSH TABLES ステートメントの FOR EXPORT オプションは、MySQL 5.6.16 以前のパーティション化された InnoDB テーブルでサポートされません (Bug #16943907)。

19.6.1 パーティショニングキー、主キー、および一意キー

このセクションでは、パーティショニングキーと主キーおよび一意キーとの関係について説明します。この関係を制御するルールは次のように表現できます。パーティション化されたテーブルのパーティショニング式で使用されるすべてのカラムは、テーブルが持つことができるすべての一意キーの一部である必要があります。

つまり、テーブルのすべての一意キーは、テーブルのパーティショニング式内のすべてのカラムを使用する必要があります(これには、テーブルの主キーも含まれます (自明で一意キーであるため)。この点については、このセクションで後述します)。たとえば、次の各テーブル作成ステートメントは無効です。

```
CREATE TABLE t1 (
    col1 INT NOT NULL,
    col2 DATE NOT NULL,
    col3 INT NOT NULL,
```

```
    col4 INT NOT NULL,
    UNIQUE KEY (col1, col2)
)
PARTITION BY HASH(col3)
PARTITIONS 4;

CREATE TABLE t2 (
    col1 INT NOT NULL,
    col2 DATE NOT NULL,
    col3 INT NOT NULL,
    col4 INT NOT NULL,
    UNIQUE KEY (col1),
    UNIQUE KEY (col3)
)
PARTITION BY HASH(col1 + col3)
PARTITIONS 4;
```

どちらの場合も、記述されたテーブルには、パーティショニング式に使用されているすべてのカラムを含んでいない一意キーが少なくとも 1 つあります。

次の各ステートメントは有効で、対応する無効なテーブル作成ステートメントを機能させる 1 つの方法を示しています。

```
CREATE TABLE t1 (
    col1 INT NOT NULL,
    col2 DATE NOT NULL,
    col3 INT NOT NULL,
    col4 INT NOT NULL,
    UNIQUE KEY (col1, col2, col3)
)
PARTITION BY HASH(col3)
PARTITIONS 4;

CREATE TABLE t2 (
    col1 INT NOT NULL,
    col2 DATE NOT NULL,
    col3 INT NOT NULL,
    col4 INT NOT NULL,
    UNIQUE KEY (col1, col3)
)
PARTITION BY HASH(col1 + col3)
PARTITIONS 4;
```

次の例は、そのような場合に生成されるエラーを示しています。

```
mysql> CREATE TABLE t3 (
    ->     col1 INT NOT NULL,
    ->     col2 DATE NOT NULL,
    ->     col3 INT NOT NULL,
    ->     col4 INT NOT NULL,
    ->     UNIQUE KEY (col1, col2),
    ->     UNIQUE KEY (col3)
    -> )
    -> PARTITION BY HASH(col1 + col3)
    -> PARTITIONS 4;
ERROR 1491 (HY000): A PRIMARY KEY must include all columns in the table's partitioning function
```

この CREATE TABLE ステートメントは、指定されたパーティショニングキーに col1 および col3 が含まれているけれども、これらのカラムのいずれもテーブルの両方の一意キーの一部でないために、失敗します。無効なテーブル定義の考えられる解決策の 1 つを次に示します。

```
mysql> CREATE TABLE t3 (
    ->     col1 INT NOT NULL,
    ->     col2 DATE NOT NULL,
    ->     col3 INT NOT NULL,
    ->     col4 INT NOT NULL,
    ->     UNIQUE KEY (col1, col2, col3),
    ->     UNIQUE KEY (col3)
    -> )
    -> PARTITION BY HASH(col3)
    -> PARTITIONS 4;
Query OK, 0 rows affected (0.05 sec)
```

この場合、指定されたパーティショニングキー col3 は両方の一意キーの一部であるため、このテーブル作成ステートメントは成功します。

次のテーブルは、両方の一意キーに属するカラムをパーティショニングキーに含めることができないため、パーティション化できません。

```
CREATE TABLE t4 (
    col1 INT NOT NULL,
    col2 INT NOT NULL,
    col3 INT NOT NULL,
    col4 INT NOT NULL,
    UNIQUE KEY (col1, col3),
    UNIQUE KEY (col2, col4)
);
```

すべての主キーは自明で一意キーであるため、この制約にはテーブルの主キーも含まれます (ある場合)。たとえば、次の 2 つのステートメントは無効です。

```
CREATE TABLE t5 (
    col1 INT NOT NULL,
    col2 DATE NOT NULL,
    col3 INT NOT NULL,
    col4 INT NOT NULL,
    PRIMARY KEY(col1, col2)
)
PARTITION BY HASH(col3)
PARTITIONS 4;

CREATE TABLE t6 (
    col1 INT NOT NULL,
    col2 DATE NOT NULL,
    col3 INT NOT NULL,
    col4 INT NOT NULL,
    PRIMARY KEY(col1, col3),
    UNIQUE KEY(col2)
)
PARTITION BY HASH( YEAR(col2) )
PARTITIONS 4;
```

どちらの場合も、パーティショニング式で参照されるすべてのカラムが主キーに含まれていません。ただし、次の 2 つのステートメントは両方とも有効です。

```
CREATE TABLE t7 (
    col1 INT NOT NULL,
    col2 DATE NOT NULL,
    col3 INT NOT NULL,
    col4 INT NOT NULL,
    PRIMARY KEY(col1, col2)
)
PARTITION BY HASH(col1 + YEAR(col2))
PARTITIONS 4;

CREATE TABLE t8 (
    col1 INT NOT NULL,
    col2 DATE NOT NULL,
    col3 INT NOT NULL,
    col4 INT NOT NULL,
    PRIMARY KEY(col1, col2, col4),
    UNIQUE KEY(col2, col1)
)
PARTITION BY HASH(col1 + YEAR(col2))
PARTITIONS 4;
```

テーブルに一意キーがない場合 (主キーがない場合を含む) はこの制約は適用されず、カラム型がパーティショニングタイプと互換性があるかぎり、パーティショニング式に任意のカラムを使用できます。

同じ理由で、テーブルのパーティショニング式で使用されるすべてのカラムが一意キーに含まれている場合を除き、パーティション化されたテーブルにあとから一意キーを追加することはできません。次のように作成されたパーティション化されたテーブルがあるとします。

```
mysql> CREATE TABLE t_no_pk (c1 INT, c2 INT)
    ->     PARTITION BY RANGE(c1) (
    ->         PARTITION p0 VALUES LESS THAN (10),
    ->         PARTITION p1 VALUES LESS THAN (20),
    ->         PARTITION p2 VALUES LESS THAN (30),
    ->         PARTITION p3 VALUES LESS THAN (40)
    ->     );
Query OK, 0 rows affected (0.12 sec)
```

次のいずれかの ALTER TABLE ステートメントを使用することで、t_no_pk に主キーを追加できます。

```
#  possible PK
mysql> ALTER TABLE t_no_pk ADD PRIMARY KEY(c1);
Query OK, 0 rows affected (0.13 sec)
```

```
Records: 0  Duplicates: 0  Warnings: 0

# drop this PK
mysql> ALTER TABLE t_no_pk DROP PRIMARY KEY;
Query OK, 0 rows affected (0.10 sec)
Records: 0  Duplicates: 0  Warnings: 0

#  use another possible PK
mysql> ALTER TABLE t_no_pk ADD PRIMARY KEY(c1, c2);
Query OK, 0 rows affected (0.12 sec)
Records: 0  Duplicates: 0  Warnings: 0

# drop this PK
mysql> ALTER TABLE t_no_pk DROP PRIMARY KEY;
Query OK, 0 rows affected (0.09 sec)
Records: 0  Duplicates: 0  Warnings: 0
```

ただし、次のステートメントは失敗します。c1 は、パーティショニングキーの一部ですが、指定された主キーの一部ではないためです。

```
#  fails with error 1503
mysql> ALTER TABLE t_no_pk ADD PRIMARY KEY(c2);
ERROR 1503 (HY000): A PRIMARY KEY must include all columns in the table's partitioning function
```

t_no_pk のパーティショニング式は c1 のみであるため、c2 のみに一意キーを追加しようとすると失敗します。ただし、c1 および c2 の両方を使用する一意キーは追加できます。

これらのルールは、既存のパーティション化されていないテーブルを ALTER TABLE ... PARTITION BY を使用してパーティション化するときにも適用されます。次のように作成されたテーブル np_pk があるとします。

```
mysql> CREATE TABLE np_pk (
    ->     id INT NOT NULL AUTO_INCREMENT,
    ->     name VARCHAR(50),
    ->     added DATE,
    ->     PRIMARY KEY (id)
    -> );
Query OK, 0 rows affected (0.08 sec)
```

次の ALTER TABLE ステートメントは、added カラムがテーブルの一意キーの一部でないため、エラーで失敗します。

```
mysql> ALTER TABLE np_pk
    ->     PARTITION BY HASH( TO_DAYS(added) )
    ->     PARTITIONS 4;
ERROR 1503 (HY000): A PRIMARY KEY must include all columns in the table's partitioning function
```

ただし、次に示すように、パーティショニングカラムに id カラムを使用する次のステートメントは有効です。

```
mysql> ALTER TABLE np_pk
    ->     PARTITION BY HASH(id)
    ->     PARTITIONS 4;
Query OK, 0 rows affected (0.11 sec)
Records: 0  Duplicates: 0  Warnings: 0
```

np_pk の場合、パーティショニング式の一部として使用できる唯一のカラムは id です。このテーブルをパーティショニング式でほかのカラムを使用してパーティション化する場合は、目的のカラムを主キーに追加するか、主キーをすべて削除することによって、まずテーブルを変更する必要があります。

19.6.2 ストレージエンジンに関連するパーティショニング制限

次の制限は、テーブルのユーザー定義パーティショニングでストレージエンジンを使用する際に適用されます。

MERGE ストレージエンジン　ユーザー定義パーティショニングと MERGE ストレージエンジンは互換性を持ちません。MERGE ストレージエンジンを使用するテーブルはパーティション化できません。パーティション化されたテーブルはマージできません。

FEDERATED ストレージエンジン　FEDERATED テーブルのパーティショニングはサポートされません。パーティション化された FEDERATED テーブルを作成することはできません。

CSV ストレージエンジン　CSV ストレージエンジンを使用するパーティション化されたテーブルはサポートされません。パーティション化された CSV テーブルを作成することはできません。

InnoDB ストレージエンジン　InnoDB 外部キーと MySQL パーティショニングは互換性を持ちません。パーティション化された InnoDB テーブルは、外部キー参照を持つことができず、外部キーによって参照されるカラムを

持つこともできません。外部キーを持っていたり、外部キーによって参照されたりする InnoDB テーブルはパーティション化できません。

また、ALTER TABLE ... OPTIMIZE PARTITION は、InnoDB ストレージエンジンを使用するパーティション化されたテーブルで正しく動作しません。そのようなテーブルの場合は、代わりに ALTER TABLE ... REBUILD PARTITION および ALTER TABLE ... ANALYZE PARTITION を使用してください。詳細は、セクション 13.1.7.1「ALTER TABLE パーティション操作」を参照してください。

NDB ストレージエンジン (MySQL Cluster)　KEY (LINEAR KEY を含む) によるパーティショニングは、NDB ストレージエンジンでサポートされる唯一のパーティショニングタイプです。MySQL Cluster NDB 7.3 では、[LINEAR] KEY 以外のパーティショニングタイプを使用して MySQL Cluster テーブルを作成することはできず、それを試みるとエラーで失敗します。

また、NDB テーブルに定義できる最大パーティション数は、クラスター内のデータノードとノードグループの数、使用している MySQL Cluster ソフトウェアのバージョン、およびその他の要因に依存します。詳細は、NDB とユーザー定義のパーティション化を参照してください。

ユーザーパーティション化された NDB テーブルが次の 2 つの要件のいずれかまたは両方を満たさなくなる CREATE TABLE ステートメントおよび ALTER TABLE ステートメントは許可されず、エラーで失敗します。

1. テーブルに明示的な主キーが存在する必要があります。
2. テーブルのパーティショニング式に指定されたすべてのカラムが主キーの一部である必要があります。

例外　ユーザーパーティション化された NDB テーブルが空のカラムリストを使用して (つまり、PARTITION BY KEY() または PARTITION BY LINEAR KEY() を使用して) 作成された場合、明示的な主キーは必要ありません。

パーティション化されたテーブルをアップグレードする　アップグレードを実行するときに、KEY によってパーティション化され、NDB 以外のストレージエンジンを使用するテーブルは、ダンプしてリロードする必要があります。

すべてのパーティションに同じストレージエンジン　パーティション化されたテーブルのすべてのパーティションは、同じストレージエンジンを使用する必要があり、それがテーブルが全体として使用する同じストレージエンジンである必要があります。また、テーブルレベルでエンジンを指定しない場合は、パーティション化されたテーブルを作成または変更するときに次のいずれかを行う必要があります。

- 任意のパーティションまたはサブパーティションにエンジンを指定しない
- すべてのパーティションまたはサブパーティションにエンジンを指定する

19.6.3 関数に関連するパーティショニング制限

このセクションでは特に、パーティショニング式で使用される関数に関連する、MySQL パーティショニングの制限について説明します。

次の表に示されている MySQL 関数のみがパーティショニング式で許可されます。

ABS()	CEILING() (CEILING() および FLOOR()を参照してください)	DAY()
DAYOFMONTH()	DAYOFWEEK()	DAYOFYEAR()
DATEDIFF()	EXTRACT() (WEEK 指定子付きの EXTRACT() 関数を参照してください)	FLOOR() (CEILING() および FLOOR()を参照してください)
HOUR()	MICROSECOND()	MINUTE()
MOD()	MONTH()	QUARTER()
SECOND()	TIME_TO_SEC()	TO_DAYS()
TO_SECONDS()	UNIX_TIMESTAMP() (TIMESTAMP カラムで、MySQL 5.6.1 から許可され、MySQL 5.6.3 から完全にサポートされています)	WEEKDAY()
YEAR()		YEARWEEK()

MySQL 5.6 では、TO_DAYS()、TO_SECONDS()、および YEAR() 関数で範囲最適化を使用できます。また、MySQL 5.6.3 以降では、UNIX_TIMESTAMP() はパーティショニング式で単調として扱われます。詳細は、セクション19.4「パーティションプルーニング」を参照してください。

CEILING() および FLOOR()　これらの各関数は、正確な数値型 (INT 型または DECIMAL 型のいずれかなど) の引数を渡された場合にのみ整数を返します。これはたとえば、次の CREATE TABLE ステートメントがここで示すように エラーで失敗することを意味します。

```
mysql> CREATE TABLE t (c FLOAT) PARTITION BY LIST( FLOOR(c) )(
    ->     PARTITION p0 VALUES IN (1,3,5),
    ->     PARTITION p1 VALUES IN (2,4,6)
    -> );
ERROR 1490 (HY000): The PARTITION function returns the wrong type
```

WEEK 指定子付きの EXTRACT() 関数　EXTRACT() 関数によって返される値は、EXTRACT(WEEK FROM col) として使用されるときに、default_week_format システム変数の値に依存します。このため、MySQL 5.6.2 以降では、EXTRACT() は、単位を WEEK として指定するときに、パーティショニング関数として許可されなくなりました (Bug #54483)。

これらの関数の戻り型についての詳細は、セクション12.6.2「数学関数」およびセクション11.2「数値型」を参照してください。

19.6.4 パーティショニングとロック

MySQL 5.6.5 以前では、DML または DDL ステートメントを実行するときにテーブルレベルロックを実際に実行する MyISAM などのストレージエンジンの場合、パーティション化されたテーブルに影響するそのようなステートメントがテーブルに全体としてロックを適用していました。つまり、ステートメントが完了するまですべてのパーティションがロックされました。MySQL 5.6.6 はパーティションロックプルーニングを実装し、これによって多くの場合に不必要なロックが排除されます。MySQL 5.6.6 以降では、パーティション化された MyISAM テーブルに対して読み取りまたは更新を行うほとんどのステートメントで、影響を受けるパーティションのみがロックされます。たとえば、MySQL 5.6.6 より前は、パーティション化 MyISAM テーブルからの SELECT でテーブル全体がロックされました。MySQL 5.6.6 以降は、SELECT ステートメントの WHERE 条件を満たす行を実際に含むパーティションのみがロックされます。これには、パーティション化された MyISAM テーブルに対する同時操作の速度および効率を向上させる効果があります。この改善は、多く (32 以上) のパーティションを持つ MyISAM テーブルを操作するときに特に顕著になります。

この動作の変更は、行レベルロックを使用し、パーティションプルーニングの前にロックを実際に実行しない (または実行する必要がない)、InnoDB などのストレージエンジンを使用するパーティション化されたテーブルに影響するステートメントには影響しません。

次のいくつかの段落では、テーブルレベルロックを使用するストレージエンジンを使用するテーブルに対する、さまざまな MySQL ステートメントへのパーティションロックプルーニングの影響について説明します。

DML ステートメントへの影響

SELECT ステートメント (union または join を含むものを含む) が、実際に読み取る必要があるパーティションのみをロックするようになります。これは SELECT ... PARTITION にも適用されます。

UPDATE は、パーティショニングカラムが更新されないテーブルにのみロックをプルーニングします。

REPLACE および INSERT は、挿入または交換される行を持つパーティションのみをロックするようになります。ただし、AUTO_INCREMENT 値がパーティショニングカラム用に生成される場合は、すべてのパーティションがロックされます。

INSERT ... ON DUPLICATE KEY UPDATE は、パーティショニングカラムが更新されないかぎりプルーニングされます。

INSERT ... SELECT は、読み取る必要があるソーステーブル内のパーティションのみをロックするようになります (ターゲットテーブル内のすべてのパーティションはロックされます)。

> **注記**
>
> INSERT DELAYED はパーティション化されたテーブルではサポートされません。

LOAD DATA ステートメントによってパーティション化されたテーブルに適用されるロックはプルーニングできません。

パーティション化されたテーブルのパーティショニングカラムを使用する BEFORE INSERT または BEFORE UPDATE トリガーが存在する場合は、このテーブルを更新する INSERT および UPDATE ステートメントでのロックをプルーニングできないことを意味します (トリガーがその値を変更する可能性があるため)。テーブルのパーティショニングカラムに対する BEFORE INSERT トリガーが存在する場合は、INSERT または REPLACE に

よって設定されるロックをプルーニングできないことを意味します (BEFORE INSERT トリガーが、行が挿入される前に行のパーティショニングカラムを変更し、それによって行が本来のものとは異なるパーティションに強制的に挿入される可能性があるため)。パーティショニングカラムでの BEFORE UPDATE トリガーは、UPDATE または INSERT ... ON DUPLICATE KEY UPDATE によって適用されるロックをプルーニングできないことを意味します。

影響を受ける DDL ステートメント

CREATE VIEW でロックが行われなくなります。

ALTER TABLE ... EXCHANGE PARTITION がロックをプルーニングするようになります。交換されるテーブルおよび交換されるパーティションのみがロックされます。

ALTER TABLE ... TRUNCATE PARTITION がロックがプルーニングするようになります。空にされるパーティションのみがロックされます。

ALTER TABLE ステートメントは引き続きテーブルレベルでメタデータロックを行います。

その他のステートメント

LOCK TABLES はパーティションロックをプルーニングできません。

CALL stored_procedure(expr) はロックプルーニングをサポートしますが、expr を評価する際はしません。

DO および SET ステートメントはパーティショニングロックプルーニングをサポートしません。

第 20 章 ストアドプログラムおよびビュー

目次

20.1 ストアドプログラムの定義 2474
20.2 ストアドルーチン (プロシージャーと関数) の使用 2474
20.2.1 ストアドルーチンの構文 2475
20.2.2 ストアドルーチンと MySQL 権限 2476
20.2.3 ストアドルーチンのメタデータ 2476
20.2.4 ストアドプロシージャー、関数、トリガー、および LAST_INSERT_ID() 2476
20.3 トリガーの使用 2477
20.3.1 トリガーの構文と例 2477
20.3.2 トリガーのメタデータ 2480
20.4 イベントスケジューラの使用 2481
20.4.1 イベントスケジューラの概要 2481
20.4.2 イベントスケジューラの構成 2482
20.4.3 イベント構文 2484
20.4.4 イベントメタデータ 2484
20.4.5 イベントスケジューラのステータス 2485
20.4.6 イベントスケジューラと MySQL 権限 2485
20.5 ビューの使用 2488
20.5.1 ビューの構文 2488
20.5.2 ビュー処理アルゴリズム 2488
20.5.3 更新可能および挿入可能なビュー 2489
20.5.4 ビューのメタデータ 2491
20.6 ストアドプログラムおよびビューのアクセスコントロール 2491
20.7 ストアドプログラムのバイナリロギング 2493

この章では、あとから実行するためにサーバーに格納された SQL コードの点で定義されたデータベースオブジェクトである、ストアドプログラムとビューについて説明します。

ストアドプログラムには次のオブジェクトが含まれます。

- ストアドルーチン、つまりストアドプロシージャーとストアドファンクション。ストアドプロシージャーは、CALL ステートメントを使用して呼び出されます。プロシージャーは、戻り値がありませんが、呼び出し元があとから検査できるようにそのパラメータを変更できます。また、クライアントプログラムに戻される結果セットも生成できます。ストアドファンクションは、組み込み関数と同様に使用されます。式で呼び出すと、式評価中に値を返します。
- トリガー。トリガーは、テーブルに関連付けられ、そのテーブルに対して挿入や更新などの特定のイベントが起きたときにアクティブ化される、名前付きのデータベースオブジェクトです。
- イベント。イベントとは、スケジュールに従ってサーバーが実行するタスクです。

ビューは、参照されたときに結果セットを生成するストアドクエリーです。ビューは仮想テーブルとして機能します。

この章では、ストアドプログラムおよびビューを使用する方法について説明します。次のセクションで、これらのオブジェクトに関連したステートメントの SQL 構文に関する追加情報を示します。

- オブジェクト型ごとに、どのオブジェクトが存在し、どのように定義されているかを制御する CREATE、ALTER、および DROP ステートメントがあります。セクション13.1「データ定義ステートメント」を参照してください。
- CALL ステートメントは、ストアドプロシージャーの呼び出しに使用されます。セクション13.2.1「CALL 構文」を参照してください。
- ストアドプログラム定義には、複合ステートメント、ループ、条件文、および宣言された変数を使用できる本体が含まれます。セクション13.6「MySQL 複合ステートメント構文」を参照してください。

MySQL 5.6.6 以降では、ストアドプログラムによって参照されるオブジェクトへのメタデータ変更が検出され、そのプログラムが次に実行されときに、影響を受けるステートメントの自動再解析が行われるようになります。

詳細については、セクション8.9.4「プリペアドステートメントおよびストアドプログラムのキャッシュ」を参照してください。

20.1 ストアドプログラムの定義

各ストアドプログラムには、SQL ステートメントから構成される本体が含まれます。このステートメントは、セミコロン (;) 文字で区切られた複数のステートメントから構成される複合ステートメントの場合があります。たとえば、次のストアドプロシージャーには、SET ステートメントと REPEAT ループ (ループ自体に別の SET ステートメントが含まれます) を含む BEGIN ... END ブロックから構成される本体があります。

```
CREATE PROCEDURE dorepeat(p1 INT)
BEGIN
  SET @x = 0;
  REPEAT SET @x = @x + 1; UNTIL @x > p1 END REPEAT;
END;
```

mysql クライアントプログラムを使用してセミコロン文字を含むストアドプログラムを定義すると、問題が発生します。デフォルトでは、mysql 自体はセミコロンをステートメント区切り文字と認識します。したがって、mysql がストアドプログラム定義全体をサーバーに渡すように、区切り文字を一時的に再定義する必要があります。

mysql の区切り文字を再定義するには、delimiter コマンドを使用します。次の例は、上記の dorepeat() プロシージャーについてこれを行う方法を示しています。区切り文字は // に変更され、定義全体を単一のステートメントとしてサーバーに渡して、プロシージャーの呼び出し前に ; にリストアできます。これにより、プロシージャー本体で使用される ; 区切り文字を、mysql 自体が解釈するのではなく、サーバーに渡すようにすることができます。

```
mysql> delimiter //

mysql> CREATE PROCEDURE dorepeat(p1 INT)
    -> BEGIN
    ->   SET @x = 0;
    ->   REPEAT SET @x = @x + 1; UNTIL @x > p1 END REPEAT;
    -> END
    -> //
Query OK, 0 rows affected (0.00 sec)

mysql> delimiter ;

mysql> CALL dorepeat(1000);
Query OK, 0 rows affected (0.00 sec)

mysql> SELECT @x;
+------+
| @x   |
+------+
| 1001 |
+------+
1 row in set (0.00 sec)
```

区切り文字を // 以外の文字列に再定義でき、区切り文字は単一の文字から構成することも、複数の文字から構成することもできます。バックスラッシュ (「\」) 文字は、MySQL のエスケープ文字なので使用しないでください。

次に、パラメータを受け取り、SQL 関数を使用して操作を実行したあと、結果を返す関数例を示します。この場合は、関数定義に内部の ; ステートメント区切り文字が含まれていないため、delimiter を使用する必要はありません。

```
mysql> CREATE FUNCTION hello (s CHAR(20))
mysql> RETURNS CHAR(50) DETERMINISTIC
    -> RETURN CONCAT('Hello, ',s,'!');
Query OK, 0 rows affected (0.00 sec)

mysql> SELECT hello('world');
+----------------+
| hello('world') |
+----------------+
| Hello, world!  |
+----------------+
1 row in set (0.00 sec)
```

20.2 ストアドルーチン (プロシージャーと関数) の使用

ストアドルーチン (プロシージャーおよび関数) は MySQL 5.6 でサポートされています。ストアドルーチンとは、サーバーに格納できる一連の SQL ステートメントです。これが行われていると、クライアントは個々のステートメントを繰り返し発行し続ける必要はなく、代わりにストアドルーチンを参照できます。

ストアドルーチンには、mysql データベース内の proc テーブルが必要です。このテーブルは、MySQL 5.6 インストール手順中に作成されます。以前のバージョンから MySQL 5.6 にアップグレードしている場合、必ず付与テーブルを更新して、proc テーブルの存在を確認してください。セクション4.4.7「mysql_upgrade — MySQL テーブルのチェックとアップグレード」を参照してください。

ストアドルーチンは特に、次のような特定の状況で役立ちます。

- クライアントアプリケーションが異なる言語で作成されているか、異なるプラットフォームで動作しているが、同じデータベース操作を実行する必要がある場合。
- セキュリティーが最重要である場合。たとえば、銀行では、すべての一般的な操作に対してストアドプロシージャーおよびストアドファンクションを使用します。これにより一貫したセキュアな環境が得られ、ルーチンによってそれぞれの操作が正しく記録されるようになります。このようなセットアップでは、アプリケーションおよびユーザーはデータベーステーブルに直接アクセスできませんが、特定のストアドルーチンだけを実行できます。

ストアドルーチンは、サーバーとクライアント間で送信する必要のある情報が少なくなるので、パフォーマンスを改善できます。そのトレードオフでは、これによりサーバー側で行われる作業が増え、クライアント (アプリケーション) 側で行われる作業が少なくなるので、データベースサーバーでの口ードが増大します。1 台または少数のデータベースサーバーだけで多数のクライアントマシン (Web サーバーなど) にサービスを提供している場合にはこれを検討してください。

ストアドルーチンを使用すれば、データベースサーバーで関数のライブラリを保持することもできます。これは、内部的に (たとえばクラスを使用して) このような設計を可能にする、現代のアプリケーション言語で共有されている機能です。これらのクライアントアプリケーションの言語機能を使用すると、データベース使用のスコープ外でもプログラマにとって利点があります。

MySQL はストアドルーチンについて SQL:2003 構文に従っており、これは IBM の DB2 でも使用されています。ここで説明するすべての構文はサポートされており、すべての制限と拡張が適宜ドキュメント化されています。

追加のリソース

- ストアドプロシージャーおよびストアドファンクションを扱うときに、ストアドプロシージャーのユーザーフォーラムが役立ちます。
- MySQL のストアドルーチンに関するよくある質問とその回答については、セクションA.4「MySQL 5.6 FAQ: ストアドプロシージャーおよびストアドファンクション」を参照してください。
- ストアドルーチンの使用にはいくつかの制限があります。セクションD.1「ストアドプログラムの制約」を参照してください。
- ストアドルーチンのバイナリロギングは、セクション20.7「ストアドプログラムのバイナリロギング」で説明しているように行われます。

20.2.1 ストアドルーチンの構文

ストアドルーチンはプロシージャーまたは関数のどちらかです。ストアドルーチンは、CREATE PROCEDURE および CREATE FUNCTION ステートメントで作成されます (セクション13.1.15「CREATE PROCEDURE および CREATE FUNCTION 構文」を参照してください)。プロシージャーは CALL ステートメントを使用して呼び出され (セクション13.2.1「CALL 構文」を参照してください)、出力変数の使用でのみ値を戻すことができます。関数は、ほかの関数とまったく同様に (つまり、関数の名前を呼び出すことによって) ステートメント内部から呼び出すことができ、スカラー値を戻すことができます。ストアドルーチンの本体では、複合ステートメントを使用できます (セクション13.6「MySQL 複合ステートメント構文」を参照してください)。

ストアドルーチンは、DROP PROCEDURE および DROP FUNCTION ステートメントで削除でき (セクション13.1.26「DROP PROCEDURE および DROP FUNCTION 構文」を参照してください)、ALTER PROCEDURE および ALTER FUNCTION ステートメントで変更できます (セクション13.1.5「ALTER PROCEDURE 構文」を参照してください)。

ストアドプロシージャーまたはストアドファンクションは、特定のデータベースに関連付けられています。これにはいくつかの問題があります。

- ルーチンが呼び出されると、暗黙の USE db_name が実行されます (その後、ルーチンが終了すると元に戻ります)。ストアドルーチン内での USE ステートメントは許可されていません。
- データベース名でルーチン名を修飾できます。これは現在のデータベースに存在しないルーチンを参照する場合に使用できます。たとえば、testデータベースに関連するストアドプロシージャー p またはストアドファンクション f を呼び出すには、CALL test.p() または test.f() と指定します。
- データベースを削除すると、そのデータベースに関連付けられたすべてのストアドルーチンも削除されます。

ストアドファンクションは再帰関数にはできません。

ストアドプロシージャーでの再帰は許可されていますが、デフォルトでは無効になっています。再帰を有効にするには、max_sp_recursion_depth サーバーシステム変数を正の値に設定します。ストアドプロシージャーの再帰により、スレッドスタック領域の要求が増加します。max_sp_recursion_depth の値を増やした場合、サーバー起動時に thread_stack の値を増やすことによってスレッドスタックサイズを増やすことが必要な場合もあります。詳細は、セクション5.1.4「サーバーシステム変数」を参照してください。

MySQL では、通常の SELECT ステートメントをストアドプロシージャー内で (つまり、カーソルまたはローカル変数を使用せずに) 使用できるようにする非常に役立つ拡張をサポートしています。このようなクエリーの結果セットは単にクライアントに直接送信されます。複数の SELECT ステートメントは複数の結果セットを生成するので、クライアントは複数の結果セットをサポートしている MySQL クライアントライブラリを使用する必要があります。これは、クライアントが、4.1 以降の MySQL のバージョンからクライアントライブラリを使用する必要があることを意味します。クライアントは、接続するときに、CLIENT_MULTI_RESULTS オプションも指定する必要があります。C プログラムの場合、これは、mysql_real_connect()C API 関数で実行できます。セクション23.7.7.53「mysql_real_connect()」およびセクション23.7.17「複数ステートメント実行の C API サポート」を参照してください。

20.2.2 ストアドルーチンと MySQL 権限

MySQL 許可システムはストアドルーチンを次のように考慮します。

- ストアドルーチンを生成するには、CREATE ROUTINE 権限が必要です。
- ストアドルーチンを変更または削除するには、ALTER ROUTINE 権限が必要です。この権限は、必要に応じて、ルーチンの作成者に自動的に与えられ、ルーチンが削除されると作成者から削除されます。
- ストアドルーチンを実行するには、EXECUTE 権限が必要です。ただし、この権限は、必要に応じて、ルーチンの作成者に自動的に与えられます (ルーチンが削除されると作成者から削除されます)。また、ルーチンのデフォルトの SQL SECURITY 特性は DEFINER であり、これにより、ルーチンが関連付けられているデータベースにアクセス可能なユーザーがルーチンを実行できるようになります。
- automatic_sp_privileges システム変数が 0 である場合、EXECUTE および ALTER ROUTINE 権限は作成者に対して自動的には付与および削除されません。
- ルーチンの作成者は、ルーチンの CREATE ステートメントを実行するために使用されるアカウントです。これは、ルーチン定義で DEFINER として名前が指定されているアカウントと同じでないことがあります。

サーバーは、ストアドルーチンを作成、変更、または削除するステートメントに応じて、mysql.proc テーブルを操作します。このテーブルの手動操作に対するサーバーでの認識はサポートされていません。

20.2.3 ストアドルーチンのメタデータ

ストアドルーチンに関するメタデータは次のように取得できます。

- INFORMATION_SCHEMA データベースの ROUTINES テーブルをクエリーします。セクション21.18「INFORMATION_SCHEMA ROUTINES テーブル」を参照してください。
- SHOW CREATE PROCEDURE および SHOW CREATE FUNCTION ステートメントを使用して、ルーチン定義を表示します。セクション13.7.5.11「SHOW CREATE PROCEDURE 構文」を参照してください。
- SHOW PROCEDURE STATUS および SHOW FUNCTION STATUS ステートメントを使用して、ルーチン特性を表示します。セクション13.7.5.29「SHOW PROCEDURE STATUS 構文」を参照してください。

20.2.4 ストアドプロシージャー、関数、トリガー、および LAST_INSERT_ID()

ストアドルーチン (プロシージャーまたは関数) またはトリガーの本体内では、LAST_INSERT_ID() の値は、このような種類のオブジェクトの本体外で実行されたステートメントと同様に変更されます (セクション12.14「情報

関数」を参照してください)。あとに続くステートメントで参照される LAST_INSERT_ID() の値でのストアドルーチンまたはトリガーの効果は、ルーチンの種類によって異なります。

- ストアドプロシージャーで LAST_INSERT_ID() の値を変更するステートメントが実行される場合は、プロシージャー呼び出しが続くステートメントで変更された値が参照されます。
- 値を変更するストアドファンクションやトリガーでは、値は関数やトリガーが終了したときにリストアされるので、後続のステートメントは変更された値を表示しません。

20.3 トリガーの使用

トリガーとは、テーブルに関連付けられ、そのテーブルに対して特定のイベントが発生するとアクティブ化される名前付きデータベースオブジェクトのことです。トリガーを使用する場合には、テーブルに挿入する値のチェックを実行したり、更新にかかわる値の計算を実行したりする場合があります。

トリガーは、関連付けられたテーブルでステートメントが行の挿入、更新、または削除を行なったときにアクティブ化するように定義されます。これらの行操作がトリガーイベントになります。たとえば、行は、INSERT または LOAD DATA ステートメントで挿入でき、挿入トリガーは挿入された行ごとにアクティブ化します。トリガーは、トリガーイベントの前または後のどちらかでアクティブ化するように設定できます。たとえば、テーブルに挿入される各行の前、または更新される各行のあとでトリガーをアクティブ化させることができます。

> **重要**
>
> MySQL のトリガーは、SQL ステートメントがテーブルに対して行なった変更の場合にのみアクティブ化します。ビューでの変更や、SQL ステートメントを MySQL Server に転送しない API がテーブルに対して行なった変更ではアクティブ化しません。これは次のことを意味します。
>
> - INFORMATION_SCHEMA または performance_schema テーブルは実際にはビューなので、トリガーは、これらのテーブルでの変更ではアクティブ化されません。
> - トリガーは、NDB API を使用して行われた更新によってアクティブ化されません。

次のセクションでは、トリガーを作成および削除するための構文について説明し、使用方法の例をいくつか挙げ、トリガーメタデータを取得する方法を示します。

追加のリソース

- トリガーを扱うときには、トリガーユーザーフォーラムが役立ちます。
- MySQL でのトリガーに関するよくある質問とその回答については、セクションA.5「MySQL 5.6 FAQ: トリガー」を参照してください。
- トリガーの使用にはいくつかの制限があります。セクションD.1「ストアドプログラムの制約」を参照してください。
- トリガーのバイナリロギングは、セクション20.7「ストアドプログラムのバイナリロギング」で説明しているように行います。

20.3.1 トリガーの構文と例

トリガーを作成したり、トリガーを削除したりするには、セクション13.1.19「CREATE TRIGGER 構文」およびセクション13.1.30「DROP TRIGGER 構文」で説明しているように、CREATE TRIGGER または DROP TRIGGER ステートメントを使用します。

次に、INSERT 操作に対してアクティブ化するトリガーをテーブルに関連付ける簡単な例を示します。このトリガーは加算器として機能し、テーブルのいずれかのカラムに挿入された値を合計します。

```
mysql> CREATE TABLE account (acct_num INT, amount DECIMAL(10,2));
Query OK, 0 rows affected (0.03 sec)

mysql> CREATE TRIGGER ins_sum BEFORE INSERT ON account
    -> FOR EACH ROW SET @sum = @sum + NEW.amount;
Query OK, 0 rows affected (0.06 sec)
```

CREATE TRIGGER ステートメントは、account テーブルに関連付けられている ins_sum という名前のトリガーを作成します。トリガーアクションタイム、トリガーイベント、およびトリガーがアクティブ化したときに行う動作を指定する句も含みます。

- キーワード BEFORE は、トリガーアクションタイムを示します。この場合、トリガーは、テーブルに挿入された各行の前にアクティブ化します。ここで許可されている別のキーワードは AFTER です。
- キーワード INSERT は、トリガーイベント、つまりトリガーをアクティブ化する操作の種類を示します。例では、INSERT 操作がトリガーのアクティブ化を引き起こします。DELETE および UPDATE 操作に対するトリガーも作成できます。
- FOR EACH ROW に続くステートメントは、トリガー本体を定義します。これは、トリガーがアクティブ化するたびに実行するステートメントであり、トリガーイベントによって影響される行ごとに一度行われます。この例では、トリガー本体は、amount カラムに挿入された値をユーザー変数に累積する単純な SET です。このステートメントは、「新しい行に挿入される amount カラムの値」を意味する NEW.amount としてカラムを参照します。

トリガーを使用するには、加算器変数をゼロにセットし、INSERT ステートメントを実行して、その後変数がどの値になっているかを確認します。

```
mysql> SET @sum = 0;
mysql> INSERT INTO account VALUES(137,14.98),(141,1937.50),(97,-100.00);
mysql> SELECT @sum AS 'Total amount inserted';
+----------------------+
| Total amount inserted |
+----------------------+
| 1852.48              |
+----------------------+
```

この場合、INSERT ステートメントの実行後の @sum の値は 14.98 + 1937.50 - 100 または 1852.48 です。

トリガーを破棄するには、DROP TRIGGER ステートメントを使用します。トリガーがデフォルトスキーマにない場合、スキーマ名を指定する必要があります。

```
mysql> DROP TRIGGER test.ins_sum;
```

テーブルを削除すると、そのテーブルのトリガーもすべて削除されます。

トリガー名はスキーマの名前空間内に存在します。つまり、すべてのトリガーがスキーマ内で一意の名前を持つ必要があります。異なるスキーマ内のトリガーは同じ名前を持つことができます。

トリガー名はスキーマに対して一意であるという要件以外に、作成できるトリガーの種類に対して別の制限があります。特に、所定のテーブルに、同じトリガーイベントとアクションタイムを持つ複数のトリガーを含めることはできません。たとえば、1 つのテーブルに対して 2 つの BEFORE UPDATE トリガーを定義することはできません。これに対処するために、FOR EACH ROW のあとで BEGIN ... END 複合ステートメント構造構文を使用することにより、複数のステートメントを実行するトリガーを定義できます。(例はこのセクションであとから示します。)

トリガー本体内で、OLD および NEW キーワードを使用すると、トリガーの影響を受ける行のカラムにアクセスできます。OLD および NEW はトリガーに対する MySQL の拡張です。これらは大文字と小文字を区別しません。

INSERT トリガー内では、NEW.col_name だけを使用できます。古い行はありません。DELETE トリガーでは、OLD.col_name だけを使用できます。新しい行はありません。UPDATE トリガーでは、OLD.col_name を使用して、更新される前の行のカラムを参照でき、NEW.col_name を使用して、更新されたあとの行のカラムを参照できます。

OLD で指名されたカラムは読み取り専用です。(それに対する SELECT 権限がある場合) 参照はできますが、変更はできません。NEW で指名されたカラムは、それに対する SELECT 権限がある場合に参照できます。BEFORE トリガーでは、それに対する UPDATE 権限がある場合、SET NEW.col_name = value でその値を変更することもできます。これは、トリガーを使用して、新しい行に挿入する値または行の更新に使用される値を変更できることを意味します。(このような SET ステートメントは、行の変更はすでに行われているため、AFTER トリガーでは効果がありません。)

BEFORE トリガーでは、AUTO_INCREMENT カラムの NEW 値は 0 であり、新しい行が実際に挿入されるときに自動的に生成されるシーケンス番号ではありません。

BEGIN ... END 構造構文を使用することにより、複数のステートメントを実行するトリガーを定義できます。BEGIN ブロック内では、条件文やループなど、ストアドルーチン内で許可されたほかの構文を使用することもできます。ただし、ストアドルーチンの場合と同様に、mysql プログラムを使用して、複数のステートメントを実行するトリガーを定義する場合、トリガー定義内で ; ステートメント区切り文字を使用できるように、mysql ステートメント区切り文字を再定義する必要があります。次の例はこれらの要点を示しています。ここでは、各

行の更新に使用する新しい値をチェックし、0 から 100 の範囲に収まるように値を変更する UPDATE トリガーを定義しています。行の更新に使用される前に値をチェックする必要があるので、これは BEFORE トリガーにする必要があります。

```
mysql> delimiter //
mysql> CREATE TRIGGER upd_check BEFORE UPDATE ON account
    -> FOR EACH ROW
    -> BEGIN
    ->     IF NEW.amount < 0 THEN
    ->         SET NEW.amount = 0;
    ->     ELSEIF NEW.amount > 100 THEN
    ->         SET NEW.amount = 100;
    ->     END IF;
    -> END;//
mysql> delimiter ;
```

ストアドプロシージャーを個別に定義してから、単純な CALL ステートメントを使用してトリガーから呼び出したほうが簡単になる場合があります。これは、複数のトリガー内から同じコードを実行する場合にも便利です。

アクティブ化したときにトリガーが実行するステートメントに表示できる対象には制限があります。

- トリガーは、CALL ステートメントを使用して、データをクライアントに戻すストアドプロシージャーや、ダイナミック SQL を使用するストアドプロシージャーの呼び出しはできません。(ストアドプロシージャーは、OUT または INOUT パラメータを通じてトリガーにデータを返すことが許可されています。)
- トリガーは、START TRANSACTION、COMMIT、ROLLBACK など、トランザクションを明示的または暗黙的に開始したり終了したりするステートメントを使用できません。

セクションD.1「ストアドプログラムの制約」も参照してください。

MySQL は次のようにトリガー実行中にエラーを処理します。

- BEFORE トリガーが失敗した場合、対応する行に対する操作は実行されません。
- BEFORE トリガーは、行を挿入または変更しようとする試行 によってアクティブ化され、その試行がその後成功するかどうかには関係ありません。
- AFTER トリガーは、すべての BEFORE トリガーと行操作の実行が成功した場合にのみ実行されます。
- BEFORE または AFTER トリガーのどちらかの実行中にエラーが発生すると、トリガーの呼び出しを起こしたステートメント全体が失敗します。
- トランザクションテーブルの場合、ステートメントの失敗により、ステートメントが実行したすべての変更がロールバックされます。トリガーの失敗はステートメントの失敗を招くので、トリガーの失敗はロールバックも引き起こします。非トランザクションテーブルの場合、このようなロールバックは行えないので、ステートメントが失敗しても、エラーの時点以前に実行されたすべて変更は有効なままです。

次の例に示す testref という名前のトリガーなど、トリガーには、名前によるテーブルへの直接の参照を含めることができます。

```
CREATE TABLE test1(a1 INT);
CREATE TABLE test2(a2 INT);
CREATE TABLE test3(a3 INT NOT NULL AUTO_INCREMENT PRIMARY KEY);
CREATE TABLE test4(
  a4 INT NOT NULL AUTO_INCREMENT PRIMARY KEY,
  b4 INT DEFAULT 0
);

delimiter |

CREATE TRIGGER testref BEFORE INSERT ON test1
  FOR EACH ROW
  BEGIN
    INSERT INTO test2 SET a2 = NEW.a1;
    DELETE FROM test3 WHERE a3 = NEW.a1;
    UPDATE test4 SET b4 = b4 + 1 WHERE a4 = NEW.a1;
  END;
|

delimiter ;

INSERT INTO test3 (a3) VALUES
  (NULL), (NULL), (NULL), (NULL), (NULL),
  (NULL), (NULL), (NULL), (NULL), (NULL);
```

```
INSERT INTO test4 (a4) VALUES
  (0), (0), (0), (0), (0), (0), (0), (0), (0), (0);
```

次に示すように、テーブル test1 に次の値を挿入するとします。

```
mysql> INSERT INTO test1 VALUES
    -> (1), (3), (1), (7), (1), (8), (4), (4);
Query OK, 8 rows affected (0.01 sec)
Records: 8  Duplicates: 0  Warnings: 0
```

この結果、4 つのテーブルに次のデータが含まれます。

```
mysql> SELECT * FROM test1;
+------+
| a1   |
+------+
|    1 |
|    3 |
|    1 |
|    7 |
|    1 |
|    8 |
|    4 |
|    4 |
+------+
8 rows in set (0.00 sec)

mysql> SELECT * FROM test2;
+------+
| a2   |
+------+
|    1 |
|    3 |
|    1 |
|    7 |
|    1 |
|    8 |
|    4 |
|    4 |
+------+
8 rows in set (0.00 sec)

mysql> SELECT * FROM test3;
+----+
| a3 |
+----+
|  2 |
|  5 |
|  6 |
|  9 |
| 10 |
+----+
5 rows in set (0.00 sec)

mysql> SELECT * FROM test4;
+----+------+
| a4 | b4   |
+----+------+
|  1 |    3 |
|  2 |    0 |
|  3 |    1 |
|  4 |    2 |
|  5 |    0 |
|  6 |    0 |
|  7 |    1 |
|  8 |    1 |
|  9 |    0 |
| 10 |    0 |
+----+------+
10 rows in set (0.00 sec)
```

20.3.2 トリガーのメタデータ

トリガーに関するメタデータは次のように取得できます。

- INFORMATION_SCHEMA データベースの TRIGGERS テーブルをクエリーします。セクション 21.26「INFORMATION_SCHEMA TRIGGERS テーブル」を参照してください。

- SHOW CREATE TRIGGER ステートメントを使用します。セクション13.7.5.13「SHOW CREATE TRIGGER 構文」を参照してください。
- SHOW TRIGGERS ステートメントを使用します。セクション13.7.5.39「SHOW TRIGGERS 構文」を参照してください。

20.4 イベントスケジューラの使用

MySQL イベントスケジューラは、イベント、つまりスケジュールに従って実行するタスクのスケジュール設定および実行を管理します。次の説明では、イベントスケジューラを取り上げ、次のセクションに分かれています。

- セクション20.4.1「イベントスケジューラの概要」では、MySQL イベントの概論と概念的概要を示します。
- セクション20.4.3「イベント構文」では、MySQL イベントを作成、変更、および削除するための SQL ステートメントについて説明します。
- セクション20.4.4「イベントメタデータ」では、イベントに関する情報の取得方法と、MySQL Server でのこの情報の格納方法を示します。
- セクション20.4.6「イベントスケジューラと MySQL 権限」では、イベントを処理するために必要な権限と、実行時に権限に関してイベントが持つ派生問題について説明します。

ストアドルーチンには、mysql データベース内の event テーブルが必要です。このテーブルは、MySQL 5.6 インストール手順中に作成されます。以前のバージョンから MySQL 5.6 にアップグレードしている場合は、必ず付与テーブルを更新して、event テーブルがあるかどうか確認してください。セクション4.4.7「mysql_upgrade — MySQL テーブルのチェックとアップグレード」を参照してください。

追加のリソース

- スケジュール設定済みイベントを扱うときには、「MySQL Event Scheduler User Forum」が役立ちます。
- イベントの使用にはいくつかの制限があります。セクションD.1「ストアドプログラムの制約」を参照してください。
- イベントのバイナリロギングは、セクション20.7「ストアドプログラムのバイナリロギング」で説明しているように行われます。

20.4.1 イベントスケジューラの概要

MySQL イベントはスケジュールに従って実行するタスクです。したがって、これらをスケジュール設定済みイベントと呼ぶことがあります。イベントの作成時には、特定の日時に開始して終了し、1 つ以上の定期的な間隔で実行される 1 つ以上の SQL ステートメントを含んだ、名前付きデータベースオブジェクトを作成します。概念的には、このことは Unix の crontab (「cron ジョブ」とも呼ばれます) や、Windows のタスクスケジューラの考え方に似ています。

この種のスケジュール設定済みのタスクは、「時間トリガー」と呼ばれる場合もあり、これらが時間の経過によってトリガーされるオブジェクトであることを示しています。これは基本的には正しいのですが、セクション20.3「トリガーの使用」で説明している種類のトリガーと混同しないように、イベントの用語を使用します。さらに厳密にいえば、イベントは「時間トリガー」と混同しないようにする必要があります。トリガーは、指定したテーブルで行われるイベントの特定の種類に応じて実行されるステートメントを持つデータベースオブジェクトですが、(スケジュール設定済み) イベントは、指定された時間間隔の経過に応じて実行されるステートメントを持つオブジェクトです。

SQL 標準にはイベントのスケジュール設定への対応はありませんが、ほかのデータベースシステムには先例があり、これらの実装と MySQL Server で見られる実装との間には一定の類似性が認められます。

MySQL イベントには次の主要機能およびプロパティーがあります。

- MySQL 5.6 では、イベントはその名前と、イベントに割り当てられているスキーマによって一意に識別されます。
- イベントは、スケジュールに従って特定のアクションを実行します。このアクションは、SQL ステートメントから構成され、必要に応じて BEGIN ... END ブロック内の複合ステートメントにできます (セクション13.6「MySQL 複合ステートメント構文」を参照してください)。イベントのタイミングは一度だけまたは繰り返しのどちらかです。一度だけのイベントは一度しか実行しません。繰り返しのイベントは、一定の間隔でアクションを繰り返し、イベントを繰り返すためのスケジュールに、特定の開始日時と終了日時の両方または一

方を割り当てるか、どちらも割り当てないことができます。(デフォルトで、繰り返しイベントのスケジュールは作成されるとすぐに開始し、無効または削除されるまで継続します。)

繰り返しイベントがスケジュール間隔内に終了しない場合は、イベントの複数のインスタンスが同時に実行される可能性があります。これが好ましくない場合は、同時インスタンスを回避するためのメカニズムを設けてください。たとえば、GET_LOCK() 関数や、行またはテーブルのロックを使用できます。

- ユーザーは、これらの目的用の SQL ステートメントを使用してスケジュール設定済みイベントを作成、変更、および削除できます。構文が無効なイベント作成および変更ステートメントは失敗し、対応するエラーメッセージが表示されます。ユーザーは、実際には自身が保有していない権限を必要とするステートメントを、イベントのアクションに含めることがあります。イベントの作成または変更ステートメントは成功しますが、イベントのアクションは失敗します。詳細は、セクション20.4.6「イベントスケジューラと MySQL 権限」を参照してください。

- イベントのプロパティーの多くは、SQL ステートメントを使用して設定または変更できます。これらのプロパティーには、イベントの名前、タイミング、持続性 (つまり、そのスケジュールの有効期限が切れたあとも保持されるかどうか)、ステータス (有効または無効)、実行するアクション、および割り当て先のスキーマが含まれます。セクション13.1.2「ALTER EVENT 構文」を参照してください。

  イベントのデフォルトの定義者は、イベントが変更されていない場合は、イベントを作成したユーザーであり、変更されている場合は、定義者はそのイベントに影響する ALTER EVENT ステートメントを最後に発行したユーザーです。イベントが定義されているデータベースに対する EVENT 権限を保有するすべてのユーザーは、そのイベントを変更できます。セクション20.4.6「イベントスケジューラと MySQL 権限」を参照してください。

- イベントのアクションステートメントには、ストアドルーチン内で許可されているほとんどの SQL ステートメントを含めることができます。制限については、セクションD.1「ストアドプログラムの制約」を参照してください。

20.4.2 イベントスケジューラの構成

イベントは、特別なイベントスケジューラスレッドによって実行されます。イベントスケジューラと呼ぶ場合、実際にはこのスレッドを指しています。実行中、イベントスケジューラスレッドとその現在の状態は、次の説明で示すように、PROCESS 権限を保有するユーザーが SHOW PROCESSLIST の出力で確認できます。

event_scheduler グローバルシステム変数によって、イベントスケジューラがサーバー上で有効であり実行しているかどうかが決まります。これは次の 3 つの値のいずれかを取り、それぞれ次に説明するようにイベントスケジュール設定に影響します。

- OFF: イベントスケジューラは停止しています。イベントスケジューラスレッドは実行されておらず、SHOW PROCESSLIST の出力に表示されておらず、スケジュール設定済みイベントが実行されていません。OFF が event_scheduler のデフォルト値です。

  イベントスケジューラが停止している場合 (event_scheduler が OFF です)、event_scheduler の値を ON に設定することで開始できます。(次の項目を参照してください。)

- ON: イベントスケジューラが開始され、イベントスケジューラスレッドがすべてのスケジュール設定済みイベントを実行しています。

  イベントスケジューラが ON の場合、イベントスケジューラスレッドは、デーモンプロセスとして SHOW PROCESSLIST の出力に一覧表示され、その状態は次に示すように表示されます。

```
mysql> SHOW PROCESSLIST\G
*************************** 1. row ***************************
     Id: 1
   User: root
   Host: localhost
     db: NULL
Command: Query
   Time: 0
  State: NULL
   Info: show processlist
*************************** 2. row ***************************
     Id: 2
   User: event_scheduler
   Host: localhost
     db: NULL
Command: Daemon
   Time: 3
  State: Waiting for next activation
```

```
   Info: NULL
2 rows in set (0.00 sec)
```

イベントスケジュール設定は、event_scheduler の値を OFF に設定することで停止できます。

- DISABLED: この値はイベントスケジューラを動作しないようにします。イベントスケジューラが DISABLED の場合、イベントスケジューラスレッドは実行していません (また、SHOW PROCESSLIST の出力にも表示されません)。また、イベントスケジューラの状態は実行時に変更できません。

イベントスケジューラのステータスが DISABLED に設定されていない場合、(SET を使用して) event_scheduler の ON と OFF を切り替えることができます。この変数を設定するときに、OFF に 0 を、ON に 1 を使用することも可能です。したがって、mysql クライアントで次の 4 つのどのステートメントを使用しても、イベントスケジューラをオンにできます。

```
SET GLOBAL event_scheduler = ON;
SET @@GLOBAL.event_scheduler = ON;
SET GLOBAL event_scheduler = 1;
SET @@GLOBAL.event_scheduler = 1;
```

同様に、次の 4 つのどのステートメントを使用してもイベントスケジューラをオフにできます。

```
SET GLOBAL event_scheduler = OFF;
SET @@GLOBAL.event_scheduler = OFF;
SET GLOBAL event_scheduler = 0;
SET @@GLOBAL.event_scheduler = 0;
```

ON と OFF には対応する数値がありますが、SELECT または SHOW VARIABLES によって event_scheduler に対して表示される値は、常に OFF、ON、または DISABLED のいずれかになります。DISABLED に対応する数値はありません。このため、この変数を設定するときに、ON と OFF は通常 1 と 0 よりも優先されます。

グローバル変数として指定しないで event_scheduler を設定しようとすると、エラーが発生します。

```
mysql< SET @@event_scheduler = OFF;
ERROR 1229 (HY000): Variable 'event_scheduler' is a GLOBAL
variable and should be set with SET GLOBAL
```

> **重要**
>
> イベントスケジューラを DISABLED に設定できるのは、サーバーの起動時だけです。event_scheduler が ON または OFF の場合、実行時にこれを DISABLED には設定できません。また、イベントスケジューラが起動時に DISABLED に設定されている場合、実行時に event_scheduler の値に変更できません。

イベントスケジューラを無効にするには、次の 2 つの方法のいずれかを使用します。

- サーバーの起動時のコマンド行オプションとして

```
--event-scheduler=DISABLED
```

- サーバー構成ファイル (Windows システムでの my.cnf または my.ini) で、サーバーが読み取る箇所 (たとえば、[mysqld] セクション内) に次の行を含めます。

```
event_scheduler=DISABLED
```

イベントスケジューラを有効にするには、必要に応じて、--event-scheduler=DISABLED コマンド行オプションを使用しないでサーバーを再起動するか、サーバー構成ファイルの event-scheduler=DISABLED を含む行を削除するかコメントアウトしたあとでサーバーを再起動します。または、サーバーの起動時に DISABLED 値の代わりに ON (または 1) か OFF (または 0) を使用できます。

> **注記**
>
> event_scheduler が DISABLED に設定されている場合、イベント操作ステートメントを発行できます。このような場合には警告もエラーも生成されません (ステートメント自体が有効であるとします)。ただし、この変数を ON (または 1) に設定するまで、スケジュール設定済みイベントは実行できません。これが行われると、イベントスケジューラスレッドは、スケジュール設定条件が満たされているすべてのイベントを実行します。

--skip-grant-tables オプションを使用して MySQL Server を起動すると、event_scheduler が DISABLED に設定され、コマンド行や my.cnf または my.ini ファイルで設定されたほかのすべての値をオーバーライドします (Bug #26807)。

イベントの作成、変更、または削除に使用される SQL ステートメントについては、セクション20.4.3「イベント構文」を参照してください。

MySQL 5.6 は、INFORMATION_SCHEMA データベースの EVENTS テーブルを提供します。このテーブルは、サーバー上で定義されているスケジュール設定済みイベントに関する情報を取得するためにクエリーできます。詳細は、セクション20.4.4「イベントメタデータ」およびセクション21.7「INFORMATION_SCHEMA EVENTS テーブル」を参照してください。

イベントスケジュール設定と MySQL 権限システムに関する情報については、セクション20.4.6「イベントスケジューラと MySQL 権限」を参照してください。

20.4.3 イベント構文

MySQL 5.6 には、スケジュール設定済みイベントを処理するための複数の SQL ステートメントが用意されています。

- 新しいイベントは CREATE EVENT ステートメントを使用して定義されます。セクション13.1.11「CREATE EVENT 構文」を参照してください。
- 既存のイベントの定義は、ALTER EVENT ステートメントによって変更できます。セクション13.1.2「ALTER EVENT 構文」を参照してください。
- スケジュール設定済みイベントが不要になった場合は、その定義者が、DROP EVENT ステートメントを使用してサーバーから削除できます。セクション13.1.22「DROP EVENT 構文」を参照してください。イベントがそのスケジュールの終了を過ぎても持続されるかどうかは、ON COMPLETION 句がある場合、この句によって決まります。セクション13.1.11「CREATE EVENT 構文」を参照してください。

  イベントが定義されているデータベースに対する EVENT 権限を保有するすべてのユーザーは、そのイベントをドロップできます。セクション20.4.6「イベントスケジューラと MySQL 権限」を参照してください。

20.4.4 イベントメタデータ

イベントに関するメタデータは次のように取得できます。

- mysql データベースの event テーブルをクエリーします。
- INFORMATION_SCHEMA データベースの EVENTS テーブルをクエリーします。セクション21.7「INFORMATION_SCHEMA EVENTS テーブル」を参照してください。
- SHOW CREATE EVENT ステートメントを使用します。セクション13.7.5.9「SHOW CREATE EVENT 構文」を参照してください。
- SHOW EVENTS ステートメントを使用します。セクション13.7.5.19「SHOW EVENTS 構文」を参照してください。

イベントスケジューラの時間表現

MySQL の各セッションには、セッションタイムゾーン (STZ) があります。これは、セッションの開始時にサーバーの time_zone グローバル値から初期化される time_zone セッション値ですが、セッション中に変更される可能性があります。

CREATE EVENT または ALTER EVENT ステートメントが実行するときに使用されているセッションタイムゾーンが、イベント定義で指定されている時間の解釈に使用されます。これがイベントタイムゾーン (ETZ) になります。つまり、イベントのスケジュール設定に使用され、イベントが実行するときにそのイベント内で有効になるタイムゾーンになります。

mysql.event テーブル内のイベント情報の表現については、execute_at、starts、および ends 時間は UTC に変換され、イベントタイムゾーンとともに格納されます。これにより、サーバータイムゾーンまたはサマータイムの影響に対し生じた変更とは無関係に、定義されたとおりにイベントの実行を処理できます。last_executed 時間も UTC で格納されます。

mysql.event から情報を選択すると、前述の時間は、UTC 値として取得されます。これらの時間は、INFORMATION_SCHEMA.EVENTS テーブルから、または SHOW EVENTS から選択して取得することもできますが、ETZ 値としてレポートされます。これらのソースから利用できるほかの時間は、イベントの作成時や最後の変更時を示します。これらは STZ 値として表示されます。次の表は、イベント時間の表現をまとめています。

値	mysql.event	INFORMATION_SCHEMA.EVENTS	SHOW EVENTS
Execute at	UTC	ETZ	ETZ
Starts	UTC	ETZ	ETZ
Ends	UTC	ETZ	ETZ
Last executed	UTC	ETZ	該当なし
Created	STZ	STZ	該当なし
Last altered	STZ	STZ	該当なし

20.4.5 イベントスケジューラのステータス

イベントスケジューラは、エラーまたは警告で終了したイベント実行に関する情報を、MySQL Server のエラーログに書き込みます。例については セクション20.4.6「イベントスケジューラと MySQL 権限」を参照してください。

デバッグおよびトラブルシューティングのためにイベントスケジューラの状態に関する状態を取得するには、mysqladmin debug を実行します (セクション4.5.2「mysqladmin — MySQL サーバーの管理を行うクライアント」を参照してください)。このコマンドの実行後に、ここに示すようなイベントスケジューラに関連した出力がサーバーのエラーログに含まれます。

```
Events status:
LLA = Last Locked At  LUA = Last Unlocked At
WOC = Waiting On Condition  DL = Data Locked

Event scheduler status:
State      : INITIALIZED
Thread id  : 0
LLA        : init_scheduler:313
LUA        : init_scheduler:318
WOC        : NO
Workers    : 0
Executed   : 0
Data locked: NO

Event queue status:
Element count   : 1
Data locked     : NO
Attempting lock : NO
LLA             : init_queue:148
LUA             : init_queue:168
WOC             : NO
Next activation : 0000-00-00 00:00:00
```

イベントスケジューラが実行するイベントの一部として生じるステートメント内で、診断メッセージ (エラーだけでなく警告も) がエラーログと、Windows ではアプリケーションイベントログに書き込まれます。頻繁に実行するイベントの場合、これにより、多数のメッセージが記録される結果になることがあります。たとえば、SELECT ... INTO var_list ステートメントの場合、クエリーが行を返さなければ、エラーコード 1329 で警告が発生し (No data)、変数値は変更されないままになります。クエリーが複数の行を返す場合は、エラー 1172 が発生します (結果が 2 行以上です)。どちらの条件についても、条件ハンドラを宣言すると、警告を記録させないようにできます。セクション13.6.7.2「DECLARE ... HANDLER 構文」を参照してください。複数の行を取得できるステートメントの場合、LIMIT 1 を使用して結果セットを単一の行に制限するという別の方法があります。

20.4.6 イベントスケジューラと MySQL 権限

スケジュール設定済みイベントの実行を有効または無効にするには、event_scheduler グローバルシステム変数の値を設定する必要があります。これには SUPER 権限が必要です。

EVENT 権限は、イベントの作成、変更、および削除を制御します。この権限は、GRANT を使用して与えることができます。たとえば、次の GRANT ステートメントは、myschema という名前のスキーマに対する EVENT 権限を、ユーザー jon@ghidora に与えます。

```
GRANT EVENT ON myschema.* TO jon@ghidora;
```

(このユーザーアカウントがすでに存在していることと、その他の点では変更されないままであると想定しています。)

この同じユーザーにすべてのスキーマに対する EVENT 権限を認めるには、次のステートメントを使用します。

```
GRANT EVENT ON *.* TO jon@ghidora;
```

EVENT 権限にはグローバルまたはスキーマレベルのスコープがあります。このため、単一のテーブルに対してこれを与えようとすると、次のようなエラーが生じます。

```
mysql> GRANT EVENT ON myschema.mytable TO jon@ghidora;
ERROR 1144 (42000): Illegal GRANT/REVOKE command; please
consult the manual to see which privileges can be used
```

イベントはその定義者の権限で実行され、定義者が必須の権限を保有していないアクションは実行できません。たとえば、jon@ghidora が myschema に対する EVENT 権限を保有しているとします。また、このユーザーは myschema に対する SELECT 権限は保有しているが、このスキーマに対するほかの権限は保有していないとします。jon@ghidora は、次のような新しいイベントを作成できます。

```
CREATE EVENT e_store_ts
  ON SCHEDULE
    EVERY 10 SECOND
  DO
    INSERT INTO myschema.mytable VALUES (UNIX_TIMESTAMP());
```

このユーザーは 1 分ほど待機したあと、テーブルに複数の新しい行が表示されることを予想して SELECT * FROM mytable; クエリーを実行します。実際は、テーブルは空です。ユーザーは該当するテーブルに対する INSERT 権限がないので、イベントの効果はありませんでした。

MySQL エラーログ (hostname.err) を調べると、イベントは実行していますが、RetCode=0 で示されているように、イベントが実行しようとしているアクションは失敗していることがわかります。

```
060209 22:39:44 [Note]    EVEX EXECUTING event newdb.e [EXPR:10]
060209 22:39:44 [Note]    EVEX EXECUTED event newdb.e  [EXPR:10]. RetCode=0
060209 22:39:54 [Note]    EVEX EXECUTING event newdb.e [EXPR:10]
060209 22:39:54 [Note]    EVEX EXECUTED event newdb.e  [EXPR:10]. RetCode=0
060209 22:40:04 [Note]    EVEX EXECUTING event newdb.e [EXPR:10]
060209 22:40:04 [Note]    EVEX EXECUTED event newdb.e  [EXPR:10]. RetCode=0
```

このユーザーは、エラーログにアクセスできない可能性が非常に高いので、直接それを実行することによって、イベントのアクションステートメントが有効であるかどうか検証できます。

```
mysql> INSERT INTO myschema.mytable VALUES (UNIX_TIMESTAMP());
ERROR 1142 (42000): INSERT command denied to user
'jon'@'ghidora' for table 'mytable'
```

INFORMATION_SCHEMA.EVENTS テーブルを調べることによって、e_store_ts が存在し有効になっているが、その LAST_EXECUTED カラムが NULL になっていることがわかります。

```
mysql> SELECT * FROM INFORMATION_SCHEMA.EVENTS
    >    WHERE EVENT_NAME='e_store_ts'
    >    AND EVENT_SCHEMA='myschema'\G
*************************** 1. row ***************************
  EVENT_CATALOG: NULL
   EVENT_SCHEMA: myschema
     EVENT_NAME: e_store_ts
        DEFINER: jon@ghidora
     EVENT_BODY: SQL
EVENT_DEFINITION: INSERT INTO myschema.mytable VALUES (UNIX_TIMESTAMP())
     EVENT_TYPE: RECURRING
     EXECUTE_AT: NULL
  INTERVAL_VALUE: 5
  INTERVAL_FIELD: SECOND
       SQL_MODE: NULL
         STARTS: 0000-00-00 00:00:00
           ENDS: 0000-00-00 00:00:00
         STATUS: ENABLED
   ON_COMPLETION: NOT PRESERVE
        CREATED: 2006-02-09 22:36:06
   LAST_ALTERED: 2006-02-09 22:36:06
  LAST_EXECUTED: NULL
  EVENT_COMMENT:
1 row in set (0.00 sec)
```

EVENT 権限を取り消すには、REVOKE ステートメントを使用します。この例では、スキーマ myschema に対する EVENT 権限が jon@ghidora ユーザーアカウントから削除されます。

```
REVOKE EVENT ON myschema.* FROM jon@ghidora;
```

> **重要**
>
> ユーザーから EVENT 権限を取り消しても、そのユーザーが作成したイベントが削除されたり無効にされたりすることはありません。
>
> 作成したユーザーの名前を変更したり削除したりしても、イベントが移行または削除されることはありません。

ユーザー jon@ghidora に、myschema スキーマに対する EVENT および INSERT 権限が与えられているとします。続いてこのユーザーが次のイベントを作成します。

```
CREATE EVENT e_insert
  ON SCHEDULE
    EVERY 7 SECOND
  DO
    INSERT INTO myschema.mytable;
```

このイベントの作成後、root は jon@ghidora の EVENT 権限を取り消します。ただし、e_insert は実行し続け、7 秒ごとに新しい行が mytable に挿入されます。root が次のどちらかのステートメントを発行した場合も、同じことが当てはまります。

- DROP USER jon@ghidora;
- RENAME USER jon@ghidora TO someotherguy@ghidora;

DROP USER または RENAME USER ステートメントの発行前後で、mysql.event テーブル (このセクションで後述します) か、INFORMATION_SCHEMA.EVENTS テーブル (セクション21.7「INFORMATION_SCHEMA EVENTS テーブル」を参照してください) を調べると、これが当てはまることを確認できます。

イベント定義は、mysql.event テーブルに格納されています。別のユーザーアカウントが作成したイベントを削除するには、MySQL root ユーザー (または必要な権限を保有する別のユーザー) がこのテーブルから行を削除できます。たとえば、前述のイベント e_insert を削除するには、root は次のステートメントを使用できます。

```
DELETE FROM mysql.event
  WHERE db = 'myschema'
    AND definer = 'jon@ghidora'
    AND name = 'e_insert';
```

mysql.event テーブルから行を削除するときには、イベント名、データベーススキーマ名、ユーザーアカウントを一致させることが重要です。これは、同じユーザーが、別々のスキーマに同じ名前の異なるイベントを作成できるためです。

ユーザーの EVENT 権限は、mysql.user および mysql.db テーブルの Event_priv カラムに格納されています。どちらの場合でも、このカラムには、「Y」または「N」のどちらかの値が含まれています。「N」がデフォルトです。指定されたユーザーがグローバルな EVENT 権限を保有している場合 (つまり、GRANT EVENT ON *.* を使用して権限が与えられた場合) にのみ、そのユーザーの mysql.user.Event_priv は「Y」に設定されます。スキーマレベルの EVENT 権限の場合、GRANT は、mysql.db に行を作成し、その行の Db カラムをスキーマの名前に、User カラムをユーザーの名前に、Event_priv カラムを「Y」に設定します。GRANT EVENT および REVOKE EVENT ステートメントがこれらのテーブルでの必要な操作を実行するので、これらのテーブルを直接操作する必要はありません。

5 つのステータス変数が、イベント関連操作のカウントを提供します (ただし、イベントが実行するステートメントのカウントは提供しません。セクションD.1「ストアドプログラムの制約」を参照してください)。これらを次に示します。

- Com_create_event: サーバーが最後に再起動してから実行された CREATE EVENT ステートメントの数。
- Com_alter_event: サーバーが最後に再起動してから実行された ALTER EVENT ステートメントの数。
- Com_drop_event: サーバーが最後に再起動してから実行された DROP EVENT ステートメントの数。
- Com_show_create_event: サーバーが最後に再起動してから実行された SHOW CREATE EVENT ステートメントの数。
- Com_show_events: サーバーが最後に再起動してから実行された SHOW EVENTS ステートメントの数。

ステートメント SHOW STATUS LIKE '%event%'; を実行すると、これらのすべての現在値を一度に表示できます。

20.5 ビューの使用

ビュー (更新可能なビューを含む) は MySQL Server 5.6 で使用できます。ビューは、呼び出されたときに結果セットを生成するストアドクエリーです。ビューは仮想テーブルとして機能します。

ビューをサポートしていなかった古いリリースから MySQL 5.6 にアップグレードした場合、ビューを使用するには、ビュー関連の権限を含めるように付与テーブルをアップグレードする必要があります。セクション 4.4.7「mysql_upgrade — MySQL テーブルのチェックとアップグレード」を参照してください。

次の説明では、ビューを作成し削除するための構文について記述し、それらの使用法の例をいくつか示します。

追加のリソース

- ビューを扱うときには、「Views User Forum」が役立ちます。
- MySQL のビューに関するよくある質問とその回答については、セクションA.6「MySQL 5.6 FAQ: ビュー」を参照してください。
- ビューの使用にはいくつかの制限があります。セクションD.5「ビューの制約」を参照してください。

20.5.1 ビューの構文

CREATE VIEW ステートメントは新しいビューを作成します (セクション13.1.20「CREATE VIEW 構文」を参照してください)。ビューの定義を変更したり、ビューを削除したりするには、ALTER VIEW (セクション 13.1.9「ALTER VIEW 構文」を参照してください) または DROP VIEW (セクション13.1.31「DROP VIEW 構文」を参照してください) を使用します。

ビューは、多くの種類の SELECT ステートメントから作成できできます。ベーステーブルまたはほかのビューを参照できます。結合、UNION、およびサブクエリーを使用できます。SELECT がテーブルをまったく参照しなくてもかまいません。次の例では、別のテーブルからの 2 つのカラムに加え、それらのカラムから計算される式を選択するビューを定義しています。

```
mysql> CREATE TABLE t (qty INT, price INT);
mysql> INSERT INTO t VALUES(3, 50), (5, 60);
mysql> CREATE VIEW v AS SELECT qty, price, qty*price AS value FROM t;
mysql> SELECT * FROM v;
+------+-------+-------+
| qty  | price | value |
+------+-------+-------+
|    3 |    50 |   150 |
|    5 |    60 |   300 |
+------+-------+-------+
mysql> SELECT * FROM v WHERE qty = 5;
+------+-------+-------+
| qty  | price | value |
+------+-------+-------+
|    5 |    60 |   300 |
+------+-------+-------+
```

20.5.2 ビュー処理アルゴリズム

CREATE VIEW または ALTER VIEW のオプションの ALGORITHM 句は、標準 SQL に対する MySQL 拡張です。これは、MySQL によるビューの処理方法に影響を与えます。ALGORITHM は、MERGE、TEMPTABLE、または UNDEFINED の 3 つの値を受け取ります。ALGORITHM 句が存在しない場合、デフォルトのアルゴリズムは UNDEFINED です。

MERGE の場合、ビューを参照するステートメントのテキストとビュー定義がマージされ、ビュー定義の部分が対応するステートメントの部分と置き換えられます。

TEMPTABLE の場合、ビューの結果が一時テーブル内に取得され、その後、ステートメントを実行するために使用されます。

UNDEFINED の場合、MySQL は使用するアルゴリズムを選択します。できるかぎり TEMPTABLE より MERGE が優先されます。これは通常、MERGE のほうが効率性が高く、一時テーブルを使用するとビューを更新できなくなるためです。

明示的に TEMPTABLE を選択する理由は、一時テーブルが作成されたあとで、ステートメントの処理を終了するために使用される前に、ベースとなるテーブルでロックを解放できるからです。その結果、MERGE アルゴリズ

ムよりもすみやかにロックが解除され、ビューを使用するほかのクライアントが長時間ブロックされることがなくなります。

次の 3 つの理由によって、ビューアルゴリズムをUNDEFINED にできます。

- CREATE VIEW ステートメントの中に ALGORITHM 句が存在しない。
- CREATE VIEW ステートメントに明示的な ALGORITHM = UNDEFINED 句が含まれている。
- 一時テーブルだけでしか処理できないビューに対して、ALGORITHM = MERGE が指定されている。この場合、MySQL は警告を発し、アルゴリズムを UNDEFINED に設定します。

前述のように、MERGE は、ビュー定義の対応する部分を、ビューを参照するステートメントにマージして処理されます。次の例で、MERGE アルゴリズムの動作について簡単に説明します。例では、次の定義を含むビュー v_merge が存在していると想定します。

```
CREATE ALGORITHM = MERGE VIEW v_merge (vc1, vc2) AS
SELECT c1, c2 FROM t WHERE c3 > 100;
```

例 1: 次のステートメントを発行するとします。

```
SELECT * FROM v_merge;
```

MySQL は次のようにステートメントを処理します。

- v_merge は t になる
- * は vc1, vc2 となり、c1, c2 と一致する
- ビュー WHERE 句が追加される

結果が実行されるステートメントは次のようになります。

```
SELECT c1, c2 FROM t WHERE c3 > 100;
```

例 2: 次のステートメントを発行するとします。

```
SELECT * FROM v_merge WHERE vc1 < 100;
```

このステートメントは、前述のステートメントと同様に処理されますが、vc1 < 100 が c1 < 100 になり、AND 連結詞を使用してビュー WHERE 句がステートメント WHERE 句に追加される点が異なります (また、句の一部が確実に正しい優先順位で実行されるように、かっこが追加されます)。結果が実行されるステートメントは次のようになります。

```
SELECT c1, c2 FROM t WHERE (c3 > 100) AND (c1 < 100);
```

事実上、実行されるステートメントには、次の形式の WHERE 句が含まれます。

```
WHERE (select WHERE) AND (view WHERE)
```

MERGE アルゴリズムを使用できない場合、一時テーブルを代わりに使用する必要があります。ビューに次のいずれかの構造構文が含まれる場合、MERGE は使用できません。

- 集計関数 (SUM()、MIN()、MAX()、COUNT() など)
- DISTINCT
- GROUP BY
- HAVING
- LIMIT
- UNION または UNION ALL
- 選択リスト内のサブクエリー
- リテラル値だけの参照 (この場合、ベースとなるテーブルがありません)

20.5.3 更新可能および挿入可能なビュー

いくつかのビューは更新可能です。つまり、これらのビューを UPDATE、DELETE、INSERT などのステートメントで使用して、ベースとなるテーブルの内容を更新できます。ビューが更新可能であるためには、そのビュー内の行とベースとなるテーブル内の行の間に 1 対 1 の関係が存在する必要があります。また、ビューを更新不可能にするその他の特定の構造構文も存在します。より具体的には、次のいずれかを含む場合、ビューは更新可能ではありません。

- 集計関数 (SUM()、MIN()、MAX()、COUNT() など)
- DISTINCT
- GROUP BY
- HAVING
- UNION または UNION ALL
- 選択リスト内のサブクエリー
- 特定の結合 (このセクションで後述する結合に関する追加説明を参照してください)
- FROM 句内の更新不可能なビュー
- FROM 句内のテーブルを参照する WHERE 句内のサブクエリー
- リテラル値だけの参照 (この場合、更新するベースとなるテーブルがありません)
- ALGORITHM = TEMPTABLE の使用 (一時テーブルを使用すると常にビューは更新不可能になります)
- ベーステーブルのいずれかのカラムに対する複数の参照。

挿入可能性 (INSERT ステートメントで更新可能であること) については、更新可能なビューがビューカラムに対する次の追加要件も満たしている場合に挿入可能になります。

- 重複したビューカラム名が存在しないようにする必要があります。
- ビューには、デフォルト値を持たない、ベーステーブル内のすべてのカラムが含まれている必要があります。
- ビューカラムは、派生カラムではなく、単純なカラム参照である必要があります。派生カラムは、単純なカラム参照ではなく、式から派生したカラムです。派生したカラムの例は次のとおりです。

```
3.14159
col1 + 3
UPPER(col2)
col3 / col4
(subquery)
```

単純なカラム参照と派生カラムが混在しているビューは挿入できませんが、派生カラム以外のカラムだけを更新する場合は、更新可能になります。次のビューを考えてみてください。

```
CREATE VIEW v AS SELECT col1, 1 AS col2 FROM t;
```

このビューは、col2 が式から派生しているので挿入できません。ただし、更新で col2 を更新しようとしていない場合は、更新可能になります。次の更新は許可されます。

```
UPDATE v SET col1 = 0;
```

次の更新は、派生カラムを更新しようとしているので、許可されません。

```
UPDATE v SET col2 = 0;
```

MERGE アルゴリズムで処理できるとすれば、複数テーブルビューが更新できる可能性があります。これを実現するには、ビューで (外部結合または UNION ではなく) 内部結合を使用する必要があります。また、ビュー定義内の単一のテーブルだけを更新できるので、SET 句は、ビュー内のいずれかのテーブルのカラムだけを指名する必要があります。UNION ALL を使用するビューは、実装が一時テーブルを使用して処理するので、理論的に更新可能でも許可されません。

更新可能な複数テーブルビューでは、INSERT は、単一のテーブルに挿入する場合に機能します。DELETE はサポートされません。

INSERT DELAYED は、ビューではサポートされません。

テーブルに AUTO_INCREMENT カラムが含まれている場合、AUTO_INCREMENT カラムが含まれていないテーブル上の挿入可能なビューに挿入すると、LAST_INSERT_ID() の値を変更しません。これは、ビューの一部ではないカラムにデフォルト値を挿入した副作用が現れないようにするためです。

更新可能なビューに対して WITH CHECK OPTION 句を指定すると、select_statement 内の WHERE 句が true である行を除く行への挿入または更新を回避できます。

更新可能なビューに対する WITH CHECK OPTION 句では、そのビューが別のビューとの関連で定義されている場合、LOCAL および CASCADED キーワードによってチェックテストのスコープが決定されます。LOCAL キーワードは、CHECK OPTION を、定義されているビューのみに制限します。CASCADED を指定すると、ベースとなるビューに対するチェックも評価されます。どちらのキーワードも指定されていない場合、デフォルトは CASCADED になります。次のテーブルと一連のビューの定義を考えてみてください。

```
mysql> CREATE TABLE t1 (a INT);
mysql> CREATE VIEW v1 AS SELECT * FROM t1 WHERE a < 2
    -> WITH CHECK OPTION;
mysql> CREATE VIEW v2 AS SELECT * FROM v1 WHERE a > 0
    -> WITH LOCAL CHECK OPTION;
mysql> CREATE VIEW v3 AS SELECT * FROM v1 WHERE a > 0
    -> WITH CASCADED CHECK OPTION;
```

ここで、v2 および v3 ビューは、v1 という別のビューの観点で定義されています。v2 には LOCAL チェックオプションがあるので、挿入は、v2 チェックに対してのみテストされます。v3 には CASCADED チェックオプションがあるので、挿入はそれ自身のチェックに対してだけでなく、ベースとなるビューのチェックに対してもテストされます。次のステートメントでこれらの違いを示しています。

```
mysql> INSERT INTO v2 VALUES (2);
Query OK, 1 row affected (0.00 sec)
mysql> INSERT INTO v3 VALUES (2);
ERROR 1369 (HY000): CHECK OPTION failed 'test.v3'
```

MySQL は、CREATE VIEW 時に、ビューの更新可能性フラグというフラグを設定します。UPDATE および DELETE (および同様の操作) がビューで有効な場合、フラグは YES (true) に設定されます。それ以外の場合、フラグは NO (false) に設定されます。INFORMATION_SCHEMA.VIEWS テーブルの IS_UPDATABLE カラムは、このフラグのステータスを表示します。これは、ビューが更新可能であるかどうかをサーバーが常に把握していることを意味します。ビューが更新可能ではない場合、UPDATE、DELETE、INSERT などのステートメントは無効であり、拒否されます。(このセクションの別の箇所で説明しているように、ビューが更新可能である場合でも、ビューへの挿入はできない場合もあります。)

ビューを更新できるかどうかは、updatable_views_with_limit システム変数の値に影響されます。セクション 5.1.4「サーバーシステム変数」を参照してください。

20.5.4 ビューのメタデータ

ビューに関するメタデータは次のように取得できます。

- INFORMATION_SCHEMA データベースの VIEWS テーブルをクエリーします。セクション 21.28「INFORMATION_SCHEMA VIEWS テーブル」を参照してください。
- SHOW CREATE VIEW ステートメントを使用します。セクション13.7.5.14「SHOW CREATE VIEW 構文」を参照してください。

20.6 ストアドプログラムおよびビューのアクセスコントロール

ストアドプログラムとビューは使用する前に定義され、参照されるときに、その権限を決定するセキュリティーのコンテキスト内で実行します。これらの権限は、その DEFINER 属性と、存在する場合はその SQL SECURITY 特性で制御されます。

すべてのストアドプログラム (プロシージャー、関数、トリガー、およびイベント) とビューには、MySQL アカウントを指名する DEFINER 属性を含めることができます。DEFINER 属性をストアドプログラムまたはビュー定義から省略した場合、デフォルトのアカウントは、オブジェクトを作成するユーザーになります。

さらに、ストアドルーチン (プロシージャーおよび関数) とビューには、値が DEFINER または INVOKER である SQL SECURITY 特性があり、オブジェクトが定義側のコンテキストで実行するか、呼び出し元のコンテキストで実行するかを指定できます。SQL SECURITY 特性を省略した場合、デフォルトは定義側のコンテキストになります。

トリガーとイベントには、SQL SECURITY 特性がなく、常に定義側のコンテキストで実行します。サーバーが必要に応じて自動的にこれらのオブジェクトを呼び出すので、呼び出し元ユーザーは存在しません。

定義側と呼び出し元のセキュリティーのコンテキストは次のように異なります。

- 定義側のセキュリティーコンテキストで実行するストアドプログラムまたはビューは、DEFINER 属性で指名されたアカウントの権限で実行します。これらの権限は、呼び出し元ユーザーの権限とは完全に異なる場合があります。呼び出し元は、オブジェクトを参照するために適切な権限 (たとえば、ストアドプロシージャーを呼び出すための EXECUTE や、ビューから選択するための SELECT) が必要ですが、オブジェクトが実行すると、呼び出し元の権限は無視され、DEFINER アカウント権限だけが重要になります。このアカウントの権限が低い場合、オブジェクトが実行できる操作は、それに応じて制限されます。DEFINER アカウントに高い権限が与えられている場合 (root アカウントなど)、呼び出し元のユーザーにかかわらず、オブジェクトは強力な操作を実行できます。
- 呼び出し元のセキュリティーコンテキストで実行するストアドルーチンまたはビューは、呼び出し元が権限を持つ操作だけを実行できます。DEFINER 属性は指定できますが、呼び出し元のコンテキストで実行するオブジェクトに対して効果はありません。

次のストアドプロシージャーを検討してください。

```
CREATE DEFINER = 'admin'@'localhost' PROCEDURE p1()
SQL SECURITY DEFINER
BEGIN
  UPDATE t1 SET counter = counter + 1;
END;
```

p1 に対する EXECUTE 権限を持つどのユーザーでも、CALL ステートメントを使用してこれを呼び出すことができます。ただし、p1 が実行するときには、DEFINER のセキュリティーコンテキストで実行するので、DEFINER 属性で指名されたアカウントである 'admin'@'localhost' の権限で実行します。このアカウントは、p1 の EXECUTE 権限のほかに、テーブル t1 の UPDATE 権限が必要です。それ以外の場合、プロシージャーは失敗します。

続いて次のストアドプロシージャーを検討してください。これは p1 と同じですが、その SQL SECURITY 特性が INVOKER である点が異なります。

```
CREATE DEFINER = 'admin'@'localhost' PROCEDURE p2()
SQL SECURITY INVOKER
BEGIN
  UPDATE t1 SET counter = counter + 1;
END;
```

p2 は p1 と異なり、INVOKER のセキュリティーコンテキストで実行します。DEFINER 属性は無関係であり、p2 は呼び出し元ユーザーの権限で実行します。呼び出し元に p2 に対する EXECUTE 権限、またはテーブル t1 に対する UPDATE 権限が不足している場合、p2 は失敗します。

MySQL は、次のルールを使用して、ユーザーがオブジェクトの DEFINER 属性で指定できるアカウントを制御します。

- SUPER 権限がある場合にかぎり、自身のアカウント以外の DEFINER 値を指定できます。
- SUPER 権限がない場合、唯一の正当なユーザー値は、文字どおり指定するか、CURRENT_USER を使用して指定した自身のアカウントです。定義者をほかのアカウントに設定することはできません。

ストアドプログラムおよびビューの作成と使用に関して考えられるリスクを最小限に抑えるため、次のガイドラインに従ってください。

- 可能な場合は、ストアドルーチンまたはビューに対して、オブジェクト定義の SQL SECURITY INVOKER を使用して、オブジェクトが実行する操作に適したアクセス許可を持つユーザーだけが使用できるようにします。
- SUPER 権限を持つアカウントの使用中に、定義側のコンテキストのストアドプログラムまたはビューを作成する場合は、オブジェクトが実行する操作に必要な権限だけを所有しているアカウントを指名する明示的な DEFINER 属性を指定します。高い権限を持つ DEFINER アカウントは、絶対に必要な場合にのみ指定してください。
- 管理者は、SUPER 権限をユーザーに与えなければ、高い権限を持つ DEFINER アカウントをユーザーが指定できないようにできます。
- 定義側のコンテキストのオブジェクトを作成するときには、呼び出し元ユーザーに権限のないデータに定義側がアクセスできる場合があります。権限のないユーザーに特定の権限を与えなければ、これらのオブジェクトへの参照を防止できる場合があります。

- ストアドプロシージャーまたはストアドファンクションに対する EXECUTE 権限を持たないユーザーは、これを参照できません。
- ビューに対する適切な権限 (ビューから選択するための SELECT、ビューに挿入するための INSERT など) を持っていないユーザーは、ビューを参照できません。

ただし、トリガーに対するこのような制御は存在しません。ユーザーが直接トリガーを参照することはないからです。トリガーは常に、DEFINER コンテキストで実行し、特別な権限を持たないユーザーによる通常のテーブルアクセスを含め、トリガーが関連付けられているテーブルへのアクセスがあるとアクティブ化されます。DEFINER アカウントに高い権限が与えられている場合、トリガーは、慎重を要する操作または危険な操作を実行できます。トリガーの作成に必要な SUPER および TRIGGER 権限が、作成したユーザーのアカウントから削除された場合にも、このことは引き続き当てはまります。管理者は、この権限の組み合わせをユーザーに認める場合、特に注意する必要があります。

20.7 ストアドプログラムのバイナリロギング

バイナリログには、データベースの内容を変更する SQL ステートメントに関する情報が含まれます。この情報は、変更について記述した「イベント」の形式で格納されます。バイナリログには 2 つの重要な目的があります。

- レプリケーションの場合、バイナリログは、スレーブサーバーに送信されるステートメントのレコードとして、マスターレプリケーションサーバー上で使用されます。マスターサーバーは、そのバイナリログに格納されているイベントをそのスレーブに送信し、スレーブはこれらのイベントを実行して、マスター上で実行されたものと同じデータ変更を実行します。セクション17.2「レプリケーションの実装」を参照してください。
- ある特定のデータリカバリ操作には、バイナリログの使用が必要です。バックアップファイルがリストアされたあと、バックアップの作成後に記録されたバイナリログ内のイベントが、再度実行されます。これらのイベントは、データベースをバックアップのポイントから最新の状態に持って行きます。セクション7.3.2「リカバリへのバックアップの使用」を参照してください。

ただし、ロギングがステートメントレベルで行われる場合、ストアドプログラム (ストアドプロシージャーおよびストアドファンクション、トリガー、イベント) に関して該当する、特定のバイナリロギングの問題があります。

- 場合によっては、ステートメントが、マスターとスレーブで別々の行セットに影響する可能性があります。
- スレーブ上で実行された複製ステートメントは、完全な権限を持つスレーブ SQL スレッドで処理されます。プロシージャーが、マスターサーバーとスレーブサーバーで別々の実行パスに従うことが可能なので、ユーザーは、スレーブ上でのみ実行し、完全な権限を持つスレーブスレッドで処理される危険なステートメントを含んだルーチンを作成できます。
- データを変更するストアドプログラムが非決定的である場合、再現可能ではありません。これにより、マスターとスレーブでデータが異なる結果になったり、リストアしたデータが元のデータと一致しなくなったりする場合があります。

このセクションでは、MySQL 5.6 のストアドプログラムのバイナリロギングの処理について説明します。ここでは、実装がストアドプログラムの使用に対して設定している現在の条件と、問題を避けるために実行可能な対処について記しています。また、これらの条件の理由に関する追加情報も示します。

一般に、ここで述べる問題は、SQL ステートメントレベルでバイナリロギングが行われるときに生じます。行ベースのバイナリロギングを使用する場合、ログには、SQL ステートメントを実行した結果として個々の行に行われた変更が含まれます。ルーチンまたはトリガーが実行されると、行の変更が記録されますが、変更を行なったステートメントは記録されません。ストアドプロシージャーの場合、これは CALL ステートメントが記録されないことを意味します。ストアドファンクションの場合、関数内で行われた行の変更が記録され、関数呼び出しは記録されません。トリガーの場合、トリガーによって行われた行の変更が記録されます。スレーブ側では、行の変更だけが表示され、ストアドプログラムの呼び出しは表示されません。行ベースのロギングに関する一般情報については、セクション17.1.2「レプリケーション形式」を参照してください。

特に明記しないかぎり、ここでの説明では、--log-bin オプションを指定してサーバーを起動することによって、バイナリロギングを有効にしていると想定しています。(セクション5.2.4「バイナリログ」を参照してください。)バイナリログが有効でない場合、レプリケーションは可能でなく、バイナリログをデータリカバリに利用することもできません。

MySQL 5.6 でストアドファンクションを使用するための現在の条件は、次のように要約できます。これらの条件は、ストアドプロシージャーまたはイベントスケジューラのイベントには適用されず、バイナリロギングが有効でないかぎり適用されません。

- ストアドファンクションを生成または変更するには、ユーザーは、通常必要になる CREATE ROUTINE 権限または ALTER ROUTINE 権限以外に、SUPER 権限が必要です。(関数定義の DEFINER 値によっては、バイナリロギングが有効かどうかにかかわらず SUPER が必要になる場合があります。セクション13.1.15「CREATE PROCEDURE および CREATE FUNCTION 構文」を参照してください。)

- ストアドファンクションを作成するとき、その関数が決定的であるということ、またはデータを変更しないということを宣言する必要があります。そのようにしないと、データリカバリまたレプリケーションにとって安全でなくなる可能性があります。

  デフォルトでは、CREATE FUNCTION ステートメントを受け入れるには、DETERMINISTIC、NO SQL、または READS SQL DATA の少なくとも 1 つを明示的に指定する必要があります。そうでない場合はエラーが発生します。

  ```
  ERROR 1418 (HY000): This function has none of DETERMINISTIC, NO SQL,
  or READS SQL DATA in its declaration and binary logging is enabled
  (you *might* want to use the less safe log_bin_trust_function_creators
  variable)
  ```

  次の関数は決定的なため (また、データを変更しません)、安全です。

  ```
  CREATE FUNCTION f1(i INT)
  RETURNS INT
  DETERMINISTIC
  READS SQL DATA
  BEGIN
    RETURN i;
  END;
  ```

  次の関数は UUID() を使用しますが、これは決定的でないため、関数も決定的でなく、安全ではありません。

  ```
  CREATE FUNCTION f2()
  RETURNS CHAR(36) CHARACTER SET utf8
  BEGIN
    RETURN UUID();
  END;
  ```

  次の関数はデータを変更するので、安全ではない可能性があります。

  ```
  CREATE FUNCTION f3(p_id INT)
  RETURNS INT
  BEGIN
    UPDATE t SET modtime = NOW() WHERE id = p_id;
    RETURN ROW_COUNT();
  END;
  ```

  関数の性質の評価は、作成者の「誠実さ」に基づいています。MySQL は、DETERMINISTIC と宣言された関数に非決定的な結果を生成するステートメントが含まれていないかどうかをチェックしません。

- DETERMINISTIC を指定しないで、決定的であるストアドファンクションを作成することは可能ですが、ステートメントベースのバイナリロギングを使用してこの関数を実行できません。このような関数を実行するには、行ベースまたは混合バイナリロギングを使用する必要があります。または、関数定義で DETERMINISTIC と明示的に指定すると、ステートメントベースのバイナリロギングを含むあらゆる種類のロギングを使用できます。

- 関数作成に関する前述の条件 (SUPER 権限を持つ必要があることと、関数が決定的であるか、データを変更しないと宣言する必要があること) を緩和するには、log_bin_trust_function_creators グローバルシステム変数を 1 に設定します。デフォルトでこの変数には 0 の値が設定されていますが、次のように変更できます。

  ```
  mysql> SET GLOBAL log_bin_trust_function_creators = 1;
  ```

  サーバーの起動時に --log-bin-trust-function-creators=1 オプションを使用することによって、この変数を設定することもできます。

  バイナリロギングが有効でない場合、log_bin_trust_function_creators は適用されません。前述のように、関数定義の DEFINER 値が必要としないかぎり、関数の作成に SUPER は必要ありません。

- レプリケーションで安全ではない可能性のある (そのため、これらを使用するストアドファンクションも安全でなくなります) 組み込み関数の詳細は、セクション17.4.1「レプリケーションの機能と問題」を参照してください。

トリガーは、ストアドファンクションと似ているので、関数に関する前述の説明がトリガーにも当てはまりますが、CREATE TRIGGER にはオプションの DETERMINISTIC 特性がないため、トリガーは常に決定的であ

ると想定されるという点が異なります。ただし、この想定は一部の場合で無効になることがあります。たとえば、UUID() 関数は非決定的です (また、複製しません)。トリガーでのこのような関数の使用には注意する必要があります。

トリガーはテーブルを更新できるので、必要な権限がない場合には、CREATE TRIGGER で、ストアドファンクションの場合と同様のエラーメッセージが表示されます。スレーブ側では、スレーブは DEFINER トリガー属性を使用して、トリガーの作成者であると思われるユーザーを特定します。

このセクションの残りの部分では、ロギングの実装とその意味に関する追加詳細について説明します。ストアドルーチンの使用に関する現在のロギング関連の条件の理論的根拠についての背景に関心がある場合には、こちらをお読みください。この説明はステートメントベースのロギングにのみ該当し、行ベースのロギングには該当しませんが、CREATE および DROP ステートメントは、ロギングモードとは無関係にステートメントとして記録されるという最初の項目は除きます。

- サーバーは、CREATE EVENT、CREATE PROCEDURE、CREATE FUNCTION、ALTER EVENT、ALTER PROCEDURE、ALTER FUNCTION、DROP EVENT、DROP PROCEDURE、および DROP FUNCTION ステートメントをバイナリログに書き込みます。
- ストアドファンクションの呼び出しは、この関数がデータを変更し、それ以外では記録されないようなステートメント内で行われた場合に、SELECT ステートメントとして記録されます。これにより、記録されないステートメントでストアドファンクションを使用した結果生じたデータの変更をレプリケーションできなくなるという事態が防止されます。たとえば、SELECT ステートメントはバイナリログに書き込まれませんが、SELECT は、変更を行うストアドファンクションを呼び出す場合があります。これを扱うため、SELECT func_name() ステートメントは、指定した関数が変更を行うときにバイナリログに書き込まれます。次のステートメントがマスターで実行されるとします。

```
CREATE FUNCTION f1(a INT) RETURNS INT
BEGIN
  IF (a < 3) THEN
    INSERT INTO t2 VALUES (a);
  END IF;
  RETURN 0;
END;

CREATE TABLE t1 (a INT);
INSERT INTO t1 VALUES (1),(2),(3);

SELECT f1(a) FROM t1;
```

SELECT ステートメントが実行されると、関数 f1() は 3 回呼び出されます。このうち 2 回の呼び出しで行を挿入し、MySQL は各行に対し SELECT ステートメントを記録します。つまり、MySQL は次のステートメントをバイナリログに書き込みます。

```
SELECT f1(1);
SELECT f1(2);
```

サーバーは、エラーを発生させるストアドプロシージャーをストアドファンクションが呼び出すときに、そのストアドファンクションの呼び出しに対する SELECT ステートメントも記録します。この場合、サーバーは、予想されるエラーコードとともに、SELECT ステートメントをログに書き込みます。スレーブ上で、同じエラーが起きた場合、これは予想される結果でありレプリケーションは継続します。それ以外の場合は、レプリケーションは停止します。

- 関数によって実行されるステートメントではなく、ストアドファンクションの呼び出しのロギングは、レプリケーションでは、次の 2 つの要因から生じるセキュリティー上の意味があります。
  - 関数が、マスターサーバーとスレーブサーバーで別々の実行パスに従うことが可能です。
  - スレーブ上で実行されたステートメントは、完全な権限を持つスレーブ SQL スレッドで処理されます。

  つまり、ユーザーは関数を作成するために CREATE ROUTINE 権限を持つ必要がありますが、完全な権限を持つスレッドで処理されるスレーブ上でのみ実行する危険なステートメントを含んだ関数を作成できます。たとえば、マスターサーバーとスレーブサーバーのサーバー ID 値がそれぞれ 1 と 2 の場合、マスターサーバー上のユーザーは、安全ではない関数 unsafe_func() を次のように作成し呼び出すことができます。

```
mysql> delimiter //
mysql> CREATE FUNCTION unsafe_func () RETURNS INT
    -> BEGIN
    ->   IF @@server_id=2 THEN dangerous_statement; END IF;
    ->   RETURN 1;
    -> END;
    -> //
mysql> delimiter ;
```

```
mysql> INSERT INTO t VALUES(unsafe_func());
```

CREATE FUNCTION ステートメントおよび INSERT ステートメントはバイナリログに書き込まれるので、スレーブサーバーはそれらを実行します。スレーブ SQL スレッドには完全な権限があるので、危険なステートメントを実行します。したがって、関数の呼び出しがマスターとスレーブに与える効果は異なり、レプリケーションは安全ではなくなります。

バイナリロギングを有効にしているサーバーに対するこの危険から保護するために、ストアドファンクションの作成者は、必要な通常の CREATE ROUTINE 権限に加え、SUPER 権限も持つ必要があります。同様に、ALTER FUNCTION を使用するには、ユーザーは ALTER ROUTINE 権限に加え、SUPER 権限を持つ必要があります。SUPER 権限がないと、エラーが発生します。

```
ERROR 1419 (HY000): You do not have the SUPER privilege and
binary logging is enabled (you *might* want to use the less safe
log_bin_trust_function_creators variable)
```

関数作成者が SUPER 権限を持つよう要求しない場合 (たとえば、システム上の CREATE ROUTINE 権限を持つすべてのユーザーが経験豊かなアプリケーション開発者である場合)、log_bin_trust_function_creators グローバルシステム変数を 1 に設定します。サーバーの起動時に --log-bin-trust-function-creators=1 オプションを使用することによって、この変数を設定することもできます。バイナリロギングが有効でない場合、log_bin_trust_function_creators は適用されません。前述のように、関数定義の DEFINER 値が必要としないかぎり、関数の作成に SUPER は必要ありません。

- 更新を実行する関数が非決定的である場合、再現可能ではありません。これは次の 2 つの望ましくない影響を及ぼす可能性があります。
  - スレーブがマスターと一致しなくなります。
  - リストアされたデータが元のデータと異なります。

  これらの問題に対処するために、MySQL では、関数を決定的であるか、データを変更しないと宣言しないかぎり、マスターサーバーでは関数の作成と変更は拒否されるという要件を強制しています。ここでは次の 2 つの関数特性セットが適用されます。
  - DETERMINISTIC 特性と NOT DETERMINISTIC 特性は、関数が一定の入力に対して常に同じ結果を生成するかどうかを示します。どちらの特性も指定されていない場合は、デフォルトは NOT DETERMINISTIC です。関数が決定的であることを宣言するには、明示的に DETERMINISTIC を指定する必要があります。
  - CONTAINS SQL、NO SQL、READS SQL DATA、および MODIFIES SQL DATA 特性は、関数がデータを読み取るか書き込むかに関する情報を示します。NO SQL または READS SQL DATA は、関数がデータを変更しないことを示しますが、特性が指定されていない場合にデフォルトは CONTAINS SQL になるので、これらのどちらかを明示的に指定する必要があります。

  デフォルトでは、CREATE FUNCTION ステートメントを受け入れるには、DETERMINISTIC、NO SQL、または READS SQL DATA の少なくとも 1 つを明示的に指定する必要があります。そうでない場合はエラーが発生します。

  ```
  ERROR 1418 (HY000): This function has none of DETERMINISTIC, NO SQL,
  or READS SQL DATA in its declaration and binary logging is enabled
  (you *might* want to use the less safe log_bin_trust_function_creators
  variable)
  ```

  log_bin_trust_function_creators を 1 に設定した場合、関数が決定的であるか、データを変更しないという要件は破棄されます。
- ストアドプロシージャーの呼び出しは CALL レベルでなく、ステートメントレベルで記録されます。つまり、サーバーは、CALL ステートメントを記録せず、実際に実行するプロシージャー内のステートメントを記録します。その結果、マスターで行われた同じ変更が、スレーブサーバーで確認されます。これにより、別々のマシン上で異なる実行パスを持つプロシージャーから生じる問題が防止されます。

  一般に、ストアドプロシージャー内で実行されるステートメントは、スタンドアロンでステートメントを実行した場合に適用されるものと同じルールを使用して、バイナリログに書き込まれます。プロシージャー内でのステートメントの実行は、非プロシージャーのコンテキストとまったく同じにはならないので、プロシージャーステートメントのロギング時には、十分に注意してください。
  - 記録されるステートメントには、ローカルプロシージャー変数への参照が含まれる場合があります。これらの変数は、ストアドプロシージャーのコンテキスト外に存在しないので、このような変数を参照するステートメントは、文字どおりには記録できません。代わりに、ローカル変数のそれぞれの参照は、ロギングのために次の構造構文に置き換えられます。

```
NAME_CONST(var_name, var_value)
```

var_name はローカル変数名であり、var_value は、ステートメントの記録時に変数に含まれていた値を示す定数です。NAME_CONST() には var_value の値と var_name の「名前」が含まれます。したがって、この関数を直接呼び出した場合、次のような結果が得られます。

```
mysql> SELECT NAME_CONST('myname', 14);
+--------+
| myname |
+--------+
|     14 |
+--------+
```

NAME_CONST() は、マスター上でストアドプロシージャー内で実行された元のステートメントと同じ効果で、記録されたスタンドアロンのステートメントを、スレーブ上でも実行できるようにします。

NAME_CONST() を使用した結果、ソースカラム式がローカル変数を参照するときに、CREATE TABLE ... SELECT ステートメントの問題が生じる場合があります。これらの参照を NAME_CONST() 式に変換した結果、マスターサーバーとスレーブサーバーでカラム名が異なったり、正当なカラム識別子としては長すぎる名前になったりすることがあります。回避策では、ローカル変数を参照するカラムのエイリアスを指定します。myvar の値が 1 の場合は次のステートメントを検討してください。

```
CREATE TABLE t1 SELECT myvar;
```

これは次のように書き換えられます。

```
CREATE TABLE t1 SELECT NAME_CONST(myvar, 1);
```

マスターテーブルとスレーブテーブルに同じカラム名があることを確認するには、次のようなステートメントを作成します。

```
CREATE TABLE t1 SELECT myvar AS myvar;
```

書き換えられたステートメントは次のようになります。

```
CREATE TABLE t1 SELECT NAME_CONST(myvar, 1) AS myvar;
```

- 記録されるステートメントには、ユーザー定義変数への参照が含まれる場合があります。これを処理するために、MySQL は、SET ステートメントをバイナリログに書き込んで、マスター上にあるものと同じ値の変数がスレーブ上にもあることを確認します。たとえば、ステートメントが変数 @my_var を参照する場合、そのステートメントはバイナリログ内で次のステートメントに優先されます。ここで、value マスター上の @my_var の値です。

  ```
  SET @my_var = value;
  ```

- プロシージャーの呼び出しは、コミットまたはロールバックしたトランザクション内で行えます。プロシージャー実行のトランザクションの側面が正しく複製されるように、トランザクションのコンテキストが説明されます。つまり、サーバーは、実際にデータを実行し修正するプロシージャー内のステートメントを記録し、BEGIN、COMMIT、および ROLLBACK ステートメントも必要に応じて記録します。たとえば、プロシージャーがトランザクションテーブルだけを更新し、ロールバックされるトランザクション内で実行される場合、これらの更新は記録されません。プロシージャーがコミットされたトランザクション内で行われた場合、BEGIN および COMMIT ステートメントが更新とともに記録されます。ロールバックしたトランザクション内で実行するプロシージャーの場合、そのステートメントは、ステートメントがスタンドアロンで実行された場合に適用されるものと同じルールを使用して記録されます。

  - トランザクションテーブルに対する更新は記録されません。
  - 非トランザクションテーブルに対する更新は、ロールバックで取り消されないので、記録されます。
  - トランザクションテーブルと非トランザクションテーブルの混在に対する更新は、スレーブがマスターと同じ変更およびロールバックを行うように、BEGIN と ROLLBACK で囲まれて記録されます。

- ストアドプロシージャーの呼び出しは、ストアドファンクション内から呼び出される場合、ステートメントレベルのバイナリログに書き込まれません。この場合、記録される唯一の対象は、関数を呼び出すステートメント (記録されたステートメント内で行われた場合) または DO ステートメント (記録されないステートメント内で行われた場合) です。このため、それ以外の場合にプロシージャー自体が安全であっても、プロシージャーを呼び出すストアドファンクションを使用するときには注意を払う必要があります。

第 21 章 INFORMATION_SCHEMA テーブル

目次

21.1 INFORMATION_SCHEMA CHARACTER_SETS テーブル 2502
21.2 INFORMATION_SCHEMA COLLATIONS テーブル 2502
21.3 INFORMATION_SCHEMA COLLATION_CHARACTER_SET_APPLICABILITY テーブル 2503
21.4 INFORMATION_SCHEMA COLUMNS テーブル 2503
21.5 INFORMATION_SCHEMA COLUMN_PRIVILEGES テーブル 2504
21.6 INFORMATION_SCHEMA ENGINES テーブル 2504
21.7 INFORMATION_SCHEMA EVENTS テーブル 2504
21.8 INFORMATION_SCHEMA GLOBAL_STATUS および SESSION_STATUS テーブル 2507
21.9 INFORMATION_SCHEMA GLOBAL_VARIABLES および SESSION_VARIABLES テーブル 2508
21.10 INFORMATION_SCHEMA KEY_COLUMN_USAGE テーブル 2508
21.11 INFORMATION_SCHEMA OPTIMIZER_TRACE テーブル 2509
21.12 INFORMATION_SCHEMA PARAMETERS テーブル 2509
21.13 INFORMATION_SCHEMA PARTITIONS テーブル 2510
21.14 INFORMATION_SCHEMA PLUGINS テーブル 2512
21.15 INFORMATION_SCHEMA PROCESSLIST テーブル 2513
21.16 INFORMATION_SCHEMA PROFILING テーブル 2514
21.17 INFORMATION_SCHEMA REFERENTIAL_CONSTRAINTS テーブル 2515
21.18 INFORMATION_SCHEMA ROUTINES テーブル 2515
21.19 INFORMATION_SCHEMA SCHEMATA テーブル 2517
21.20 INFORMATION_SCHEMA SCHEMA_PRIVILEGES テーブル 2517
21.21 INFORMATION_SCHEMA STATISTICS テーブル 2517
21.22 INFORMATION_SCHEMA TABLES テーブル 2518
21.23 INFORMATION_SCHEMA TABLESPACES テーブル 2519
21.24 INFORMATION_SCHEMA TABLE_CONSTRAINTS テーブル 2520
21.25 INFORMATION_SCHEMA TABLE_PRIVILEGES テーブル 2520
21.26 INFORMATION_SCHEMA TRIGGERS テーブル 2520
21.27 INFORMATION_SCHEMA USER_PRIVILEGES テーブル 2522
21.28 INFORMATION_SCHEMA VIEWS テーブル 2522
21.29 InnoDB の INFORMATION_SCHEMA テーブル 2523
21.29.1 INFORMATION_SCHEMA INNODB_CMP および INNODB_CMP_RESET テーブル 2524
21.29.2 INFORMATION_SCHEMA INNODB_CMP_PER_INDEX および INNODB_CMP_PER_INDEX_RESET テーブル 2525
21.29.3 INFORMATION_SCHEMA INNODB_CMPMEM および INNODB_CMPMEM_RESET テーブル 2526
21.29.4 INFORMATION_SCHEMA INNODB_TRX テーブル 2527
21.29.5 INFORMATION_SCHEMA INNODB_LOCKS テーブル 2529
21.29.6 INFORMATION_SCHEMA INNODB_LOCK_WAITS テーブル 2530
21.29.7 INFORMATION_SCHEMA INNODB_SYS_TABLES テーブル 2531
21.29.8 INFORMATION_SCHEMA INNODB_SYS_INDEXES テーブル 2532
21.29.9 INFORMATION_SCHEMA INNODB_SYS_COLUMNS テーブル 2533
21.29.10 INFORMATION_SCHEMA INNODB_SYS_FIELDS テーブル 2534
21.29.11 INFORMATION_SCHEMA INNODB_SYS_FOREIGN テーブル 2535
21.29.12 INFORMATION_SCHEMA INNODB_SYS_FOREIGN_COLS テーブル 2535
21.29.13 INFORMATION_SCHEMA INNODB_SYS_TABLESTATS ビュー 2536
21.29.14 INFORMATION_SCHEMA INNODB_SYS_DATAFILES テーブル 2537
21.29.15 INFORMATION_SCHEMA INNODB_SYS_TABLESPACES テーブル 2537
21.29.16 INFORMATION_SCHEMA INNODB_BUFFER_PAGE テーブル 2540
21.29.17 INFORMATION_SCHEMA INNODB_BUFFER_PAGE_LRU テーブル 2541
21.29.18 INFORMATION_SCHEMA INNODB_BUFFER_POOL_STATS テーブル 2543
21.29.19 INFORMATION_SCHEMA INNODB_METRICS テーブル 2545
21.29.20 INFORMATION_SCHEMA INNODB_FT_CONFIG テーブル 2546
21.29.21 INFORMATION_SCHEMA INNODB_FT_DEFAULT_STOPWORD テーブル 2547
21.29.22 INFORMATION_SCHEMA INNODB_FT_INDEX_TABLE テーブル 2548
21.29.23 INFORMATION_SCHEMA INNODB_FT_INDEX_CACHE テーブル 2549
21.29.24 INFORMATION_SCHEMA INNODB_FT_DELETED テーブル 2550
21.29.25 INFORMATION_SCHEMA INNODB_FT_BEING_DELETED テーブル 2551
21.30 MySQL Cluster の INFORMATION_SCHEMA テーブル 2552
21.30.1 INFORMATION_SCHEMA FILES テーブル 2552
21.30.2 INFORMATION_SCHEMA ndb_transid_mysql_connection_map テーブル 2556

21.31 スレッドプールの INFORMATION_SCHEMA テーブル 2557
21.31.1 INFORMATION_SCHEMA TP_THREAD_STATE テーブル 2558
21.31.2 INFORMATION_SCHEMA TP_THREAD_GROUP_STATE テーブル 2558
21.31.3 INFORMATION_SCHEMA TP_THREAD_GROUP_STATS テーブル 2560
21.32 SHOW ステートメントの拡張 2561

INFORMATION_SCHEMA では、データベースメタデータへのアクセスを実現し、データベースまたはテーブルの名前、カラムのデータ型、アクセス権限などの MySQL Server に関する情報を提供します。この情報に使用されることがある別の用語が、データディクショナリとシステムカタログです。

INFORMATION_SCHEMA データベースの使用上の注意

INFORMATION_SCHEMA は、各 MySQL インスタンス内のデータベースであり、MySQL Server が保持するほかのすべてのデータベースに関する情報を格納する場所です。INFORMATION_SCHEMA データベースには複数の読み取り専用テーブルが含まれます。これらには実際にはビューがあるので、関連付けられたファイルはなく、トリガーは設定できません。また、その名前を持つデータベースディレクトリもありません。

USE ステートメントを使用してデフォルトデータベースとして INFORMATION_SCHEMA を選択できますが、実行できる操作はテーブル内容の読み取りだけで、テーブルに対する INSERT、UPDATE、DELETE 操作は実行できません。

例

次に INFORMATION_SCHEMA から情報を取り出すステートメントの例を示します。

```
mysql> SELECT table_name, table_type, engine
    -> FROM information_schema.tables
    -> WHERE table_schema = 'db5'
    -> ORDER BY table_name;
+------------+------------+--------+
| table_name | table_type | engine |
+------------+------------+--------+
| fk         | BASE TABLE | InnoDB |
| fk2        | BASE TABLE | InnoDB |
| goto       | BASE TABLE | MyISAM |
| into       | BASE TABLE | MyISAM |
| k          | BASE TABLE | MyISAM |
| kurs       | BASE TABLE | MyISAM |
| loop       | BASE TABLE | MyISAM |
| pk         | BASE TABLE | InnoDB |
| t          | BASE TABLE | MyISAM |
| t2         | BASE TABLE | MyISAM |
| t3         | BASE TABLE | MyISAM |
| t7         | BASE TABLE | MyISAM |
| tables     | BASE TABLE | MyISAM |
| v          | VIEW       | NULL   |
| v2         | VIEW       | NULL   |
| v3         | VIEW       | NULL   |
| v56        | VIEW       | NULL   |
+------------+------------+--------+
17 rows in set (0.01 sec)
```

説明: このステートメントは、データベース db5 内のすべてのテーブルのリストを要求し、テーブルの名前、種類、ストレージエンジンという 3 つの情報だけを示します。

文字セットの考慮事項

文字カラム (TABLES.TABLE_NAME など) の定義は、通常、VARCHAR(N) CHARACTER SET utf8 であり、ここで N は少なくとも 64 です。MySQL は、このようなカラムでのすべての検索、ソート、比較、およびほかの文字列操作に、この文字セット (utf8_general_ci) のデフォルトの照合順序を使用します。

一部の MySQL オブジェクトはファイルとして表されるので、INFORMATION_SCHEMA 文字列カラムでの検索は、ファイルシステムでの大文字と小文字の区別によって影響を受けることがあります。詳細は、セクション 10.1.7.9「照合順序と INFORMATION_SCHEMA 検索」を参照してください。

SHOW ステートメントの代替方法としての INFORMATION_SCHEMA

SELECT ... FROM INFORMATION_SCHEMA ステートメントは、MySQL がサポートするさまざまな SHOW ステートメント (SHOW DATABASES、SHOW TABLES など) により提供された情報にアクセスするための一貫性の高い方法として用意されています。SELECT は、SHOW に比べて次の利点があります。

- すべてのアクセスがテーブル上で行われるので、Codd のルールに準拠しています。
- SELECT ステートメントの使い慣れた構文を使用でき、学習が必要なものは一部のテーブルおよびカラムの名前だけです。
- 実装者はキーワードの追加を心配する必要がありません。
- INFORMATION_SCHEMA クエリーからの結果をフィルタリング、ソート、連結でき、構文解析するデータ構造やテキスト表現など、アプリケーションで必要なあらゆる形式に変換できます。
- この手法は、ほかのデータベースシステムとの相互運用可能性に優れています。たとえば、Oracle Database ユーザーは、Oracle データディクショナリのテーブルのクエリーに慣れています。

SHOW はよく知られ、広く使用されているので、SHOW ステートメントは引き続き代替方法として有効です。実際、セクション21.32「SHOW ステートメントの拡張」で説明しているように、INFORMATION_SCHEMA の実装に伴い、SHOW に拡張が行われています。

権限

各 MySQL ユーザーには、これらのテーブルへのアクセス権がありますが、ユーザーが適切なアクセス権限を持つオブジェクトに対応したテーブル内の行だけを表示できます。一部の場合 (たとえば INFORMATION_SCHEMA.ROUTINES テーブル内の ROUTINE_DEFINITION カラム) では、権限が不足しているユーザーには NULL と表示されます。これらの制限は InnoDB テーブルには適用されません。PROCESS 権限だけがあればこれらを表示できます。

同じ権限が、INFORMATION_SCHEMA からの情報の選択と、SHOW ステートメントを通じた同じ情報の表示に適用されます。どちらの場合でも、オブジェクトに関する情報を表示するには、そのオブジェクトに対する何らかの権限が必要です。

パフォーマンスに関する考慮事項

複数のデータベースの情報を検索する INFORMATION_SCHEMA クエリーは、長時間かかり、パフォーマンスに影響を及ぼす可能性があります。クエリーの効率を確認するには、EXPLAIN を使用できます。EXPLAIN 出力を使用した INFORMATION_SCHEMA クエリーの調整に関する詳細は、セクション 8.2.4「INFORMATION_SCHEMA クエリーの最適化」を参照してください。

標準に関する考慮事項

MySQL での INFORMATION_SCHEMA テーブル構造の実装は ANSI/ISO SQL:2003 標準パート 11 の Schemata に準拠しています。SQL:2003 の中核機能の F021 基本情報スキーマにほぼ準拠することを意図しています。

SQL Server 2000 (これも標準に準拠しています) のユーザーであれば、非常に類似していることがわかるでしょう。ただし、MySQL では実装に関連しない多くのカラムを省略し、MySQL 固有のカラムを追加しています。このようなカラムの 1 つが、INFORMATION_SCHEMA.TABLES テーブル内の ENGINE カラムです。

ほかの DBMS では syscat や system などのさまざまな名前を使用していますが、標準の名前は INFORMATION_SCHEMA です。

標準または DB2、SQL Server、Oracle で予約されている名前を使用しないように、「MySQL 拡張」のマークを付けて一部のカラムの名前を変更しています。(たとえば、TABLES テーブルでの COLLATION を TABLE_COLLATION に変更しました。)https://web.archive.org/web/20070428032454/http://www.dbazine.com/db2/db2-disarticles/gulutzan5 の記事の末尾近くにある予約語のリストを参照してください。

INFORMATION_SCHEMA 参照セクションでの規則

次のセクションでは、INFORMATION_SCHEMA でのテーブルおよびカラムのそれぞれについて説明します。カラムごとに次の 3 つの情報があります。

- 「INFORMATION_SCHEMA 名」には、INFORMATION_SCHEMA テーブルのカラムの名前が示されます。これは、「備考」フィールドで「MySQL 拡張」と記されていないかぎり、標準の SQL 名に一致します。
- 「SHOW 名」には、もっとも近い SHOW ステートメントに同等のフィールド名がある場合は、この名前が示されます。
- 「備考」には、必要に応じて追加情報が記されます。このフィールドが NULL の場合、カラムの値が常に NULL であることを意味します。このフィールドに「MySQL 拡張」とある場合、そのカラムは標準 SQL に対する MySQL 拡張です。

多くのセクションは、どの SHOW ステートメントが、INFORMATION_SCHEMA から情報を取得する SELECT と同等であるかを示します。デフォルトデータベースの情報を表示する SHOW ステートメントでは、FROM db_name 句を省略した場合、INFORMATION_SCHEMA テーブルから情報を取得するクエリーの WHERE 句に AND TABLE_SCHEMA = SCHEMA() 条件を追加すると、デフォルトデータベースの情報を選択できます。

InnoDB ストレージエンジンに固有の INFORMATION_SCHEMA テーブルの詳細は、セクション21.29「InnoDB の INFORMATION_SCHEMA テーブル」を参照してください。スレッドプールプラグインに固有の INFORMATION_SCHEMA テーブルの詳細は、セクション21.31「スレッドプールの INFORMATION_SCHEMA テーブル」を参照してください。

INFORMATION_SCHEMA データベースに関するよくある質問とその回答については、セクションA.7「MySQL 5.6 FAQ: INFORMATION_SCHEMA」を参照してください。

21.1 INFORMATION_SCHEMA CHARACTER_SETS テーブル

CHARACTER_SETS テーブルは、利用できる文字セットに関する情報を提供します。

INFORMATION_SCHEMA 名	SHOW 名	備考
CHARACTER_SET_NAME	Charset	
DEFAULT_COLLATE_NAME	Default collation	
DESCRIPTION	Description	MySQL 拡張
MAXLEN	Maxlen	MySQL 拡張

次のステートメントは同等です。

```
SELECT * FROM INFORMATION_SCHEMA.CHARACTER_SETS
  [WHERE CHARACTER_SET_NAME LIKE 'wild']

SHOW CHARACTER SET
  [LIKE 'wild']
```

21.2 INFORMATION_SCHEMA COLLATIONS テーブル

COLLATIONS テーブルは、各文字セットの照合順序に関する情報を提供します。

INFORMATION_SCHEMA 名	SHOW 名	備考
COLLATION_NAME	Collation	
CHARACTER_SET_NAME	Charset	MySQL 拡張
ID	Id	MySQL 拡張
IS_DEFAULT	Default	MySQL 拡張
IS_COMPILED	Compiled	MySQL 拡張
SORTLEN	Sortlen	MySQL 拡張

- COLLATION_NAME は照合順序名です。
- CHARACTER_SET_NAME は、照合順序が関連付けられている文字セットの名前です。
- ID は照合順序 ID です。
- IS_DEFAULT は、照合順序がその文字セットのデフォルトであるかどうかを示します。
- IS_COMPILED は、文字セットがサーバー内にコンパイルされるかどうかを示します。
- SORTLEN は、文字セットで表された文字列をソートするために必要なメモリー容量に関係しています。

照合順序情報は、SHOW COLLATION ステートメントから取得することもできます。次のステートメントは同等です。

```
SELECT COLLATION_NAME FROM INFORMATION_SCHEMA.COLLATIONS
  [WHERE COLLATION_NAME LIKE 'wild']

SHOW COLLATION
  [LIKE 'wild']
```

21.3 INFORMATION_SCHEMA COLLATION_CHARACTER_SET_APPLICABILITY テーブル

COLLATION_CHARACTER_SET_APPLICABILITY テーブルは、どの文字セットがどの照合順序に適用されるかを示します。カラムは SHOW COLLATION から取得する最初の 2 つの表示フィールドと同等です。

INFORMATION_SCHEMA 名	SHOW 名	備考
COLLATION_NAME	Collation	
CHARACTER_SET_NAME	Charset	

21.4 INFORMATION_SCHEMA COLUMNS テーブル

COLUMNS テーブルは、テーブル内のカラムに関する情報を提供します。

INFORMATION_SCHEMA 名	SHOW 名	備考
TABLE_CATALOG		def
TABLE_SCHEMA		
TABLE_NAME		
COLUMN_NAME	Field	
ORDINAL_POSITION		注を参照
COLUMN_DEFAULT	Default	
IS_NULLABLE	Null	
DATA_TYPE	Type	
CHARACTER_MAXIMUM_LENGTH	Type	
CHARACTER_OCTET_LENGTH		
NUMERIC_PRECISION	Type	
NUMERIC_SCALE	Type	
DATETIME_PRECISION	Type	
CHARACTER_SET_NAME		
COLLATION_NAME	Collation	
COLUMN_TYPE	Type	MySQL 拡張
COLUMN_KEY	Key	MySQL 拡張
EXTRA	Extra	MySQL 拡張
PRIVILEGES	Privileges	MySQL 拡張
COLUMN_COMMENT	Comment	MySQL 拡張

注:

- SHOW では、Type 表示には、異なる複数の COLUMNS カラムの値が含まれます。
- ORDINAL_POSITION は ORDER BY ORDINAL_POSITION と記す場合があるため必要です。SHOW とは異なり、SELECT には自動順序付けはありません。
- CHARACTER_OCTET_LENGTH は、マルチバイト文字セットを除き、CHARACTER_MAXIMUM_LENGTH と同じである必要があります。
- CHARACTER_SET_NAME は Collation から派生できます。たとえば、SHOW FULL COLUMNS FROM t と指定し、Collation カラムに latin1_swedish_ci の値が表示されている場合は、文字セットは最初の下線の前にあるもの、つまり latin1 になります。
- MySQL 5.6.4 で DATETIME_PRECISION が追加されました。

次のステートメントはほぼ同等です。

```
SELECT COLUMN_NAME, DATA_TYPE, IS_NULLABLE, COLUMN_DEFAULT
```

```
  FROM INFORMATION_SCHEMA.COLUMNS
  WHERE table_name = 'tbl_name'
  [AND table_schema = 'db_name']
  [AND column_name LIKE 'wild']

SHOW COLUMNS
  FROM tbl_name
  [FROM db_name]
  [LIKE 'wild']
```

21.5 INFORMATION_SCHEMA COLUMN_PRIVILEGES テーブル

COLUMN_PRIVILEGES テーブルは、カラムの権限に関する情報を提供します。この情報は mysql.columns_priv 付与テーブルからのものです。

INFORMATION_SCHEMA 名	SHOW 名	備考
GRANTEE		'user_name'@'host_name' 値
TABLE_CATALOG		def
TABLE_SCHEMA		
TABLE_NAME		
COLUMN_NAME		
PRIVILEGE_TYPE		
IS_GRANTABLE		

注:

- SHOW FULL COLUMNS の出力では、権限はすべて 1 つのフィールドにあり、select,insert,update,references のように小文字で記されます。COLUMN_PRIVILEGES では、行ごとに 1 つの権限があり、大文字で記されます。
- PRIVILEGE_TYPE は、SELECT、INSERT、UPDATE、REFERENCES の値のいずれか (1 つのみ) を含むことができます。
- ユーザーに GRANT OPTION 権限がある場合、IS_GRANTABLE は YES になります。それ以外の場合は、IS_GRANTABLE は NO になります。出力には、GRANT OPTION が個別の権限としてリストされません。

次のステートメントは同等ではありません。

```
SELECT ... FROM INFORMATION_SCHEMA.COLUMN_PRIVILEGES

SHOW GRANTS ...
```

21.6 INFORMATION_SCHEMA ENGINES テーブル

ENGINES テーブルは、ストレージエンジンに関する情報を提供します。

INFORMATION_SCHEMA 名	SHOW 名	備考
ENGINE	Engine	MySQL 拡張
SUPPORT	Support	MySQL 拡張
COMMENT	Comment	MySQL 拡張
TRANSACTIONS	Transactions	MySQL 拡張
XA	XA	MySQL 拡張
SAVEPOINTS	Savepoints	MySQL 拡張

注:

- ENGINES テーブルは、非標準テーブルです。この内容は、SHOW ENGINES ステートメントのカラムに対応します。そのカラムの説明については、セクション13.7.5.17「SHOW ENGINES 構文」を参照してください。

セクション13.7.5.17「SHOW ENGINES 構文」も参照してください。

21.7 INFORMATION_SCHEMA EVENTS テーブル

EVENTS テーブルは、セクション20.4「イベントスケジューラの使用」で説明しているスケジュール設定済みのイベントに関する情報を提供します。SHOW Name 値は、SHOW EVENTS ステートメントのカラム名に対応します。

INFORMATION_SCHEMA 名	SHOW 名	備考
EVENT_CATALOG		def、MySQL 拡張
EVENT_SCHEMA	Db	MySQL 拡張
EVENT_NAME	Name	MySQL 拡張
DEFINER	Definer	MySQL 拡張
TIME_ZONE	Time zone	MySQL 拡張
EVENT_BODY		MySQL 拡張
EVENT_DEFINITION		MySQL 拡張
EVENT_TYPE	Type	MySQL 拡張
EXECUTE_AT	Execute at	MySQL 拡張
INTERVAL_VALUE	Interval value	MySQL 拡張
INTERVAL_FIELD	Interval field	MySQL 拡張
SQL_MODE		MySQL 拡張
STARTS	Starts	MySQL 拡張
ENDS	Ends	MySQL 拡張
STATUS	Status	MySQL 拡張
ON_COMPLETION		MySQL 拡張
CREATED		MySQL 拡張
LAST_ALTERED		MySQL 拡張
LAST_EXECUTED		MySQL 拡張
EVENT_COMMENT		MySQL 拡張
ORIGINATOR	Originator	MySQL 拡張
CHARACTER_SET_CLIENT	character_set_client	MySQL 拡張
COLLATION_CONNECTION	collation_connection	MySQL 拡張
DATABASE_COLLATION	Database Collation	MySQL 拡張

注:

- EVENTS テーブルは、非標準テーブルです。
- EVENT_CATALOG: このカラムの値は常に def です。
- EVENT_SCHEMA: このイベントが属しているスキーマ (データベース) の名前です。
- EVENT_NAME: イベントの名前です。
- DEFINER: 'user_name'@'host_name' 形式で示した、イベントを作成したユーザーのアカウントです。
- TIME_ZONE: イベントタイムゾーンです。これはイベントをスケジュール設定するために使用され、実行するときにイベント内で有効なタイムゾーンです。デフォルト値は SYSTEM です。
- EVENT_BODY: イベントの DO 句に使用される言語です。MySQL 5.6 ではこれは常に SQL です。

  このカラムは、以前の MySQL バージョンに存在していたものと同じ名前 (現在は EVENT_DEFINITION という名前) のカラムと混同しないでください。
- EVENT_DEFINITION: イベントの DO 句を構成する SQL ステートメントのテキストです。つまり、このイベントで実行されたステートメントです。
- EVENT_TYPE: イベントの繰り返しタイプであり、ONE TIME (一時的) または RECURRING (繰り返し) のどちらかになります。
- EXECUTE_AT: 一度だけのイベントの場合、これはイベントの作成に使用された CREATE EVENT ステートメント、またはイベントを変更した最後の ALTER EVENT ステートメントの AT 句で指定された DATETIME 値

です。このカラムに表示された値は、イベントの AT 句に含まれた、INTERVAL 値の加算または減算に影響します。たとえば、イベントが ON SCHEDULE AT CURRENT_TIMESTAMP + '1:6' DAY_HOUR を使用して作成され、イベントが 2006-02-09 の 14:05:30 に作成された場合、カラムに表示される値は '2006-02-10 20:05:30' になります。

イベントのタイミングが AT 句ではなく EVERY 句で決定される場合 (つまり、イベントが繰り返しである場合)、このカラムの値は NULL になります。

- INTERVAL_VALUE: 繰り返しイベントの場合、このカラムにはイベント EVERY 句の数値部分が含まれます。

  一度だけのイベント (AT 句によってタイミングが決定されるイベント) の場合、このカラムは NULL になります。

- INTERVAL_FIELD: 繰り返しイベントの場合、このカラムは、イベントのタイミングを制御する EVERY 句の単位部分が含まれます。したがって、このカラムには、「YEAR」、「QUARTER」、「DAY」などの値が含まれます。

  一度だけのイベント (AT 句によってタイミングが決定されるイベント) の場合、このカラムは NULL になります。

- SQL_MODE: イベントが作成または変更されたときに有効であり、イベントが実行したときの SQL モードです。指定可能な値については、セクション5.1.7「サーバー SQL モード」を参照してください。

- STARTS: 定義に STARTS 句を含む繰り返しイベントでは、このカラムには対応する DATETIME 値が含まれます。EXECUTE_AT カラムの場合と同様に、この値は使用されている式を解きます。

  イベントのタイミングに影響する STARTS 句がない場合、このカラムは NULL です

- ENDS: 定義に ENDS 句を含む繰り返しイベントでは、このカラムには対応する DATETIME 値が含まれます。EXECUTE_AT カラムの場合と同様に、この値は使用されている式を解きます。

  イベントのタイミングに影響する ENDS 句がない場合、このカラムは NULL です。

- STATUS: ENABLED、DISABLED、SLAVESIDE_DISABLED の 3 つの値のいずれかになります。

  SLAVESIDE_DISABLED は、イベントの作成が、レプリケーションマスターとして機能している別の MySQL サーバーで行われ、スレーブとして機能している現在の MySQL サーバーに複製されたが、イベントは現時点でスレーブ上で実行されていないことを示します。詳細は、セクション17.4.1.11「呼び出される機能のレプリケーション」を参照してください。

- ON_COMPLETION: PRESERVE または NOT PRESERVE の 2 つの値のどちらかになります。

- CREATED: イベントが作成された日時です。これは TIMESTAMP 値です。

- LAST_ALTERED: イベントが最後に変更された日時です。これは TIMESTAMP 値です。イベントが作成されてから変更されなかった場合、このカラムには CREATED カラムと同じ値が保持されます。

- LAST_EXECUTED: イベントが最後に実行された日時です。DATETIME 値。イベントが一度も実行されていない場合、このカラムは NULL です。

  LAST_EXECUTED はイベントが開始した時点を示します。このため、ENDS カラムが LAST_EXECUTED より小さくなることは決してありません。

- EVENT_COMMENT: イベントにコメントがある場合のコメントのテキストです。ない場合は、このカラムの値は空の文字列になります。

- ORIGINATOR: このイベントが作成された MySQL サーバーのサーバー ID で、レプリケーションで使用されます。デフォルト値は 0 です。

- CHARACTER_SET_CLIENT: イベントが作成されたときの character_set_client システム変数のセッション値です。

- COLLATION_CONNECTION: イベントが作成されたときの collation_connection システム変数のセッション値です。

- DATABASE_COLLATION: イベントが関連付けられているデータベースの照合順序です。

例: 次に示すように、ユーザー jon@ghidora が e_daily という名前のイベントを作成し、ALTER EVENT ステートメントを使用して数分後に変更するとします。

```
DELIMITER |
```

```
CREATE EVENT e_daily
   ON SCHEDULE
     EVERY 1 DAY
   COMMENT 'Saves total number of sessions then clears the table each day'
   DO
     BEGIN
       INSERT INTO site_activity.totals (time, total)
         SELECT CURRENT_TIMESTAMP, COUNT(*)
           FROM site_activity.sessions;
       DELETE FROM site_activity.sessions;
     END |

DELIMITER ;

ALTER EVENT e_daily
   ENABLED;
```

(コメントは複数の行にわたって記述できます。)

このユーザーは続いて次の SELECT ステートメントを実行し、次の出力が表示されます。

```
mysql> SELECT * FROM INFORMATION_SCHEMA.EVENTS
    > WHERE EVENT_NAME = 'e_daily'
    > AND EVENT_SCHEMA = 'myschema'\G
*************************** 1. row ***************************
       EVENT_CATALOG: def
        EVENT_SCHEMA: test
          EVENT_NAME: e_daily
             DEFINER: me@localhost
           TIME_ZONE: SYSTEM
          EVENT_BODY: SQL
    EVENT_DEFINITION: BEGIN
       INSERT INTO site_activity.totals (time, total)
         SELECT CURRENT_TIMESTAMP, COUNT(*)
           FROM site_activity.sessions;
       DELETE FROM site_activity.sessions;
     END
       EVENT_TYPE: RECURRING
       EXECUTE_AT: NULL
    INTERVAL_VALUE: 1
    INTERVAL_FIELD: DAY
         SQL_MODE:
          STARTS: 2008-09-03 12:13:39
            ENDS: NULL
          STATUS: ENABLED
      ON_COMPLETION: NOT PRESERVE
            CREATED: 2008-09-03 12:13:39
       LAST_ALTERED: 2008-09-03 12:13:39
      LAST_EXECUTED: NULL
      EVENT_COMMENT: Saves total number of sessions then clears the
                     table each day
         ORIGINATOR: 1
CHARACTER_SET_CLIENT: latin1
COLLATION_CONNECTION: latin1_swedish_ci
  DATABASE_COLLATION: latin1_swedish_ci
```

セクション20.4.4「イベントメタデータ」で説明しているように、EVENTS テーブル内の時間は、イベントタイムゾーンまたは現在のセッションタイムゾーンを使用して表示されます。

セクション13.7.5.19「SHOW EVENTS 構文」も参照してください。

21.8 INFORMATION_SCHEMA GLOBAL_STATUS および SESSION_STATUS テーブル

GLOBAL_STATUS および SESSION_STATUS テーブルは、サーバーのステータス変数に関する情報を提供します。それらの内容は、SHOW GLOBAL STATUS および SHOW SESSION STATUS ステートメントが生成する情報に対応します (セクション13.7.5.36「SHOW STATUS 構文」を参照してください)。

INFORMATION_SCHEMA 名	SHOW 名	備考
VARIABLE_NAME	Variable_name	
VARIABLE_VALUE	値	

注:

- これらのテーブルそれぞれの VARIABLE_VALUE カラムは VARCHAR(1024) として定義されます。

21.9 INFORMATION_SCHEMA GLOBAL_VARIABLES および SESSION_VARIABLES テーブル

GLOBAL_VARIABLES および SESSION_VARIABLES テーブルは、サーバーのステータス変数に関する情報を提供します。これらの内容は、SHOW GLOBAL VARIABLES および SHOW SESSION VARIABLES ステートメントが生成した情報に対応します (セクション13.7.5.40「SHOW VARIABLES 構文」を参照してください)。

INFORMATION_SCHEMA 名	SHOW 名	備考
VARIABLE_NAME	Variable_name	
VARIABLE_VALUE	値	

注:

- これらのテーブルそれぞれの VARIABLE_VALUE カラムは VARCHAR(1024) として定義されます。完全には表示されない非常に長い値を持つ変数の場合、回避策として SELECT を使用します。例:

```
SELECT @@GLOBAL.innodb_data_file_path;
```

21.10 INFORMATION_SCHEMA KEY_COLUMN_USAGE テーブル

KEY_COLUMN_USAGE テーブルは、どのキーカラムに制約があるかを説明します。

INFORMATION_SCHEMA 名	SHOW 名	備考
CONSTRAINT_CATALOG		def
CONSTRAINT_SCHEMA		
CONSTRAINT_NAME		
TABLE_CATALOG		def
TABLE_SCHEMA		
TABLE_NAME		
COLUMN_NAME		
ORDINAL_POSITION		
POSITION_IN_UNIQUE_CONSTRAINT		
REFERENCED_TABLE_SCHEMA		
REFERENCED_TABLE_NAME		
REFERENCED_COLUMN_NAME		

注:

- 制約が外部キーの場合、これは外部キーのカラムで、外部キーが参照するカラムではありません。
- ORDINAL_POSITION の値は制約内のカラムの位置であり、テーブル内のカラムの位置ではありません。カラムの位置には 1 から始まる番号が付けられています。
- POSITION_IN_UNIQUE_CONSTRAINT の値は、一意の主キーの制約に対し NULL です。外部キーの制約の場合、これは、参照されているテーブルのキーの順序位置になります。

  次の定義を持つ t1 および t3 という 2 つのテーブルがあるとします。

```
CREATE TABLE t1
(
   s1 INT,
   s2 INT,
   s3 INT,
   PRIMARY KEY(s3)
) ENGINE=InnoDB;

CREATE TABLE t3
(
   s1 INT,
   s2 INT,
```

```
  s3 INT,
  KEY(s1),
  CONSTRAINT CO FOREIGN KEY (s2) REFERENCES t1(s3)
) ENGINE=InnoDB;
```

これらの 2 つのテーブルに対し、KEY_COLUMN_USAGE テーブルには次の 2 つの行があります。

- CONSTRAINT_NAME = 'PRIMARY'、TABLE_NAME = 't1'、COLUMN_NAME = 's3'、ORDINAL_POSITION = 1、POSITION_IN_UNIQUE_CONSTRAINT = NULL を含む 1 つの行。
- CONSTRAINT_NAME = 'CO'、TABLE_NAME = 't3'、COLUMN_NAME = 's2'、ORDINAL_POSITION = 1、POSITION_IN_UNIQUE_CONSTRAINT = 1 を含む 1 つの行。

21.11 INFORMATION_SCHEMA OPTIMIZER_TRACE テーブル

OPTIMIZER_TRACE テーブルは、オプティマイザトレース機能で生成される情報を提供します。追跡を有効にするには、optimizer_trace システム変数を使用します。詳細については、「MySQL Internals: Tracing the Optimizer」を参照してください。

MySQL 5.6.3 では OPTIMIZER_TRACE テーブルが追加されました。

21.12 INFORMATION_SCHEMA PARAMETERS テーブル

PARAMETERS テーブルは、ストアドプロシージャーとストアドファンクションのパラメータと、ストアドファンクションの戻り値に関する情報を提供します。パラメータ情報は、mysql.proc テーブルの param_list カラムの内容と類似しています。

INFORMATION_SCHEMA 名	mysql.proc 名	備考
SPECIFIC_CATALOG		def
SPECIFIC_SCHEMA	db	ルーチンデータベース
SPECIFIC_NAME	name	ルーチン名
ORDINAL_POSITION		1、2、3、... はパラメータ用、0 は関数 RETURNS 句用
PARAMETER_MODE		IN、OUT、INOUT (RETURNS には NULL)
PARAMETER_NAME		パラメータ名 (RETURNS には NULL)
DATA_TYPE		COLUMNS テーブルの場合と同じ
CHARACTER_MAXIMUM_LENGTH		COLUMNS テーブルの場合と同じ
CHARACTER_OCTET_LENGTH		COLUMNS テーブルの場合と同じ
NUMERIC_PRECISION		COLUMNS テーブルの場合と同じ
NUMERIC_SCALE		COLUMNS テーブルの場合と同じ
DATETIME_PRECISION		COLUMNS テーブルの場合と同じ
CHARACTER_SET_NAME		COLUMNS テーブルの場合と同じ
COLLATION_NAME		COLUMNS テーブルの場合と同じ
DTD_IDENTIFIER		COLUMNS テーブルの場合と同じ
ROUTINE_TYPE	type	ROUTINES テーブルの場合と同じ

注:

- ストアドプロシージャーとストアドファンクションの連続したパラメータについては、ORDINAL_POSITION 値は 1、2、3 などです。ストアドファンクションについては、RETURNS 句のデータ型について説明している行もあります。戻り値は真のパラメータではないので、これについて説明している行には次の一意の特性があります。
  - ORDINAL_POSITION 値は 0 です。
  - 戻り値には名前がなく、モードが適用されないので、PARAMETER_NAME 値と PARAMETER_MODE 値は NULL です。
  - MySQL 5.6.4 で DATETIME_PRECISION が追加されました。

21.13 INFORMATION_SCHEMA PARTITIONS テーブル

PARTITIONS テーブルは、テーブルパーティションに関する情報を提供します。テーブルのパーティション化に関する詳細は、第19章「パーティション化」を参照してください。

INFORMATION_SCHEMA 名	SHOW 名	備考
TABLE_CATALOG		MySQL 拡張
TABLE_SCHEMA		MySQL 拡張
TABLE_NAME		MySQL 拡張
PARTITION_NAME		MySQL 拡張
SUBPARTITION_NAME		MySQL 拡張
PARTITION_ORDINAL_POSITION		MySQL 拡張
SUBPARTITION_ORDINAL_POSITION		MySQL 拡張
PARTITION_METHOD		MySQL 拡張
SUBPARTITION_METHOD		MySQL 拡張
PARTITION_EXPRESSION		MySQL 拡張
SUBPARTITION_EXPRESSION		MySQL 拡張
PARTITION_DESCRIPTION		MySQL 拡張
TABLE_ROWS		MySQL 拡張
AVG_ROW_LENGTH		MySQL 拡張
DATA_LENGTH		MySQL 拡張
MAX_DATA_LENGTH		MySQL 拡張
INDEX_LENGTH		MySQL 拡張
DATA_FREE		MySQL 拡張
CREATE_TIME		MySQL 拡張
UPDATE_TIME		MySQL 拡張
CHECK_TIME		MySQL 拡張
CHECKSUM		MySQL 拡張
PARTITION_COMMENT		MySQL 拡張
NODEGROUP		MySQL 拡張
TABLESPACE_NAME		MySQL 拡張

注:

- PARTITIONS テーブルは、非標準テーブルです。

  このテーブルの各レコードは、パーティション化されたテーブルの個々のパーティションまたはサブパーティションに対応しています。

- TABLE_CATALOG: このカラムは常に def です。
- TABLE_SCHEMA: このカラムはテーブルが属しているデータベースの名前を含みます。
- TABLE_NAME: このカラムはパーティションを含むテーブルの名前を含みます。

- PARTITION_NAME: パーティションの名前です。
- SUBPARTITION_NAME: PARTITIONS テーブルレコードがサブパーティションを表す場合、このカラムはサブパーティションの名前を含みます。そうでない場合は、NULL になります。
- PARTITION_ORDINAL_POSITION: すべてのパーティションは、定義されたときと同じ順序でインデックスが付けられます。最初のパーティションに 1 の番号が割り当てられます。インデックス設定は、パーティションが追加、削除、再編成されると変わることがあります。このカラムに表示された番号は、インデックス設定の変更を考慮した現在の順序を表します。
- SUBPARTITION_ORDINAL_POSITION: テーブル内でパーティションにインデックスを付ける場合と同様に、指定されたパーティション内のサブパーティションにもインデックスの設定および変更が行われます。
- PARTITION_METHOD: RANGE、LIST、HASH、LINEAR HASH、KEY、または LINEAR KEY のいずれかの値です。つまり、セクション19.2「パーティショニングタイプ」で説明しているように、利用可能ないずれかのパーティショニングタイプになります。
- SUBPARTITION_METHOD: HASH、LINEAR HASH、KEY、または LINEAR KEY のいずれかの値です。つまり、セクション19.2.6「サブパーティショニング」で説明しているように、利用可能ないずれかのサブパーティショニングタイプになります。
- PARTITION_EXPRESSION: これは、テーブルの現在のパーティション化スキームを作成した CREATE TABLE または ALTER TABLE ステートメントで使用されたパーティション化関数の式です。

  たとえば、次のステートメントを使用して test データベースに作成されたパーティション化されたテーブルを考えてみます。

```
CREATE TABLE tp (
    c1 INT,
    c2 INT,
    c3 VARCHAR(25)
)
PARTITION BY HASH(c1 + c2)
PARTITIONS 4;
```

  このテーブルからのパーティションに対する PARTITIONS テーブルレコード内の PARTITION_EXPRESSION カラムには、次に示すように c1 + c2 と表示されます。

```
mysql> SELECT DISTINCT PARTITION_EXPRESSION
    >     FROM INFORMATION_SCHEMA.PARTITIONS
    >     WHERE TABLE_NAME='tp' AND TABLE_SCHEMA='test';
+----------------------+
| PARTITION_EXPRESSION |
+----------------------+
| c1 + c2              |
+----------------------+
1 row in set (0.09 sec)
```

- SUBPARTITION_EXPRESSION: これは、テーブルのパーティション化を定義するために使用されるパーティショニング式に対する PARTITION_EXPRESSION の動作と同様に、サブパーティション化を定義するサブパーティショニング式に対して動作します。

  テーブルにサブパーティションがない場合、このカラムは NULL になります。
- PARTITION_DESCRIPTION: このカラムは RANGE および LIST パーティションに使用されます。RANGE パーティションの場合、パーティションの VALUES LESS THAN 句で設定された値が含まれます。これには整数か MAXVALUE のどちらかを指定できます。LIST パーティションの場合、このカラムには、パーティションの VALUES IN 句で定義された値が含まれます。これは整数値のカンマ区切りのリストになります。

  PARTITION_METHOD が RANGE または LIST 以外であるパーティションの場合、このカラムは常に NULL になります。
- TABLE_ROWS: パーティションのテーブル行の数です。

  パーティション化された InnoDB テーブルの場合、TABLE_ROWS カラムで指定された行数は、SQL 最適化で使用される推定値に過ぎず、必ずしも正確であるとはかぎりません。
- AVG_ROW_LENGTH: このパーティションまたはサブパーティションに格納された行の、バイトで表した平均長です。

  これは TABLE_ROWS で分割された DATA_LENGTH と同じです。

- DATA_LENGTH: パーティションまたはサブパーティションに格納されたすべての行の、バイトで表された合計長です。つまり、パーティションまたはサブパーティションに格納されたバイトの合計数です。
- MAX_DATA_LENGTH: このパーティションまたはサブパーティションに格納できる最大のバイト数です。
- INDEX_LENGTH: このパーティションおよびサブパーティションのインデックスファイルの、バイトで表された長さです。
- DATA_FREE: パーティションまたはサブパーティションに割り当てられているが使用されていないバイト数です。
- CREATE_TIME: パーティションまたはサブパーティションが作成された時間です。
- UPDATE_TIME: パーティションまたはサブパーティションが最後に変更された時間です。
- CHECK_TIME: パーティションまたはサブパーティションが属しているテーブルが最後に確認された時間です。

  > **注記**
  >
  > 一部のストレージエンジンは、この時間を更新しません。これらのストレージエンジンを使用したテーブルの場合、この値は常に NULL です。

- CHECKSUM: チェックサム値になります (ある場合)。ない場合は、このカラムは NULL になります。
- PARTITION_COMMENT: このカラムには、パーティションに対して行われたコメントのテキストが含まれます。

  MySQL 5.6.6 より前では、このカラムの表示幅は 80 文字でしたが、この長さを超えたパーティションコメントはこの文字数まで切り捨てられていました。MySQL 5.6.6 以降では、パーティションコメントの最大長は 1024 文字に定義され、PARTITION_COMMENT カラムの表示幅はこの上限に一致するように 1024 文字に増やされています (Bug #11748924、Bug #37728)。

  このカラムのデフォルト値は空の文字列です。
- NODEGROUP: これは、パーティションが属しているノードグループです。これは、MySQL クラスタテーブルにのみ関連します。そうでない場合は、このカラムの値は常に 0 です。
- TABLESPACE_NAME: このカラムには、パーティションが属しているテーブルスペースの名前が含まれます。現在、このカラムの値は常に DEFAULT です。
- パーティション化されていないテーブルには、INFORMATION_SCHEMA.PARTITIONS に 1 つのレコードがあります。ただし、PARTITION_NAME、SUBPARTITION_NAME、PARTITION_ORDINAL_POSITION、SUBPARTITION_ORDINAL_POSITおよび PARTITION_DESCRIPTION カラムの値はすべて NULL になります。(この場合の PARTITION_COMMENT カラムはブランクです。)

21.14 INFORMATION_SCHEMA PLUGINS テーブル

PLUGINS テーブルは、サーバーのプラグインに関する情報を提供します。

INFORMATION_SCHEMA 名	SHOW 名	備考
PLUGIN_NAME	Name	MySQL 拡張
PLUGIN_VERSION		MySQL 拡張
PLUGIN_STATUS	Status	MySQL 拡張
PLUGIN_TYPE	Type	MySQL 拡張
PLUGIN_TYPE_VERSION		MySQL 拡張
PLUGIN_LIBRARY	Library	MySQL 拡張
PLUGIN_LIBRARY_VERSION		MySQL 拡張
PLUGIN_AUTHOR		MySQL 拡張
PLUGIN_DESCRIPTION		MySQL 拡張
PLUGIN_LICENSE	License	MySQL 拡張
LOAD_OPTION		MySQL 拡張

注:

- PLUGINS テーブルは、非標準テーブルです。
- PLUGIN_NAME は、INSTALL PLUGIN や UNINSTALL PLUGIN などのステートメントでプラグインを参照するために使用される名前です。
- PLUGIN_VERSION は、プラグインの一般的なタイプのディスクリプタのバージョンです。
- PLUGIN_STATUS は、ACTIVE、INACTIVE、DISABLED、または DELETED のいずれかのプラグインステータスを示します。
- PLUGIN_TYPE は、STORAGE ENGINE、INFORMATION_SCHEMA、AUTHENTICATION などのプラグインのタイプを示します。
- PLUGIN_TYPE_VERSION は、プラグインのタイプ固有のディスクリプタのバージョンです。
- PLUGIN_LIBRARY は、プラグイン共有オブジェクトファイルの名前です。これは、INSTALL PLUGIN や UNINSTALL PLUGIN などのステートメントでプラグインファイルを参照するために使用される名前です。このファイルは、plugin_dir システム変数によって指名されたディレクトリに置かれます。ライブラリ名が NULL である場合、プラグインはコンパイルされますが、UNINSTALL PLUGIN でアンインストールできません。
- PLUGIN_LIBRARY_VERSION は、プラグイン API インタフェースバージョンを示します。
- PLUGIN_AUTHOR はプラグイン作成者の名前を示します。
- PLUGIN_DESCRIPTION は、プラグインの簡単な説明を提供します。
- PLUGIN_LICENSE は、GPL などのライセンスがどのようにプラグインに付与されているかを示します。
- LOAD_OPTION は、プラグインがどのようにロードされたかを示します。値は、OFF、ON、FORCE、または FORCE_PLUS_PERMANENT です。セクション5.1.8.1「プラグインのインストールおよびアンインストール」を参照してください。

INSTALL PLUGIN でインストールされたプラグインの場合、PLUGIN_NAME および PLUGIN_LIBRARY 値は、mysql.plugin テーブルにも登録されます。

次のステートメントは同等です。

```
SELECT
  PLUGIN_NAME, PLUGIN_STATUS, PLUGIN_TYPE,
  PLUGIN_LIBRARY, PLUGIN_LICENSE
FROM INFORMATION_SCHEMA.PLUGINS;

SHOW PLUGINS;
```

PLUGINS テーブル内の情報のベースを形成するプラグインデータ構造の詳細は、セクション24.2「MySQL プラグイン API」を参照してください。

プラグイン情報は、SHOW PLUGINS ステートメントを使用して利用することもできます。セクション 13.7.5.26「SHOW PLUGINS 構文」を参照してください。

21.15 INFORMATION_SCHEMA PROCESSLIST テーブル

PROCESSLIST テーブルは、どのスレッドが動作しているかに関する情報を提供します。

INFORMATION_SCHEMA 名	SHOW 名	備考
ID	Id	MySQL 拡張
USER	User	MySQL 拡張
HOST	Host	MySQL 拡張
DB	db	MySQL 拡張
COMMAND	Command	MySQL 拡張
TIME	Time	MySQL 拡張
STATE	State	MySQL 拡張

INFORMATION_SCHEMA 名	SHOW 名	備考
INFO	Info	MySQL 拡張

テーブルカラムの幅広い説明については、セクション13.7.5.30「SHOW PROCESSLIST 構文」を参照してください。

注:

- PROCESSLIST テーブルは、非標準テーブルです。
- 対応する SHOW ステートメントの出力と同様に、PROCESSLIST テーブルは PROCESS の権限がない場合は、自身のスレッドに関する情報だけが表示されます。権限がある場合、ほかのスレッドに関する情報も表示されます。匿名ユーザーとしてでは、どの行も表示できません。
- SQL ステートメントが INFORMATION_SCHEMA.PROCESSLIST を参照している場合、MySQL はステートメントの実行が開始されるとテーブル全体を一度移入するので、ステートメント中の読み取りの一貫性が保たれます。ただし、複数ステートメントトランザクションでは読み取りの一貫性はありません。
- 処理情報は、performance_schema.threads テーブルから利用することもできます。ただし、threads へのアクセスには相互排他ロックは必要なく、サーバーパフォーマンスへの影響は最小です。INFORMATION_SCHEMA.PROCESSLIST および SHOW PROCESSLIST は、相互排他ロックを必要とするので、負のパフォーマンスの結果になります。threads はまた、バックグラウンドスレッドに関する情報も表示しますが、INFORMATION_SCHEMA.PROCESSLIST および SHOW PROCESSLIST は表示しません。これは、threads は、ほかのスレッド情報源では行えないアクティビティーのモニターに使用できることを意味します。

次のステートメントは同等です。

```
SELECT * FROM INFORMATION_SCHEMA.PROCESSLIST

SHOW FULL PROCESSLIST
```

21.16 INFORMATION_SCHEMA PROFILING テーブル

PROFILING テーブルは、ステートメントプロファイリング情報を提供します。その内容は、SHOW PROFILES および SHOW PROFILE ステートメントで生成される情報に対応します (セクション13.7.5.32「SHOW PROFILES 構文」を参照してください)。profiling セッション変数は 1 に設定されないかぎり、テーブルは空です。

INFORMATION_SCHEMA 名	SHOW 名	備考
QUERY_ID	Query_ID	
SEQ		
STATE	Status	
DURATION	Duration	
CPU_USER	CPU_user	
CPU_SYSTEM	CPU_system	
CONTEXT_VOLUNTARY	Context_voluntary	
CONTEXT_INVOLUNTARY	Context_involuntary	
BLOCK_OPS_IN	Block_ops_in	
BLOCK_OPS_OUT	Block_ops_out	
MESSAGES_SENT	Messages_sent	
MESSAGES_RECEIVED	Messages_received	
PAGE_FAULTS_MAJOR	Page_faults_major	
PAGE_FAULTS_MINOR	Page_faults_minor	
SWAPS	Swaps	
SOURCE_FUNCTION	Source_function	
SOURCE_FILE	Source_file	
SOURCE_LINE	Source_line	

注:

- QUERY_ID は数値のステートメント識別子です。
- SEQ は、同じ QUERY_ID 値を持つ行の表示順を示すシーケンス番号です。
- STATE は、行の測定が適用されるプロファイリング状態です。
- DURATION は、ステートメントの実行が、一定の状態に持続していた時間を秒単位で示します。
- CPU_USER および CPU_SYSTEM はユーザーおよびシステムの CPU 使用率を秒単位で示します。
- CONTEXT_VOLUNTARY および CONTEXT_INVOLUNTARY は、自発的および強制的コンテキストスイッチが行われた回数を示します。
- BLOCK_OPS_IN および BLOCK_OPS_OUT は、ブロック入力および出力操作の数を示します。
- MESSAGES_SENT および MESSAGES_RECEIVED は、送受信された通信メッセージの数を示します。
- PAGE_FAULTS_MAJOR および PAGE_FAULTS_MINOR は、メジャーおよびマイナーページフォルトの数を示します。
- SWAPS はスワップが行われた数を示します。
- SOURCE_FUNCTION、SOURCE_FILE、および SOURCE_LINE は、プロファイルされた状態で実行されるソースコード内の箇所を示す情報を提供します。

21.17 INFORMATION_SCHEMA REFERENTIAL_CONSTRAINTS テーブル

REFERENTIAL_CONSTRAINTS テーブルは、外部キーに関する情報を提供します。

INFORMATION_SCHEMA 名	SHOW 名	備考
CONSTRAINT_CATALOG		def
CONSTRAINT_SCHEMA		
CONSTRAINT_NAME		
UNIQUE_CONSTRAINT_CATALOG		def
UNIQUE_CONSTRAINT_SCHEMA		
UNIQUE_CONSTRAINT_NAME		
MATCH_OPTION		
UPDATE_RULE		
DELETE_RULE		
TABLE_NAME		
REFERENCED_TABLE_NAME		

注:

- TABLE_NAME の値は、INFORMATION_SCHEMA.TABLE_CONSTRAINTS の TABLE_NAME と同じです。
- CONSTRAINT_SCHEMA および CONSTRAINT_NAME は外部キーを識別します。
- UNIQUE_CONSTRAINT_SCHEMA、UNIQUE_CONSTRAINT_NAME、および REFERENCED_TABLE_NAME は参照キーを識別します。
- このとき MATCH_OPTION の唯一有効な値は NONE です。
- UPDATE_RULE または DELETE_RULE に指定可能な値は、CASCADE、SET NULL、SET DEFAULT、RESTRICT、NO ACTION です。

21.18 INFORMATION_SCHEMA ROUTINES テーブル

ROUTINES テーブルはストアドルーチン (プロシージャーおよび関数の両方) に関する情報を提供します。ROUTINES テーブルには、ユーザー定義関数 (UDF) は含まれません。

「mysql.proc 名」は、存在する場合、INFORMATION_SCHEMA.ROUTINES テーブルカラムに対応する mysql.proc テーブルカラムを示します。

INFORMATION_SCHEMA 名	mysql.proc 名	備考
SPECIFIC_NAME	specific_name	
ROUTINE_CATALOG		def
ROUTINE_SCHEMA	db	
ROUTINE_NAME	name	
ROUTINE_TYPE	type	{PROCEDURE|FUNCTION}
DATA_TYPE		COLUMNS テーブルの場合と同じ
CHARACTER_MAXIMUM_LENGTH		COLUMNS テーブルの場合と同じ
CHARACTER_OCTET_LENGTH		COLUMNS テーブルの場合と同じ
NUMERIC_PRECISION		COLUMNS テーブルの場合と同じ
NUMERIC_SCALE		COLUMNS テーブルの場合と同じ
DATETIME_PRECISION		COLUMNS テーブルの場合と同じ
CHARACTER_SET_NAME		COLUMNS テーブルの場合と同じ
COLLATION_NAME		COLUMNS テーブルの場合と同じ
DTD_IDENTIFIER		データ型ディスクリプタ
ROUTINE_BODY		SQL
ROUTINE_DEFINITION	body_utf8	
EXTERNAL_NAME		NULL
EXTERNAL_LANGUAGE	language	NULL
PARAMETER_STYLE		SQL
IS_DETERMINISTIC	is_deterministic	
SQL_DATA_ACCESS	sql_data_access	
SQL_PATH		NULL
SECURITY_TYPE	security_type	
CREATED	created	
LAST_ALTERED	modified	
SQL_MODE	sql_mode	MySQL 拡張
ROUTINE_COMMENT	comment	MySQL 拡張
DEFINER	definer	MySQL 拡張
CHARACTER_SET_CLIENT		MySQL 拡張
COLLATION_CONNECTION		MySQL 拡張
DATABASE_COLLATION		MySQL 拡張

注:

- MySQL は EXTERNAL_LANGUAGE を次のように計算します。
  - mysql.proc.language='SQL' の場合、EXTERNAL_LANGUAGE は NULL になります
  - それ以外の場合、EXTERNAL_LANGUAGE は mysql.proc.language に存在します。ただし、まだ外部言語がないので、これは常に NULL になります。

- CREATED: ルーチンが作成された日時です。これは TIMESTAMP 値です。
- LAST_ALTERED: ルーチンが最後に変更された日時です。これは TIMESTAMP 値です。ルーチンが作成されてから変更されなかった場合、このカラムは CREATED カラムと同じ値を保持します。
- SQL_MODE: ルーチンが作成または変更されたときに有効であり、ルーチンを実行するときの SQL モードです。指定可能な値については、セクション5.1.7「サーバー SQL モード」を参照してください。
- CHARACTER_SET_CLIENT: ルーチンが作成されたときの character_set_client システム変数のセッション値です。
- COLLATION_CONNECTION: ルーチンが作成されたときの collation_connection システム変数のセッション値です。
- DATABASE_COLLATION: ルーチンが関連付けられているデータベースの照合順序です。
- DATA_TYPE、CHARACTER_MAXIMUM_LENGTH、CHARACTER_OCTET_LENGTH、NUMERIC_PRECISION、NUME および COLLATION_NAME カラムは、ストアドファンクションの RETURNS 句に対するデータ型に関する情報を提供します。ストアドルーチンがストアドプロシージャーである場合、これらのカラムはすべて NULL になります。MySQL 5.6.4 で DATETIME_PRECISION が追加されました。
- ストアドファンクション RETURNS データ型に関する情報は、PARAMETERS テーブルでも利用できます。関数に対する戻り値のデータ型の行は、ORDINAL_POSITION 値が 0 である行として識別できます。

21.19 INFORMATION_SCHEMA SCHEMATA テーブル

スキーマはデータベースなので、SCHEMATA テーブルはデータベースに関する情報を提供します。

INFORMATION_SCHEMA 名	SHOW 名	備考
CATALOG_NAME		def
SCHEMA_NAME	Database	
DEFAULT_CHARACTER_SET_NAME		
DEFAULT_COLLATION_NAME		
SQL_PATH		NULL

次のステートメントは同等です。

```
SELECT SCHEMA_NAME AS `Database`
  FROM INFORMATION_SCHEMA.SCHEMATA
  [WHERE SCHEMA_NAME LIKE 'wild']

SHOW DATABASES
  [LIKE 'wild']
```

21.20 INFORMATION_SCHEMA SCHEMA_PRIVILEGES テーブル

SCHEMA_PRIVILEGES テーブルは、スキーマ (データベース) 権限に関する情報を提供します。この情報は mysql.db 付与テーブルにあります。

INFORMATION_SCHEMA 名	SHOW 名	備考
GRANTEE		'user_name'@'host_name' 値、MySQL 拡張
TABLE_CATALOG		def、MySQL 拡張
TABLE_SCHEMA		MySQL 拡張
PRIVILEGE_TYPE		MySQL 拡張
IS_GRANTABLE		MySQL 拡張

注:

- これは非標準テーブルです。その値を mysql.db テーブルから取得します。

21.21 INFORMATION_SCHEMA STATISTICS テーブル

STATISTICS テーブルは、テーブルインデックスの情報を提供します。

INFORMATION_SCHEMA 名	SHOW 名	備考
TABLE_CATALOG		def
TABLE_SCHEMA		= Database
TABLE_NAME	Table	
NON_UNIQUE	Non_unique	
INDEX_SCHEMA		= Database
INDEX_NAME	Key_name	
SEQ_IN_INDEX	Seq_in_index	
COLUMN_NAME	Column_name	
COLLATION	Collation	
CARDINALITY	Cardinality	
SUB_PART	Sub_part	MySQL 拡張
PACKED	Packed	MySQL 拡張
NULLABLE	Null	MySQL 拡張
INDEX_TYPE	Index_type	MySQL 拡張
COMMENT	Comment	MySQL 拡張

注:

- インデックスには標準のテーブルはありません。前述のリストは SQL Server 2000 が sp_statistics に対して返すものと似ていますが、名前 QUALIFIER を CATALOG に、名前 OWNER を SCHEMA に置き換えた点が異なります。

  明らかに、前述のテーブルと、SHOW INDEX の出力は同じ親から派生したものです。このため、すでに密接な相関関係になっています。

次のステートメントは同等です。

```
SELECT * FROM INFORMATION_SCHEMA.STATISTICS
  WHERE table_name = 'tbl_name'
  AND table_schema = 'db_name'

SHOW INDEX
  FROM tbl_name
  FROM db_name
```

21.22 INFORMATION_SCHEMA TABLES テーブル

TABLES テーブルは、データベース内のテーブルに関する情報を提供します。

INFORMATION_SCHEMA 名	SHOW 名	備考
TABLE_CATALOG		def
TABLE_SCHEMA	Table_...	
TABLE_NAME	Table_...	
TABLE_TYPE		
ENGINE	Engine	MySQL 拡張
VERSION	Version	テーブルの .frm ファイルのバージョン番号、MySQL 拡張
ROW_FORMAT	Row_format	MySQL 拡張
TABLE_ROWS	Rows	MySQL 拡張
AVG_ROW_LENGTH	Avg_row_length	MySQL 拡張
DATA_LENGTH	Data_length	MySQL 拡張
MAX_DATA_LENGTH	Max_data_length	MySQL 拡張

INFORMATION_SCHEMA 名	SHOW 名	備考
INDEX_LENGTH	Index_length	MySQL 拡張
DATA_FREE	Data_free	MySQL 拡張
AUTO_INCREMENT	Auto_increment	MySQL 拡張
CREATE_TIME	Create_time	MySQL 拡張
UPDATE_TIME	Update_time	MySQL 拡張
CHECK_TIME	Check_time	MySQL 拡張
TABLE_COLLATION	Collation	MySQL 拡張
CHECKSUM	Checksum	MySQL 拡張
CREATE_OPTIONS	Create_options	MySQL 拡張
TABLE_COMMENT	Comment	MySQL 拡張

注:

- TABLE_SCHEMA および TABLE_NAME は、Table_in_db1 など、SHOW 表示の単一のフィールドです。
- TABLE_TYPE は BASE TABLE または VIEW になります。現在、TABLES テーブルは TEMPORARY テーブルをリストしません。
- パーティション化されたテーブルの場合、ENGINE カラムには、すべてのパーティションで使用されるストレージエンジンの名前が表示されます。(以前は、このカラムには、このようなテーブルに対し PARTITION が表示されていました。)
- テーブルが INFORMATION_SCHEMA データベースにある場合、TABLE_ROWS カラムは NULL です。

  InnoDB テーブルの場合、行カウントは SQL 最適化で使用される単なる概算です。(InnoDB テーブルがパーティション化されている場合も、これは当てはまります。)
- DATA_FREE カラムには、InnoDB テーブルの空き領域がバイト単位で表示されます。
- テーブルのデフォルトの文字セットはありません。照合順序名は文字セット名で始まるため、TABLE_COLLATION は閉じています。
- テーブルがパーティション化されている場合、CREATE_OPTIONS カラムには partitioned と表示されます。

次のステートメントは同等です。

```
SELECT table_name FROM INFORMATION_SCHEMA.TABLES
  WHERE table_schema = 'db_name'
  [AND table_name LIKE 'wild']

SHOW TABLES
  FROM db_name
  [LIKE 'wild']
```

21.23 INFORMATION_SCHEMA TABLESPACES テーブル

TABLESPACES テーブルは、アクティブなテーブルスペースに関する情報を提供します。

INFORMATION_SCHEMA 名	SHOW 名	備考
TABLESPACE_NAME		MySQL 拡張
ENGINE		MySQL 拡張
TABLESPACE_TYPE		MySQL 拡張
LOGFILE_GROUP_NAME		MySQL 拡張
EXTENT_SIZE		MySQL 拡張
AUTOEXTEND_SIZE		MySQL 拡張
MAXIMUM_SIZE		MySQL 拡張
NODEGROUP_ID		MySQL 拡張
TABLESPACE_COMMENT		MySQL 拡張

21.24 INFORMATION_SCHEMA TABLE_CONSTRAINTS テーブル

TABLE_CONSTRAINTS テーブルは、どのテーブルに制約があるかを説明します。

INFORMATION_SCHEMA 名	SHOW 名	備考
CONSTRAINT_CATALOG		def
CONSTRAINT_SCHEMA		
CONSTRAINT_NAME		
TABLE_SCHEMA		
TABLE_NAME		
CONSTRAINT_TYPE		

注:

- CONSTRAINT_TYPE 値は、UNIQUE、PRIMARY KEY、または FOREIGN KEY になります。
- UNIQUE および PRIMARY KEY 情報は、Non_unique フィールドが 0 のときに SHOW INDEX の出力の Key_name フィールドから得られる情報とほぼ同じになります。
- CONSTRAINT_TYPE カラムには、UNIQUE、PRIMARY KEY、FOREIGN KEY、CHECK のいずれかの値を含められます。これは CHAR (非 ENUM) カラムです。CHECK 値は CHECK をサポートするまでは利用できません。

21.25 INFORMATION_SCHEMA TABLE_PRIVILEGES テーブル

TABLE_PRIVILEGES テーブルは、テーブル権限に関する情報を提供します。この情報は mysql.tables_priv 付与テーブルからのものです。

INFORMATION_SCHEMA 名	SHOW 名	備考
GRANTEE		'user_name'@'host_name' 値
TABLE_CATALOG		def
TABLE_SCHEMA		
TABLE_NAME		
PRIVILEGE_TYPE		
IS_GRANTABLE		

注:

- PRIVILEGE_TYPE は、SELECT、INSERT、UPDATE、REFERENCES、ALTER、INDEX、DROP、CREATE VIEW の値のいずれか (1 つのみ) を含むことができます。

次のステートメントは同等ではありません。

```
SELECT ... FROM INFORMATION_SCHEMA.TABLE_PRIVILEGES

SHOW GRANTS ...
```

21.26 INFORMATION_SCHEMA TRIGGERS テーブル

TRIGGERS テーブルはトリガーに関する情報を提供します。自身が TRIGGER 権限を保有しているデータベースとテーブルの情報だけを表示できます。

INFORMATION_SCHEMA 名	SHOW 名	備考
TRIGGER_CATALOG		def
TRIGGER_SCHEMA		
TRIGGER_NAME	Trigger	
EVENT_MANIPULATION	Event	

INFORMATION_SCHEMA 名	SHOW 名	備考
EVENT_OBJECT_CATALOG		def
EVENT_OBJECT_SCHEMA		
EVENT_OBJECT_TABLE	Table	
ACTION_ORDER		0
ACTION_CONDITION		NULL
ACTION_STATEMENT	Statement	
ACTION_ORIENTATION		ROW
ACTION_TIMING	Timing	
ACTION_REFERENCE_OLD_TABLE		NULL
ACTION_REFERENCE_NEW_TABLE		NULL
ACTION_REFERENCE_OLD_ROW		OLD
ACTION_REFERENCE_NEW_ROW		NEW
CREATED	Created	
SQL_MODE	sql_mode	MySQL 拡張
DEFINER	Definer	MySQL 拡張
CHARACTER_SET_CLIENT	character_set_client	MySQL 拡張
COLLATION_CONNECTION	collation_connection	MySQL 拡張
DATABASE_COLLATION	Database Collation	MySQL 拡張

注:

- 「SHOW 名」カラム内の名前は、SHOW CREATE TRIGGER ではなく SHOW TRIGGERS ステートメントを示します。セクション13.7.5.39「SHOW TRIGGERS 構文」を参照してください。
- TRIGGER_SCHEMA および TRIGGER_NAME: それぞれ、トリガーが実行されるデータベースの名前とトリガー名です。
- EVENT_MANIPULATION: トリガーイベントです。これは、トリガーが有効になる、関連付けられたテーブルに対する操作の種類です。値は、'INSERT' (行が挿入された場合)、'DELETE' (行が削除された場合)、または 'UPDATE' (行が変更された場合) です。
- EVENT_OBJECT_SCHEMA および EVENT_OBJECT_TABLE: セクション20.3「トリガーの使用」に記したように、すべてのトリガーは厳密に 1 つのテーブルに関連付けられます。これらのカラムには、このテーブルが存在するデータベースと、テーブル名がそれぞれ示されます。
- ACTION_ORDER: 同じテーブルで類似したトリガーのリスト内の、トリガーアクションの順序位置です。現在、この値は常に 0 です。これは、同じテーブルで同じ EVENT_MANIPULATION および ACTION_TIMING のトリガーを複数持つことはできないためです。
- ACTION_STATEMENT: トリガー本体です。つまり、トリガーの有効時に実行されるステートメントです。このテキストは UTF-8 エンコーディングを使用します。
- ACTION_ORIENTATION: 常に 'ROW' の値を含みます。
- ACTION_TIMING: トリガーが有効になるのが、トリガーイベントの前かあとかを指定します。値は 'BEFORE' または 'AFTER' です。
- ACTION_REFERENCE_OLD_ROW および ACTION_REFERENCE_NEW_ROW: それぞれ古いカラム識別子と新しいカラム識別子です。このことは、ACTION_REFERENCE_OLD_ROW が常に 'OLD' の値を含み、ACTION_REFERENCE_NEW_ROW が常に 'NEW' の値を含むことを意味します。
- SQL_MODE: トリガーが作成されたときに有効であり、トリガーを実行するときの SQL モードです。指定可能な値については、セクション5.1.7「サーバー SQL モード」を参照してください。
- DEFINER: 'user_name'@'host_name' 形式で示した、トリガーを作成したユーザーのアカウントです。
- CHARACTER_SET_CLIENT: トリガーが作成されたときの character_set_client システム変数のセッション値です。

- COLLATION_CONNECTION: トリガーが作成されたときの collation_connection システム変数のセッション値です。
- DATABASE_COLLATION: トリガーが関連付けられているデータベースの照合順序です。
- ACTION_CONDITION、ACTION_REFERENCE_OLD_TABLE、ACTION_REFERENCE_NEW_TABLE、および CREATED のカラムには現在、NULL が常に含まれます。

セクション20.3「トリガーの使用」で定義された ins_sum トリガーを使用した例です。

```
mysql> SELECT * FROM INFORMATION_SCHEMA.TRIGGERS
    -> WHERE TRIGGER_SCHEMA='test' AND TRIGGER_NAME='ins_sum'\G
*************************** 1. row ***************************
           TRIGGER_CATALOG: def
            TRIGGER_SCHEMA: test
              TRIGGER_NAME: ins_sum
        EVENT_MANIPULATION: INSERT
      EVENT_OBJECT_CATALOG: def
       EVENT_OBJECT_SCHEMA: test
        EVENT_OBJECT_TABLE: account
             ACTION_ORDER: 0
         ACTION_CONDITION: NULL
         ACTION_STATEMENT: SET @sum = @sum + NEW.amount
       ACTION_ORIENTATION: ROW
            ACTION_TIMING: BEFORE
ACTION_REFERENCE_OLD_TABLE: NULL
ACTION_REFERENCE_NEW_TABLE: NULL
  ACTION_REFERENCE_OLD_ROW: OLD
  ACTION_REFERENCE_NEW_ROW: NEW
                CREATED: NULL
              SQL_MODE: NO_ENGINE_SUBSTITUTION
                DEFINER: me@localhost
     CHARACTER_SET_CLIENT: utf8
     COLLATION_CONNECTION: utf8_general_ci
       DATABASE_COLLATION: latin1_swedish_ci
```

21.27 INFORMATION_SCHEMA USER_PRIVILEGES テーブル

USER_PRIVILEGES テーブルは、グローバル権限に関する情報を提供します。この情報は mysql.user 付与テーブルにあります。

INFORMATION_SCHEMA 名	SHOW 名	備考
GRANTEE		'user_name'@'host_name' 値、MySQL 拡張
TABLE_CATALOG		def、MySQL 拡張
PRIVILEGE_TYPE		MySQL 拡張
IS_GRANTABLE		MySQL 拡張

注:

- これは非標準テーブルです。この値は、mysql.user テーブルから取得されます。

21.28 INFORMATION_SCHEMA VIEWS テーブル

VIEWS テーブルは、データベース内のビューに関する情報を提供します。このテーブルにアクセスするには SHOW VIEW 権限が必要です。

INFORMATION_SCHEMA 名	SHOW 名	備考
TABLE_CATALOG		def
TABLE_SCHEMA		
TABLE_NAME		
VIEW_DEFINITION		
CHECK_OPTION		
IS_UPDATABLE		

INFORMATION_SCHEMA 名	SHOW 名	備考
DEFINER		
SECURITY_TYPE		
CHARACTER_SET_CLIENT		MySQL 拡張
COLLATION_CONNECTION		MySQL 拡張

注:

- VIEW_DEFINITION カラムには、SHOW CREATE VIEW が生成する Create Table フィールドに表示されるほとんどのものが存在します。SELECT より前の語をスキップし、WITH CHECK OPTION の語をスキップします。元のステートメントが次のとおりだったとします。

```
CREATE VIEW v AS
  SELECT s2,s1 FROM t
  WHERE s1 > 5
  ORDER BY s1
  WITH CHECK OPTION;
```

  この場合、ビュー定義は次のようになります。

```
SELECT s2,s1 FROM t WHERE s1 > 5 ORDER BY s1
```

- CHECK_OPTION カラムは、NONE、CASCADE、または LOCAL の値になります。
- MySQL は、CREATE VIEW 時に、ビューの更新可能性フラグというフラグを設定します。UPDATE および DELETE (および同様の操作) がビューで有効な場合、フラグは YES (true) に設定されます。それ以外の場合、フラグは NO (false) に設定されます。VIEWS テーブルの IS_UPDATABLE カラムは、このフラグのステータスを表示します。これは、ビューが更新可能であるかどうかをサーバーが常に把握していることを意味します。ビューが更新可能ではない場合、UPDATE、DELETE、INSERT などのステートメントは無効であり、拒否されます。(ビューが更新可能な場合でも、挿入できない場合があります。詳細は、セクション13.1.20「CREATE VIEW 構文」を参照してください。)
- DEFINER: 'user_name'@'host_name' 形式で示した、ビューを作成したユーザーのアカウントです。SECURITY_TYPE には、DEFINER または INVOKER の値があります。
- CHARACTER_SET_CLIENT: ビューが作成されたときの character_set_client システム変数のセッション値です。
- COLLATION_CONNECTION: ビューが作成されたときの collation_connection システム変数のセッション値です。

MySQL では、異なる sql_mode 設定を使用すると、サポートする SQL 構文のタイプをサーバーに指示できます。たとえば、ANSI SQL モードを使用すると、クエリーで、MySQL で標準 SQL 連結演算子の二重バー (||) が正しく解釈されます。その後、項目を連結するビューを作成した場合、sql_mode 設定を ANSI とは別の値に変更すると、そのビューが無効になるという懸念がある場合があります。ただし、そのようなことはありません。MySQL は、記述方法には関係なく、常にビュー定義を正規の形式で同じ方法で格納します。サーバーが二重バーの連結演算子を CONCAT() 関数にどのように変更するかを示す例を次に示します。

```
mysql> SET sql_mode = 'ANSI';
Query OK, 0 rows affected (0.00 sec)

mysql> CREATE VIEW test.v AS SELECT 'a' || 'b' as col1;
Query OK, 0 rows affected (0.00 sec)

mysql> SELECT VIEW_DEFINITION FROM INFORMATION_SCHEMA.VIEWS
    -> WHERE TABLE_SCHEMA = 'test' AND TABLE_NAME = 'v';
+----------------------------------+
| VIEW_DEFINITION                  |
+----------------------------------+
| select concat('a','b') AS `col1` |
+----------------------------------+
1 row in set (0.00 sec)
```

ビュー定義を正規の形式で格納する利点は、あとで sql_mode の値を変更しても、ビューの結果に影響を与えないことにあります。ただし、SELECT の前にあるコメントが、サーバーによって定義から取り除かれるというその他の影響があります。

21.29 InnoDB の INFORMATION_SCHEMA テーブル

このセクションでは、InnoDB INFORMATION_SCHEMA テーブルのテーブル定義について説明します。関連する情報と例については、セクション14.14「InnoDB INFORMATION_SCHEMA テーブル」を参照してください。

InnoDB INFORMATION_SCHEMA テーブルは、進行中の InnoDB アクティビティーのモニタリング、問題になる前の非効率性の検出、またはパフォーマンスおよび容量の問題のトラブルシューティングに使用できます。データベースが増大しさらにビジーになり、ハードウェアの容量の限度に達したときに、データベースがスムーズに実行し続けられるようにこれらの点をモニターし調整します。

21.29.1 INFORMATION_SCHEMA INNODB_CMP および INNODB_CMP_RESET テーブル

INNODB_CMP および INNODB_CMP_RESET テーブルには、圧縮 InnoDB テーブルに関連した操作に関するステータス情報が含まれます。

表 21.1 INNODB_CMP および INNODB_CMP_RESET のカラム

カラム名	説明
PAGE_SIZE	圧縮ページのサイズ (バイト単位)。
COMPRESS_OPS	サイズ PAGE_SIZE の B ツリーページが圧縮された回数。空のページが作成されたり、非圧縮変更ログ用の領域が不足したりするたびに、ページが圧縮されます。
COMPRESS_OPS_OK	サイズ PAGE_SIZE の B ツリーページの圧縮が成功した回数。このカウントが COMPRESS_OPS を超えることは決してありません。
COMPRESS_TIME	サイズ PAGE_SIZE の B ツリーページの圧縮への試行で費やされた合計時間 (秒単位)。
UNCOMPRESS_OPS	サイズ PAGE_SIZE の B ツリーページが圧縮解除された回数。圧縮が失敗したときや、圧縮解除されたページがバッファープールに存在しない場合に最初にアクセスするときに、B ツリーページは圧縮解除されます。
UNCOMPRESS_TIME	サイズ PAGE_SIZE の B ツリーページの圧縮解除に費やされた合計時間 (秒単位)。

例:

```
mysql> SELECT * FROM INFORMATION_SCHEMA.INNODB_CMP \G
*************************** 1. row ***************************
      page_size: 1024
   compress_ops: 0
compress_ops_ok: 0
  compress_time: 0
 uncompress_ops: 0
uncompress_time: 0
*************************** 2. row ***************************
      page_size: 2048
   compress_ops: 0
compress_ops_ok: 0
  compress_time: 0
 uncompress_ops: 0
uncompress_time: 0
*************************** 3. row ***************************
      page_size: 4096
   compress_ops: 0
compress_ops_ok: 0
  compress_time: 0
 uncompress_ops: 0
uncompress_time: 0
*************************** 4. row ***************************
      page_size: 8192
   compress_ops: 86955
compress_ops_ok: 81182
  compress_time: 27
 uncompress_ops: 26828
uncompress_time: 5
*************************** 5. row ***************************
      page_size: 16384
   compress_ops: 0
compress_ops_ok: 0
  compress_time: 0
 uncompress_ops: 0
uncompress_time: 0
5 rows in set (0.00 sec)
```

注:

- これらのテーブルを使用して、データベース内の InnoDB テーブルの圧縮の有効性を測定します。
- DESCRIBE または SHOW COLUMNS を使用すると、データ型とデフォルト値を含む、これらのテーブルのカラムに関する追加情報を表示できます。
- このテーブルをクエリーするには PROCESS 権限が必要です。
- 使用法については、セクション14.7.4「実行時の圧縮のモニタリング」およびセクション14.14.1.3「圧縮情報スキーマテーブルの使用」を参照してください。InnoDB テーブルの圧縮に関する一般情報については、セクション14.7「InnoDB 圧縮テーブル」を参照してください。

21.29.2 INFORMATION_SCHEMA INNODB_CMP_PER_INDEX および INNODB_CMP_PER_INDEX_RESET テーブル

INNODB_CMP_PER_INDEX および INNODB_CMP_PER_INDEX_RESET テーブルには、圧縮 InnoDB テーブルおよびインデックスに関連した操作に関するステータス情報と、データベース、テーブル、およびインデックスの組み合わせごとの統計が含まれており、特定のテーブルの圧縮のパフォーマンスと有用性を評価する場合に役立ちます。

圧縮 InnoDB テーブルについては、テーブルデータとすべてのセカンダリインデックスの両方が圧縮されます。このコンテキストでは、テーブルデータは、単なる別のインデックス、たまたますべてのカラムが含まれているインデックス (クラスタ化されたインデックス) として扱われます。

表 21.2 INNODB_CMP_PER_INDEX および INNODB_CMP_PER_INDEX_RESET のカラム

カラム名	説明
DATABASE_NAME	適用可能なテーブルを含むデータベース。
TABLE_NAME	圧縮統計についてモニターするテーブル。
INDEX_NAME	圧縮統計についてモニターするインデックス。
COMPRESS_OPS	圧縮操作の試行回数。空のページが作成されたり、非圧縮変更ログ用の領域が不足したりするたびに、ページが圧縮されます。
COMPRESS_OPS_OK	成功した圧縮操作の数。COMPRESS_OPS 値から引き算すると、圧縮の失敗の回数が求められます。COMPRESS_OPS 値で割り算すると、圧縮の失敗の割合が求められます。
COMPRESS_TIME	このインデックスのデータの圧縮に使用された CPU 使用時間の合計 (秒単位)。
UNCOMPRESS_OPS	実行された圧縮解除操作の回数。圧縮 InnoDB ページは、圧縮が失敗した場合はいつでも圧縮解除されます。または、バッファープールにおいてはじめて圧縮ページへのアクセスがあり、圧縮解除されたページが存在しないときに圧縮解除されます。
UNCOMPRESS_TIME	このインデックスのデータの圧縮解除に使用された CPU 使用時間の合計 (秒単位)。

例:

```
mysql> SELECT * FROM INFORMATION_SCHEMA.INNODB_CMP_PER_INDEX \G
*************************** 1. row ***************************
  database_name: employees
     table_name: salaries
     index_name: PRIMARY
   compress_ops: 0
compress_ops_ok: 0
  compress_time: 0
 uncompress_ops: 23451
uncompress_time: 4
*************************** 2. row ***************************
  database_name: employees
     table_name: salaries
     index_name: emp_no
   compress_ops: 0
compress_ops_ok: 0
  compress_time: 0
 uncompress_ops: 1597
uncompress_time: 0
2 rows in set (0.00 sec)
```

注:

- これらのテーブルを使用して、特定のテーブルまたはインデックス、あるいはその両方について InnoDB テーブル圧縮の有効性を測定します。
- DESCRIBE または SHOW COLUMNS を使用すると、データ型とデフォルト値を含む、これらのテーブルのカラムに関する追加情報を表示できます。
- これらのテーブルをクエリーするには PROCESS 権限が必要です。
- すべてのインデックスで個別に測定値を収集すると、大幅なパフォーマンスオーバーヘッドが発生するので、デフォルトでは INNODB_CMP_PER_INDEX および INNODB_CMP_PER_INDEX_RESET 統計は収集されません。モニターする圧縮テーブルで操作を実行する前に、innodb_cmp_per_index_enabled 構成オプションを有効にする必要があります。
- 使用法については、セクション14.7.4「実行時の圧縮のモニタリング」およびセクション14.14.1.3「圧縮情報スキーマテーブルの使用」を参照してください。InnoDB テーブルの圧縮に関する一般情報については、セクション14.7「InnoDB 圧縮テーブル」を参照してください。

21.29.3 INFORMATION_SCHEMA INNODB_CMPMEM および INNODB_CMPMEM_RESET テーブル

INNODB_CMPMEM および INNODB_CMPMEM_RESET テーブルには、InnoDB バッファープール内の圧縮ページに関するステータス情報が含まれます。

表 21.3 INNODB_CMPMEM および INNODB_CMPMEM_RESET のカラム

カラム名	説明
PAGE_SIZE	ブロックサイズ (バイト単位)。このテーブルの各レコードは、このサイズのブロックを記述します。
BUFFER_POOL_INSTANCE	バッファープールインスタンスの一意の識別子。
PAGES_USED	現在使用されているサイズ PAGE_SIZE のブロックの数。
PAGES_FREE	現在割り当てに使用できるサイズ PAGE_SIZE のブロックの数。このカラムは、メモリープールの外部断片化を示します。これらの数値は最大 1 にしてください。
RELOCATION_OPS	サイズ PAGE_SIZE のブロックが再配置された回数。バディーシステムは、より大きな解放されたブロックを生成しようとするときに、解放されたブロックの割り当て済みの「バディー」を再配置できます。テーブル INNODB_CMPMEM_RESET から読み取ると、このカウントはリセットされます。
RELOCATION_TIME	サイズ PAGE_SIZE のブロックの再配置に費やされた合計時間 (マイクロ秒)。テーブル INNODB_CMPMEM_RESET から読み取ると、このカウントはリセットされます。

例:

```
mysql> SELECT * FROM INFORMATION_SCHEMA.INNODB_CMPMEM \G
*************************** 1. row ***************************
         page_size: 1024
buffer_pool_instance: 0
        pages_used: 0
        pages_free: 0
    relocation_ops: 0
   relocation_time: 0
*************************** 2. row ***************************
         page_size: 2048
buffer_pool_instance: 0
        pages_used: 0
        pages_free: 0
    relocation_ops: 0
   relocation_time: 0
*************************** 3. row ***************************
         page_size: 4096
buffer_pool_instance: 0
        pages_used: 0
        pages_free: 0
    relocation_ops: 0
```

```
    relocation_time: 0
*************************** 4. row ***************************
         page_size: 8192
buffer_pool_instance: 0
        pages_used: 7673
        pages_free: 15
     relocation_ops: 4638
    relocation_time: 0
*************************** 5. row ***************************
         page_size: 16384
buffer_pool_instance: 0
        pages_used: 0
        pages_free: 0
     relocation_ops: 0
    relocation_time: 0
5 rows in set (0.00 sec)
```

注:

- これらのテーブルを使用して、データベース内の InnoDB テーブルの圧縮の有効性を測定します。
- DESCRIBE または SHOW COLUMNS を使用すると、データ型とデフォルト値を含む、これらのテーブルのカラムに関する追加情報を表示できます。
- このテーブルをクエリーするには PROCESS 権限が必要です。
- 使用法については、セクション14.7.4「実行時の圧縮のモニタリング」およびセクション14.14.1.3「圧縮情報スキーマテーブルの使用」を参照してください。InnoDB テーブルの圧縮に関する一般情報については、セクション14.7「InnoDB 圧縮テーブル」を参照してください。

21.29.4 INFORMATION_SCHEMA INNODB_TRX テーブル

INNODB_TRX テーブルには、トランザクションがロックを待機しているかどうか、トランザクションが開始した時点、トランザクションが実行している SQL ステートメント (存在する場合) など、InnoDB 内部で現在実行している (読み取り専用トランザクションを除く) すべてのトランザクションに関する情報が含まれます。

表 21.4 INNODB_TRX のカラム

カラム名	説明
TRX_ID	InnoDB 内部の一意のトランザクション ID 番号。(MySQL 5.6 以降、これらの ID は、読み取り専用で非ロックのトランザクションでは作成されません。詳細は、セクション14.13.14「InnoDB の読み取り専用トランザクションの最適化」を参照してください。)
TRX_WEIGHT	トランザクションの重み。これは、トランザクションが変更した行の数とロックした行の数を反映したものです (ただし必ずしも正確な数ではありません)。デッドロックを解決するために、InnoDB は、ロールバックする「対象」として重みがもっとも小さなトランザクションを選択します。非トランザクションテーブルを変更したトランザクションは、変更された行およびロックされた行の数に関係なく、ほかのトランザクションより重みが高いと見なされます。
TRX_STATE	トランザクション実行の状態。RUNNING、LOCK WAIT、ROLLING BACK、または COMMITTING のいずれかです。
TRX_STARTED	トランザクションの開始時間。
TRX_REQUESTED_LOCK_ID	トランザクションが現在待機しているロック ID (TRX_STATE が LOCK WAIT である場合。それ以外は NULL)。ロックに関する詳細は、LOCK_ID で INNODB_LOCKS と結合することによって参照できます。
TRX_WAIT_STARTED	トランザクションがロックを待機し始めた時間 (TRX_STATE が LOCK WAITの場合。それ以外は NULL)。
TRX_MYSQL_THREAD_ID	MySQL スレッド ID。ID で PROCESSLIST と結合する場合に使用できます。PROCESSLIST データとの不整合の可能性を参照してください。
TRX_QUERY	トランザクションにより実行されている SQL クエリー。
TRX_OPERATION_STATE	トランザクションの現在の操作、または NULL。
TRX_TABLES_IN_USE	このトランザクションの現在の SQL ステートメントを処理しているときに使用される InnoDB テーブルの数。

カラム名	説明
TRX_TABLES_LOCKED	現在の SQL ステートメントが行ロックを持っている、InnoDB テーブルの数。(これらはテーブルロックではなく行ロックなので、一部の行がロックされているかどうかにかかわらず、通常、複数のトランザクションによるテーブルからの読み取りおよびテーブルへの書き込みを実行できます。)
TRX_LOCK_STRUCTS	トランザクションで予約されたロックの数。
TRX_LOCK_MEMORY_BYTES	このトランザクションのロック構造によってメモリー内で使用された合計サイズ。
TRX_ROWS_LOCKED	このトランザクションによってロックされた行の概数。この値には、物理的には存在するがトランザクションから認識できない削除マークが付けられた行が含まれる場合があります。
TRX_ROWS_MODIFIED	このトランザクションで変更および挿入された行の数。
TRX_CONCURRENCY_TICKETS	innodb_concurrency_tickets オプションによる指定に従い、スワップアウトされる前に現在のトランザクションで行える作業量を示す値。
TRX_ISOLATION_LEVEL	現在のトランザクションの分離レベル。
TRX_UNIQUE_CHECKS	現在のトランザクションで一意チェックがオンになっているか、オフになっているか。(たとえば一括データロード中にオフになっている場合があります。)
TRX_FOREIGN_KEY_CHECKS	現在のトランザクションで外部キーチェックがオンになっているか、オフになっているか。(たとえば一括データロード中にオフになっている場合があります。)
TRX_LAST_FOREIGN_KEY_ERROR	最後の FK エラーの詳細なエラーメッセージか、NULL。
TRX_ADAPTIVE_HASH_LATCHED	アダプティブハッシュインデックスが現在のトランザクションによってロックされているかどうか。(一度に 1 つのトランザクションだけがアダプティブハッシュインデックスを変更できます。)
TRX_ADAPTIVE_HASH_TIMEOUT	アダプティブハッシュインデックスの検索ラッチをすぐに破棄するか、MySQL からの呼び出し全体で保持するか。AHI 競合がない場合、この値はゼロのままで、ステートメントは終了するまでラッチを保持します。競合の間、ゼロまでカウントダウンし、ステートメントは各行ルックアップの直後にラッチを解放します。
TRX_IS_READ_ONLY	1 の値は、トランザクションが読み取り専用であることを示します (5.6.4 以降。)
TRX_AUTOCOMMIT_NON_LOCKING	1 の値は、トランザクションが、FOR UPDATE 句または LOCK IN SHARED MODE 句を使用していない SELECT ステートメントであり、この 1 つのステートメントだけがトランザクションに含まれるように autocommit 設定をオンにして実行していることを示します (5.6.4 以降)。このカラムと TRX_IS_READ_ONLY がどちらも 1 である場合、InnoDB は、テーブルデータを変更するトランザクションと関連付けられたオーバーヘッドを軽減するように、トランザクションを最適化します。

例:

```
mysql> SELECT * FROM INFORMATION_SCHEMA.INNODB_TRX \G
*************************** 1. row ***************************
                    trx_id: 3298
                 trx_state: RUNNING
               trx_started: 2014-11-19 13:54:39
     trx_requested_lock_id: NULL
          trx_wait_started: NULL
                trx_weight: 316436
       trx_mysql_thread_id: 2
                 trx_query: DELETE FROM employees.salaries WHERE salary > 65000
       trx_operation_state: updating or deleting
         trx_tables_in_use: 1
         trx_tables_locked: 1
          trx_lock_structs: 1621
     trx_lock_memory_bytes: 243240
           trx_rows_locked: 759343
         trx_rows_modified: 314815
   trx_concurrency_tickets: 0
       trx_isolation_level: REPEATABLE READ
         trx_unique_checks: 1
    trx_foreign_key_checks: 1
```

```
trx_last_foreign_key_error: NULL
 trx_adaptive_hash_latched: 0
 trx_adaptive_hash_timeout: 10000
          trx_is_read_only: 0
trx_autocommit_non_locking: 0
```

注:

- このテーブルを使用すると、負荷の大きな同時ロードの時間中に生じるパフォーマンスの問題の診断に役立ちます。その内容は、セクション14.14.2.3「InnoDB トランザクションおよびロックテーブルのデータ永続性および一貫性」で説明しているように更新されます。
- DESCRIBE または SHOW COLUMNS を使用して、データ型とデフォルト値を含む、このテーブルのカラムに関する追加情報を表示します。
- このテーブルをクエリーするには PROCESS 権限が必要です。
- 使用法については、セクション14.14.2.1「InnoDB トランザクションおよびロックテーブルの使用例」を参照してください。

21.29.5 INFORMATION_SCHEMA INNODB_LOCKS テーブル

INNODB_LOCKS テーブルには、InnoDB トランザクションが要求したがまだ取得されていない各ロックに関する情報と、別のトランザクションをブロックしているトランザクションが保持している各ロックに関する情報が含まれます。

表 21.5 INNODB_LOCKS のカラム

カラム名	説明
LOCK_ID	InnoDB 内部の一意のロック ID 番号。これは不明瞭な文字列として扱ってください。LOCK_ID には現在、TRX_ID が含まれますが、LOCK_ID 内のデータの形式は今後のリリースでも同じである保証はありません。LOCK_ID 値を解析するプログラムを作成しないでください。
LOCK_TRX_ID	このロックを保持するトランザクションの ID。トランザクションに関する詳細は、TRX_ID で INNODB_TRX と結合すると参照できます。
LOCK_MODE	ロックのモード。S、X、IS、IX、S_GAP、X_GAP、IS_GAP、IX_GAP、または AUTO_INC のいずれかになり、それぞれ共有、排他的、インテンション共有、インテンション排他的行ロック、共有および排他的ギャップロック、インテンション共有およびインテンション排他的ギャップロック、および自動インクリメントテーブルレベルロックを表します。InnoDB ロックに関する情報については、セクション14.2.3「InnoDB のロックモード」およびセクション14.2.2「InnoDB のトランザクションモデルおよびロック」のセクションを参照してください。
LOCK_TYPE	ロックのタイプ。RECORD または TABLE のいずれかで、それぞれレコード (行) レベル、テーブルレベルのロックを表します。
LOCK_TABLE	ロックされているテーブルか、ロックされたレコードを含むテーブルの名前。
LOCK_INDEX	LOCK_TYPE='RECORD' の場合はインデックスの名前、それ以外の場合は NULL。
LOCK_SPACE	LOCK_TYPE='RECORD' の場合はロックされたレコードのテーブルスペース ID、それ以外の場合は NULL。
LOCK_PAGE	LOCK_TYPE='RECORD' の場合はロックされたレコードのページ番号、それ以外の場合は NULL。
LOCK_REC	LOCK_TYPE='RECORD' の場合はページ内のロックされたレコードのヒープ番号、それ以外の場合は NULL。
LOCK_DATA	LOCK_TYPE='RECORD' の場合はロックされたレコードの主キー値、それ以外の場合は NULL。このカラムには、ロックされた行の主キーカラムの値が含まれ、有効な SQL 文字列として書式設定されています (SQL コマンドにコピーできるようになっています)。主キーがない場合、InnoDB 内部の一意の行 ID 番号が使用されます。インデックスの最大値を超えるキー値または範囲に対してギャップロックが行われた場合、LOCK_DATA は「supremum pseudo-record」とレポートします。ロックされたレコードを含むページがバッファープール内に存在しない場合 (ロックが保持されていた間にディスクにページアウトされた場合)、InnoDB は、不要なディスク操作を回避するするために、ディスクからページをフェッチしません。代わりに、LOCK_DATA が NULL に設定されます。

例:

```
mysql> SELECT * FROM INFORMATION_SCHEMA.INNODB_LOCKS \G
*************************** 1. row ***************************
    lock_id: 3723:72:3:2
lock_trx_id: 3723
  lock_mode: X
  lock_type: RECORD
 lock_table: `mysql`.`t`
 lock_index: PRIMARY
 lock_space: 72
  lock_page: 3
   lock_rec: 2
  lock_data: 1, 9
*************************** 2. row ***************************
    lock_id: 3722:72:3:2
lock_trx_id: 3722
  lock_mode: S
  lock_type: RECORD
 lock_table: `mysql`.`t`
 lock_index: PRIMARY
 lock_space: 72
  lock_page: 3
   lock_rec: 2
  lock_data: 1, 9
2 rows in set (0.01 sec)
```

注:

- このテーブルを使用すると、負荷の大きな同時ロードの時間中に生じるパフォーマンスの問題の診断に役立ちます。その内容は、セクション14.14.2.3「InnoDB トランザクションおよびロックテーブルのデータ永続性および一貫性」で説明しているように更新されます。
- DESCRIBE または SHOW COLUMNS を使用して、データ型とデフォルト値を含む、このテーブルのカラムに関する追加情報を表示します。
- このテーブルをクエリーするには PROCESS 権限が必要です。
- 使用法については、セクション14.14.2.1「InnoDB トランザクションおよびロックテーブルの使用例」を参照してください。

21.29.6 INFORMATION_SCHEMA INNODB_LOCK_WAITS テーブル

INNODB_LOCK_WAITS テーブルには、ブロックされている InnoDB トランザクションごとに 1 つ以上の行が含まれ、トランザクションで要求したロックと、その要求をブロックしているすべてのロックを示します。

表 21.6 INNODB_LOCK_WAITS のカラム

カラム名	説明
REQUESTING_TRX_ID	要求しているトランザクションの ID。
REQUESTED_LOCK_ID	トランザクションが待機しているロックの ID。ロックに関する詳細は、LOCK_ID で INNODB_LOCKS と結合することによって参照できます。
BLOCKING_TRX_ID	ブロックしているトランザクションの ID。
BLOCKING_LOCK_ID	別のトランザクションが処理できないようにブロックしているトランザクションが保持しているロックの ID。ロックに関する詳細は、LOCK_ID で INNODB_LOCKS と結合することによって参照できます。

例:

```
mysql> SELECT * FROM INFORMATION_SCHEMA.INNODB_LOCK_WAITS \G
*************************** 1. row ***************************
requesting_trx_id: 3396
requested_lock_id: 3396:91:3:2
  blocking_trx_id: 3395
 blocking_lock_id: 3395:91:3:2
1 row in set (0.00 sec)
```

注:

- このテーブルを使用すると、負荷の大きな同時ロードの時間中に生じるパフォーマンスの問題の診断に役立ちます。その内容は、セクション14.14.2.3「InnoDB トランザクションおよびロックテーブルのデータ永続性および一貫性」で説明しているように更新されます。

- DESCRIBE または SHOW COLUMNS を使用して、データ型とデフォルト値を含む、このテーブルのカラムに関する追加情報を表示します。
- このテーブルをクエリーするには PROCESS 権限が必要です。
- 使用法については、セクション14.14.2.1「InnoDB トランザクションおよびロックテーブルの使用例」を参照してください。

21.29.7 INFORMATION_SCHEMA INNODB_SYS_TABLES テーブル

INNODB_SYS_TABLES テーブルは、InnoDB データディクショナリ内の SYS_TABLES テーブルの情報と同等の、InnoDB テーブルに関するメタデータを提供します。

関連する使用法と使用例については、セクション14.14.3「InnoDB INFORMATION_SCHEMA システムテーブル」を参照してください。

表 21.7 INNODB_SYS_TABLES のカラム

カラム名	説明
TABLE_ID	インスタンスのすべてのデータベースで一意の各 InnoDB テーブルの識別子。
NAME	テーブルの名前。test/t1 のように、必要に応じてデータベース名が前に置かれます。InnoDB システムテーブル名はすべて大文字です。データベースおよびユーザーテーブルの名前の大文字/小文字は、もともと定義されていたものと同じで、lower_case_table_names 設定による影響を受ける可能性があります。
FLAG	この値は、行フォーマット、圧縮ページサイズ (該当する場合)、DATA DIRECTORY 句の CREATE TABLE または ALTER TABLE での使用の有無などのテーブルフォーマットおよびストレージ特性に関するビットレベル情報を提供します。
N_COLS	テーブル内のカラムの数。レポートされる数値には、InnoDB によって作成される 3 つの非表示カラム (DB_ROW_ID、DB_TRX_ID、DB_ROLL_PTR) が含まれます。
SPACE	テーブルが存在するテーブルスペースの識別子。0 は InnoDB システムテーブルスペースを示します。ほかの数値はすべて、個別の .ibd ファイルを使用して file-per-table モードで作成されたテーブルを表します。この識別子は、TRUNCATE TABLE ステートメントのあとでも同じままです。ゼロ以外の値では、この識別子は、このインスタンス内のすべてのデータベース間のテーブルで一意です。
FILE_FORMAT	テーブルのファイル形式 (Antelope または Barracuda)。
ROW_FORMAT	テーブルの行フォーマット (Compact、Redundant、Dynamic、または Compressed)。
ZIP_PAGE_SIZE	Zip ページサイズ。Compressed 行フォーマットを使用するテーブルにのみ適用されます。

例:

```
mysql> SELECT * FROM INFORMATION_SCHEMA.INNODB_SYS_TABLES WHERE TABLE_ID = 74 \G
*************************** 1. row ***************************
    TABLE_ID: 74
        NAME: test/t1
        FLAG: 1
      N_COLS: 6
       SPACE: 60
 FILE_FORMAT: Antelope
  ROW_FORMAT: Compact
ZIP_PAGE_SIZE: 0
1 row in set (0.00 sec)
```

注:

- DESCRIBE または SHOW COLUMNS を使用して、データ型とデフォルト値を含む、このテーブルのカラムに関する追加情報を表示します。
- このテーブルをクエリーするには PROCESS 権限が必要です。

INNODB_SYS_TABLES.FLAG カラム値の解釈

INNODB_SYS_TABLES.FLAG カラムは、テーブルのフォーマットおよびストレージ特性に関するビットレベル情報を提供します。次の表に示された該当する 10 進数値を一緒に付加すると、FLAG カラム値を解釈できます。

表 21.8 INNODB_SYS_TABLES FLAG カラムデータを解釈するためのビット位置値

ビット位置	説明	10 進数値
0	行フォーマットが REDUNDANT でない場合、このビットが設定されます。つまり、行フォーマットが COMPACT、DYNAMIC、または COMPRESSED の場合に設定されます。	• 0 - REDUNDANT • 1 - COMPACT、DYNAMIC、または COMPRESSED
1-4	これらの 4 つのビットには、テーブルの圧縮されたページサイズを表す小さな数値が含まれます。INNODB_SYS_TABLES.ZIP_PAGE_SIZE フィールドにも、該当する場合、圧縮されたページサイズがレポートされます。	• 0 - 圧縮なし • 2 - 1024 バイトの圧縮ページサイズ • 4 - 2048 バイトの圧縮ページサイズ • 6 - 4096 バイトの圧縮ページサイズ • 8 - 8192 バイトの圧縮ページサイズ • 10 - 16384 バイトの圧縮ページサイズ
5	行フォーマットが DYNAMIC または COMPRESSED の場合、このビットが設定されます。	• 0 - REDUNDANT または COMPACT • 32 - DYNAMIC または COMPRESSED
6	CREATE TABLE または ALTER TABLE で DATA DIRECTORY オプションが使用された場合、このビットが設定されます。デフォルトのデータディレクトリ (datadir) 以外のディレクトリに置かれた file-per-table テーブルスペースに、このビットが設定されます。これらのテーブルについては、tablename.isl ファイルは tablename.frm ファイルと同じ場所に存在します。tablename.isl ファイルには、tablename.ibd file-per-table テーブルスペースファイルへの実際のディレクトリパスが格納されます。	• 0 - リモートの file-per-table テーブルスペースではない場合 • 64 - リモートの file-per-table テーブルスペース

次のテーブル t1 では ROW_FORMAT=DYNAMIC を使用し、FLAG 値は 33 です。前述の表の情報に従うと、DYNAMIC 行フォーマットのテーブルでビット位置 0 が 1 に、ビット位置 5 が 32 に設定されることがわかります。これらの値は合計 33 の FLAG 値になります。

```
mysql> use test;
Database changed

mysql> SET GLOBAL innodb_file_format=Barracuda;
Query OK, 0 rows affected (0.00 sec)

mysql> CREATE TABLE t1 (c1 int) ROW_FORMAT=DYNAMIC;
Query OK, 0 rows affected (0.02 sec)

mysql> SELECT * FROM INFORMATION_SCHEMA.INNODB_SYS_TABLES WHERE NAME LIKE 'test/t1' \G
*************************** 1. row ***************************
     TABLE_ID: 89
         NAME: test/t1
         FLAG: 33
       N_COLS: 4
        SPACE: 75
  FILE_FORMAT: Barracuda
   ROW_FORMAT: Dynamic
ZIP_PAGE_SIZE: 0
1 row in set (0.01 sec)
```

21.29.8 INFORMATION_SCHEMA INNODB_SYS_INDEXES テーブル

INNODB_SYS_INDEXES テーブルは、InnoDB データディクショナリの内部 SYS_INDEXES テーブル内の情報と同等の、InnoDB インデックスに関するメタデータを提供します。

関連する使用法と使用例については、セクション14.14.3「InnoDB INFORMATION_SCHEMA システムテーブル」を参照してください。

表 21.9 INNODB_SYS_INDEXES のカラム

カラム名	説明
INDEX_ID	インスタンス内のすべてのデータベースで一意である各インデックスの識別子。
NAME	インデックスの名前。InnoDB によって暗黙的に作成されたほとんどのインデックスには一貫した名前が付けられていますが、インデックス名は必ずしも一意ではありません。たとえば、主キーインデックスには PRIMARY、インデックスが指定されていない場合に主キーを表すインデックスには GEN_CLUST_INDEX、および外部キー制約には ID_IND、FOR_IND、および REF_IND があります。
TABLE_ID	インデックスに関連付けられているテーブルを表す識別子。INNODB_SYS_TABLES.TABLE_ID の値と同じです。
TYPE	インデックスの種類を示す数値識別子。0 = セカンダリインデックス、1 = クラスタ化されたインデックス、2 = 一意インデックス、3 = プライマリインデックス、32 = 全文インデックスです。
N_FIELDS	インデックスキーのカラムの数。GEN_CLUST_INDEX インデックスの場合、実際のテーブルカラムではなく人為的な値を使用してインデックスが作成されているので、この値は 0 です。
PAGE_NO	インデックス B ツリーのルートページ番号。全文インデックスの場合、複数の B ツリー (補助テーブル) でレイアウトされているので、PAGE_NO フィールドは未使用で、-1 (FIL_NULL) に設定されます。
SPACE	インデックスが存在するテーブルスペースの識別子。0 は InnoDB システムテーブルスペースを示します。ほかの数値はすべて、個別の .ibd ファイルを使用して file-per-table モードで作成されたテーブルを表します。この識別子は、TRUNCATE TABLE ステートメントのあとでも同じままです。テーブルのすべてのインデックスが、テーブルと同じテーブルスペースに存在するので、この値は必ずしも一意にはなりません。

例:

```
mysql> SELECT * FROM INFORMATION_SCHEMA.INNODB_SYS_INDEXES WHERE TABLE_ID = 74 \G
*************************** 1. row ***************************
INDEX_ID: 116
    NAME: GEN_CLUST_INDEX
TABLE_ID: 74
    TYPE: 1
N_FIELDS: 0
 PAGE_NO: 3
   SPACE: 60
*************************** 2. row ***************************
INDEX_ID: 117
    NAME: i1
TABLE_ID: 74
    TYPE: 0
N_FIELDS: 1
 PAGE_NO: 4
   SPACE: 60
2 rows in set (0.00 sec)
```

注:

- DESCRIBE または SHOW COLUMNS を使用して、データ型とデフォルト値を含む、このテーブルのカラムに関する追加情報を表示します。
- このテーブルをクエリーするには PROCESS 権限が必要です。

21.29.9 INFORMATION_SCHEMA INNODB_SYS_COLUMNS テーブル

INNODB_SYS_COLUMNS は、InnoDB データディクショナリの SYS_COLUMNS テーブルの情報と同等の、InnoDB テーブルカラムに関するメタデータを提供します。

関連する使用法と使用例については、セクション14.14.3「InnoDB INFORMATION_SCHEMA システムテーブル」を参照してください。

表 21.10 INNODB_SYS_COLUMNS のカラム

カラム名	説明
TABLE_ID	カラムに関連付けられているテーブルを表す識別子。INNODB_SYS_TABLES.TABLE_ID の値と同じです。
NAME	各テーブルのそれぞれのカラムの名前。これらの名前の大文字/小文字は、lower_case_table_names 設定に応じて異なります。カラムの特別なシステム予約名はありません。
POS	0 から始まり連続的に増加する、テーブル内のカラムの順序位置。あるカラムを削除すると、順序に欠落ができないように残りのカラムの順序が変更されます。
MTYPE	「メインの型」を表します。カラム型の数値識別子。1 = VARCHAR、2 = CHAR、3 = FIXBINARY、4 = BINARY、5 = BLOB、6 = INT、7 = SYS_CHILD、8 = SYS、9 = FLOAT、10 = DOUBLE、11 = DECIMAL、12 = VARMYSQL、13 = MYSQL です。
PRTYPE	InnoDB の「正確な型」。MySQL データ型、文字セット、および NULL 可能性を表すビットを含むバイナリ値です。
LEN	カラム長。たとえば INT には 4、BIGINT には 8 です。マルチバイト文字セットの文字カラムの場合、この長さ値は、VARCHAR(N) などの定義を表すために必要なバイト単位の最大長です。つまり、文字エンコーディングに応じて、2*N、3*N などになります。

例:

```
mysql> SELECT * FROM INFORMATION_SCHEMA.INNODB_SYS_COLUMNS where TABLE_ID = 71 \G
*************************** 1. row ***************************
TABLE_ID: 71
    NAME: col1
     POS: 0
   MTYPE: 6
  PRTYPE: 1027
     LEN: 4
*************************** 2. row ***************************
TABLE_ID: 71
    NAME: col2
     POS: 1
   MTYPE: 2
  PRTYPE: 524542
     LEN: 10
*************************** 3. row ***************************
TABLE_ID: 71
    NAME: col3
     POS: 2
   MTYPE: 1
  PRTYPE: 524303
     LEN: 10
3 rows in set (0.00 sec)
```

注:

- DESCRIBE または SHOW COLUMNS を使用して、データ型とデフォルト値を含む、このテーブルのカラムに関する追加情報を表示します。
- このテーブルをクエリーするには PROCESS 権限が必要です。

21.29.10 INFORMATION_SCHEMA INNODB_SYS_FIELDS テーブル

INNODB_SYS_FIELDS テーブルは、InnoDB データディクショナリの SYS_FIELDS テーブルの情報と同等の、InnoDB インデックスのキーカラム (フィールド) に関するメタデータを提供します。

関連する使用法と使用例については、セクション14.14.3「InnoDB INFORMATION_SCHEMA システムテーブル」を参照してください。

表 21.11 INNODB_SYS_FIELDS のカラム

カラム名	説明
INDEX_ID	このキーフィールドに関連付けられたインデックスの識別子。INNODB_SYS_INDEXES.INDEX_ID と同じ値を使用します。

カラム名	説明
NAME	テーブルの元のカラム名。INNODB_SYS_COLUMNS.NAME と同じ値を使用します。
POS	0 から始まり連続的に増加する、インデックス内のキーフィールドの順序位置。あるカラムを削除すると、順序に欠落ができないように残りのカラムの順序が変更されます。

例:

```
mysql> SELECT * FROM INFORMATION_SCHEMA.INNODB_SYS_FIELDS where INDEX_ID = 117 \G
*************************** 1. row ***************************
INDEX_ID: 117
    NAME: col1
     POS: 0
1 row in set (0.00 sec)
```

注:

- DESCRIBE または SHOW COLUMNS を使用して、データ型とデフォルト値を含む、このテーブルのカラムに関する追加情報を表示します。
- このテーブルをクエリーするには PROCESS 権限が必要です。

21.29.11 INFORMATION_SCHEMA INNODB_SYS_FOREIGN テーブル

INNODB_SYS_FOREIGN テーブルは、InnoDB データディクショナリの SYS_FOREIGN テーブルの情報と同等の、InnoDB 外部キーに関するメタデータを提供します。

関連する使用法と使用例については、セクション14.14.3「InnoDB INFORMATION_SCHEMA システムテーブル」を参照してください。

表 21.12 INNODB_SYS_FOREIGN のカラム

カラム名	説明
ID	外部キーインデックスの名前 (数値以外)。test/products_fk のように、データベース名が前に付けられます。
FOR_NAME	この外部キー関係の子テーブルの名前。
REF_NAME	この外部キー関係の親テーブルの名前。
N_COLS	外部キーインデックスのカラム数。
TYPE	外部キーカラム ORed に関する情報を伴うビットフラグの集まり。1 = ON DELETE CASCADE、2 = ON UPDATE SET NULL、4 = ON UPDATE CASCADE、8 = ON UPDATE SET NULL、16 = ON DELETE NO ACTION、32 = ON UPDATE NO ACTION です。

例:

```
mysql> SELECT * FROM INFORMATION_SCHEMA.INNODB_SYS_FOREIGN \G
*************************** 1. row ***************************
      ID: test/fk1
FOR_NAME: test/child
REF_NAME: test/parent
  N_COLS: 1
    TYPE: 1
1 row in set (0.00 sec)
```

注:

- DESCRIBE または SHOW COLUMNS を使用して、データ型とデフォルト値を含む、このテーブルのカラムに関する追加情報を表示します。
- このテーブルをクエリーするには PROCESS 権限が必要です。

21.29.12 INFORMATION_SCHEMA INNODB_SYS_FOREIGN_COLS テーブル

INNODB_SYS_FOREIGN_COLS テーブルは、InnoDB データディクショナリ内の SYS_FOREIGN_COLS テーブルの情報と同等の、InnoDB 外部キーのカラムに関するステータス情報を提供します。

関連する使用法と使用例については、セクション14.14.3「InnoDB INFORMATION_SCHEMA システムテーブル」を参照してください。

表 21.13 INNODB_SYS_FOREIGN_COLS のカラム

カラム名	説明
ID	このインデックスキーフィールドに関連付けられた外部キーインデックス。INNODB_SYS_FOREIGN.ID と同じ値を使用します。
FOR_COL_NAME	子テーブルに関連付けられたカラムの名前。
REF_COL_NAME	親テーブルに関連付けられたカラムの名前。
POS	0 から始まる、外部キーインデックス内のこのキーフィールドの順序位置。

例:

```
mysql> SELECT * FROM INFORMATION_SCHEMA.INNODB_SYS_FOREIGN_COLS WHERE ID = 'test/fk1' \G
*************************** 1. row ***************************
        ID: test/fk1
FOR_COL_NAME: parent_id
REF_COL_NAME: id
       POS: 0
1 row in set (0.00 sec)
```

注:

- DESCRIBE または SHOW COLUMNS を使用して、データ型とデフォルト値を含む、このテーブルのカラムに関する追加情報を表示します。
- このテーブルをクエリーするには PROCESS 権限が必要です。

21.29.13 INFORMATION_SCHEMA INNODB_SYS_TABLESTATS ビュー

INNODB_SYS_TABLESTATS は、InnoDB テーブルに関する低レベルのステータス情報のビューを提供します。このデータは、InnoDB テーブルのクエリー時に使用するインデックスを計算するために MySQL オプティマイザによって使用されます。この情報は、ディスクに格納されるデータに従うのではなく、インメモリーデータ構造から派生されます。対応する内部 InnoDB システムテーブルはありません。

InnoDB テーブルは、最後にサーバーを再起動してから開かれており、テーブルキャッシュから削除されていない場合、このビューに表示されます。永続的統計を利用できるテーブルは、このビューに常に表示されます。

関連する使用法と使用例については、セクション14.14.3「InnoDB INFORMATION_SCHEMA システムテーブル」を参照してください。

表 21.14 INNODB_SYS_TABLESTATS のカラム

カラム名	説明
TABLE_ID	統計を利用できるテーブルを表す識別子。INNODB_SYS_TABLES.TABLE_ID と同じ値を使用します。
NAME	テーブルの名前。INNODB_SYS_TABLES.NAME と同じ値を使用します。
STATS_INITIALIZED	値は、統計がすでに収集されている場合は Initialized で、収集されていない場合は Uninitialized です。
NUM_ROWS	現在の推定されるテーブル内の行数。それぞれの DML 操作後に更新されます。コミットされていないトランザクションがテーブルに挿入されたり、テーブルから削除されたりする場合、正確ではなくなることがあります。
CLUST_INDEX_SIZE	クラスタ化されたインデックスを格納するディスク上のページ数。主キー順に InnoDB テーブルデータを保持します。この値は、テーブルの統計がまだ収集されていない場合は NULL になることがあります。
OTHER_INDEX_SIZE	テーブルに関するすべてのセカンダリインデックスを格納する、ディスク上のページの数。この値は、テーブルの統計がまだ収集されていない場合は NULL になることがあります。
MODIFIED_COUNTER	INSERT、UPDATE、DELETE などの DML 操作と外部キーのカスケード操作によって変更された行の数。このカラムは、テーブル統計が再計算されるごとにリセットされます。

カラム名	説明
AUTOINC	すべての自動インクリメントベースの操作で発行される次の番号。AUTOINC 値の変更ベースは、自動インクリメント番号が要求された回数と、要求ごとに認められる番号の数に応じて異なります。
REF_COUNT	このカウンタがゼロになると、テーブルメタデータをテーブルキャッシュから削除できます。

例:

```
mysql> SELECT * FROM INFORMATION_SCHEMA.INNODB_SYS_TABLESTATS where TABLE_ID = 71 \G
*************************** 1. row ***************************
        TABLE_ID: 71
            NAME: test/t1
STATS_INITIALIZED: Initialized
        NUM_ROWS: 1
 CLUST_INDEX_SIZE: 1
 OTHER_INDEX_SIZE: 0
 MODIFIED_COUNTER: 1
          AUTOINC: 0
       REF_COUNT: 1
1 row in set (0.00 sec)
```

注:

- このテーブルは、主に、専門家レベルのパフォーマンスモニタリングや、MySQL のパフォーマンス関連の拡張を開発するときに役立ちます。
- DESCRIBE または SHOW COLUMNS を使用して、データ型とデフォルト値を含む、このテーブルのカラムに関する追加情報を表示します。
- このテーブルをクエリーするには PROCESS 権限が必要です。

21.29.14 INFORMATION_SCHEMA INNODB_SYS_DATAFILES テーブル

INNODB_SYS_DATAFILES テーブルは、InnoDB データディクショナリの SYS_DATAFILES テーブル内の情報と同等の、InnoDB テーブルスペースのデータファイルパス情報を提供します。

関連する使用法と使用例については、セクション14.14.3「InnoDB INFORMATION_SCHEMA システムテーブル」を参照してください。

表 21.15 INNODB_SYS_DATAFILES のカラム

カラム名	説明
SPACE	テーブルスペースのスペース ID。
PATH	テーブルスペースデータファイルパス (たとえば 「.\world\innodb\city.ibd」)。CREATE TABLE ステートメントの DATA DIRECTORY 句を使用して、file-per-table テーブルスペースを MySQL データディレクトリ外の場所に作成した場合、テーブルスペース PATH フィールドには完全修飾ディレクトリパスが表示されます。

例:

```
mysql> SELECT * FROM INFORMATION_SCHEMA.INNODB_SYS_DATAFILES WHERE SPACE = 57 \G
*************************** 1. row ***************************
SPACE: 57
 PATH: ./test/t1.ibd
1 row in set (0.01 sec)
```

注:

- DESCRIBE または SHOW COLUMNS を使用して、データ型とデフォルト値を含む、このテーブルのカラムに関する追加情報を表示します。
- このテーブルをクエリーするには PROCESS 権限が必要です。

21.29.15 INFORMATION_SCHEMA INNODB_SYS_TABLESPACES テーブル

INNODB_SYS_TABLESPACES テーブルは、InnoDB データディクショナリの SYS_TABLESPACES テーブル内の情報と同等の、InnoDB テーブルスペースに関するメタデータを提供します。

関連する使用法と使用例については、セクション14.14.3「InnoDB INFORMATION_SCHEMA システムテーブル」を参照してください。

表 21.16 INNODB_SYS_TABLESPACES のカラム

カラム名	説明
SPACE	テーブルスペースのスペース ID。
NAME	データベースおよびテーブル名 (たとえば world_innodb\city)
FLAG	この値は、テーブルスペースの形式とストレージ特性に関するビットレベルの情報を提供します。
FILE_FORMAT	テーブルスペースファイル形式 (たとえば Antelope や Barracuda)。このフィールドのデータは、.ibd ファイル内に存在するテーブルスペースフラグ情報から解釈されます。InnoDB ファイル形式の詳細は、セクション14.8「InnoDB のファイル形式管理」を参照してください。
ROW_FORMAT	テーブルスペースの行フォーマット (Compact または Redundant、Dynamic、Compressed)。このフィールドのデータは、.ibd ファイル内に存在するテーブルスペースフラグ情報から解釈されます。
PAGE_SIZE	テーブルスペースのページサイズ。このフィールドのデータは、.ibd ファイル内に存在するテーブルスペースフラグ情報から解釈されます。
ZIP_PAGE_SIZE	テーブルスペースの Zip ページサイズ。このフィールドのデータは、.ibd ファイル内に存在するテーブルスペースフラグ情報から解釈されます。

例:

```
mysql> SELECT * FROM INFORMATION_SCHEMA.INNODB_SYS_TABLESPACES WHERE SPACE = 57 \G
*************************** 1. row ***************************
       SPACE: 57
        NAME: test/t1
        FLAG: 0
 FILE_FORMAT: Antelope
  ROW_FORMAT: Compact or Redundant
   PAGE_SIZE: 16384
ZIP_PAGE_SIZE: 0
1 row in set (0.00 sec)
```

注:

- DESCRIBE または SHOW COLUMNS を使用して、データ型とデフォルト値を含む、このテーブルのカラムに関する追加情報を表示します。
- このテーブルをクエリーするには PROCESS 権限が必要です。
- すべての Antelope ファイル形式のテーブルスペースフラグは (テーブルフラグとは異なり) 常にゼロなので、テーブルスペースの行フォーマットが Redundant または Compact であるかをこのフラグ整数から判断することはできません。この結果、ROW_FORMAT フィールドに指定可能な値は、「Compact または Redundant」、「Compressed」、「Dynamic」 になります。

INNODB_SYS_TABLESPACES.FLAG カラム値の解釈:

INNODB_SYS_TABLESPACES.FLAG カラムは、テーブルスペースの形式およびストレージ特性に関するビットレベルの情報を提供します。

MySQL 5.6 までは、テーブルフラグとテーブルスペースフラグはビット位置 0 の設定を除き同じでした。MySQL 5.6 では、論理ページサイズを保持するために追加の 4 ビットを必要とする、4K および 8K ページのサポートが追加されました。MySQL 5.6 ではまた、CREATE TABLE および ALTER TABLE DATA DIRECTORY 句のサポートが追加され、MySQL データディレクトリ外の場所に file-per-table テーブルスペースを格納できるようになりました。この機能には、テーブルフラグとテーブルスペースフラグの両方に追加ビットが必要でしたが、位置は同じではありません。

次の表に示された該当する 10 進数値を一緒に付加すると、テーブルスペース FLAG カラム値を解釈できます。

表 21.17 INNODB_SYS_TABLESPACES FLAG カラムデータを解釈するためのビット位置値

ビット位置	説明	10 進数値
0	テーブルスペース内のテーブルの行フォーマットが DYNAMIC または COMPRESSED の場合、このビットが設定されます。この情報を使用して、Antelope ファ	• 0 - REDUNDANT または COMPACT (FILE_FORMAT=Antelope)

ビット位置	説明	10 進数値
	イル形式と Barracuda ファイル形式を区別できますが、REDUNDANT ファイル形式と COMPACT ファイル形式は区別できません (DYNAMIC 行フォーマットと COMPRESSED 行フォーマットには Barracuda ファイル形式が必要です)。file-per-table テーブルスペースである場合、INNODB_SYS_TABLES をクエリーして、2 つの Antelope 行フォーマット (REDUNDANT または COMPACT) のどちらが使用されているかを判断する必要があります。	• 1 - DYNAMIC または COMPRESSED (FILE_FORMAT=Barracuda)
1-4	これらの 4 つのビットには、テーブルスペースの圧縮されたページサイズ (KEY_BLOCK_SIZE または「物理的なブロックサイズ」) を表す小さな数字が含まれます。	• 0 - 圧縮なし • 2 - 1024 バイトの圧縮ページサイズ • 4 - 2048 バイトの圧縮ページサイズ • 6 - 4096 バイトの圧縮ページサイズ • 8 - 8192 バイトの圧縮ページサイズ • 10 - 16384 バイトの圧縮ページサイズ
5	テーブルの行フォーマットが DYNAMIC または COMPRESSED である場合に、file-per-table テーブルスペースにこのビットが設定されます。	• 0 - REDUNDANT または COMPACT • 32 - DYNAMIC または COMPRESSED
6-9	これらの 4 つのビットには、テーブルスペースの圧縮解除されたページサイズ (論理ページサイズ) を表す小さな数字が含まれます。論理ページサイズが 16K の元の InnoDB デフォルトページサイズである場合、この設定はゼロです。	• 192 - 4096 バイトの論理/非圧縮ページサイズ • 256 - 8192 バイトの論理/非圧縮ページサイズ • 0 - 16384 バイトの論理/非圧縮ページサイズ
10	CREATE TABLE または ALTER TABLE で DATA DIRECTORY オプションが使用された場合、このビットが設定されます。デフォルトのデータディレクトリ (datadir) 以外のディレクトリに置かれた file-per-table テーブルスペースに、このビットが設定されます。これらのテーブルについては、tablename.isl ファイルは tablename.frm ファイルと同じ場所に存在します。tablename.isl ファイルには、tablename.ibd file-per-table テーブルスペースファイルへの実際のディレクトリパスが格納されます。	• 0 - リモートの file-per-table テーブルスペースではない場合 • 1024 - リモートの file-per-table テーブルスペース

次の例では、テーブル t1 は innodb_file_per_table=ON で作成されます。これはテーブル t1 を独自のテーブルスペース内に作成します。INNODB_SYS_TABLESPACES をクエリーすると、テーブルスペースの FLAG 値が 33 であることがわかります。どのようにこの値になるかを判断するには、前述の表に記されているビット値を参照してください。テーブル t1 で DYNAMIC 行フォーマットを使用するので、ビット 0 は 1 の値になります。テーブルスペースが DYNAMIC 行フォーマットを使用する file-per-table テーブルスペースなので、ビット 5 は 32 の値になります。innodb_page_size はデフォルトの 16K 値に設定されているので、ビット位置 6-9 は 0 です。ほかのビット値は適用できないため 0 に設定されます。ビット位置 0 とビット位置 5 の値は合計して 33 の FLAG 値になります。

```
mysql> use test;
Database changed

mysql> SHOW VARIABLES LIKE 'innodb_file_per_table';
+-----------------------+-------+
| Variable_name         | Value |
```

```
+-----------------------+-------+
| innodb_file_per_table | ON    |
+-----------------------+-------+
1 row in set (0.00 sec)

mysql> SHOW VARIABLES LIKE 'innodb_page_size';
+------------------+-------+
| Variable_name    | Value |
+------------------+-------+
| innodb_page_size | 16384 |
+------------------+-------+
1 row in set (0.00 sec)

mysql> SET GLOBAL innodb_file_format=Barracuda;
Query OK, 0 rows affected (0.00 sec)

mysql> CREATE TABLE t1 (c1 int) ROW_FORMAT=DYNAMIC;
Query OK, 0 rows affected (0.02 sec)

mysql> SELECT * FROM INFORMATION_SCHEMA.INNODB_SYS_TABLESPACES WHERE NAME LIKE 'test/t1' \G
*************************** 1. row ***************************
       SPACE: 75
        NAME: test/t1
        FLAG: 33
 FILE_FORMAT: Barracuda
  ROW_FORMAT: Dynamic
   PAGE_SIZE: 16384
ZIP_PAGE_SIZE: 0
1 row in set (0.00 sec)
```

21.29.16 INFORMATION_SCHEMA INNODB_BUFFER_PAGE テーブル

INNODB_BUFFER_PAGE テーブルは、各ページに関する情報を、InnoDB バッファープールに保持します。

関連する使用法と使用例については、セクション14.14.5「InnoDB INFORMATION_SCHEMA バッファープールテーブル」を参照してください。

> **警告**
>
> INNODB_BUFFER_PAGE テーブルをクエリーすると、大幅なパフォーマンスのオーバーヘッドが生じる可能性があります。クエリーによって発生する可能性のあるパフォーマンスへの影響を認識し、かつそれが許容可能であると判断されていないかぎり、本番システムではこのテーブルをクエリーしないでください。パフォーマンスへの影響を避けるには、調べる問題をテストインスタンスに再現し、テストインスタンスの INNODB_BUFFER_PAGE テーブルをクエリーします。

表 21.18 INNODB_BUFFER_PAGE のカラム

カラム名	説明
POOL_ID	バッファープール ID。複数のバッファープールインスタンスを区別する識別子。
BLOCK_ID	バッファープールブロック ID。
SPACE	テーブルスペース ID。INNODB_SYS_TABLES.SPACE と同じ値を使用します。
PAGE_NUMBER	ページ番号。
PAGE_TYPE	ページタイプ。ALLOCATED (新しく割り当てられたページ)、INDEX (B ツリーノード)、UNDO_LOG (Undo ログページ)、INODE (インデックスノード)、IBUF_FREE_LIST (挿入バッファー空きリスト)、IBUF_BITMAP (挿入バッファービットマップ)、SYSTEM (システムページ)、TRX_SYSTEM (トランザクションシステムデータ)、FILE_SPACE_HEADER (ファイル領域ヘッダー)、EXTENT_DESCRIPTOR (エクステントディスクリプタページ)、BLOB (非圧縮 BLOB ページ)、COMPRESSED_BLOB (最初の圧縮 BLOB ページ)、COMPRESSED_BLOB2 (後続の圧縮 BLOB ページ)、IBUF_INDEX (挿入バッファーインデックス)、UNKNOWN (不明) のいずれかです。
FLUSH_TYPE	フラッシュタイプ。
FIX_COUNT	バッファープール内でこのブロックを使用するスレッドの数。ゼロのとき、ブロックは削除対象です。
IS_HASHED	ハッシュインデックスがこのページ上に構築されているかどうか。
NEWEST_MODIFICATION	もっとも新しい変更のログシーケンス番号。

カラム名	説明
OLDEST_MODIFICATION	もっとも古い変更のログシーケンス番号。
ACCESS_TIME	ページの最初のアクセス時間の判断に使用される無名数。
TABLE_NAME	ページが属しているテーブルの名前。このカラムは、タイプ INDEX のページにのみ適用可能です。
INDEX_NAME	ページが属しているインデックスの名前。これは、クラスタ化されたインデックスの名前にもセカンダリインデックスの名前にもなります。このカラムは、タイプ INDEX のページにのみ適用可能です。
NUMBER_RECORDS	ページ内のレコードの数。
DATA_SIZE	レコードのサイズの合計。このカラムは、タイプ INDEX のページにのみ適用可能です。
COMPRESSED_SIZE	圧縮されたページサイズ。圧縮されていないページは NULL になります。
PAGE_STATE	ページ状態。有効なデータを伴うページの状態は、FILE_PAGE (ファイルからデータのページをバッファーします)、MEMORY (インメモリーオブジェクトからのページをバッファーします)、COMPRESSED のいずれかです。ほかの可能な (InnoDB によって管理される) 状態は NULL、READY_FOR_USE、NOT_USED、REMOVE_HASH です。
IO_FIX	I/O がこのページに対して保留中であるかどうかを指定します。IO_NONE = 保留中の I/O なし、IO_READ = 読み取り保留中、IO_WRITE = 書き込み保留中です。
IS_OLD	ブロックが LRU リストの古いブロックのサブリストにあるかどうかを指定します。
FREE_PAGE_CLOCK	ブロックが最後に LRU リストの先頭に置かれたときの freed_page_clock カウンタの値。freed_page_clock カウンタは、LRU リストの末尾から削除されたブロックの数を追跡します。

例:

```
mysql> SELECT * FROM INFORMATION_SCHEMA.INNODB_BUFFER_PAGE LIMIT 1\G
*************************** 1. row ***************************
         POOL_ID: 0
       BLOCK_ID: 0
          SPACE: 97
     PAGE_NUMBER: 2473
      PAGE_TYPE: INDEX
     FLUSH_TYPE: 1
      FIX_COUNT: 0
      IS_HASHED: YES
NEWEST_MODIFICATION: 733855581
OLDEST_MODIFICATION: 0
      ACCESS_TIME: 3378385672
       TABLE_NAME: `employees`.`salaries`
       INDEX_NAME: PRIMARY
   NUMBER_RECORDS: 468
        DATA_SIZE: 14976
  COMPRESSED_SIZE: 0
       PAGE_STATE: FILE_PAGE
          IO_FIX: IO_NONE
          IS_OLD: YES
  FREE_PAGE_CLOCK: 66
1 row in set (0.03 sec)
```

注:

- このテーブルは、主に、専門家レベルのパフォーマンスモニタリングや、MySQL のパフォーマンス関連の拡張を開発するときに役立ちます。
- DESCRIBE または SHOW COLUMNS を使用して、データ型とデフォルト値を含む、このテーブルのカラムに関する追加情報を表示します。
- このテーブルをクエリーするには PROCESS 権限が必要です。

21.29.17 INFORMATION_SCHEMA INNODB_BUFFER_PAGE_LRU テーブル

INNODB_BUFFER_PAGE_LRU テーブルは、InnoDB バッファープール内のページに関する情報、特に、いっぱいになったときにバッファープールからどのページをエビクションするかを決定する LRU リスト内の各ページの順序を保持します。

INNODB_BUFFER_PAGE_LRU テーブルには、INNODB_BUFFER_PAGE テーブルと同じカラムがありますが、INNODB_BUFFER_PAGE_LRU テーブルには BLOCK_ID カラムではなく LRU_POSITION カラムがある点が異なります。

関連する使用法と使用例については、セクション14.14.5「InnoDB INFORMATION_SCHEMA バッファープールテーブル」を参照してください。

> **警告**
>
> INNODB_BUFFER_PAGE_LRU テーブルをクエリーすると、大幅なパフォーマンスのオーバーヘッドが生じる可能性があります。クエリーによって発生する可能性のあるパフォーマンスへの影響を認識し、かつそれが許容可能であると判断されていないかぎり、本番システムではこのテーブルをクエリーしないでください。パフォーマンスへの影響を避けるには、調べる問題をテストインスタンスに再現し、テストインスタンスの INNODB_BUFFER_PAGE_LRU テーブルをクエリーします。

表 21.19 INNODB_BUFFER_PAGE_LRU のカラム

カラム名	説明
POOL_ID	バッファープール ID。複数のバッファープールインスタンスを区別する識別子。
LRU_POSITION	LRU リストでのページの位置。
SPACE	テーブルスペース ID。INNODB_SYS_TABLES.SPACE と同じ値を使用します。
PAGE_NUMBER	ページ番号。
PAGE_TYPE	ページタイプ。ALLOCATED (新しく割り当てられたページ)、INDEX (B ツリーノード)、UNDO_LOG (Undo ログページ)、INODE (インデックスノード)、IBUF_FREE_LIST (挿入バッファー空きリスト)、IBUF_BITMAP (挿入バッファービットマップ)、SYSTEM (システムページ)、TRX_SYSTEM (トランザクションシステムデータ)、FILE_SPACE_HEADER (ファイル領域ヘッダー)、EXTENT_DESCRIPTOR (エクステントディスクリプタページ)、BLOB (非圧縮 BLOB ページ)、COMPRESSED_BLOB (最初の圧縮 BLOB ページ)、COMPRESSED_BLOB2 (後続の圧縮 BLOB ページ)、IBUF_INDEX (挿入バッファーインデックス)、UNKNOWN (不明) のいずれかです。
FLUSH_TYPE	フラッシュタイプ。
FIX_COUNT	バッファープール内でこのブロックを使用するスレッドの数。ゼロのとき、ブロックは削除対象です。
IS_HASHED	ハッシュインデックスがこのページ上に構築されているかどうか。
NEWEST_MODIFICATION	もっとも新しい変更のログシーケンス番号。
OLDEST_MODIFICATION	もっとも古い変更のログシーケンス番号。
ACCESS_TIME	ページの最初のアクセス時間の判断に使用される無名数。
TABLE_NAME	ページが属しているテーブルの名前。このカラムは、タイプ INDEX のページにのみ適用可能です。
INDEX_NAME	ページが属しているインデックスの名前。これは、クラスタ化されたインデックスの名前にもセカンダリインデックスの名前にもなります。このカラムは、タイプ INDEX のページにのみ適用可能です。
NUMBER_RECORDS	ページ内のレコードの数。
DATA_SIZE	レコードのサイズの合計。このカラムは、タイプ INDEX のページにのみ適用可能です。
COMPRESSED_SIZE	圧縮されたページサイズ。圧縮されていないページは NULL になります。
PAGE_STATE	ページ状態。有効なデータを伴うページの状態は、FILE_PAGE (ファイルからデータのページをバッファーします)、MEMORY (インメモリーオブジェクトからのページをバッファーします)、COMPRESSED のいずれかです。ほかの可能な (InnoDB によって管理される) 状態は NULL、READY_FOR_USE、NOT_USED、REMOVE_HASH です。
IO_FIX	I/O がこのページに対して保留中であるかどうかを指定します。IO_NONE = 保留中の I/O なし、IO_READ = 読み取り保留中、IO_WRITE = 書き込み保留中です。
IS_OLD	ブロックが LRU リストの古いブロックのサブリストにあるかどうかを指定します。

カラム名	説明
FREE_PAGE_CLOCK	ブロックが最後に LRU リストの先頭に置かれたときの freed_page_clock カウンタの値。freed_page_clock カウンタは、LRU リストの末尾から削除されたブロックの数を追跡します。

例

```
mysql> SELECT * FROM INFORMATION_SCHEMA.INNODB_BUFFER_PAGE_LRU LIMIT 1\G
*************************** 1. row ***************************
          POOL_ID: 0
      LRU_POSITION: 0
             SPACE: 97
       PAGE_NUMBER: 1984
         PAGE_TYPE: INDEX
        FLUSH_TYPE: 1
         FIX_COUNT: 0
         IS_HASHED: YES
NEWEST_MODIFICATION: 719490396
OLDEST_MODIFICATION: 0
       ACCESS_TIME: 3378383796
        TABLE_NAME: `employees`.`salaries`
        INDEX_NAME: PRIMARY
    NUMBER_RECORDS: 468
         DATA_SIZE: 14976
   COMPRESSED_SIZE: 0
        COMPRESSED: NO
            IO_FIX: IO_NONE
            IS_OLD: YES
   FREE_PAGE_CLOCK: 0
1 row in set (0.02 sec)
```

メモ

- このテーブルは、主に、専門家レベルのパフォーマンスモニタリングや、MySQL のパフォーマンス関連の拡張を開発するときに役立ちます。
- DESCRIBE または SHOW COLUMNS を使用して、データ型とデフォルト値を含む、このテーブルのカラムに関する追加情報を表示します。
- このテーブルをクエリーした場合、MySQL は、バッファープール内のアクティブなページ数に 64 バイトをかけた容量以上の連続したメモリーの大きなブロックを割り当てることが必要になる可能性があります。この割り当ては、特に数 G バイトのバッファープールを持つシステムで、メモリー不足エラーを引き起こす可能性があります。
- このテーブルをクエリーした場合、MySQLで、LRU リストのトラバース中、バッファープールを表すデータ構造をロックする必要があります。これにより、特に数 G バイトのバッファープールを持つシステムで並列性を軽減できます。
- このテーブルをクエリーするには PROCESS 権限が必要です。

21.29.18 INFORMATION_SCHEMA INNODB_BUFFER_POOL_STATS テーブル

INNODB_BUFFER_POOL_STATS テーブルは、SHOW ENGINE INNODB STATUS 出力で提供されるものと同じバッファープール情報の多くを提供します。同じ情報の多くは、InnoDB バッファープールのサーバーステータス変数を使用して取得することもできます。

バッファープール内のページに「新しい」または「新しくない」というマークを付けるという考えは、バッファープールデータ構造の先頭と末尾にあるサブリスト間にそれらを転送することを意味します。「新しい」とされたページは、バッファープールから削除されるまでに長い時間がかかりますが、「新しくない」とされたページは、エビクションの時点に近づけられます。

関連する使用法と使用例については、セクション14.14.5「InnoDB INFORMATION_SCHEMA バッファープールテーブル」を参照してください。

表 21.20 INNODB_BUFFER_POOL_STATS のカラム

カラム名	説明
POOL_ID	バッファープール ID。複数のバッファープールインスタンスを区別する一意の識別子。

カラム名	説明
POOL_SIZE	InnoDB バッファープールのサイズ (ページ単位)。
FREE_BUFFERS	InnoDB バッファープール内の空きページの数
DATABASE_PAGES	データを含む InnoDB バッファープール内のページの数。数には、ダーティーページとクリーンページの両方を含みます。
OLD_DATABASE_PAGES	old バッファープールサブリスト内のページの数。
MODIFIED_DATABASE_PAGE	変更された (ダーティー) データベースページの数
PENDING_DECOMPRESS	圧縮解除保留中のページの数
PENDING_READS	読み取り保留中の数
PENDING_FLUSH_LRU	LRU 内のフラッシュ保留中のページの数
PENDING_FLUSH_LIST	フラッシュリスト内のフラッシュ保留中のページの数
PAGES_MADE_YOUNG	新しいとされたページの数
PAGES_NOT_MADE_YOUNG	新しいとされていないページの数
PAGES_MADE_YOUNG_RATE	秒あたりに新しいとされたページの数 (最後の出力/経過時間以降に新しいとされたページ)
PAGES_MADE_NOT_YOUNG_RATE	秒あたりに新しいとされなかったページの数 (最後の出力/経過時間以降に新しいとされなかったページ)
NUMBER_PAGES_READ	読み取られたページの数
NUMBER_PAGES_CREATED	作成されたページの数
NUMBER_PAGES_WRITTEN	書き込まれたページの数
PAGES_READ_RATE	秒あたりに読み取られたページの数 (最後の出力/経過時間以降に読み取られたページ)
PAGES_CREATE_RATE	秒あたりに作成されたページの数 (最後の出力/経過時間以降に作成されたページ)
PAGES_WRITTEN_RATE	秒あたりに書き込まれたページの数 (最後の出力/経過時間以降に書き込まれたページ)
NUMBER_PAGES_GET	論理読み取りリクエスト数。
HIT_RATE	バッファープールのヒット率
YOUNG_MAKE_PER_THOUSAND_GETS	1000 の取得あたりに新しいとされたページの数
NOT_YOUNG_MAKE_PER_THOUSAND_GETS	1000 の取得あたりに新しいとされていないページの数
NUMBER_PAGES_READ_AHEAD	先読みされたページの数
NUMBER_READ_AHEAD_EVICTED	先読みバックグラウンドスレッドによって InnoDB バッファープールに読み取られ、クエリーでアクセスされることなくその後に削除されたページの数。
READ_AHEAD_RATE	秒あたりの先読み率 (最後の出力/経過時間以降に先読みされたページ)
READ_AHEAD_EVICTED_RATE	アクセスなしに削除された、先読みされたページの秒あたりの数 (最後の出力/経過時間以降にアクセスされていない先読みページ)
LRU_IO_TOTAL	LRU IO 合計
LRU_IO_CURRENT	現在の間隔の LRU IO
UNCOMPRESS_TOTAL	圧縮解除されたページの合計数
UNCOMPRESS_CURRENT	現在の間隔で圧縮解除されたページの数

例:

```
mysql> SELECT * FROM INFORMATION_SCHEMA.INNODB_BUFFER_POOL_STATS\G
*************************** 1. row ***************************
                     POOL_ID: 0
                   POOL_SIZE: 8192
                FREE_BUFFERS: 1
              DATABASE_PAGES: 8085
          OLD_DATABASE_PAGES: 2964
      MODIFIED_DATABASE_PAGES: 0
           PENDING_DECOMPRESS: 0
              PENDING_READS: 0
           PENDING_FLUSH_LRU: 0
          PENDING_FLUSH_LIST: 0
```

```
            PAGES_MADE_YOUNG: 22821
        PAGES_NOT_MADE_YOUNG: 3544303
       PAGES_MADE_YOUNG_RATE: 357.62602199870594
   PAGES_MADE_NOT_YOUNG_RATE: 0
         NUMBER_PAGES_READ: 2389
      NUMBER_PAGES_CREATED: 12385
      NUMBER_PAGES_WRITTEN: 13111
          PAGES_READ_RATE: 0
        PAGES_CREATE_RATE: 0
       PAGES_WRITTEN_RATE: 0
         NUMBER_PAGES_GET: 33322210
                 HIT_RATE: 1000
 YOUNG_MAKE_PER_THOUSAND_GETS: 18
NOT_YOUNG_MAKE_PER_THOUSAND_GETS: 0
     NUMBER_PAGES_READ_AHEAD: 2024
   NUMBER_READ_AHEAD_EVICTED: 0
         READ_AHEAD_RATE: 0
    READ_AHEAD_EVICTED_RATE: 0
            LRU_IO_TOTAL: 0
          LRU_IO_CURRENT: 0
        UNCOMPRESS_TOTAL: 0
      UNCOMPRESS_CURRENT: 0
1 row in set (0.00 sec)
```

注:

- このテーブルは、主に、専門家レベルのパフォーマンスモニタリングや、MySQL のパフォーマンス関連の拡張を開発するときに役立ちます。
- DESCRIBE または SHOW COLUMNS を使用して、データ型とデフォルト値を含む、このテーブルのカラムに関する追加情報を表示します。
- このテーブルをクエリーするには PROCESS 権限が必要です。

21.29.19 INFORMATION_SCHEMA INNODB_METRICS テーブル

この INFORMATION_SCHEMA テーブルは、さまざまな InnoDB パフォーマンス情報を表し、InnoDB の PERFORMANCE_SCHEMA テーブルの特定のフォーカス領域を補完します。単純なクエリーで、システムのヘルス全体を確認できます。より詳細なクエリーを使用すると、パフォーマンスのボトルネック、リソース不足、アプリケーション問題などの問題を診断できます。

各モニターは、カウンタ情報を収集するようにインストゥルメントされる InnoDB ソースコード内でのポイントを表します。各カウンタは、開始、停止、およびリセットできます。共通のモジュール名を使用して、カウンタのグループに対して、これらのアクションを実行することもできます。

デフォルトでは、比較的少量のデータが収集されます。カウンタを開始、停止、およびリセットするには、カウンタの名前、モジュールの名前、「%」文字を使用した名前などのワイルドカード一致、特殊キーワード all を使用して、innodb_monitor_enable、innodb_monitor_disable、innodb_monitor_reset、または innodb_monitor_reset_all のいずれかの構成オプションを設定します。

使用法については、セクション14.14.6「InnoDB INFORMATION_SCHEMA メトリックテーブル」を参照してください。

表 21.21 INNODB_METRICS のカラム

カラム名	説明
NAME	カウンタの一意の名前。
SUBSYSTEM	メトリックが適用される InnoDB の側面。SET GLOBAL 構文で使用する対応するモジュール名については、表に続くリストを参照してください。
COUNT	カウンタが有効になってからの値。
MAX_COUNT	カウンタが有効になってからの最大値。
MIN_COUNT	カウンタが有効になってからの最小値。
AVG_COUNT	カウンタが有効になってからの平均値。
COUNT_RESET	最後にリセットされてからのカウンタ値。(_RESET フィールドは、ストップウォッチのラップカウンタのように機能します。一定の時間間隔中のアクティビティーを測定できますが、累積数値は COUNT、MAX_COUNT などのフィールドで引き続き使用できます。)
MAX_COUNT_RESET	最後にリセットされてからの最大カウンタ値。

カラム名	説明
MIN_COUNT_RESET	最後にリセットされてからの最小カウンタ値。
AVG_COUNT_RESET	最後にリセットされてからの平均カウンタ値。
TIME_ENABLED	最後の開始時のタイムスタンプ。
TIME_DISABLED	最後の停止時のタイムスタンプ。
TIME_ELAPSED	カウンタが開始してからの経過時間 (秒単位)。
TIME_RESET	最後の停止時のタイムスタンプ。
STATUS	カウンタがまだ実行中であるのか ()、または停止しているのか ()。
TYPE	項目が累積カウンタであるのか、または一部のリソースの現在値を測定しているのか。
COMMENT	カウンタの説明。

例:

```
mysql>  SELECT * FROM INFORMATION_SCHEMA.INNODB_METRICS WHERE NAME="dml_inserts"\G
*************************** 1. row ***************************
           NAME: dml_inserts
      SUBSYSTEM: dml
          COUNT: 3
      MAX_COUNT: 3
      MIN_COUNT: NULL
      AVG_COUNT: 0.046153846153846156
    COUNT_RESET: 3
MAX_COUNT_RESET: 3
MIN_COUNT_RESET: NULL
AVG_COUNT_RESET: NULL
   TIME_ENABLED: 2014-12-04 14:18:28
  TIME_DISABLED: NULL
   TIME_ELAPSED: 65
     TIME_RESET: NULL
         STATUS: enabled
           TYPE: status_counter
        COMMENT: Number of rows inserted
1 row in set (0.00 sec)
```

注:

- このテーブルをクエリーするには PROCESS 権限が必要です。
- DESCRIBE または SHOW COLUMNS を使用して、データ型とデフォルト値を含む、このテーブルのカラムに関する追加情報を表示します。

21.29.20 INFORMATION_SCHEMA INNODB_FT_CONFIG テーブル

INNODB_FT_CONFIG テーブルは、FULLTEXT インデックスに関するメタデータと、InnoDB テーブルに関連する処理を表示します。

このテーブルをクエリーする前に、構成変数 innodb_ft_aux_table を、test/articles など、FULLTEXT インデックスを含むテーブルの名前 (データベース名など) に設定します。

関連する使用法と使用例については、セクション14.14.4「InnoDB INFORMATION_SCHEMA FULLTEXT インデックステーブル」を参照してください。

表 21.22 INNODB_FT_CONFIG のカラム

カラム名	説明
KEY	FULLTEXT インデックスを含んだ InnoDB テーブルのメタデータの項目を指定する名前。
VALUE	InnoDB テーブルの FULLTEXT インデックスの側面に対する一部の制限または現在値を反映した、KEY カラムに関連付けられた値。

例:

```
mysql> SELECT * FROM INFORMATION_SCHEMA.INNODB_FT_CONFIG;
+-------------------------+------------------+
| KEY                     | VALUE            |
+-------------------------+------------------+
```

```
| optimize_checkpoint_limit | 180              |
| synced_doc_id             | 0                |
| stopword_table_name       | test/my_stopwords |
| use_stopword              | 1                |
+---------------------------+------------------+
4 rows in set (0.00 sec)
```

注:

- この表では、内部構成のみが対象になります。統計情報の使用は想定されていません。
- DESCRIBE または SHOW COLUMNS を使用して、データ型とデフォルト値を含む、このテーブルのカラムに関する追加情報を表示します。
- このテーブルをクエリーするには PROCESS 権限が必要です。
- KEY カラムの値は、InnoDB 全文処理のパフォーマンスチューニングおよびデバッグの必要性に応じて異なる場合があります。現在、キー値は次のものがあります。
  - optimize_checkpoint_limit: OPTIMIZE TABLE の実行が停止するまでの秒数です。
  - synced_doc_id: 次に発行される DOC_ID です。
  - stopword_table_name: ユーザー定義のストップワードテーブルに対する database/table の名前です。ユーザー定義のストップワードテーブルがない場合、このフィールドは空です。
  - use_stopword: ストップワードテーブルを使用するかどうかを示します。これは、FULLTEXT インデックスの作成時に定義されます。
- InnoDB FULLTEXT 検索の詳細は、セクション14.2.13.3「FULLTEXT インデックス」およびセクション12.9「全文検索関数」を参照してください。

21.29.21 INFORMATION_SCHEMA INNODB_FT_DEFAULT_STOPWORD テーブル

INNODB_FT_DEFAULT_STOPWORD テーブルは、InnoDB テーブルでの FULLTEXT インデックスの作成時にデフォルトで使用されるストップワードのリストを保持します。デフォルトの InnoDB ストップワードリストに関する情報と、自身のストップワードリストの定義方法については、セクション12.9.4「全文ストップワード」を参照してください。

関連する使用法と使用例については、セクション14.14.4「InnoDB INFORMATION_SCHEMA FULLTEXT インデックステーブル」を参照してください。

表 21.23 INNODB_FT_DEFAULT_STOPWORD のカラム

カラム名	説明
value	InnoDB テーブルで FULLTEXT インデックスのストップワードとしてデフォルトで使用される単語。innodb_ft_server_stopword_table または innodb_ft_user_stopword_table オプションのどちらかを使用してデフォルトのストップワード処理をオーバーライドする場合は使用されません。

例:

```
mysql> SELECT * FROM INFORMATION_SCHEMA.INNODB_FT_DEFAULT_STOPWORD;
+-------+
| value |
+-------+
| a     |
| about |
| an    |
| are   |
| as    |
| at    |
| be    |
| by    |
| com   |
| de    |
| en    |
| for   |
| from  |
| how   |
```

```
| i    |
| in   |
| is   |
| it   |
| la   |
| of   |
| on   |
| or   |
| that |
| the  |
| this |
| to   |
| was  |
| what |
| when |
| where|
| who  |
| will |
| with |
| und  |
| the  |
| www  |
+-------+
36 rows in set (0.00 sec)
```

注:

- DESCRIBE または SHOW COLUMNS を使用して、データ型とデフォルト値を含む、このテーブルのカラムに関する追加情報を表示します。
- このテーブルをクエリーするには PROCESS 権限が必要です。
- InnoDB FULLTEXT 検索の詳細は、セクション14.2.13.3「FULLTEXT インデックス」およびセクション12.9「全文検索関数」を参照してください。

21.29.22 INFORMATION_SCHEMA INNODB_FT_INDEX_TABLE テーブル

INNODB_FT_INDEX_TABLE テーブルには、InnoDB テーブルの FULLTEXT インデックスに対してテキスト検索を処理するために使用される、転置インデックスに関する情報が表示されます。

このテーブルをクエリーする前に、構成変数 innodb_ft_aux_table を、test/articles など、FULLTEXT インデックスを含むテーブルの名前 (データベース名など) に設定します。

関連する使用法と使用例については、セクション14.14.4「InnoDB INFORMATION_SCHEMA FULLTEXT インデックステーブル」を参照してください。

表 21.24 INNODB_FT_INDEX_TABLE のカラム

カラム名	説明
WORD	FULLTEXT の一部であるカラムのテキストから抽出された単語。
FIRST_DOC_ID	FULLTEXT インデックスでこの単語が最初に表示されるドキュメント ID。
LAST_DOC_ID	FULLTEXT インデックスでこの単語が最後に表示されるドキュメント ID。
DOC_COUNT	FULLTEXT インデックスでこの単語が表示される行の数。同じ単語は、DOC_ID 値と POSITION 値の組み合わせごとに一度ずつ、キャッシュテーブル内で複数回現れる可能性があります。
DOC_ID	単語を含む行のドキュメント ID。この値は、ベースとなるテーブルに対して定義した ID カラムの値を反映することも、テーブルに適切なカラムが含まれないときに InnoDB によって生成されたシーケンス値にすることもできます。
POSITION	DOC_ID 値で識別された関連ドキュメント内のこの単語の特定のインスタンス位置。

注:

- このテーブルは最初から、構成変数 innodb_ft_aux_table の値を設定するまで空です。次の例は、innodb_ft_aux_table オプションを使用して、指定されたテーブルの FULLTEXT インデックスに関する情報を表示する方法について説明します。新しく挿入された行の情報が INNODB_FT_INDEX_TABLE に表示される前に、FULLTEXT インデックスキャッシュは、ディスクにフラッシュされる必要があります。これは、innodb_optimize_fulltext_only=ON でインデックスのついたテーブルで OPTIMIZE TABLE 操作により実行されます。

```
mysql> use test;

mysql> CREATE TABLE articles (
      id INT UNSIGNED AUTO_INCREMENT NOT NULL PRIMARY KEY,
      title VARCHAR(200),
      body TEXT,
      FULLTEXT (title,body)
    ) ENGINE=InnoDB;

mysql> INSERT INTO articles (title,body) VALUES
    ('MySQL Tutorial','DBMS stands for DataBase ...'),
    ('How To Use MySQL Well','After you went through a ...'),
    ('Optimizing MySQL','In this tutorial we will show ...'),
    ('1001 MySQL Tricks','1. Never run mysqld as root. 2. ...'),
    ('MySQL vs. YourSQL','In the following database comparison ...'),
    ('MySQL Security','When configured properly, MySQL ...');

mysql> SET GLOBAL innodb_optimize_fulltext_only=ON;
Query OK, 0 rows affected (0.00 sec)

mysql> OPTIMIZE TABLE articles;
+---------------+----------+----------+----------+
| Table         | Op       | Msg_type | Msg_text |
+---------------+----------+----------+----------+
| test.articles | optimize | status   | OK       |
+---------------+----------+----------+----------+
1 row in set (0.00 sec)

mysql> SET GLOBAL innodb_ft_aux_table = 'test/articles';
Query OK, 0 rows affected (0.00 sec)

mysql> USE INFORMATION_SCHEMA;

mysql> SELECT word, doc_count, doc_id, position FROM INNODB_FT_INDEX_TABLE LIMIT 5;
+------------+-----------+--------+----------+
| word       | doc_count | doc_id | position |
+------------+-----------+--------+----------+
| 1001       |         1 |      4 |        0 |
| after      |         1 |      2 |       22 |
| comparison |         1 |      5 |       44 |
| configured |         1 |      6 |       20 |
| database   |         2 |      1 |       31 |
+------------+-----------+--------+----------+
5 rows in set (0.01 sec)
```

- DESCRIBE または SHOW COLUMNS を使用して、データ型とデフォルト値を含む、このテーブルのカラムに関する追加情報を表示します。
- このテーブルをクエリーするには PROCESS 権限が必要です。
- InnoDB FULLTEXT 検索の詳細は、セクション14.2.13.3「FULLTEXT インデックス」およびセクション12.9「全文検索関数」を参照してください。

21.29.23 INFORMATION_SCHEMA INNODB_FT_INDEX_CACHE テーブル

INNODB_FT_INDEX_CACHE: FULLTEXT インデックス内の新しく挿入された行に関するトークン情報が含まれています。DML 操作中の負荷の大きなインデックスの再編成を避けるために、新しくインデックスが付けられた単語に関する情報は個別に格納され、OPTIMIZE TABLE の実行時、サーバーのシャットダウン時、またはキャッシュサイズが innodb_ft_cache_size や innodb_ft_total_cache_size で定義された制限を超えたときにのみ、メインの検索インデックスと組み合わされます。

このテーブルをクエリーする前に、構成変数 innodb_ft_aux_table を、test/articles など、FULLTEXT インデックスを含むテーブルの名前 (データベース名など) に設定します。

関連する使用法と使用例については、セクション14.14.4「InnoDB INFORMATION_SCHEMA FULLTEXT インデックステーブル」を参照してください。

表 21.25 INNODB_FT_INDEX_CACHE のカラム

カラム名	説明
WORD	新しく挿入された行のテキストから抽出された単語。
FIRST_DOC_ID	FULLTEXT インデックスでこの単語が最初に表示されるドキュメント ID。

カラム名	説明
LAST_DOC_ID	FULLTEXT インデックスでこの単語が最後に表示されるドキュメント ID。
DOC_COUNT	FULLTEXT インデックスでこの単語が表示される行の数。同じ単語は、DOC_ID 値と POSITION 値の組み合わせごとに一度ずつ、キャッシュテーブル内で複数回現れる可能性があります。
DOC_ID	新しく挿入された行のドキュメント ID。この値は、ベースとなるテーブルに対して定義した ID カラムの値を反映することも、テーブルに適切なカラムが含まれないときに InnoDB によって生成されたシーケンス値にすることもできます。
POSITION	DOC_ID 値で識別された関連ドキュメント内のこの単語の特定のインスタンス位置。値は絶対位置を表しません。その単語の 1 つ前のインスタンスの POSITION に追加されたオフセットです。

注:

- このテーブルは最初から、構成変数 innodb_ft_aux_table の値を設定するまで空です。次の例は、innodb_ft_aux_table オプションを使用して、指定されたテーブルの FULLTEXT インデックスに関する情報を表示する方法について説明します。

```
mysql> USE test;

mysql> CREATE TABLE articles (
      id INT UNSIGNED AUTO_INCREMENT NOT NULL PRIMARY KEY,
      title VARCHAR(200),
      body TEXT,
      FULLTEXT (title,body)
    ) ENGINE=InnoDB;

mysql> INSERT INTO articles (title,body) VALUES
    ('MySQL Tutorial','DBMS stands for DataBase ...'),
    ('How To Use MySQL Well','After you went through a ...'),
    ('Optimizing MySQL','In this tutorial we will show ...'),
    ('1001 MySQL Tricks','1. Never run mysqld as root. 2. ...'),
    ('MySQL vs. YourSQL','In the following database comparison ...'),
    ('MySQL Security','When configured properly, MySQL ...');

mysql> SET GLOBAL innodb_ft_aux_table = 'test/articles';
Query OK, 0 rows affected (0.00 sec)

mysql> USE INFORMATION_SCHEMA;

mysql> SELECT word, doc_count, doc_id, position FROM INNODB_FT_INDEX_CACHE LIMIT 5;
+-------------+-----------+--------+----------+
| word        | doc_count | doc_id | position |
+-------------+-----------+--------+----------+
| 1001        |         1 |      4 |        0 |
| after       |         1 |      2 |       22 |
| comparison  |         1 |      5 |       44 |
| configured  |         1 |      6 |       20 |
| database    |         2 |      1 |       31 |
+-------------+-----------+--------+----------+
5 rows in set (0.00 sec)
```

- DESCRIBE または SHOW COLUMNS を使用して、データ型とデフォルト値を含む、このテーブルのカラムに関する追加情報を表示します。
- このテーブルをクエリーするには PROCESS 権限が必要です。
- InnoDB FULLTEXT 検索の詳細は、セクション14.2.13.3「FULLTEXT インデックス」およびセクション12.9「全文検索関数」を参照してください。

21.29.24 INFORMATION_SCHEMA INNODB_FT_DELETED テーブル

INNODB_FT_DELETED テーブルは、InnoDB テーブルの FULLTEXT インデックスから削除された行を記録します。InnoDB FULLTEXT インデックスに対する DML 操作中の負荷の大きなインデックスの再編成を避けるために、新しく削除された単語に関する情報は個別に格納され、テキスト検索を行うときに検索結果からフィルタで除去されて、InnoDB テーブルの OPTIMIZE TABLE ステートメントを発行するときにのみメインの検索インデックスから削除されます。詳細は、InnoDB 全文インデックスの最適化を参照してください。

このテーブルは最初から、構成変数 innodb_ft_aux_table の値を、test/articles など、FULLTEXT インデックスを含むテーブルの名前 (データベース名を含む) に設定するまで空です。

関連する使用法と使用例については、セクション14.14.4「InnoDB INFORMATION_SCHEMA FULLTEXT インデックステーブル」を参照してください。

表 21.26 INNODB_FT_DELETED のカラム

カラム名	説明
DOC_ID	新たに削除された行のドキュメント ID。この値は、ベースとなるテーブルに対して定義した ID カラムの値を反映することも、テーブルに適切なカラムが含まれないときに InnoDB によって生成されたシーケンス値にすることもできます。削除された行のデータが OPTIMIZE TABLE ステートメントによって FULLTEXT インデックスから物理的に削除される前にテキスト検索を行う場合、この値は innodb_ft_index_table テーブルの行をスキップするために使用されます。詳細は、InnoDB 全文インデックスの最適化を参照してください。

例:

```
mysql> SELECT * FROM INFORMATION_SCHEMA.INNODB_FT_DELETED;
+--------+
| DOC_ID |
+--------+
|      6 |
|      7 |
|      8 |
+--------+
3 rows in set (0.00 sec)
```

注:

- DESCRIBE または SHOW COLUMNS を使用して、データ型とデフォルト値を含む、このテーブルのカラムに関する追加情報を表示します。
- このテーブルをクエリーするには PROCESS 権限が必要です。
- InnoDB FULLTEXT 検索の詳細は、セクション14.2.13.3「FULLTEXT インデックス」およびセクション12.9「全文検索関数」を参照してください。

21.29.25 INFORMATION_SCHEMA INNODB_FT_BEING_DELETED テーブル

INNODB_FT_BEING_DELETED テーブルは、OPTIMIZE TABLE の保守操作中にのみ使用される INNODB_FT_DELETED テーブルのスナップショットです。OPTIMIZE TABLE が実行されると、INNODB_FT_BEING_DELETED テーブルは空になり、INNODB_FT_DELETED テーブルから DOC_ID が削除されます。INNODB_FT_BEING_DELETED の内容は一般に有効期間が短いため、モニタリングやデバッグでのこのテーブルの有用性はかぎられます。FULLTEXT インデックスを持つテーブルでの OPTIMIZE TABLE の実行の詳細は、セクション12.9.6「MySQL の全文検索の微調整」を参照してください。

このテーブルは最初から、構成変数 innodb_ft_aux_table の値を設定するまで空です。出力は、INNODB_FT_DELETED テーブルに対して用意された出力例に似ています。

関連する使用法と使用例については、セクション14.14.4「InnoDB INFORMATION_SCHEMA FULLTEXT インデックステーブル」を参照してください。

表 21.27 INNODB_FT_BEING_DELETED のカラム

カラム名	説明
DOC_ID	削除されている過程にある行のドキュメント ID。この値は、ベースとなるテーブルに対して定義した ID カラムの値を反映することも、テーブルに適切なカラムが含まれないときに InnoDB によって生成されたシーケンス値にすることもできます。削除された行のデータが OPTIMIZE TABLE ステートメントによって FULLTEXT インデックスから物理的に削除される前にテキスト検索を行う場合、この値は innodb_ft_index_table テーブルの行をスキップするために使用されます。詳細は、InnoDB 全文インデックスの最適化を参照してください。

注:

- DESCRIBE または SHOW COLUMNS を使用して、データ型とデフォルト値を含む、このテーブルのカラムに関する追加情報を表示します。

- このテーブルをクエリーするには PROCESS 権限が必要です。
- InnoDB FULLTEXT 検索の詳細は、セクション14.2.13.3「FULLTEXT インデックス」およびセクション12.9「全文検索関数」を参照してください。

21.30 MySQL Cluster の INFORMATION_SCHEMA テーブル

次のセクションでは、MySQL Cluster に固有である INFORMATION_SCHEMA テーブルに関する情報について説明します。(FILES テーブルは標準 MySQL 5.6 で使用できますが、ここでは使用していません。)ndb_transid_mysql_connection_map テーブルは、MySQL Cluster バイナリまたはソース内でのみ利用できる INFORMATION_SCHEMA プラグインとして実装されており、MySQL Server 5.6 には存在しません。

MySQL Cluster のトランザクション、操作、スレッド、ブロック、およびその他のパフォーマンスの側面に関する追加統計およびその他のデータは、ndbinfo データベース内のテーブルから取得できます。これらのテーブルの詳細は、セクション18.5.10「ndbinfo MySQL Cluster 情報データベース」を参照してください。

21.30.1 INFORMATION_SCHEMA FILES テーブル

FILES テーブルは MySQL NDB ディスクデータテーブルが格納されるファイルに関する情報を提供します。

> **注記**
>
> このテーブルは、ディスクデータファイルに関する情報だけを提供します。個々の NDB テーブルのディスク領域の割り当てまたは可用性の判断には使用できません。ただし、ndb_desc を使用して、データがディスク上に格納された NDB テーブルごとにどれだけの領域が割り当てられているかとともに、そのテーブルに対しディスク上のデータのストレージに利用できる領域がどれだけ残っているかも表示できます。詳細は、セクション18.4.10「ndb_desc — NDB テーブルの表示」を参照してください。

INFORMATION_SCHEMA 名	SHOW 名	備考
FILE_ID		MySQL 拡張
FILE_NAME		MySQL 拡張
FILE_TYPE		MySQL 拡張
TABLESPACE_NAME		MySQL 拡張
TABLE_CATALOG		MySQL 拡張
TABLE_SCHEMA		MySQL 拡張
TABLE_NAME		MySQL 拡張
LOGFILE_GROUP_NAME		MySQL 拡張
LOGFILE_GROUP_NUMBER		MySQL 拡張
ENGINE		MySQL 拡張
FULLTEXT_KEYS		MySQL 拡張
DELETED_ROWS		MySQL 拡張
UPDATE_COUNT		MySQL 拡張
FREE_EXTENTS		MySQL 拡張
TOTAL_EXTENTS		MySQL 拡張
EXTENT_SIZE		MySQL 拡張
INITIAL_SIZE		MySQL 拡張
MAXIMUM_SIZE		MySQL 拡張
AUTOEXTEND_SIZE		MySQL 拡張
CREATION_TIME		MySQL 拡張
LAST_UPDATE_TIME		MySQL 拡張
LAST_ACCESS_TIME		MySQL 拡張
RECOVER_TIME		MySQL 拡張
TRANSACTION_COUNTER		MySQL 拡張

INFORMATION_SCHEMA 名	SHOW 名	備考
VERSION		MySQL 拡張
ROW_FORMAT		MySQL 拡張
TABLE_ROWS		MySQL 拡張
AVG_ROW_LENGTH		MySQL 拡張
DATA_LENGTH		MySQL 拡張
MAX_DATA_LENGTH		MySQL 拡張
INDEX_LENGTH		MySQL 拡張
DATA_FREE		MySQL 拡張
CREATE_TIME		MySQL 拡張
UPDATE_TIME		MySQL 拡張
CHECK_TIME		MySQL 拡張
CHECKSUM		MySQL 拡張
STATUS		MySQL 拡張
EXTRA		MySQL 拡張

注:

- FILE_ID カラムの値は自動生成されます。
- FILE_NAME は、CREATE LOGFILE GROUP または ALTER LOGFILE GROUP によって作成された UNDO ログファイルの名前か、CREATE TABLESPACE または ALTER TABLESPACE で作成されたデータファイルの名前です。
- FILE_TYPE は、UNDOFILE、DATAFILE、または TABLESPACE のいずれかの値です。
- TABLESPACE_NAME は、ファイルが関連付けられているテーブルスペースの名前です。
- 現在、TABLESPACE_CATALOG カラムの値は常に NULL です。
- TABLE_NAME は、関連するファイルがある場合、そのファイルに関連付けられたディスクデータテーブルの名前です。
- LOGFILE_GROUP_NAME カラムは、ログファイルまたはデータファイルが属しているログファイルのグループに名前を付けます。
- UNDO ログファイルでは、LOGFILE_GROUP_NUMBER には、ログファイルが属しているログファイルグループの自動生成された ID 番号が含まれます。
- MySQL Cluster ディスクデータのログファイルまたはデータファイルに対し、ENGINE カラムの値は常に NDB または NDBCLUSTER になります。
- MySQL Cluster ディスクデータのログファイルまたはデータファイルでは、FULLTEXT_KEYS カラムの値は常に空です。
- FREE EXTENTS カラムには、まだファイルが使用していないエクステントの数が表示されます。TOTAL EXTENTS カラムには、ファイルに割り当てられたエクステントの総数が表示されます。

  これらの 2 つのカラムの差が、現在ファイルで使用されているエクステントの数です。

```
SELECT TOTAL_EXTENTS - FREE_EXTENTS AS extents_used
   FROM INFORMATION_SCHEMA.FILES
   WHERE FILE_NAME = 'myfile.dat';
```

  この差と EXTENT_SIZE カラムの値 (バイト単位でファイルのエクステントのサイズを指定します) とを乗算すると、ファイルで使用されているディスク容量を概算できます。

```
SELECT (TOTAL_EXTENTS - FREE_EXTENTS) * EXTENT_SIZE AS bytes_used
   FROM INFORMATION_SCHEMA.FILES
   WHERE FILE_NAME = 'myfile.dat';
```

  同様に、FREE_EXTENTS と EXTENT_SIZE とを乗算すると、特定のファイルに利用できる残りの領域の容量を見積もることができます。

```
SELECT FREE_EXTENTS * EXTENT_SIZE AS bytes_free
    FROM INFORMATION_SCHEMA.FILES
    WHERE FILE_NAME = 'myfile.dat';
```

> **重要**
>
> 前述のクエリーで生成されたバイト値は概算に過ぎず、その精度は EXTENT_SIZE の値に反比例します。つまり、EXTENT_SIZE が大きくなれば、概算の精度は低くなります。

エクステントがいったん使用されると、エクステントが含まれているデータファイルを破棄せずに再度解放することはできません。これにより、ディスクデータのテーブルからの削除はディスクスペースを解放しないことになります。

エクステントサイズは CREATE TABLESPACE ステートメントで設定できます。詳細は、セクション 13.1.18「CREATE TABLESPACE 構文」を参照してください。

- INITIAL_SIZE カラムには、ファイルのサイズがバイトで表示されます。これは、ファイルの作成に使用された CREATE LOGFILE GROUP、ALTER LOGFILE GROUP、CREATE TABLESPACE、または ALTER TABLESPACE ステートメントの INITIAL_SIZE 句に使用された値と同じです。

  MySQL Cluster ディスクデータファイルの場合、MAXIMUM_SIZE カラムの値は常に、INITIAL_SIZE と同じであり、AUTOEXTEND_SIZE カラムは常に空です。

- CREATION_TIME カラムには、ファイルが作成された日時が表示されます。LAST_UPDATE_TIME カラムには、ファイルが最後に変更された日時が表示されます。LAST_ACCESSED カラムは、ファイルが最後にサーバーによってアクセスされた日時が表示されます。

  現在は、これらのカラムの値はオペレーティングシステムからレポートされたものであり、NDB ストレージエンジンにより提供されたものではありません。オペレーティングシステムから値が提供されていない場合には、これらのカラムには 0000-00-00 00:00:00 が表示されます。

- MySQL クラスタディスクデータのファイルでは、RECOVER_TIME および TRANSACTION_COUNTER カラムは常に 0 です。

- MySQL Cluster ディスクデータファイルでは、次のカラムは常に NULL になります。

  - VERSION
  - ROW_FORMAT
  - TABLE_ROWS
  - AVG_ROW_LENGTH
  - DATA_LENGTH
  - MAX_DATA_LENGTH
  - INDEX_LENGTH
  - DATA_FREE
  - CREATE_TIME
  - UPDATE_TIME
  - CHECK_TIME
  - CHECKSUM

- MySQL Cluster ディスクデータファイルでは、STATUS カラムの値は常に NORMAL です。

- MySQL Cluster ディスクデータファイルでは、各データノードがファイルのコピーを持っているため、EXTRA カラムには、ファイルが属しているデータノードが表示されます。4 つのデータノードを持つ MySQL Cluster で、次のステートメントを使用するとします。

```
CREATE LOGFILE GROUP mygroup
    ADD UNDOFILE 'new_undo.dat'
    INITIAL_SIZE 2G
```

```
   ENGINE NDB;
```

CREATE LOGFILE GROUP ステートメントの実行が成功したあと、次に示すものに似た、FILES テーブルに対するこのクエリーの結果が表示されます。

```
mysql> SELECT LOGFILE_GROUP_NAME, FILE_TYPE, EXTRA
    ->     FROM INFORMATION_SCHEMA.FILES
    ->     WHERE FILE_NAME = 'new_undo.dat';
+--------------------+-------------+----------------+
| LOGFILE_GROUP_NAME | FILE_TYPE   | EXTRA          |
+--------------------+-------------+----------------+
| mygroup            | UNDO FILE   | CLUSTER_NODE=3 |
| mygroup            | UNDO FILE   | CLUSTER_NODE=4 |
| mygroup            | UNDO FILE   | CLUSTER_NODE=5 |
| mygroup            | UNDO FILE   | CLUSTER_NODE=6 |
+--------------------+-------------+----------------+
4 rows in set (0.01 sec)
```

- FILES テーブルは、非標準テーブルです。
- ログファイルグループの作成すると、追加の行が FILES テーブルに存在します。この行は FILE_NAME カラムの値に対して NULL です。この行に対し、FILE_ID カラムの値は常に 0 で、FILE_TYPE カラムの値は常に UNDO FILE で、STATUS カラムの値は常に NORMAL になります。現在、ENGINE カラムの値は常に NDBCLUSTER です。

  この行の FREE_EXTENTS カラムには、名前と番号がそれぞれ LOGFILE_GROUP_NAME カラムと LOGFILE_GROUP_NUMBER カラムに表示される特定のログファイルグループに属す、すべての Undo ファイルで利用可能な空きエクステントの合計数が表示されます。

  MySQL Cluster に既存のログファイルグループが存在せず、次のステートメントを使用してログファイルグループを作成するとします。

```
mysql> CREATE LOGFILE GROUP lg1
    ->   ADD UNDOFILE 'undofile.dat'
    ->   INITIAL_SIZE = 16M
    ->   UNDO_BUFFER_SIZE = 1M
    ->   ENGINE = NDB;
Query OK, 0 rows affected (3.81 sec)
```

  FILES テーブルにクエリーすると、NULL 行が表示されます。

```
mysql> SELECT DISTINCT
    ->   FILE_NAME AS File,
    ->   FREE_EXTENTS AS Free,
    ->   TOTAL_EXTENTS AS Total,
    ->   EXTENT_SIZE AS Size,
    ->   INITIAL_SIZE AS Initial
    ->   FROM INFORMATION_SCHEMA.FILES;
+--------------+---------+---------+------+----------+
| File         | Free    | Total   | Size | Initial  |
+--------------+---------+---------+------+----------+
| undofile.dat |    NULL | 4194304 |    4 | 16777216 |
| NULL         | 4184068 |    NULL |    4 |     NULL |
+--------------+---------+---------+------+----------+
2 rows in set (0.01 sec)
```

  Undo ロギングに利用できる空きエクステントの総数は常に、Undo ファイルの維持に必要なオーバーヘッドのために、ログファイルグループ内のすべての Undo ファイルの TOTAL_EXTENTS カラムの値の合計よりもいくぶん少なくなります。これは、ログファイルグループに 2 番目の Undo ファイルを追加しから、FILES テーブルに対して前述のクエリーを繰り返すと表示できます。

```
mysql> ALTER LOGFILE GROUP lg1
    ->   ADD UNDOFILE 'undofile02.dat'
    ->   INITIAL_SIZE = 4M
    ->   ENGINE = NDB;
Query OK, 0 rows affected (1.02 sec)

mysql> SELECT DISTINCT
    ->   FILE_NAME AS File,
    ->   FREE_EXTENTS AS Free,
    ->   TOTAL_EXTENTS AS Total,
    ->   EXTENT_SIZE AS Size,
    ->   INITIAL_SIZE AS Initial
    ->   FROM INFORMATION_SCHEMA.FILES;
+----------------+---------+---------+------+----------+
| File           | Free    | Total   | Size | Initial  |
```

```
+----------------+---------+---------+------+----------+
| undofile.dat   |    NULL | 4194304 |    4 | 16777216 |
| undofile02.dat |    NULL | 1048576 |    4 |  4194304 |
| NULL           | 5223944 |    NULL |    4 |     NULL |
+----------------+---------+---------+------+----------+
3 rows in set (0.01 sec)
```

このログファイルグループを使用したディスクデータテーブルが Undo ログに利用できる空き領域の容量 (バイト単位) は、空きエクステントの数と初期サイズとを乗算すると概算できます。

```
mysql> SELECT
    ->   FREE_EXTENTS AS 'Free Extents',
    ->   FREE_EXTENTS * EXTENT_SIZE AS 'Free Bytes'
    ->   FROM INFORMATION_SCHEMA.FILES
    ->   WHERE LOGFILE_GROUP_NAME = 'lg1'
    ->   AND FILE_NAME IS NULL;
+--------------+------------+
| Free Extents | Free Bytes |
+--------------+------------+
|      5223944 |   20895776 |
+--------------+------------+
1 row in set (0.02 sec)
```

MySQL Cluster ディスクデータテーブルを作成し、複数の行を挿入した場合、たとえば、その後 Undo ログに使用できる領域がどれだけ残っているかを概算で確認できます。

```
mysql> CREATE TABLESPACE ts1
    ->   ADD DATAFILE 'data1.dat'
    ->   USE LOGFILE GROUP lg1
    ->   INITIAL_SIZE 512M
    ->   ENGINE = NDB;
Query OK, 0 rows affected (8.71 sec)

mysql> CREATE TABLE dd (
    ->   c1 INT NOT NULL PRIMARY KEY,
    ->   c2 INT,
    ->   c3 DATE
    ->   )
    ->   TABLESPACE ts1 STORAGE DISK
    ->   ENGINE = NDB;
Query OK, 0 rows affected (2.11 sec)

mysql> INSERT INTO dd VALUES
    ->   (NULL, 1234567890, '2007-02-02'),
    ->   (NULL, 1126789005, '2007-02-03'),
    ->   (NULL, 1357924680, '2007-02-04'),
    ->   (NULL, 1642097531, '2007-02-05');
Query OK, 4 rows affected (0.01 sec)

mysql> SELECT
    ->   FREE_EXTENTS AS 'Free Extents',
    ->   FREE_EXTENTS * EXTENT_SIZE AS 'Free Bytes'
    ->   FROM INFORMATION_SCHEMA.FILES
    ->   WHERE LOGFILE_GROUP_NAME = 'lg1'
    ->   AND FILE_NAME IS NULL;
+--------------+------------+
| Free Extents | Free Bytes |
+--------------+------------+
|      5207565 |   20830260 |
+--------------+------------+
1 row in set (0.01 sec)
```

- データファイルがテーブルスペースに関連付けられているかどうかを問わず、MySQL Cluster テーブルスペースの追加の行が FILES テーブルに存在します。この行は FILE_NAME カラムの値に対して NULL です。この行に対し、FILE_ID カラムの値は常に 0 で、FILE_TYPE カラムの値は常に TABLESPACE で、STATUS カラムの値は常に NORMAL になります。現在、ENGINE カラムの値は常に NDBCLUSTER です。
- FILES テーブルに関連付けられた SHOW ステートメントはありません。
- MySQL ディスクデータオブジェクトの作成と削除の追加情報と例については、セクション18.5.12「MySQL Cluster ディスクデータテーブル」を参照してください。

21.30.2 INFORMATION_SCHEMA ndb_transid_mysql_connection_map テーブル

ndb_transid_mysql_connection_map テーブルは、NDB トランザクション、NDB トランザクションコーディネータ、API ノードとして MySQL Cluster に接続された MySQL Server 間のマッピングを提供します。この情報

は、ndbinfo MySQL Cluster 情報データベースの server_operations および server_transactions テーブルに移入するときに使用されます。

INFORMATION_SCHEMA 名	SHOW 名	備考
mysql_connection_id		MySQL Server 接続 ID
node_id		トランザクションコーディネータノード ID
ndb_transid		NDB トランザクション ID

mysql_connection_id は、SHOW PROCESSLIST の出力に示された接続またはセッション ID と同じです。

このテーブルに関連付けられた SHOW ステートメントはありません。

これは、MySQL Cluster に固有の非標準テーブルです。これは、INFORMATION_SCHEMA プラグインとして実装されます。SHOW PLUGINS の出力を確認して、このプラグインがサポートされていることを検証できます。ndb_transid_mysql_connection_map サポートが有効である場合、次に (強調テキストを使用して) 示すように、このステートメントの出力には、この名前を持ち、タイプが INFORMATION SCHEMA で、ステータスが ACTIVE であるプラグインが含まれます。

```
mysql> SHOW PLUGINS;
+----------------------------------+--------+--------------------+---------+---------+
| Name                             | Status | Type               | Library | License |
+----------------------------------+--------+--------------------+---------+---------+
| binlog                           | ACTIVE | STORAGE ENGINE     | NULL    | GPL     |
| mysql_native_password            | ACTIVE | AUTHENTICATION     | NULL    | GPL     |
| mysql_old_password               | ACTIVE | AUTHENTICATION     | NULL    | GPL     |
| CSV                              | ACTIVE | STORAGE ENGINE     | NULL    | GPL     |
| MEMORY                           | ACTIVE | STORAGE ENGINE     | NULL    | GPL     |
| MRG_MYISAM                       | ACTIVE | STORAGE ENGINE     | NULL    | GPL     |
| MyISAM                           | ACTIVE | STORAGE ENGINE     | NULL    | GPL     |
| PERFORMANCE_SCHEMA               | ACTIVE | STORAGE ENGINE     | NULL    | GPL     |
| BLACKHOLE                        | ACTIVE | STORAGE ENGINE     | NULL    | GPL     |
| ARCHIVE                          | ACTIVE | STORAGE ENGINE     | NULL    | GPL     |
| ndbcluster                       | ACTIVE | STORAGE ENGINE     | NULL    | GPL     |
| ndbinfo                          | ACTIVE | STORAGE ENGINE     | NULL    | GPL     |
| ndb_transid_mysql_connection_map | ACTIVE | INFORMATION SCHEMA | NULL    | GPL     |
| InnoDB                           | ACTIVE | STORAGE ENGINE     | NULL    | GPL     |
| INNODB_TRX                       | ACTIVE | INFORMATION SCHEMA | NULL    | GPL     |
| INNODB_LOCKS                     | ACTIVE | INFORMATION SCHEMA | NULL    | GPL     |
| INNODB_LOCK_WAITS                | ACTIVE | INFORMATION SCHEMA | NULL    | GPL     |
| INNODB_CMP                       | ACTIVE | INFORMATION SCHEMA | NULL    | GPL     |
| INNODB_CMP_RESET                 | ACTIVE | INFORMATION SCHEMA | NULL    | GPL     |
| INNODB_CMPMEM                    | ACTIVE | INFORMATION SCHEMA | NULL    | GPL     |
| INNODB_CMPMEM_RESET              | ACTIVE | INFORMATION SCHEMA | NULL    | GPL     |
| partition                        | ACTIVE | STORAGE ENGINE     | NULL    | GPL     |
+----------------------------------+--------+--------------------+---------+---------+
22 rows in set (0.00 sec)
```

プラグインはデフォルトで有効になっています。--ndb-transid-mysql-connection-map オプションを付けてサーバーを起動すると、これを無効 (または、プラグインが開始していないかぎりサーバーが実行できないように強制する) にできます。プラグインが無効の場合、SHOW PLUGINS でステータスが DISABLED と表示されます。実行時にプラグインは有効または無効にできません。

このテーブルとカラムの名前は小文字を使用して表示されますが、SQL ステートメントで参照するときには大文字も小文字も使用できます。

このテーブルを作成するには、MySQL Server は、MySQL Cluster 配布で提供されたバイナリか、NDB ストレージエンジンサポートを有効にして MySQL Cluster ソースから構築されたバイナリである必要があります。これは、標準 MySQL 5.6 Server では利用できません。

21.31 スレッドプールの INFORMATION_SCHEMA テーブル

次のセクションでは、スレッドプールプラグインに関連付けられた INFORMATION_SCHEMA テーブルについて説明します。スレッドプール操作に関する情報を示します。

- TP_THREAD_STATE: スレッドプールのスレッド状態に関する情報
- TP_THREAD_GROUP_STATE: スレッドプールのスレッドグループ状態に関する情報
- TP_THREAD_GROUP_STATS: スレッドグループ統計

これらのテーブル内の行は、特定時点のスナップショットを表します。TP_THREAD_STATE の場合、スレッドグループのすべての行で特定時点のスナップショットを構成します。そのため、MySQL Server では、スナップショットの生成中にスレッドグループの相互排他ロックを保持します。ただし、TP_THREAD_STATE に対するステートメントが MySQL Server 全体をブロックできないようにするため、すべてのスレッドグループで同時に相互排他ロックを保持することはありません。

スレッドプール INFORMATION_SCHEMA テーブルは、個々のプラグインによって実装され、特定のテーブルをロードするかどうかは、ほかのテーブルとは無関係に決定できます (セクション8.11.6.1「スレッドプールコンポーネントとインストール」を参照してください)。ただし、すべてのテーブルの内容は、有効なスレッドプールプラグインに応じて異なります。テーブルプラグインが有効だが、スレッドプールプラグインが有効ではない場合、テーブルは表示可能になり、アクセス可能ですが、空になります。

21.31.1 INFORMATION_SCHEMA TP_THREAD_STATE テーブル

このテーブルには、接続を処理するためにスレッドプールで作成されたスレッドごとに 1 行が存在します。テーブルには次のカラムがあります。

- TP_GROUP_ID

  スレッドグループ ID です。

- TP_THREAD_NUMBER

  スレッドグループ内のスレッドの ID です。TP_GROUP_ID と TP_THREAD_NUMBER の組み合わせが、テーブル内での一意のキーになります。

- PROCESS_COUNT

  このスレッドを使用しているステートメントが現在実行している 10 ミリ秒間隔です。0 は実行しているステートメントがないことを示し、1 はステートメントが最初の 10 ミリ秒に存在していることを示す、というようになります。

- WAIT_TYPE

  スレッドの待機のタイプです。NULL はスレッドがブロックされていることを示します。それ以外の場合、スレッドは thd_wait_begin() の呼び出しによってブロックされ、値は待機のタイプを指定します。TP_THREAD_GROUP_STATS テーブルの xxx_WAIT カラムは、待機タイプごとのカウントを累積します。

  WAIT_TYPE 値は、次の表に示すように、待機のタイプを記述した文字列です。

  表 21.28 WAIT_TYPE の値

待機タイプ	意味
THD_WAIT_SLEEP	スリープの待機
THD_WAIT_DISKIO	ディスク IO の待機
THD_WAIT_ROW_LOCK	行ロックの待機
THD_WAIT_GLOBAL_LOCK	グローバルロックの待機
THD_WAIT_META_DATA_LOCK	メタデータロックの待機
THD_WAIT_TABLE_LOCK	テーブルロックの待機
THD_WAIT_USER_LOCK	ユーザーロックの待機
THD_WAIT_BINLOG	binlog の待機
THD_WAIT_GROUP_COMMIT	グループコミットの待機
THD_WAIT_SYNC	fsync の待機

21.31.2 INFORMATION_SCHEMA TP_THREAD_GROUP_STATE テーブル

このテーブルには、スレッドプール内のスレッドグループごとに 1 行存在します。それぞれの行には、グループの現在の状態に関する情報が表示されます。テーブルには次のカラムがあります。

- TP_GROUP_ID

  スレッドグループ ID です。これはテーブル内の一意のキーです。

- CONSUMER THREADS

  コンシューマスレッドの数です。アクティブなスレッドが停止またはブロックされると、実行開始可能なスレッドが最大 1 つ存在します。

- RESERVE_THREADS

  予約状態のスレッドの数です。これは、新しいスレッドをウェイクする必要が生じ、コンシューマスレッドがなくなるまで開始されないことを意味します。これは、通常の操作で必要になる数よりも多くのスレッドをスレッドグループが作成したときに、ほとんどのスレッドが最終的に達する状態です。多くの場合、スレッドグループは短い間追加スレッドを必要とし、その後しばらくの間はふたたび必要とすることはありません。この場合には予約状態になり、ふたたび必要とされるまでその状態のままです。ある程度のメモリーリソースをべつに使用しますが、追加のコンピューティングリソースは必要ありません。

- CONNECTION_COUNT

  このスレッドグループを使用した接続の数です。

- QUEUED_QUERIES

  優先度の高いキューで待機しているステートメントの数です。

- QUEUED_TRANSACTIONS

  優先度の低いキューで待機しているステートメントの数です。これらは、開始されていないトランザクションの初期ステートメントなので、キューに入れられたトランザクションも表します。

- STALL_LIMIT

  スレッドグループに関する thread_pool_stall_limit 変数の値です。これはすべてのスレッドグループで同じ値です。

- PRIO_KICKUP_TIMER

  スレッドグループに関する thread_pool_prio_kickup_timer の値です。これはすべてのスレッドグループで同じ値です。

- ALGORITHM

  スレッドグループに関する thread_pool_algorithm の値です。これはすべてのスレッドグループで同じ値です。

- THREAD_COUNT

  このスレッドグループの一部としてスレッドプール内で開始されたスレッドの数です。

- ACTIVE_THREAD_COUNT

  ステートメントを実行しているアクティブなスレッドの数です。

- MAX_THREAD_IDS_IN_GROUP

  グループ内のスレッドの最大スレッド ID です。これは、TP_THREAD_GROUP_STATE テーブルから選択したときのスレッドの MAX(TP_THREAD_NUMBER) と同じです。つまり、次の 2 つのクエリーは同等です。

```
SELECT TP_GROUP_ID, MAX_THREAD_IDS_IN_GROUP
FROM TP_THREAD_GROUP_STATE;

SELECT TP_GROUP_ID, MAX(TP_THREAD_NUMBER)
FROM TP_THREAD_STATE GROUP BY TP_GROUP_ID;
```

- STALLED_THREAD_COUNT

  停止したスレッドグループ内のステートメントの数です。停止したステートメントが実行している可能性はありますが、スレッドプールから見ると停止して、進行していません。長時間実行しているステートメントはすぐにこのカテゴリで終了します。

- WAITING_THREAD_NUMBER

  スレッドグループ内のステートメントのポーリングを処理するスレッドがある場合、これにより、このスレッドグループ内のスレッド番号が指定されます。このスレッドがステートメントを実行している可能性があります。

- OLDEST_QUEUED

  もっとも古いキューに入れられたステートメントが実行待機した時間です (ミリ秒単位)。

21.31.3 INFORMATION_SCHEMA TP_THREAD_GROUP_STATS テーブル

このテーブルはスレッドグループあたりの統計をレポートします。グループごとに 1 行があります。テーブルには次のカラムがあります。

- TP_GROUP_ID

  スレッドグループ ID です。これはテーブル内の一意のキーです。

- CONNECTIONS_STARTED

  開始した接続の数です。

- CONNECTIONS_CLOSED

  終了した接続の数です。

- QUERIES_EXECUTED

  実行したステートメントの数です。この数は、ステートメントが実行を終了したときではなく、開始したときに増えます。

- QUERIES_QUEUED

  実行を待ってキューに入れられた、受け取ったステートメントの数です。これは、スレッドグループがキューに入れることなく即座に実行を開始できたステートメントはカウントされません。即座に実行を開始できるのは、セクション8.11.6.2「スレッドプール操作」に記した条件に該当する場合です。

- THREADS_STARTED

  開始したスレッドの数です。

- PRIO_KICKUPS

  thread_pool_prio_kickup_timer システム変数の値に基づいて、優先度の低いキューから優先度の高いキューに移動したステートメントの数です。この数が急速に増えた場合、変数の値を増やしてください。急速に増えるカウンタは、トランザクションが非常に早くから開始しないようにする優先度システムが機能していないことを意味します。InnoDB の場合、これは、同時トランザクションが非常に多いためにパフォーマンスが低下している可能性が高くなっています。

- STALLED_QUERIES_EXECUTED

  thread_pool_stall_limit システム変数の値よりも長い時間実行しているため、停止したと定義されるようになったステートメントの数です。

- BECOME_CONSUMER_THREAD

  コンシューマスレッドロールがスレッドに割り当てられた回数です。

- BECOME_RESERVE_THREAD

  予約スレッドロールがスレッドに割り当てられた回数です。

- BECOME_WAITING_THREAD

  待機スレッドロールがスレッドに割り当てられた回数です。ステートメントがキューに入れられると、これは、通常の操作であっても、非常に頻繁に起こります。したがって、ステートメントがキューに入れられたシステムの負荷が高い場合には、この値が急速に増加しても正常です。

- WAKE_THREAD_STALL_CHECKER

  複数のステートメントをできるだけ処理したり、待機スレッドロールに対処したりするために、スレッドのウェイクアップまたは作成を、停止チェックスレッドで決定した回数です。

- SLEEP_WAITS

THD_WAIT_SLEEP 待機の数です。たとえば SLEEP() 関数を呼び出すことによってスレッドがスリープになると発生します。

- DISK_IO_WAITS

  THD_WAIT_DISKIO 待機の数です。ファイルシステムキャッシュにヒットしない可能性のあるディスク I/O をスレッドが実行すると発生します。ファイルに対する通常の読み取りおよび書き込みの場合ではなく、バッファープールがディスクに対してデータを読み取りおよび書き込むときに、このような待機が起こります。

- ROW_LOCK_WAITS

  別のトランザクションによる行ロックの解放を待っている THD_WAIT_ROW_LOCK 待機の数です。

- GLOBAL_LOCK_WAITS

  グローバルロックの解放を待っている THD_WAIT_GLOBAL_LOCK 待機の数です。

- META_DATA_LOCK_WAITS

  メタデータロックの解放を待っている THD_WAIT_META_DATA_LOCK 待機の数です。

- TABLE_LOCK_WAITS

  ステートメントがアクセスする必要のあるテーブルのロック解除を待っている THD_WAIT_TABLE_LOCK 待機の数です。

- USER_LOCK_WAITS

  ユーザースレッドで構築された固有のロックを待っている THD_WAIT_USER_LOCK 待機の数です。

- BINLOG_WAITS

  バイナリログの解放を待っている THD_WAIT_BINLOG_WAITS 待機の数です。

- GROUP_COMMIT_WAITS

  THD_WAIT_GROUP_COMMIT 待機の数です。その他のパーティーがトランザクションの担当分を完了するまで、グループコミットが待機する必要があるときに発生します。

- FSYNC_WAITS

  ファイル同期操作を待っている THD_WAIT_SYNC 待機の数です。

21.32 SHOW ステートメントの拡張

SHOW ステートメントの一部の拡張は、INFORMATION_SCHEMA の実装を伴います。

- SHOW を使用すると、INFORMATION_SCHEMA 自体の構造に関する情報を取得できます。
- いくつかの SHOW ステートメントでは、表示する行をより柔軟に指定できる WHERE 句を使用できます。

INFORMATION_SCHEMA は情報データベースなので、その名前は SHOW DATABASES の出力に含まれます。同様に、SHOW TABLES を INFORMATION_SCHEMA と一緒に使用すると、テーブルのリストを取得できます。

```
mysql> SHOW TABLES FROM INFORMATION_SCHEMA;
+---------------------------------------+
| Tables_in_INFORMATION_SCHEMA          |
+---------------------------------------+
| CHARACTER_SETS                        |
| COLLATIONS                            |
| COLLATION_CHARACTER_SET_APPLICABILITY |
| COLUMNS                               |
| COLUMN_PRIVILEGES                     |
| ENGINES                               |
| EVENTS                                |
| FILES                                 |
| GLOBAL_STATUS                         |
| GLOBAL_VARIABLES                      |
| KEY_COLUMN_USAGE                      |
```

```
| PARTITIONS                            |
| PLUGINS                               |
| PROCESSLIST                           |
| REFERENTIAL_CONSTRAINTS               |
| ROUTINES                              |
| SCHEMATA                              |
| SCHEMA_PRIVILEGES                     |
| SESSION_STATUS                        |
| SESSION_VARIABLES                     |
| STATISTICS                            |
| TABLES                                |
| TABLE_CONSTRAINTS                     |
| TABLE_PRIVILEGES                      |
| TRIGGERS                              |
| USER_PRIVILEGES                       |
| VIEWS                                 |
+---------------------------------------+
27 rows in set (0.00 sec)
```

SHOW COLUMNS および DESCRIBE は個々の INFORMATION_SCHEMA テーブルのカラムに関する情報を表示できます。

表示される行を制限する LIKE 句を受け入れる SHOW ステートメントは、選択した行が満たす必要のあるより一般的な条件を指定する WHERE 句も許可します。

```
SHOW CHARACTER SET
SHOW COLLATION
SHOW COLUMNS
SHOW DATABASES
SHOW FUNCTION STATUS
SHOW INDEX
SHOW OPEN TABLES
SHOW PROCEDURE STATUS
SHOW STATUS
SHOW TABLE STATUS
SHOW TABLES
SHOW TRIGGERS
SHOW VARIABLES
```

WHERE 句がある場合、これは SHOW ステートメントで表示されるカラム名に対して評価されます。たとえば、SHOW CHARACTER SET ステートメントはこれらの出力カラムを生成します。

```
mysql> SHOW CHARACTER SET;
+----------+-----------------------------+---------------------+--------+
| Charset  | Description                 | Default collation   | Maxlen |
+----------+-----------------------------+---------------------+--------+
| big5     | Big5 Traditional Chinese    | big5_chinese_ci     |      2 |
| dec8     | DEC West European           | dec8_swedish_ci     |      1 |
| cp850    | DOS West European           | cp850_general_ci    |      1 |
| hp8      | HP West European            | hp8_english_ci      |      1 |
| koi8r    | KOI8-R Relcom Russian       | koi8r_general_ci    |      1 |
| latin1   | cp1252 West European        | latin1_swedish_ci   |      1 |
| latin2   | ISO 8859-2 Central European | latin2_general_ci   |      1 |
...
```

WHERE 句を SHOW CHARACTER SET と一緒に使用するには、これらのカラム名を参照します。たとえば、次のステートメントは、デフォルトの照合順序が文字列 'japanese' を含む文字セットに関する情報を表示します。

```
mysql> SHOW CHARACTER SET WHERE `Default collation` LIKE '%japanese%';
+---------+---------------------------+---------------------+--------+
| Charset | Description               | Default collation   | Maxlen |
+---------+---------------------------+---------------------+--------+
| ujis    | EUC-JP Japanese           | ujis_japanese_ci    |      3 |
| sjis    | Shift-JIS Japanese        | sjis_japanese_ci    |      2 |
| cp932   | SJIS for Windows Japanese | cp932_japanese_ci   |      2 |
| eucjpms | UJIS for Windows Japanese | eucjpms_japanese_ci |      3 |
+---------+---------------------------+---------------------+--------+
```

このステートメントはマルチバイト文字セットを表示します。

```
mysql> SHOW CHARACTER SET WHERE Maxlen > 1;
+---------+---------------------------+---------------------+--------+
| Charset | Description               | Default collation   | Maxlen |
+---------+---------------------------+---------------------+--------+
| big5    | Big5 Traditional Chinese  | big5_chinese_ci     |      2 |
| ujis    | EUC-JP Japanese           | ujis_japanese_ci    |      3 |
| sjis    | Shift-JIS Japanese        | sjis_japanese_ci    |      2 |
| euckr   | EUC-KR Korean             | euckr_korean_ci     |      2 |
```

```
| gb2312  | GB2312 Simplified Chinese | gb2312_chinese_ci   |      2 |
| gbk     | GBK Simplified Chinese    | gbk_chinese_ci      |      2 |
| utf8    | UTF-8 Unicode             | utf8_general_ci     |      3 |
| ucs2    | UCS-2 Unicode             | ucs2_general_ci     |      2 |
| cp932   | SJIS for Windows Japanese | cp932_japanese_ci   |      2 |
| eucjpms | UJIS for Windows Japanese | eucjpms_japanese_ci |      3 |
+---------+---------------------------+---------------------+--------+
```

第 22 章 MySQL パフォーマンススキーマ

目次

22.1 パフォーマンススキーマクイックスタート 2566
22.2 パフォーマンススキーマ構成 2571
22.2.1 パフォーマンススキーマビルド構成 2571
22.2.2 パフォーマンススキーマ起動構成 2572
22.2.3 パフォーマンススキーマ実行時構成 2574
22.3 パフォーマンススキーマクエリー 2589
22.4 パフォーマンススキーマインストゥルメント命名規則 2589
22.5 パフォーマンススキーマステータスモニタリング 2591
22.6 パフォーマンススキーマの原子的および分子的イベント 2594
22.7 パフォーマンススキーマステートメントダイジェスト 2595
22.8 パフォーマンススキーマの一般的なテーブル特性 2596
22.9 パフォーマンススキーマテーブルの説明 2596
22.9.1 パフォーマンススキーマテーブルインデックス 2597
22.9.2 パフォーマンススキーマセットアップテーブル 2598
22.9.3 パフォーマンススキーマインスタンステーブル 2602
22.9.4 パフォーマンススキーマ待機イベントテーブル 2606
22.9.5 パフォーマンススキーマステージイベントテーブル 2609
22.9.6 パフォーマンススキーマステートメントイベントテーブル 2612
22.9.7 パフォーマンススキーマ接続テーブル 2617
22.9.8 パフォーマンススキーマ接続属性テーブル 2620
22.9.9 パフォーマンススキーマサマリーテーブル 2620
22.9.10 パフォーマンススキーマのその他のテーブル 2631
22.10 パフォーマンススキーマオプションおよび変数リファレンス 2637
22.11 パフォーマンススキーマコマンドオプション 2639
22.12 パフォーマンススキーマシステム変数 2640
22.13 パフォーマンススキーマステータス変数 2648
22.14 パフォーマンススキーマとプラグイン 2650
22.15 問題を診断するためのパフォーマンススキーマの使用 2650

MySQL パフォーマンススキーマは低レベルで MySQL サーバーの実行をモニタリングするための機能です。パフォーマンススキーマには、これらの特性があります。

- パフォーマンススキーマは、実行時にサーバーの内部実行を検査する方法を提供します。それは、PERFORMANCE_SCHEMA ストレージエンジンおよび performance_schema データベースを使用して実装されます。パフォーマンススキーマは、主にパフォーマンスデータに焦点を合わせています。これは、メタデータの検査に使用される INFORMATION_SCHEMA と異なります。

- パフォーマンススキーマはサーバーイベントをモニターします。「イベント」は、時間がかかり、タイミング情報を収集できるようにインストゥルメントされた、サーバーが実行するすべてのことです。一般に、イベントは、関数呼び出し、オペレーティングシステムの待機、解析やソートなどの SQL ステートメント実行のステージ、またはステートメント全体やステートメントのグループなどになります。現在、イベントコレクションは、サーバーおよびいくつかのストレージエンジンの同期呼び出し (相互排他ロックのためなど) のファイルおよびテーブル I/O、テーブルロックなどに関する情報へのアクセスを提供します。

- パフォーマンススキーマイベントは、サーバーのバイナリログに書き込まれるイベント (これはデータの変更を説明する) やイベントスケジューラのイベント (これはストアドプログラムの一種である) とは区別されます。

- パフォーマンススキーマイベントは MySQL サーバーの特定のインスタンスに固有です。MySQL 5.6.9 以降では、パフォーマンススキーマテーブルはサーバーにローカルとみなされ、それらへの変更はバイナリログにレプリケートされたり、書き込まれたりしません。(Bug #14741537)

- 現在のイベントに加えて、イベントの履歴とサマリーを取得できます。これにより、インストゥルメントされたアクティビティーが何回実行され、それらにどのくらいの時間がかかったかを判断できます。イベント情報を取得して、特定のスレッドのアクティビティー、または相互排他ロックやファイルなどの特定のオブジェクトに関連付けられているアクティビティーを表示できます。

- PERFORMANCE_SCHEMA ストレージエンジンは、サーバーソースコードで「インストゥルメンテーションポイント」を使用して、イベントデータを収集します。

- 収集されたイベントは、performance_schema データベース内のテーブルに格納されます。これらのテーブルは、ほかのテーブルと同様に SELECT ステートメントを使用してクエリーできます。
- パフォーマンススキーマの構成は、SQL ステートメントから performance_schema データベース内のテーブルを更新することによって、動的に変更できます。構成の変更はデータコレクションにすぐに影響しません。
- performance_schema データベース内のテーブルは、永続的なディスク上ストレージを使用しないビューや一時テーブルです。
- モニタリングは MySQL でサポートされているすべてのプラットフォームで使用できます。

  いくつかの制限が適用されることがあります。タイマーの種類はプラットフォームごとに異なることがあります。ストレージエンジンに適用されるインストゥルメントは、すべてのストレージエンジンに実装されていないことがあります。各サードパーティーエンジンのインストゥルメンテーションはエンジン管理者の責任です。セクションD.8「パフォーマンススキーマの制約」も参照してください。
- データコレクションは、サーバーソースコードを変更し、インストゥルメンテーションを追加することによって実装されます。レプリケーションやイベントスケジューラなどのほかの機能と異なり、パフォーマンススキーマに関連付けられた個別のスレッドはありません。

パフォーマンススキーマは、サーバーパフォーマンスに与える影響を最小にしながら、サーバー実行に関する有益な情報へのアクセスを提供することを目的としています。実装はこれらの設計目標に従います。

- パフォーマンススキーマのアクティブ化によって、サーバーの動作は変更されません。たとえば、それによってスレッドスケジューリングが変更されず、クエリー実行計画 (EXPLAIN によって表示される) が変更されません。
- サーバーの起動時に行われた以上のメモリー割り当ては行われません。固定サイズでの構造の早期割り当てを使用することで、それらのサイズ変更や再割り当てをする必要がなくなりますが、これは十分な実行時パフォーマンスの達成に重要です。
- サーバーのモニタリングはごくわずかなオーバーヘッドで、継続的かつ目立たずに行われます。パフォーマンススキーマのアクティブ化によって、サーバーが使用不能になりません。
- パーサーは変更されません。新しいキーワードやステートメントはありません。
- パフォーマンススキーマが内部で失敗しても、サーバーコードの実行は正常に続行されます。
- 初期のイベントの収集時に処理を実行するか、あとのイベント取得時に処理を実行するかのどちらかを選択する場合、収集が速くなるほうが優先されます。これは、取得がオンデマンドで、まったく行われないこともあるのに対し、収集は進行中であるためです。
- 新しいインストゥルメンテーションポイントを追加することは簡単です。
- インストゥルメンテーションはバージョン管理されます。インストゥルメンテーションの実装が変更された場合、以前にインストゥルメントされたコードが動作し続けます。これは、最新のパフォーマンススキーマの変更と同期させておくために、各プラグインをアップグレードする必要がないため、サードパーティープラグインの開発者にメリットがあります。

22.1 パフォーマンススキーマクイックスタート

このセクションでは、その使用方法を示す例によって、パフォーマンススキーマについて簡単に紹介します。追加の例については、セクション22.15「問題を診断するためのパフォーマンススキーマの使用」を参照してください。

パフォーマンススキーマを使用できるようにするには、MySQL の構築時にそのサポートが構成されている必要があります。これが当てはまるかどうかは、サーバーのヘルプ出力をチェックして確認できます。パフォーマンススキーマを使用できる場合、出力に、performance_schema で始まる名前の付いたいくつかの変数が示されます。

```
shell> mysqld --verbose --help
...
  --performance_schema
                      Enable the performance schema.
  --performance_schema_events_waits_history_long_size=#
                      Number of rows in events_waits_history_long.
...
```

そのような変数が出力に表示されない場合、サーバーはパフォーマンススキーマをサポートするように構築されていません。この場合は、セクション22.2「パフォーマンススキーマ構成」を参照してください。

パフォーマンススキーマを使用できるものとして、MySQL 5.6.6 以降、それはデフォルトで有効にされます。5.6.6 より前では、それはデフォルトで無効にされます。それを明示的に有効または無効にするには、performance_schema 変数を適切な値に設定して、サーバーを起動します。たとえば、my.cnf ファイルでこれらの行を使用します。

```
[mysqld]
performance_schema=on
```

サーバーは起動すると、performance_schema を確認し、パフォーマンススキーマの初期化を試みます。初期化の成功を確認するには、このステートメントを使用します。

```
mysql> SHOW VARIABLES LIKE 'performance_schema';
+--------------------+-------+
| Variable_name      | Value |
+--------------------+-------+
| performance_schema | ON    |
+--------------------+-------+
```

ON の値はパフォーマンススキーマが正常に初期化され、使用する準備ができていることを意味します。OFF の値は何らかのエラーが発生していることを意味します。何に異常が発生したかに関する情報については、サーバーエラーログをチェックしてください。

パフォーマンススキーマはストレージエンジンとして実装されます。このエンジンを使用できる場合 (先にすでにチェックしているはずです)、INFORMATION_SCHEMA.ENGINES テーブルまたは SHOW ENGINES ステートメントからの出力に、SUPPORT 値が YES でそれが表示されていることでわかるはずです。

```
mysql> SELECT * FROM INFORMATION_SCHEMA.ENGINES
    -> WHERE ENGINE='PERFORMANCE_SCHEMA'\G
*************************** 1. row ***************************
      ENGINE: PERFORMANCE_SCHEMA
     SUPPORT: YES
     COMMENT: Performance Schema
TRANSACTIONS: NO
          XA: NO
  SAVEPOINTS: NO

mysql> SHOW ENGINES\G
...
      Engine: PERFORMANCE_SCHEMA
     Support: YES
     Comment: Performance Schema
Transactions: NO
          XA: NO
  Savepoints: NO
...
```

PERFORMANCE_SCHEMA ストレージエンジンは、performance_schema データベース内のテーブルを操作します。そのテーブルへの参照をデータベース名で修飾する必要がないように、performance_schema をデフォルトのデータベースにすることができます。

```
mysql> USE performance_schema;
```

この章の多くの例では、performance_schema がデフォルトのデータベースであるとしています。

パフォーマンススキーマテーブルは performance_schema データベースに格納されます。このデータベースとそのテーブルの構造に関する情報を取得するには、ほかのすべてのデータベースのように、INFORMATION_SCHEMA データベースから選択するか、SHOW ステートメントを使用します。たとえば、どのパフォーマンススキーマテーブルが存在するか確認するには、これらのいずれかのステートメントを使用します。

```
mysql> SELECT TABLE_NAME FROM INFORMATION_SCHEMA.TABLES
    -> WHERE TABLE_SCHEMA = 'performance_schema';
+----------------------------------------------------+
| TABLE_NAME                                         |
+----------------------------------------------------+
| accounts                                           |
| cond_instances                                     |
| events_stages_current                              |
| events_stages_history                              |
| events_stages_history_long                         |
| events_stages_summary_by_account_by_event_name     |
| events_stages_summary_by_host_by_event_name        |
| events_stages_summary_by_thread_by_event_name      |
| events_stages_summary_by_user_by_event_name        |
| events_stages_summary_global_by_event_name         |
```

```
| events_statements_current                  |
| events_statements_history                  |
| events_statements_history_long             |
...
| file_instances                             |
| file_summary_by_event_name                 |
| file_summary_by_instance                   |
| host_cache                                 |
| hosts                                      |
| mutex_instances                            |
| objects_summary_global_by_type             |
| performance_timers                         |
| rwlock_instances                           |
| session_account_connect_attrs              |
| session_connect_attrs                      |
| setup_actors                               |
| setup_consumers                            |
| setup_instruments                          |
| setup_objects                              |
| setup_timers                               |
| socket_instances                           |
| socket_summary_by_event_name               |
| socket_summary_by_instance                 |
| table_io_waits_summary_by_index_usage      |
| table_io_waits_summary_by_table            |
| table_lock_waits_summary_by_table          |
| threads                                    |
| users                                      |
+--------------------------------------------+

mysql> SHOW TABLES FROM performance_schema;
+--------------------------------------------+
| Tables_in_performance_schema               |
+--------------------------------------------+
| accounts                                   |
| cond_instances                             |
| events_stages_current                      |
| events_stages_history                      |
| events_stages_history_long                 |
...
```

パフォーマンススキーマテーブルの数は、追加のインストゥルメンテーションの実装が進行するにつれて、時間とともに増加することが予想されます。

performance_schema データベースの名前は小文字で、その中のテーブルの名前も同様です。クエリーでは名前を小文字で指定してください。

個々のテーブルの構造を表示するには、SHOW CREATE TABLE を使用します。

```
mysql> SHOW CREATE TABLE setup_timers\G
*************************** 1. row ***************************
       Table: setup_timers
Create Table: CREATE TABLE `setup_timers` (
  `NAME` varchar(64) NOT NULL,
  `TIMER_NAME` enum('CYCLE','NANOSECOND','MICROSECOND','MILLISECOND','TICK')
    NOT NULL
) ENGINE=PERFORMANCE_SCHEMA DEFAULT CHARSET=utf8
```

テーブル構造は、INFORMATION_SCHEMA.COLUMNS などのテーブルから選択するか、SHOW COLUMNS などのステートメントを使用して取得することもできます。

performance_schema データベース内のテーブルはそれらの中の情報の種類 (現在のイベント、イベント履歴およびサマリー、オブジェクトインスタンス、およびセットアップ (構成) 情報) に従ってグループ化できます。次の例に、これらのテーブルのいくつかの使用方法を示します。各グループのテーブルに関する詳細については、セクション22.9「パフォーマンススキーマテーブルの説明」を参照してください。

最初に、すべてのインストゥルメントとコンシューマが有効にされていないため、パフォーマンススキーマはすべてのイベントを収集しません。これらのすべてをオンにし、イベントタイミングを有効にするには、2 つのステートメントを実行します (行のカウントは MySQL バージョンによって異なることがあります)。

```
mysql> UPDATE setup_instruments SET ENABLED = 'YES', TIMED = 'YES';
Query OK, 338 rows affected (0.12 sec)
mysql> UPDATE setup_consumers SET ENABLED = 'YES';
Query OK, 8 rows affected (0.00 sec)
```

現在サーバーが何を行なっているかを確認するには、events_waits_current テーブルを調査します。それには、スレッドごとに、各スレッドの最新のモニターされたイベントを示す 1 行が含まれます。

```
mysql> SELECT * FROM events_waits_current\G
*************************** 1. row ***************************
           THREAD_ID: 0
            EVENT_ID: 5523
          EVENT_NAME: wait/synch/mutex/mysys/THR_LOCK::mutex
              SOURCE: thr_lock.c:525
         TIMER_START: 201660494489586
           TIMER_END: 201660494576112
          TIMER_WAIT: 86526
               SPINS: NULL
       OBJECT_SCHEMA: NULL
         OBJECT_NAME: NULL
         OBJECT_TYPE: NULL
OBJECT_INSTANCE_BEGIN: 142270668
    NESTING_EVENT_ID: NULL
          OPERATION: lock
     NUMBER_OF_BYTES: NULL
               FLAGS: 0
...
```

このイベントは、スレッド 0 が THR_LOCK::mutex のロック、mysys サブシステム内の相互排他ロックを獲得するために、86,526 ピコ秒待機していたことを示しています。最初のいくつかのカラムは次の情報を提供します。

- ID カラムはイベントの発生元のスレッドとイベント番号を示します。
- EVENT_NAME はインストゥルメントされたものを示し、SOURCE は、インストゥルメントされたコードを含むソースファイルを示します。
- タイマーカラムはイベントが開始および停止したタイミングとそれにかかった時間を示します。イベントがまだ進行中の場合は、TIMER_END と TIMER_WAIT の値が NULL になります。タイマー値は概算で、ピコ秒で表されます。タイマーおよびイベント時間コレクションについては、セクション22.2.3.1「パフォーマンススキーマイベントタイミング」を参照してください。

履歴テーブルには、現在のイベントテーブルと同じ種類の行が含まれますが、ほかの行もあり、サーバーが「現在」ではなく、「最近」何を実行していたかが示されます。events_waits_history および events_waits_history_long テーブルにはスレッドごとに最新の 10 イベントと最新の 10,000 イベントがそれぞれ含まれます。たとえば、スレッド 13 によって生成された最新イベントの情報を表示するには、次を実行します。

```
mysql> SELECT EVENT_ID, EVENT_NAME, TIMER_WAIT
    -> FROM events_waits_history WHERE THREAD_ID = 13
    -> ORDER BY EVENT_ID;
+----------+-----------------------------------------+------------+
| EVENT_ID | EVENT_NAME                              | TIMER_WAIT |
+----------+-----------------------------------------+------------+
|       86 | wait/synch/mutex/mysys/THR_LOCK::mutex  |     686322 |
|       87 | wait/synch/mutex/mysys/THR_LOCK_malloc  |     320535 |
|       88 | wait/synch/mutex/mysys/THR_LOCK_malloc  |     339390 |
|       89 | wait/synch/mutex/mysys/THR_LOCK_malloc  |     377100 |
|       90 | wait/synch/mutex/sql/LOCK_plugin        |     614673 |
|       91 | wait/synch/mutex/sql/LOCK_open          |     659925 |
|       92 | wait/synch/mutex/sql/THD::LOCK_thd_data |     494001 |
|       93 | wait/synch/mutex/mysys/THR_LOCK_malloc  |     222489 |
|       94 | wait/synch/mutex/mysys/THR_LOCK_malloc  |     214947 |
|       95 | wait/synch/mutex/mysys/LOCK_alarm       |     312993 |
+----------+-----------------------------------------+------------+
```

履歴テーブルに新しいイベントが追加されると、テーブルがいっぱいである場合、古いイベントが破棄されます。

サマリーテーブルは、時間をかけてすべてのイベントについて集計された情報を提供します。このグループのテーブルには、さまざまな方法で、イベントデータが要約されます。もっとも多くの回数実行されたか、またはもっとも待機時間がかかったインストゥルメントを確認するには、COUNT_STAR または SUM_TIMER_WAIT カラムで events_waits_summary_global_by_event_name テーブルをソートします。これらのカラムはすべてのイベント全体で計算された、COUNT(*) または SUM(TIMER_WAIT) 値にそれぞれ対応します。

```
mysql> SELECT EVENT_NAME, COUNT_STAR
    -> FROM events_waits_summary_global_by_event_name
    -> ORDER BY COUNT_STAR DESC LIMIT 10;
+---------------------------------------------+------------+
| EVENT_NAME                                  | COUNT_STAR |
+---------------------------------------------+------------+
| wait/synch/mutex/mysys/THR_LOCK_malloc      |       6419 |
| wait/io/file/sql/FRM                        |        452 |
| wait/synch/mutex/sql/LOCK_plugin            |        337 |
| wait/synch/mutex/mysys/THR_LOCK_open        |        187 |
| wait/synch/mutex/mysys/LOCK_alarm           |        147 |
```

```
| wait/synch/mutex/sql/THD::LOCK_thd_data          |       115 |
| wait/io/file/myisam/kfile                        |       102 |
| wait/synch/mutex/sql/LOCK_global_system_variables |        89 |
| wait/synch/mutex/mysys/THR_LOCK::mutex           |        89 |
| wait/synch/mutex/sql/LOCK_open                   |        88 |
+--------------------------------------------------+-----------+

mysql> SELECT EVENT_NAME, SUM_TIMER_WAIT
    -> FROM events_waits_summary_global_by_event_name
    -> ORDER BY SUM_TIMER_WAIT DESC LIMIT 10;
+----------------------------------------+---------------+
| EVENT_NAME                             | SUM_TIMER_WAIT |
+----------------------------------------+---------------+
| wait/io/file/sql/MYSQL_LOG             |    1599816582 |
| wait/synch/mutex/mysys/THR_LOCK_malloc |    1530083250 |
| wait/io/file/sql/binlog_index          |    1385291934 |
| wait/io/file/sql/FRM                   |    1292823243 |
| wait/io/file/myisam/kfile              |     411193611 |
| wait/io/file/myisam/dfile              |     322401645 |
| wait/synch/mutex/mysys/LOCK_alarm      |     145126935 |
| wait/io/file/sql/casetest              |     104324715 |
| wait/synch/mutex/sql/LOCK_plugin       |      86027823 |
| wait/io/file/sql/pid                   |      72591750 |
+----------------------------------------+---------------+
```

これらの結果には、THR_LOCK_malloc 相互排他ロックが、その使用される頻度とスレッドがそれを獲得しようとして待機する時間の量の両方に関して、「ホット」であることが示されます。

> **注記**
>
> THR_LOCK_malloc 相互排他ロックはデバッグビルドでのみ使用されます。本番ビルドでは、それが存在しないため、ホットではありません。

インスタンステーブルは、インストゥルメントされたオブジェクトの種類を記述します。インストゥルメントされたオブジェクトは、サーバーによって使われると、イベントを生成します。これらのテーブルは、イベント名と説明のメモまたはステータス情報を提供します。たとえば、file_instances テーブルは、ファイル I/O 操作のインストゥルメントのインスタンスとそれらに関連付けられたファイルを一覧表示します。

```
mysql> SELECT * FROM file_instances\G
*************************** 1. row ***************************
 FILE_NAME: /opt/mysql-log/60500/binlog.000007
EVENT_NAME: wait/io/file/sql/binlog
OPEN_COUNT: 0
*************************** 2. row ***************************
 FILE_NAME: /opt/mysql/60500/data/mysql/tables_priv.MYI
EVENT_NAME: wait/io/file/myisam/kfile
OPEN_COUNT: 1
*************************** 3. row ***************************
 FILE_NAME: /opt/mysql/60500/data/mysql/columns_priv.MYI
EVENT_NAME: wait/io/file/myisam/kfile
OPEN_COUNT: 1
...
```

セットアップテーブルは、モニタリング特性の構成と表示に使われます。たとえば、選択されているイベントタイマーを表示するには、setup_timers テーブルをクエリーします。

```
mysql> SELECT * FROM setup_timers;
+-----------+-------------+
| NAME      | TIMER_NAME  |
+-----------+-------------+
| idle      | MICROSECOND |
| wait      | CYCLE       |
| stage     | NANOSECOND  |
| statement | NANOSECOND  |
+-----------+-------------+
```

setup_instruments は、イベントを収集できる一連のインストゥルメントを一覧表示し、それらのうちどれが有効にされているかを示します。

```
mysql> SELECT * FROM setup_instruments;
+--------------------------------------------------------------+---------+-------+
| NAME                                                         | ENABLED | TIMED |
+--------------------------------------------------------------+---------+-------+
...
| wait/synch/mutex/sql/LOCK_global_read_lock                   | YES     | YES   |
| wait/synch/mutex/sql/LOCK_global_system_variables            | YES     | YES   |
| wait/synch/mutex/sql/LOCK_lock_db                            | YES     | YES   |
```

```
| wait/synch/mutex/sql/LOCK_manager                    | YES     | YES   |
...
| wait/synch/rwlock/sql/LOCK_grant                     | YES     | YES   |
| wait/synch/rwlock/sql/LOGGER::LOCK_logger            | YES     | YES   |
| wait/synch/rwlock/sql/LOCK_sys_init_connect          | YES     | YES   |
| wait/synch/rwlock/sql/LOCK_sys_init_slave            | YES     | YES   |
...
| wait/io/file/sql/binlog                              | YES     | YES   |
| wait/io/file/sql/binlog_index                        | YES     | YES   |
| wait/io/file/sql/casetest                            | YES     | YES   |
| wait/io/file/sql/dbopt                               | YES     | YES   |
...
```

インストゥルメント名の解釈方法を理解するには、セクション22.4「パフォーマンススキーマインストゥルメント命名規則」を参照してください。

インストゥルメントのイベントを収集するかどうかを制御するには、その ENABLED 値を YES または NO に設定します。例:

```
mysql> UPDATE setup_instruments SET ENABLED = 'NO'
    -> WHERE NAME = 'wait/synch/mutex/sql/LOCK_mysql_create_db';
```

パフォーマンススキーマは収集されたイベントを使用して、イベント情報の「コンシューマ」として機能する performance_schema データベース内のテーブルを更新します。setup_consumers テーブルは、使用可能なコンシューマとどれが有効にされているかを示します。

```
mysql> SELECT * FROM setup_consumers;
+--------------------------------+---------+
| NAME                           | ENABLED |
+--------------------------------+---------+
| events_stages_current          | NO      |
| events_stages_history          | NO      |
| events_stages_history_long     | NO      |
| events_statements_current      | YES     |
| events_statements_history      | NO      |
| events_statements_history_long | NO      |
| events_waits_current           | NO      |
| events_waits_history           | NO      |
| events_waits_history_long      | NO      |
| global_instrumentation         | YES     |
| thread_instrumentation         | YES     |
| statements_digest              | YES     |
+--------------------------------+---------+
```

パフォーマンススキーマがコンシューマをイベント情報の宛先として保守するかどうかを制御するには、その ENABLED 値を設定します。

セットアップテーブルについてとそれらを使用して、イベント収集を制御する詳細については、セクション22.2.3.2「パフォーマンススキーマイベントフィルタリング」を参照してください。

先述のグループのいずれにも分類されないその他のテーブルがいくつかあります。たとえば、performance_timers は、使用可能なイベントタイマーとそれらの特性を一覧表示します。タイマーの詳細については、セクション22.2.3.1「パフォーマンススキーマイベントタイミング」を参照してください。

22.2 パフォーマンススキーマ構成

MySQL パフォーマンススキーマを使用するには、これらの構成の考慮事項が適用されます。

- パフォーマンススキーマを使用できるようにするには、構築時に MySQL サーバーにそれを構成する必要があります。パフォーマンススキーマのサポートはバイナリ MySQL 配布に含まれています。ソースから構築する場合は、セクション22.2.1「パフォーマンススキーマビルド構成」に説明するように、それがビルドに構成されていることを確認する必要があります。
- イベント収集が行われるようにするには、サーバーの起動時にパフォーマンススキーマが有効にされている必要があります。サーバーの起動時または実行時に特定のパフォーマンススキーマ機能を有効にして、行われるイベント収集の種類を制御できます。セクション22.2.2「パフォーマンススキーマ起動構成」、セクション22.2.3「パフォーマンススキーマ実行時構成」、およびセクション22.2.3.2「パフォーマンススキーマイベントフィルタリング」を参照してください。

22.2.1 パフォーマンススキーマビルド構成

パフォーマンススキーマを使用できるようにするには、構築時に MySQL サーバーにそれを構成する必要があります。Oracle Corporation によって提供されているバイナリ MySQL 配布は、パフォーマンススキーマをサポート

するように構成されています。別のプロバイダのバイナリ MySQL 配布を使用する場合は、プロバイダに、配布が適切に構成されているかどうかを確認してください。

ソース配布から MySQL を構築する場合は、WITH_PERFSCHEMA_STORAGE_ENGINE オプションを有効にして、CMake を実行し、パフォーマンススキーマを有効にします。

```
shell> cmake . -DWITH_PERFSCHEMA_STORAGE_ENGINE=1
```

-DWITHOUT_PERFSCHEMA_STORAGE_ENGINE=1 オプションを使用して MySQL を構成すると、パフォーマンススキーマが含まれないため、それを含める場合は、このオプションを使用しないでください。セクション 2.9.4「MySQL ソース構成オプション」を参照してください。

パフォーマンススキーマなし (または現在のすべてのテーブルを含まない可能性がある古いバージョンのパフォーマンススキーマ) で構成された以前のインストールに MySQL をインストールする場合は、サーバーの起動後に mysql_upgrade を実行して、すべての現在のテーブルを使用して performance_schema データベースが存在するようにしてください。次に、サーバーを再起動します。これを行う必要があることを示す 1 つの兆候は、エラーログ内に次のようなメッセージが存在することです。

```
[ERROR] Native table 'performance_schema'.'events_waits_history'
has the wrong structure
[ERROR] Native table 'performance_schema'.'events_waits_history_long'
has the wrong structure
...
```

サーバーにパフォーマンススキーマのサポートが組み込まれているかどうかを確認するには、そのヘルプ出力をチェックします。パフォーマンススキーマを使用できる場合、出力に、performance_schema で始まる名前の付いたいくつかの変数が示されます。

```
shell> mysqld --verbose --help
...
  --performance_schema
                    Enable the performance schema.
  --performance_schema_events_waits_history_long_size=#
                    Number of rows in events_waits_history_long.
...
```

サーバーに接続し、SHOW ENGINES からの出力で PERFORMANCE_SCHEMA ストレージエンジンの名前が挙げられた行を探すこともできます。

```
mysql> SHOW ENGINES\G
...
      Engine: PERFORMANCE_SCHEMA
     Support: YES
     Comment: Performance Schema
Transactions: NO
          XA: NO
  Savepoints: NO
...
```

構築時にサーバーにパフォーマンススキーマが構成されていない場合、SHOW ENGINES からの出力に、PERFORMANCE_SCHEMA の行が表示されません。SHOW DATABASES からの出力に、performance_schema が示されていることもありますが、それにはテーブルがなく、それを使用することはできません。

SHOW ENGINES 出力の PERFORMANCE_SCHEMA の行は、パフォーマンススキーマを使用できることを意味し、それが有効にされていることを意味しているわけではありません。それを有効にするには、次のセクションで説明するように、サーバーの起動時にそうする必要があります。

22.2.2 パフォーマンススキーマ起動構成

パフォーマンススキーマを使用できるものとして、MySQL 5.6.6 以降、それはデフォルトで有効にされます。5.6.6 より前では、それはデフォルトで無効にされます。それを明示的に有効または無効にするには、performance_schema 変数を適切な値に設定して、サーバーを起動します。たとえば、my.cnf ファイルでこれらの行を使用します。

```
[mysqld]
performance_schema=on
```

パフォーマンススキーマの初期化時に、サーバーが内部バッファーを割り当てることができない場合、パフォーマンススキーマは自動的に無効になり、performance_schema を OFF に設定して、サーバーがインストゥルメンテーションなしで実行します。

MySQL 5.6.4 以降、パフォーマンススキーマは、サーバー起動時のインストゥルメントおよびコンシューマの構成を許可します。これは、以前 setup_instruments および setup_consumers テーブルに UPDATE ステートメントを使用して、実行時にのみ可能でした。この変更は、サーバーの起動時にすでに初期化されているインストゥルメントを、実行時の構成で無効にするには遅すぎるために行われました。たとえば、wait/synch/mutex/sql/LOCK_open 相互排他ロックはサーバー起動時に 1 回初期化されるため、実行時に対応するインストゥルメントを無効にする試みは無効です。

サーバー起動時のインストゥルメントを制御するには、この形式のオプションを使用します。

```
--performance-schema-instrument='instrument_name=value'
```

ここで instrument_name は wait/synch/mutex/sql/LOCK_open などのインストゥルメント名で、value はこれらのいずれかの値です。

- off、false、または 0: インストゥルメントを無効にします
- on、true、または 1: インストゥルメントを有効にして時間を測定します
- counted: インストゥルメントを有効にしてカウント (時間測定ではなく) します

各 --performance-schema-instrument オプションではインストゥルメント名を 1 つしか指定できませんが、オプションの複数のインスタンスを指定して、複数のインストゥルメントを構成できます。さらに、インストゥルメント名にパターンを使用でき、パターンに一致するインストゥルメントを構成します。すべての条件同期インストゥルメントを有効で、カウント対象として構成するには、次のオプションを使用します。

```
--performance-schema-instrument='wait/synch/cond/%=counted'
```

すべてのインストゥルメントを無効にするには、次のオプションを使用します。

```
--performance-schema-instrument='%=off'
```

長いインストゥルメント名文字列は、順序に関係なく、短いパターン名より優先されます。インストゥルメントを選択するためのパターンの指定については、セクション22.2.3.4「フィルタリング操作のインストゥルメントまたはコンシューマの指定」を参照してください。

認識されないインストゥルメント名は無視されます。あとでインストールされたプラグインによってインストゥルメントを作成することは可能で、そのときに名前が認識され、構成されます。

サーバー起動時のコンシューマを制御するには、この形式のオプションを使用します。

```
--performance-schema-consumer-consumer_name=value
```

ここで、consumer_name は events_waits_history などのコンシューマ名で、value はこれらのいずれかです。

- off、false、または 0: コンシューマのイベントを収集しません
- on、true、または 1: コンシューマのイベントを収集します

たとえば、events_waits_history コンシューマを有効にするには、次のオプションを使用します。

```
--performance-schema-consumer-events-waits-history=on
```

許可されるコンシューマ名は、setup_consumers テーブルを調べるとわかります。パターンは許可されません。setup_consumers テーブル内のコンシューマ名は下線が使われますが、起動時に設定されたコンシューマでは、名前の中のダッシュと下線は同等です。

パフォーマンススキーマには、構成情報を提供するいくつかのシステム変数が含まれます。

```
mysql> SHOW VARIABLES LIKE 'perf%';
+----------------------------------------------------+---------+
| Variable_name                                      | Value   |
+----------------------------------------------------+---------+
| performance_schema                                 | ON      |
| performance_schema_accounts_size                   | 100     |
| performance_schema_digests_size                    | 200     |
| performance_schema_events_stages_history_long_size | 10000   |
| performance_schema_events_stages_history_size      | 10      |
| performance_schema_events_statements_history_long_size | 10000   |
| performance_schema_events_statements_history_size  | 10      |
| performance_schema_events_waits_history_long_size  | 10000   |
```

```
| performance_schema_events_waits_history_size          | 10     |
| performance_schema_hosts_size                         | 100    |
| performance_schema_max_cond_classes                   | 80     |
| performance_schema_max_cond_instances                 | 1000   |
...
```

performance_schema 変数は ON または OFF で、パフォーマンススキーマが有効か無効かを示します。ほかの変数はテーブルサイズ (行数) やメモリー割り当て値を示します。

> **注記**
>
> パフォーマンススキーマが有効にされている場合、パフォーマンススキーマインスタンスの数は、おそらく大きくサーバーメモリーフットプリントに影響します。パフォーマンススキーマのシステム変数の値をチューニングして、インストゥルメントの不足と過剰なメモリー消費のバランスをとるインスタンスの数を見つける必要がある可能性があります。

パフォーマンススキーマシステム変数の値を変更するには、それらをサーバー起動時に設定します。たとえば、my.cnf ファイルに、履歴テーブルのサイズを変更するために次の行を入れます。

```
[mysqld]
performance_schema
performance_schema_events_waits_history_size=20
performance_schema_events_waits_history_long_size=15000
```

MySQL 5.6.6 以降、パフォーマンススキーマは、そのパラメータのいくつかの値を、それらが明示的に設定されていない場合に、サーバーの起動時に自動的にサイズ設定します。たとえば、それはイベント待機テーブルのサイズを制御するパラメータをこのようにサイズ設定します。どのパラメータがこのポリシーでサイズ設定されるかを確認するには、mysqld --verbose --help を使用し、-1 のデフォルト値を持つものを探すか、セクション 22.12「パフォーマンススキーマシステム変数」を参照してください。

サーバー起動時に設定されていない (または -1 に設定されている) 自動サイズ設定される各パラメータについて、パフォーマンススキーマは、次のシステム変数の値に基づいてその値を設定する方法を判断します。それらは、MySQL サーバーをどのように構成しているかに関する「ヒント」とみなされます。

```
max_connections
open_files_limit
table_definition_cache
table_open_cache
```

特定のパラメータの自動サイズ設定をオーバーライドするには、起動時にそれに -1 以外の値を設定します。この場合、パフォーマンススキーマはそれに指定された値を割り当てます。

実行時に、SHOW VARIABLES では、自動サイズ設定されたパラメータに設定されていた実際の値が表示されます。

パフォーマンススキーマが無効にされている場合、その自動サイズ設定されたパラメータは -1 に設定されたままになり、SHOW VARIABLES では -1 が表示されます。

22.2.3 パフォーマンススキーマ実行時構成

パフォーマンススキーマセットアップテーブルには、モニタリング構成に関する情報が含まれます。

```
mysql> SELECT TABLE_NAME FROM INFORMATION_SCHEMA.TABLES
    -> WHERE TABLE_SCHEMA = 'performance_schema'
    -> AND TABLE_NAME LIKE 'setup%';
+-------------------+
| TABLE_NAME        |
+-------------------+
| setup_actors      |
| setup_consumers   |
| setup_instruments |
| setup_objects     |
| setup_timers      |
+-------------------+
```

これらのテーブルの内容を調査して、パフォーマンススキーマのモニタリング特性に関する情報を取得できます。UPDATE 権限を持っている場合は、セットアップテーブルを変更して、モニタリングが行われる方法に影響するパフォーマンススキーマ操作を変更できます。これらのテーブルの追加の詳細については、セクション 22.9.2「パフォーマンススキーマセットアップテーブル」を参照してください。

選択されているイベントタイマーを表示するには、setup_timers テーブルをクエリーします。

```
mysql> SELECT * FROM setup_timers;
+-----------+-------------+
| NAME      | TIMER_NAME  |
+-----------+-------------+
| idle      | MICROSECOND |
| wait      | CYCLE       |
| stage     | NANOSECOND  |
| statement | NANOSECOND  |
+-----------+-------------+
```

NAME 値はタイマーが適用されるインストゥルメントの種類を示し、TIMER_NAME はそれらのインストゥルメントに適用されるタイマーを示します。タイマーは、名前が NAME 値に一致するコンポーネントから始まるインストゥルメントに適用されます。

タイマーを変更するには、NAME 値を更新します。たとえば、wait タイマーに NANOSECOND タイマーを使用するには:

```
mysql> UPDATE setup_timers SET TIMER_NAME = 'NANOSECOND'
    -> WHERE NAME = 'wait';
mysql> SELECT * FROM setup_timers;
+-----------+-------------+
| NAME      | TIMER_NAME  |
+-----------+-------------+
| idle      | MICROSECOND |
| wait      | NANOSECOND  |
| stage     | NANOSECOND  |
| statement | NANOSECOND  |
+-----------+-------------+
```

タイマーの説明については、セクション22.2.3.1「パフォーマンススキーマイベントタイミング」を参照してください。

setup_instruments および setup_consumers テーブルは、イベントを収集できるインストゥルメントと、イベント情報が実際に収集されるコンシューマの種類をそれぞれ一覧表示します。その他のセットアップテーブルにより、モニタリング構成をさらに変更できます。セクション22.2.3.2「パフォーマンススキーマイベントフィルタリング」では、イベントコレクションに影響するように、これらのテーブルをどのように変更できるかについて説明しています。

実行時に SQL ステートメントを使用して行う必要があるパフォーマンススキーマ構成の変更があり、これらの変更をサーバーが起動するたびに有効になるようにする場合、ファイルにステートメントを入れ、--init-file=file_name オプションを使用してサーバーを起動します。この戦略は、簡易サーバーヘルスモニタリング、インシデント調査、アプリケーション動作のトラブルシューティングなど、さまざまな種類のモニタリングを生成するようにそれぞれカスタマイズされている複数のモニタリング構成がある場合にも役に立つ可能性があります。各モニタリング構成のステートメントをそれらの固有のファイルに入れ、サーバーの起動時に --init-file 引数として、該当するファイルを指定します。

22.2.3.1 パフォーマンススキーマイベントタイミング

イベントは、サーバーソースコードに追加されたインストゥルメンテーションを使用して収集されます。インストゥルメントはイベントの時間を測定しますが、これはパフォーマンススキーマがイベントにどれくらいの時間がかかるかを知らせる方法です。タイミング情報を収集しないようにインストゥルメントを構成することもできます。このセクションでは、使用可能なタイマーとそれらの特性、およびイベント内のタイミング値を表す方法について説明します。

2 つのテーブルはタイマー情報を提供します。

- performance_timers は、使用可能なタイマーとそれらの特性を一覧表示します。
- setup_timers は、どのタイマーがどのインストゥルメントに使用されるかを示します。

setup_timers 内の各タイマー行は、performance_timers に示されているいずれかのタイマーを表している必要があります。

タイマーの精度とそれらに伴うオーバーヘッドの量はさまざまに異なります。使用可能なタイマーとそれらの特性を確認するには、performance_timers テーブルをチェックしてください。

```
mysql> SELECT * FROM performance_timers;
+-------------+-----------------+------------------+----------------+
```

```
| TIMER_NAME  | TIMER_FREQUENCY | TIMER_RESOLUTION | TIMER_OVERHEAD |
+-------------+-----------------+------------------+----------------+
| CYCLE       |      2389029850 |                1 |             72 |
| NANOSECOND  |            NULL |             NULL |           NULL |
| MICROSECOND |         1000000 |                1 |            585 |
| MILLISECOND |            1035 |                1 |            738 |
| TICK        |             101 |                1 |            630 |
+-------------+-----------------+------------------+----------------+
```

TIMER_NAME カラムには使用可能なタイマーの名前が表示されます。CYCLE は CPU (プロセッサ) サイクルカウンタに基づいたタイマーを表します。特定のタイマーに関連付けられた値が NULL の場合、そのタイマーはプラットフォームでサポートされていません。NULLのない行は、setup_timers で使用できるタイマーを示しています。

TIMER_FREQUENCY は、1 秒あたりのタイマー単位数を示します。サイクルタイマーの場合、頻度は一般に CPU 速度に関連します。示された値は、2.4GHz プロセッサを搭載するシステムで取得されました。ほかのタイマーは固定の数秒に基づきます。TICK の場合、頻度はプラットフォームごとに異なることがあります (たとえば、100 ティック/秒を使用するものや、1000 ティック/秒を使用するものがあります)。

TIMER_RESOLUTION は、タイマー値が一度に増加するタイマー単位数を示します。タイマーの分解能が 10 の場合、その値は毎回 10 ずつ増加します。

TIMER_OVERHEAD は特定のタイマーで 1 つのタイミングを取得するためのオーバーヘッドの最小サイクル数です。タイマーはイベントの開始と終了で呼び出されるため、イベントあたりのオーバーヘッドは表示される値の 2 倍になります。

有効なタイマーを確認し、タイマーを変更するには、setup_timers テーブルにアクセスします。

```
mysql> SELECT * FROM setup_timers;
+-----------+-------------+
| NAME      | TIMER_NAME  |
+-----------+-------------+
| idle      | MICROSECOND |
| wait      | NANOSECOND  |
| stage     | NANOSECOND  |
| statement | NANOSECOND  |
+-----------+-------------+

mysql> UPDATE setup_timers SET TIMER_NAME = 'MICROSECOND'
    -> WHERE NAME = 'wait';
mysql> SELECT * FROM setup_timers;
+-----------+-------------+
| NAME      | TIMER_NAME  |
+-----------+-------------+
| idle      | MICROSECOND |
| wait      | MICROSECOND |
| stage     | NANOSECOND  |
| statement | NANOSECOND  |
+-----------+-------------+
```

デフォルトで、パフォーマンススキーマは各インストゥルメントの種類で使用可能な最高のタイマーを使用しますが、ユーザーが別のものを選択することもできます。

待機の時間を測定する場合、もっとも重要な基準は、タイマーの精度を犠牲にする可能性があっても、オーバーヘッドを削減することであるため、CYCLE タイマーを使用することがもっとも適切です。

ステートメント (またはステージ) の実行にかかる時間は、一般に単一の待機の実行にかかる時間より桁違いに大きくなります。ステートメントの時間を測定するために、もっとも重要な基準は、プロセッサの周波数の変更に影響を受けない正確な測定基準を設定することであるため、サイクルに基づいていないタイマーを使用することがもっとも適切です。ステートメントのデフォルトのタイマーは NANOSECOND です。タイマーを 2 回呼び出す (ステートメントの起動時に 1 回、その終了時に 1 回) ことによって発生するオーバーヘッドは、ステートメント自体の実行に使用される CPU 時間と比較して桁違いに小さいため、サイクルタイマーと比較した追加の「オーバーヘッド」は大きくありません。CYCLE タイマーを使用することにはメリットがなく、欠点だけです。

サイクルカウンタによって提供される精度はプロセッサ速度によって異なります。プロセッサが 1 GHz (10 億サイクル/秒) 以上で実行する場合、サイクルカウンタはナノ秒未満の精度を実現します。サイクルカウンタを使用することは、実際の時間を取得するより、はるかに負荷が小さくなります。たとえば、標準 gettimeofday() 関数は数百サイクルかかる可能性があり、これは 1 秒あたり数千または数百万回発生する可能性のあるデータ収集では、許容できないオーバーヘッドです。

サイクルカウンタには欠点もあります。

- エンドユーザーは、何分の 1 秒などの時計の単位でタイミングを知ることを期待します。サイクルから秒に変換することは大きな負荷がかかる可能性があります。このため、変換はすばやいかなり概算的な乗算演算です。
- ラップトップが節電モードになったときや熱の生成を抑えるために、CPU の速度が低下したときなどに、プロセッササイクルレートが変わることがあります。プロセッサのサイクルレートが変動した場合、サイクルから実時間単位への変換でエラーが発生することがあります。
- サイクルカウンタは、プロセッサやオペレーティングシステムによって、信頼できない場合や使用できない場合があります。たとえば、Pentium では、命令は RDTSC (C 命令ではなくアセンブリ言語) であり、理論上オペレーティングシステムはユーザーモードプログラムのその使用を妨げることができます。
- 異常実行またはマルチプロセッサ同期に関する一部のプロセッサの詳細により、カウンタが最大 1000 サイクルごとに高速になったり、低速になったりするように見えることがあります。

現在、MySQL は、x386 (Windows、OS X、Linux、Solaris、およびその他の Unix フレーバー)、PowerPC、および IA-64 のサイクルカウンタと連携します。

setup_instruments テーブルにはイベントを収集するインストゥルメントを示す ENABLED カラムがあります。このテーブルには、時間が測定されるインストゥルメントを示す TIMED カラムもあります。インストゥルメントが有効にされていない場合、イベントを生成しません。有効にされているインストゥルメントの時間が測定されない場合、インストゥルメントによって生成されたイベントの TIMER_START、TIMER_END、および TIMER_WAIT タイマー値が NULL になります。これによって、サマリーテーブルの合計、最小、最大、および平均の時間値の計算時に、それらの値が無視されます。

イベント内で、イベントのタイミングの開始時に、有効なタイマーによって指定された単位で時間が保存されます。表示には、選択されたタイマーに関係なく、時間は標準単位に正規化するために、ピコ秒 (1 兆分の 1 秒) で示されます。

setup_timers テーブルへの変更はただちにモニタリングに影響します。すでに進行中のイベントは、開始時間に元のタイマーを使用し、終了時間に新しいタイマーを使用する可能性があるため、予測不可能な結果に至る場合があります。タイマーを変更した場合、TRUNCATE TABLE を使用して、パフォーマンススキーマの統計をリセットする必要がある場合があります。

サーバー起動時のパフォーマンススキーマの初期化で、タイマーベースライン (「時間ゼロ」) が発生します。イベント内の TIMER_START および TIMER_END 値はベースライン以降のピコ秒を表します。TIMER_WAIT 値はピコ秒での期間です。

イベント内のピコ秒値は概算です。それらの精度は、ある単位から別の単位への変換に伴う通常の誤差の形式に左右されます。CYCLE タイマーが使われ、プロセッサのレートがさまざまに異なる場合、ドリフトが発生することがあります。このため、サーバーの起動から経過した時間の正確な測定基準として、イベントの TIMER_START 値を見ることは妥当ではありません。一方、開始時間や期間によって、イベントを順序付けるために、ORDER BY 句に TIMER_START 値または TIMER_WAIT 値を使用することは適切です。

イベントでマイクロ秒などの値ではなく、ピコ秒を選択したことには、パフォーマンス上の根拠があります。1 つの実装目標は、タイマーに関係なく、統一された時間単位で結果を表示することでした。理想の世界では、この時間単位は時計の単位のように見え、適度に正確である、つまりマイクロ秒です。ただし、サイクルまたはナノ秒をマイクロ秒に変換するには、すべてのインストゥルメンテーションで除算を実行する必要がある場合があります。除算は多くのプラットフォームで高い負荷がかかります。乗算は負荷が高くないため、それを使用しています。そのため、時間単位は、大きな精度の損失がないように十分に大きな乗数を使用した、可能なかぎり最大の TIMER_FREQUENCY 値の整数の倍数です。その結果、時間単位が「ピコ秒」になります。この精度は疑似ですが、この決定によりオーバーヘッドを最小にすることができます。

22.2.3.2 パフォーマンススキーマイベントフィルタリング

イベントはプロデューサ/コンシューマ方式で処理されます。

- インストゥルメントされたコードは、イベントのソースで、収集されるイベントを生成します。setup_instruments テーブルは、イベントを収集できるインストゥルメント、それらが有効にされているかどうか、および (有効にされているインストゥルメントの場合) タイミング情報を収集するかどうかを一覧表示します。

```
mysql> SELECT * FROM setup_instruments;
+---------------------------------------------------------------+---------+-------+
| NAME                                                          | ENABLED | TIMED |
+---------------------------------------------------------------+---------+-------+
...
```

```
| wait/synch/mutex/sql/LOCK_global_read_lock              | YES     | YES   |
| wait/synch/mutex/sql/LOCK_global_system_variables       | YES     | YES   |
| wait/synch/mutex/sql/LOCK_lock_db                       | YES     | YES   |
| wait/synch/mutex/sql/LOCK_manager                       | YES     | YES   |
...
```

setup_instruments テーブルはイベント生成のもっとも基本的な制御の形式を提供します。モニターされるオブジェクトやスレッドの種類に基づいて、イベント生成をさらに絞り込むには、セクション22.2.3.3「イベントの事前フィルタリング」に説明するように、ほかのテーブルを使用できます。

- パフォーマンススキーマテーブルは、イベントの宛先で、イベントを消費します。setup_consumers テーブルは、イベント情報を送信できるコンシューマの種類とそれらが有効にされているかどうかを一覧表示します。

```
mysql> SELECT * FROM setup_consumers;
+--------------------------------+---------+
| NAME                           | ENABLED |
+--------------------------------+---------+
| events_stages_current          | NO      |
| events_stages_history          | NO      |
| events_stages_history_long     | NO      |
| events_statements_current      | YES     |
| events_statements_history      | NO      |
| events_statements_history_long | NO      |
| events_waits_current           | NO      |
| events_waits_history           | NO      |
| events_waits_history_long      | NO      |
| global_instrumentation         | YES     |
| thread_instrumentation         | YES     |
| statements_digest              | YES     |
+--------------------------------+---------+
```

フィルタリングはパフォーマンスモニタリングのさまざまなステージで実行できます。

- 事前フィルタリング。これは、プロデューサから特定の種類のイベントのみが収集され、収集されたイベントが特定のコンシューマのみを更新するように、パフォーマンススキーマ構成を変更することによって行われます。これを実行するには、インストゥルメントまたはコンシューマを有効または無効にします。事前フィルタリングは、パフォーマンススキーマによって行われ、すべてのユーザーに適用されるグローバルな効果を持ちます。

  事前フィルタリングを使用する理由:

  - オーバーヘッドを削減するため。パフォーマンススキーマのオーバーヘッドは、すべてのインストゥルメントを有効にしていても最小であるはずですが、さらに縮小したいと考える可能性があります。または、タイミングイベントに関心がなく、タイミングオーバーヘッドを解消するために、タイミングコードを無効にしたいと考えます。

  - 関心のないイベントの現在のイベントまたは履歴テーブルへの入力を妨げるため。事前フィルタリングにより、これらのテーブル内に、有効なインストゥルメントの種類の行のインスタンス用に多くの「空き」を残します。事前フィルタリングでファイルインストゥルメントのみを有効にしている場合、非ファイルインストゥルメントの行は収集されません。事後フィルタリングで、非ファイルイベントが収集され、ファイルイベントの少ない行が残されます。

  - 特定の種類のイベントテーブルの保守を回避するため。コンシューマを無効にすると、サーバーはそのコンシューマの宛先の保守に時間を費やさなくなります。たとえば、イベント履歴に関心がない場合、履歴テーブルコンシューマを無効にして、パフォーマンスを向上できます。

- 事後フィルタリング。これには、クエリー内で、パフォーマンススキーマテーブルから情報を選択する WHERE 句を使用して、使用可能なイベントのうち表示したいものを指定することが含まれます。事後フィルタリングは、各ユーザーが使用可能なイベントのうち関心のあるものを選択するため、ユーザー単位で実行されます。

  事後フィルタリングを使用する理由:

  - 関心のあるイベント情報に関して、個々のユーザーの決断を避けるため。

  - 事前フィルタリングの使用に課せられる制限が前もってわからない場合に、パフォーマンススキーマを使用して、パフォーマンスの問題を調査するため。

次のセクションでは、事前フィルタリングの詳細を説明し、フィルタリング操作でインストゥルメントやコンシューマを指定するためのガイドラインを提供します。情報を取得するためのクエリーの書き方 (事後フィルタリング) については、セクション22.3「パフォーマンススキーマクエリー」を参照してください。

22.2.3.3 イベントの事前フィルタリング

事前フィルタリングは、パフォーマンススキーマによって行われ、すべてのユーザーに適用されるグローバルな効果を持ちます。事前フィルタリングは、イベント処理のプロデューサまたはコンシューマステージに適用できます。

- プロデューサステージで事前フィルタリングを構成するには、いくつかのテーブルを使用できます。
  - setup_instruments は使用可能なインストゥルメントを示します。このテーブルで無効にされているインストゥルメントは、ほかの生成関連セットアップテーブルの内容に関係なく、イベントを生成しません。このテーブルで有効にされているインストゥルメントは、イベントの生成が許可され、ほかのテーブルの内容に依存します。
  - setup_objects は、パフォーマンススキーマが特定のテーブルオブジェクトをモニターするかどうかを制御します。
  - threads スレッドは各サーバースレッドでモニタリングが有効にされているかどうかを示します。
  - setup_actors は、新しいフォアグラウンドスレッドの初期モニタリング状態を決定します。
- コンシューマステージで事前フィルタリングを構成するには、setup_consumers テーブルを変更します。これによって、イベントの送信先が決まります。setup_consumers はイベント生成にも暗黙的に影響します。特定のイベントがどの宛先にも送信されない (つまり消費されない) 場合、パフォーマンススキーマはそれを生成しません。

これらのいずれかのテーブルの変更は、setup_actors を除いて、ただちにモニタリングに影響します。setup_actors の変更は、変更後に作成されるフォアグラウンドスレッドにのみ影響します。

モニタリング構成を変更すると、パフォーマンススキーマは履歴テーブルをフラッシュしません。すでに収集されたイベントは、新しいイベントによって置き換えられるまで、現在のイベントと履歴テーブルに残ります。インストゥルメントを無効にする場合、それらのイベントが関心のある新しいイベントによって置き換えられるまで、しばらく待つ必要がある場合があります。または、TRUNCATE TABLE を使用して、履歴テーブルを空にします。

インストゥルメンテーションを変更したら、サマリーテーブルを切り捨てて、以前に収集されたイベントの集計情報をクリアする必要がある場合があります。events_statements_summary_by_digest を除き、サマリーテーブルの TRUNCATE TABLE の効果は、サマリーカラムを 0 または NULL にリセットすることで、行を削除することではありません。

次のセクションでは、特定のテーブルを使用して、パフォーマンススキーマの事前フィルタリングを制御する方法について説明します。

インストゥルメントによる事前フィルタリング

setup_instruments テーブルは使用可能なインストゥルメントを一覧表示します。

```
mysql> SELECT * FROM setup_instruments;
+---------------------------------------------------------+---------+-------+
| NAME                                                    | ENABLED | TIMED |
+---------------------------------------------------------+---------+-------+
...
| wait/synch/mutex/sql/LOCK_global_read_lock              | YES     | YES   |
| wait/synch/mutex/sql/LOCK_global_system_variables       | YES     | YES   |
| wait/synch/mutex/sql/LOCK_lock_db                       | YES     | YES   |
| wait/synch/mutex/sql/LOCK_manager                       | YES     | YES   |
...
| wait/synch/rwlock/sql/LOCK_grant                        | YES     | YES   |
| wait/synch/rwlock/sql/LOGGER::LOCK_logger               | YES     | YES   |
| wait/synch/rwlock/sql/LOCK_sys_init_connect             | YES     | YES   |
| wait/synch/rwlock/sql/LOCK_sys_init_slave               | YES     | YES   |
...
| wait/io/file/sql/binlog                                 | YES     | YES   |
| wait/io/file/sql/binlog_index                           | YES     | YES   |
| wait/io/file/sql/casetest                               | YES     | YES   |
| wait/io/file/sql/dbopt                                  | YES     | YES   |
...
```

インストゥルメントを有効にするかどうかを制御するには、その ENABLED カラムを YES または NO に設定します。有効にされたインストゥルメントのタイミング情報を収集するかどうかを構成するには、その TIMED 値を YES または NO に設定します。TIMED カラムを設定すると、セクション22.2.3.1「パフォーマンススキーマイベントタイミング」に説明するように、パフォーマンススキーマテーブルの内容に影響します。

setup_instruments テーブルへの変更はただちにモニタリングに影響します。

setup_instruments テーブルはイベント生成のもっとも基本的な制御の形式を提供します。モニターされるオブジェクトやスレッドの種類に基づいて、イベント生成をさらに絞り込むには、セクション22.2.3.3「イベントの事前フィルタリング」に説明するように、ほかのテーブルを使用できます。

次の例に、setup_instruments テーブルへの可能な操作を示します。ほかの事前フィルタリング操作と同様に、これらの変更はすべてのユーザーに影響します。これらの一部のクエリーでは、LIKE 演算子とパターンマッチインストゥルメント名を使用しています。インストゥルメントを選択するためのパターンの指定に関する追加情報については、セクション22.2.3.4「フィルタリング操作のインストゥルメントまたはコンシューマの指定」を参照してください。

- すべてのインストゥルメントを無効にします。

```
mysql> UPDATE setup_instruments SET ENABLED = 'NO';
```

  これでイベントが収集されなくなります。

- すべてのインストゥルメントを無効にし、それらを現在の無効にされているインストゥルメントのセットに追加します。

```
mysql> UPDATE setup_instruments SET ENABLED = 'NO'
    -> WHERE NAME LIKE 'wait/io/file/%';
```

- ファイルインストゥルメントのみを無効にし、ほかのすべてのインストゥルメントを有効にします。

```
mysql> UPDATE setup_instruments
    -> SET ENABLED = IF(NAME LIKE 'wait/io/file/%', 'NO', 'YES');
```

- mysys ライブラリ内のインストゥルメントを除くすべてのインストゥルメントを有効にします。

```
mysql> UPDATE setup_instruments
    -> SET ENABLED = CASE WHEN NAME LIKE '%/mysys/%' THEN 'YES' ELSE 'NO' END;
```

- 特定のインストゥルメントを無効にします。

```
mysql> UPDATE setup_instruments SET ENABLED = 'NO'
    -> WHERE NAME = 'wait/synch/mutex/mysys/TMPDIR_mutex';
```

- インストゥルメントの状態を切り替えるには、その ENABLED 値を「反転」します。

```
mysql> UPDATE setup_instruments
    -> SET ENABLED = IF(ENABLED = 'YES', 'NO', 'YES')
    -> WHERE NAME = 'wait/synch/mutex/mysys/TMPDIR_mutex';
```

- すべてのイベントのタイミングを無効にします。

```
mysql> UPDATE setup_instruments SET TIMED = 'NO';
```

オブジェクトによる事前フィルタリング

setup_objects テーブルは、パフォーマンススキーマが特定のテーブルオブジェクトをモニターするかどうかを制御します。初期 setup_objects の内容は次のように見えます。

```
mysql> SELECT * FROM setup_objects;
+-------------+--------------------+-------------+---------+-------+
| OBJECT_TYPE | OBJECT_SCHEMA      | OBJECT_NAME | ENABLED | TIMED |
+-------------+--------------------+-------------+---------+-------+
| TABLE       | mysql              | %           | NO      | NO    |
| TABLE       | performance_schema | %           | NO      | NO    |
| TABLE       | information_schema | %           | NO      | NO    |
| TABLE       | %                  | %           | YES     | YES   |
+-------------+--------------------+-------------+---------+-------+
```

setup_objects テーブルへの変更はただちにオブジェクトモニタリングに影響します。

OBJECT_TYPE カラムは行が適用されるオブジェクトの種類を示します。TABLE フィルタリングはテーブル I/O イベント (wait/io/table/sql/handler インストゥルメント) およびテーブルロックイベント (wait/lock/table/sql/handler インストゥルメント) に影響します。

OBJECT_SCHEMA および OBJECT_NAME カラムにはリテラルのスキーマまたはテーブル名、または任意の名前に一致する '%' が格納されているべきです。

ENABLED カラムは一致するオブジェクトがモニターされているかどうかを示し、TIMED はタイミング情報を収集するかどうかを示します。

デフォルトのオブジェクト構成の効果は、mysql、INFORMATION_SCHEMA、および performance_schema データベースのテーブルを除くすべてのテーブルをインストゥルメントすることです。(INFORMATION_SCHEMA データベース内のテーブルは、setup_objects の内容に関係なくインストゥルメントされず、information_schema.% の行は単にこのデフォルトを明示します。)

パフォーマンススキーマは、setup_objects の一致をチェックする場合、まずより詳細な一致を見つけようとします。たとえば、テーブル db1.t1 では、'db1' と 't1'、次に 'db1' と '%'、次に '%' と '%' の一致を検索します。さまざまな一致する setup_objects 行はさまざまな ENABLED 値と TIMED 値を持つ可能性があるため、一致が発生する順序が重要です。

テーブル関連イベントの場合、パフォーマンススキーマは setup_objects の内容と setup_instruments を組み合わせて、インストゥルメントを有効にするかどうか、および有効にされているインストゥルメントの時間を測定するかどうかを判断します。

- setup_objects 内の行に一致するテーブルでは、テーブルインストゥルメントは、setup_instruments と setup_objects の両方で、ENABLED が YES である場合にのみイベントを生成します。
- 両方の値が YES の場合にのみ、タイミング情報が収集されるように、2 つのテーブル内の TIMED 値が組み合わされます。

setup_objects には、db1、db2、および db3 に適用される次の行が含まれるとします。

```
+-------------+---------------+-------------+---------+-------+
| OBJECT_TYPE | OBJECT_SCHEMA | OBJECT_NAME | ENABLED | TIMED |
+-------------+---------------+-------------+---------+-------+
| TABLE       | db1           | t1          | YES     | YES   |
| TABLE       | db1           | t2          | NO      | NO    |
| TABLE       | db2           | %           | YES     | YES   |
| TABLE       | db3           | %           | NO      | NO    |
| TABLE       | %             | %           | YES     | YES   |
+-------------+---------------+-------------+---------+-------+
```

setup_instruments のテーブル関連インストゥルメントに NO の ENABLED 値がある場合、オブジェクトのイベントはモニターされません。ENABLED 値が YES の場合、イベントモニタリングは、関連 setup_objects 行内の ENABLED 値に従って行われます。

- db1.t1 イベントはモニターされます
- db1.t2 イベントはモニターされません
- db2.t3 イベントはモニターされます
- db3.t4 イベントはモニターされません
- db4.t5 イベントはモニターされます

setup_instruments および setup_objects テーブルの TIMED カラムを組み合わせて、イベントタイミング情報を収集するかどうかを判断する場合も同様の口ジックが当てはまります。

永続的テーブルと一時テーブルが同じ名前を持つ場合、setup_objects 行に対する照合が両方に対して同様に行われます。一方のテーブルのモニタリングを有効にして、他方を有効にしないことはできません。ただし、各テーブルは個別にインストゥルメントされます。

ENABLED カラムは MySQL 5.6.3 で追加されました。ENABLED カラムがない初期バージョンでは、setup_objects はテーブル内の一部の行に一致するオブジェクトのモニタリングを有効にするためにのみ使用されます。テーブルによるインストゥルメンテーションを明示的に無効にする方法はありません。

スレッドによる事前フィルタリング

threads テーブルは各サーバースレッドの行を格納します。各行は、スレッドに関する情報を格納し、それに対するモニタリングが有効にされているかどうかを示します。スレッドをモニターするパフォーマンススキーマの場合、これらのことが当てはまる必要があります。

- setup_consumers テーブル内の thread_instrumentation コンシューマは YES である必要があります。
- thread.INSTRUMENTED カラムは YES である必要があります。

- setup_instruments テーブル内で有効にされているインストゥルメントから生成されたスレッドイベントに対してのみ、モニタリングが行われます。

threads テーブルの INSTRUMENTED カラムは各スレッドのモニタリング状態を示します。フォアグラウンドスレッド (クライアント接続の結果) では、初期 INSTRUMENTED 値は、スレッドに関連付けられているユーザーアカウントが setup_actors テーブル内のいずれかの行に一致するかどうかによって決定されます。バックグラウンドスレッドの場合、INSTRUMENTED はデフォルトで YES で、バックグラウンドスレッドに関連付けられたユーザーはないため、setup_actors は参照されません。どのスレッドでも、スレッドの有効期間の間にその INSTRUMENTED 値が変更されることがあります。

初期 setup_actors の内容は次のように見えます。

```
mysql> SELECT * FROM setup_actors;
+------+------+------+
| HOST | USER | ROLE |
+------+------+------+
| %    | %    | %    |
+------+------+------+
```

パフォーマンススキーマは HOST および USER カラムを使用し、新しい各フォアグラウンドスレッドと照合します。(ROLE は使用されません。)いずれかの行が一致した場合、スレッドの INSTRUMENTED 値は YES になり、それ以外の場合 NO になります。これにより、ホスト、ユーザー、またはホストとユーザーの組み合わせごとに、インストゥルメントが選択して適用されます。

HOST および USER カラムにはリテラルのホストまたはユーザー名、または任意の名前に一致する '%' が格納されているべきです。setup_actors テーブルには最初に HOST と USER のどちらにも '%' を含む行が格納されるため、デフォルトで、モニタリングはすべての新しいフォアグラウンドスレッドに対して有効にされます。一部のフォアグラウンドスレッドにのみモニタリングを有効にするなど、限定された照合を実行するには、この行はいずれかの接続に一致するため、これを削除する必要があります。

次のように setup_actors を変更するとします。

```
DELETE FROM setup_actors;
```

現在 setup_actors は空であるため、着信接続に一致する可能性のある行はありません。したがって、パフォーマンススキーマは、新しいすべてのフォアグラウンドスレッドの INSTRUMENTED カラムを NO に設定します。

さらに setup_actors を変更するとします。

```
INSERT INTO setup_actors (HOST,USER,ROLE) VALUES('localhost','joe','%');
INSERT INTO setup_actors (HOST,USER,ROLE) VALUES('%','sam','%');
```

ここで、パフォーマンススキーマは、次のように新しい接続スレッドの INSTRUMENTED 値の設定方法を判断します。

- joe がローカルホストから接続する場合、接続は最初に挿入された行に一致します。
- joe がほかのすべてのホストから接続する場合、一致はありません。
- sam が任意のホストから接続する場合、接続は 2 番目に挿入された行に一致します。
- ほかのすべての接続の場合、一致はありません。

setup_actors テーブルへの変更は既存のスレッドに影響しません。

コンシューマによる事前フィルタリング

setup_consumers テーブルは使用可能なコンシューマの種類とどれが有効にされているかを一覧表示します。

```
mysql> SELECT * FROM setup_consumers;
+--------------------------------+---------+
| NAME                           | ENABLED |
+--------------------------------+---------+
| events_stages_current          | NO      |
| events_stages_history          | NO      |
| events_stages_history_long     | NO      |
| events_statements_current      | YES     |
| events_statements_history      | NO      |
| events_statements_history_long | NO      |
| events_waits_current           | NO      |
| events_waits_history           | NO      |
```

```
| events_waits_history_long      | NO      |
| global_instrumentation         | YES     |
| thread_instrumentation         | YES     |
| statements_digest              | YES     |
+--------------------------------+---------+
```

コンシューマステージで、事前フィルタリングに影響するように、setup_consumers テーブルを変更し、イベントの送信先を決定します。コンシューマを有効または無効にするには、その ENABLED 値を YES または NO に設定します。

setup_consumers テーブルへの変更はただちにモニタリングに影響します。

コンシューマを無効にすると、サーバーはそのコンシューマの宛先の保守に時間を費やさなくなります。たとえば、履歴イベント情報に関心がない場合、履歴コンシューマを無効にします。

```
mysql> UPDATE setup_consumers
    -> SET ENABLED = 'NO' WHERE NAME LIKE '%history%';
```

setup_consumers テーブル内のコンシューマ設定は、高いレベルから低いレベルまでの階層を形成します。次の原則が当てはまります。

- パフォーマンススキーマがコンシューマをチェックし、コンシューマが有効にされていないかぎり、コンシューマに関連付けられている宛先はイベントを受信しません。
- コンシューマは、それが依存するすべてのコンシューマ (ある場合) が有効にされている場合にのみチェックされます。
- コンシューマがチェックされていない場合、またはチェックされているが無効にされている場合、それに依存するほかのコンシューマはチェックされません。
- 依存コンシューマには独自の依存コンシューマがあることがあります。
- イベントがどの宛先にも送信されない場合、パフォーマンススキーマはそれを生成しません。

次のリストに、使用可能なコンシューマ値を説明します。いくつかの代表的なコンシューマ構成とそれらのインストゥルメンテーションへの効果については、コンシューマ構成の例を参照してください。

グローバルおよびスレッドコンシューマ

- global_instrumentation は最高レベルのコンシューマです。global_instrumentation が NO の場合、それはグローバルインストゥルメンテーションを無効にします。ほかのすべての設定は低いレベルで、チェックされず、何が設定されているかは重要でありません。グローバルまたはスレッドごとに情報が保守されず、個々のイベントが現在のイベントまたはイベント履歴テーブルに収集されません。global_instrumentation が YES の場合、パフォーマンススキーマはグローバル状態の情報を保守し、thread_instrumentation コンシューマもチェックします。
- thread_instrumentation は global_instrumentation が YES の場合のみチェックされます。そうでなければ、thread_instrumentation が NO の場合、スレッド固有のインストゥルメンテーションが無効になり、すべての低レベル設定が無視されます。スレッドごとに情報が保守されず、個々のイベントが現在のイベントまたはイベント履歴テーブルに収集されません。thread_instrumentation が YES の場合、パフォーマンススキーマはスレッド固有の情報を保守し、events_xxx_current コンシューマもチェックします。

ステートメントダイジェストコンシューマ

このコンシューマは global_instrumentation が YES である必要があり、そうでないとそれはチェックされません。ステートメントイベントコンシューマへの依存関係がないため、events_statements_current に統計を収集する必要なく、ダイジェストごとに統計を取得することができ、これはオーバーヘッドの点で有利です。逆に、ダイジェストなしで、events_statements_current で詳細ステートメントを取得できます (DIGEST および DIGEST_TEXT カラムは NULL になります)。

待機イベントコンシューマ

これらのコンシューマには global_instrumentation と thread_instrumentation の両方が YES である必要があり、そうでないと、それらはチェックされません。チェックされた場合、それらは次のように動作します。

- events_waits_currentは NO の場合、events_waits_current テーブル内の個々の待機イベントの収集を無効にします。YES の場合、待機イベント収集を有効にし、パフォーマンススキーマは events_waits_history および events_waits_history_long コンシューマをチェックします。

- events_waits_history は event_waits_current が NO の場合にチェックされません。そうでない場合、NO または YES の events_waits_history 値は、events_waits_history テーブルへの待機イベントの収集を無効または有効にします。
- events_waits_history_long は event_waits_current が NO の場合にチェックされません。そうでない場合、NO または YES の events_waits_history_long 値は、events_waits_history_long テーブルへの待機イベントの収集を無効または有効にします。

ステージイベントコンシューマ

これらのコンシューマには global_instrumentation と thread_instrumentation の両方が YES である必要があり、そうでないと、それらはチェックされません。チェックされた場合、それらは次のように動作します。

- events_stages_currentは、NO の場合に、events_stages_current テーブルへの個々のステージイベントの収集を無効にします。YES の場合、ステージイベント収集を有効にし、パフォーマンススキーマは events_stages_history および events_stages_history_long コンシューマをチェックします。
- events_stages_history は event_stages_current が NO の場合にチェックされません。そうでない場合、NO または YES の events_stages_history 値は、events_stages_history テーブルへのステージイベントの収集を無効または有効にします。
- events_stages_history_long は event_stages_current が NO の場合にチェックされません。そうでない場合、NO または YES の events_stages_history_long 値は、events_stages_history_long テーブルへのステージイベントの収集を無効または有効にします。

ステートメントイベントコンシューマ

これらのコンシューマには global_instrumentation と thread_instrumentation の両方が YES である必要があり、そうでないと、それらはチェックされません。チェックされた場合、それらは次のように動作します。

- events_statements_currentは NO の場合、events_statements_current テーブルへの個々のステートメントイベントの収集を無効にします。YES の場合、ステートメントイベント収集を有効にし、パフォーマンススキーマは events_statements_history および events_statements_history_long コンシューマをチェックします。
- events_statements_history は events_statements_current が NO の場合にチェックされません。そうでない場合、NO または YES の events_statements_history 値は、events_statements_history テーブルへのステートメントイベントの収集を無効または有効にします。
- events_statements_history_long は events_statements_current が NO の場合にチェックされません。そうでない場合、NO または YES の events_statements_history_long 値は、events_statements_history_long テーブルへのステートメントイベントの収集を無効または有効にします。

コンシューマ構成の例

setup_consumers テーブル内のコンシューマ設定は、高いレベルから低いレベルまでの階層を形成します。次の説明では、コンシューマのしくみを説明し、コンシューマ設定が高から低に段階に有効にされたときの特定の構成とそれらの効果を示します。示されているコンシューマ値は代表的なものです。ここで説明する一般原則は、使用可能なほかのコンシューマ値に当てはまります。

構成の説明は、機能とオーバーヘッドが増加する順番で示しています。低レベルの設定を有効にして提供される情報が必要でない場合は、それらを無効にすると、パフォーマンススキーマがユーザーのために実行するコードが少なくなり、ユーザーが取捨選択する情報が少なくなります。

setup_consumers テーブルには次の値の階層が格納されます。

```
global_instrumentation
 thread_instrumentation
   events_waits_current
     events_waits_history
     events_waits_history_long
   events_stages_current
     events_stages_history
     events_stages_history_long
   events_statements_current
     events_statements_history
     events_statements_history_long
 statements_digest
```

> **注記**
>
> コンシューマ階層で、待機、ステージ、およびステートメントのコンシューマはすべて同じレベルになります。これは、待機イベントがステージイベント内にネストし、ス

テージイベントがステートメントイベント内にネストするイベントネスト階層とは異なります。

特定のコンシューマ設定が NO の場合、パフォーマンススキーマはコンシューマに関連付けられたインストゥルメンテーションを無効にし、すべての低レベルの設定を無視します。特定の設定が YES の場合、パフォーマンススキーマはそれに関連付けられたインストゥルメンテーションを有効にし、次の最低レベルの設定をチェックします。各コンシューマのルールの説明については、コンシューマによる事前フィルタリングを参照してください。

たとえば、global_instrumentation が有効にされている場合、thread_instrumentation がチェックされます。thread_instrumentation が有効にされている場合、events_xxx_current コンシューマがチェックされます。これらのうち events_waits_current が有効にされている場合、events_waits_history および events_waits_history_long がチェックされます。

次の構成の各説明は、パフォーマンススキーマがチェックするセットアップ要素と、それが保守する出力テーブル (つまり、それが情報を収集するテーブル) を示します。

インストゥルメンテーションなし

サーバー構成の状態:

```
mysql> SELECT * FROM setup_consumers;
+--------------------------+---------+
| NAME                     | ENABLED |
+--------------------------+---------+
| global_instrumentation   | NO      |
...
+--------------------------+---------+
```

この構成では、何もインストゥルメントされません。

チェックされるセットアップ要素:

- テーブル setup_consumers、コンシューマ global_instrumentation

保守される出力テーブル:

- なし

グローバルインストゥルメンテーションのみ

サーバー構成の状態:

```
mysql> SELECT * FROM setup_consumers;
+--------------------------+---------+
| NAME                     | ENABLED |
+--------------------------+---------+
| global_instrumentation   | YES     |
| thread_instrumentation   | NO      |
...
+--------------------------+---------+
```

この構成では、インストゥルメンテーションがグローバル状態に対してのみ保守されます。スレッドごとのインストゥルメンテーションは無効にされます。

先述の構成に関連して、チェックされる追加のセットアップ要素:

- テーブル setup_consumers、コンシューマ thread_instrumentation
- テーブル setup_instruments
- テーブル setup_objects
- テーブル setup_timers

先述の構成に関連して保守される追加の出力テーブル:

- mutex_instances
- rwlock_instances

- cond_instances
- file_instances
- users
- hosts
- accounts
- socket_summary_by_event_name
- file_summary_by_instance
- file_summary_by_event_name
- objects_summary_global_by_type
- table_lock_waits_summary_by_table
- table_io_waits_summary_by_index_usage
- table_io_waits_summary_by_table
- events_waits_summary_by_instance
- events_waits_summary_global_by_event_name
- events_stages_summary_global_by_event_name
- events_statements_summary_global_by_event_name

グローバルおよびスレッドインストゥルメンテーションのみ

サーバー構成の状態:

```
mysql> SELECT * FROM setup_consumers;
+--------------------------------+---------+
| NAME                           | ENABLED |
+--------------------------------+---------+
| global_instrumentation         | YES     |
| thread_instrumentation         | YES     |
| events_waits_current           | NO      |
...
| events_stages_current          | NO      |
...
| events_statements_current      | YES     |
...
+--------------------------------+---------+
```

この構成では、インストゥルメンテーションがグローバルおよびスレッドごとに保守されます。現在のイベントまたはイベント履歴テーブルで、個々のイベントは収集されません。

先述の構成に関連して、チェックされる追加のセットアップ要素:

- テーブル setup_consumers、コンシューマ events_xxx_current。ここで xxx は waits、stages、statements です
- テーブル setup_actors
- カラム threads.instrumented

先述の構成に関連して保守される追加の出力テーブル:

- events_xxx_summary_by_yyy_by_event_name。ここで xxx は waits、stages、statements で、および yyy は thread、user、host、account です

グローバル、スレッド、および現在のイベントインストゥルメンテーション

サーバー構成の状態:

```
mysql> SELECT * FROM setup_consumers;
+--------------------------------+---------+
```

```
| NAME                          | ENABLED |
+-------------------------------+---------+
| global_instrumentation        | YES     |
| thread_instrumentation        | YES     |
| events_waits_current          | YES     |
| events_waits_history          | NO      |
| events_waits_history_long     | NO      |
| events_stages_current         | YES     |
| events_stages_history         | NO      |
| events_stages_history_long    | NO      |
| events_statements_current     | YES     |
| events_statements_history     | NO      |
| events_statements_history_long | NO     |
...
+-------------------------------+---------+
```

この構成では、インストゥルメンテーションがグローバルおよびスレッドごとに保守されます。個々のイベントが現在のイベントテーブルに収集されますが、イベント履歴テーブルには収集されません。

先述の構成に関連して、チェックされる追加のセットアップ要素:

- コンシューマ events_xxx_history。ここで xxx は waits、stages、statements です
- コンシューマ events_xxx_history_long。ここで xxx は waits、stages、statements です

先述の構成に関連して保守される追加の出力テーブル:

- events_xxx_current。ここで xxx は waits、stages、statements です

グローバル、スレッド、現在のイベント、およびイベント履歴インストゥルメンテーション

events_xxx_history および events_xxx_history_long コンシューマは無効にされているため、先述の構成はイベント履歴を収集しません。それらのコンシューマは個別またはまとめて有効にして、スレッドごと、グローバルに、またはその両方でイベント履歴を収集できます。

この構成はスレッドごとにイベントを収集しますが、グローバルには収集しません。

```
mysql> SELECT * FROM setup_consumers;
+-------------------------------+---------+
| NAME                          | ENABLED |
+-------------------------------+---------+
| global_instrumentation        | YES     |
| thread_instrumentation        | YES     |
| events_waits_current          | YES     |
| events_waits_history          | YES     |
| events_waits_history_long     | NO      |
| events_stages_current         | YES     |
| events_stages_history         | YES     |
| events_stages_history_long    | NO      |
| events_statements_current     | YES     |
| events_statements_history     | YES     |
| events_statements_history_long | NO     |
...
+-------------------------------+---------+
```

この構成で保守されるイベント履歴テーブル:

- events_xxx_history。ここで xxx は waits、stages、statements です

この構成はグローバルにイベント履歴を収集しますが、スレッドごとには収集しません。

```
mysql> SELECT * FROM setup_consumers;
+-------------------------------+---------+
| NAME                          | ENABLED |
+-------------------------------+---------+
| global_instrumentation        | YES     |
| thread_instrumentation        | YES     |
| events_waits_current          | YES     |
| events_waits_history          | NO      |
| events_waits_history_long     | YES     |
| events_stages_current         | YES     |
| events_stages_history         | NO      |
| events_stages_history_long    | YES     |
| events_statements_current     | YES     |
| events_statements_history     | NO      |
| events_statements_history_long | YES    |
```

```
...
+--------------------------------+---------+
```

この構成で保守されるイベント履歴テーブル:

- events_xxx_history_long。ここで xxx は waits、stages、statements です

この構成はスレッドごとおよびグローバルにイベントを収集します。

```
mysql> SELECT * FROM setup_consumers;
+--------------------------------+---------+
| NAME                           | ENABLED |
+--------------------------------+---------+
| global_instrumentation         | YES     |
| thread_instrumentation         | YES     |
| events_waits_current           | YES     |
| events_waits_history           | YES     |
| events_waits_history_long      | YES     |
| events_stages_current          | YES     |
| events_stages_history          | YES     |
| events_stages_history_long     | YES     |
| events_statements_current      | YES     |
| events_statements_history      | YES     |
| events_statements_history_long | YES     |
...
+--------------------------------+---------+
```

この構成で保守されるイベント履歴テーブル:

- events_xxx_history。ここで xxx は waits、stages、statements です
- events_xxx_history_long。ここで xxx は waits、stages、statements です

22.2.3.4 フィルタリング操作のインストゥルメントまたはコンシューマの指定

フィルタリング操作に指定する名前は、必要に応じて具体的なものでも一般的なものでも指定できます。単一のインストゥルメントまたはコンシューマを示すには、その名前を省略せずに指定します。

```
mysql> UPDATE setup_instruments
    -> SET ENABLED = 'NO'
    -> WHERE NAME = 'wait/synch/mutex/myisammrg/MYRG_INFO::mutex';

mysql> UPDATE setup_consumers
    -> SET ENABLED = 'NO' WHERE NAME = 'events_waits_current';
```

インストゥルメントまたはコンシューマのグループを指定するには、グループメンバーに一致するパターンを使用します。

```
mysql> UPDATE setup_instruments
    -> SET ENABLED = 'NO'
    -> WHERE NAME LIKE 'wait/synch/mutex/%';

mysql> UPDATE setup_consumers
    -> SET ENABLED = 'NO' WHERE NAME LIKE '%history%';
```

パターンを使用する場合、それは関心のあるすべての項目に一致し、ほかの項目に一致しないように、選択してください。たとえば、すべてのファイル I/O インストゥルメントを選択するには、インストゥルメント名プリフィクス全体を含むパターンを使用することをお勧めします。

```
... WHERE NAME LIKE 'wait/io/file/%';
```

'%/file/%' のパターンは、名前の任意の位置に '/file/' のコンポーネントがあるほかのインストゥルメントに一致します。パターン '%file%' は wait/synch/mutex/sql/LOCK_des_key_file など、名前の任意の位置に 'file' のあるインストゥルメントに一致するため、あまり適切ではありません。

パターンに一致するインストゥルメントまたはコンシューマ名をチェックするには、簡単なテストを実行します。

```
mysql> SELECT NAME FROM setup_instruments WHERE NAME LIKE 'pattern';

mysql> SELECT NAME FROM setup_consumers WHERE NAME LIKE 'pattern';
```

サポートされる名前の種類については、セクション22.4「パフォーマンススキーマインストゥルメント命名規則」を参照してください。

22.2.3.5 インストゥルメントされているものの特定

パフォーマンススキーマにどのインストゥルメントが含まれているかを特定するには、常に setup_instruments テーブルをチェックすることで可能です。たとえば、InnoDB ストレージエンジンに、どのファイル関連イベントがインストゥルメントされているかを確認するには、次のクエリーを使用します。

```
mysql> SELECT * FROM setup_instruments WHERE NAME LIKE 'wait/io/file/innodb/%';
+----------------------------------------+---------+-------+
| NAME                                   | ENABLED | TIMED |
+----------------------------------------+---------+-------+
| wait/io/file/innodb/innodb_data_file | YES     | YES   |
| wait/io/file/innodb/innodb_log_file  | YES     | YES   |
| wait/io/file/innodb/innodb_temp_file | YES     | YES   |
+----------------------------------------+---------+-------+
```

このドキュメントでは、いくつかの理由で、正確に何がインストゥルメントされるかについて詳細に説明していません。

- 何がインストゥルメントされるかは、サーバーコードです。このコードへの変更は頻繁に行われ、インストゥルメントのセットにも影響します。
- すべてのインストゥルメントは数百もあるため、それらを挙げることは現実的ではありません。
- 先述のように、setup_instruments テーブルをクエリーすることによって見つけることができます。この情報は使用している MySQL のバージョンに常に最新であり、コアサーバーに含まれておらず、自動化されたツールで使用可能な、インストールしている可能性があるインストゥルメント済みのプラグインのインストゥルメンテーションも含みます。

22.3 パフォーマンススキーマクエリー

事前フィルタリングは収集されるイベント情報を制限し、特定のユーザーと関係ありません。対照的に、事後フィルタリングは、各ユーザーが、事前フィルタリングの適用後に使用可能なイベントから選択するイベント情報を制限する、適切な WHERE 句でクエリーを使用することによって実行されます。

セクション22.2.3.3「イベントの事前フィルタリング」で、ファイルインストゥルメントの事前フィルタリング方法の例を示しました。イベントテーブルにファイルと非ファイルの両方の情報が格納されている場合、事後フィルタリングはファイルイベントのみの情報を表示するもう 1 つの方法です。イベント選択を適切に制限するには、クエリーに WHERE 句を追加します。

```
mysql> SELECT THREAD_ID, NUMBER_OF_BYTES
    -> FROM events_waits_history
    -> WHERE EVENT_NAME LIKE 'wait/io/file/%'
    -> AND NUMBER_OF_BYTES IS NOT NULL;
+-----------+-----------------+
| THREAD_ID | NUMBER_OF_BYTES |
+-----------+-----------------+
|        11 |              66 |
|        11 |              47 |
|        11 |             139 |
|         5 |              24 |
|         5 |             834 |
+-----------+-----------------+
```

22.4 パフォーマンススキーマインストゥルメント命名規則

インストゥルメント名は '/' 文字で区切られた一連のコンポーネントから構成されます。名前の例:

```
wait/io/file/myisam/log
wait/io/file/mysys/charset
wait/lock/table/sql/handler
wait/synch/cond/mysys/COND_alarm
wait/synch/cond/sql/BINLOG::update_cond
wait/synch/mutex/mysys/BITMAP_mutex
wait/synch/mutex/sql/LOCK_delete
wait/synch/rwlock/sql/Query_cache_query::lock
stage/sql/closing tables
stage/sql/Sorting result
statement/com/Execute
statement/com/Query
statement/sql/create_table
statement/sql/lock_tables
```

インストゥルメントの名前空間はツリー状の構造を持ちます。インストゥルメント名のコンポーネントは左から右に、より一般的からより具体的に進みます。名前のコンポーネント数はインストゥルメントの種類によって異なります。

名前の特定のコンポーネントの解釈は、その左側のコンポーネントによって異なります。たとえば、myisam は次の名前の両方に見られますが、最初の名前の myisam はファイル I/O に関連し、2 番目のそれは同期インストゥルメントに関連しています。

```
wait/io/file/myisam/log
wait/synch/cond/myisam/MI_SORT_INFO::cond
```

インストゥルメント名は、パフォーマンススキーマ実装によって定義された構造を持つプリフィクスと、インストゥルメントコードを実装する開発者によって定義されるサフィクスから構成されます。インストゥルメントプリフィクスのトップレベルコンポーネントはインストゥルメントの種類を示します。このコンポーネントによって、インストゥルメントに適用される setup_timers テーブル内のイベントタイマーも決まります。インストゥルメントのプリフィクス部分のトップレベルはインストゥルメントの種類を示します。

インストゥルメント名のサフィクス部分は、インストゥルメント自体のコードから生成されます。サフィクスには次のようなレベルが含まれることがあります。

- 主要コンポーネントの名前 (myisam、innodb、mysys、または sql などのサーバーモジュール) またはプラグイン名。
- 形式 XXX (グローバル変数) または CCC::MMM (クラス CCC のメンバー MMM) でのコード内の変数の名前。例: COND_thread_cache、THR_LOCK_myisam、BINLOG::LOCK_index。

トップレベルインストゥルメントコンポーネント

- idle: インストゥルメントされたアイドルイベント。このインストゥルメントにはそれ以上のコンポーネントがありません。
- stage: インストゥルメントされたステージイベント。
- statement: インストゥルメントされたステートメントイベント。
- wait: インストゥルメントされた待機イベント。

アイドルインストゥルメントコンポーネント

- idle

  アイドルインストゥルメント。パフォーマンススキーマは、セクション22.9.3.5「socket_instances テーブル」の socket_instances.STATE カラムの説明に示されているように、アイドルイベントを生成します。

ステージインストゥルメントコンポーネント

ステージインストゥルメントは、形式 stage/code_area/stage_name の名前を持ちます。ここで code_area は sql や myisam などの名前で、stage_name は、Sorting result や Sending data などのステートメント処理のステージを示します。ステージは SHOW PROCESSLIST によって表示されるか、または INFORMATION_SCHEMA.PROCESSLIST テーブルに表示されるスレッドの状態に対応します。

ステートメントインストゥルメントコンポーネント

- statement/abstract/*: ステートメント操作の抽象インストゥルメント。抽象インストゥルメントは、正確なステートメントの種類が分かる前のステートメント分類の初期段階時に使用され、その後種類がわかったときに、より具体的なステートメントインストゥルメントに変更されます。このプロセスの説明については、セクション22.9.6「パフォーマンススキーマステートメントイベントテーブル」を参照してください。
- statement/com: インストゥルメントされたコマンド操作。これらは COM_xxx 操作に対応する名前を持ちます (mysql_com.h ヘッダーファイルおよび sql/sql_parse.cc を参照してください。たとえば、statement/com/Connect および statement/com/Init DB インストゥルメントは COM_CONNECT および COM_INIT_DB コマンドに対応します。
- statement/sql: インストゥルメントされた SQL ステートメント操作。たとえば、statement/sql/create_db および statement/sql/select インストゥルメントは CREATE DATABASE および SELECT ステートメントに使用されます。

待機インストゥルメントコンポーネント

- wait/io

インストゥルメントされた I/O 操作。

- wait/io/file

  インストゥルメントされたファイル I/O 操作。ファイルの場合、待機はファイル操作の完了 (たとえば、fwrite() の呼び出し) を待機する時間です。キャッシュのため、この呼び出し内で、ディスクへの物理的なファイル I/O は行われない可能性があります。

- wait/io/socket

  インストゥルメントされたソケット操作。ソケットインストゥルメントは形式 wait/io/socket/sql/socket_type の名前を持ちます。サーバーには、それがサポートする各ネットワークプロトコルの待機ソケットがあります。TCP/IP または Unix ソケットファイル接続の待機ソケットに関連付けられているインストゥルメントは、それぞれ server_tcpip_socket または server_unix_socket の socket_type 値を持ちます。待機ソケットが接続を検出すると、サーバーは接続を、個別のスレッドによって管理される新しいソケットに転送します。新しい接続スレッドのインストゥルメントは、client_connection の socket_type 値を持ちます。

- wait/io/table

  インストゥルメントされたテーブル I/O 操作。これらには、永続的ベーステーブルまたは一時テーブルへの行レベルアクセスが含まれます。行に影響する操作は、フェッチ、挿入、更新、および削除です。ビューの場合、待機はビューによって参照されるベーステーブルに関連付けられます。

  ほとんどの待機と同様、テーブル I/O の待機にはほかの待機も含まれることがあります。たとえば、テーブル I/O にはファイル I/O またはメモリー操作が含まれることがあります。そのため、テーブル I/O 待機の events_waits_current には通常 2 行あります。詳細については、セクション22.6「パフォーマンススキーマの原子的および分子的イベント」を参照してください。

  一部の行操作では、複数のテーブル I/O 待機が発生することがあります。たとえば、挿入は更新を発生させるトリガーをアクティブにすることがあります。

- wait/lock

  インストゥルメントされたロック操作。

  - wait/lock/table

    インストゥルメントされたテーブルロック操作。

- wait/synch

  インストゥルメントされた同期オブジェクト。同期オブジェクトでは、TIMER_WAIT 時間には、オブジェクトへのロックがある場合に、その獲得を試みている間のブロックされる時間の量が含まれます。

  - wait/synch/cond

    条件は、1 つのスレッドによって、ほかのスレッドに、それらが待機している何かが発生したことを伝えるために使用されます。単一のスレッドが条件を待機していた場合、それはウェイクアップし、その実行を再開できます。複数のスレッドが待機していた場合、それらすべてがウェイクアップし、それらが待機していたリソースを奪い合うことがあります。

  - wait/synch/mutex

    ほかのスレッドのリソースへのアクセスを妨げながら、リソース (実行可能コードのセクションなど) へのアクセスを許可するために使用される相互排他オブジェクト。

  - wait/synch/rwlock

    ほかのスレッドによるその使用を妨げながら、特定の変数のアクセスをロックするために使用される読み取り/書き込みロックオブジェクト。共有読み取りロックは複数のスレッドによって同時に獲得できます。排他的書き込みロックは、一度に 1 つのスレッドだけが獲得できます。

22.5 パフォーマンススキーマステータスモニタリング

パフォーマンススキーマに関連付けられたいくつかのステータス変数があります。

```
mysql> SHOW STATUS LIKE 'perf%';
```

```
+-----------------------------------------------+-------+
| Variable_name                                 | Value |
+-----------------------------------------------+-------+
| Performance_schema_accounts_lost              | 0     |
| Performance_schema_cond_classes_lost          | 0     |
| Performance_schema_cond_instances_lost        | 0     |
| Performance_schema_digest_lost                | 0     |
| Performance_schema_file_classes_lost          | 0     |
| Performance_schema_file_handles_lost          | 0     |
| Performance_schema_file_instances_lost        | 0     |
| Performance_schema_hosts_lost                 | 0     |
| Performance_schema_locker_lost                | 0     |
| Performance_schema_mutex_classes_lost         | 0     |
| Performance_schema_mutex_instances_lost       | 0     |
| Performance_schema_rwlock_classes_lost        | 0     |
| Performance_schema_rwlock_instances_lost      | 0     |
| Performance_schema_session_connect_attrs_lost | 0     |
| Performance_schema_socket_classes_lost        | 0     |
| Performance_schema_socket_instances_lost      | 0     |
| Performance_schema_stage_classes_lost         | 0     |
| Performance_schema_statement_classes_lost     | 0     |
| Performance_schema_table_handles_lost         | 0     |
| Performance_schema_table_instances_lost       | 0     |
| Performance_schema_thread_classes_lost        | 0     |
| Performance_schema_thread_instances_lost      | 0     |
| Performance_schema_users_lost                 | 0     |
+-----------------------------------------------+-------+
```

パフォーマンススキーマステータス変数は、メモリー制約のためロードまたは作成できなかったインストゥルメンテーションに関する情報を提供します。これらの変数の名前にはいくつかの形式があります。

- Performance_schema_xxx_classes_lost は、種類 xxx のロードできなかったインストゥルメントの数を示します。
- Performance_schema_xxx_instances_lost は、オブジェクトの種類 xxx の作成できなかったインストゥルメントの数を示します。
- Performance_schema_xxx_handles_lost は、オブジェクトの種類 xxx のオープンできなかったインストゥルメントの数を示します。
- Performance_schema_locker_lost は、「失われた」か、または記録されなかったイベント数を示します。

たとえば、相互排他ロックがサーバーソースにインストゥルメントされているが、サーバーが実行時にインストゥルメンテーション用のメモリーを割り当てることができない場合、それは Performance_schema_mutex_classes_lost を増分します。相互排他ロックは、まだ同期オブジェクトとして機能します (つまり、サーバーは正常に機能し続けます) が、そのパフォーマンスデータは収集されません。インストゥルメントを割り当てることができる場合、それはインストゥルメントされた相互排他ロックインスタンスの初期化に使用できます。グローバル相互排他ロックなどのシングルトン相互排他ロックの場合、1 つのインスタンスだけが存在します。その他の相互排他ロックは、接続あたり、または各種キャッシュおよびデータバッファー内のページあたりに 1 つのインスタンスを持つため、インスタンスの数は時間の経過とともに変わります。接続の最大数または一部のバッファーの最大サイズを増やすと、一度に割り当てることができるインスタンスの最大数が増えます。サーバーが特定のインストゥルメントされた相互排他ロックインスタンスを作成できない場合、それは Performance_schema_mutex_instances_lost を増分します。

次の条件が維持されているとします。

- サーバーは --performance_schema_max_mutex_classes=200 オプションを使用して起動されているため、200 相互排他ロックインストゥルメントのための空きがあります。
- 150 相互排他ロックインストゥルメントはすでにロードされています。
- plugin_a という名前のプラグインには 40 相互排他ロックインストゥルメントが含まれます。
- plugin_b という名前のプラグインには 20 相互排他ロックインストゥルメントが含まれます。

サーバーは、次の一連のステートメントに示すように、プラグインに、それらが必要とする数と使用可能な数に応じて、相互排他ロックインストゥルメントを割り当てます。

```
INSTALL PLUGIN plugin_a
```

現在サーバーには 150+40 = 190 相互排他ロックインストゥルメントがあります。

```
UNINSTALL PLUGIN plugin_a;
```

サーバーにはまだ 190 インストゥルメントがあります。プラグインコードによって生成されたすべての履歴データはまだ使用できますが、インストゥルメントの新しいイベントは収集されません。

```
INSTALL PLUGIN plugin_a;
```

サーバーは 40 インストゥルメントがすでに定義されていることを検出したため、新しいインストゥルメントが作成されず、以前に割り当てられた内部メモリーバッファーが再利用されます。サーバーにはまだ 190 インストゥルメントがあります。

```
INSTALL PLUGIN plugin_b;
```

サーバーは 200-190 = 10 インストゥルメント (この例では、相互排他ロッククラス) の空きがあり、プラグインに 20 個の新しいインストゥルメントが含まれていることを認識します。10 個のインストゥルメントがロードされ、10 個は破棄されるか「失われます。」Performance_schema_mutex_classes_lost は失われたインストゥルメント (相互排他ロッククラス) の数を示します。

```
mysql> SHOW STATUS LIKE "perf%mutex_classes_lost";
+---------------------------------------+-------+
| Variable_name                         | Value |
+---------------------------------------+-------+
| Performance_schema_mutex_classes_lost | 10    |
+---------------------------------------+-------+
1 row in set (0.10 sec)
```

インストゥルメンテーションはまだ機能し、plugin_b の (部分) データを収集します。

サーバーが相互排他ロックインストゥルメントを作成できないと、次の結果が発生します。

- インストゥルメントの行が setup_instruments テーブルに挿入されません。
- Performance_schema_mutex_classes_lost が 1 増加します。
- Performance_schema_mutex_instances_lost は変更されません。(相互排他ロックインストゥルメントが作成されない場合、あとでインストゥルメントされた相互排他ロックインスタンスの作成にそれを使用できません。)

先述のパターンは相互排他ロックだけでなく、すべての種類のインストゥルメントに当てはまります。

0 より大きい Performance_schema_mutex_classes_lost の値は 2 つのケースで発生する可能性があります。

- 数バイトのメモリーを節約するには、サーバーを --performance_schema_max_mutex_classes=N で起動します。ここで N はデフォルト値未満です。デフォルト値を選択すれば、MySQL 配布で提供されるすべてのプラグインをロードするために十分ですが、一部のプラグインをロードしない場合にこれを減らすことができます。たとえば、配布内の一部のストレージエンジンをロードしないように選択できます。
- パフォーマンススキーマ用にインストゥルメントされたサードパーティープラグインをロードしますが、サーバーの起動時に、プラグインのインストゥルメンテーションメモリー要件を考慮しません。それはサードパーティー製であるため、このエンジンのインストゥルメントメモリー消費は performance_schema_max_mutex_classes で選択されたデフォルト値で考慮されていません。

  サーバーにプラグインのインストゥルメント用の十分なリソースがなく、ユーザーが --performance_schema_max_mutex_classes=N を使用して、明示的に多く割り当てていない場合、このプラグインをロードすると、インストゥルメントが不足します。

performance_schema_max_mutex_classes に選択されている値が小さすぎる場合、エラーログにエラーが報告されず、実行時に障害が発生しません。ただし、performance_schema データベース内のテーブルの内容にイベントが含まれません。Performance_schema_mutex_classes_lost ステータス変数は、インストゥルメントの作成の失敗のために、一部のイベントが内部で削除されたことを示す唯一の目に見えるしるしです。

インストゥルメントが失われていない場合、それはパフォーマンススキーマに認識され、インスタンスのインストゥルメント時に使用されます。たとえば、wait/synch/mutex/sql/LOCK_delete は setup_instruments テーブル内の相互排他ロックインストゥルメントの名前です。サーバーの実行時に相互排他ロックの多くのインスタンスが必要であっても、コード内 (THD::LOCK_delete 内) で相互排他ロックの作成時に、この単一のインストゥルメントが使用されます。この場合、LOCK_delete は接続 (THD) ごとの相互排他ロックであるため、サーバーに 1000 接続がある場合、1000 スレッドと 1000 個のインストゥルメントされた LOCK_delete 相互排他ロックインスタンス (THD::LOCK_delete) が存在します。

サーバーにこれらの 1000 個のインストゥルメントされた相互排他ロック (インスタンス) すべてのための空きがない場合、一部の相互排他ロックはインストゥルメンテーション付きで作成され、一部はインストゥルメンテーションなしで作成されます。サーバーが 800 インスタンスしか作成できない場合、200 インスタンスが失われま

す。サーバーは実行し続けますが、Performance_schema_mutex_instances_lost を 200 増分し、インスタンスを作成できなかったことを示します。

0 より大きい Performance_schema_mutex_instances_lost は、--performance_schema_max_mutex_instances=N で割り当てられたものより多くの相互排他ロックを、コードで実行時に初期化した場合に発生する可能性があります。

結論として、SHOW STATUS LIKE 'perf%' に何も失われていないことが示されている (すべての値がゼロ) 場合に、パフォーマンススキーマデータは正確で信頼できます。何かが失われた場合、データは不完全であり、使用するように与えられたメモリーの量が不十分であったとすると、パフォーマンススキーマはすべてを記録できていません。この場合、特定の Performance_schema_xxx_lost 変数に問題の領域が示されます。

場合によって、意図的にインストゥルメントの不足を発生させることが適切なことがあります。たとえば、ファイル I/O のパフォーマンスデータに関心がない場合は、ファイル I/O に関連するすべてのパフォーマンススキーマのパラメータを 0 に設定して、サーバーを起動できます。ファイル関連のクラス、インスタンス、またはハンドルにメモリーが割り当てられず、すべてのファイルイベントが失われます。

SHOW ENGINE PERFORMANCE_SCHEMA STATUS を使用して、パフォーマンススキーマコードの内部操作を検査します。

```
mysql> SHOW ENGINE PERFORMANCE_SCHEMA STATUS\G
...
*************************** 3. row ***************************
  Type: performance_schema
  Name: events_waits_history.row_size
Status: 76
*************************** 4. row ***************************
  Type: performance_schema
  Name: events_waits_history.row_count
Status: 10000
*************************** 5. row ***************************
  Type: performance_schema
  Name: events_waits_history.memory
Status: 760000
...
*************************** 57. row ***************************
  Type: performance_schema
  Name: performance_schema.memory
Status: 26459600
...
```

このステートメントは、さまざまなパフォーマンススキーマオプションがメモリー要件に与える効果について、DBA が理解できるようにすることを目的としています。フィールドの意味の説明については、セクション 13.7.5.16「SHOW ENGINE 構文」を参照してください。

22.6 パフォーマンススキーマの原子的および分子的イベント

テーブル I/O イベントの場合、events_waits_current には通常 1 行ではなく、2 行あります。たとえば、行フェッチにより、次のような行になる可能性があります。

```
Row# EVENT_NAME               TIMER_START TIMER_END
---- ---------- ----------- ---------
   1 wait/io/file/myisam/dfile       10001 10002
   2 wait/io/table/sql/handler       10000 NULL
```

行フェッチによりファイルが読み取られます。例では、テーブル I/O フェッチイベントがファイル I/O イベントの前に起動していますが、終了していません (その TIMER_END 値が NULL です)。ファイル I/O イベントはテーブル I/O イベント内に「ネスト」されます。

これは、相互排他ロックやファイル I/O などのほかの「原子的」待機イベントと異なり、テーブル I/O イベントは「分子的」で、ほかのイベントを含んで (重複して) います。events_waits_current で、テーブル I/O イベントには通常 2 つの行があります。

- 最新のテーブル I/O 待機イベントについての 1 行
- 任意の種類の最新の待機イベントについての 1 行

通常、ただし常にではありませんが、「どの種類の」待機イベントもテーブル I/O イベントと異なります。各従属イベントが完了すると、それは events_waits_current から消去されます。この時点で、および次の従属イベントが開始されるまで、テーブル I/O 待機は、あらゆる種類の最新の待機でもあります。

22.7 パフォーマンススキーマステートメントダイジェスト

MySQL 5.6.5 現在、パフォーマンススキーマはステートメントダイジェスト情報を保守します。ダイジェストは、SQL ステートメントを正規化形式に変換し、結果のハッシュ値を計算します。正規化により、類似のステートメントがグループ化され、要約されて、サーバーが実行しているステートメントの種類とそれらが発生する頻度に関する情報が公開されます。このセクションでは、どのようにステートメントの正規化が行われ、どのように役立つ可能性があるかについて説明します。

ステートメントのダイジェストには、これらのパフォーマンススキーマコンポーネントが関係します。

- setup_consumers テーブル内の statement_digest コンシューマは、パフォーマンススキーマがダイジェスト情報を保守するかどうかを制御します。
- ステートメントイベントテーブル (events_statements_current、events_statements_history、および events_statements_history_long) には、ダイジェスト MD5 値と対応する正規化されたステートメントテキスト文字列を格納する DIGEST および DIGEST_TEXT カラムがあります。
- events_statements_summary_by_digest テーブルは集計されたステートメントダイジェスト情報を提供します。

ステートメントの正規化によって、ステートメントテキストは、一般的なステートメント構造を保持しながら、構造に不可欠でない情報を削除する、より標準化された文字列表現に変換されます。データベースまたはテーブル名などのオブジェクト識別子は保持されます。値とコメントは削除され、空白は調整されます。パフォーマンススキーマは、名前、パスワード、日付などの情報を保持しません。

これらのステートメントを考慮してください。

```
SELECT * FROM orders WHERE customer_id=10 AND quantity>20
SELECT * FROM orders WHERE customer_id = 20 AND quantity > 100
```

これらのステートメントを正規化するため、パフォーマンススキーマはデータ値を ? に置換し、空白を調整します。どちらのステートメントも同じ正規化形式になるため、「同じ」とみなされます。

```
SELECT * FROM orders WHERE customer_id = ? AND quantity > ?
```

正規化されたステートメントは、格納される情報は少ないですが、引き続き元のステートメントを代表しています。さまざまな比較値を持つほかの類似のステートメントは同じ正規化形式になります。

ここで、これらのステートメントを考慮してください。

```
SELECT * FROM customers WHERE customer_id = 1000
SELECT * FROM orders WHERE customer_id = 1000
```

この場合、ステートメントは「同じ」ではありません。オブジェクト識別子が異なるため、ステートメントは異なる正規化形式になります。

```
SELECT * FROM customers WHERE customer_id = ?
SELECT * FROM orders WHERE customer_id = ?
```

正規化されたステートメントは固定長です。DIGEST_TEXT 値の最大長は 1024 バイトです。この最大を変更するオプションはありません。正規化によって、この長さを超えるステートメントが生成された場合、テキストは「...」 で終わります。「...」 のあとの部分のみが異なる長いステートメントは同じとみなされます。これらのステートメントを考慮してください。

```
SELECT * FROM mytable WHERE cola = 10 AND colb = 20
SELECT * FROM mytable WHERE cola = 10 AND colc = 20
```

AND の直後にカットオフが発生した場合、両方のステートメントがこの正規化形式になります。

```
SELECT * FROM mytable WHERE cola = ? AND ...
```

この場合、2 つ目のカラム名の違いが失われ、両方のステートメントが同じとみなされます。

正規化された各ステートメントについて、パフォーマンススキーマはハッシュダイジェスト値を計算し、その値とステートメントをステートメントイベントテーブル (events_statements_current、events_statements_history、および events_statements_history_long) の DIGEST および DIGEST_TEXT カラムに格納します。さらに、同じ SCHEMA_NAME および DIGEST 値を持つステートメントの情報が events_statements_summary_by_digest サマリーテーブルに集計されます。パフォーマンススキーマは MD5 ハッシュ値を使用します。それらは計算が速く、競合を最小にする望ましい統計的分布を持つためです。

events_statements_summary_by_digest サマリーテーブルは固定サイズであるため、それがいっぱいになると、テーブル内の既存の値に一致しない SCHEMA_NAME および DIGEST 値のあるステートメントは、SCHEMA_NAME および DIGEST が NULL に設定された特別な行にグループ化されます。これにより、すべてのステートメントがカウントされます。ただし、特別な行が、実行されるステートメントの大きな割合を占める場合、サマリーテーブルのサイズを増やすことが望ましい可能性があります。これを実行するには、サーバーの起動時に、performance_schema_digests_size システム変数を大きな値に設定します。performance_schema_digests_size 値が指定されていない場合、サーバーは起動時に使用する値を推定します。(MySQL 5.6.9 より前では、SCHEMA_NAME カラム値がなく、特別な行の DIGEST は NULL に設定されます。)

ステートメントダイジェストサマリーテーブルは、サーバーによって実行されるステートメントのプロファイルを提供します。それは、アプリケーションが実行しているステートメントの種類とその頻度を示します。アプリケーション開発者はこの情報をテーブル内のほかの情報と組み合わせて使用し、アプリケーションのパフォーマンス特性を査定できます。たとえば、待機時間、ロック時間、またはインデックスの使用を示すテーブルカラムは不十分なクエリーの種類を強調表示することができます。これにより、開発者が注意が必要なアプリケーションの部分を把握できます。

22.8 パフォーマンススキーマの一般的なテーブル特性

performance_schema データベースの名前は小文字で、その中のテーブルの名前も同様です。クエリーでは名前を小文字で指定してください。

performance_schema データベース内のほとんどのテーブルは読み取り専用で、変更できません。セットアップテーブルの一部には、パフォーマンススキーマ操作に影響するように変更できるカラムがあります。さらに行の挿入や削除を許可するものもあります。収集されたイベントをクリアするために切り捨てが許可されるため、events_waits_ のプリフィクスのある名前のテーブルなど、それらの種類の情報を格納するテーブルで、TRUNCATE TABLE を使用できます。

TRUNCATE TABLE は、events_statements_summary_by_digest を除くサマリーテーブルでも使用できますが、その効果は行の削除ではなく、サマリーカラムを 0 または NULL にリセットすることです。

ほかのデータベースやテーブルに対する権限は次のようになります。

- performance_schema テーブルから取得するには、SELECT 権限が必要です。
- 変更可能なそれらのカラムを変更するには、UPDATE 権限が必要です。
- 切り捨て可能なテーブルを切り捨てるには、DROP 権限が必要です。

22.9 パフォーマンススキーマテーブルの説明

performance_schema データベース内のテーブルは次のようにグループ化できます。

- セットアップテーブル。これらのテーブルは、モニタリング特性の構成と表示に使われます。
- 現在のイベントテーブル。events_waits_current テーブルには各スレッドの最新のイベントが格納されます。その他の類似のテーブルには、イベント階層のさまざまなレベルの現在のイベントが格納されます (ステージイベント用の events_stages_current とステートメントイベント用の events_statements_current)。
- 履歴テーブル。これらのテーブルは現在のイベントテーブルと同じ構造を持ちますが、多くの行を格納します。たとえば、待機イベントの場合、events_waits_history テーブルにはスレッドあたり最新の 10 イベントが格納されます。events_waits_history_long には最新の 10,000 イベントが格納されます。ステージおよびステートメント履歴用にほかの類似テーブルが存在します。

  履歴テーブルのサイズを変更するには、サーバー起動時に、該当するシステム変数を設定します。たとえば、待機イベント履歴テーブルのサイズを設定するには、performance_schema_events_waits_history_size および performance_schema_events_waits_history_long_size を設定します。
- サマリーテーブル。これらのテーブルには、履歴テーブルから破棄されたものを含むイベントのグループ全体で集計された情報が格納されます。
- インスタンステーブル。これらのテーブルは、インストゥルメントされたオブジェクトの種類を記述します。インストゥルメントされたオブジェクトは、サーバーによって使われると、イベントを生成します。これらのテーブルは、イベント名と説明のメモまたはステータス情報を提供します。
- その他のテーブル。これらはほかのどのテーブルグループにも分類されません。

22.9.1 パフォーマンススキーマテーブルインデックス

次の表に、各パフォーマンススキーマテーブルを一覧表示し、それぞれの簡単な説明を提供します。

表 22.1 パフォーマンススキーマテーブル

テーブル名	説明
accounts	クライアントアカウントごとの接続統計
cond_instances	同期オブジェクトインスタンス
events_stages_current	現在のステージイベント
events_stages_history	各スレッドの最新ステージイベント
events_stages_history_long	全体の最新ステージイベント
events_stages_summary_by_account_by_event_name	アカウントおよびイベント名ごとのステージイベント
events_stages_summary_by_host_by_event_name	ホスト名およびイベント名ごとのステージイベント
events_stages_summary_by_thread_by_event_name	スレッドおよびイベント名ごとのステージ待機
events_stages_summary_by_user_by_event_name	ユーザー名およびイベント名ごとのステージイベント
events_stages_summary_global_by_event_name	イベント名ごとのステージ待機
events_statements_current	現在のステートメントイベント
events_statements_history	各スレッドの最新ステートメントイベント
events_statements_history_long	全体の最新ステートメントイベント
events_statements_summary_by_account_by_event_name	アカウントおよびイベント名ごとのステートメントイベント
events_statements_summary_by_digest	スキーマおよびダイジェスト値ごとのステートメントイベント
events_statements_summary_by_host_by_event_name	ホスト名およびイベント名ごとのステートメントイベント
events_statements_summary_by_thread_by_event_name	スレッドおよびイベント名ごとのステートメントイベント
events_statements_summary_by_user_by_event_name	ユーザー名およびイベント名ごとのステートメントイベント
events_statements_summary_global_by_event_name	イベント名ごとのステートメントイベント
events_waits_current	現在の待機イベント
events_waits_history	各スレッドの最新待機イベント
events_waits_history_long	全体の最新待機イベント
events_waits_summary_by_account_by_event_name	アカウントおよびイベント名ごとの待機イベント
events_waits_summary_by_host_by_event_name	ホスト名およびイベント名ごとの待機イベント
events_waits_summary_by_instance	インスタンスごとの待機イベント
events_waits_summary_by_thread_by_event_name	スレッドおよびイベント名ごとの待機イベント
events_waits_summary_by_user_by_event_name	ユーザー名およびイベント名ごとの待機イベント
events_waits_summary_global_by_event_name	イベント名ごとの待機イベント
file_instances	ファイルインスタンス
file_summary_by_event_name	イベント名ごとのファイルイベント
file_summary_by_instance	ファイルインスタンスごとのファイルイベント
host_cache	内部ホストキャッシュからの情報
hosts	クライアントホスト名ごとの接続統計
mutex_instances	相互排他ロック同期オブジェクトインスタンス
objects_summary_global_by_type	オブジェクトサマリー

テーブル名	説明
performance_timers	使用可能なイベントタイマー
rwlock_instances	ロック同期オブジェクトインスタンス
session_account_connect_attrs	現在のセッションごとの接続属性
session_connect_attrs	すべてのセッションの接続属性
setup_actors	新しいフォアグラウンドスレッドのモニタリングの初期化方法
setup_consumers	イベント情報を格納できるコンシューマ
setup_instruments	イベントを収集できるインストゥルメントされたオブジェクトのクラス
setup_objects	モニターすべきオブジェクト
setup_timers	現在のイベントタイマー
socket_instances	アクティブな接続インスタンス
socket_summary_by_event_name	イベント名ごとのソケット待機と I/O
socket_summary_by_instance	インスタンスごとのソケット待機と I/O
table_io_waits_summary_by_index_usage	インデックスごとのテーブル I/O 待機
table_io_waits_summary_by_table	テーブルごとのテーブル I/O 待機
table_lock_waits_summary_by_table	テーブルごとのテーブルロック待機
threads	サーバースレッドに関する情報
users	クライアントユーザー名ごとの接続統計

22.9.2 パフォーマンススキーマセットアップテーブル

セットアップテーブルは現在のインストゥルメンテーションに関する情報を提供し、モニタリング構成の変更を可能にします。このため、UPDATE 権限を持つ場合、これらのテーブルの一部のカラムを変更できます。

セットアップ情報に個々の変数ではなく、テーブルを使用することで、パフォーマンススキーマ構成の変更の高度な柔軟性を提供します。たとえば、標準 SQL 構文の単一のステートメントを使用して、複数の同時の構成変更ができます。

これらのセットアップテーブルを使用できます。

- setup_actors: 新しいフォアグラウンドスレッドのモニタリングの初期化方法
- setup_consumers: イベント情報を送信および格納できる宛先
- setup_instruments: イベントを収集できるインストゥルメントされたオブジェクトのクラス
- setup_objects: モニターすべきオブジェクト
- setup_timers: 現在のイベントタイマー

22.9.2.1 setup_actors テーブル

setup_actors テーブルには、新しいフォアグラウンドサーバースレッド、つまりクライアント接続に関連付けられたスレッドのモニタリングを有効にするかどうかを決定する情報が格納されます。このテーブルはデフォルトで 100 行の最大サイズになります。テーブルサイズを変更するには、サーバー起動時に performance_schema_setup_actors_size システム変数を変更します。

新しいフォアグラウンドスレッドごとに、パフォーマンススキーマはスレッドのユーザーとホストを、setup_actors テーブルの行に対して照合します。一致する行があるかどうかに基づいて、スレッドの threads テーブル行の INSTRUMENTED カラムが YES または NO に設定されます。これにより、ホスト、ユーザー、またはホストとユーザーの組み合わせごとに、インストゥルメントが選択して適用されます。

setup_actors テーブルの初期の内容は、任意のユーザーとホストの組み合わせに一致するため、デフォルトですべてのフォアグラウンドスレッドのモニタリングが有効にされます。

```
mysql> SELECT * FROM setup_actors;
+------+------+------+
| HOST | USER | ROLE |
+------+------+------+
```

```
| %    | %    | %    |
+------+------+------+
```

setup_actors テーブルへの変更は既存のスレッドに影響しません。

イベントモニタリングで setup_actors テーブルを使用する方法については、スレッドによる事前フィルタリングを参照してください。

setup_actors テーブルにはこれらのカラムがあります。

- HOST

  ホスト名。これらはリテラル名または「任意のホスト」を意味する '%' であるべきです。

- USER

  ユーザー名。これはリテラル名または「任意のユーザー」を意味する '%' であるべきです。

- ROLE

  使用されません。

22.9.2.2 setup_consumers テーブル

setup_consumers テーブルは、イベント情報を格納でき、有効にされているコンシューマの種類を一覧表示します。

```
mysql> SELECT * FROM setup_consumers;
+--------------------------------+---------+
| NAME                           | ENABLED |
+--------------------------------+---------+
| events_stages_current          | NO      |
| events_stages_history          | NO      |
| events_stages_history_long     | NO      |
| events_statements_current      | YES     |
| events_statements_history      | NO      |
| events_statements_history_long | NO      |
| events_waits_current           | NO      |
| events_waits_history           | NO      |
| events_waits_history_long      | NO      |
| global_instrumentation         | YES     |
| thread_instrumentation         | YES     |
| statements_digest              | YES     |
+--------------------------------+---------+
```

setup_consumers テーブル内のコンシューマ設定は、高いレベルから低いレベルまでの階層を形成します。さまざまなコンシューマを有効にすることの効果の詳細については、コンシューマによる事前フィルタリングを参照してください。

setup_consumers テーブルへの変更はただちにモニタリングに影響します。

setup_consumers テーブルにはこれらのカラムがあります。

- NAME

  コンシューマ名。

- ENABLED

  コンシューマが有効にされているかどうか。このカラムは変更できます。コンシューマを無効にすると、サーバーはそれにイベント情報を追加する時間を費やさなくなります。

22.9.2.3 setup_instruments テーブル

setup_instruments テーブルは、イベントを収集できるインストゥルメントされたオブジェクトのクラスを一覧表示します。

```
mysql> SELECT * FROM setup_instruments;
+---------------------------------------------------------+---------+-------+
| NAME                                                    | ENABLED | TIMED |
+---------------------------------------------------------+---------+-------+
...
| wait/synch/mutex/sql/LOCK_global_read_lock              | YES     | YES   |
| wait/synch/mutex/sql/LOCK_global_system_variables       | YES     | YES   |
```

```
| wait/synch/mutex/sql/LOCK_lock_db                 | YES     | YES   |
| wait/synch/mutex/sql/LOCK_manager                 | YES     | YES   |
...
| wait/synch/rwlock/sql/LOCK_grant                  | YES     | YES   |
| wait/synch/rwlock/sql/LOGGER::LOCK_logger         | YES     | YES   |
| wait/synch/rwlock/sql/LOCK_sys_init_connect       | YES     | YES   |
| wait/synch/rwlock/sql/LOCK_sys_init_slave         | YES     | YES   |
...
| wait/io/file/sql/binlog                           | YES     | YES   |
| wait/io/file/sql/binlog_index                     | YES     | YES   |
| wait/io/file/sql/casetest                         | YES     | YES   |
| wait/io/file/sql/dbopt                            | YES     | YES   |
...
```

ソースコードに追加された各インストゥルメントは、インストゥルメントされたコードが実行されない場合でもこのテーブルの行を提供します。インストゥルメントが有効にされており、実行されると、*_instances テーブルに表示されるインストゥルメントされたインスタンスが作成されます。

setup_instruments テーブルへの変更はただちにモニタリングに影響します。

イベントフィルタリングにおける setup_instruments テーブルの役割の詳細については、セクション22.2.3.3「イベントの事前フィルタリング」を参照してください。

setup_instruments テーブルにはこれらのカラムがあります。

- NAME

  インストゥルメント名。セクション22.4「パフォーマンススキーマインストゥルメント命名規則」に説明するように、インストゥルメント名には複数の部分があり、階層を形成することがあります。インストゥルメントの実行から生成されるイベントには、インストゥルメント NAME 値から取得される EVENT_NAME 値があります。(イベントには実際には「名前」がありませんが、これによって、イベントをインストゥルメントに関連付ける方法を提供します。)

- ENABLED

  インストゥルメントが有効にされているかどうか。このカラムは変更できます。無効にされたインストゥルメントはイベントを生成しません。

- TIMED

  インストゥルメントの時間が測定されるかどうか。このカラムは変更できます。

  有効にされたインストゥルメントの時間が測定されない場合、インストゥルメントコードは有効ですが、タイマーは有効ではありません。インストゥルメントによって生成されたイベントの TIMER_START、TIMER_END、および TIMER_WAIT タイマー値が NULL になります。これによって、サマリーテーブルの合計、最小、最大、および平均の時間値の計算時に、それらの値が無視されます。

22.9.2.4 setup_objects テーブル

setup_objects テーブルは、パフォーマンススキーマが特定のオブジェクトをモニターするかどうかを制御します。このテーブルはデフォルトで 100 行の最大サイズになります。テーブルサイズを変更するには、サーバー起動時に performance_schema_setup_objects_size システム変数を変更します。

初期 setup_objects の内容は次のように見えます。

```
mysql> SELECT * FROM setup_objects;
+-------------+--------------------+-------------+---------+-------+
| OBJECT_TYPE | OBJECT_SCHEMA      | OBJECT_NAME | ENABLED | TIMED |
+-------------+--------------------+-------------+---------+-------+
| TABLE       | mysql              | %           | NO      | NO    |
| TABLE       | performance_schema | %           | NO      | NO    |
| TABLE       | information_schema | %           | NO      | NO    |
| TABLE       | %                  | %           | YES     | YES   |
+-------------+--------------------+-------------+---------+-------+
```

setup_objects テーブルへの変更はただちにオブジェクトモニタリングに影響します。

setup_objects に示されているオブジェクトの種類では、パフォーマンススキーマはそれらのモニター方法にテーブルを使用します。オブジェクトの一致は OBJECT_SCHEMA および OBJECT_NAME カラムに基づきます。一致のないオブジェクトはモニターされません。

デフォルトのオブジェクト構成の効果は、mysql、INFORMATION_SCHEMA、および performance_schema データベースのテーブルを除くすべてのテーブルをインストゥルメントすることです。(INFORMATION_SCHEMA

データベース内のテーブルは、setup_objects の内容に関係なくインストゥルメントされず、information_schema.% の行は単にこのデフォルトを明示します。)

パフォーマンススキーマは、setup_objects の一致をチェックする場合、まずより詳細な一致を見つけようとします。たとえば、テーブル db1.t1 では、'db1' と 't1'、次に 'db1' と '%'、次に '%' と '%' の一致を検索します。さまざまな一致する setup_objects 行はさまざまな ENABLED 値と TIMED 値を持つ可能性があるため、一致が発生する順序が重要です。

テーブルへの INSERT または DELETE 権限を持つユーザーが、setup_objects に行を挿入したり、削除したりできます。既存の行では、テーブルへの UPDATE 権限を持つユーザーによって、ENABLED および TIMED カラムのみを変更できます。

イベントフィルタリングにおける setup_objects テーブルの役割の詳細については、セクション22.2.3.3「イベントの事前フィルタリング」を参照してください。

setup_objects テーブルにはこれらのカラムがあります。

- OBJECT_TYPE

  インストゥルメントするオブジェクトの種類。現在、これは常に 'TABLE' (ベーステーブル) です。

  TABLE フィルタリングはテーブル I/O イベント (wait/io/table/sql/handler インストゥルメント) およびテーブルロックイベント (wait/lock/table/sql/handler インストゥルメント) に影響します。

- OBJECT_SCHEMA

  オブジェクトを格納するスキーマ。これはリテラル名、または「任意のスキーマ」を意味する '%' であるべきです。

- OBJECT_NAME

  インストゥルメントされたオブジェクトの名前。これはリテラル名、または「任意のオブジェクト」を意味する '%' であるべきです。

- ENABLED

  オブジェクトのイベントがインストゥルメントされるかどうか。このカラムは変更できます。

  このカラムは、MySQL 5.6.3 で追加されました。それが存在しない初期バージョンの場合、パフォーマンススキーマはテーブル内の一部の行に一致するオブジェクトに対してのみモニタリングを有効にします。一致しないオブジェクトに対しては、モニタリングは暗黙的に無効にされます。

- TIMED

  オブジェクトのイベントの時間が測定されるかどうか。このカラムは変更できます。

22.9.2.5 setup_timers テーブル

setup_timers テーブルには現在選択されているイベントタイマーが表示されます。

```
mysql> SELECT * FROM setup_timers;
+-----------+-------------+
| NAME      | TIMER_NAME  |
+-----------+-------------+
| idle      | MICROSECOND |
| wait      | CYCLE       |
| stage     | NANOSECOND  |
| statement | NANOSECOND  |
+-----------+-------------+
```

setup_timers.TIMER_NAME 値を変更して、異なるタイマーを選択できます。値は performance_timers.TIMER_NAME カラム内の任意の値にすることができます。イベントタイミングがどのように行われるかの説明については、セクション22.2.3.1「パフォーマンススキーマイベントタイミング」を参照してください。

setup_timers テーブルへの変更はただちにモニタリングに影響します。すでに進行中のイベントは、開始時間に元のタイマーを使用し、終了時間に新しいタイマーを使用する可能性があるため、予測不可能な結果に至る場合があります。タイマーを変更した場合、TRUNCATE TABLE を使用して、パフォーマンススキーマの統計をリセットする必要がある場合があります。

setup_timers テーブルにはこれらのカラムがあります。

- NAME

  タイマーが使用されるインストゥルメントの種類。

- TIMER_NAME

  インストゥルメントの種類に適用されるタイマー。このカラムは変更できます。

22.9.3 パフォーマンススキーマインスタンステーブル

インスタンステーブルは、インストゥルメントされたオブジェクトの種類を記述します。それらは、イベント名と説明のメモまたはステータス情報を提供します。

- cond_instances: 条件同期オブジェクトインスタンス
- file_instances: ファイルインスタンス
- mutex_instances: 相互排他ロック同期オブジェクトインスタンス
- rwlock_instances: ロック同期オブジェクトインスタンス
- socket_instances: アクティブな接続インスタンス

これらのテーブルはインストゥルメントされた同期オブジェクト、ファイル、および接続を一覧表示します。3種類の同期オブジェクト cond、mutex、および rwlock があります。各インスタンステーブルには、各行に関連付けられているインストゥルメントを示す EVENT_NAME または NAME カラムがあります。セクション22.4「パフォーマンススキーマインストゥルメント命名規則」に説明するように、インストゥルメント名には複数の部分があり、階層を形成することがあります。

mutex_instances.LOCKED_BY_THREAD_ID および rwlock_instances.WRITE_LOCKED_BY_THREAD_ID カラムは、パフォーマンスボトルネックまたはデッドロックの調査にきわめて重要です。この目的でそれらを使用する方法の例については、セクション22.15「問題を診断するためのパフォーマンススキーマの使用」を参照してください

22.9.3.1 cond_instances テーブル

cond_instances テーブルは、サーバーの実行中にパフォーマンススキーマによって確認されるすべての条件を一覧表示します。条件は、この条件を待機しているスレッドが作業を再開できるように、特定のイベントが発生したことを伝えるために、コードで使用される同期メカニズムです。

スレッドが何かの発生を待機している場合、条件名は、スレッドが待機しているものを示しますが、ほかのどのスレッドが条件を発生させるかを伝えるための直接の方法はありません。

cond_instances テーブルにはこれらのカラムがあります。

- NAME

  条件に関連付けられているインストゥルメント名。

- OBJECT_INSTANCE_BEGIN

  インストゥルメントされた条件のメモリー内のアドレス。

22.9.3.2 file_instances テーブル

file_instances テーブルは、ファイル I/O インストゥルメンテーションの実行中にパフォーマンススキーマによって確認されるすべてのファイルを一覧表示します。ディスク上のファイルが開かれたことがない場合、file_instances に含まれません。ファイルがディスクから削除されると、file_instances テーブルからも削除されます。

file_instances テーブルにはこれらのカラムがあります。

- FILE_NAME

  ファイル名。

- EVENT_NAME

  ファイルに関連付けられているインストゥルメント名。

- OPEN_COUNT

  ファイルへのオープンハンドルのカウント。ファイルが開かれ、閉じられた場合、それは 1 回開かれていますが、OPEN_COUNT は 0 になります。サーバーによって現在開かれているすべてのファイルを一覧表示するには、WHERE OPEN_COUNT > 0 を使用します。

22.9.3.3 mutex_instances テーブル

mutex_instances テーブルは、サーバーの実行中にパフォーマンススキーマによって確認されるすべての相互排他ロックを一覧表示します。相互排他ロックは、特定の時間に 1 つだけのスレッドが特定の共通リソースにアクセスできるようにする、コードで使用される同期メカニズムです。リソースは相互排他ロックによって「保護されている」と呼ばれます。

サーバーで実行している 2 つのスレッド (たとえば、クエリーを同時に実行している 2 つのユーザーセッション) が同じリソース (ファイル、バッファー、データの一部) にアクセスする必要がある場合、これらの 2 つのスレッドは互いに競争するため、相互排他ロックのロックを取得する最初のクエリーによって、ほかのクエリーは最初のクエリーが終了し、相互排他ロックを解除するまで待たされます。

相互排他ロックの保持中に実行される作業は「クリティカルセクション」にあると呼ばれ、複数のクエリーがこのクリティカルセクションを連続して (一度に 1 つずつ) 実行するため、これは潜在的なボトルネックになります。

mutex_instances テーブルにはこれらのカラムがあります。

- NAME

  相互排他ロックに関連付けられているインストゥルメント名。

- OBJECT_INSTANCE_BEGIN

  インストゥルメントされた相互排他ロックのメモリー内のアドレス。

- LOCKED_BY_THREAD_ID

  スレッドが現在相互排他ロックされている場合、LOCKED_BY_THREAD_ID はロックしているスレッドの THREAD_ID になり、そうでない場合、それは NULL になります。

コードにインストゥルメントされた各相互排他ロックについて、パフォーマンススキーマは次の情報を提供します。

- setup_instruments テーブルは、プリフィクス wait/synch/mutex/ を付けて、インストゥルメンテーションポイントの名前を一覧表示します。
- 一部のコードで相互排他ロックが作成されると、行が mutex_instances テーブルに追加されます。OBJECT_INSTANCE_BEGIN カラムは相互排他ロックを一意に識別するプロパティーです。
- スレッドが相互排他ロックのロックを試みた場合、events_waits_current テーブルにそのスレッドの行が表示され、それが相互排他ロックを待機していることが示され (EVENT_NAME カラム内)、待機されている相互排他ロックが示されます (OBJECT_INSTANCE_BEGIN カラム内)。
- スレッドが相互排他ロックのロックに成功した場合:
  - events_waits_current は相互排他ロックへの待機が完了したことを示します (TIMER_END および TIMER_WAIT カラム内)
  - 完了した待機イベントは events_waits_history および events_waits_history_long テーブルに追加されます。
  - mutex_instances は相互排他ロックがスレッドによって所有されるようになったことを示します (THREAD_ID カラム内)。
- スレッドが相互排他ロックのロックを解除すると、mutex_instances は相互排他ロックに所有者がいなくなったことを示します (THREAD_ID カラムが NULL になります)。
- 相互排他ロックオブジェクトが破棄されると、対応する行が mutex_instances から削除されます。

次の両方のテーブルに対してクエリーを実行することによって、モニタリングアプリケーションまたは DBA は相互排他ロックを伴うスレッド間のボトルネックやデッドロックを検出できます。

- events_waits_current、スレッドが待機している相互排他ロックを確認する場合

- mutex_instances、相互排他ロックを現在所有しているほかのスレッドを確認する場合

22.9.3.4 rwlock_instances テーブル

rwlock_instances テーブルは、サーバーの実行中にパフォーマンススキーマによって確認されるすべての rwlock インスタンス (読み取り書き込みロック) を一覧表示します。rwlock は、特定の時間にそのスレッドが、次の特定のルールに従って、一部の共通リソースにアクセスできるようにするために、コードで使用される同期メカニズムです。リソースは rwlock によって「保護されている」と呼ばれます。アクセスは共有されている (多くのスレッドが同時に読み取りロックを持つことができる) かまたは排他的 (特定の時間に 1 つのスレッドだけが書き込みロックを持つことができる) のいずれかです。

ロックをリクエストしているスレッドの数、およびリクエストされているロックの性質に応じて、アクセスが共有モードで許可されるか、排他モードで許可されるか、またはまったく許可されないかのいずれかになる可能性があり、まずほかのスレッドが終了するのを待機します。

rwlock_instances テーブルにはこれらのカラムがあります。

- NAME

  ロックに関連付けられているインストゥルメント名。

- OBJECT_INSTANCE_BEGIN

  インストゥルメントされたロックのメモリー内のアドレス。

- WRITE_LOCKED_BY_THREAD_ID

  スレッドが現在排他 (書き込み) モードでロックされた rwlock を持つ場合、WRITE_LOCKED_BY_THREAD_ID はロックしているスレッドの THREAD_ID になり、そうでない場合、それは NULL になります。

- READ_LOCKED_BY_COUNT

  スレッドが現在共有 (読み取り) モードでロックされた rwlock を持つ場合、READ_LOCKED_BY_COUNT が 1 増分されます。これはカウンタのみであるため、読み取りロックを保持するスレッドを見つけるためにそれを直接使用することはできませんが、rwlock に対して読み取りの競合があるかどうかを確認し、現在アクティブなリーダー数を確認するために使用することができます。

次の両方のテーブルに対してクエリーを実行することによって、モニタリングアプリケーションまたは DBA はロックを伴うスレッド間の何らかのボトルネックやデッドロックを検出できます。

- events_waits_current、スレッドが待機している rwlock を確認する場合
- rwlock_instances、rwlock を現在所有しているほかのスレッドを確認する場合

制限があります。rwlock_instances は、書き込みロックを保持しているスレッドの識別にのみ使用できますが、読み取りロックを保持しているスレッドの識別には使用できません。

22.9.3.5 socket_instances テーブル

socket_instances テーブルは MySQL サーバーへのアクティブな接続のリアルタイムスナップショットを提供します。テーブルには、TCP/IP または Unix ソケットファイル接続ごとに 1 行含まれます。このテーブルで使用可能な情報は、サーバーへのアクティブな接続のリアルタイムスナップショットを提供します。(ソケット操作や送受信されたバイト数などのネットワークアクティビティーを含む、追加の情報はソケットサマリーテーブルで入手できます。セクション22.9.9.8「ソケットサマリーテーブル」を参照してください)。

```
mysql> SELECT * FROM socket_instances\G
*************************** 1. row ***************************
        EVENT_NAME: wait/io/socket/sql/server_unix_socket
OBJECT_INSTANCE_BEGIN: 4316619408
         THREAD_ID: 1
         SOCKET_ID: 16
                IP:
              PORT: 0
             STATE: ACTIVE
*************************** 2. row ***************************
        EVENT_NAME: wait/io/socket/sql/client_connection
OBJECT_INSTANCE_BEGIN: 4316644608
         THREAD_ID: 21
         SOCKET_ID: 39
                IP: 127.0.0.1
              PORT: 55233
             STATE: ACTIVE
```

```
*************************** 3. row ***************************
        EVENT_NAME: wait/io/socket/sql/server_tcpip_socket
OBJECT_INSTANCE_BEGIN: 4316699040
          THREAD_ID: 1
          SOCKET_ID: 14
                 IP: 0.0.0.0
               PORT: 50603
              STATE: ACTIVE
```

ソケットインストゥルメントは形式 wait/io/socket/sql/socket_type の名前を持ち、次のように使用されます。

1. サーバーには、それがサポートする各ネットワークプロトコルの待機ソケットがあります。TCP/IP または Unix ソケットファイル接続の待機ソケットに関連付けられているインストゥルメントは、それぞれ server_tcpip_socket または server_unix_socket の socket_type 値を持ちます。
2. 待機ソケットが接続を検出すると、サーバーは接続を、個別のスレッドによって管理される新しいソケットに転送します。新しい接続スレッドのインストゥルメントは、client_connection の socket_type 値を持ちます。
3. 接続が終了すると、socket_instances 内のそれに対応する行が削除されます。

socket_instances テーブルにはこれらのカラムがあります。

- EVENT_NAME

  イベントを生成した wait/io/socket/* インストゥルメントの名前。これは setup_instruments テーブルからの NAME 値です。セクション22.4「パフォーマンススキーマインストゥルメント命名規則」に説明するように、インストゥルメント名には複数の部分があり、階層を形成することがあります。

- OBJECT_INSTANCE_BEGIN

  このカラムは一意にソケットを識別します。この値はメモリー内のオブジェクトのアドレスです。

- THREAD_ID

  サーバーによって割り当てられた内部スレッド識別子。各ソケットは単一のスレッドによって管理されるため、各ソケットはスレッドにマップでき、スレッドはサーバープロセスにマップできます。

- SOCKET_ID

  ソケットに割り当てられている内部ファイルハンドル。

- IP

  クライアントの IP アドレス。この値は IPv4 または IPv6 アドレスのいずれか、または Unix ソケットファイル接続を示すブランクになります。

- PORT

  0 から 65535 の範囲の TCP/IP ポート番号。

- STATE

  IDLE または ACTIVE のいずれかのソケットステータス。アクティブなソケットの待機時間は、対応するソケットインストゥルメントを使用して追跡されます。アイドルソケットの待機時間は、idle インストゥルメントを使用して追跡されます。

  ソケットはクライアントからのリクエストを待機している場合、アイドルになります。ソケットがアイドルになると、ソケットを追跡している socket_instances 内のイベント行が ACTIVE のステータスから IDLE に切り替わります。EVENT_NAME 値は wait/io/socket/* のままになりますが、インストゥルメントのタイミングは一時停止されます。代わりに、idle の EVENT_NAME 値で events_waits_current テーブルにイベントが生成されます。

  次のリクエストを受信すると、idle イベントが終了し、ソケットインスタンスが IDLE から ACTIVE に切り替わり、ソケットインストゥルメントのタイミングが再開します。

IP:PORT カラムの組み合わせの値は接続を識別します。この組み合わせの値は、ソケットイベントの発生元の接続を識別するために、events_waits_xxx テーブルの OBJECT_NAME カラムで使用されます。

- Unix ドメインリスナーソケット (server_unix_socket) の場合、ポートは 0 で IP は '' です。
- Unix ドメインリスナー経由のクライアント接続 (client_connection) の場合、ポートは 0 で IP は '' です。

- TCP/IP サーバーリスナーソケット (server_tcpip_socket) の場合、ポートは常にマスターポート (たとえば 3306) で、IP は常に 0.0.0.0 です。
- TCP/IP リスナー経由のクライアント接続 (client_connection) の場合、ポートはサーバーが割り当てたものになりますが、0 にはなりません。IP は発信元ホストの IP (ローカルホストの場合 127.0.0.1 または ::1) です

socket_instances テーブルは MySQL 5.6.3 で追加されました。

22.9.4 パフォーマンススキーマ待機イベントテーブル

これらのテーブルは待機イベントを格納します。

- events_waits_current: 現在の待機イベント
- events_waits_history: 各スレッドの最新の待機イベント
- events_waits_history_long: 全体の最新の待機イベント

次のセクションでそれらのテーブルについて説明します。待機イベントに関する情報を集計するサマリーテーブルもあります。セクション22.9.9.1「イベント待機サマリーテーブル」を参照してください。

待機イベント構成

待機イベントの収集を有効にするには、関連インストゥルメントとコンシューマを有効にします。

setup_instruments テーブルには、wait で始まる名前を持つインストゥルメントが格納されます。例:

```
mysql> SELECT * FROM setup_instruments WHERE NAME LIKE 'wait/io/file/innodb%';
+---------------------------------------+---------+-------+
| NAME                                  | ENABLED | TIMED |
+---------------------------------------+---------+-------+
| wait/io/file/innodb/innodb_data_file | YES     | YES   |
| wait/io/file/innodb/innodb_log_file  | YES     | YES   |
| wait/io/file/innodb/innodb_temp_file | YES     | YES   |
+---------------------------------------+---------+-------+
mysql> SELECT * FROM setup_instruments WHERE NAME LIKE 'wait/io/socket/%';
+---------------------------------------+---------+-------+
| NAME                                  | ENABLED | TIMED |
+---------------------------------------+---------+-------+
| wait/io/socket/sql/server_tcpip_socket | NO      | NO    |
| wait/io/socket/sql/server_unix_socket  | NO      | NO    |
| wait/io/socket/sql/client_connection   | NO      | NO    |
+---------------------------------------+---------+-------+
```

待機イベントの収集を変更するには、関連インストゥルメントの ENABLED および TIMING カラムを変更します。例:

```
mysql> UPDATE setup_instruments SET ENABLED = 'YES', TIMED = 'YES'
    -> WHERE NAME LIKE 'wait/io/socket/sql/%';
```

setup_consumers テーブルには現在および最近の待機イベントテーブル名に対応する名前を持つコンシューマ値が格納されます。これらのコンシューマは待機イベントのコレクションをフィルタ処理するために使用できます。待機コンシューマはデフォルトで無効にされています。

```
mysql> SELECT * FROM setup_consumers WHERE NAME LIKE '%waits%';
+--------------------------+---------+
| NAME                     | ENABLED |
+--------------------------+---------+
| events_waits_current     | NO      |
| events_waits_history     | NO      |
| events_waits_history_long | NO      |
+--------------------------+---------+
```

すべての待機コンシューマを有効にするには、次を実行します。

```
mysql> UPDATE setup_consumers SET ENABLED = 'YES'
    -> WHERE NAME LIKE '%waits%';
```

setup_timers テーブルには、待機イベントのタイミングの単位を示す wait の NAME 値のある行が格納されます。デフォルトの単位は CYCLE です。

```
mysql> SELECT * FROM setup_timers WHERE NAME = 'wait';
+------+------------+
| NAME | TIMER_NAME |
+------+------------+
```

```
| wait | CYCLE     |
+------+-----------+
```

タイミングの単位を変更するには、TIMER_NAME 値を変更します。

```
mysql> UPDATE setup_timers SET TIMER_NAME = 'NANOSECOND'
    -> WHERE NAME = 'wait';
```

イベント収集の構成に関する追加情報については、セクション22.2「パフォーマンススキーマ構成」を参照してください。

22.9.4.1 events_waits_current テーブル

events_waits_current テーブルには、スレッドの最新のモニター対象待機イベントの現在のステータスを示すスレッドごとに 1 行で現在の待機イベントが格納されます。

events_waits_current テーブルは TRUNCATE TABLE で切り捨てることができます。

待機イベント行を格納するテーブルのうち、events_waits_current はもっとも基本的です。待機イベント行を格納するほかのテーブルは論理的に現在のイベントから派生します。たとえば、events_waits_history および events_waits_history_long テーブルは固定の行数以下の最新の待機イベントのコレクションです。

待機イベント収集の構成については、セクション22.9.4「パフォーマンススキーマ待機イベントテーブル」を参照してください。

events_waits_current テーブルにはこれらのカラムがあります。

- THREAD_ID、EVENT_ID

  イベントに関連付けられたスレッドとイベントの起動時のスレッドの現在のイベント番号。一緒に取得された THREAD_ID および EVENT_ID 値は、行を一意に識別する主キーを形成します。2 つの行が同じ値のペアを持つことはありません。

- END_EVENT_ID

  このカラムは、イベントの起動時に NULL に設定され、イベントの終了時にスレッドの現在のイベント番号に更新されます。このカラムは、MySQL 5.6.4 で追加されました。

- EVENT_NAME

  イベントを生成したインストゥルメントの名前。これは setup_instruments テーブルからの NAME 値です。セクション22.4「パフォーマンススキーマインストゥルメント命名規則」に説明するように、インストゥルメント名には複数の部分があり、階層を形成することがあります。

- SOURCE

  イベントを生成した、インストゥルメントされたコードを格納するソースファイルの名前と、インストゥルメンテーションが行われたファイルの行番号。これにより、ソースをチェックして、コードに含まれるものを正確に判断することができます。たとえば、相互排他ロックまたはロックがブロックされた場合、これが発生するコンテキストをチェックできます。

- TIMER_START、TIMER_END、TIMER_WAIT

  イベントのタイミング情報。これらの値の単位はピコ秒 (秒の 1 兆分の 1) です。TIMER_START および TIMER_END 値は、イベントのタイミングが開始されたときと終了したときを示します。TIMER_WAIT はイベントの経過時間 (期間) です。

  イベントが終了していない場合、TIMER_END と TIMER_WAIT は NULL です。

  イベントが TIMED = NO のインストゥルメントから生成されている場合、タイミング情報は収集されず、TIMER_START、TIMER_END、および TIMER_WAIT はすべて NULL になります。

  イベント時間の単位としてのピコ秒および時間値に影響する要因については、セクション22.2.3.1「パフォーマンススキーマイベントタイミング」を参照してください。

- SPINS

  相互排他ロックの場合、スピンラウンドの数。値が NULL の場合、コードはスピンラウンドを使用しないか、スピニングがインストゥルメントされません。

- OBJECT_SCHEMA、OBJECT_NAME、OBJECT_TYPE、OBJECT_INSTANCE_BEGIN

これらのカラムは「作用している」オブジェクトを識別します。その意味は、オブジェクトの種類によって異なります。

同期オブジェクト (cond、mutex、および rwlock) の場合:

- OBJECT_SCHEMA、OBJECT_NAME、および OBJECT_TYPE は NULL です。
- OBJECT_INSTANCE_BEGIN はメモリー内の同期オブジェクトのアドレスです。

ファイル I/O オブジェクトの場合:

- OBJECT_SCHEMA は NULL です。
- OBJECT_NAME はファイル名です。
- OBJECT_TYPE は FILE です。
- OBJECT_INSTANCE_BEGIN はメモリー内のアドレスです。

ソケットオブジェクトの場合:

- OBJECT_NAME はソケットの IP:PORT 値です。
- OBJECT_INSTANCE_BEGIN はメモリー内のアドレスです。

テーブル I/O オブジェクトの場合:

- OBJECT_SCHEMA はテーブルを格納するスキーマの名前です。
- OBJECT_NAME はテーブル名です。
- OBJECT_TYPE は永続的ベーステーブルの TABLE または一時テーブルの TEMPORARY TABLE です。
- OBJECT_INSTANCE_BEGIN はメモリー内のアドレスです。

OBJECT_INSTANCE_BEGIN 値自体には、さまざまな値がさまざまなオブジェクトを示すことを除いて、意味がありません。OBJECT_INSTANCE_BEGIN はデバッグに使用できます。たとえば、それを GROUP BY OBJECT_INSTANCE_BEGIN で使用して、1,000 相互排他ロック (つまり、データの 1,000 ページまたはブロックを保護する) の負荷が均等に広がっているか、または少数のボトルネックだけに関わっているかを確認できます。これにより、ログファイルやほかのデバッグまたはパフォーマンスツールで同じオブジェクトアドレスが見られた場合に、情報のほかのソースと関連付けることができます。

- INDEX_NAME

  使用されるインデックスの名前。PRIMARY はテーブルプライマリインデックスを示します。NULL はインデックスが使用されなかったことを意味します。

- NESTING_EVENT_ID

  このイベントが中にネストされているイベントの EVENT_ID 値。MySQL 5.6.3 より前では、このカラムは常に NULL です。

- NESTING_EVENT_TYPE

  ネストしているイベントの種類。値は STATEMENT、STAGE、または WAIT です。このカラムは、MySQL 5.6.3 で追加されました。

- OPERATION

  lock、read、または write などの実行される操作の種類。

- NUMBER_OF_BYTES

  操作によって読み取りまたは書き込まれるバイト数。テーブル I/O 待機 (wait/io/table/sql/handler インストゥルメントのイベント) の場合、NUMBER_OF_BYTES は NULL になります。

- FLAGS

  将来使用するために予約されています。

22.9.4.2 events_waits_history テーブル

events_waits_history テーブルには、スレッドごとの最新の N 待機イベントが格納されます。N の値はサーバー起動時に自動サイズ設定されます。テーブルサイズを明示的に設定するには、サーバー起動時に performance_schema_events_waits_history_size システム変数を設定します。待機イベントはそれらが終了するまでテーブルに追加されません。新しいイベントが追加されたときに、テーブルがいっぱいである場合、古いイベントが破棄されます。

events_waits_history テーブルは events_waits_current と同じ構造を持ちます。セクション 22.9.4.1「events_waits_current テーブル」を参照してください。

events_waits_history テーブルは TRUNCATE TABLE で切り捨てることができます。

待機イベント収集の構成については、セクション22.9.4「パフォーマンススキーマ待機イベントテーブル」を参照してください。

22.9.4.3 events_waits_history_long テーブル

events_waits_history_long テーブルには、最新の N 待機イベントが格納されます。N の値はサーバー起動時に自動サイズ設定されます。テーブルサイズを明示的に設定するには、サーバー起動時に performance_schema_events_waits_history_long_size システム変数を設定します。待機イベントはそれらが終了するまでテーブルに追加されません。新しいイベントが追加されたときに、テーブルがいっぱいである場合、古いイベントが破棄されます。

events_waits_history_long テーブルは events_waits_current と同じ構造を持ちます。セクション 22.9.4.1「events_waits_current テーブル」を参照してください。

events_waits_history_long テーブルは TRUNCATE TABLE で切り捨てることができます。

待機イベント収集の構成については、セクション22.9.4「パフォーマンススキーマ待機イベントテーブル」を参照してください。

22.9.5 パフォーマンススキーマステージイベントテーブル

MySQL 5.6.3 現在、パフォーマンススキーマは、ステートメントの解析、テーブルのオープン、または filesort 操作の実行などのステートメント実行プロセス中のステップであるステージをインストゥルメントします。ステージは SHOW PROCESSLIST によって表示されるか、または INFORMATION_SCHEMA.PROCESSLIST テーブルに表示されるスレッドの状態に対応します。ステージは、状態値が変化したときに開始および終了します。

イベント階層内で、待機イベントはステージイベント内にネストし、ステージイベントはステートメントイベント内にネストします。

これらのテーブルはステージイベントを格納します。

- events_stages_current: 現在のステージイベント
- events_stages_history: 各スレッドの最新のステージイベント
- events_stages_history_long: 全体の最新のステージイベント

次のセクションでそれらのテーブルについて説明します。ステージイベントに関する情報を集計するサマリーテーブルもあります。セクション22.9.9.2「ステージサマリーテーブル」を参照してください。

ステージイベント構成

ステージイベントの収集を有効にするには、関連インストゥルメントとコンシューマを有効にします。

setup_instruments テーブルには、stage で始まる名前を持つインストゥルメントが格納されます。これらのインストゥルメントはデフォルトで無効にされています。例:

```
mysql> SELECT * FROM setup_instruments WHERE NAME RLIKE 'stage/sql/[a-c]';
+----------------------------------------------------+---------+-------+
| NAME                                               | ENABLED | TIMED |
+----------------------------------------------------+---------+-------+
| stage/sql/After create                             | NO      | NO    |
| stage/sql/allocating local table                   | NO      | NO    |
| stage/sql/altering table                           | NO      | NO    |
| stage/sql/committing alter table to storage engine | NO      | NO    |
| stage/sql/Changing master                          | NO      | NO    |
| stage/sql/Checking master version                  | NO      | NO    |
```

```
| stage/sql/checking permissions                    | NO      | NO    |
| stage/sql/checking privileges on cached query     | NO      | NO    |
| stage/sql/checking query cache for query          | NO      | NO    |
| stage/sql/cleaning up                             | NO      | NO    |
| stage/sql/closing tables                          | NO      | NO    |
| stage/sql/Connecting to master                    | NO      | NO    |
| stage/sql/converting HEAP to MyISAM               | NO      | NO    |
| stage/sql/Copying to group table                  | NO      | NO    |
| stage/sql/Copying to tmp table                    | NO      | NO    |
| stage/sql/copy to tmp table                       | NO      | NO    |
| stage/sql/Creating delayed handler                | NO      | NO    |
| stage/sql/Creating sort index                     | NO      | NO    |
| stage/sql/creating table                          | NO      | NO    |
| stage/sql/Creating tmp table                      | NO      | NO    |
+---------------------------------------------------+---------+-------+
```

ステージイベントの収集を変更するには、関連インストゥルメントの ENABLED および TIMING カラムを変更します。例:

```
mysql> UPDATE setup_instruments SET ENABLED = 'YES', TIMED = 'YES'
    -> WHERE NAME = 'stage/sql/altering table';
```

setup_consumers テーブルには現在および最近のステージイベントテーブル名に対応する名前を持つコンシューマ値が格納されます。これらのコンシューマはステージイベントのコレクションをフィルタ処理するために使用できます。ステージコンシューマはデフォルトで無効にされています。

```
mysql> SELECT * FROM setup_consumers WHERE NAME LIKE '%stages%';
+----------------------------+---------+
| NAME                       | ENABLED |
+----------------------------+---------+
| events_stages_current      | NO      |
| events_stages_history      | NO      |
| events_stages_history_long | NO      |
+----------------------------+---------+
```

すべてのステージコンシューマを有効にするには、次を実行します。

```
mysql> UPDATE setup_consumers SET ENABLED = 'YES'
    -> WHERE NAME LIKE '%stages%';
```

setup_timers テーブルには、ステージイベントのタイミングの単位を示す stage の NAME 値のある行が格納されます。デフォルトの単位は NANOSECOND です。

```
mysql> SELECT * FROM setup_timers WHERE NAME = 'stage';
+-------+-------------+
| NAME  | TIMER_NAME |
+-------+-------------+
| stage | NANOSECOND |
+-------+-------------+
```

タイミングの単位を変更するには、TIMER_NAME 値を変更します。

```
mysql> UPDATE setup_timers SET TIMER_NAME = 'MICROSECOND'
    -> WHERE NAME = 'stage';
```

イベント収集の構成に関する追加情報については、セクション22.2「パフォーマンススキーマ構成」を参照してください。

22.9.5.1 events_stages_current テーブル

events_stages_current テーブルには、スレッドの最新のモニター対象ステージイベントの現在のステータスを示すスレッドごとに 1 行で現在のステージイベントが格納されます。

events_stages_current テーブルは TRUNCATE TABLE で切り捨てることができます。

ステージイベント行を格納するテーブルのうち、events_stages_current はもっとも基本的です。ステージイベント行を格納するほかのテーブルは論理的に現在のイベントから派生します。たとえば、events_stages_history および events_stages_history_long テーブルは固定の行数以下の最新のステージイベントのコレクションです。

ステージイベント収集の構成については、セクション22.9.5「パフォーマンススキーマステージイベントテーブル」を参照してください。

events_stages_current テーブルにはこれらのカラムがあります。

- THREAD_ID、EVENT_ID

イベントに関連付けられたスレッドとイベントの起動時のスレッドの現在のイベント番号。一緒に取得された THREAD_ID および EVENT_ID 値は、行を一意に識別する主キーを形成します。2 つの行が同じ値のペアを持つことはありません。

- END_EVENT_ID

  このカラムは、イベントの起動時に NULL に設定され、イベントの終了時にスレッドの現在のイベント番号に更新されます。このカラムは、MySQL 5.6.4 で追加されました。

- EVENT_NAME

  イベントを生成したインストゥルメントの名前。これは setup_instruments テーブルからの NAME 値です。セクション22.4「パフォーマンススキーマインストゥルメント命名規則」に説明するように、インストゥルメント名には複数の部分があり、階層を形成することがあります。

- SOURCE

  イベントを生成した、インストゥルメントされたコードを格納するソースファイルの名前と、インストゥルメンテーションが行われたファイルの行番号。これにより、ソースをチェックして、コードに含まれるものを正確に判断することができます。

- TIMER_START、TIMER_END、TIMER_WAIT

  イベントのタイミング情報。これらの値の単位はピコ秒 (秒の 1 兆分の 1) です。TIMER_START および TIMER_END 値は、イベントのタイミングが開始されたときと終了したときを示します。TIMER_WAIT はイベントの経過時間 (期間) です。

  イベントが終了していない場合、TIMER_END と TIMER_WAIT は NULL です。

  イベントが TIMED = NO のインストゥルメントから生成されている場合、タイミング情報は収集されず、TIMER_START、TIMER_END、および TIMER_WAIT はすべて NULL になります。

  イベント時間の単位としてのピコ秒および時間値に影響する要因については、セクション22.2.3.1「パフォーマンススキーマイベントタイミング」を参照してください。

- NESTING_EVENT_ID

  このイベントが中にネストされているイベントの EVENT_ID 値。ステージイベントのネストしているイベントは通常ステートメントイベントです。

- NESTING_EVENT_TYPE

  ネストしているイベントの種類。値は STATEMENT、STAGE、または WAIT です。

events_stages_current テーブルは MySQL 5.6.3 で追加されました。

22.9.5.2 events_stages_history テーブル

events_stages_history テーブルにはスレッドごとに最新の N ステージイベントが格納されます。N の値はサーバー起動時に自動サイズ設定されます。テーブルサイズを明示的に設定するには、サーバー起動時に performance_schema_events_stages_history_size システム変数を設定します。ステージイベントは終了するまでテーブルに追加されません。新しいイベントが追加されたときに、テーブルがいっぱいである場合、古いイベントが破棄されます。

events_stages_history テーブルは events_stages_current と同じ構造を持ちます。セクション22.9.5.1「events_stages_current テーブル」を参照してください。

events_stages_history テーブルは TRUNCATE TABLE で切り捨てることができます。

events_stages_history テーブルは MySQL 5.6.3 で追加されました。

ステージイベント収集の構成については、セクション22.9.5「パフォーマンススキーマステージイベントテーブル」を参照してください。

22.9.5.3 events_stages_history_long テーブル

events_stages_history_long テーブルには最新の N ステージイベントが格納されます。N の値はサーバー起動時に自動サイズ設定されます。テーブルサイズを明示的に設定するには、サーバー起動時に performance_schema_events_stages_history_long_size システム変数を設定します。ステージイベントは終了す

るまでテーブルに追加されません。新しいイベントが追加されたときに、テーブルがいっぱいである場合、古いイベントが破棄されます。

events_stages_history_long テーブルは events_stages_current と同じ構造を持ちます。セクション22.9.5.1「events_stages_current テーブル」を参照してください。

events_stages_history_long テーブルは TRUNCATE TABLE で切り捨てることができます。

events_stages_history_long テーブルは MySQL 5.6.3 で追加されました。

ステージイベント収集の構成については、セクション22.9.5「パフォーマンススキーマステージイベントテーブル」を参照してください。

22.9.6 パフォーマンススキーマステートメントイベントテーブル

MySQL 5.6.3 現在、パフォーマンススキーマはステートメント実行をインストゥルメントします。ステートイベントイベントはイベント階層の高いレベルで発生します。待機イベントはステージイベント内にネストし、ステージイベントはステートメントイベント内にネストします。

これらのテーブルはステートメントイベントを格納します。

- events_statements_current: 現在のステートメントイベント
- events_statements_history: 各スレッドの最新のステートメントイベント
- events_statements_history_long: 全体の最新のステートメントイベント

次のセクションでそれらのテーブルについて説明します。ステートメントイベントに関する情報を集計するサマリーテーブルもあります。セクション22.9.9.3「ステートメントサマリーテーブル」を参照してください。

ステートメントイベント構成

ステートメントイベントの収集を有効にするには、関連インストゥルメントとコンシューマを有効にします。

setup_instruments テーブルには、statement で始まる名前を持つインストゥルメントが格納されます。これらのインストゥルメントはデフォルトで有効にされています。

```
mysql> SELECT * FROM setup_instruments WHERE NAME LIKE 'statement/%';
+----------------------------------------+---------+-------+
| NAME                                   | ENABLED | TIMED |
+----------------------------------------+---------+-------+
| statement/sql/select                   | YES     | YES   |
| statement/sql/create_table             | YES     | YES   |
| statement/sql/create_index             | YES     | YES   |
...
| statement/sp/stmt                      | YES     | YES   |
| statement/sp/set                       | YES     | YES   |
| statement/sp/set_trigger_field         | YES     | YES   |
| statement/scheduler/event              | YES     | YES   |
| statement/com/Sleep                    | YES     | YES   |
| statement/com/Quit                     | YES     | YES   |
| statement/com/Init DB                  | YES     | YES   |
...
| statement/abstract/Query               | YES     | YES   |
| statement/abstract/new_packet          | YES     | YES   |
| statement/abstract/relay_log           | YES     | YES   |
+----------------------------------------+---------+-------+
```

ステートメントイベントの収集を変更するには、関連インストゥルメントの ENABLED および TIMING カラムを変更します。例:

```
mysql> UPDATE setup_instruments SET ENABLED = 'NO'
    -> WHERE NAME LIKE 'statement/com/%';
```

setup_consumers テーブルには現在および最近のステートメントイベントテーブル名に対応する名前を持つコンシューマ値とステートメントダイジェストコンシューマが格納されます。これらのコンシューマはステートメントイベントのコレクションとステートメントダイジェストをフィルタ処理するために使用できます。events_statements_current と statements_digest のみがデフォルトで有効にされています。

```
mysql> SELECT * FROM setup_consumers WHERE NAME LIKE '%statements%';
+--------------------------------+---------+
| NAME                           | ENABLED |
+--------------------------------+---------+
| events_statements_current      | YES     |
```

```
| events_statements_history      | NO      |
| events_statements_history_long | NO      |
| statements_digest              | YES     |
+--------------------------------+---------+
```

すべてのステートメントコンシューマを有効にするには、次を実行します。

```
mysql> UPDATE setup_consumers SET ENABLED = 'YES'
    -> WHERE NAME LIKE '%statements%';
```

setup_timers テーブルには、ステートメントイベントのタイミングの単位を示す statement の NAME 値のある行が格納されます。デフォルトの単位は NANOSECOND です。

```
mysql> SELECT * FROM setup_timers WHERE NAME = 'statement';
+-----------+-------------+
| NAME      | TIMER_NAME |
+-----------+-------------+
| statement | NANOSECOND |
+-----------+-------------+
```

タイミングの単位を変更するには、TIMER_NAME 値を変更します。

```
mysql> UPDATE setup_timers SET TIMER_NAME = 'MICROSECOND'
    -> WHERE NAME = 'statement';
```

イベント収集の構成に関する追加情報については、セクション22.2「パフォーマンススキーマ構成」を参照してください。

ステートメントモニタリング

ステートメントのモニタリングは、サーバーがスレッドに対してアクティビティーがリクエストされていることを確認した時点から、すべてのアクティビティーが終了した時点までに開始されます。一般に、これはサーバーがクライアントから最初のパケットを受け取ったときから、サーバーが応答の送信を終了したときまでを意味します。モニタリングは、トップレベルステートメントに対してのみ行われます。ストアドプログラムとサブクエリー内のステートメントは別々に見られません。

パフォーマンススキーマがリクエスト (サーバーコマンドまたは SQL ステートメント) をインストゥルメントする場合、最終的なインストゥルメント名に到達するまで、より一般的 (または「抽象的」) から、より具体的へと段階を追って進むインストゥルメント名を使用します。

最終インストゥルメント名はサーバーコマンドと SQL ステートメントに対応します。

- サーバーコマンドは mysql_com.h ヘッダーファイルに定義され、sql/sql_parse.cc で処理される COM_xxx codes に対応します。例は COM_PING と COM_QUIT です。コマンドのインストゥルメントは、statement/com/Ping や statement/com/Quit などの statement/com から始まる名前を持ちます。
- SQL ステートメントは DELETE FROM t1 または SELECT * FROM t2 などのテキストとして表されます。SQL ステートメントのインストゥルメントは、statement/sql/delete や statement/sql/select などの statement/sql から始まる名前を持ちます。

いくつかの最終インストゥルメント名はエラー処理に固有です。

- statement/com/Error は帯域外のサーバーによって受信されたメッセージから構成されます。これはサーバーが理解しないクライアントによって送信されたコマンドを検出するために使用できます。これは、構成が誤っているか、サーバーよりも新しい MySQL のバージョンを使用しているクライアントや、サーバーへの攻撃を試みているクライアントの識別などの目的で役に立つことがあります。
- statement/sql/error は解析に失敗した SQL ステートメントから構成されます。これはクライアントによって送信された不正な形式のクエリーを検出するために使用できます。解析に失敗するクエリーは、解析するが、実行中のエラーのために失敗するクエリーと異なります。たとえば、SELECT * FROM は不正な形式で、statement/sql/error インストゥルメントが使用されます。対照的に SELECT * は解析しますが、「表が指定されていません」エラーを伴って失敗します。この場合、statement/sql/select が使用され、ステートメントイベントにはエラーの性質を示す情報が含まれます。

リクエストはこれらの任意のソースから取得できます。

- リクエストをパケットとして送信するクライアントからのコマンドまたはステートメントリクエストとして
- レプリケーションスレーブ上のリレーログから読み取られたステートメント文字列として (MySQL 5.6.13 以降)

リクエストの詳細は最初は不明で、パフォーマンススキーマはリクエストのソースに依存する順序で、抽象から特定のインストゥルメント名に進みます。

クライアントから受信したリクエストの場合:

1. サーバーがソケットレベルで新しいパケットを検出すると、新しいステートメントが statement/abstract/new_packet の抽象インストゥルメント名で開始されます。
2. サーバーはパケット番号を読み取ると、受信したリクエストの種類について詳しく知り、パフォーマンススキーマがインストゥルメント名を絞り込みます。たとえば、リクエストが COM_PING パケットの場合、インストゥルメント名は statement/com/Ping になり、それが最終名になります。リクエストが COM_QUERY パケットの場合、それは SQL ステートメントに対応するが、特定のステートメントの種類ではないことがわかります。この場合、インストゥルメントはある抽象名から、やや具体的だが、まだ抽象名である statement/abstract/Query に変更され、リクエストはさらに分類する必要があります。
3. リクエストがステートメントである場合、ステートメントテキストが読み取られ、パーサーに提供されます。解析後、正確なステートメントの種類が認識されます。リクエストがたとえば INSERT ステートメントの場合、パフォーマンススキーマはインストゥルメント名を statement/abstract/Query から最終名である statement/sql/insert に絞り込みます。

レプリケーションスレーブ上のリレーログからステートメントとして読み取られるリクエストの場合:

1. リレーログ内のステートメントはテキストとして保存され、そのように読み取られます。ネットワークプロトコルはないため、statement/abstract/new_packet インストゥルメントは使用されません。代わりに、初期インストゥルメントは statement/abstract/relay_log になります。
2. ステートメントが解析されると、正確なステートメントの種類が認識されます。リクエストがたとえば INSERT ステートメントの場合、パフォーマンススキーマはインストゥルメント名を statement/abstract/Query から最終名である statement/sql/insert に絞り込みます。

先述の説明はステートメントベースのレプリケーションにのみ適用されます。行ベースのレプリケーションでは、行の変更を処理しながら、スレーブで実行されるテーブル I/O をインストゥルメントできますが、リレーログ内の行イベントは、個別のステートメントとして表示されません。

ステートメントに対して収集される統計の場合、各ステートメントの種類に使用される最終 statement/sql/* インストゥルメントを有効にするだけでは十分ではありません。抽象 statement/abstract/* インストゥルメントも有効にする必要があります。すべてのステートメントインストゥルメントがデフォルトで有効にされるため、これは通常問題にならないはずです。ただし、ステートメントインストゥルメントを選択して有効または無効にするアプリケーションは、抽象インストゥルメントを無効にすると、個々のステートメントインストゥルメントの統計収集も無効になることを考慮する必要があります。たとえば、INSERT ステートメントの統計を収集するには、statement/sql/insert を有効にする必要がありますが、statement/abstract/new_packet と statement/abstract/Query も有効にする必要があります。同様に、レプリケートされたステートメントをインストゥルメントするには、statement/abstract/relay_log が有効にされている必要があります。

ステートメントが最終ステートメント名として抽象インストゥルメントに分類されることはないため、statement/abstract/Query などの抽象インストゥルメントに対して統計は集計されません。

先述の説明の抽象インストゥルメント名は MySQL 5.6.15 現在です。5.6 より前では、それらの名前が決定されるまでに、いくらかの名前の変更がありました。

- MySQL 5.6.14 では statement/abstract/new_packet は statement/com/ で、MySQL 5.6.13 では statement/com/new_packet で、それ以前は statement/com/ でした。
- MySQL 5.6.15 より前では、statement/abstract/Query は statement/com/Query でした。
- MySQL 5.6.13 から 5.6.14 では statement/abstract/relay_log は statement/rpl/relay_log で、それより前は存在していませんでした。

22.9.6.1 events_statements_current テーブル

events_statements_current テーブルには、スレッドの最新のモニター対象ステートメントイベントの現在のステータスを示す、スレッドごとに 1 行で現在のステートメントイベントが格納されます。

events_statements_current テーブルは TRUNCATE TABLE で切り捨てることができます。

ステートメントイベント行を格納するテーブルのうち、events_statements_current はもっとも基本的です。ステートメントイベント行を格納するほかのテーブルは論理的に現在のイベントから派生します。たとえば、events_statements_history および events_statements_history_long テーブルは固定の行数以下の最新のステートメントイベントのコレクションです。

ステートメントイベント収集の構成については、セクション22.9.6「パフォーマンススキーマステートメントイベントテーブル」を参照してください。

events_statements_current テーブルにはこれらのカラムがあります。

- THREAD_ID、EVENT_ID

  イベントに関連付けられたスレッドとイベントの起動時のスレッドの現在のイベント番号。一緒に取得された THREAD_ID および EVENT_ID 値は、行を一意に識別する主キーを形成します。2 つの行が同じ値のペアを持つことはありません。

- END_EVENT_ID

  このカラムは、イベントの起動時に NULL に設定され、イベントの終了時にスレッドの現在のイベント番号に更新されます。このカラムは、MySQL 5.6.4 で追加されました。

- EVENT_NAME

  イベントが収集されたインストゥルメントの名前。これは setup_instruments テーブルからの NAME 値です。セクション22.4「パフォーマンススキーマインストゥルメント命名規則」に説明するように、インストゥルメント名には複数の部分があり、階層を形成することがあります。

  SQL ステートメントの場合、ステートメントが解析されるまで、EVENT_NAME 値は最初 statement/com/Query であり、その後セクション22.9.6「パフォーマンススキーマステートメントイベントテーブル」に説明するように、より適切な値に変更されます。

- SOURCE

  イベントを生成した、インストゥルメントされたコードを格納するソースファイルの名前と、インストゥルメンテーションが行われたファイルの行番号。これにより、ソースをチェックして、コードに含まれるものを正確に判断することができます。

- TIMER_START、TIMER_END、TIMER_WAIT

  イベントのタイミング情報。これらの値の単位はピコ秒 (秒の 1 兆分の 1) です。TIMER_START および TIMER_END 値は、イベントのタイミングが開始されたときと終了したときを示します。TIMER_WAIT はイベントの経過時間 (期間) です。

  イベントが終了していない場合、TIMER_END と TIMER_WAIT は NULL です。

  イベントが TIMED = NO のインストゥルメントから生成されている場合、タイミング情報は収集されず、TIMER_START、TIMER_END、および TIMER_WAIT はすべて NULL になります。

  イベント時間の単位としてのピコ秒および時間値に影響する要因については、セクション22.2.3.1「パフォーマンススキーマイベントタイミング」を参照してください。

- LOCK_TIME

  テーブルロックの待機に費やされた時間。この値はマイクロ秒で計算されますが、ほかのパフォーマンススキーマタイマーとの比較を容易にするため、ピコ秒に正規化されます。

- SQL_TEXT

  SQL ステートメントのテキスト。SQL ステートメントに関連付けられていないコマンドの場合、値は NULL です。

- DIGEST

  32 個の 16 進文字の文字列としてのステートメントダイジェスト MD5 値または statement_digest コンシューマが no の場合は NULL。ステートメントダイジェストの詳細については、セクション22.7「パフォーマンススキーマステートメントダイジェスト」を参照してください。このカラムは、MySQL 5.6.5 で追加されました。

- DIGEST_TEXT

  正規化されたステートメントダイジェストテキストまたは statement_digest コンシューマが no の場合は NULL。ステートメントダイジェストの詳細については、セクション22.7「パフォーマンススキーマステートメントダイジェスト」を参照してください。このカラムは、MySQL 5.6.5 で追加されました。

- CURRENT_SCHEMA

  ステートメントのデフォルトのデータベース、何もない場合は NULL。

- OBJECT_SCHEMA、OBJECT_NAME、OBJECT_TYPE

予約済み。現在は NULL。

- OBJECT_INSTANCE_BEGIN

  このカラムはステートメントを識別します。この値はメモリー内のオブジェクトのアドレスです。

- MYSQL_ERRNO

  ステートメント診断領域からのステートメントエラー番号。

- RETURNED_SQLSTATE

  ステートメント診断領域からのステートメント SQLSTATE 値。

- MESSAGE_TEXT

  ステートメント診断領域からのステートメントエラーメッセージ。

- ERRORS

  ステートメントにエラーが発生したかどうか。SQLSTATE 値が 00 (完了) または 01 (警告) から始まる場合、値は 0 です。SQLSTATE 値がほかの値の場合、値は 1 です。

- WARNINGS

  ステートメント診断領域からの警告数。

- ROWS_AFFECTED

  ステートメントに影響を受けた行数。「影響を受けた」の意味については、セクション 23.7.7.1「mysql_affected_rows()」を参照してください。

- ROWS_SENT

  ステートメントから返された行数。

- ROWS_EXAMINED

  ステートメント実行中にストレージエンジンから読み取られた行数。

- CREATED_TMP_DISK_TABLES

  Created_tmp_disk_tables ステータス変数と同様ですが、ステートメントに固有です。

- CREATED_TMP_TABLES

  Created_tmp_tables ステータス変数と同様ですが、ステートメントに固有です。

- SELECT_FULL_JOIN

  Select_full_join ステータス変数と同様ですが、ステートメントに固有です。

- SELECT_FULL_RANGE_JOIN

  Select_full_range_join ステータス変数と同様ですが、ステートメントに固有です。

- SELECT_RANGE

  Select_range ステータス変数と同様ですが、ステートメントに固有です。

- SELECT_RANGE_CHECK

  Select_range_check ステータス変数と同様ですが、ステートメントに固有です。

- SELECT_SCAN

  Select_scan ステータス変数と同様ですが、ステートメントに固有です。

- SORT_MERGE_PASSES

  Sort_merge_passes ステータス変数と同様ですが、ステートメントに固有です。

- SORT_RANGE

  Sort_range ステータス変数と同様ですが、ステートメントに固有です。

- SORT_ROWS

  Sort_rows ステータス変数と同様ですが、ステートメントに固有です。

- SORT_SCAN

  Sort_scan ステータス変数と同様ですが、ステートメントに固有です。

- NO_INDEX_USED

  ステートメントがインデックスを使用せずにテーブルスキャンを実行した場合は 1、そうでない場合は 0。

- NO_GOOD_INDEX_USED

  サーバーがステートメントに使用する適切なインデックスを見つけられなかった場合は 1、そうでない場合は 0。追加の情報については、セクション8.8.2「EXPLAIN 出力フォーマット」で、EXPLAIN 出力の Extra カラムの Range checked for each record 値の説明を参照してください。

- NESTING_EVENT_ID、NESTING_EVENT_TYPE

  予約済み。現在は NULL。

events_statements_current テーブルは MySQL 5.6.3 で追加されました。

22.9.6.2 events_statements_history テーブル

events_statements_history テーブルには、スレッドごとの最新の N ステートメントイベントが格納されます。N の値はサーバー起動時に自動サイズ設定されます。テーブルサイズを明示的に設定するには、サーバー起動時に performance_schema_events_statements_history_size システム変数を設定します。ステートメントイベントは終了するまでテーブルに追加されません。新しいイベントが追加されたときに、テーブルがいっぱいである場合、古いイベントが破棄されます。

events_statements_history テーブルは events_statements_current と同じ構造を持ちます。セクション 22.9.6.1「events_statements_current テーブル」を参照してください。

events_statements_history テーブルは TRUNCATE TABLE で切り捨てることができます。

events_statements_history テーブルは MySQL 5.6.3 で追加されました。

ステートメントイベント収集の構成については、セクション22.9.6「パフォーマンススキーマステートメントイベントテーブル」を参照してください。

22.9.6.3 events_statements_history_long テーブル

events_statements_history_long テーブルには、最新の N ステートメントイベントが格納されます。N の値はサーバー起動時に自動サイズ設定されます。テーブルサイズを明示的に設定するには、サーバー起動時に performance_schema_events_statements_history_long_size システム変数を設定します。ステートメントイベントは終了するまでテーブルに追加されません。新しいイベントが追加されたときに、テーブルがいっぱいである場合、古いイベントが破棄されます。

events_statements_history_long テーブルは events_statements_current と同じ構造を持ちます。セクション 22.9.6.1「events_statements_current テーブル」を参照してください。

events_statements_history_long テーブルは TRUNCATE TABLE で切り捨てることができます。

events_statements_history_long テーブルは MySQL 5.6.3 で追加されました。

ステートメントイベント収集の構成については、セクション22.9.6「パフォーマンススキーマステートメントイベントテーブル」を参照してください。

22.9.7 パフォーマンススキーマ接続テーブル

MySQL 5.6.3 以降、パフォーマンススキーマはサーバーへの接続に関する統計を提供します。クライアントが接続する場合、特定のユーザー名で特定のホストからそれを行います。パフォーマンススキーマは、これらのテー

ブルを使用して、アカウント (ユーザー名とホスト名) ごとに、およびユーザー名ごととホスト名ごとの個別に、接続を追跡します。

- accounts: クライアントアカウントごとの接続統計
- hosts: クライアントホスト名ごとの接続統計
- users: クライアントユーザー名ごとの接続統計

接続に関する情報を集計するサマリーテーブルもあります。セクション22.9.9.7「接続サマリーテーブル」を参照してください。

接続テーブルでの「アカウント」の意味は、その用語がユーザーとホスト値の組み合わせを表すという点で、mysql データベース内の MySQL 付与テーブルでのその意味に似ています。付与テーブルでそれらが異なる点は、アカウントのホスト部分をパターンにできるのに対し、接続テーブルではホスト値は常に固有の非パターンホスト名であることです。

接続テーブルにはすべて CURRENT_CONNECTIONS および TOTAL_CONNECTIONS カラムがあり、統計が基づく「追跡値」ごとの現在と合計の接続数を追跡します。テーブルは、それらが追跡値に使用するものに違いがあります。accounts テーブルには USER および HOST カラムがあり、ユーザー名とホスト名の組み合わせごとに接続を追跡します。users および hosts テーブルには USER および HOST カラムがあり、それぞれユーザー名ごとおよびホスト名ごとに接続を追跡します。

user1 と user2 というクライアントがそれぞれ hosta と hostb から一度に接続するとします。パフォーマンススキーマは次のように接続を追跡します。

- accounts テーブルには、user1/hosta、user1/hostb、user2/hosta、および user2/hostb アカウント値の 4 つの行があり、各行はアカウントごとに 1 つの接続をカウントします。
- users テーブルには、user1 と user2 値の 2 つの行があり、各行はユーザー名ごとに 2 つの接続をカウントします。
- hosts テーブルには、hosta と hostb 値の 2 つの行があり、各行はホスト名ごとに 2 つの接続をカウントします。

クライアントが接続すると、パフォーマンススキーマは、各テーブルに該当する追跡値を使用して、接続に適用する各接続テーブル内の行を判断します。そのような行がない場合、追加されます。次に、パフォーマンススキーマはその行の CURRENT_CONNECTIONS および TOTAL_CONNECTIONS カラムを 1 つ増分します。

クライアントが切断すると、パフォーマンススキーマはその行の CURRENT_CONNECTIONS カラムを 1 つ減分し、TOTAL_CONNECTIONS カラムは変更しないままにします。

パフォーマンススキーマは内部スレッドと認証に失敗したユーザーセッションのスレッドもカウントします。これらは、NULL の値の USER および HOST カラムのある行でカウントされます。

各接続テーブルは TRUNCATE TABLE で切り捨てることができ、次の効果があります。

- CURRENT_CONNECTIONS = 0 の行が削除されます。
- CURRENT_CONNECTIONS > 0 の行で、TOTAL_CONNECTIONS が CURRENT_CONNECTIONS にリセットされます。
- 接続テーブルに依存する接続サマリーテーブルは暗黙的に切り捨てられます (サマリー値は 0 に設定されます)。暗黙的な切り捨ての詳細については、セクション22.9.9.7「接続サマリーテーブル」を参照してください。

22.9.7.1 accounts テーブル

accounts テーブルは、MySQL サーバーに接続した各アカウントの行を格納します。各アカウントについて、テーブルは現在と合計の接続数をカウントします。テーブルサイズはサーバー起動時に自動サイズ設定されます。テーブルサイズを明示的に設定するには、サーバー起動時に performance_schema_accounts_size システム変数を設定します。アカウント統計を無効にするには、この変数を 0 に設定します。

accounts テーブルには次のカラムがあります。TRUNCATE TABLE の効果を含む、パフォーマンススキーマがこのテーブルの行を保守する方法については、セクション22.9.7「パフォーマンススキーマ接続テーブル」を参照してください。

- USER

接続のクライアントユーザー名、または内部スレッドまたは認証に失敗したユーザーセッションの場合 NULL。

- HOST

  クライアントの接続元のホスト名、または内部スレッドまたは認証に失敗したユーザーセッションの場合 NULL。

- CURRENT_CONNECTIONS

  アカウントの現在の接続数。

- TOTAL_CONNECTIONS

  アカウントの合計の接続数。

accounts テーブルは MySQL 5.6.3 で追加されました。

22.9.7.2 hosts テーブル

hosts テーブルは、クライアントが MySQL サーバーに接続している各ホストの行を格納します。各ホスト名について、テーブルは現在と合計の接続数をカウントします。テーブルサイズはサーバー起動時に自動サイズ設定されます。テーブルサイズを明示的に設定するには、サーバー起動時に performance_schema_hosts_size システム変数を設定します。ホスト統計を無効にするには、この変数を 0 に設定します。

hosts テーブルには次のカラムがあります。TRUNCATE TABLE の効果を含む、パフォーマンススキーマがこのテーブルの行を保守する方法については、セクション22.9.7「パフォーマンススキーマ接続テーブル」を参照してください。

- HOST

  クライアントの接続元のホスト名、または内部スレッドまたは認証に失敗したユーザーセッションの場合 NULL。

- CURRENT_CONNECTIONS

  ホストの現在の接続数。

- TOTAL_CONNECTIONS

  ホストの合計の接続数。

hosts テーブルは MySQL 5.6.3 で追加されました。

22.9.7.3 users テーブル

users テーブルは、MySQL サーバーに接続している各ユーザーの行を格納します。各ユーザー名について、テーブルは現在と合計の接続数をカウントします。テーブルサイズはサーバー起動時に自動サイズ設定されます。テーブルサイズを明示的に設定するには、サーバー起動時に performance_schema_users_size システム変数を設定します。ユーザー統計を無効にするには、この変数を 0 に設定します。

users テーブルには次のカラムがあります。TRUNCATE TABLE の効果を含む、パフォーマンススキーマがこのテーブルの行を保守する方法については、セクション22.9.7「パフォーマンススキーマ接続テーブル」を参照してください。

- USER

  接続のクライアントユーザー名、または内部スレッドまたは認証に失敗したユーザーセッションの場合 NULL。

- CURRENT_CONNECTIONS

  ユーザーの現在の接続数。

- TOTAL_CONNECTIONS

  ユーザーの合計の接続数。

users テーブルは MySQL 5.6.3 で追加されました。

22.9.8 パフォーマンススキーマ接続属性テーブル

MySQL 5.6.6 以降、アプリケーションプログラムでは、mysql_options() および mysql_options4() C API 関数を使用して、接続時にサーバーに渡されるキー/値接続属性を提供できます。パフォーマンススキーマはこれらのテーブルからこの情報を公開します。

- session_account_connect_attrs: 現在のアカウントのセッションの接続属性。
- session_connect_attrs: すべてのセッションの接続属性。

22.9.8.1 session_account_connect_attrs テーブル

MySQL 5.6.6 以降、アプリケーションプログラムでは、mysql_options() および mysql_options4() C API 関数を使用して、接続時にサーバーに渡されるキー/値接続属性を提供できます。

session_account_connect_attrs テーブルは、自分のアカウントに対してオープンしているセッションの接続属性のみを格納します。すべてのセッションの接続属性を確認するには、session_connect_attrs テーブルを調べてください。

session_account_connect_attrs テーブルは、これらのカラムを格納します。

- PROCESSLIST_ID

  セッションの接続識別子。

- ATTR_NAME

  属性名。

- ATTR_VALUE

  属性値。

- ORDINAL_POSITION

  属性が一連の接続属性に追加された順序。

22.9.8.2 session_connect_attrs テーブル

MySQL 5.6.6 以降、アプリケーションプログラムでは、mysql_options() および mysql_options4() C API 関数を使用して、接続時にサーバーに渡されるキー/値接続属性を提供できます。

session_connect_attrs テーブルはすべてのセッションの接続属性を格納します。自分のアカウントに対してオープンしているセッションの接続属性のみを確認するには、session_account_connect_attrs テーブルを調べてください。

session_connect_attrs テーブルは、これらのカラムを格納します。

- PROCESSLIST_ID

  セッションの接続識別子。

- ATTR_NAME

  属性名。

- ATTR_VALUE

  属性値。

- ORDINAL_POSITION

  属性が一連の接続属性に追加された順序。

22.9.9 パフォーマンススキーマサマリーテーブル

サマリーテーブルは、時間の経過とともに強制終了されたイベントについて集計された情報を提供します。このグループのテーブルには、さまざまな方法で、イベントデータが要約されます。

イベント待機サマリー:

- events_waits_summary_global_by_event_name: イベント名ごとに要約された待機イベント
- events_waits_summary_by_instance: インスタンスごとに要約された待機イベント
- events_waits_summary_by_thread_by_event_name: スレッドおよびイベント名ごとに要約された待機イベント

ステージサマリー:

- events_stages_summary_by_thread_by_event_name: スレッドおよびイベント名ごとに要約されたステージ待機
- events_stages_summary_global_by_event_name: イベント名ごとに要約されたステージ待機

ステートメントサマリー:

- events_statements_summary_by_digest: スキーマおよびダイジェスト値ごとに要約されたステートメントイベント
- events_statements_summary_by_thread_by_event_name: スレッドおよびイベント名ごとに要約されたステートメントイベント
- events_statements_summary_global_by_event_name: イベント名ごとに要約されたステートメントイベント

オブジェクト待機サマリー:

- objects_summary_global_by_type: オブジェクトサマリー

ファイル I/O サマリー:

- file_summary_by_event_name: イベント名ごとに要約されたファイルイベント
- file_summary_by_instance: ファイルインスタンスごとに要約されたファイルイベント

テーブル I/O およびロック待機サマリー:

- table_io_waits_summary_by_index_usage: インデックスごとのテーブル I/O 待機
- table_io_waits_summary_by_table: テーブルごとのテーブル I/O 待機
- table_lock_waits_summary_by_table: テーブルごとのテーブルロック待機

接続サマリー:

- events_waits_summary_by_account_by_event_name: アカウントおよびイベント名ごとに要約された待機イベント
- events_waits_summary_by_user_by_event_name: ユーザー名およびイベント名ごとに要約された待機イベント
- events_waits_summary_by_host_by_event_name: ホスト名およびイベント名ごとに要約された待機イベント
- events_stages_summary_by_account_by_event_name: アカウントおよびイベント名ごとに要約されたステージイベント
- events_stages_summary_by_user_by_event_name: ユーザー名およびイベント名ごとに要約されたステージイベント
- events_stages_summary_by_host_by_event_name: ホスト名およびイベント名ごとに要約されたステージイベント
- events_statements_summary_by_digest: スキーマおよびダイジェスト値ごとに要約されたステートメントイベント
- events_statements_summary_by_account_by_event_name: アカウントおよびイベント名ごとに要約されたステートメントイベント
- events_statements_summary_by_user_by_event_name: ユーザー名およびイベント名ごとに要約されたステートメントイベント
- events_statements_summary_by_host_by_event_name: ホスト名およびイベント名ごとに要約されたステートメントイベント

ソケットサマリー:

- socket_summary_by_instance: インスタンスごとに要約されたソケット待機および I/O
- socket_summary_by_event_name: イベント名ごとに要約されたソケット待機および I/O

各サマリーテーブルには、集計するデータのグループ化の方法を決定するグループ化カラムと、集計値を格納するサマリーカラムがあります。同様の方法でイベントを要約するテーブルには、多くの場合に類似の一連のサマリーカラムがあり、イベントの集計方法を決定するために使用されるグループ化カラムのみ異なります。

サマリーテーブルは TRUNCATE TABLE で切り捨てることができます。events_statements_summary_by_digest を除き、その効果は、サマリーカラムを 0 または NULL にリセットすることで、行を削除することではありません。これにより、収集された値をクリアし、アグリゲーションを再開できます。それは、実行時構成の変更を行なったあとなどに便利な場合があります。

22.9.9.1 イベント待機サマリーテーブル

パフォーマンススキーマは現在および最近の待機イベントを収集するためのテーブルを保守し、その情報をサマリーテーブルに集計します。セクション22.9.4「パフォーマンススキーマ待機イベントテーブル」に待機サマリーが基づいているイベントについて説明しています。待機イベントの内容、現在および最近の待機イベントテーブル、および待機イベント収集の制御方法に関する情報については、その説明を参照してください。

各イベント待機サマリーテーブルには、テーブルのイベントの集計方法を示す 1 つまたは複数のグループ化カラムがあります。イベント名は、setup_instruments テーブル内のイベントインストゥルメントの名前を表します。

- events_waits_summary_global_by_event_name には EVENT_NAME カラムがあります。各行は特定のイベント名のイベントを要約します。インストゥルメントを使用して、インストゥルメントされるオブジェクトの複数のインスタンスを作成できます。たとえば、接続ごとに作成される相互排他ロックのインストゥルメントがある場合、接続と同じ数のインスタンスがあります。インストゥルメントのサマリー行は、これらのすべてのインスタンス全体を要約します。
- events_waits_summary_by_instance には EVENT_NAME および OBJECT_INSTANCE_BEGIN カラムがあります。各行は特定のイベント名とオブジェクトのイベントを要約します。複数のインスタンスを作成するためにインストゥルメントが使用される場合、各インスタンスには一意の OBJECT_INSTANCE_BEGIN 値があるため、これらのインスタンスはこのテーブルで個別に要約されます。
- events_waits_summary_by_thread_by_event_name には THREAD_ID および EVENT_NAME カラムがあります。各行は特定のスレッドおよびイベント名のイベントを要約します。

すべてのイベント待機サマリーテーブルには、集計された値を格納するこれらのサマリーカラムがあります。

- COUNT_STAR

  要約されたイベントの数。この値には、時間付きか時間なしかに関係なく、すべてのイベントが含まれます。

- SUM_TIMER_WAIT

  要約された時間付きイベントの合計待機時間。時間なしイベントは NULL の待機時間を持つため、この値は時間付きイベントに対してのみ計算されます。同じことがほかの xxx_TIMER_WAIT 値にも当てはまります。

- MIN_TIMER_WAIT

  要約された時間付きイベントの最小待機時間。

- AVG_TIMER_WAIT

  要約された時間付きイベントの平均待機時間。

- MAX_TIMER_WAIT

  要約された時間付きイベントの最大待機時間。

待機イベントサマリー情報の例:

```
mysql> SELECT * FROM events_waits_summary_global_by_event_name\G
...
*************************** 6. row ***************************
   EVENT_NAME: wait/synch/mutex/sql/BINARY_LOG::LOCK_index
   COUNT_STAR: 8
SUM_TIMER_WAIT: 2119302
MIN_TIMER_WAIT: 196092
AVG_TIMER_WAIT: 264912
```

```
MAX_TIMER_WAIT: 569421
...
*************************** 9. row ***************************
   EVENT_NAME: wait/synch/mutex/sql/hash_filo::lock
   COUNT_STAR: 69
SUM_TIMER_WAIT: 16848828
MIN_TIMER_WAIT: 0
AVG_TIMER_WAIT: 244185
MAX_TIMER_WAIT: 735345
...
```

TRUNCATE TABLE は待機サマリーテーブルに使用できます。それは、行を削除するのではなく、サマリーカラムを 0 にリセットします。

22.9.9.2 ステージサマリーテーブル

MySQL 5.6.3 以降、パフォーマンススキーマは現在および最近のステージイベントを収集するためのテーブルを保守し、その情報をサマリーテーブルに集計します。セクション22.9.5「パフォーマンススキーマステージイベントテーブル」で、ステージサマリーが基づいているイベントについて説明しています。ステージイベントの内容、現在および最近のステージイベントテーブル、およびステージイベント収集の制御方法に関する情報については、その説明を参照してください。

各ステージサマリーテーブルには、テーブルのイベントの集計方法を示す 1 つまたは複数のグループ化カラムがあります。イベント名は、setup_instruments テーブル内のイベントインストゥルメントの名前を表します。

- events_stages_summary_by_thread_by_event_name には THREAD_ID および EVENT_NAME カラムがあります。各行は特定のスレッドおよびイベント名のイベントを要約します。
- events_stages_summary_global_by_event_name には EVENT_NAME カラムがあります。各行は特定のイベント名のイベントを要約します。

すべてのステージサマリーテーブルには、集計された値を格納するこれらのサマリーカラムがあります。COUNT_STAR、SUM_TIMER_WAIT、MIN_TIMER_WAIT, AVG_TIMER_WAIT、および MAX_TIMER_WAIT。ステージサマリーテーブルでは、events_waits_current ではなく events_stages_current から待機を集計することを除き、これらのカラムはイベント待機サマリーテーブル (セクション22.9.9.1「イベント待機サマリーテーブル」を参照) の同じ名前のカラムに似ています。

ステージイベントサマリー情報の例:

```
mysql> SELECT * FROM events_stages_summary_global_by_event_name\G
...
*************************** 5. row ***************************
   EVENT_NAME: stage/sql/checking permissions
   COUNT_STAR: 57
SUM_TIMER_WAIT: 26501888880
MIN_TIMER_WAIT: 7317456
AVG_TIMER_WAIT: 464945295
MAX_TIMER_WAIT: 12858936792
...
*************************** 9. row ***************************
   EVENT_NAME: stage/sql/closing tables
   COUNT_STAR: 37
SUM_TIMER_WAIT: 662606568
MIN_TIMER_WAIT: 1593864
AVG_TIMER_WAIT: 17907891
MAX_TIMER_WAIT: 437977248
...
```

TRUNCATE TABLE はステージサマリーテーブルに使用できます。それは、行を削除するのではなく、サマリーカラムを 0 にリセットします。

22.9.9.3 ステートメントサマリーテーブル

MySQL 5.6.3 以降、パフォーマンススキーマは現在および最近のステートメントイベントを集計するためのテーブルを保守し、その情報をサマリーテーブルに集計します。セクション22.9.6「パフォーマンススキーマステートメントイベントテーブル」で、ステートメントサマリーが基づいているイベントについて説明しています。ステートメントイベントの内容、現在および最近のステートメントイベントテーブル、およびステートメントイベント収集の制御方法に関する情報については、その説明を参照してください。

各ステートメントサマリーテーブルには、テーブルのイベントの集計方法を示す 1 つまたは複数のグループ化カラムがあります。イベント名は、setup_instruments テーブル内のイベントインストゥルメントの名前を表します。

- events_statements_summary_by_digest には SCHEMA_NAME および DIGEST カラムがあります。各行は特定のスキーマ/ダイジェスト値のイベントを要約します。(DIGEST_TEXT カラムは、対応する正規化されたステートメントダイジェストテキストを格納しますが、グループ化カラムでもサマリーカラムでもありません。)

  このテーブルは、5.6.5 で追加されました。MySQL 5.6.9 より前では、SCHEMA_NAME カラム値がなく、グループ化は DIGEST 値のみに基づきます。

- events_statements_summary_by_thread_by_event_name には THREAD_ID および EVENT_NAME カラムがあります。各行は特定のスレッドおよびイベント名のイベントを要約します。

- events_statements_summary_global_by_event_name には EVENT_NAME カラムがあります。各行は特定のイベント名のイベントを要約します。

すべてのステートメントサマリーテーブルには、集計された値を格納するこれらのサマリーカラムがあります。

- COUNT_STAR、SUM_TIMER_WAIT、MIN_TIMER_WAIT、AVG_TIMER_WAIT、MAX_TIMER_WAIT

  ステートメントサマリーテーブルは、events_waits_current ではなく events_statements_current からイベントを集計することを除き、これらのカラムはイベント待機サマリーテーブル (セクション22.9.9.1「イベント待機サマリーテーブル」を参照) の同じ名前のカラムに似ています。

- SUM_xxx

  events_statements_current テーブル内の対応する xxx カラムの集計。たとえば、ステートメントサマリーテーブルの SUM_LOCK_TIME および SUM_ERRORS カラムは events_statements_current テーブルの LOCK_TIME および ERRORS カラムの集計です。

events_statements_summary_by_digest テーブルにはこれらの追加のサマリーカラムがあります。

- FIRST_SEEN_TIMESTAMP、LAST_SEEN_TIMESTAMP

  特定のダイジェスト値を持つステートメントが最初に確認された時間と最後に確認された時間。

ステートメントイベントサマリー情報の例:

```
mysql> SELECT * FROM events_statements_summary_global_by_event_name\G
*************************** 1. row ***************************
            EVENT_NAME: statement/sql/select
            COUNT_STAR: 25
        SUM_TIMER_WAIT: 1535983999000
        MIN_TIMER_WAIT: 209823000
        AVG_TIMER_WAIT: 61439359000
        MAX_TIMER_WAIT: 1363397650000
         SUM_LOCK_TIME: 20186000000
            SUM_ERRORS: 0
          SUM_WARNINGS: 0
     SUM_ROWS_AFFECTED: 0
        SUM_ROWS_SENT: 388
    SUM_ROWS_EXAMINED: 370
SUM_CREATED_TMP_DISK_TABLES: 0
  SUM_CREATED_TMP_TABLES: 0
    SUM_SELECT_FULL_JOIN: 0
SUM_SELECT_FULL_RANGE_JOIN: 0
      SUM_SELECT_RANGE: 0
 SUM_SELECT_RANGE_CHECK: 0
       SUM_SELECT_SCAN: 6
  SUM_SORT_MERGE_PASSES: 0
      SUM_SORT_RANGE: 0
       SUM_SORT_ROWS: 0
       SUM_SORT_SCAN: 0
    SUM_NO_INDEX_USED: 6
SUM_NO_GOOD_INDEX_USED: 0
...
```

TRUNCATE TABLE はステートメントサマリーテーブルに使用できます。events_statements_summary_by_digest に対して、それはテーブルを空にします。ほかのステートメントサマリーテーブルに対して、それは行を削除するのではなく、サマリーカラムを 0 にリセットします。

ステートメントダイジェストアグリゲーションルール

statement_digest コンシューマを有効にすると、ステートメントの完了時に、events_statements_summary_by_digest へのアグリゲーションが次のように行われます。アグリゲーションはステートメントに対して計算された DIGEST 値に基づきます。

- 完了したばかりのステートメントのダイジェスト値のある events_statements_summary_by_digest 行がすでに存在する場合、ステートメントの統計はその行に集計されます。LAST_SEEN カラムは現在の時間に更新されます。
- 完了したばかりのステートメントのダイジェスト値のある行がなく、テーブルがいっぱいでない場合、そのステートメントに対して新しい行が作成されます。FIRST_SEEN および LAST_SEEN カラムは現在の時間で初期化されます。
- 完了したばかりのステートメントのステートメントダイジェスト値のある行がなく、テーブルがいっぱいである場合、完了したばかりのステートメントの統計が、必要に応じて作成される特別な「多目的」行に、DIGEST = NULL で追加されます。この行が作成された場合、FIRST_SEEN および LAST_SEEN カラムは現在の時間で初期化されます。そうでない場合、LAST_SEEN カラムが現在の時間で更新されます。

パフォーマンススキーマテーブルには、メモリー制約による最大サイズがあるため、DIGEST = NULL の行は保守されます。DIGEST = NULL 行は、ほかの行に一致しないダイジェストが、サマリーテーブルがいっぱいである場合でも、共通の「ほかの」バケットを使用して、カウントされることを許可します。この行は、ダイジェストサマリーが代表的であるかどうかを推定するのに役立ちます。

- すべてのダイジェストのうち 5% を表す COUNT_STAR 値がある DIGEST = NULL 行は、ダイジェストサマリーテーブルがきわめて代表的であることを示します。ほかの行が、存在するステートメントの 95% を占めます。
- すべてのダイジェストのうち 50% を表す COUNT_STAR 値がある DIGEST = NULL 行は、ダイジェストサマリーテーブルがあまり代表的でないことを示します。ほかの行は、存在するステートメントの半分しか占めません。たいていの場合に DBA は DIGEST = NULL 行にカウントされる行の多くが、代わりにより具体的な行を使用してカウントされるように、最大テーブルサイズを拡大するべきです。これを実行するには、サーバーの起動時に、performance_schema_digests_size システム変数を大きな値に設定します。デフォルトサイズは 200 です。

22.9.9.4 オブジェクト待機サマリーテーブル

objects_summary_global_by_type テーブルはオブジェクト待機イベントを集計します。これにはテーブルのイベントの集計方法を示すグループ化カラム OBJECT_TYPE、OBJECT_SCHEMA、および OBJECT_NAME があります。各行は特定のオブジェクトのイベントを要約します。

objects_summary_global_by_type には events_waits_summary_by_xxx テーブルと同じサマリーカラムがあります。セクション22.9.9.1「イベント待機サマリーテーブル」を参照してください。

オブジェクト待機イベントサマリー情報の例:

```
mysql> SELECT * FROM objects_summary_global_by_type\G
...
*************************** 3. row ***************************
  OBJECT_TYPE: TABLE
OBJECT_SCHEMA: test
  OBJECT_NAME: t
   COUNT_STAR: 3
SUM_TIMER_WAIT: 263126976
MIN_TIMER_WAIT: 1522272
AVG_TIMER_WAIT: 87708678
MAX_TIMER_WAIT: 258428280
...
*************************** 10. row ***************************
  OBJECT_TYPE: TABLE
OBJECT_SCHEMA: mysql
  OBJECT_NAME: user
   COUNT_STAR: 14
SUM_TIMER_WAIT: 365567592
MIN_TIMER_WAIT: 1141704
AVG_TIMER_WAIT: 26111769
MAX_TIMER_WAIT: 334783032
...
```

TRUNCATE TABLE はオブジェクトサマリーテーブルに使用できます。それは、行を削除するのではなく、サマリーカラムを 0 にリセットします。

22.9.9.5 ファイル I/O サマリーテーブル

ファイル I/O サマリーテーブルは I/O 操作に関する情報を集計します。

各ファイル I/O サマリーテーブルには、テーブルのイベントの集計方法を示す 1 つまたは複数のグループ化カラムがあります。イベント名は、setup_instruments テーブル内のイベントインストゥルメントの名前を表します。

- file_summary_by_event_name には EVENT_NAME カラムがあります。各行は特定のイベント名のイベントを要約します。
- file_summary_by_instance には FILE_NAME、EVENT_NAME、および (MySQL 5.6.4 現在) OBJECT_INSTANCE_BEGIN カラムがあります。各行は特定のファイルおよびイベント名のイベントを要約します。

すべてのファイル I/O サマリーテーブルには、集計された値を格納する次のサマリーカラムがあります。(MySQL 5.6.4 より前では、テーブルに COUNT_READ COUNT_WRITE SUM_NUMBER_OF_BYTES_READ、および SUM_NUMBER_OF_BYTES_WRITE アグリゲーションカラムのみが格納されます。)一部のカラムは一般的で、より詳細なカラムの値の合計と同じ値を持ちます。このように、低レベルカラムを合計するユーザー定義ビューを必要とせずに、高レベルでのアグリゲーションを直接取得できます。

- COUNT_STAR、SUM_TIMER_WAIT、MIN_TIMER_WAIT、AVG_TIMER_WAIT、MAX_TIMER_WAIT

  これらのカラムはすべての I/O 操作を集計します。

- COUNT_READ、SUM_TIMER_READ、MIN_TIMER_READ、AVG_TIMER_READ、MAX_TIMER_READ、SUM_NUMBER_C

  これらのカラムは FGETS、FGETC、FREAD、および READ を含むすべての読み取り操作を集計します。

- COUNT_WRITE、SUM_TIMER_WRITE、MIN_TIMER_WRITE、AVG_TIMER_WRITE、MAX_TIMER_WRITE、SUM_NUMBE

  これらのカラムは FPUTS、FPUTC、FPRINTF、VFPRINTF、FWRITE、および PWRITE を含むすべての書き込み操作を集計します。

- COUNT_MISC、SUM_TIMER_MISC、MIN_TIMER_MISC、AVG_TIMER_MISC、MAX_TIMER_MISC

  これらのカラムは CREATE、DELETE、OPEN、CLOSE、STREAM_OPEN、STREAM_CLOSE、SEEK、TELL、FLUSH、STAT、FSTAT、CH および SYNC を含むその他のすべての I/O 操作を集計します。これらの操作のバイトカウントはありません。

ファイル I/O イベントサマリー情報の例:

```
mysql> SELECT * FROM file_summary_by_event_name\G
...
*************************** 2. row ***************************
            EVENT_NAME: wait/io/file/sql/binlog
            COUNT_STAR: 31
        SUM_TIMER_WAIT: 8243784888
        MIN_TIMER_WAIT: 0
        AVG_TIMER_WAIT: 265928484
        MAX_TIMER_WAIT: 6490658832
...
mysql> SELECT * FROM file_summary_by_instance\G
...
*************************** 2. row ***************************
             FILE_NAME: /var/mysql/share/english/errmsg.sys
            EVENT_NAME: wait/io/file/sql/ERRMSG
            EVENT_NAME: wait/io/file/sql/ERRMSG
  OBJECT_INSTANCE_BEGIN: 4686193384
            COUNT_STAR: 5
        SUM_TIMER_WAIT: 13990154448
        MIN_TIMER_WAIT: 26349624
        AVG_TIMER_WAIT: 2798030607
        MAX_TIMER_WAIT: 8150662536
...
```

TRUNCATE TABLE はファイル I/O サマリーテーブルに使用できます。それは、行を削除するのではなく、サマリーカラムを 0 にリセットします。

MySQL サーバーは、いくつかの技法を使用して、ファイルから読み取られた情報をキャッシュすることによって I/O 操作を回避するため、I/O イベントが発生すると予想されるようなステートメントでも発生しない可能性があります。キャッシュをフラッシュするか、サーバーを再起動して、その状態をリセットすることによって、I/O を発生させることができる場合があります。

22.9.9.6 テーブル I/O およびロック待機サマリーテーブル

次のセクションでは、テーブル I/O およびロック待機サマリーテーブルについて説明します。

- table_io_waits_summary_by_index_usage: インデックスごとのテーブル I/O 待機
- table_io_waits_summary_by_table: テーブルごとのテーブル I/O 待機

- table_lock_waits_summary_by_table: テーブルごとのテーブルロック待機

table_io_waits_summary_by_table テーブル

table_io_waits_summary_by_table テーブルは、wait/io/table/sql/handler インストゥルメントによって生成されるすべてのテーブル I/O 待機イベントを集計します。グループ化はテーブル単位です。

table_io_waits_summary_by_table テーブルには、テーブルのイベントの集計方法を示すこれらのグループ化カラムがあります。OBJECT_TYPE、OBJECT_SCHEMA、および OBJECT_NAME。これらのカラムは、events_waits_current テーブル内と同じ意味を持ちます。それらは、行の適用先のテーブルを識別します。

table_io_waits_summary_by_table には集計された値を格納する次のサマリーカラムがあります。カラムの説明に示すように、一部のカラムは一般的で、より詳細なカラムの値の合計と同じ値を持ちます。たとえば、すべての書き込みを集計するカラムは、挿入、更新、および削除を集計する対応するカラムの合計を保持します。このように、低レベルカラムを合計するユーザー定義ビューを必要とせずに、高レベルでのアグリゲーションを直接取得できます。

- COUNT_STAR、SUM_TIMER_WAIT、MIN_TIMER_WAIT、AVG_TIMER_WAIT、MAX_TIMER_WAIT

  これらのカラムはすべての I/O 操作を集計します。それらは対応する xxx_READ および xxx_WRITE カラムの合計と同じです。

- COUNT_READ、SUM_TIMER_READ、MIN_TIMER_READ、AVG_TIMER_READ、MAX_TIMER_READ

  これらのカラムはすべての読み取り操作を集計します。それらは対応する xxx_FETCH カラムの合計と同じです。

- COUNT_WRITE、SUM_TIMER_WRITE、MIN_TIMER_WRITE、AVG_TIMER_WRITE、MAX_TIMER_WRITE

  これらのカラムはすべての書き込み操作を集計します。それらは対応する xxx_INSERT、xxx_UPDATE、および xxx_DELETE カラムの合計と同じです。

- COUNT_FETCH、SUM_TIMER_FETCH、MIN_TIMER_FETCH、AVG_TIMER_FETCH、MAX_TIMER_FETCH

  これらのカラムはすべてのフェッチ操作を集計します。

- COUNT_INSERT、SUM_TIMER_INSERT、MIN_TIMER_INSERT、AVG_TIMER_INSERT、MAX_TIMER_INSERT

  これらのカラムはすべての挿入操作を集計します。

- COUNT_UPDATE、SUM_TIMER_UPDATE、MIN_TIMER_UPDATE、AVG_TIMER_UPDATE、MAX_TIMER_UPDATE

  これらのカラムはすべての更新操作を集計します。

- COUNT_DELETE、SUM_TIMER_DELETE、MIN_TIMER_DELETE、AVG_TIMER_DELETE、MAX_TIMER_DELETE

  これらのカラムはすべての削除操作を集計します。

TRUNCATE TABLE はテーブル I/O サマリーテーブルに使用できます。それは、行を削除するのではなく、サマリーカラムを 0 にリセットします。このテーブルを切り捨てると、table_io_waits_summary_by_index_usage テーブルも切り捨てられます。

table_io_waits_summary_by_index_usage テーブル

table_io_waits_summary_by_index_usage テーブルは、wait/io/table/sql/handler インストゥルメントによって生成されるすべてのテーブルインデックス I/O 待機イベントを集計します。グループ化はテーブルインデックス単位です。

table_io_waits_summary_by_index_usage の構造は、table_io_waits_summary_by_table とほぼ同一です。唯一の違いは、テーブル I/O 待機イベントが記録されたときに使用されたインデックスの名前に対応する、追加のグループカラム INDEX_NAME です。

- PRIMARY の値はテーブル I/O でプライマリインデックスが使用されたことを示します。
- NULL の値はテーブル I/O でインデックスが使用されなかったことを示します。
- 挿入は INDEX_NAME = NULL に対してカウントされます。

TRUNCATE TABLE はテーブル I/O サマリーテーブルに使用できます。それは、行を削除するのではなく、サマリーカラムを 0 にリセットします。このテーブルも、table_io_waits_summary_by_table テーブルの切り捨てに

よって切り捨てられます。テーブルのインデックス構造を変更する DDL 操作により、インデックスごとの統計がリセットされることがあります。

table_lock_waits_summary_by_table テーブル

table_lock_waits_summary_by_table テーブルは、wait/lock/table/sql/handler インストゥルメントによって生成されるすべてのテーブルロック待機イベントを集計します。グループ化はテーブル単位です。

このテーブルには、内部および外部ロックに関する情報が格納されます。

- 内部ロックは、SQL レイヤーのロックに対応します。これは現在 thr_lock() への呼び出しによって実装されます。イベント行で、これらのロックは、次のいずれかの値を持つ OPERATION カラムによって区別されます。

```
read normal
read with shared locks
read high priority
read no insert
write allow write
write concurrent insert
write delayed
write low priority
write normal
```

- 外部ロックはストレージエンジンレイヤーのロックに対応します。これは現在 handler::external_lock() への呼び出しによって実装されます。イベント行で、これらのロックは、次のいずれかの値を持つ OPERATION カラムによって区別されます。

```
read external
write external
```

table_lock_waits_summary_by_table テーブルには、テーブルのイベントの集計方法を示すこれらのグループ化カラムがあります。OBJECT_TYPE、OBJECT_SCHEMA、および OBJECT_NAME。これらのカラムは、events_waits_current テーブル内と同じ意味を持ちます。それらは、行の適用先のテーブルを識別します。

table_lock_waits_summary_by_table には集計された値を格納する次のサマリーカラムがあります。カラムの説明に示すように、一部のカラムは一般的で、より詳細なカラムの値の合計と同じ値を持ちます。たとえば、すべてのロックを集計するカラムは、読み取りおよび書き込みロックを集計する対応するカラムの合計を保持します。このように、低レベルカラムを合計するユーザー定義ビューを必要とせずに、高レベルでのアグリゲーションを直接取得できます。

- COUNT_STAR、SUM_TIMER_WAIT、MIN_TIMER_WAIT、AVG_TIMER_WAIT、MAX_TIMER_WAIT

  これらのカラムはすべてのロック操作を集計します。それらは対応する xxx_READ および xxx_WRITE カラムの合計と同じです。

- COUNT_READ、SUM_TIMER_READ、MIN_TIMER_READ、AVG_TIMER_READ、MAX_TIMER_READ

  これらのカラムはすべての読み取りロック操作を集計します。それらは対応する xxx_READ_NORMAL、xxx_READ_WITH_SHARED_LOCKS、xxx_READ_HIGH_PRIORITY、および xxx_READ_NO_INSERT カラムの合計と同じです。

- COUNT_WRITE、SUM_TIMER_WRITE、MIN_TIMER_WRITE、AVG_TIMER_WRITE、MAX_TIMER_WRITE

  これらのカラムはすべての書き込みロック操作を集計します。それらは対応する xxx_WRITE_ALLOW_WRITE、xxx_WRITE_CONCURRENT_INSERT、xxx_WRITE_DELAYED、xxx_WRITE_LOW_PRIORIT および xxx_WRITE_NORMAL カラムの合計と同じです。

- COUNT_READ_NORMAL、SUM_TIMER_READ_NORMAL、MIN_TIMER_READ_NORMAL、AVG_TIMER_READ_NORMAL

  これらのカラムは内部読み取りロックを集計します。

- COUNT_READ_WITH_SHARED_LOCKS、SUM_TIMER_READ_WITH_SHARED_LOCKS、MIN_TIMER_READ_WITH_SHAR

  これらのカラムは内部読み取りロックを集計します。

- COUNT_READ_HIGH_PRIORITY、SUM_TIMER_READ_HIGH_PRIORITY、MIN_TIMER_READ_HIGH_PRIORITY、AVG_TI

  これらのカラムは内部読み取りロックを集計します。

- COUNT_READ_NO_INSERT、SUM_TIMER_READ_NO_INSERT、MIN_TIMER_READ_NO_INSERT、AVG_TIMER_READ_

  これらのカラムは内部読み取りロックを集計します。

- COUNT_READ_EXTERNAL、SUM_TIMER_READ_EXTERNAL、MIN_TIMER_READ_EXTERNAL、AVG_TIMER_READ

  これらのカラムは外部読み取りロックを集計します。

- COUNT_WRITE_ALLOW_WRITE、SUM_TIMER_WRITE_ALLOW_WRITE、MIN_TIMER_WRITE_ALLOW_WRITE、AVG

  これらのカラムは内部書き込みロックを集計します。

- COUNT_WRITE_CONCURRENT_INSERT、SUM_TIMER_WRITE_CONCURRENT_INSERT、MIN_TIMER_WRITE_CON

  これらのカラムは内部書き込みロックを集計します。

- COUNT_WRITE_DELAYED、SUM_TIMER_WRITE_DELAYED、MIN_TIMER_WRITE_DELAYED、AVG_TIMER_WRITE

  これらのカラムは内部書き込みロックを集計します。

  MySQL 5.6.6 以降、DELAYED 挿入は非推奨であるため、これらのカラムは将来のリリースで削除されます。

- COUNT_WRITE_LOW_PRIORITY、SUM_TIMER_WRITE_LOW_PRIORITY、MIN_TIMER_WRITE_LOW_PRIORITY、AV

  これらのカラムは内部書き込みロックを集計します。

- COUNT_WRITE_NORMAL、SUM_TIMER_WRITE_NORMAL、MIN_TIMER_WRITE_NORMAL、AVG_TIMER_WRITE_N

  これらのカラムは内部書き込みロックを集計します。

- COUNT_WRITE_EXTERNAL、SUM_TIMER_WRITE_EXTERNAL、MIN_TIMER_WRITE_EXTERNAL、AVG_TIMER_WR

  これらのカラムは外部書き込みロックを集計します。

TRUNCATE TABLE はテーブルロックサマリーテーブルに使用できます。それは、行を削除するのではなく、サマリーカラムを 0 にリセットします。

22.9.9.7 接続サマリーテーブル

接続サマリーテーブルは、スレッド単位ではなく、アカウント、ユーザー、またはホストごとにアグリゲーションが行われることを除いて、対応する events_xxx_summary_by_thread_by_event_name テーブルに似ています。

パフォーマンススキーマは、イベント名とアカウント、ユーザー、またはホストで接続統計を集計するサマリーテーブルを保守します。待機、ステージ、およびステートメントイベントを集計する個別のテーブルのグループを使用でき、それによってこの一連の接続サマリーテーブルが得られます。

- events_waits_summary_by_account_by_event_name: アカウントおよびイベント名ごとに要約された待機イベント
- events_waits_summary_by_user_by_event_name: ユーザー名およびイベント名ごとに要約された待機イベント
- events_waits_summary_by_host_by_event_name: ホスト名およびイベント名ごとに要約された待機イベント
- events_stages_summary_by_account_by_event_name: アカウントおよびイベント名ごとに要約されたステージイベント
- events_stages_summary_by_user_by_event_name: ユーザー名およびイベント名ごとに要約されたステージイベント
- events_stages_summary_by_host_by_event_name: ホスト名およびイベント名ごとに要約されたステージイベント
- events_statements_summary_by_account_by_event_name: アカウントおよびイベント名ごとに要約されたステートメントイベント
- events_statements_summary_by_user_by_event_name: ユーザー名およびイベント名ごとに要約されたステートメントイベント
- events_statements_summary_by_host_by_event_name: ホスト名およびイベント名ごとに要約されたステートメントイベント

つまり、接続サマリーテーブルは、形式 events_xxx_summary_yyy_by_event_name の名前を持ちます。ここで xxx は waits、stages、または statements で、yyy は account、user、または host です。

接続サマリーテーブルは中間アグリゲーションレベルを提供します。

- xxx_summary_by_thread_by_event_name テーブルは接続サマリーテーブルより詳細です
- xxx_summary_global_by_event_name テーブルは接続サマリーテーブルより詳細ではありません

各接続サマリーテーブルには、テーブルのイベントの集計方法を示す 1 つまたは複数のグループ化カラムがあります。イベント名は、setup_instruments テーブル内のイベントインストゥルメントの名前を表します。

- 名前に _by_account のあるテーブルでは、USER、HOST、および EVENT_NAME カラムで、アカウントとイベント名ごとにイベントがグループ化されます。
- 名前に _by_host のあるテーブルでは、HOST および EVENT_NAME カラムで、ホスト名とイベント名ごとにイベントがグループ化されます。
- 名前に _by_user のあるテーブルでは、USER および EVENT_NAME カラムで、ユーザー名とイベント名ごとにイベントがグループ化されます。

すべての接続サマリーテーブルには、集計された値を格納するこれらのサマリーカラムがあります。COUNT_STAR、SUM_TIMER_WAIT、MIN_TIMER_WAIT, AVG_TIMER_WAIT、および MAX_TIMER_WAIT。これらは events_waits_summary_by_instance テーブル内の同じ名前のカラムに似ています。ステートメントの接続サマリーテーブルには、ステートメントの種類を集計する追加の SUM_xxx カラムがあります。

接続サマリーテーブルは MySQL 5.6.3 で追加されました。

TRUNCATE TABLE は接続サマリーテーブルに使用できます。それは、行を削除するのではなく、サマリーカラムを 0 にリセットします。さらに、接続サマリーテーブルは、それらが依存する接続テーブルが切り捨てられた場合に、暗黙的に切り捨てられます。表22.2「暗黙的なテーブル切り捨ての効果」 に接続テーブルの切り捨てと暗黙的に切り捨てられるテーブルの関係を説明しています。

表 22.2 暗黙的なテーブル切り捨ての効果

切り捨てられたテーブル	暗黙的に切り捨てられたサマリーテーブル
accounts	%_by_account%、%_by_thread% に一致する名前を持つテーブル
hosts	%_by_account%、%_by_host%、%_by_thread% に一致する名前を持つテーブル
users	%_by_account%、%_by_user%、%_by_thread% に一致する名前を持つテーブル

22.9.9.8 ソケットサマリーテーブル

これらのソケットサマリーテーブルは、ソケット操作のタイマーおよびバイトカウント情報を集計します。

- socket_summary_by_instance: ソケットインスタンスごとに、すべてのソケット I/O 操作の wait/io/socket/* インストゥルメントによって生成されたタイマーおよびバイトカウント統計を集計します。接続が終了すると、socket_summary_by_instance 内のそれに対応する行が削除されます。
- socket_summary_by_event_name: ソケットインストゥルメントごとに、すべてのソケット I/O 操作の wait/io/socket/* インストゥルメントによって生成されたタイマーおよびバイトカウント統計を集計します。

ソケットサマリーテーブルは、ソケットがクライアントからの次のリクエストを待っている間に、idle イベントによって生成される待機は集計しません。idle イベントアグリゲーションには、待機イベントサマリーテーブルを使用します。セクション22.9.9.1「イベント待機サマリーテーブル」を参照してください。

各ソケットサマリーテーブルには、テーブルのイベントの集計方法を示す 1 つまたは複数のグループ化カラムがあります。イベント名は、setup_instruments テーブル内のイベントインストゥルメントの名前を表します。

- socket_summary_by_instance には OBJECT_INSTANCE_BEGIN カラムがあります。各行は特定のオブジェクトのイベントを要約します。
- socket_summary_by_event_name には EVENT_NAME カラムがあります。各行は特定のイベント名のイベントを要約します。

すべてのソケットサマリーテーブルには、集計された値を格納するこれらのサマリーカラムがあります。

- COUNT_STAR、SUM_TIMER_WAIT、MIN_TIMER_WAIT、AVG_TIMER_WAIT、MAX_TIMER_WAIT

  これらのカラムはすべての操作を集計します。

- COUNT_READ、SUM_TIMER_READ、MIN_TIMER_READ、AVG_TIMER_READ、MAX_TIMER_READ、SUM_NUMBE

  これらのカラムはすべての受信操作 (RECV、RECVFROM、および RECVMSG) を集計します。

- COUNT_WRITE、SUM_TIMER_WRITE、MIN_TIMER_WRITE、AVG_TIMER_WRITE、MAX_TIMER_WRITE、SUM_NU

  これらのカラムはすべての送信操作 (SEND、SENDTO、および SENDMSG) を集計します。

- COUNT_MISC、SUM_TIMER_MISC、MIN_TIMER_MISC、AVG_TIMER_MISC、MAX_TIMER_MISC

  これらのカラムは CONNECT、LISTEN、ACCEPT、CLOSE、および SHUTDOWN などのほかのすべてのソケット操作を集計します。これらの操作のバイトカウントはありません。

socket_summary_by_instance テーブルには、ソケットのクラス (client_connection、server_tcpip_socket、server_unix_socket) を示す EVENT_NAME カラムもあります。このカラムは、たとえば、クライアントアクティビティーをサーバー待機ソケットのそれから分離するために、グループ化できます。

これらのテーブルは MySQL 5.6.3 で追加されました。

TRUNCATE TABLE はソケットサマリーテーブルに使用できます。events_statements_summary_by_digest を除き、tt は行を削除するのではなく、サマリーカラムを 0 にリセットします。

22.9.10 パフォーマンススキーマのその他のテーブル

次のセクションでは、先述のセクションで説明したテーブルカテゴリに収まらないテーブルについて説明します。

- host_cache: 内部ホストキャッシュからの情報
- performance_timers: 使用可能なイベントタイマー
- threads: サーバースレッドに関する情報

22.9.10.1 host_cache テーブル

host_cache は、クライアントホスト名および IP アドレス情報を格納し、DNS ルックアップを回避するために使用されるホストキャッシュの内容へのアクセスを提供します。(セクション8.11.5.2「DNS ルックアップの最適化とホストキャッシュ」を参照してください。)host_cache テーブルは、SELECT ステートメントを使用して調査できるようにホストキャッシュの内容を公開します。パフォーマンススキーマを有効にしている必要があり、そうでなければこのテーブルは空になります。

FLUSH HOSTS と TRUNCATE TABLE host_cache は同じ効果を持ちます。それらはホストキャッシュをクリアします。これは host_cache テーブルを空にし (それはキャッシュの可視表現であるため)、ブロックされたすべてのホストのブロックも解除します (セクションB.5.2.6「ホスト 'host_name' は拒否されました」を参照してください)。FLUSH HOSTS には RELOAD 権限が必要です。TRUNCATE TABLE では host_cache テーブルへの DROP 権限が必要です。

host_cache テーブルにはこれらのカラムがあります。

- IP

  文字列で表された、サーバーに接続しているクライアントの IP アドレス。

- HOST

  そのクライアント IP の解決済みの DNS ホスト名、または名前が不明な場合は NULL。

- HOST_VALIDATED

  クライアント IP に対し、IP からホスト名、ホスト名から IP の DNS 解決が正常に実行されたかどうか。HOST_VALIDATED が YES の場合、DNS への呼び出しを回避できるように、IP に対応するホスト名として HOST カラムが使用されます。HOST_VALIDATED が NO である間、各接続について、最終的にそれが有効な結果または永続的エラーのいずれかで完了するまで、DNS 解決が試みられます。この情報により、サーバーは、クライアントに永久に影響することがある、一時的な DNS 障害発生時の不正または失われたホスト名のキャッシュを避けることができます。

- SUM_CONNECT_ERRORS

「ブロック」とみなされる接続エラーの数 (max_connect_errors システム変数に対して評価される)。現在、プロトコルハンドシェークエラーのみが、検証に合格したホスト (HOST_VALIDATED = YES) に対してのみカウントされます。

- COUNT_HOST_BLOCKED_ERRORS

  SUM_CONNECT_ERRORS が max_connect_errors システム変数の値を超えたため、ブロックされた接続の数。

- COUNT_NAMEINFO_TRANSIENT_ERRORS

  IP からホスト名への DNS 解決時の一時的なエラーの数。

- COUNT_NAMEINFO_PERMANENT_ERRORS

  IP からホスト名への DNS 解決時の永続的なエラーの数。

- COUNT_FORMAT_ERRORS

  ホスト名形式エラーの数。MySQL は、1.2.example.com など、名前の最初のコンポーネントの 1 つまたは複数がすべて数値であるホスト名に対して、mysql.user テーブル内の Host カラム値の照合を実行しません。代わりにクライアント IP アドレスが使用されます。この種類の照合が行われない理由については、セクション 6.2.3「アカウント名の指定」を参照してください。

- COUNT_ADDRINFO_TRANSIENT_ERRORS

  ホスト名から IP への逆引き DNS 解決時の一時的なエラーの数。

- COUNT_ADDRINFO_PERMANENT_ERRORS

  ホスト名から IP への逆引き DNS 解決時の永続的なエラーの数。

- COUNT_FCRDNS_ERRORS

  Forward-confirmed reverse DNS エラーの数。これらのエラーは、IP からホスト名、ホスト名から IP の DNS 解決で、クライアントの発信元の IP アドレスに一致しない IP アドレスが生成された場合に発生します。

- COUNT_HOST_ACL_ERRORS

  クライアントホストからログインできる可能性のあるユーザーがいないために発生するエラーの数。そのような場合、サーバーは ER_HOST_NOT_PRIVILEGED を返し、ユーザー名やパスワードも要求しません。

- COUNT_NO_AUTH_PLUGIN_ERRORS

  使用できない認証プラグインのリクエストによるエラーの数。プラグインが使用できない可能性があるのは、たとえば、それがロードされていないか、ロードの試みに失敗した場合です。

- COUNT_AUTH_PLUGIN_ERRORS

  認証プラグインによって報告されるエラーの数。

  認証プラグインは、障害の原因を示すために、さまざまなエラーコードを報告することがあります。エラーの種類に応じて、これらのいずれかのカラムが増分されます。COUNT_AUTHENTICATION_ERRORS、COUNT_AUTH_PLUGIN_ERRORS、COUNT_HANDSHAKE_ERRORS。新しいリターンコードは、既存のプラグイン API へのオプションの拡張です。不明または予期しないプラグインエラーは COUNT_AUTH_PLUGIN_ERRORS カラムにカウントされます。

- COUNT_HANDSHAKE_ERRORS

  有線プロトコルレベルで検出されたエラーの数。

- COUNT_PROXY_USER_ERRORS

  プロキシユーザー A が、存在しない別のユーザー B にプロキシ設定された場合に、検出されたエラーの数。

- COUNT_PROXY_USER_ACL_ERRORS

  存在はしているが、それに対してプロキシユーザー A が PROXY 権限を持たない別のユーザー B に A がプロキシ設定された場合に、検出されたエラーの数。

- COUNT_AUTHENTICATION_ERRORS

  失敗した認証によって発生したエラーの数。

- COUNT_SSL_ERRORS

  SSL の問題によるエラーの数。

- COUNT_MAX_USER_CONNECTIONS_ERRORS

  ユーザーごとの接続割り当てを超えることによって発生したエラーの数。セクション6.3.4「アカウントリソース制限の設定」を参照してください。

- COUNT_MAX_USER_CONNECTIONS_PER_HOUR_ERRORS

  時間あたりのユーザーごとの接続割り当てを超えることによって発生したエラーの数。セクション6.3.4「アカウントリソース制限の設定」を参照してください。

- COUNT_DEFAULT_DATABASE_ERRORS

  デフォルトのデータベースに関連するエラーの数。たとえば、データベースが存在していないか、ユーザーがそれにアクセスするための権限を持っていませんでした。

- COUNT_INIT_CONNECT_ERRORS

  init_connect システム変数値内のステートメントの実行の失敗によって発生したエラーの数。

- COUNT_LOCAL_ERRORS

  サーバー実装にローカルで、ネットワーク、認証、または承認に関連しないエラーの数。たとえば、メモリー不足状況はこのカテゴリに収まります。

- COUNT_UNKNOWN_ERRORS

  このテーブルのほかのカラムによって報告されない、ほかの不明なエラーの数。このカラムは、新しいエラー状況を報告する必要がある場合や、下位互換性を維持し、host_cache テーブルのテーブル構造を必要とする場合に備えて、将来使用するために予約されています。

- FIRST_SEEN

  IP カラム内のクライアントから確認された最初の接続の試みのタイムスタンプ。

- LAST_SEEN

  IP カラム内のクライアントから確認された最後の接続の試みのタイムスタンプ。

- FIRST_ERROR_SEEN

  IP カラム内のクライアントから確認された最初のエラーのタイムスタンプ。

- LAST_ERROR_SEEN

  IP カラム内のクライアントから確認された最後のエラーのタイムスタンプ。

host_cache テーブルは MySQL 5.6.5 で追加されました。

22.9.10.2 performance_timers テーブル

performance_timers テーブルは使用可能なイベントタイマーを示します。

```
mysql> SELECT * FROM performance_timers;
+-------------+-----------------+------------------+----------------+
| TIMER_NAME  | TIMER_FREQUENCY | TIMER_RESOLUTION | TIMER_OVERHEAD |
+-------------+-----------------+------------------+----------------+
| CYCLE       |      2389029850 |                1 |             72 |
| NANOSECOND  |            NULL |             NULL |           NULL |
| MICROSECOND |         1000000 |                1 |            585 |
| MILLISECOND |            1035 |                1 |            738 |
| TICK        |             101 |                1 |            630 |
+-------------+-----------------+------------------+----------------+
```

特定のタイマーに関連付けられた値が NULL の場合、そのタイマーはプラットフォームでサポートされていません。NULL を含まない行は、setup_timers で使用できるタイマーを示しています。

performance_timers テーブルにはこれらのカラムがあります。

- TIMER_NAME

  setup_timers テーブルの構成時に、タイマーを参照する名前。

- TIMER_FREQUENCY

  秒あたりのタイマーユニットの数。サイクルタイマーの場合、頻度は一般に CPU 速度に関連します。たとえば、2.4GHz プロセッサを搭載するシステムでは、CYCLE は 2400000000 に近い可能性があります。

- TIMER_RESOLUTION

  タイマー値が増加するタイマーユニット数を示します。タイマーの分解能が 10 の場合、その値は毎回 10 ずつ増加します。

- TIMER_OVERHEAD

  特定のタイマーで 1 つのタイミングを取得するためのオーバーヘッドの最小サイクル数。パフォーマンススキーマは、初期化時にタイマーを 20 回呼び出し、最小値を選択することによって、この値を決定します。インストゥルメンテーションは、各イベントの開始と終了でタイマーを呼び出すため、実際の合計のオーバーヘッドはこの量の 2 倍になります。タイマーコードは、時間付きイベントにのみ呼び出されるため、このオーバーヘッドは時間なしイベントには適用されません。

  テーブルの最大行数は、サーバー起動時に自動サイズ設定されます。この最大を明示的に設定するには、サーバー起動時に performance_schema_digests_size システム変数を設定します。

22.9.10.3 スレッドテーブル

threads テーブルは各サーバースレッドの行を格納します。各行は、スレッドに関する情報を格納し、それに対するモニタリングが有効にされているかどうかを示します。

```
mysql> SELECT * FROM threads\G
*************************** 1. row ***************************
          THREAD_ID: 1
               NAME: thread/sql/main
               TYPE: BACKGROUND
     PROCESSLIST_ID: NULL
   PROCESSLIST_USER: NULL
   PROCESSLIST_HOST: NULL
     PROCESSLIST_DB: NULL
PROCESSLIST_COMMAND: NULL
   PROCESSLIST_TIME: 80284
  PROCESSLIST_STATE: NULL
   PROCESSLIST_INFO: NULL
   PARENT_THREAD_ID: NULL
               ROLE: NULL
       INSTRUMENTED: YES
...
*************************** 4. row ***************************
          THREAD_ID: 51
               NAME: thread/sql/one_connection
               TYPE: FOREGROUND
     PROCESSLIST_ID: 34
   PROCESSLIST_USER: paul
   PROCESSLIST_HOST: localhost
     PROCESSLIST_DB: performance_schema
PROCESSLIST_COMMAND: Query
   PROCESSLIST_TIME: 0
  PROCESSLIST_STATE: Sending data
   PROCESSLIST_INFO: SELECT * FROM threads
   PARENT_THREAD_ID: 1
               ROLE: NULL
       INSTRUMENTED: YES
...
```

threads テーブルの初期の内容は、パフォーマンススキーマの初期化が行われるときに、存在しているスレッドに基づきます。その後、サーバーがスレッドを作成するたびに新しい行が追加されます。

スレッドの終了時に、threads テーブルからの行の削除が行われます。クライアントセッションに関連付けられたスレッドでは、セッションの終了時に削除が行われます。クライアントの自動再接続が有効にされており、セッションが切断後に再接続した場合、そのセッションは異なる PROCESSLIST_ID 値を持つ threads テーブル内の新しい行に関連付けられます。新しいスレッドの初期 INSTRUMENTED 値は元のスレッドの値と異なることがあ

ります。その間に setup_actors テーブルが変更されている可能性があり、元のスレッドの INSTRUMENTED 値が初期化後に変更されている場合、その変更は新しいスレッドに持ち越されません。

PROCESSLIST_ のプリフィクスのある名前を持つ threads テーブルカラムは、INFORMATION_SCHEMA.PROCESSLIST テーブルまたは SHOW PROCESSLIST ステートメントから入手できるものに似た情報を提供します。そのため、これらのすべてのソースはスレッドモニタリング情報を提供します。threads の使用は、次の点で、ほかの 2 つのソースの使用と異なります。

- threads へのアクセスには相互排他ロックは必要なく、サーバーパフォーマンスへの影響は最小です。INFORMATION_SCHEMA.PROCESSLIST と SHOW PROCESSLIST では相互排他ロックが必要になるため、パフォーマンスの低下につながります。
- threads は、スレッドがフォアグラウンドスレッドかバックグラウンドスレッドか、およびスレッドに関連付けられているサーバー内の場所などの、各スレッドの追加情報を提供します。
- threads はバックグラウンドスレッドに関する情報を提供するため、ほかのスレッド情報ソースでは不可能なアクティビティーのモニターに使用できます。
- スレッドモニタリングを有効または無効にできます (つまり、スレッドによって実行されるイベントがインストゥルメントされるかどうか)。既存のスレッドのモニタリングを制御するには、threads テーブルの INSTRUMENTED カラムを設定します。新しいフォアグラウンドスレッドの初期 INSTRUMENTED 値を制御するには、setup_actors テーブルを使用します。(スレッドモニタリングが行われる状況の詳細については、INSTRUMENTED カラムの説明を参照してください。)

これらの理由で、INFORMATION_SCHEMA.PROCESSLIST または SHOW PROCESSLIST を使用して、サーバーモニタリングを実行する DBA は、代わりに threads を使用してモニターしたいと考える場合があります。

> 注記
>
> INFORMATION_SCHEMA.PROCESSLIST および SHOW PROCESSLIST では、ほかのユーザーのスレッドに関する情報は、現在のユーザーに PROCESS 権限がある場合にのみ表示されます。これは threads テーブルには当てはまりません。テーブルの SELECT 権限を持つすべてのユーザーに、すべての行が表示されます。ほかのユーザーのスレッドを表示できないようにすべきであるユーザーには、その権限を与えないでください。

threads テーブルにはこれらのカラムがあります。

- THREAD_ID

  一意のスレッド識別子。

- NAME

  サーバーのスレッドインストゥルメンテーションコードに関連付けられている名前。たとえば、thread/sql/one_connection はユーザー接続の処理を担当するコード内のスレッド関数に対応し、thread/sql/main はサーバーの main() 関数を表します。

- TYPE

  FOREGROUND または BACKGROUND のスレッドの種類。ユーザー接続スレッドはフォアグラウンドスレッドです。内部サーバーアクティビティーに関連付けられているスレッドはバックグラウンドスレッドです。例は、内部 InnoDB スレッド、スレーブに情報を送信する 「Binlog Dump」 スレッド、スレーブ I/O および SQL スレッドです。

- PROCESSLIST_ID

  INFORMATION_SCHEMA.PROCESSLIST テーブルに表示されるスレッドの場合、これはそのテーブルの ID カラムに表示される同じ値です。それは、SHOW PROCESSLIST 出力の Id カラムに表示される値と、そのスレッド内で CONNECTION_ID() が返す値でもあります。

  バックグラウンドスレッド (ユーザー接続に関連付けられないスレッド) の場合、PROCESSLIST_ID は NULL であるため、値は一意ではありません。(MySQL 5.6.9 より前では、バックグラウンドスレッドの場合の値は 0 です。)

- PROCESSLIST_USER

  フォアグラウンドスレッドに関連付けられているユーザー、バックグラウンドスレッドの場合は NULL。

- PROCESSLIST_HOST

フォアグラウンドスレッドに関連付けられているクライアントのホスト名、バックグラウンドスレッドの場合は NULL。

- PROCESSLIST_DB

  スレッドのデフォルトのデータベース、何もない場合は NULL。

- PROCESSLIST_COMMAND

  スレッドが実行しているコマンドの種類。スレッドコマンドの説明については、セクション8.12.5「スレッド情報の検査」を参照してください。このカラムの値は、クライアント/サーバープロトコルの COM_xxx コマンドと Com_xxx ステータス変数に対応します。セクション5.1.6「サーバーステータス変数」を参照してください

- PROCESSLIST_TIME

  スレッドが現在の状態になってからの秒数。

- PROCESSLIST_STATE

  スレッドが行なっていることを示すアクション、イベント、または状態。PROCESSLIST_STATE 値の説明については、セクション8.12.5「スレッド情報の検査」を参照してください。

  ほとんどの状態がきわめてすばやい操作に対応します。スレッドが何秒間も特定の状態にとどまっている場合は、問題が発生している可能性があり、調査が必要です。

- PROCESSLIST_INFO

  スレッドが実行しているステートメント、またはそれがどのステートメントも実行していない場合は NULL。このステートメントは、サーバーに送信されるステートメント、またはこのステートメントがほかのステートメントを実行する場合は、もっとも内側のステートメントである可能性があります。たとえば、CALL ステートメントが SELECT ステートメントを実行しているストアドプロシージャーを実行する場合、PROCESSLIST_INFO 値には SELECT ステートメントが示されます。

- PARENT_THREAD_ID

  このスレッドがサブスレッド (別のスレッドによって生成される) である場合、これは生成されるスレッドの THREAD_ID 値です。スレッドの生成は、INSERT DELAYED ステートメントからの行の挿入を処理するためなどに発生します。

- ROLE

  使用されません。

- INSTRUMENTED

  スレッドがインストゥルメントされるかどうか。これは、スレッドの threads テーブル行に影響せず、スレッドによって実行されるイベントがインストゥルメントされるかどうかに影響します。

  - フォアグラウンドスレッドでは、初期 INSTRUMENTED 値は、スレッドに関連付けられているユーザーアカウントが setup_actors テーブル内の任意の行に一致するかどうかによって決定されます。照合は PROCESSLIST_USER および PROCESSLIST_HOST カラムの値に基づきます。

    スレッドがサブスレッドを生成する場合、そのサブスレッドに対して照合が再度行われます。

  - バックグラウンドスレッドの場合、INSTRUMENTED はデフォルトで YES です。バックグラウンドスレッドに関連付けられたユーザーはないため、setup_actors は参照されません。

  - どのスレッドでも、スレッドの有効期間の間にその INSTRUMENTED 値が変更されることがあります。これは、唯一の変更可能な threads テーブルカラムです。

  スレッドによって実行されるイベントのモニタリングが行われる場合、これらのことが当てはまる必要があります。

  - setup_consumers テーブル内の thread_instrumentation コンシューマは YES である必要があります。
  - thread.INSTRUMENTED カラムは YES である必要があります。
  - setup_instruments テーブル内で有効にされているインストゥルメントから生成されたスレッドイベントに対してのみ、モニタリングが行われます。

22.10 パフォーマンススキーマオプションおよび変数リファレンス

表 22.3 パフォーマンススキーマ変数リファレンス

名前	コマンド行	オプションファイル	システム変数	ステータス変数	変数スコープ	動的
performance_schema	はい	はい	はい		グローバル	いい
Performance_schema_accounts_lost				はい	グローバル	いい
performance_schema_accounts_size	はい	はい	はい		グローバル	いい
Performance_schema_cond_classes_lost				はい	グローバル	いい
Performance_schema_cond_instances_lost				はい	グローバル	いい
performance-schema-consumer-events-stages-current	はい	はい				
performance-schema-consumer-events-stages-history	はい	はい				
performance-schema-consumer-events-stages-history-long	はい	はい				
performance-schema-consumer-events-statements-current	はい	はい				
performance-schema-consumer-events-statements-history	はい	はい				
performance-schema-consumer-events-statements-history-long	はい	はい				
performance-schema-consumer-events-waits-current	はい	はい				
performance-schema-consumer-events-waits-history	はい	はい				
performance-schema-consumer-events-waits-history-long	はい	はい				
performance-schema-consumer-global-instrumentation	はい	はい				
performance-schema-consumer-statements-digest	はい	はい				
performance-schema-consumer-thread-instrumentation	はい	はい				
Performance_schema_digest_lost				はい	グローバル	いい
performance_schema_digests_size	はい	はい	はい		グローバル	いい
performance_schema_events_stages_history_long_size	はい	はい	はい		グローバル	いい
performance_schema_events_stages_history_size	はい	はい	はい		グローバル	いい
performance_schema_events_statements_history_long_size	はい	はい	はい		グローバル	いい
performance_schema_events_statements_history_size	はい	はい	はい		グローバル	いい
performance_schema_events_waits_history_long_size	はい	はい	はい		グローバル	いい

名前	コマンド行	オプションファイル	システム変数	ステータス変数	変数スコープ	動的
performance_schema_events_waits_history_size	はい	はい	はい		グローバル	いいえ
Performance_schema_file_classes_lost				はい	グローバル	いいえ
Performance_schema_file_handles_lost				はい	グローバル	いいえ
Performance_schema_file_instances_lost				はい	グローバル	いいえ
Performance_schema_hosts_lost				はい	グローバル	いいえ
performance_schema_hosts_size	はい	はい	はい		グローバル	いいえ
performance-schema-instrument	はい	はい				
Performance_schema_locker_lost				はい	グローバル	いいえ
performance_schema_max_cond_classes	はい	はい	はい		グローバル	いいえ
performance_schema_max_cond_instances	はい	はい	はい		グローバル	いいえ
performance_schema_max_file_classes	はい	はい	はい		グローバル	いいえ
performance_schema_max_file_handles	はい	はい	はい		グローバル	いいえ
performance_schema_max_file_instances	はい	はい	はい		グローバル	いいえ
performance_schema_max_mutex_classes	はい	はい	はい		グローバル	いいえ
performance_schema_max_mutex_instances	はい	はい	はい		グローバル	いいえ
performance_schema_max_rwlock_classes	はい	はい	はい		グローバル	いいえ
performance_schema_max_rwlock_instances	はい	はい	はい		グローバル	いいえ
performance_schema_max_socket_classes	はい	はい	はい		グローバル	いいえ
performance_schema_max_socket_instances	はい	はい	はい		グローバル	いいえ
performance_schema_max_stage_classes	はい	はい	はい		グローバル	いいえ
performance_schema_max_statement_classes	はい	はい	はい		グローバル	いいえ
performance_schema_max_table_handles	はい	はい	はい		グローバル	いいえ
performance_schema_max_table_instances	はい	はい	はい		グローバル	いいえ
performance_schema_max_thread_classes	はい	はい	はい		グローバル	いいえ
performance_schema_max_thread_instances	はい	はい	はい		グローバル	いいえ
Performance_schema_mutex_classes_lost				はい	グローバル	いいえ
Performance_schema_mutex_instances_lost				はい	グローバル	いいえ
Performance_schema_rwlock_classes_lost				はい	グローバル	いいえ
Performance_schema_rwlock_instances_lost				はい	グローバル	いいえ
Performance_schema_session_connect_attrs_lost				はい	グローバル	いいえ
performance_schema_session_connect_attrs_size	はい	はい	はい		グローバル	いいえ
performance_schema_setup_actors_size	はい	はい	はい		グローバル	いいえ
performance_schema_setup_objects_size	はい	はい	はい		グローバル	いいえ
Performance_schema_socket_classes_lost				はい	グローバル	いいえ
Performance_schema_socket_instances_lost				はい	グローバル	いいえ
Performance_schema_stage_classes_lost				はい	グローバル	いいえ
Performance_schema_statement_classes_lost				はい	グローバル	いいえ
Performance_schema_table_handles_lost				はい	グローバル	いいえ
Performance_schema_table_instances_lost				はい	グローバル	いいえ
Performance_schema_thread_classes_lost				はい	グローバル	いいえ
Performance_schema_thread_instances_lost				はい	グローバル	いいえ
Performance_schema_users_lost				はい	グローバル	いいえ
performance_schema_users_size	はい	はい	はい		グローバル	いいえ

22.11 パフォーマンススキーマコマンドオプション

パフォーマンススキーマパラメータは、サーバー起動時にコマンド行またはオプションファイルに指定して、パフォーマンススキーマのインストゥルメントおよびコンシューマを構成できます。多くの場合に実行時構成も可能ですが (セクション22.2.3「パフォーマンススキーマ実行時構成」 を参照)、実行時構成で、起動プロセス中にすでに初期化されているインストゥルメントに影響を与えるには遅すぎる場合は、起動時構成を使用する必要があります。

パフォーマンススキーマのコンシューマおよびインストゥルメントは、次の構文を使用して起動時に構成できます。追加の詳細については、セクション22.2.2「パフォーマンススキーマ起動構成」を参照してください。

- --performance-schema-consumer-consumer_name=value

  パフォーマンススキーマコンシューマを構成します。setup_consumers テーブル内のコンシューマ名は下線が使われますが、起動時に設定されたコンシューマでは、名前の中のダッシュと下線は同等です。各コンシューマを構成するためのオプションについては、このセクションの後半で詳しく説明します。

- --performance-schema-instrument=instrument_name=value

  パフォーマンススキーマインストゥルメントを構成します。名前をパターンとして指定して、パターンに一致するインストゥルメントを構成できます。

次の項目は各コンシューマを構成します。

- --performance-schema-consumer-events-stages-current=value

  events-stages-current コンシューマを構成します。このオプションは MySQL 5.6.4 で追加されました。

- --performance-schema-consumer-events-stages-history=value

  events-stages-history コンシューマを構成します。このオプションは MySQL 5.6.4 で追加されました。

- --performance-schema-consumer-events-stages-history-long=value

  events-stages-history-long コンシューマを構成します。このオプションは MySQL 5.6.4 で追加されました。

- --performance-schema-consumer-events-statements-current=value

  events-statements-current コンシューマを構成します。このオプションは MySQL 5.6.4 で追加されました。

- --performance-schema-consumer-events-statements-history=value

  events-statements-history コンシューマを構成します。このオプションは MySQL 5.6.4 で追加されました。

- --performance-schema-consumer-events-statements-history-long=value

  events-statements-history-long コンシューマを構成します。このオプションは MySQL 5.6.4 で追加されました。

- --performance-schema-consumer-events-waits-current=value

  events-waits-current コンシューマを構成します。このオプションは MySQL 5.6.4 で追加されました。

- --performance-schema-consumer-events-waits-history=value

  events-waits-history コンシューマを構成します。このオプションは MySQL 5.6.4 で追加されました。

- --performance-schema-consumer-events-waits-history-long=value

  events-waits-history-long コンシューマを構成します。このオプションは MySQL 5.6.4 で追加されました。

- --performance-schema-consumer-global-instrumentation=value

  global-instrumentation コンシューマを構成します。このオプションは MySQL 5.6.4 で追加されました。

- --performance-schema-consumer-statements-digest=value

  statements-digest コンシューマを構成します。このオプションは MySQL 5.6.5 で追加されました。

- --performance-schema-consumer-thread-instrumentation=value

thread-instrumentation コンシューマを構成します。このオプションは MySQL 5.6.4 で追加されました。

22.12 パフォーマンススキーマシステム変数

パフォーマンススキーマは、構成情報を提供するいくつかのシステム変数を実装しています。

```
mysql> SHOW VARIABLES LIKE 'perf%';
+----------------------------------------------------+---------+
| Variable_name                                      | Value   |
+----------------------------------------------------+---------+
| performance_schema                                 | ON      |
| performance_schema_accounts_size                   | 100     |
| performance_schema_digests_size                    | 200     |
| performance_schema_events_stages_history_long_size | 10000   |
| performance_schema_events_stages_history_size      | 10      |
| performance_schema_events_statements_history_long_size | 10000   |
| performance_schema_events_statements_history_size  | 10      |
| performance_schema_events_waits_history_long_size  | 10000   |
| performance_schema_events_waits_history_size       | 10      |
| performance_schema_hosts_size                      | 100     |
| performance_schema_max_cond_classes                | 80      |
| performance_schema_max_cond_instances              | 1000    |
| performance_schema_max_file_classes                | 50      |
| performance_schema_max_file_handles                | 32768   |
| performance_schema_max_file_instances              | 10000   |
| performance_schema_max_mutex_classes               | 200     |
| performance_schema_max_mutex_instances             | 1000000 |
| performance_schema_max_rwlock_classes              | 30      |
| performance_schema_max_rwlock_instances            | 1000000 |
| performance_schema_max_socket_classes              | 10      |
| performance_schema_max_socket_instances            | 1000    |
| performance_schema_max_stage_classes               | 150     |
| performance_schema_max_statement_classes           | 165     |
| performance_schema_max_table_handles               | 10000   |
| performance_schema_max_table_instances             | 1000    |
| performance_schema_max_thread_classes              | 50      |
| performance_schema_max_thread_instances            | 1000    |
| performance_schema_session_connect_attrs_size      | 512     |
| performance_schema_setup_actors_size               | 100     |
| performance_schema_setup_objects_size              | 100     |
| performance_schema_users_size                      | 100     |
+----------------------------------------------------+---------+
```

パフォーマンススキーマシステム変数は、コマンド行またはオプションファイルでサーバーの起動時に設定でき、多くは実行時に設定できます。セクション22.10「パフォーマンススキーマオプションおよび変数リファレンス」を参照してください。

MySQL 5.6.6 以降、パフォーマンススキーマは、そのパラメータのいくつかの値を、それらが明示的に設定されていない場合に、サーバーの起動時に自動的にサイズ設定します。詳細については、セクション22.2.2「パフォーマンススキーマ起動構成」を参照してください。

パフォーマンススキーマシステム変数には次の意味があります。

- performance_schema

コマンド行形式	--performance-schema=#
システム変数	performance_schema
スコープ	グローバル
動的	いいえ
型	ブール
デフォルト (≥ 5.6.6)	ON
デフォルト (≤ 5.6.5)	OFF

この変数の値は ON または OFF で、パフォーマンススキーマが有効にされているかどうかを示します。デフォルトで、この値は MySQL 5.6.6 以降 ON で、それより前では OFF です。サーバーの起動時に、この変数を値なし、またはそれを有効にする ON または 1 の値、またはそれを無効にする OFF または 0 の値で指定できます。

- performance_schema_accounts_size

コマンド行形式	--performance-schema-accounts-size=#
導入	5.6.3
システム変数	performance_schema_accounts_size
スコープ	グローバル
動的	いいえ
型	数値
デフォルト (≥ 5.6.6)	-1 (autosized)
デフォルト (≤ 5.6.5)	10
最小値 (≥ 5.6.6)	-1
最小値 (≤ 5.6.5)	0
最大値	1048576

accounts テーブルの行数。この変数が 0 の場合、パフォーマンススキーマは accounts テーブル内の接続統計を保守しません。この変数は MySQL 5.6.3 で追加されました。

- performance_schema_digests_size

コマンド行形式	--performance-schema-digests-size=#
導入	5.6.5
システム変数	performance_schema_digests_size
スコープ	グローバル
動的	いいえ
型	数値
デフォルト	-1 (autosized)
最小値	-1
最大値	1048576

events_statements_summary_by_digest テーブル内の最大行数。この変数は MySQL 5.6.5 で追加されました。この最大を超えたため、ダイジェストをインストゥルメントできない場合、パフォーマンススキーマは Performance_schema_digest_lost ステータス変数を増分します。

- performance_schema_events_stages_history_long_size

コマンド行形式	--performance-schema-events-stages-history-long-size=#
導入	5.6.3
システム変数	performance_schema_events_stages_history_long_size
スコープ	グローバル
動的	いいえ
型	数値
デフォルト (≥ 5.6.6)	-1 (autosized)
デフォルト (≤ 5.6.5)	10000

events_stages_history_long テーブルの行数。この変数は MySQL 5.6.3 で追加されました。

- performance_schema_events_stages_history_size

コマンド行形式	--performance-schema-events-stages-history-size=#
導入	5.6.3
システム変数	performance_schema_events_stages_history_size
スコープ	グローバル
動的	いいえ

型	数値
デフォルト (≥ 5.6.6)	-1 (autosized)
デフォルト (≤ 5.6.5)	10

events_stages_history テーブルのスレッドあたりの行数。この変数は MySQL 5.6.3 で追加されました。

- performance_schema_events_statements_history_long_size

コマンド行形式	--performance-schema-events-statements-history-long-size=#
導入	5.6.3
システム変数	performance_schema_events_statements_history_long_size
スコープ	グローバル
動的	いいえ
型	数値
デフォルト (≥ 5.6.6)	-1 (autosized)
デフォルト (≤ 5.6.5)	10000

events_statements_history_long テーブル内の行数。この変数は MySQL 5.6.3 で追加されました。

- performance_schema_events_statements_history_size

コマンド行形式	--performance-schema-events-statements-history-size=#
導入	5.6.3
システム変数	performance_schema_events_statements_history_size
スコープ	グローバル
動的	いいえ
型	数値
デフォルト (≥ 5.6.6)	-1 (autosized)
デフォルト (≤ 5.6.5)	10

events_statements_history テーブル内のスレッドあたりの行数。この変数は MySQL 5.6.3 で追加されました。

- performance_schema_events_waits_history_long_size

コマンド行形式	--performance-schema-events-waits-history-long-size=#
システム変数	performance_schema_events_waits_history_long_size
スコープ	グローバル
動的	いいえ
型	数値
デフォルト (≥ 5.6.6)	-1 (autosized)
デフォルト (≤ 5.6.5)	10000

events_waits_history_long テーブル内の行数。

- performance_schema_events_waits_history_size

コマンド行形式	--performance-schema-events-waits-history-size=#
システム変数	performance_schema_events_waits_history_size
スコープ	グローバル
動的	いいえ
型	数値
デフォルト (≥ 5.6.6)	-1 (autosized)
デフォルト (≤ 5.6.5)	10

events_waits_history テーブル内のスレッドあたりの行数。

- performance_schema_hosts_size

コマンド行形式	--performance-schema-hosts-size=#
導入	5.6.3
システム変数	performance_schema_hosts_size
スコープ	グローバル
動的	いいえ
型	数値
デフォルト (≥ 5.6.6)	-1 (autosized)
デフォルト (≤ 5.6.5)	10
最小値 (≥ 5.6.6)	-1
最小値 (≤ 5.6.5)	0
最大値	1048576

hosts テーブル内の行数。この変数が 0 の場合、パフォーマンススキーマは hosts テーブル内の接続統計を保守しません。この変数は MySQL 5.6.3 で追加されました。

- performance_schema_max_cond_classes

コマンド行形式	--performance-schema-max-cond-classes=#
システム変数	performance_schema_max_cond_classes
スコープ	グローバル
動的	いいえ
型	数値
デフォルト	80

条件インストゥルメントの最大数。

- performance_schema_max_cond_instances

コマンド行形式	--performance-schema-max-cond-instances=#
システム変数	performance_schema_max_cond_instances
スコープ	グローバル
動的	いいえ
型	数値
デフォルト (≥ 5.6.6)	-1 (autosized)
デフォルト (≤ 5.6.5)	1000

インストゥルメントされた条件オブジェクトの最大数。

- performance_schema_max_file_classes

コマンド行形式	--performance-schema-max-file-classes=#
システム変数	performance_schema_max_file_classes
スコープ	グローバル
動的	いいえ
型	数値
デフォルト	50

ファイルインストゥルメントの最大数。

- performance_schema_max_file_handles

コマンド行形式	--performance-schema-max-file-handles=#
システム変数	performance_schema_max_file_handles
スコープ	グローバル
動的	いいえ
型	数値
デフォルト	32768

オープンしているファイルオブジェクトの最大数。

performance_schema_max_file_handles の値は、open_files_limit の値より大きくしてください。open_files_limit は、サーバーがサポート可能なオープンファイルハンドルの最大数に影響し、performance_schema_max_file_handles はこれらのファイルハンドルのうちインストゥルメント可能な数に影響します。

- performance_schema_max_file_instances

コマンド行形式	--performance-schema-max-file-instances=#
システム変数	performance_schema_max_file_instances
スコープ	グローバル
動的	いいえ
型	数値
デフォルト (≥ 5.6.6)	-1 (autosized)
デフォルト (≤ 5.6.5)	10000

インストゥルメントされるファイルオブジェクトの最大数。

- performance_schema_max_mutex_classes

コマンド行形式	--performance-schema-max-mutex-classes=#
システム変数	performance_schema_max_mutex_classes
スコープ	グローバル
動的	いいえ
型	数値
デフォルト	200

相互排他ロックインストゥルメントの最大数。

- performance_schema_max_mutex_instances

コマンド行形式	--performance-schema-max-mutex-instances=#
システム変数	performance_schema_max_mutex_instances
スコープ	グローバル
動的	いいえ
型	数値
デフォルト (≥ 5.6.6)	-1 (autosized)
デフォルト (≤ 5.6.5)	1000

インストゥルメントされる相互排他ロックオブジェクトの最大数。

- performance_schema_max_rwlock_classes

コマンド行形式	--performance-schema-max-rwlock-classes=#
システム変数	performance_schema_max_rwlock_classes
スコープ	グローバル

動的	いいえ
型	数値
デフォルト (≥ 5.6.15)	40
デフォルト (≥ 5.6.1, ≤ 5.6.14)	30
デフォルト (5.6.0)	20

rwlock インストゥルメントの最大数。

- performance_schema_max_rwlock_instances

コマンド行形式	--performance-schema-max-rwlock-instances=#
システム変数	performance_schema_max_rwlock_instances
スコープ	グローバル
動的	いいえ
型	数値
デフォルト (≥ 5.6.6)	-1 (autosized)
デフォルト (≤ 5.6.5)	1000

インストゥルメントされる rwlock オブジェクトの最大数。

- performance_schema_max_socket_classes

コマンド行形式	--performance-schema-max-socket-classes=#
導入	5.6.3
システム変数	performance_schema_max_socket_classes
スコープ	グローバル
動的	いいえ
型	数値
デフォルト	10

ソケットインストゥルメントの最大数。この変数は MySQL 5.6.3 で追加されました。

- performance_schema_max_socket_instances

コマンド行形式	--performance-schema-max-socket-instances=#
導入	5.6.3
システム変数	performance_schema_max_socket_instances
スコープ	グローバル
動的	いいえ
型	数値
デフォルト (≥ 5.6.6)	-1 (autosized)
デフォルト (≤ 5.6.5)	1000

インストゥルメントされるソケットオブジェクトの最大数。この変数は MySQL 5.6.3 で追加されました。

- performance_schema_max_stage_classes

コマンド行形式	--performance-schema-max-stage-classes=#
導入	5.6.3
システム変数	performance_schema_max_stage_classes
スコープ	グローバル
動的	いいえ
型	数値

デフォルト	150

ステージインストゥルメントの最大数。この変数は MySQL 5.6.3 で追加されました。

- performance_schema_max_statement_classes

コマンド行形式	--performance-schema-max-statement-classes=#
導入	5.6.3
システム変数	performance_schema_max_statement_classes
スコープ	グローバル
動的	いいえ
型	数値
デフォルト	(autosized)

ステートメントインストゥルメントの最大数。デフォルト値は、サーバー構築時に、クライアント/サーバープロトコルのコマンド数とサーバーでサポートされる SQL ステートメントの種類の数に基づいて計算されます。

この変数は、それを 0 に設定して、すべてのステートメントインストゥルメンテーションを無効にし、それに関連付けられているすべてのメモリーを節約しないかぎり、変更するべきではありません。変数をデフォルトでない 0 以外の値に設定してもメリットはありません。特にデフォルトより大きい値は、必要以上のメモリーが割り当てられます。

この変数は MySQL 5.6.3 で追加されました。

- performance_schema_max_table_handles

コマンド行形式	--performance-schema-max-table-handles=#
システム変数	performance_schema_max_table_handles
スコープ	グローバル
動的	いいえ
型	数値
デフォルト (≥ 5.6.6)	-1 (autosized)
デフォルト (≤ 5.6.5)	100000

オープンしているテーブルオブジェクトの最大数。

- performance_schema_max_table_instances

コマンド行形式	--performance-schema-max-table-instances=#
システム変数	performance_schema_max_table_instances
スコープ	グローバル
動的	いいえ
型	数値
デフォルト (≥ 5.6.6)	-1 (autosized)
デフォルト (≤ 5.6.5)	50000

インストゥルメントされるテーブルオブジェクトの最大数。

- performance_schema_max_thread_classes

コマンド行形式	--performance-schema-max-thread-classes=#
システム変数	performance_schema_max_thread_classes
スコープ	グローバル
動的	いいえ
型	数値

デフォルト	50

スレッドインストゥルメントの最大数。

- performance_schema_max_thread_instances

コマンド行形式	--performance-schema-max-thread-instances=#
システム変数	performance_schema_max_thread_instances
スコープ	グローバル
動的	いいえ
型	数値
デフォルト (≥ 5.6.6)	-1 (autosized)
デフォルト (≤ 5.6.5)	1000

インストゥルメントされるスレッドオブジェクトの最大数。この値は threads テーブルのサイズを制御します。この最大を超えたため、スレッドをインストゥルメントできない場合、パフォーマンススキーマは Performance_schema_thread_instances_lost ステータス変数を増分します。

max_connections システム変数はサーバーで実行するスレッド数に影響します。performance_schema_max_thread_instances はこれらの実行中のスレッドのうちインストゥルメント可能なスレッド数に影響します。performance_schema_max_thread_instances のデフォルト値は、max_connections の値に基づいて自動サイズ設定されます。

- performance_schema_session_connect_attrs_size

コマンド行形式	--performance-schema-session-connect-attrs-size=#
導入	5.6.6
システム変数	performance_schema_session_connect_attrs_size
スコープ	グローバル
動的	いいえ
型	数値
デフォルト	-1 (autosized)
最小値	-1
最大値	1048576

接続属性文字列を保持するために使用される、スレッドごとに事前割り当てされるメモリーの量。接続属性文字列が予約されたストレージより大きい場合、Performance_schema_session_connect_attrs_lost ステータス変数が増分されます。この変数は MySQL 5.6.7 で追加されました。

- performance_schema_setup_actors_size

コマンド行形式	--performance-schema-setup-actors-size=#
導入	5.6.1
システム変数	performance_schema_setup_actors_size
スコープ	グローバル
動的	いいえ
型	数値
デフォルト	100

setup_actors テーブル内の行数。

- performance_schema_setup_objects_size

コマンド行形式	--performance-schema-setup-objects-size=#
導入	5.6.1
システム変数	performance_schema_setup_objects_size

スコープ	グローバル
動的	いいえ
型	数値
デフォルト	100

setup_objects テーブル内の行数。

- performance_schema_users_size

コマンド行形式	--performance-schema-users-size=#
導入	5.6.3
システム変数	performance_schema_users_size
スコープ	グローバル
動的	いいえ
型	数値
デフォルト (≥ 5.6.6)	-1 (autosized)
デフォルト (≤ 5.6.5)	10
最小値 (≥ 5.6.6)	-1
最小値 (≤ 5.6.5)	0
最大値	1048576

users テーブル内の行数。この変数が 0 の場合、パフォーマンススキーマは users テーブル内の接続統計を保守しません。この変数は MySQL 5.6.3 で追加されました。

22.13 パフォーマンススキーマステータス変数

パフォーマンススキーマは、メモリー制約のためロードまたは作成できなかったインストゥルメンテーションついての情報を提供するいくつかのステータス変数を実装しています。

```
mysql> SHOW STATUS LIKE 'perf%';
+-------------------------------------------+-------+
| Variable_name                             | Value |
+-------------------------------------------+-------+
| Performance_schema_accounts_lost          | 0     |
| Performance_schema_cond_classes_lost      | 0     |
| Performance_schema_cond_instances_lost    | 0     |
| Performance_schema_file_classes_lost      | 0     |
| Performance_schema_file_handles_lost      | 0     |
| Performance_schema_file_instances_lost    | 0     |
| Performance_schema_hosts_lost             | 0     |
| Performance_schema_locker_lost            | 0     |
| Performance_schema_mutex_classes_lost     | 0     |
| Performance_schema_mutex_instances_lost   | 0     |
| Performance_schema_rwlock_classes_lost    | 0     |
| Performance_schema_rwlock_instances_lost  | 0     |
| Performance_schema_socket_classes_lost    | 0     |
| Performance_schema_socket_instances_lost  | 0     |
| Performance_schema_stage_classes_lost     | 0     |
| Performance_schema_statement_classes_lost | 0     |
| Performance_schema_table_handles_lost     | 0     |
| Performance_schema_table_instances_lost   | 0     |
| Performance_schema_thread_classes_lost    | 0     |
| Performance_schema_thread_instances_lost  | 0     |
| Performance_schema_users_lost             | 0     |
+-------------------------------------------+-------+
```

パフォーマンススキーマステータス変数には次の意味があります。

- Performance_schema_accounts_lost

  accounts テーブルがいっぱいであったため、行を追加できなかった回数。この変数は MySQL 5.6.3 で追加されました。

- Performance_schema_cond_classes_lost

ロードできなかった条件インストゥルメントの数。

- Performance_schema_cond_instances_lost

  作成できなかった条件インストゥルメントインスタンスの数。

- Performance_schema_digest_lost

  events_statements_summary_by_digest テーブルにインストゥルメントできなかったダイジェストインスタンスの数。performance_schema_digests_size の値が小さすぎる場合、これは 0 以外になることがあります。この変数は MySQL 5.6.5 で追加されました。

- Performance_schema_file_classes_lost

  ロードできなかったファイルインストゥルメントの数。

- Performance_schema_file_handles_lost

  オープンできなかったファイルインストゥルメントインスタンスの数。

- Performance_schema_file_instances_lost

  作成できなかったファイルインストゥルメントインスタンスの数。

- Performance_schema_hosts_lost

  hosts テーブルがいっぱいであったため、行を追加できなかった回数。この変数は MySQL 5.6.3 で追加されました。

- Performance_schema_locker_lost

  次の状況のため、「失われる」か記録されないイベント数。

  - イベントが再帰的です (たとえば、A を待機することによって B で待機が発生し、それによって C で待機が発生したなど)。
  - ネストしたイベントスタックの深さが、実装によって課せられる制限より大きいです。

  現在、パフォーマンススキーマによって記録されるイベントは再帰的でないため、この変数は常に 0 になるはずです。

- Performance_schema_mutex_classes_lost

  ロードできなかった相互排他ロックインストゥルメントの数。

- Performance_schema_mutex_instances_lost

  作成できなかった相互排他ロックインストゥルメントインスタンスの数。

- Performance_schema_rwlock_classes_lost

  ロードできなかった rwlock インストゥルメントの数。

- Performance_schema_rwlock_instances_lost

  作成できなかった rwlock インストゥルメントインスタンスの数。

- Performance_schema_session_connect_attrs_lost

  接続属性文字列が予約されたストレージより大きかった回数。この変数は MySQL 5.6.7 で追加されました。

- Performance_schema_socket_classes_lost

  ロードできなかったソケットインストゥルメントの数。この変数は MySQL 5.6.3 で追加されました。

- Performance_schema_socket_instances_lost

  作成できなかったソケットインストゥルメントインスタンスの数。この変数は MySQL 5.6.3 で追加されました。

- Performance_schema_stage_classes_lost

  ロードできなかったステージインストゥルメントの数。この変数は MySQL 5.6.3 で追加されました。

- Performance_schema_statement_classes_lost

  ロードできなかったステートメントインストゥルメントの数。この変数は MySQL 5.6.3 で追加されました。

- Performance_schema_table_handles_lost

  オープンできなかったテーブルインストゥルメントインスタンスの数。

- Performance_schema_table_instances_lost

  作成できなかったテーブルインストゥルメントインスタンスの数。

- Performance_schema_thread_classes_lost

  ロードできなかったスレッドインストゥルメントの数。

- Performance_schema_thread_instances_lost

  threads テーブルにインストゥルメントできなかったスレッドインスタンスの数。performance_schema_max_thread_instances の値が小さすぎる場合、これは 0 以外になることがあります。

- Performance_schema_users_lost

  users テーブルがいっぱいであったため、行を追加できなかった回数。この変数は MySQL 5.6.3 で追加されました。

これらの変数を使用して、パフォーマンススキーマのステータスをチェックすることについては、セクション 22.5「パフォーマンススキーマステータスモニタリング」を参照してください。

22.14 パフォーマンススキーマとプラグイン

UNINSTALL PLUGIN によってプラグインを削除することは、そのプラグインのコードですでに収集された情報には影響しません。プラグインのロード中にコードの実行に費やされた時間は、プラグインがあとでアンロードされた場合でも費やされます。集計情報を含む関連付けられたイベント情報は、performance_schema データベーステーブルで読み取り可能なままになります。プラグインのインストールと削除の効果の追加情報については、セクション22.5「パフォーマンススキーマステータスモニタリング」を参照してください。

プラグインコードをインストゥルメントするプラグイン実装者は、プラグインをロードするユーザーがその要件を把握できるように、インストゥルメンテーションの特性をドキュメント化してください。たとえば、サードパーティーストレージエンジンでは、そのドキュメントに、エンジンが相互排他ロックおよびその他のインストゥルメントに必要とするメモリーの量を記載してください。

22.15 問題を診断するためのパフォーマンススキーマの使用

パフォーマンススキーマは、DBA が「大まかな推量」ではなく、実際に測定して、パフォーマンスのチューニングを行うのに役立つツールです。このセクションでは、この目的でパフォーマンススキーマを使用するいくつかの方法について説明します。ここでの説明は、セクション22.2.3.2「パフォーマンススキーマイベントフィルタリング」で説明しているイベントフィルタリングの使用に依存します。

次の例では、パフォーマンスボトルネックの調査など、反復される問題の分析に使用できる 1 つの方法を示しています。開始するには、パフォーマンスが「遅すぎる」とみなされ、最適化が必要な反復可能なユースケースが必要であり、すべてのインストゥルメンテーションを有効にすべきです (事前フィルタリングなし)。

1. ユースケースを実行します。
2. パフォーマンススキーマテーブルを使用して、パフォーマンスの問題の原因を分析します。この分析は事後フィルタリングに大きく依存します。
3. 除外する問題領域については、対応するインストゥルメントを無効にします。たとえば、問題が特定のストレージエンジンのファイル I/O に関連していないことが分析に示されている場合、そのエンジンのファイル I/O インストゥルメントを無効にします。次に、履歴およびサマリーテーブルを切り捨て、これまでに収集されたイベントを削除します。

4. ステップ 1 のプロセスを繰り返します。

   繰り返しのたびに、パフォーマンススキーマの出力、特に events_waits_history_long テーブルに格納される、無意味なインストゥルメントによって発生する「ノイズ」が減っていき、このテーブルが固定のサイズである場合、当面の問題の分析に関連するデータが増えていきます。

   繰り返しのたびに、調査は問題の原因に近づいていくはずであり、「シグナル/ノイズ」率が改善するにつれて、分析が簡単になります。

5. パフォーマンスボトルネックの原因を突き止めたら、次のような適切な修正措置をとります。

   - サーバーパラメータ (キャッシュサイズ、メモリーなど) をチューニングします。
   - クエリーを違う方法で書いてチューニングします。
   - データベーススキーマ (テーブル、インデックスなど) をチューニングします。
   - コードをチューニングします (これはストレージエンジンまたはサーバー開発者のみに適用されます)。

6. ステップ 1 から再開して、パフォーマンスへの変更の効果を確認します。

mutex_instances.LOCKED_BY_THREAD_ID および rwlock_instances.WRITE_LOCKED_BY_THREAD_ID カラムは、パフォーマンスボトルネックまたはデッドロックの調査にきわめて重要です。これは、次のようなパフォーマンススキーマインストゥルメンテーションによって可能になります。

1. スレッド 1 は相互排他ロックを待機してスタックしているとします。
2. スレッドが何を待機しているかを特定できます。

   ```
   SELECT * FROM events_waits_current WHERE THREAD_ID = thread_1;
   ```

   たとえば、events_waits_current.OBJECT_INSTANCE_BEGIN にあるように、スレッドが相互排他ロック A を待機していることがクエリー結果に示されます。

3. 相互排他ロック A を保持しているスレッドを特定できます。

   ```
   SELECT * FROM mutex_instances WHERE OBJECT_INSTANCE_BEGIN = mutex_A;
   ```

   たとえば、mutex_instances.LOCKED_BY_THREAD_ID にあるように、相互排他ロック A を保持しているのがスレッド 2 であることがクエリー結果に示されます。

4. スレッド 2 が何を実行しているかを確認できます。

   ```
   SELECT * FROM events_waits_current WHERE THREAD_ID = thread_2;
   ```

第 23 章 Connector および API

目次

23.1 MySQL Connector/ODBC 2656
23.2 MySQL Connector/Net 2656
23.3 MySQL Connector/J 2656
23.4 MySQL Connector/C++ 2657
23.5 MySQL Connector/Python 2657
23.6 組み込み MySQL サーバーライブラリ libmysqld 2657
    23.6.1 libmysqld によるプログラムのコンパイル 2657
    23.6.2 組み込み MySQL サーバーを使用する場合の制限 2658
    23.6.3 組み込みサーバーとオプション 2658
    23.6.4 組み込みサーバーの例 2658
23.7 MySQL C API 2661
    23.7.1 MySQL C API の実装 2662
    23.7.2 MySQL サーバーと MySQL Connector/C の同時インストール 2663
    23.7.3 C API クライアントプログラムの例 2664
    23.7.4 C API クライアントプログラムの構築と実行 2664
    23.7.5 C API データ構造 2667
    23.7.6 C API 関数の概要 2671
    23.7.7 C API 関数の説明 2675
    23.7.8 C API プリペアドステートメント 2722
    23.7.9 C API プリペアドステートメントデータ構造 2722
    23.7.10 C API プリペアドステートメント関数の概要 2727
    23.7.11 C API プリペアドステートメント関数の説明 2730
    23.7.12 C API スレッド関数の説明 2750
    23.7.13 C API 組み込みサーバー関数の説明 2751
    23.7.14 C API クライアントプラグイン関数 2752
    23.7.15 C API を使用する場合の一般的な質問と問題 2754
    23.7.16 自動再接続動作の制御 2756
    23.7.17 複数ステートメント実行の C API サポート 2757
    23.7.18 C API プリペアドステートメントの問題 2758
    23.7.19 C API プリペアドステートメントの日時値の処理 2759
    23.7.20 C API のプリペアド CALL ステートメントのサポート 2760
23.8 MySQL PHP API 2763
23.9 MySQL Perl API 2763
23.10 MySQL Python API 2764
23.11 MySQL Ruby API 2764
    23.11.1 MySQL/Ruby API 2764
    23.11.2 Ruby/MySQL API 2765
23.12 MySQL Tcl API 2765
23.13 MySQL Eiffel ラッパー 2765

MySQL Connector はクライアントプログラムに MySQL サーバーへの接続を提供します。API は MySQL プロトコルおよび MySQL リソースへの低レベルアクセスを提供します。Connector と API のどちらも、ODBC、Java (JDBC)、Perl、Python、PHP、Ruby、およびネイティブ C と組み込み MySQL インスタンスなどのほかの言語や環境から、MySQL ステートメントに接続し、実行することができます。

> **注記**
>
> Connector バージョン番号は MySQL Server バージョン番号と関係がありません。表 23.2「MySQL Connector のバージョンと MySQL Server のバージョン」を参照してください。

MySQL Connector

Oracle では多数のコネクタを開発しています。

- Connector/ODBC は ODBC (Open Database Connectivity) API を使用して、MySQL に接続するためのドライバサポートを提供します。サポートは、Windows、Unix、および OS X プラットフォームからの ODBC 接続で使用できます。

- Connector/Net により、開発者は MySQL に接続する .NET アプリケーションを作成できます。Connector/Net は完全に機能する ADO.NET インタフェースを実装し、ADO.NET 対応ツールで使用するためのサポートを提供します。Connector/Net を使用するアプリケーションは、サポートされる任意の .NET 言語で書くことができます。

  MySQL Visual Studio Plugin は Connector/Net および Visual Studio 2005 と連携します。このプラグインは MySQL DDEX プロバイダであり、これは Visual Studio で使用可能なスキーマおよびデータ操作ツールを使用して、MySQL データベース内にオブジェクトを作成し、編集できることを意味します。

- Connector/J は標準 JDBC (Java Database Connectivity) API を使用して、Java アプリケーションから、MySQL に接続するためのドライバサポートを提供します。

- Connector/Python は Python DB API バージョン 2.0 に準拠する API を使用して、Python アプリケーションから、MySQL に接続するためのドライバサポートを提供します。追加の Python モジュールまたは MySQL クライアントライブラリは必要ありません。

- Connector/C++ により、C++ アプリケーションは MySQL に接続できます。

MySQL C API

C アプリケーション内で、MySQL をネイティブに使用することへの直接のアクセスには、2 つの方法があります。

- C API は、libmysqlclient クライアントライブラリ経由で、MySQL クライアント/サーバープロトコルへの低レベルアクセスを提供します。これは、MySQL サーバーのインスタンスに接続するために使用する主な方法で、MySQL コマンド行クライアントと、ここで詳しく説明している多くの MySQL Connector およびサードパーティー API のどちらにも使用されています。

- libmysqld は MySQL サーバーのインスタンスを C アプリケーションに組み込むことができる組み込み MySQL サーバーライブラリです。

  libmysqld は MySQL 配布に含まれますが。

セクション23.7.1「MySQL C API の実装」も参照してください。

C アプリケーションから MySQL にアクセスするか、この章の Connector や API でサポートされていない言語で、MySQL へのインタフェースを構築するには、C API から始めます。このプロセスに役立つ、多くのプログラマ向けユーティリティーがあります。セクション4.7「MySQL プログラム開発ユーティリティー」を参照してください。

サードパーティー MySQL API

この章で説明している残りの API は、特定のアプリケーション言語から MySQL へのインタフェースを提供します。これらのサードパーティーソリューションは Oracle で開発されていないか、サポートされていません。それらの使用と機能に関する基本情報は、参考目的でのみここで提供しています。

すべてのサードパーティー言語 API は、libmysqlclient を使用するか、または ネイティブドライバを実装するか、2 つの方法のいずれかを使用して開発されています。2 つのソリューションには異なるメリットがあります。

- libmysqlclient は MySQL クライアントアプリケーションと同じライブラリを使用するため、MySQL と完全に互換性があります。ただし、機能セットは、libmysqlclient から公開された実装とインタフェースに制限され、データがネイティブ言語と MySQL API コンポーネント間でコピーされるため、パフォーマンスが低下することがあります。

- ネイティブドライバはホスト言語または環境内に完全に収まる MySQL ネットワークプロトコルの実装です。ネイティブドライバはコンポーネント間でのデータのコピーが少ないため高速であり、標準 MySQL API によって使用できない高度な機能を提供できます。さらに、ネイティブドライバコンポーネントの構築には、MySQL クライアントライブラリのコピーが必要ないため、ネイティブドライバは、エンドユーザーにとって構築とデプロイが簡単です。

表23.1「MySQL API およびインタフェース」に MySQL で使用可能な多くのライブラリとインタフェースを一覧表示しています。表23.2「MySQL Connector のバージョンと MySQL Server のバージョン」に、各コネクタがサポートする MySQL サーバーのバージョンを示しています。

表 23.1 MySQL API およびインタフェース

環境	API	型	メモ
Ada	GNU Ada MySQL バインディング	libmysqlclient	GNU Ada 用の MySQL バインディングに関するドキュメントを参照してください。

環境	API	型	メモ
C	C API	libmysqlclient	セクション23.7「MySQL C API」を参照してください。
C++	Connector/C++	libmysqlclient	「MySQL Connector/C++ 8.0 Developer Guide」を参照してください。
	MySQL++	libmysqlclient	MySQL++ の Web サイトを参照してください。
	MySQL wrapped	libmysqlclient	「MySQL wrapped」を参照してください。
Cocoa	MySQL-Cocoa	libmysqlclient	Objective-C Cocoa 環境と互換性があります。http://mysql-cocoa.sourceforge.net/ を参照してください。
D	MySQL for D	libmysqlclient	MySQL for D を参照してください。
Eiffel	Eiffel MySQL	libmysqlclient	セクション23.13「MySQL Eiffel ラッパー」を参照してください。
Erlang	erlang-mysql-driver	libmysqlclient	「erlang-mysql-driver」を参照してください。
Haskell	Haskell MySQL バインディング	ネイティブドライバ	Brian O'Sullivan のピュア Haskell MySQL バインディングに関するドキュメントを参照してください。
	hsql-mysql	libmysqlclient	Haskell 用の MySQL ドライバに関するドキュメントを参照してください。
Java/ JDBC	Connector/J	ネイティブドライバ	「MySQL Connector/J 5.1 Developer Guide」を参照してください。
Kaya	MyDB	libmysqlclient	MyDB に関するドキュメントを参照してください。
Lua	LuaSQL	libmysqlclient	LuaSQL に関するドキュメントを参照してください。
.NET/ Mono	Connector/Net	ネイティブドライバ	「MySQL Connector/NET Developer Guide」を参照してください。
Objective Caml	OBjective Caml MySQL バインディング	libmysqlclient	Objective Caml 用の MySQL バインディングに関するドキュメントを参照してください。
Octave	GNU Octave 用データベースバインディング	libmysqlclient	GNU Octave 用データベースバインディングに関するドキュメントを参照してください。
ODBC	Connector/ODBC	libmysqlclient	「MySQL Connector/ODBC Developer Guide」を参照してください。
Perl	DBI/DBD::mysql	libmysqlclient	セクション23.9「MySQL Perl API」を参照してください。
	Net::MySQL	ネイティブドライバ	CPAN の「Net::MySQL」を参照してください。
PHP	mysql、ext/mysql インタフェース (非推奨)	libmysqlclient	「Original MySQL API」を参照してください。
	mysqli、ext/mysqli インタフェース	libmysqlclient	「MySQL Improved Extension」を参照してください。
	PDO_MYSQL	libmysqlclient	「MySQL Functions (PDO_MYSQL)」を参照してください。
	PDO mysqlnd	ネイティブドライバ	
Python	Connector/Python	ネイティブドライバ	「MySQL Connector/Python Developer Guide」を参照してください。
	MySQLdb	libmysqlclient	セクション23.10「MySQL Python API」を参照してください。

環境	API	型	メモ
Ruby	MySQL/Ruby	libmysqlclient	libmysqlclient を使用します。セクション 23.11.1「MySQL/Ruby API」を参照してください。
	Ruby/MySQL	ネイティブドライバ	セクション23.11.2「Ruby/MySQL API」を参照してください。
Scheme	Myscsh	libmysqlclient	Myscsh に関するドキュメントを参照してください。
SPL	sql_mysql	libmysqlclient	SPL の sql_mysql を参照してください。
Tcl	MySQLtcl	libmysqlclient	セクション23.12「MySQL Tcl API」を参照してください。

表 23.2 MySQL Connector のバージョンと MySQL Server のバージョン

Connector	Connector のバージョン	MySQL Server のバージョン
Connector/C	6.1.0 GA	5.6、5.5、5.1、5.0、4.1
Connector/C++	1.0.5 GA	5.6、5.5、5.1
Connector/J	5.1.8	5.6、5.5、5.1、5.0、4.1
Connector/Net	6.5	5.6、5.5、5.1、5.0
Connector/Net	6.4	5.6、5.5、5.1、5.0
Connector/Net	6.3	5.6、5.5、5.1、5.0
Connector/Net	6.2 (サポートされなくなりました)	5.6、5.5、5.1、5.0
Connector/Net	6.1 (サポートされなくなりました)	5.6、5.5、5.1、5.0
Connector/Net	6.0 (サポートされなくなりました)	5.6、5.5、5.1、5.0
Connector/Net	5.2 (サポートされなくなりました)	5.6、5.5、5.1、5.0
Connector/Net	1.0 (サポートされなくなりました)	5.0、4.0
Connector/ODBC	5.1	5.6、5.5、5.1、5.0、4.1.1+
Connector/ODBC	3.51 (Unicode はサポートされていません)	5.6、5.5、5.1、5.0、4.1

23.1 MySQL Connector/ODBC

MySQL Connector/ODBC のマニュアルは、MySQL リファレンスマニュアルの一部としてではなく、スタンドアロン形式で発行されるようになりました。詳細については、これらのドキュメントを参照してください。

- 主要マニュアル: MySQL Connector/ODBC Developer Guide
- リリースノート: MySQL Connector/ODBC リリースノート

23.2 MySQL Connector/Net

MySQL Connector/Net のマニュアルは、MySQL リファレンスマニュアルの一部としてではなく、スタンドアロン形式で発行されるようになりました。詳細については、これらのドキュメントを参照してください。

- 主要マニュアル: MySQL Connector/NET Developer Guide
- リリースノート: MySQL Connector/Net リリースノート

23.3 MySQL Connector/J

MySQL Connector/J のマニュアルは、MySQL リファレンスマニュアルの一部としてではなく、スタンドアロン形式で発行されるようになりました。詳細については、これらのドキュメントを参照してください。

- 主要マニュアル: MySQL Connector/J 5.1 Developer Guide
- リリースノート: MySQL Connector/J リリースノート

23.4 MySQL Connector/C++

MySQL Connector/C++ のマニュアルは、MySQL リファレンスマニュアルの一部としてではなく、スタンドアロン形式で発行されるようになりました。詳細については、これらのドキュメントを参照してください。

- 主要マニュアル: MySQL Connector/C++ 8.0 Developer Guide
- リリースノート: MySQL Connector/C++ リリースノート

23.5 MySQL Connector/Python

MySQL Connector/Python のマニュアルは、MySQL リファレンスマニュアルの一部としてではなく、スタンドアロン形式で発行されるようになりました。詳細については、これらのドキュメントを参照してください。

- 主要マニュアル: MySQL Connector/Python Developer Guide
- リリースノート: MySQL Connector/Python リリースノート

23.6 組み込み MySQL サーバーライブラリ libmysqld

組み込み MySQL サーバーライブラリは、クライアントアプリケーション内で、完全な機能を備えた MySQL サーバーを実行できるようにします。この主なメリットは組み込みアプリケーションの速度の向上と管理の単純化です。

組み込みサーバーライブラリは、C/C++ で書かれている MySQL のクライアント/サーバーバージョンに基づいています。そのため、組み込みサーバーも C/C++ で書かれています。ほかの言語で利用可能な組み込みサーバーはありません。

API は組み込み MySQL バージョンとクライアント/サーバーバージョンで同じです。組み込みライブラリを使用するようにスレッドアプリケーションを変更するには、通常次の関数への呼び出しを追加する必要があるだけです。

表 23.3 MySQL 組み込みサーバーライブラリ関数

関数	呼び出すタイミング
mysql_library_init()	ほかの MySQL 関数が呼び出される前、できれば main() 関数の早期に呼び出します。
mysql_library_end()	プログラムが終了する前に呼び出します。
mysql_thread_init()	MySQL にアクセスする、作成する各スレッド内で呼び出します。
mysql_thread_end()	pthread_exit() を呼び出す前に、呼び出します。

次に、コードを libmysqlclient.a の代わりに libmysqld.a とリンクします。アプリケーションとサーバーライブラリ間のバイナリ互換性を確保するには、常に、サーバーライブラリのコンパイルに使用された同じ一連の MySQL のヘッダーに対してアプリケーションをコンパイルします。たとえば、libmysqld が MySQL 5.1 ヘッダーに対してコンパイルされていた場合、アプリケーションを MySQL 5.5 ヘッダーに対してコンパイルしないでください。また逆も同様です。

mysql_library_xxx() 関数は libmysqlclient.a にも含まれているため、アプリケーションを正しいライブラリとリンクするだけで、組み込みバージョンとクライアント/サーバーバージョン間の変更が可能です。セクション 23.7.7.40「mysql_library_init()」を参照してください。

組み込みサーバーとスタンドアロンサーバーの 1 つの違いは、組み込みサーバーの場合、接続のための認証がデフォルトで無効にされていることです。

23.6.1 libmysqld によるプログラムのコンパイル

組み込みサーバーライブラリ libmysqld を含むプリコンパイル済みのバイナリ MySQL 配布では、MySQL は、適切なベンダーコンパイラがあればそれを使用して、ライブラリを構築します。

自分でソースから MySQL を構築する場合に、libmysqld ライブラリを取得するには、-DWITH_EMBEDDED_SERVER=1 オプションを使用して、MySQL を構成してください。セクション 2.9.4「MySQL ソース構成オプション」を参照してください。

プログラムを libmysqld とリンクさせる場合、システム固有の pthread ライブラリおよび MySQL サーバーが使用するいくつかのライブラリも含める必要があります。mysql_config --libmysqld-libs を実行して、ライブラリの完全なリストを取得できます。

コードでスレッド関数を直接呼び出さない場合でも、スレッドプログラムをコンパイルし、リンクするための正しいフラグを使用する必要があります。

C プログラムをコンパイルして、MySQL サーバーライブラリをプログラムの実行可能バージョンに組み込むために必要なファイルを含めるには、コンパイラは各種ファイルを見つける場所を知る必要があり、プログラムのコンパイル方法についての指示を必要とします。次の例に、GNU C コンパイラ gcc を使用するものとして、コマンド行からプログラムをコンパイルする方法を示します。

```
gcc mysql_test.c -o mysql_test \
`/usr/local/mysql/bin/mysql_config --include --libmysqld-libs`
```

gcc コマンドの直後は、C プログラムソースファイルの名前です。その後に、-o オプションを指定して、後続のファイル名が、コンパイラが出力ファイルのコンパイル済みプログラムに指定することになる名前であることを示します。コードの次の行は、コンパイラにインクルードファイルおよびライブラリ、それがコンパイルされるシステムのその他の設定の場所を取得するように伝えます。mysql_config コマンドは、単一引用符ではなく、逆引用符で囲みます。

一部の gcc 以外のプラットフォームでは、組み込みライブラリは C++ 実行時ライブラリに依存するため、組み込みライブラリに対してリンクすると、シンボルなしエラーが発生することがあります。これを解決するには、C++ コンパイラを使用してリンクするか、リンクコマンド行で、必要なライブラリを明示的に挙げます。

23.6.2 組み込み MySQL サーバーを使用する場合の制限

組み込みサーバーには次の制限があります。

- ユーザー定義関数 (UDF) なし。
- コアダンプへのスタックトレースなし。
- これをマスターまたはスレーブ (レプリケーションなし) としてセットアップできません。
- 著しく大きい結果セットはメモリーの少ないシステムで使用できないことがあります。
- ソケットまたは TCP/IP を使用して、外部プロセスから組み込みサーバーに接続することはできません。ただし、ユーザーは中間アプリケーションに接続することができ、それが次にリモートクライアントや外部プロセスに代わって、組み込みサーバーに接続することができます。
- InnoDB は組み込みサーバーに再入可能でなく、連続でも同時でも複数の接続に使用することはできません。
- イベントスケジューラは使用できません。このため、event_scheduler システム変数が無効にされます。

これらの制限の一部は、mysql_embed.h インクルードファイルを編集し、MySQL を再コンパイルすることによって変更できます。

23.6.3 組み込みサーバーとオプション

mysqld サーバーデーモンで指定できるすべてのオプションは、組み込みサーバーライブラリと一緒に使用できます。サーバーオプションは、サーバーを初期化する mysql_library_init() への引数として、配列で指定できます。それらは、my.cnf のようなオプションファイルに指定することもできます。C プログラムにオプションファイルを指定するには、--defaults-file オプションを mysql_library_init() 関数の 2 番目の引数の要素の 1 つとして使用します。mysql_library_init() 関数に関する詳細については、セクション23.7.7.40「mysql_library_init()」を参照してください。

オプションファイルを使用すると、クライアント/サーバーアプリケーションと MySQL が組み込まれているものとの間の切り替えを容易にすることができます。共通オプションを [server] グループの下に置きます。これらは両方の MySQL バージョンによって読み取られます。クライアント/サーバー固有のオプションは [mysqld] セクションの下に置いてください。組み込み MySQL サーバーライブラリに固有のオプションは [embedded] セクションに置きます。アプリケーションに固有なオプションは、[ApplicationName_SERVER] とラベル付けされたセクションの下に置きます。セクション4.2.6「オプションファイルの使用」を参照してください。

23.6.4 組み込みサーバーの例

これらの 2 つのプログラム例は、Linux または FreeBSD システムを変更せずに、機能するはずです。その他のオペレーティングシステムでは、たいていファイルパスによる小さな変更が必要です。これらの例は、実際のアプリケーションの必要部分である余計なものなしに、問題を理解するために十分な詳細を与えるように設計され

ています。最初の例は、きわめて単純です。2 番目の例は、いくつかのエラーチェックを伴うやや高度なものです。最初のもののあとに、プログラムをコンパイルするためのコマンド行エントリが続いています。2 つ目のあとには、代わりにコンパイルに使用できる GNU メイクファイルが続いています。

例 1

test1_libmysqld.c

```
#include <stdio.h>
#include <stdlib.h>
#include <stdarg.h>
#include "mysql.h"

MYSQL *mysql;
MYSQL_RES *results;
MYSQL_ROW record;

static char *server_options[] = \
      { "mysql_test", "--defaults-file=my.cnf", NULL };
int num_elements = (sizeof(server_options) / sizeof(char *)) - 1;

static char *server_groups[] = { "libmysqld_server",
                                 "libmysqld_client", NULL };

int main(void)
{
  mysql_library_init(num_elements, server_options, server_groups);
  mysql = mysql_init(NULL);
  mysql_options(mysql, MYSQL_READ_DEFAULT_GROUP, "libmysqld_client");
  mysql_options(mysql, MYSQL_OPT_USE_EMBEDDED_CONNECTION, NULL);

  mysql_real_connect(mysql, NULL,NULL,NULL, "database1", 0,NULL,0);

  mysql_query(mysql, "SELECT column1, column2 FROM table1");

  results = mysql_store_result(mysql);

  while((record = mysql_fetch_row(results))) {
    printf("%s - %s \n", record[0], record[1]);
  }

  mysql_free_result(results);
  mysql_close(mysql);
  mysql_library_end();

  return 0;
}
```

これは、上のプログラムをコンパイルするためのコマンド行です。

```
gcc test1_libmysqld.c -o test1_libmysqld \
 `/usr/local/mysql/bin/mysql_config --include --libmysqld-libs`
```

例 2

例を試してみるには、MySQL ソースディレクトリと同じレベルで test2_libmysqld ディレクトリを作成します。test2_libmysqld.c ソースと GNUmakefile をこのディレクトリに保存し、test2_libmysqld ディレクトリ内から GNU make を実行します。

test2_libmysqld.c

```
/*
 * A simple example client, using the embedded MySQL server library
*/

#include <mysql.h>
#include <stdarg.h>
#include <stdio.h>
#include <stdlib.h>

MYSQL *db_connect(const char *dbname);
void db_disconnect(MYSQL *db);
void db_do_query(MYSQL *db, const char *query);

const char *server_groups[] = {
  "test2_libmysqld_SERVER", "embedded", "server", NULL
};

int
```

```
main(int argc, char **argv)
{
  MYSQL *one, *two;

  /* mysql_library_init() must be called before any other mysql
   * functions.
   *
   * You can use mysql_library_init(0, NULL, NULL), and it
   * initializes the server using groups = {
   *   "server", "embedded", NULL
   *  }.
   *
   * In your $HOME/.my.cnf file, you probably want to put:

[test2_libmysqld_SERVER]
language = /path/to/source/of/mysql/sql/share/english

   * You could, of course, modify argc and argv before passing
   * them to this function.  Or you could create new ones in any
   * way you like.  But all of the arguments in argv (except for
   * argv[0], which is the program name) should be valid options
   * for the MySQL server.
   *
   * If you link this client against the normal mysqlclient
   * library, this function is just a stub that does nothing.
   */
  mysql_library_init(argc, argv, (char **)server_groups);

  one = db_connect("test");
  two = db_connect(NULL);

  db_do_query(one, "SHOW TABLE STATUS");
  db_do_query(two, "SHOW DATABASES");

  mysql_close(two);
  mysql_close(one);

  /* This must be called after all other mysql functions */
  mysql_library_end();

  exit(EXIT_SUCCESS);
}

static void
die(MYSQL *db, char *fmt, ...)
{
  va_list ap;
  va_start(ap, fmt);
  vfprintf(stderr, fmt, ap);
  va_end(ap);
  (void)putc('\n', stderr);
  if (db)
    db_disconnect(db);
  exit(EXIT_FAILURE);
}

MYSQL *
db_connect(const char *dbname)
{
  MYSQL *db = mysql_init(NULL);
  if (!db)
    die(db, "mysql_init failed: no memory");
  /*
   * Notice that the client and server use separate group names.
   * This is critical, because the server does not accept the
   * client's options, and vice versa.
   */
  mysql_options(db, MYSQL_READ_DEFAULT_GROUP, "test2_libmysqld_CLIENT");
  if (!mysql_real_connect(db, NULL, NULL, NULL, dbname, 0, NULL, 0))
    die(db, "mysql_real_connect failed: %s", mysql_error(db));

  return db;
}

void
db_disconnect(MYSQL *db)
{
  mysql_close(db);
}
```

```
void
db_do_query(MYSQL *db, const char *query)
{
  if (mysql_query(db, query) != 0)
    goto err;

  if (mysql_field_count(db) > 0)
  {
    MYSQL_RES   *res;
    MYSQL_ROW    row, end_row;
    int num_fields;

    if (!(res = mysql_store_result(db)))
      goto err;
    num_fields = mysql_num_fields(res);
    while ((row = mysql_fetch_row(res)))
    {
      (void)fputs(">> ", stdout);
      for (end_row = row + num_fields; row < end_row; ++row)
        (void)printf("%s\t", row ? (char*)*row : "NULL");
      (void)fputc('\n', stdout);
    }
    (void)fputc('\n', stdout);
    mysql_free_result(res);
  }
  else
    (void)printf("Affected rows: %lld\n", mysql_affected_rows(db));

  return;

err:
  die(db, "db_do_query failed: %s [%s]", mysql_error(db), query);
}
```

GNUmakefile

```
# This assumes the MySQL software is installed in /usr/local/mysql
inc      := /usr/local/mysql/include/mysql
lib      := /usr/local/mysql/lib

# If you have not installed the MySQL software yet, try this instead
#inc      := $(HOME)/mysql-5.6/include
#lib      := $(HOME)/mysql-5.6/libmysqld

CC       := gcc
CPPFLAGS := -I$(inc) -D_THREAD_SAFE -D_REENTRANT
CFLAGS   := -g -W -Wall
LDFLAGS  := -static
# You can change -lmysqld to -lmysqlclient to use the
# client/server library
LDLIBS    = -L$(lib) -lmysqld -lm -ldl -lcrypt

ifneq (,$(shell grep FreeBSD /COPYRIGHT 2>/dev/null))
# FreeBSD
LDFLAGS += -pthread
else
# Assume Linux
LDLIBS += -lpthread
endif

# This works for simple one-file test programs
sources := $(wildcard *.c)
objects := $(patsubst %c,%o,$(sources))
targets := $(basename $(sources))

all: $(targets)

clean:
	rm -f $(targets) $(objects) *.core
```

23.7 MySQL C API

C API は、MySQL クライアント/サーバープロトコルへの低レベルアクセスを提供し、C プログラムがデータベースの内容にアクセスできるようにします。C API コードは MySQL とともに配布され、libmysqlclient ライブラリに実装されています。セクション23.7.1「MySQL C API の実装」を参照してください。

ほかのほとんどのクライアント API は libmysqlclient ライブラリを使用して、MySQL サーバーと通信します。(例外は Connector/J および Connector/Net です。)これは、たとえば、ほかのクライアントプログラムによって使

用されている多くの同じ環境変数を利用できることを意味します。それらは、ライブラリから参照されているためです。これらの環境変数のリストについては、セクション4.1「MySQL プログラムの概要」を参照してください。

C API を使用してクライアントプログラムを構築する手順については、セクション23.7.4.1「C API クライアントプログラムの構築」を参照してください。スレッドによるプログラミングについては、セクション23.7.4.2「C API スレッドクライアントプログラムの作成」を参照してください。「サーバー」と「クライアント」を同じプログラムに含む (および外部 MySQL サーバーと通信しない) スタンドアロンアプリケーションを作成するには、セクション23.6「組み込み MySQL サーバーライブラリ libmysqld」を参照してください。

> 注記
>
> アップグレード後に、コンパイル済みのクライアントプログラムに、「コマンドは同期されていません」または予期しないコアダンプなどの問題が発生した場合は、プログラムが古いヘッダーファイルまたはライブラリファイルを使用してコンパイルされた可能性があります。この場合、コンパイルに使用された mysql.h ファイルおよび libmysqlclient.a ライブラリの日付をチェックして、それらが新しい MySQL 配布からのものであることを確認します。そうでない場合には、プログラムを新しいヘッダーおよびライブラリで再コンパイルします。ライブラリのメジャーバージョン番号が変更された (libmysqlclient.so.17 から libmysqlclient.so.18 など) 場合に、共有クライアントライブラリに対してコンパイルされているプログラムでも、再コンパイルが必要になる可能性があります。追加の互換性情報については、セクション23.7.4.3「C API クライアントプログラムの実行」を参照してください。

クライアントは最大の通信バッファーサイズを持ちます。最初に割り当てられるバッファーのサイズ (16K バイト) は最大サイズ (デフォルトで 16M バイト) まで自動的に増やされます。バッファーサイズは、必要に応じてのみ増やされるため、最大制限を単純に増やしても、それ自体で使用されるリソースが増えるわけではありません。このサイズチェックは、主に誤ったステートメントと通信パケットに対する予防措置です。

通信バッファーは、単一の SQL ステートメント (クライアントからサーバーへのトラフィックの) と 1 行の返されるデータ (サーバーからクライアントへのトラフィックの) を格納するために十分な大きさがある必要があります。各セッションの通信バッファーは、クエリーまたは行を処理するために、最大制限まで動的に拡大されます。たとえば、16M バイトまでのデータを格納する BLOB 値がある場合、少なくとも 16M バイトの通信バッファー制限が必要です (サーバーとクライアントの両方で)。クライアントライブラリに組み込まれたデフォルトの最大は 1G バイトですが、サーバーでのデフォルトの最大は 1M バイトです。これを増やすには、サーバー起動時に max_allowed_packet パラメータの値を変更します。セクション8.11.2「サーバーパラメータのチューニング」を参照してください。

MySQL サーバーは各クエリー後、各通信バッファーを net_buffer_length バイトに縮小します。クライアントでは、接続に関連付けられたバッファーのサイズは接続がクローズされるまで減らされません。その時点で、クライアントのメモリーが再利用されます。

23.7.1 MySQL C API の実装

MySQL C API は、C で書かれたクライアントアプリケーションが MySQL サーバーとの通信に使用できる C ベースの API です。クライアントプログラムはコンパイル時に C API ヘッダーを参照し、リンク時に C API ライブラリファイルにリンクします。ライブラリは、アプリケーションがサーバーとどのように通信することを意図しているかに応じて、2 つのバージョンで提供されています。

- libmysqlclient: スタンドアロンサーバープロセスのクライアントとして、ネットワーク接続経由で通信するアプリケーションで使用される、ライブラリのクライアントバージョン。
- libmysqld: アプリケーション自体の中に組み込み MySQL サーバーを含めることを目的としたアプリケーションで使用される、ライブラリの組み込みサーバーバージョン。アプリケーションはその独自のプライベートサーバーインスタンスと通信します。

どちらのライブラリも同じインタフェースを持ちます。C API 呼び出しに関して、アプリケーションは、組み込みサーバーと通信する同じ方法で、スタンドアロンサーバーと通信します。特定のクライアントを、構築時に libmysqlclient に対してリンクするか、libmysqld に対してリンクするかに応じて、スタンドアロンまたは組み込みサーバーと通信するように構築できます。

C API クライアントプログラムの構築に必要な C API ヘッダーおよびライブラリファイルを取得するには、2 つの方法があります。

- MySQL サーバー配布をインストールします。サーバー配布には libmysqlclient と libmysqld の両方が含まれています。

- MySQL Connector/C 配布をインストールします。Connector/C 配布には libmysqlclient のみ含まれます。それらには libmysqld は含まれません。

MySQL サーバーと MySQL Connector/C のどちらでも、事前構築された C API ファイルを含むバイナリ配布をインストールするか、ソース配布を使用して、自分で C API ファイルを構築できます。

通常、MySQL サーバー配布または MySQL Connector/C 配布のいずれかをインストールし、両方はインストールしません。MySQL サーバーと MySQL Connector/C の同時インストールに伴う問題については、セクション 23.7.2「MySQL サーバーと MySQL Connector/C の同時インストール」を参照してください。

C API クライアントアプリケーションのリンク時に使用するライブラリファイルの名前は、ライブラリの種類と配布が構築されるプラットフォームによって異なります。

- Unix (および Unix に類似する) システムで、静的ライブラリは libmysqlclient.a です。動的ライブラリはほとんどの Unix システムで libmysqlclient.so および OS X で libmysqlclient.dylib です。

  組み込みサーバーライブラリを含む配布の場合、対応するライブラリ名は libmysqlclient ではなく、libmysqld から始まります。

- Windows では、静的ライブラリは mysqlclient.lib で、動的ライブラリは libmysql.dll です。Windows 配布には、動的ライブラリを使用するために必要な静的インポートライブラリの libmysql.lib も含まれます。

  組み込みサーバーライブラリを含む配布の場合、対応するライブラリ名は mysqlserver.lib、libmysqld.dll、および libmysqld.lib です。

  Windows 配布には一連のデバッグライブラリも含まれます。これらは非デバッグライブラリと同じ名前を持ちますが、lib/debug ライブラリ内に置かれます。デバッグ C ランタイムを使用して構築されたクライアントのコンパイル時には、デバッグライブラリを使用する必要があります。

Unix では、名前に _r を含むライブラリも見られることがあります。MySQL 5.5 より前では、これらは、非 _r ライブラリとは別にスレッドセーフ (再入可能) ライブラリとして構築されました。5.5 以降、両方のライブラリが同じになり、_r 名は、対応する非 _r 名へのシンボリックリンクになります。_r ライブラリを使用する必要はありません。たとえば、mysql_config を使用して、リンカーフラグを取得する場合、スレッドクライアントでも、すべての場合に mysql_config --libs を使用できます。mysql_config --libs_r を使用する必要はありません。

23.7.2 MySQL サーバーと MySQL Connector/C の同時インストール

MySQL サーバーと MySQL Connector/C のインストールパッケージは、どちらも MySQL C API クライアントプログラムを構築して実行するために必要なファイルを提供します。このセクションでは、両方の製品を同じシステムにインストールできる場合について説明します。一部のパッケージング形式では、これは競合せずに可能です。ほかの場合は、両方の製品を同時にインストールすることはできません。

この説明では、両方の製品に対して類似したパッケージの種類 (たとえば両方の製品に対して RPM パッケージ) を使用することを前提としています。パッケージングの種類間の共存 (一方の製品に対して RPM パッケージを使用し、他方に対して tar ファイルパッケージを使用するなど) については説明しません。Oracle によって提供されたパッケージとサードパーティーベンダーによって提供されたパッケージの共存についても説明しません。

両方の製品をインストールする場合、ヘッダーファイルとライブラリのいずれかのセットを選択するように、開発ツールや実行時環境を調整する必要があることがあります。セクション23.7.4.1「C API クライアントプログラムの構築」およびセクション23.7.4.3「C API クライアントプログラムの実行」を参照してください。

tar および Zip ファイルパッケージは、それらのアンパック先のディレクトリの下にインストールされます。たとえば、MySQL サーバーと MySQL Connector/C の tar パッケージを /usr/local の下にアンパックでき、それらは競合せずに、個別のディレクトリ名にアンパックされます。

Windows MSI インストーラはそれらの独自のインストールディレクトリを使用するため、MySQL サーバーと MySQL Connector/C のインストーラは競合しません。

OS X DMG パッケージは、同じ親ディレクトリの下でも、異なるサブディレクトリ内にインストールされるため、競合はありません。例:

```
/usr/local/mysql-5.6.11-osx10.7-x86_64/
/usr/local/mysql-connector-c-6.1.0-osx10.7-x86/
```

Solaris PKG パッケージは、同じ親ディレクトリの下でも、異なるサブディレクトリ内にインストールされるため、競合はありません。例:

```
/opt/mysql/mysql
/opt/mysql/connector-c
```

Solaris MySQL Connector/C インストーラは、/usr/bin や /usr/lib などのシステムディレクトリから、インストールディレクトリへのシンボリックリンクを作成しません。それは、インストール後に必要に応じて、手動で行う必要があります。

RPM インストールでは、複数の種類の RPM パッケージがあります。MySQL サーバーの shared および devel RPM パッケージは対応する MySQL Connector/C RPM パッケージに似ています。MySQL サーバー RPM パッケージと MySQL Connector/C RPM パッケージは、クライアントライブラリ関連ファイルに同じインストールの場所を使用するため、これらの RPM パッケージの種類は共存できません。これは次の条件が当てはまることを意味します。

- MySQL サーバー shared および devel RPM パッケージがインストールされている場合、それらは C API ヘッダーおよびライブラリを提供し、MySQL Connector/C RPM パッケージをインストールする必要はありません。とにかく MySQL Connector/C パッケージをインストールする場合は、まず対応する MySQL サーバーパッケージを削除する必要があります。
- MySQL Connector/C RPM パッケージをすでにインストールしている場合に MySQL サーバー RPM パッケージをインストールするには、まず MySQL Connector/C RPM パッケージを削除する必要があります。

shared および devel 以外の MySQL サーバー RPM パッケージは MySQL Connector/C パッケージと競合しないため、MySQL Connector/C がインストールされている場合にもインストールできます。これには mysqld サーバー自体を含むメインサーバー RPM が含まれます。

23.7.3 C API クライアントプログラムの例

mysql、mysqladmin、および mysqlshow など、MySQL ソース配布内のクライアントの多くは C で書かれています。C API の使用方法を示す例を探している場合は、これらのクライアントを調べてください。ソース配布を取得して、その client ディレクトリを調べます。セクション2.1.3「MySQL の取得方法」を参照してください。

23.7.4 C API クライアントプログラムの構築と実行

次のセクションでは、C API を使用するクライアントプログラムの構築に関する情報を提供します。トピックには、クライアントのコンパイルとリンク、スレッドクライアントの作成、および実行時の問題のトラブルシューティングが含まれます。

23.7.4.1 C API クライアントプログラムの構築

このセクションでは、MySQL C API を使用する C プログラムのコンパイルのガイドラインを提供します。

Unix での MySQL クライアントのコンパイル

MySQL ヘッダーファイルを使用するクライアントプログラムのコンパイル時に、コンパイラがそれらを見つけることができるように、-I オプションを指定する必要がある場合があります。たとえば、ヘッダーファイルが /usr/local/mysql/include にインストールされている場合、コンパイルコマンドでこのオプションを使用します。

```
-I/usr/local/mysql/include
```

リンクコマンドで、-lmysqlclient オプションを使用して、MySQL クライアントをリンクする必要があります。リンカーにライブラリを見つける場所を伝えるために、-Lオプションを指定する必要がある場合もあります。たとえば、ライブラリが /usr/local/mysql/lib にインストールされている場合、リンクコマンドでこれらのオプションを使用します。

```
-L/usr/local/mysql/lib -lmysqlclient
```

パス名は、使用しているシステムで異なることがあります。必要に応じて、-I オプションと -L オプションを調整してください。

Unix で MySQL プログラムのコンパイルを簡単にするには、mysql_config スクリプトを使用します。セクション4.7.2「mysql_config — クライアントのコンパイル用オプションの表示」を参照してください。

mysql_config は、コンパイルやリンクに必要なオプションを表示します。

```
shell> mysql_config --cflags
```

```
shell> mysql_config --libs
```

それらのコマンドを実行して、正しいオプションを取得し、それらを手動でコンパイルまたはリンクコマンドに追加できます。または、mysql_config からの出力を逆引用符を使用して、コマンド行に直接含めます。

```
shell> gcc -c `mysql_config --cflags` progname.c
shell> gcc -o progname progname.o `mysql_config --libs`
```

Microsoft Windows での MySQL クライアントのコンパイル

ヘッダーおよびライブラリファイルの場所を指定するには、開発環境によって提供されている機能を使用します。

Windows で C API クライアントを構築するには、C クライアントライブラリのほか、Windows ws2_32 ソケットライブラリおよび Secur32 セキュリティライブラリにリンクする必要があります。

Windows では、コードを動的または静的 C クライアントライブラリとリンクできます。静的ライブラリは mysqlclient.lib という名前で、動的ライブラリは libmysql.dll という名前です。さらに、動的ライブラリを使用するために libmysql.lib 静的インポートライブラリが必要です。

静的ライブラリとリンクする場合、これらの状況が満たされていないと、障害が発生する可能性があります。

- クライアントアプリケーションは、ライブラリをコンパイルするために使用される Visual Studio の同じバージョンでコンパイルする必要があります。
- クライアントアプリケーションは /MT コンパイラオプションを使用して、C ランタイムを静的にリンクしてください。

クライアントアプリケーションがデバッグモードで構築され、静的デバッグ C ランタイムを使用する (/MTd コンパイラオプション) 場合、mysqlclient.lib 静的ライブラリが同じオプションを使用して構築されている場合に、それにリンクできます。クライアントアプリケーションが動的 C ランタイムを使用する (/MD オプション、またはデバッグモードでの /MDd オプション) 場合、それを libmysql.dll 動的ライブラリにリンクする必要があります。それは静的クライアントライブラリにリンクできません。

リンクオプションについて説明する MSDN ページは、http://msdn.microsoft.com/en-us/library/2kzt1wy3.aspx にあります。

MySQL クライアントライブラリへのリンクの問題のトラブルシューティング

MySQL 5.6 では、MySQL クライアントライブラリに SSL サポートが組み込まれて含まれています。アプリケーションで、リンク時に OpenSSL ライブラリからの -lssl または -lcrypto が必要な場合、-lmysqlclient の前にそれらを指定する必要があります。

リンカーが MySQL クライアントライブラリを見つけられない場合、ここに示すような、mysql_ から始まるシンボルの未定義参照エラーを受け取ることがあります。

```
/tmp/ccFKsdPa.o: In function `main':
/tmp/ccFKsdPa.o(.text+0xb): undefined reference to `mysql_init'
/tmp/ccFKsdPa.o(.text+0x31): undefined reference to `mysql_real_connect'
/tmp/ccFKsdPa.o(.text+0x69): undefined reference to `mysql_error'
/tmp/ccFKsdPa.o(.text+0x9a): undefined reference to `mysql_close'
```

この問題は、リンクコマンドの末尾に -Ldir_path -lmysqlclient を追加して解決できます。ここで dir_path はクライアントライブラリが存在するディレクトリのパス名を表します。正しいディレクトリを判断するには、このコマンドを試してみます。

```
shell> mysql_config --libs
```

mysql_config からの出力に、リンクコマンドに同様に指定すべきほかのライブラリが示されることがあります。コンパイルまたはリンクコマンド内に、逆引用符を使用して、mysql_config 出力を直接含めることができます。例:

```
shell> gcc -o progname progname.o `mysql_config --libs`
```

リンク時に floor シンボルが未定義であるというエラーが発生した場合、コンパイル/リンク行の末尾に -lm を追加して、数学ライブラリにリンクします。同様に、connect() などのシステムに存在しているべきほかの関数の未定義参照エラーを受け取った場合は、問題の関数のマニュアルページをチェックして、リンクコマンドに追加すべきライブラリを判断します。

システムに存在しない関数の次のような未定義参照エラーを受け取った場合、それは通常 MySQL クライアントライブラリが、使用しているシステムと 100% の互換性がないシステムでコンパイルされたことを意味します。

```
mf_format.o(.text+0x201): undefined reference to `__lxstat'
```

この場合は、最新の MySQL または MySQL Connector/C ソース配布をダウンロードし、自分で MySQL クライアントライブラリをコンパイルしてください。セクション2.9「ソースから MySQL をインストールする」およびを参照してください。

23.7.4.2 C API スレッドクライアントプログラムの作成

クライアントライブラリはほとんどスレッドセーフです。最大の問題は、ソケットから読み取られる net.c 内のサブルーチンが割り込みセーフ (interrupt-safe) でないことです。これは、サーバーへの長い読み取りを中断できる独自のアラームを必要とする可能性があるという考えで行われました。SIGPIPE 割り込みの割り込みハンドラをインストールする場合、ソケット処理はスレッドセーフであるべきです。

接続の終了時のプログラムの中止を避けるため、MySQL は、mysql_library_init()、mysql_init()、または mysql_connect() への最初の呼び出しで SIGPIPE をブロックします。独自の SIGPIPE ハンドラを使用するには、まず mysql_library_init() を呼び出してから、ハンドラをインストールします。

libmysqlclient クライアントライブラリに対するリンク時に、「未定義のシンボル」エラーが発生した場合、多くの場合、これはリンク/コンパイルコマンドでスレッドライブラリを含めていなかったためです。

クライアントライブラリは接続ごとにスレッドセーフです。2 つのスレッドに、同じ接続を共有させることができますが、次の警告を伴います。

- 複数のスレッドは同じ接続で同時に MySQL サーバーにクエリーを送信できません。特に、1 つのスレッドでの mysql_query() と mysql_store_result() の呼び出しの間に、ほかのスレッドが同じ接続を使用しないことを確認する必要があります。mysql_query() および mysql_store_result() 呼び出しのペアを相互排他ロックで囲む必要があります。mysql_store_result() が戻ったら、ロックを解除でき、ほかのスレッドが同じ接続をクエリーできます。

  POSIX スレッドを使用する場合、pthread_mutex_lock() および pthread_mutex_unlock() を使用して、相互排他ロックを確立し、解除できます。

- 多くのスレッドは、mysql_store_result() によって取得されるさまざまな結果セットにアクセスできます。

- mysql_use_result() を使用するには、結果セットが閉じられるまで、同じ接続を使用しているほかのスレッドがないことを確認する必要があります。ただし、同じ接続を共有するスレッドクライアントが mysql_store_result() を使用することが実際にもっとも適切です。

MySQL データベースへの接続を作成していないが、MySQL 関数を呼び出しているスレッドがある場合、次のことを知る必要があります。

mysql_init() を呼び出すと、MySQL は、特にデバッグライブラリによって使用されるスレッドのスレッド固有の変数を作成します。スレッドが mysql_init() を呼び出す前に、MySQL 関数を呼び出した場合、そのスレッドには必要なスレッド固有変数が設定されていないため、遅かれ早かれコアダンプが生成される可能性があります。問題を回避するには、次を実行する必要があります。

1. ほかのすべての MySQL 関数の前に mysql_library_init() を呼び出します。それはスレッドセーフでないため、スレッドが作成される前にそれを呼び出すか、相互排他ロックで呼び出しを保護します。

2. mysql_thread_init() が、スレッドハンドラ内の MySQL 関数を呼び出す前の早期に呼び出されるように調整します。mysql_init() を呼び出すと、それが自動的に mysql_thread_init() を呼び出します。

3. スレッド内で、pthread_exit() を呼び出す前に mysql_thread_end() を呼び出します。これにより、MySQL スレッド固有変数によって使われたメモリーを解放します。

mysql_init() に関する先述の注記は、mysql_init() を呼び出す mysql_connect() にも適用します。

23.7.4.3 C API クライアントプログラムの実行

アップグレード後に、コンパイル済みのクライアントプログラムに、「コマンドは同期されていません」または予期しないコアダンプなどの問題が発生した場合は、プログラムが古いヘッダーファイルまたはライブラリファイルを使用してコンパイルされた可能性があります。この場合、コンパイルに使用された mysql.h ファイルおよび libmysqlclient.a ライブラリの日付をチェックして、それらが新しい MySQL 配布からのものであることを確認します。そうでない場合には、プログラムを新しいヘッダーおよびライブラリで再コンパイルします。ライブラ

リのメジャーバージョン番号が変更された (libmysqlclient.so.17 から libmysqlclient.so.18 など) 場合に、共有クライアントライブラリに対してコンパイルされているプログラムでも、再コンパイルが必要になる可能性があります。

メジャークライアントライブラリバージョンによって互換性が決まります。(たとえば、libmysqlclient.so.18.1.0 の場合、メジャーバージョンは 18 です。)このため、新しいバージョンの MySQL に付属するライブラリは、同じメジャー番号を持つ古いバージョンの簡単に取り換え可能な代替です。メジャーライブラリバージョンが同じであるかぎり、ライブラリをアップグレードでき、古いアプリケーションはそれと連携し続けます。

MySQL プログラムを実行しようとすると、実行時に未定義参照エラーが発生することがあります。これらのエラーで mysql_ から始まるシンボルが指定されているか、libmysqlclient ライブラリが見つからないことを示している場合、それはシステムが共有 libmysqlclient.so ライブラリを見つけられないことを意味します。この問題の解決方法は、システムにそのライブラリが存在するディレクトリ内で共有ライブラリを検索するように伝えることです。次の方法のうち使用しているシステムに適切な方法を使用します。

- libmysqlclient.so が存在するディレクトリのパスを LD_LIBRARY_PATH または LD_LIBRARY 環境変数に追加します。
- OS X では、libmysqlclient.dylib が存在するディレクトリのパスを DYLD_LIBRARY_PATH 環境変数に追加します。
- 共有ライブラリファイル (libmysqlclient.so など) を、/lib などシステムによって検索されるいくつかのディレクトリにコピーし、ldconfig を実行して、共有ライブラリ情報を更新します。すべての関連ファイルをコピーしてください。共有ライブラリは、代替名を提供するシンボリックリンクを使用して、複数の名前で存在することがあります。

アプリケーションが組み込みサーバーライブラリにリンクされている場合、実行時エラーメッセージに、libmysqlclient ライブラリではなく、libmysqld が示されますが、この問題の解決方法は説明したばかりのものと同じです。

23.7.4.4 C API サーバーおよびクライアントライブラリのバージョン

MySQL サーバーのバージョンの文字列と数値形式は、コンパイル時に MYSQL_SERVER_VERSION および MYSQL_VERSION_ID マクロの値として、実行時に mysql_get_server_info() および mysql_get_server_version() 関数の値として入手できます。

クライアントライブラリのバージョンは MySQL のバージョンです。Connector/C では、これは Connector/C 配布がベースにしている MySQL のバージョンです。このバージョンの文字列と数値形式は、コンパイル時に MYSQL_SERVER_VERSION および MYSQL_VERSION_ID マクロの値として、実行時に mysql_get_client_info() および mysql_get_client_version() 関数の値として入手できます。

23.7.5 C API データ構造

このセクションでは、プリペアドステートメントに使用されるもの以外の C API データ構造について説明します。後者に関する情報については、セクション23.7.9「C API プリペアドステートメントデータ構造」を参照してください。

- MYSQL

  この構造は 1 つのデータベース接続へのハンドルを表します。それはほとんどすべての MySQL 関数に使われます。MYSQL 構造のコピーを作成しようとしないでください。そのようなコピーが使用可能である保証はありません。

- MYSQL_RES

  この構造は行 (SELECT、SHOW、DESCRIBE、EXPLAIN) を返すクエリーの結果を表します。クエリーから返される情報は、このセクションの残りの部分で、結果セットと呼ばれています。

- MYSQL_ROW

  これは 1 行のデータのタイプセーフな表現です。それは現在カウントされるバイト文字列の配列として実装されています。(フィールド値にバイナリデータが含まれている可能性がある場合、内部でそれらの値に NULL バイトが格納される可能性があるため、これらを NULL 終端文字列として扱うことはできません。)行は mysql_fetch_row() を呼び出すことによって取得します。

- MYSQL_FIELD

この構造には、メタデータ (フィールドの名前、型、サイズなどのフィールドに関する情報) が格納されます。そのメンバーについては、このセクションのあとの方でさらに詳しく説明しています。mysql_fetch_field() を繰り返し呼び出すことによって、フィールドごとに、MYSQL_FIELD 構造を取得できます。フィールド値はこの構造に含まれません。それらは MYSQL_ROW 構造に含まれます。

- MYSQL_FIELD_OFFSET

  これは MySQL フィールドリスト内へのオフセットのタイプセーフな表現です。(mysql_field_seek() で使用されます。)オフセットは行内のゼロから始まるフィールド番号です。

- my_ulonglong

  行数と mysql_affected_rows()、mysql_num_rows()、および mysql_insert_id() に使用される型。この型は、0 から 1.84e19 までの範囲を提供します。

  一部のシステムでは、型 my_ulonglong の値を出力しようとしても機能しません。このような値を出力するには、それを unsigned long に変換して、%lu 出力形式を使用します。例:

```
printf ("Number of rows: %lu\n",
        (unsigned long) mysql_num_rows(result));
```

- my_bool

  true (ゼロ以外) または false (ゼロ) の値のブール型。

MYSQL_FIELD 構造には、次のリストに説明するメンバーが格納されます。この定義は、SELECT ステートメントによって生成されるものなど、主に結果セットのカラムに適用します。MySQL 5.6 で、MYSQL_FIELD 構造は、プリペアド CALL ステートメントを使用して実行されたストアドプロシージャーから返される OUT および INOUT パラメータのメタデータを提供するためにも使用されます。そのようなパラメータでは、構造メンバーの一部はカラム値の意味と異なる意味を持ちます。

- char * name

  NULL 終端文字列としてのフィールドの名前。フィールドに AS 句によって エイリアスが与えられている場合、name の値は エイリアスになります。プロシージャーパラメータの場合は、パラメータ名。

- char * org_name

  NULL 終端文字列としてのフィールドの名前。エイリアスは無視されます。式の場合、値は空の文字列です。プロシージャーパラメータの場合は、パラメータ名。

- char * table

  このフィールドが計算されるフィールドでない場合、それを格納するテーブルの名前。計算されるフィールドの場合、table 値は空の文字列です。カラムがビューから選択される場合、table はそのビューを指定します。テーブルまたはビューに AS 句によって エイリアスが与えられている場合、table の値は エイリアスになります。UNION の場合、値は空の文字列です。プロシージャーパラメータの場合はプロシージャー名。

- char * org_table

  NULL 終端文字列としてのテーブルの名前。エイリアスは無視されます。カラムがビューから選択される場合、org_table はそのビューを指定します。UNION の場合、値は空の文字列です。プロシージャーパラメータの場合はプロシージャー名。

- char * db

  NULL 終端文字列としてのフィールドの取得元のデータベースの名前。フィールドが計算されるフィールドである場合、db は空の文字列になります。UNION の場合、値は空の文字列です。プロシージャーパラメータの場合は、プロシージャーを格納するデータベースの名前。

- char * catalog

  カタログ名。この値は常に "def" です。

- char * def

  NULL 終端文字列としてのこのフィールドのデフォルト値。これは mysql_list_fields() を使用する場合にのみ設定します。

- unsigned long length

  フィールドの幅。これはバイト単位での表示の長さに対応します。

  サーバーは結果セットを生成する前に、length 値を判断するため、これは、結果セットに対するクエリーによって生成される実際の値を前もって知ることなく、結果カラムの可能性のある最大値を保持できる、データ型に必要な最小の長さになります。

- unsigned long max_length

  結果セットのフィールドの最大幅 (結果セット内の実際の行の最大長フィールド値のバイト単位での長さ)。mysql_store_result() または mysql_list_fields() を使用した場合、これにはフィールドの最大長が含まれます。mysql_use_result() を使用した場合、この変数の値はゼロになります。

  max_length の値は、結果セット内の値の文字列表現の長さになります。たとえば、FLOAT カラムを取得し、「最大幅」の値が -12.345 である場合、max_length は 7 ('-12.345' の長さ) となります。

  プリペアドステートメントを使用している場合、バイナリプロトコルでは、値の長さが結果セット内の値の型によって異なるため、max_length はデフォルトで設定されません。(セクション23.7.9「C API プリペアドステートメントデータ構造」を参照してください。)とにかく max_length 値が必要な場合、mysql_stmt_attr_set() で STMT_ATTR_UPDATE_MAX_LENGTH オプションを有効にすると、mysql_stmt_store_result() を呼び出したときにその長さが設定されます。(セクション23.7.11.3「mysql_stmt_attr_set()」およびセクション23.7.11.28「mysql_stmt_store_result()」を参照してください。)

- unsigned int name_length

  name の長さ。

- unsigned int org_name_length

  org_name の長さ。

- unsigned int table_length

  table の長さ。

- unsigned int org_table_length

  org_table の長さ。

- unsigned int db_length

  db の長さ。

- unsigned int catalog_length

  catalog の長さ。

- unsigned int def_length

  def の長さ。

- unsigned int flags

  フィールドを説明するビットフラグ。flags 値は、次の表に示すゼロ以上のビットセットを持つ可能性があります。

フラグ値	フラグの説明
NOT_NULL_FLAG	フィールドは NULL にできません
PRI_KEY_FLAG	フィールドは主キーの一部です
UNIQUE_KEY_FLAG	フィールドは一意のキーの一部です
MULTIPLE_KEY_FLAG	フィールドは一意でないキーの一部です
UNSIGNED_FLAG	フィールドは UNSIGNED 属性を持ちます
ZEROFILL_FLAG	フィールドは ZEROFILL 属性を持ちます
BINARY_FLAG	フィールドは BINARY 属性を持ちます
AUTO_INCREMENT_FLAG	フィールドは AUTO_INCREMENT 属性を持ちます

フラグ値	フラグの説明
ENUM_FLAG	フィールドは ENUM です
SET_FLAG	フィールドは SET です
BLOB_FLAG	フィールドは BLOB または TEXT (非推奨) です
TIMESTAMP_FLAG	フィールドは TIMESTAMP (非推奨) です
NUM_FLAG	フィールドは数値です。表のあとの補注を参照してください
NO_DEFAULT_VALUE_FLAG	フィールドにはデフォルト値がありません。表のあとの補注を参照してください

これらのフラグの一部はデータ型情報を示し、後述する field->type メンバーの MYSQL_TYPE_xxx 値によって置き換えられるか、一緒に使用されます。

- BLOB または TIMESTAMP 値をチェックするには、type が MYSQL_TYPE_BLOB であるか、または MYSQL_TYPE_TIMESTAMP であるかをチェックします。(BLOB_FLAG および TIMESTAMP_FLAG フラグは必要ありません。)
- ENUM および SET 値は文字列として返されます。これらについては、type 値が MYSQL_TYPE_STRING で、flags 値に ENUM_FLAG または SET_FLAG フラグが設定されていることを確認します。

NUM_FLAG はカラムが数値であることを示します。これには、MYSQL_TYPE_DECIMAL、MYSQL_TYPE_NEWDECIMAL、MYSQL_TYPE_TINY、MYSQL_TYPE_SHORT、MYSQL_T および MYSQL_TYPE_YEAR の型のカラムが含まれます。

NO_DEFAULT_VALUE_FLAG は、カラムの定義に DEFAULT 句がないことを示しています。これは NULL カラム (そのようなカラムは NULL のデフォルトを持つため) または AUTO_INCREMENT カラム (これは暗黙的なデフォルト値を持ちます) には適用されません。

次の例に flags 値の一般的な使用を示します。

```
if (field->flags & NOT_NULL_FLAG)
   printf("Field cannot be null\n");
```

次の表に示す便利なマクロを使用して、flags 値のブールステータスを判断できます。

フラグのステータス	説明
IS_NOT_NULL(flags)	このフィールドが NOT NULL と定義されている場合は true
IS_PRI_KEY(flags)	このフィールドが主キーである場合は true
IS_BLOB(flags)	このフィールドが BLOB または TEXT (非推奨、代わりに field->type をテストしてください) の場合は true

- unsigned int decimals

  数値フィールドの小数点以下の桁数および (MySQL 5.6.4 以降) 時間フィールドの小数秒の精度。

- unsigned int charsetnr

  フィールドの文字セット/照合順序ペアを示す ID 番号。

  通常、結果セットの文字値は、character_set_results システム変数で示された文字セットに変換されます。この場合、charsetnr はその変数で示された文字セットに対応します。文字セットの変換は character_set_results を NULL に設定することによってサポートできます。この場合、charsetnr は元のテーブルカラムまたは式の文字セットに対応します。セクション10.1.4「接続文字セットおよび照合順序」も参照してください。

  文字列データ型のバイナリおよび非バイナリデータを区別するには、charsetnr 値が 63 であるかどうかをチェックします。その場合、文字セットは binary で、これは非バイナリデータではなく、バイナリを示します。これにより、BINARY と CHAR、VARBINARY と VARCHAR、および BLOB 型と TEXT 型を区別できます。

  charsetnr 値は SHOW COLLATION ステートメントの Id カラムまたは INFORMATION_SCHEMA COLLATIONS テーブルの ID カラムに表示される値と同じです。それらの情報ソースを使用して、特定の charsetnr 値がどの文字セットと照合順序を示しているかを確認できます。

```
mysql> SHOW COLLATION WHERE Id = 63;
+----------+---------+----+---------+----------+---------+
```

```
| Collation | Charset | Id | Default | Compiled | Sortlen |
+-----------+---------+----+---------+----------+---------+
| binary    | binary  | 63 | Yes     | Yes      |       1 |
+-----------+---------+----+---------+----------+---------+

mysql> SELECT COLLATION_NAME, CHARACTER_SET_NAME
    -> FROM INFORMATION_SCHEMA.COLLATIONS WHERE ID = 33;
+-----------------+--------------------+
| COLLATION_NAME  | CHARACTER_SET_NAME |
+-----------------+--------------------+
| utf8_general_ci | utf8               |
+-----------------+--------------------+
```

- enum enum_field_types type

  フィールドの型。type 値は次の表に示す MYSQL_TYPE_ シンボルのいずれかになります。

型の値	型の説明
MYSQL_TYPE_TINY	TINYINT フィールド
MYSQL_TYPE_SHORT	SMALLINT フィールド
MYSQL_TYPE_LONG	INTEGER フィールド
MYSQL_TYPE_INT24	MEDIUMINT フィールド
MYSQL_TYPE_LONGLONG	BIGINT フィールド
MYSQL_TYPE_DECIMAL	DECIMAL または NUMERIC フィールド
MYSQL_TYPE_NEWDECIMAL	高精度計算 DECIMAL または NUMERIC
MYSQL_TYPE_FLOAT	FLOAT フィールド
MYSQL_TYPE_DOUBLE	DOUBLE または REAL フィールド
MYSQL_TYPE_BIT	BIT フィールド
MYSQL_TYPE_TIMESTAMP	TIMESTAMP フィールド
MYSQL_TYPE_DATE	DATE フィールド
MYSQL_TYPE_TIME	TIME フィールド
MYSQL_TYPE_DATETIME	DATETIME フィールド
MYSQL_TYPE_YEAR	YEAR フィールド
MYSQL_TYPE_STRING	CHAR または BINARY フィールド
MYSQL_TYPE_VAR_STRING	VARCHAR または VARBINARY フィールド
MYSQL_TYPE_BLOB	BLOB または TEXT フィールド (max_length を使用して最大長を判断します)
MYSQL_TYPE_SET	SET フィールド
MYSQL_TYPE_ENUM	ENUM フィールド
MYSQL_TYPE_GEOMETRY	空間フィールド
MYSQL_TYPE_NULL	NULL 型フィールド

  MYSQL_TYPE_TIME2、MYSQL_TYPE_DATETIME2、および MYSQL_TYPE_TIMESTAMP2 タイプコードはサーバー側でのみ使用されます。クライアントは、MYSQL_TYPE_TIME、MYSQL_TYPE_DATETIME、および MYSQL_TYPE_TIMESTAMP コードを参照します。

  IS_NUM() マクロを使用して、フィールドが数値型を持つかどうかをテストできます。type 値を IS_NUM() に渡すと、それはフィールドが数値である場合、TRUE と評価します。

```
if (IS_NUM(field->type))
  printf("Field is numeric\n");
```

  ENUM および SET 値は文字列として返されます。これらについては、type 値が MYSQL_TYPE_STRING で、flags 値に ENUM_FLAG または SET_FLAG フラグが設定されていることを確認します。

23.7.6 C API 関数の概要

C API で使用可能な関数について、ここで概要を説明し、あとのセクションで詳しく説明します。セクション 23.7.7「C API 関数の説明」を参照してください。

表 23.4 C API 関数名と説明

関数	説明
my_init()	グローバル変数とスレッドセーフプログラムのスレッドハンドラを初期化します
mysql_affected_rows()	最後の UPDATE、DELETE、または INSERT クエリーによって変更/削除/挿入された行数を返します
mysql_autocommit()	自動コミットモードのオン/オフを切り替えます
mysql_change_user()	開いている接続のユーザーおよびデータベースを変更します
mysql_character_set_name()	現在の接続のデフォルトの文字セット名を返します
mysql_client_find_plugin()	プラグインへのポインタを返します
mysql_client_register_plugin()	プラグインを登録します
mysql_close()	サーバー接続を閉じます
mysql_commit()	トランザクションをコミットします
mysql_connect()	MySQL サーバーに接続します (この関数は非推奨です。代わりに mysql_real_connect() を使用してください)
mysql_create_db()	データベースを作成します (この関数は非推奨です。代わりに SQL ステートメント CREATE DATABASE を使用してください)
mysql_data_seek()	クエリー結果セット内の任意の行番号にシークします
mysql_debug()	指定された文字列で DBUG_PUSH を実行します
mysql_drop_db()	データベースを削除します (この関数は非推奨です。代わりに SQL ステートメント DROP DATABASE を使用してください)
mysql_dump_debug_info()	サーバーにデバッグ情報をログに書き込ませます
mysql_eof()	結果セットの最後の行が読み取られているかどうかを判断します (この関数は非推奨です。代わりに mysql_errno() または mysql_error() を使用できます)
mysql_errno()	最近呼び出された MySQL 関数のエラー番号を返します
mysql_error()	最近呼び出された MySQL 関数のエラーメッセージを返します
mysql_escape_string()	SQL ステートメントで使用するために文字列内の特殊文字をエスケープします
mysql_fetch_field()	次のテーブルフィールドの型を返します
mysql_fetch_field_direct()	フィールド番号を指定して、テーブルフィールドの型を返します
mysql_fetch_fields()	すべてのフィールド構造の配列を返します
mysql_fetch_lengths()	現在の行内のすべてのカラムの長さを返します
mysql_fetch_row()	結果セットから次の行をフェッチします
mysql_field_count()	最近のステートメントの結果カラムの数を返します
mysql_field_seek()	指定したカラムにカラムカーソルを置きます
mysql_field_tell()	最後の mysql_fetch_field() に使用されたフィールドカーソルの位置を返します
mysql_free_result()	結果セットに使用されたメモリーを解放します
mysql_get_character_set_info()	デフォルトの文字セットに関する情報を返します
mysql_get_client_info()	クライアントバージョン情報を文字列として返します
mysql_get_client_version()	クライアントバージョン情報を整数として返します
mysql_get_host_info()	接続を説明する文字列を返します
mysql_get_proto_info()	接続によって使用されたプロトコルバージョンを返します
mysql_get_server_info()	サーバーバージョン番号を返します
mysql_get_server_version()	サーバーのバージョン番号を整数として返します
mysql_get_ssl_cipher()	現在の SSL 暗号を返します
mysql_hex_string()	文字列を 16 進形式でエンコードします

関数	説明
mysql_info()	最近実行されたクエリーに関する情報を返します
mysql_init()	MYSQL 構造を取得するか初期化します
mysql_insert_id()	前のクエリーによって AUTO_INCREMENT カラムに対して生成された ID を返します
mysql_kill()	指定されたスレッドを強制終了します
mysql_library_end()	MySQL C API ライブラリをファイナライズします
mysql_library_init()	MySQL C API ライブラリを初期化します
mysql_list_dbs()	単純な正規表現に一致するデータベース名を返します
mysql_list_fields()	単純な正規表現に一致するフィールド名を返します
mysql_list_processes()	現在のサーバースレッドのリストを返します。
mysql_list_tables()	単純な正規表現に一致するテーブル名を返します
mysql_load_plugin()	プラグインをロードします
mysql_load_plugin_v()	プラグインをロードします
mysql_more_results()	それ以上の結果が存在するかどうかをチェックします
mysql_next_result()	複数結果の実行での次の結果を返すか、初期化します
mysql_num_fields()	結果セット内のカラムの数を返します
mysql_num_rows()	結果セット内の行数を返します
mysql_options()	mysql_real_connect() の接続オプションを設定します
mysql_options4()	mysql_real_connect() の接続オプションを設定します
mysql_ping()	サーバーへの接続が動作しているかどうかをチェックし、必要に応じて再接続します
mysql_plugin_options()	プラグインオプションを設定します
mysql_query()	NULL 終端文字列として指定された SQL クエリーを実行します
mysql_real_connect()	MySQL サーバーに接続します
mysql_real_escape_string()	SQL ステートメントで使用するために文字列内の特殊文字をエスケープし、接続の現在の文字セットを考慮します
mysql_real_query()	カウントされる文字列として指定された SQL クエリーを実行します
mysql_refresh()	テーブルおよびキャッシュをフラッシュするか、リセットします
mysql_reload()	サーバーに付与テーブルをリロードするように伝えます
mysql_rollback()	トランザクションをロールバックします
mysql_row_seek()	mysql_row_tell() から返される値を使用して、結果セット内の行オフセットにシークします
mysql_row_tell()	行カーソル位置を返します
mysql_select_db()	データベースを選択します
mysql_server_end()	MySQL C API ライブラリをファイナライズします
mysql_server_init()	MySQL C API ライブラリを初期化します
mysql_set_character_set()	現在の接続のデフォルトの文字セットを設定します
mysql_set_local_infile_default()	LOAD DATA LOCAL INFILE ハンドラコールバックをそれらのデフォルト値に設定します
mysql_set_local_infile_handler()	アプリケーション固有の LOAD DATA LOCAL INFILE ハンドラコールバックをインストールします
mysql_set_server_option()	接続のオプション (multi-statements など) を設定します。
mysql_sqlstate()	最後のエラーの SQLSTATE エラーコードを返します
mysql_shutdown()	データベースサーバーをシャットダウンします
mysql_ssl_set()	サーバーへの SSL 接続の確立を準備します
mysql_stat()	サーバーステータスを文字列として返します

関数	説明
mysql_store_result()	クライアントへの完全な結果セットを取得します
mysql_thread_end()	スレッドハンドラをファイナライズします
mysql_thread_id()	現在のスレッド ID を返します
mysql_thread_init()	スレッドハンドラを初期化します
mysql_thread_safe()	クライアントがスレッドセーフとしてコンパイルされた場合に 1 を返します
mysql_use_result()	行ごとの結果セットの取得を開始します
mysql_warning_count()	以前の SQL ステートメントの警告カウントを返します

アプリケーションプログラムでは、MySQL との対話にこの概要を使用してください。

1. mysql_library_init() を呼び出して、MySQL ライブラリを初期化します。この関数は、libmysqlclient C クライアントライブラリと libmysqld 組み込みサーバーライブラリのどちらにも存在するため、-libmysqlclient フラグでリンクして、通常のクライアントプログラムを構築するか、または -libmysqld フラグでリンクして、組み込みサーバーアプリケーションを構築するかどうかに関係なく使われます。

2. mysql_init() を呼び出して、接続ハンドラを初期化し、mysql_real_connect() を呼び出して、サーバーに接続します。

3. SQL ステートメントを発行して、それらの結果を処理します。(次の説明で、これを行う方法に関する詳細を提供します。)

4. mysql_close() を呼び出して、MySQL サーバーへの接続をクローズします。

5. mysql_library_end() を呼び出して、MySQL ライブラリの使用を終了します。

mysql_library_init() と mysql_library_end() の呼び出しの目的は、MySQL ライブラリの正しい初期化とファイナライズを提供することです。クライアントライブラリとリンクされたアプリケーションでは、それらは改善されたメモリー管理を提供します。mysql_library_end() を呼び出さない場合、メモリーのブロックは割り当てられたままになります。(これによって、アプリケーションによって使用されるメモリーの量は増えませんが、何らかのメモリーリーク検出ツールがそれに関して報告します。)組み込みサーバーとリンクされたアプリケーションでは、これらの呼び出しはサーバーを起動および停止します。

非マルチスレッド環境では、mysql_init() が必要に応じて自動的に mysql_library_init() を呼び出すため、この呼び出しを省略できます。ただし、mysql_library_init() はマルチスレッド環境でスレッドセーフでないため、mysql_library_init() を呼び出す mysql_init() もスレッドセーフでありません。スレッドを生成する前に mysql_library_init() を呼び出すか、さもなければ mysql_library_init() を呼び出すか、mysql_init() 経由で間接的に呼び出すかに関係なく、相互排他ロックを使用して、呼び出しを保護します。これは、ほかのすべてのクライアントライブラリ呼び出しの前に実行すべきです。

サーバーに接続するには、mysql_init() を呼び出して、接続ハンドラを初期化し、次にそのハンドラで (ホスト名、ユーザー名、およびパスワードなどのその他の情報とともに) mysql_real_connect() を呼び出します。接続時に、mysql_real_connect() は、reconnect フラグ (MYSQL 構造の一部) を、5.0.3 より古い API のバージョンでは 1 の値、または新しいバージョンでは 0 に設定します。このフラグの 1 の値は、切断された接続のためにステートメントを実行できない場合、諦める前にサーバーに再接続しようとすることを示します。mysql_options() に MYSQL_OPT_RECONNECT オプションを使用して、再接続動作を制御できます。接続を終了した場合、mysql_close() を呼び出して、それを終了させます。

接続がアクティブである間に、クライアントは mysql_query() または mysql_real_query() を使用して、SQL ステートメントをサーバーに送信できます。2 つの違いは、mysql_query() では、クエリーが NULL 終端文字列として指定されることを期待しますが、mysql_real_query() ではカウントされる文字列を期待することです。文字列にバイナリデータ (NULL バイトを含む可能性のある) が含まれている場合、mysql_real_query() を使用する必要があります。

SELECT 以外の各クエリー (INSERT、UPDATE、DELETE など) では、mysql_affected_rows() を呼び出すことによって、変更された (影響を受けた) 行数を調べることができます。

SELECT クエリーでは、選択された行を結果セットとして取得します。(一部のステートメントは、行を返すという点で、SELECT に似ていることに注目してください。これらには SHOW、DESCRIBE、および EXPLAIN が含まれます。これらのステートメントを SELECT ステートメントと同じように扱います。)

クライアントが結果セットを処理する方法は 2 つあります。1 つの方法は、mysql_store_result() を呼び出して、結果セット全体を一度に取得することです。この関数はサーバーからクエリーによって返されたすべての行を取

得して、それらをクライアントに格納します。2 つ目の方法は、クライアントが mysql_use_result() を呼び出して、行ごとに結果セットの取得を開始することです。この関数は取得を初期化しますが、実際にサーバーから行を取得しません。

どちらの場合も、mysql_fetch_row() を呼び出して、行にアクセスします。mysql_store_result() では、mysql_fetch_row() は、以前にサーバーからフェッチされた行にアクセスします。mysql_use_result() では、mysql_fetch_row() はサーバーから実際に行を取得します。mysql_fetch_lengths() を呼び出すことによって、各行内のデータのサイズに関する情報を入手できます。

結果セットの処理を終了したら、mysql_free_result() を呼び出して、それに使用されたメモリーを解放します。

2 つの取得メカニズムは補完的です。各クライアントアプリケーションにもっとも適切なアプローチを選択してください。実際には、クライアントは mysql_store_result() をより一般的に使用する傾向があります。

mysql_store_result() の利点は、行がすべてクライアントにフェッチされているため、行に順次アクセスできるだけでなく、mysql_data_seek() または mysql_row_seek() を使用して、結果セット内を前後に移動して、結果セット内の現在の行の位置を変更できることです。mysql_num_rows() を呼び出すことによって、そこに存在している行の数を知ることもできます。一方、大きな結果セットに対する mysql_store_result() のメモリー要件はきわめて高くなることがあるため、メモリー不足状況が発生する可能性が高くなります。

mysql_use_result() の利点は、一度に 1 行しか保持しないため、クライアントが結果セットのために必要とするメモリーが少なくなることです (さらに割り当てのオーバーヘッドが少ないため、mysql_use_result() の方が高速になる可能性があります)。欠点は、サーバーの動作の停止を避けるため、各行を迅速に処理する必要があり、結果セット内の行にランダムアクセスできず (行に順次アクセスしかできません)、結果セット内の行数がそれらをすべて取得するまで不明であることです。さらに、取得の途中で探していた情報が見つかったと判断しても、すべての行を取得する必要があります。

API により、クライアントは、ステートメントが SELECT であるかどうかを知ることなく、適切にステートメントに応答できます (必要に応じて行を取得します)。これを行うには、各 mysql_query() (または mysql_real_query()) のあとに、mysql_store_result() を呼び出します。結果セットの呼び出しが成功すると、ステートメントは SELECT であったため、行を読み取ることができます。結果セットの呼び出しが失敗した場合は、mysql_field_count() を呼び出して、結果が実際に期待したものであったかどうかを判断します。mysql_field_count() がゼロを返す場合、ステートメントはデータを返しておらず (それは INSERT、UPDATE、DELETE などであったことを示す)、行を返すことが期待されていません。mysql_field_count() がゼロ以外の場合、ステートメントは行を返しているべきですが、返していません。これはステートメントが失敗した SELECT であったことを示します。これを実行できる方法の例については、mysql_field_count() の説明を参照してください。

mysql_store_result() と mysql_use_result() のどちらも結果セットを構成するフィールドに関する情報 (フィールド数、それらの名前と型など) を取得できます。mysql_fetch_field() を繰り返し呼び出すか、または行内のフィールド番号ごとに mysql_fetch_field_direct() を呼び出すことによって、行内のフィールド情報に順次にアクセスできます。mysql_field_seek() を呼び出すことによって、現在のフィールドカーソル位置を変更できます。フィールドカーソルを設定すると mysql_fetch_field() へのその後の呼び出しに影響します。mysql_fetch_fields() を呼び出すことによって、フィールドの情報をすべて一度に取得することもできます。

エラーを検出して報告するために、MySQL は mysql_errno() 関数および mysql_error() 関数によって、エラー情報へのアクセスを提供します。これらは最近呼び出された成功または失敗した可能性のある関数のエラーコードやエラーメッセージを返すため、エラーが発生した時期とそれが何であったかを判断できます。

23.7.7 C API 関数の説明

ここの説明で、NULL のパラメータまたは戻り値は、MySQL NULL 値ではなく、C プログラミング言語の意味での NULL を意味します。

値を返す関数は一般にポインタまたは整数を返します。ほかに指定がないかぎり、ポインタを返す関数は、成功を示す NULL 以外の値またはエラーを示す NULL 値を返し、整数を返す関数は成功を示すゼロまたはエラーを示すゼロ以外を返します。「ゼロ以外」はそれだけを意味します。関数の説明にほかに指示がないかぎり、ゼロ以外の値に対してテストしないでください。

```
if (result)               /* correct */
   ... error ...

if (result < 0)           /* incorrect */
   ... error ...

if (result == -1)         /* incorrect */
   ... error ...
```

関数がエラーを返す場合、関数の説明の「エラー」サブセクションに、可能性のあるエラーの種類を一覧表示しています。mysql_errno() を呼び出すことによって、これらのうちどれが発生したかを知ることができます。mysql_error() を呼び出すことによって、エラーの文字列表現を取得できます。

23.7.7.1 mysql_affected_rows()

my_ulonglong mysql_affected_rows(MYSQL *mysql)

説明

mysql_affected_rows() は、mysql_query() または mysql_real_query() によるステートメントの実行直後に呼び出すことができます。それは、最後のステートメントが UPDATE、DELETE、または INSERT であった場合に、それによって変更、削除、または挿入された行数を返します。SELECT ステートメントの場合、mysql_affected_rows() は mysql_num_rows() のように動作します。

UPDATE ステートメントの場合、デフォルトで影響を受けた行の値は実際に変更された行の数です。mysqld への接続時に CLIENT_FOUND_ROWS フラグを mysql_real_connect() に指定した場合、影響を受けた行の値は「見つかった」、つまり WHERE 句に一致した行数です。

REPLACE ステートメントの場合、影響を受けた行の値は、新しい行が古い行に置き換わった場合 2 です。この場合、重複が削除されたあとに行が挿入されたためです。

INSERT ... ON DUPLICATE KEY UPDATE ステートメントの場合、行ごとの影響を受けた行の値は、その行が新しい行として挿入された場合は 1、既存の行が更新された場合は 2、既存の行がその現在の値に設定された場合は 0 です。CLIENT_FOUND_ROWS フラグを指定した場合、影響を受けた行の値は、既存の行がその現在の値に設定された場合は (0 ではなく) 1 になります。

ストアドプロシージャーの CALL ステートメントに続く mysql_affected_rows() は、プロシージャー内の最後に実行されたステートメントに対して返す値か、またはそのステートメントが -1 を返す場合は 0 を返します。プロシージャー内で、SQL レベルで ROW_COUNT() を使用して、個々のステートメントの影響を受けた行の値を取得できます。

MySQL 5.6 では、mysql_affected_rows() はより幅広いステートメントに対して有効な値を返します。詳細については、セクション12.14「情報関数」 の ROW_COUNT() の説明を参照してください。

戻り値

ゼロより大きい整数は影響を受けたか、取得された行の数を示します。ゼロは、UPDATE ステートメントに対してレコードが更新されなかったか、クエリー内の WHERE 句に一致した行がなかったか、クエリーがまだ実行されていないことを示します。-1 は、クエリーがエラーを返したか、SELECT クエリーの場合に、mysql_store_result() を呼び出す前に、mysql_affected_rows() が呼び出されたことを示します。

mysql_affected_rows() は符号なし値を返すため、戻り値を (my_ulonglong)-1 (または同等である (my_ulonglong)~0) と比較することによって、-1 をチェックできます。

エラー

なし。

例

```
char *stmt = "UPDATE products SET cost=cost*1.25
            WHERE group=10";
mysql_query(&mysql,stmt);
printf("%ld products updated",
      (long) mysql_affected_rows(&mysql));
```

23.7.7.2 mysql_autocommit()

my_bool mysql_autocommit(MYSQL *mysql, my_bool mode)

説明

mode が 1 の場合、自動コミットモードをオンに、mode が 0 の場合、オフに設定します。

戻り値

成功の場合はゼロ。エラーが発生した場合、ゼロ以外。

エラー

なし。

23.7.7.3 mysql_change_user()

my_bool mysql_change_user(MYSQL *mysql, const char *user, const char *password, const char *db)

説明

ユーザーを変更して、db で指定されたデータベースが、mysql で指定された接続でのデフォルト (現在) のデータベースになるようにします。その後のクエリーで、このデータベースは明示的なデータベース指定子を含まないテーブル参照のデフォルトになります。

mysql_change_user() は接続されたユーザーを認証できなかったか、データベースを使用する権限を持たない場合に失敗します。この場合、ユーザーとデータベースは変更されません。

デフォルトのデータベースが必要ない場合は、NULL の db パラメータを渡します。

この関数は、新しい接続を確立し、再認証されたかのように、セッションの状態をリセットします。(セクション23.7.16「自動再接続動作の制御」を参照してください。)それは常に、アクティブなトランザクションの ROLLBACK を実行し、すべての一時テーブルを閉じて削除し、ロックされているすべてのテーブルのロックを解除します。セッションシステム変数が、対応するグローバルシステム変数の値にリセットされます。プリペアドステートメントが解放され、HANDLER 変数が閉じられます。GET_LOCK() によって取得されたロックが解放されます。これらの効果は、ユーザーが変更しなかった場合でも発生します。

戻り値

成功の場合はゼロ。エラーが発生した場合、ゼロ以外。

エラー

mysql_real_connect() から取得する可能性があるものに加えて:

- CR_COMMANDS_OUT_OF_SYNC

  コマンドが正しくない順番で実行されました。

- CR_SERVER_GONE_ERROR

  MySQL サーバーが存在しなくなりました。

- CR_SERVER_LOST

  サーバーへの接続がクエリー中に失われました。

- CR_UNKNOWN_ERROR

  不明なエラーが発生しました。

- ER_UNKNOWN_COM_ERROR

  MySQL サーバーはこのコマンドを実装しません (古いサーバーの可能性があります)。

- ER_ACCESS_DENIED_ERROR

  ユーザーまたはパスワードが誤っています。

- ER_BAD_DB_ERROR

  データベースが存在していません。

- ER_DBACCESS_DENIED_ERROR

  ユーザーはデータベースに対するアクセス権を持っていませんでした。

- ER_WRONG_DB_NAME

データベース名が長すぎます。

例

```
if (mysql_change_user(&mysql, "user", "password", "new_database"))
{
  fprintf(stderr, "Failed to change user.  Error: %s\n",
          mysql_error(&mysql));
}
```

23.7.7.4 mysql_character_set_name()

```
const char *mysql_character_set_name(MYSQL *mysql)
```

説明

現在の接続のデフォルトの文字セット名を返します。

戻り値

デフォルトの文字セット名

エラー

なし。

23.7.7.5 mysql_close()

```
void mysql_close(MYSQL *mysql)
```

説明

以前にオープンされた接続をクローズします。mysql_close() は、mysql によって指示された接続ハンドルが mysql_init() または mysql_connect() によって自動的に割り当てられている場合に、そのハンドルの割り当ても解除します。

戻り値

なし。

エラー

なし。

23.7.7.6 mysql_commit()

```
my_bool mysql_commit(MYSQL *mysql)
```

説明

現在のトランザクションをコミットします。

この関数のアクションは、completion_type システム変数の値に影響を受けます。特に、completion_type の値が RELEASE (または 2) の場合、トランザクションの終了後、サーバーは解放を実行し、クライアント接続を閉じます。クライアントプログラムから mysql_close() を呼び出して、クライアント側から接続をクローズします。

戻り値

成功の場合はゼロ。エラーが発生した場合、ゼロ以外。

エラー

なし。

23.7.7.7 mysql_connect()

MYSQL *mysql_connect(MYSQL *mysql, const char *host, const char *user, const char *passwd)

説明

この関数は非推奨です。代わりに mysql_real_connect() を使用します。

mysql_connect() はホストで実行している MySQL データベースエンジンへの接続を確立します。mysql_connect() は、mysql_get_client_info() を除き、ほかのすべての API 関数を実行する前に正常に完了している必要があります。

パラメータの意味は、接続パラメータを NULL にできることを除き、mysql_real_connect() の対応するパラメータの場合と同じです。この場合、C API は、自動的に接続構造にメモリーを割り当て、mysql_close() が呼び出されると、それを解放します。このアプローチの短所は、接続が失敗した場合にエラーメッセージを取得できないことです。(mysql_errno() または mysql_error() から、エラー情報を得るには、有効な MYSQL ポインタを提供する必要があります。)

戻り値

mysql_real_connect() の場合と同じ。

エラー

mysql_real_connect() の場合と同じ。

23.7.7.8 mysql_create_db()

int mysql_create_db(MYSQL *mysql, const char *db)

説明

db パラメータによって指定されたデータベースを作成します。

この関数は非推奨です。代わりに、mysql_query() を使用して、SQL CREATE DATABASE ステートメントを発行することをお勧めします。

戻り値

成功の場合はゼロ。エラーが発生した場合、ゼロ以外。

エラー

- CR_COMMANDS_OUT_OF_SYNC

  コマンドが正しくない順番で実行されました。

- CR_SERVER_GONE_ERROR

  MySQL サーバーが存在しなくなりました。

- CR_SERVER_LOST

  サーバーへの接続がクエリー中に失われました。

- CR_UNKNOWN_ERROR

  不明なエラーが発生しました。

例

```
if(mysql_create_db(&mysql, "my_database"))
{
  fprintf(stderr, "Failed to create new database.  Error: %s\n",
          mysql_error(&mysql));
}
```

23.7.7.9 mysql_data_seek()

void mysql_data_seek(MYSQL_RES *result, my_ulonglong offset)

説明

クエリー結果セット内の任意の行にシークします。offset 値は行番号です。0 から mysql_num_rows(result)-1 までの範囲の値を指定します。

この関数は、結果セット構造にクエリーの結果全体を格納する必要があるため、mysql_data_seek() は、mysql_use_result() ではなく、mysql_store_result() とのみ一緒に使用できます。

戻り値

なし。

エラー

なし。

23.7.7.10 mysql_debug()

void mysql_debug(const char *debug)

説明

指定された文字列で DBUG_PUSH を実行します。mysql_debug() は Fred Fish デバッグライブラリを使用します。この関数を使うには、デバッグをサポートするように、クライアントライブラリをコンパイルする必要があります。セクション24.4.3「DBUG パッケージ」を参照してください。

戻り値

なし。

エラー

なし。

例

ここに示す呼び出しによって、クライアントライブラリに、クライアントマシンの /tmp/client.trace 内にトレースファイルを生成させます。

```
mysql_debug("d:t:O,/tmp/client.trace");
```

23.7.7.11 mysql_drop_db()

int mysql_drop_db(MYSQL *mysql, const char *db)

説明

db パラメータによって指定されたデータベースを削除します。

この関数は非推奨です。代わりに、mysql_query() を使用して、SQL DROP DATABASE ステートメントを発行することをお勧めします。

戻り値

成功の場合はゼロ。エラーが発生した場合、ゼロ以外。

エラー

- CR_COMMANDS_OUT_OF_SYNC

  コマンドが正しくない順番で実行されました。

- CR_SERVER_GONE_ERROR

MySQL サーバーが存在しなくなりました。

- CR_SERVER_LOST

  サーバーへの接続がクエリー中に失われました。

- CR_UNKNOWN_ERROR

  不明なエラーが発生しました。

例

```
if(mysql_drop_db(&mysql, "my_database"))
  fprintf(stderr, "Failed to drop the database: Error: %s\n",
          mysql_error(&mysql));
```

23.7.7.12 mysql_dump_debug_info()

```
int mysql_dump_debug_info(MYSQL *mysql)
```

説明

サーバーにデバッグ情報をエラーログに書き込むように指示します。接続されたユーザーは SUPER 権限を持っている必要があります。

戻り値

成功の場合はゼロ。エラーが発生した場合、ゼロ以外。

エラー

- CR_COMMANDS_OUT_OF_SYNC

  コマンドが正しくない順番で実行されました。

- CR_SERVER_GONE_ERROR

  MySQL サーバーが存在しなくなりました。

- CR_SERVER_LOST

  サーバーへの接続がクエリー中に失われました。

- CR_UNKNOWN_ERROR

  不明なエラーが発生しました。

23.7.7.13 mysql_eof()

```
my_bool mysql_eof(MYSQL_RES *result)
```

説明

この関数は非推奨です。代わりに mysql_errno() または mysql_error() を使用できます。

mysql_eof() は結果セットの最後の行が読み取られているかどうかを判断します。

mysql_store_result() への成功した呼び出しから結果セットを取得する場合、クライアントは 1 回の操作でセット全体を受け取ります。この場合、mysql_fetch_row() からの NULL の戻り値は常に、結果セットの終わりに達したため、mysql_eof() を呼び出す必要がないことを意味します。mysql_store_result() と一緒に使うと、mysql_eof() は常に true を返します。

一方、mysql_use_result() を使用して、結果セットの取得を開始する場合、mysql_fetch_row() を繰り返し呼び出すと、セットの行がサーバーから 1 つずつ取得されます。このプロセス中に接続でエラーが発生する可能性があるため、mysql_fetch_row() からの NULL の戻り値は、必ずしも結果セットの終わりに正常に達したことを意味するとはかぎりません。この場合、mysql_eof() を使用して、何が発生したかを判断できます。mysql_eof() は、結果セットの終わりに達した場合にゼロ以外の値を返し、エラーが発生した場合にゼロを返します。

歴史的に、mysql_eof() は、標準 MySQL エラー関数の mysql_errno() や mysql_error() より前から存在しています。それらのエラー関数は同じ情報を提供するため、非推奨である mysql_eof() よりそれらの使用が優先されます。(実際、それらはより多くの情報を提供します。エラー関数はエラーが発生するとその理由を示すのに対して、mysql_eof() はブール値しか返さないためです。)

戻り値

成功の場合はゼロ。結果セットの終わりに達した場合はゼロ以外。

エラー

なし。

例

次の例は、mysql_eof() の使用方法を示しています。

```
mysql_query(&mysql,"SELECT * FROM some_table");
result = mysql_use_result(&mysql);
while((row = mysql_fetch_row(result)))
{
   // do something with data
}
if(!mysql_eof(result))  // mysql_fetch_row() failed due to an error
{
   fprintf(stderr, "Error: %s\n", mysql_error(&mysql));
}
```

ただし、標準 MySQL エラー関数で同じ効果を達成できます。

```
mysql_query(&mysql,"SELECT * FROM some_table");
result = mysql_use_result(&mysql);
while((row = mysql_fetch_row(result)))
{
   // do something with data
}
if(mysql_errno(&mysql))  // mysql_fetch_row() failed due to an error
{
   fprintf(stderr, "Error: %s\n", mysql_error(&mysql));
}
```

23.7.7.14 mysql_errno()

unsigned int mysql_errno(MYSQL *mysql)

説明

mysql で指定された接続に対し、mysql_errno() は、成功したか失敗した可能性のある最近呼び出された API 関数のエラーコードを返します。ゼロの戻り値はエラーが発生しなかったことを意味します。クライアントのエラーメッセージ番号は、MySQL errmsg.h ヘッダーファイルに一覧表示されています。サーバーのエラーメッセージ番号は、mysqld_error.h に一覧表示されています。エラーは付録B「エラー、エラーコード、および一般的な問題」にも一覧表示しています。

mysql_fetch_row() のような一部の関数は成功した場合に、mysql_errno() を設定しません。

原則として、サーバーに情報を求める必要があるすべての関数は、成功した場合に mysql_errno() をリセットします。

mysql_errno() によって返される MySQL 固有のエラー番号は、mysql_sqlstate() によって返される SQLSTATE 値とは異なります。たとえば、mysql クライアントプログラムは、次の形式を使用してエラーを表示します。ここで、1146 は mysql_errno() 値で、'42S02' は対応する mysql_sqlstate() 値です。

```
shell> SELECT * FROM no_such_table;
ERROR 1146 (42S02): Table 'test.no_such_table' doesn't exist
```

戻り値

最後の mysql_xxx() 呼び出しが失敗した場合は、そのエラーコード値。ゼロはエラーが発生しなかったことを意味します。

エラー

なし。

23.7.7.15 mysql_error()

const char *mysql_error(MYSQL *mysql)

説明

mysql で指定された接続に対し、mysql_error() は、最近呼び出されて失敗した API 関数のエラーメッセージを含む、NULL 終端文字列を返します。関数が失敗しなかった場合、mysql_error() の戻り値は前のエラーか、またはエラーがないことを示す空の文字列になります。

原則として、サーバーに情報を求める必要があるすべての関数は、成功した場合に mysql_error() をリセットします。

mysql_error() をリセットする関数の場合、これらの 2 つのテストのどちらかを使用して、エラーをチェックできます。

```
if(*mysql_error(&mysql))
{
  // an error occurred
}

if(mysql_error(&mysql)[0])
{
  // an error occurred
}
```

クライアントエラーメッセージの言語は、MySQL クライアントライブラリを再コンパイルすることによって変更できます。現在、数種類の言語のエラーメッセージを選択できます。セクション10.2「エラーメッセージ言語の設定」を参照してください。

戻り値

エラーを説明する NULL 終端文字列。エラーが発生しなかった場合は空の文字列。

エラー

なし。

23.7.7.16 mysql_escape_string()

代わりに mysql_real_escape_string() を使用してください。

この関数は、mysql_real_escape_string() がその最初の引数として、接続ハンドラをとり、文字列を現在の文字セットに従ってエスケープすることを除き、mysql_real_escape_string() と同じです。mysql_escape_string() は接続引数をとらず、現在の文字セットを考慮しません。

23.7.7.17 mysql_fetch_field()

MYSQL_FIELD *mysql_fetch_field(MYSQL_RES *result)

説明

MYSQL_FIELD 構造として設定された結果セットの 1 つのカラムの定義を返します。この関数を繰り返し呼び出して、結果セット内のすべてのカラムに関する情報を取得します。それ以上残っているフィールドがなくなると、mysql_fetch_field() は NULL を返します。

新しい SELECT クエリーを実行するたびに、mysql_fetch_field() はリセットされて、最初のフィールドに関する情報を返します。mysql_fetch_field() によって返されるフィールドは、mysql_field_seek() への呼び出しによっても影響を受けます。

mysql_query() を呼び出して、テーブルに対する SELECT を実行したが、mysql_store_result() を呼び出さなかった場合、mysql_fetch_field() を呼び出して、BLOB フィールドの長さを求めると、MySQL はデフォルト BLOB の長さ (8K バイト) を返します。(MySQL は BLOB の最大長を知らないため、8K バイトのサイズが選択されます。

これはいずれ構成可能にされます。)結果セットを取得したら、field->max_length には、特定のクエリー内のこのカラムの最大値の長さが格納されます。

戻り値

現在のカラムの MYSQL_FIELD 構造。カラムが残っていない場合は NULL。

エラー

なし。

例

```
MYSQL_FIELD *field;

while((field = mysql_fetch_field(result)))
{
    printf("field name %s\n", field->name);
}
```

23.7.7.18 mysql_fetch_field_direct()

MYSQL_FIELD *mysql_fetch_field_direct(MYSQL_RES *result, unsigned int fieldnr)

説明

結果セット内のカラムのフィールド番号 fieldnr を指定すると、そのカラムのフィールド定義が MYSQL_FIELD 構造として返されます。この関数を使用して、任意のカラムの定義を取得します。0 から mysql_num_fields(result)-1 までの範囲で、fieldnr の値を指定します。

戻り値

指定したカラムの MYSQL_FIELD 構造。

エラー

なし。

例

```
unsigned int num_fields;
unsigned int i;
MYSQL_FIELD *field;

num_fields = mysql_num_fields(result);
for(i = 0; i < num_fields; i++)
{
    field = mysql_fetch_field_direct(result, i);
    printf("Field %u is %s\n", i, field->name);
}
```

23.7.7.19 mysql_fetch_fields()

MYSQL_FIELD *mysql_fetch_fields(MYSQL_RES *result)

説明

結果セットのすべての MYSQL_FIELD 構造の配列を返します。各構造は、結果セットの 1 つのカラムのフィールドの定義を示します。

戻り値

結果セットのすべてのカラムの MYSQL_FIELD 構造の配列。

エラー

なし。

例

```
unsigned int num_fields;
unsigned int i;
MYSQL_FIELD *fields;

num_fields = mysql_num_fields(result);
fields = mysql_fetch_fields(result);
for(i = 0; i < num_fields; i++)
{
   printf("Field %u is %s\n", i, fields[i].name);
}
```

23.7.7.20 mysql_fetch_lengths()

```
unsigned long *mysql_fetch_lengths(MYSQL_RES *result)
```

説明

結果セット内の現在の行のカラムの長さを返します。フィールド値をコピーする予定がある場合、この長さ情報は、strlen() の呼び出しを避けることができるため、最適化にも役立ちます。さらに、結果セットにバイナリデータが含まれている場合、strlen() は NULL 文字を含むフィールドについて誤った結果を返すため、この関数を使用して、データのサイズを判断する必要があります。

空のカラムおよび NULL 値を含むカラムの長さはゼロです。これらの 2 つのケースを区別する方法を確認するには、mysql_fetch_row() の説明を参照してください。

戻り値

各カラム (終端の NULL 文字を含まない) のサイズを表す符号なし long 整数の配列。エラーが発生した場合は NULL。

エラー

mysql_fetch_lengths() は結果セットの現在の行に対してのみ有効です。それを mysql_fetch_row() を呼び出す前または結果内のすべての行を取得したあとに呼び出すと、それは NULL を返します。

例

```
MYSQL_ROW row;
unsigned long *lengths;
unsigned int num_fields;
unsigned int i;

row = mysql_fetch_row(result);
if (row)
{
   num_fields = mysql_num_fields(result);
   lengths = mysql_fetch_lengths(result);
   for(i = 0; i < num_fields; i++)
   {
       printf("Column %u is %lu bytes in length.\n",
            i, lengths[i]);
   }
}
```

23.7.7.21 mysql_fetch_row()

```
MYSQL_ROW mysql_fetch_row(MYSQL_RES *result)
```

説明

結果セットの次の行を取得します。mysql_store_result() のあとで使用すると、mysql_fetch_row() は、それ以上取得する行がない場合に、NULL を返します。mysql_use_result() のあとで使用すると、mysql_fetch_row() は、それ以上取得する行がないか、エラーが発生した場合に NULL を返します。

行内の値の数は mysql_num_fields(result) によって指定されます。row が mysql_fetch_row() への呼び出しからの戻り値を保持している場合、その値へのポインタは、row[0] から row[mysql_num_fields(result)-1] までとしてアクセスされます。行内の NULL 値は NULL ポインタによって示されます。

行内のフィールド値の長さは、mysql_fetch_lengths() を呼び出すことによって取得できます。空のフィールドと NULL を含むフィールドはどちらも長さが 0 です。これらを区別するには、フィールド値のポインタをチェックします。ポインタが NULL である場合、フィールドは NULL です。そうでない場合、フィールドは空です。

戻り値

次の行の MYSQL_ROW 構造。それ以上取得する行がないか、エラーが発生した場合は NULL。

エラー

mysql_fetch_row() の呼び出しと呼び出しの間に、エラーはリセットされません。

- CR_SERVER_LOST

  サーバーへの接続がクエリー中に失われました。

- CR_UNKNOWN_ERROR

  不明なエラーが発生しました。

例

```
MYSQL_ROW row;
unsigned int num_fields;
unsigned int i;

num_fields = mysql_num_fields(result);
while ((row = mysql_fetch_row(result)))
{
  unsigned long *lengths;
  lengths = mysql_fetch_lengths(result);
  for(i = 0; i < num_fields; i++)
  {
     printf("[%.*s] ", (int) lengths[i],
            row[i] ? row[i] : "NULL");
  }
  printf("\n");
}
```

23.7.7.22 mysql_field_count()

unsigned int mysql_field_count(MYSQL *mysql)

説明

接続での最近のクエリーのカラムの数を返します。

この関数を通常使用するときは、mysql_store_result() が NULL を返した (および、そのために結果セットポインタがない) ときです。この場合、mysql_field_count() を呼び出して、mysql_store_result() が空でない結果を生成しているかどうかを判断できます。これにより、クライアントプログラムは、クエリーが SELECT (または SELECT に似た) ステートメントであったかどうかを知らなくても、正しいアクションをとることができます。ここに示す例では、これを実行する方法を説明しています。

セクション23.7.15.1「mysql_query() が成功を返したあとに mysql_store_result() が NULL を返すことがあるのはなぜか」を参照してください。

戻り値

結果セット内のカラムの数を表す符号なし整数。

エラー

なし。

例

```
MYSQL_RES *result;
unsigned int num_fields;
unsigned int num_rows;
```

```
if (mysql_query(&mysql,query_string))
{
   // error
}
else // query succeeded, process any data returned by it
{
   result = mysql_store_result(&mysql);
   if (result)  // there are rows
   {
      num_fields = mysql_num_fields(result);
      // retrieve rows, then call mysql_free_result(result)
   }
   else  // mysql_store_result() returned nothing; should it have?
   {
      if(mysql_field_count(&mysql) == 0)
      {
         // query does not return data
         // (it was not a SELECT)
         num_rows = mysql_affected_rows(&mysql);
      }
      else // mysql_store_result() should have returned data
      {
         fprintf(stderr, "Error: %s\n", mysql_error(&mysql));
      }
   }
}
```

代替方法は mysql_field_count(&mysql) 呼び出しを mysql_errno(&mysql) で置き換えることです。この場合、mysql_field_count() の値から、ステートメントが SELECT であったかどうかを推定するのではなく、mysql_store_result() から直接、エラーをチェックします。

23.7.7.23 mysql_field_seek()

MYSQL_FIELD_OFFSET mysql_field_seek(MYSQL_RES *result, MYSQL_FIELD_OFFSET offset)

説明

フィールドカーソルを、指定されたオフセットに設定します。mysql_fetch_field() への次の呼び出しは、そのオフセットに関連付けられているカラムのフィールド定義を取得します。

行の先頭にシークするには、ゼロの offset 値を渡します。

戻り値

フィールドカーソルの以前の値。

エラー

なし。

23.7.7.24 mysql_field_tell()

MYSQL_FIELD_OFFSET mysql_field_tell(MYSQL_RES *result)

説明

最後の mysql_fetch_field() に使用されたフィールドカーソルの位置を返します。この値は mysql_field_seek() への引数として使用できます。

戻り値

フィールドカーソルの現在のオフセット。

エラー

なし。

23.7.7.25 mysql_free_result()

void mysql_free_result(MYSQL_RES *result)

説明

mysql_store_result()、mysql_use_result()、mysql_list_dbs() などによって結果セットに割り当てられたメモリーを解放します。結果セットの処理が終了したら、mysql_free_result() を呼び出して、それが使用しているメモリーを解放する必要があります。

それを解放したあとに、結果セットにアクセスしようとしないでください。

戻り値

なし。

エラー

なし。

23.7.7.26 mysql_get_character_set_info()

void mysql_get_character_set_info(MYSQL *mysql, MY_CHARSET_INFO *cs)

説明

この関数はデフォルトのクライアント文字セットに関する情報を提供します。デフォルトの文字セットは mysql_set_character_set() 関数で変更できます。

例

この例に、MY_CHARSET_INFO 構造の使用可能なフィールドを示します。

```
if (!mysql_set_character_set(&mysql, "utf8"))
{
  MY_CHARSET_INFO cs;
  mysql_get_character_set_info(&mysql, &cs);
  printf("character set information:\n");
  printf("character set+collation number: %d\n", cs.number);
  printf("character set name: %s\n", cs.name);
  printf("collation name: %s\n", cs.csname);
  printf("comment: %s\n", cs.comment);
  printf("directory: %s\n", cs.dir);
  printf("multi byte character min. length: %d\n", cs.mbminlen);
  printf("multi byte character max. length: %d\n", cs.mbmaxlen);
}
```

23.7.7.27 mysql_get_client_info()

const char *mysql_get_client_info(void)

説明

"5.6.23" など、MySQL クライアントライブラリバージョンを表す文字列を返します。

この関数値は MySQL バージョンです。Connector/C では、これは Connector/C 配布がベースにしている MySQL のバージョンです。詳細については、セクション23.7.4.4「C API サーバーおよびクライアントライブラリのバージョン」を参照してください。

戻り値

MySQL クライアントライブラリバージョンを表す文字列。

エラー

なし。

23.7.7.28 mysql_get_client_version()

unsigned long mysql_get_client_version(void)

説明

MySQL クライアントライブラリバージョンを表す整数を返します。値の形式は XYYZZ であり、ここで X はメジャーバージョン、YY はリリースレベル (またはマイナーバージョン)、および ZZ はリリースレベル内のサブバージョンです。

```
major_version*10000 + release_level*100 + sub_version
```

たとえば、"5.6.23" は 50623 と返されます。

この関数値は MySQL バージョンです。Connector/C では、これは Connector/C 配布がベースにしている MySQL のバージョンです。詳細については、セクション23.7.4.4「C API サーバーおよびクライアントライブラリのバージョン」を参照してください。

戻り値

MySQL クライアントライブラリバージョンを表す整数。

エラー

なし。

23.7.7.29 mysql_get_host_info()

```
const char *mysql_get_host_info(MYSQL *mysql)
```

説明

サーバーホスト名を含む使用している接続の種類を示す文字列を返します。

戻り値

サーバーホスト名および接続の種類を表す文字列。

エラー

なし。

23.7.7.30 mysql_get_proto_info()

```
unsigned int mysql_get_proto_info(MYSQL *mysql)
```

説明

現在の接続で使用されているプロトコルのバージョンを返します。

戻り値

現在の接続で使用されているプロトコルのバージョンを表す符号なし整数。

エラー

なし。

23.7.7.31 mysql_get_server_info()

```
const char *mysql_get_server_info(MYSQL *mysql)
```

説明

"5.6.23" など、MySQL サーバーバージョンを表す文字列を返します。

戻り値

MySQL サーバーバージョンを表す文字列。

エラー

なし。

23.7.7.32 mysql_get_server_version()

```
unsigned long mysql_get_server_version(MYSQL *mysql)
```

説明

MySQL サーバーバージョンを表す整数を返します。値の形式は XYYZZ であり、ここで X はメジャーバージョン、YY はリリースレベル (またはマイナーバージョン)、および ZZ はリリースレベル内のサブバージョンです。

```
major_version*10000 + release_level*100 + sub_version
```

たとえば、"5.6.23" は 50623 と返されます。

この関数は、クライアントプログラムで、何らかのバージョン固有のサーバー機能が存在するかどうかを判断するために役立ちます。

戻り値

MySQL サーバーバージョンを表す整数。

エラー

なし。

23.7.7.33 mysql_get_ssl_cipher()

```
const char *mysql_get_ssl_cipher(MYSQL *mysql)
```

説明

mysql_get_ssl_cipher() はサーバーへの指定された接続で使用される SSL 暗号を返します。mysql は mysql_init() から返される接続ハンドラです。

戻り値

接続に使われる SSL 暗号を指定する文字列、または暗号が使われていない場合は NULL。

23.7.7.34 mysql_hex_string()

```
unsigned long mysql_hex_string(char *to, const char *from, unsigned long length)
```

説明

この関数は、SQL ステートメントで使用できる正当な SQL 文字列を作成するために使われます。セクション 9.1.1「文字列リテラル」を参照してください。

from 内の文字列は、各文字が 2 桁の 16 進数としてエンコードされた 16 進形式にエンコードされます。結果は to に置かれ、終端の NULL バイトが付加されます。

from によって指示される文字列は、length バイトの長さである必要があります。to バッファーを少なくとも length*2+1 バイト長になるように割り当てる必要があります。mysql_hex_string() が戻ると、to の内容は NULL 終端文字列になっています。戻り値は、終端の NULL 文字を含まない、エンコードされた文字列の長さです。

戻り値は 0xvalue または X'value' 形式を使用して、SQL ステートメントに配置できます。ただし、戻り値には、0x または X'...' は含まれません。呼び出し元は、要求されたものをなんでも提供する必要があります。

例

```
char query[1000],*end;
```

```
end = strmov(query,"INSERT INTO test_table values(");
end = strmov(end,"0x");
end += mysql_hex_string(end,"What is this",12);
end = strmov(end,",0x");
end += mysql_hex_string(end,"binary data: \0\r\n",16);
*end++ = ')';

if (mysql_real_query(&mysql,query,(unsigned int) (end - query)))
{
  fprintf(stderr, "Failed to insert row, Error: %s\n",
          mysql_error(&mysql));
}
```

例で使用している strmov() 関数は、libmysqlclient ライブラリに含まれ、strcpy() のように機能しますが、最初のパラメータの終端の NULL へのポインタを返します。

戻り値

to に置かれた、終端の NULL 文字を含まない値の長さ。

エラー

なし。

23.7.7.35 mysql_info()

const char *mysql_info(MYSQL *mysql)

説明

最近実行されたステートメント、ただしここに示すステートメントのみに関する情報を提供する文字列を取得します。ほかのステートメントに対しては、mysql_info() は NULL を返します。ここで説明するように、文字列の書式はステートメントの種類によって変わります。数字は一例にすぎず、文字列にはステートメントに適切な値が含まれます。

- INSERT INTO ... SELECT ...

  文字列の書式: レコード数: 100 重複: 0 警告: 0

- INSERT INTO ... VALUES (...),(...),(...)...

  文字列の書式: レコード数: 3 重複: 0 警告: 0

- LOAD DATA INFILE ...

  文字列の書式: レコード数: 1 削除: 0 スキップ: 0 警告: 0

- ALTER TABLE

  文字列の書式: レコード数: 3 重複: 0 警告: 0

- UPDATE

  文字列の書式: 該当した行: 40 変更: 40 警告: 0

mysql_info() は INSERT ... VALUES ステートメントの複数行形式に対してのみ、NULL 以外の値を返します (つまり、複数値のリストが指定されている場合のみ)。

戻り値

最近実行されたステートメントに関する追加情報を表す文字列。ステートメントに対する情報を使用できない場合は NULL。

エラー

なし。

23.7.7.36 mysql_init()

MYSQL *mysql_init(MYSQL *mysql)

説明

mysql_real_connect() に適切な MYSQL オブジェクトを割り当てるか初期化します。mysql が NULL ポインタである場合、関数は新しいオブジェクトを割り当てて、初期化して返します。そうでない場合、オブジェクトは初期化され、オブジェクトのアドレスが返されます。mysql_init() が新しいオブジェクトを割り当てた場合、それは、mysql_close() が呼び出され、接続がクローズされると解放されます。

非マルチスレッド環境では、mysql_init() は、必要に応じて自動的に mysql_library_init() を呼び出します。ただし、mysql_library_init() はマルチスレッド環境でスレッドセーフでないため、mysql_init() もスレッドセーフでありません。mysql_init() を呼び出す前に、スレッドの生成前に mysql_library_init() を呼び出すか、または相互排他ロックを使用して、mysql_library_init() 呼び出しを保護します。これは、ほかのすべてのクライアントライブラリ呼び出しの前に実行すべきです。

戻り値

初期化された MYSQL* ハンドル。新しいオブジェクトを割り当てるために十分なメモリーがなかった場合は NULL。

エラー

メモリーが不十分な場合、NULL が返されます。

23.7.7.37 mysql_insert_id()

my_ulonglong mysql_insert_id(MYSQL *mysql)

説明

前の INSERT または UPDATE ステートメントによって、AUTO_INCREMENT カラムに生成された値を返します。この関数は、AUTO_INCREMENT フィールドを含むテーブルへの INSERT ステートメントを実行したか、INSERT または UPDATE を使用して、LAST_INSERT_ID(expr) によってカラム値を設定した後に使用します。

mysql_insert_id() の戻り値は、次のいずれかの条件で、明示的に更新されないかぎり、常にゼロです。

- 値を AUTO_INCREMENT カラムに格納する INSERT ステートメント。これは値が、特殊な値 NULL や 0 をカラムに格納することによって自動的に生成されるか、明示的な特殊でない値であるかに関係なく当てはまります。
- 複数行の INSERT ステートメントの場合、mysql_insert_id() は、最初に正常に挿入された、自動生成された AUTO_INCREMENT 値を返します。

  正常に挿入された行がない場合、mysql_insert_id() は 0 を返します。
- INSERT ... SELECT ステートメントが実行され、正常に挿入された自動生成された値がない場合、mysql_insert_id() は最後に挿入された行の ID を返します。
- INSERT ... SELECT ステートメントで LAST_INSERT_ID(expr) を使用した場合、mysql_insert_id() は expr を返します。
- LAST_INSERT_ID(expr) を任意のカラムに挿入するか、または任意のカラムを LAST_INSERT_ID(expr) に更新することによって、AUTO_INCREMENT 値を生成するステートメントを INSERT します。
- 前のステートメントがエラーを返した場合、mysql_insert_id() の値は未定義になります。

mysql_insert_id() の戻り値は、次のシーケンスに簡略化できます。

1. AUTO_INCREMENT カラムがあり、自動的に生成された値が正常に挿入された場合、それらの最初の値を返します。
2. ステートメントで LAST_INSERT_ID(expr) が行われた場合、影響を受けるテーブルに AUTO_INCREMENT カラムがあった場合でも、expr を返します。
3. 戻り値は使用されたステートメントによって変わります。INSERT ステートメントのあとに呼び出された場合:

- テーブルに AUTO_INCREMENT カラムがあり、テーブルに正常に挿入されたこのカラムのいくつかの明示的な値があった場合、最後の明示的な値を返します。

INSERT ... ON DUPLICATE KEY UPDATE ステートメントのあとに呼び出された場合:

- テーブルに AUTO_INCREMENT カラムがあり、テーブルにいくつかの明示的な正常に挿入された値またはいくつかの更新された値があった場合、最後に挿入されたか、または更新された値を返します。

前のステートメントが AUTO_INCREMENT 値を使用していない場合、mysql_insert_id() は 0 を返します。値をあとのために保存する必要がある場合、値を生成するステートメントの直後に mysql_insert_id() を呼び出してください。

mysql_insert_id() の値は、現在のクライアント接続内で発行されたステートメントによってのみ影響を受けます。それは、ほかのクライアントによって発行されたステートメントに影響を受けません。

LAST_INSERT_ID() SQL 関数は、正常に挿入された、最初に自動生成された値を格納します。LAST_INSERT_ID() の値はサーバーで保持されるため、この関数はステートメント間でリセットされません。mysql_insert_id() とのもう 1 つの違いは、AUTO_INCREMENT カラムを特定の特殊でない値に設定した場合に、LAST_INSERT_ID() が更新されないことです。セクション12.14「情報関数」を参照してください。

mysql_insert_id() は、AUTO_INCREMENT 値を生成するストアドプロシージャーの CALL ステートメントのあとに、0 を返します。この場合、mysql_insert_id() が CALL に適用され、プロシージャー内のステートメントには適用されないためです。プロシージャー内で、LAST_INSERT_ID() を SQL レベルで使用して、AUTO_INCREMENT 値を取得できます。

LAST_INSERT_ID() と mysql_insert_id() の違いの理由は、LAST_INSERT_ID() がスクリプトで使いやすく、mysql_insert_id() は AUTO_INCREMENT カラムに起こったことに関してより正確な情報を提供しようとすることです。

戻り値

先述の説明で示しています。

エラー

なし。

23.7.7.38 mysql_kill()

```
int mysql_kill(MYSQL *mysql, unsigned long pid)
```

説明

サーバーに、pid で指定されたスレッドを強制終了するように求めます。

この関数は非推奨です。代わりに SQL KILL ステートメントを発行するために、mysql_query() を使用することをお勧めします。

mysql_kill() は 32 ビットより大きい値を処理できませんが、MySQL 5.6.9 以降、誤ったスレッドの強制終了に対して保護するため、これらの場合にエラーを返します。

- 32 ビットより大きい ID が指定された場合、mysql_kill() は CR_INVALID_CONN_HANDLE エラーを返します。
- サーバーの内部スレッド ID カウンタが 32 ビットより大きい値に達したら、それは mysql_kill() のすべての呼び出しに対して、ER_DATA_OUT_OF_RANGE エラーを返し、mysql_kill() が失敗します。

戻り値

成功の場合はゼロ。エラーが発生した場合、ゼロ以外。

エラー

- CR_COMMANDS_OUT_OF_SYNC

  コマンドが正しくない順番で実行されました。

- CR_INVALID_CONN_HANDLE

  pid が 32 ビットより大きいです。

- CR_SERVER_GONE_ERROR

  MySQL サーバーが存在しなくなりました。

- CR_SERVER_LOST

  サーバーへの接続がクエリー中に失われました。

- CR_UNKNOWN_ERROR

  不明なエラーが発生しました。

- ER_DATA_OUT_OF_RANGE

  サーバーの内部スレッド ID カウンタが 32 ビットより大きい値に達しており、その時点で、それはすべての mysql_kill() の呼び出しを拒否します。

23.7.7.39 mysql_library_end()

```
void mysql_library_end(void)
```

説明

この関数は MySQL ライブラリをファイナライズします。ライブラリの使用を終了したとき (サーバーからの切断後など) に、呼び出します。呼び出しによって行われるアクションは、アプリケーションが MySQL クライアントライブラリにリンクされているか、または MySQL 組み込みサーバーライブラリにリンクされているかによって異なります。-lmysqlclient フラグを使用して、libmysqlclient ライブラリに対してリンクされたクライアントプログラムの場合、mysql_library_end() は、何らかのメモリー管理を実行して、クリーンアップします。-lmysqld フラグを使用して、libmysqld ライブラリに対してリンクされた組み込みサーバーアプリケーションの場合、mysql_library_end() は、組み込みサーバーをシャットダウンしてから、クリーンアップします。

使用方法については、セクション23.7.6「C API 関数の概要」およびセクション23.7.7.40「mysql_library_init()」を参照してください。

23.7.7.40 mysql_library_init()

```
int mysql_library_init(int argc, char **argv, char **groups)
```

説明

アプリケーションが通常のクライアントプログラムであるか、組み込みサーバーを使用するかに関係なく、この関数は、ほかのすべての MySQL 関数を呼び出す前に、MySQL ライブラリを初期化するために呼び出します。アプリケーションで組み込みサーバーを使用している場合、この呼び出しによってサーバーを起動し、そのサーバーが使用するすべてのサブシステム (mysys、InnoDB など) を初期化します。

アプリケーションが MySQL ライブラリの使用を終了したら、mysql_library_end() を呼び出してクリーンアップします。セクション23.7.7.39「mysql_library_end()」を参照してください。

アプリケーションが通常のクライアントとして動作するか、または組み込みサーバーを使用するかどうかの選択は、リンク時に、最終の実行可能ファイルを生成するために、libmysqlclient ライブラリを使用するか、libmysqld ライブラリを使用するかによって異なります。追加情報については、セクション23.7.6「C API 関数の概要」を参照してください。

非マルチスレッド環境では、mysql_init() が必要に応じて自動的に mysql_library_init() を呼び出すため、この呼び出しを省略できます。ただし、mysql_library_init() はマルチスレッド環境でスレッドセーフでないため、mysql_library_init() を呼び出す mysql_init() もスレッドセーフでありません。スレッドを生成する前に mysql_library_init() を呼び出すか、さもなければ mysql_library_init() を呼び出すか、mysql_init() 経由で間接的に呼び出すかに関係なく、相互排他ロックを使用して、呼び出しを保護します。これは、ほかのすべてのクライアントライブラリの呼び出しの前に実行してください。

argc および argv 引数は main() への引数に似ており、組み込みサーバーにオプションを渡すことができます。便宜上、サーバーへのコマンド行引数がない場合、argc を 0 (ゼロ) にできます。これは、通常の (組み込みでない)

クライアントとしてのみ使用することを目的としたアプリケーションの通常のケースであり、呼び出しは一般に mysql_library_init(0, NULL, NULL) と書かれます。

```
#include <mysql.h>
#include <stdlib.h>

int main(void) {
  if (mysql_library_init(0, NULL, NULL)) {
    fprintf(stderr, "could not initialize MySQL library\n");
    exit(1);
  }

  /* Use any MySQL API functions here */

  mysql_library_end();

  return EXIT_SUCCESS;
}
```

引数が渡される (argc が 0 より大きい) と、argv の最初の要素は無視されます (それには一般にプログラム名が格納されています)。mysql_library_init() は引数のコピーを作成するため、呼び出し後 argv または groups を破棄しても安全です。

組み込みアプリケーションの場合、組み込みサーバーを起動せずに、外部サーバーに接続する場合、argc に負の値を指定する必要があります。

groups 引数は、オプションの読み取り元のオプションファイル内のグループを示す文字列の配列です。セクション4.2.6「オプションファイルの使用」を参照してください。配列の最終エントリを NULL にします。便宜上、groups 引数自体が NULL である場合、デフォルトで [server] グループと [embedded] グループが使われます。

```
#include <mysql.h>
#include <stdlib.h>

static char *server_args[] = {
  "this_program",       /* this string is not used */
  "--datadir=.",
  "--key_buffer_size=32M"
};
static char *server_groups[] = {
  "embedded",
  "server",
  "this_program_SERVER",
  (char *)NULL
};

int main(void) {
  if (mysql_library_init(sizeof(server_args) / sizeof(char *),
                         server_args, server_groups)) {
    fprintf(stderr, "could not initialize MySQL library\n");
    exit(1);
  }

  /* Use any MySQL API functions here */

  mysql_library_end();

  return EXIT_SUCCESS;
}
```

戻り値

成功の場合はゼロ。エラーが発生した場合、ゼロ以外。

23.7.7.41 mysql_list_dbs()

MYSQL_RES *mysql_list_dbs(MYSQL *mysql, const char *wild)

説明

wild パラメータによって指定された簡単な正規表現に一致するサーバー上のデータベース名から構成される結果セットを返します。wild にはワイルドカード文字 「%」 または 「_」 を含めることができ、またはすべてのデータベースに一致する NULL ポインタにできます。mysql_list_dbs() の呼び出しは、クエリー SHOW DATABASES [LIKE wild] の実行に似ています。

mysql_free_result() によって結果セットを解放する必要があります。

戻り値

成功した場合は MYSQL_RES 結果セット。エラーが発生した場合は NULL。

エラー

- CR_COMMANDS_OUT_OF_SYNC

  コマンドが正しくない順番で実行されました。

- CR_OUT_OF_MEMORY

  メモリー不足。

- CR_SERVER_GONE_ERROR

  MySQL サーバーが存在しなくなりました。

- CR_SERVER_LOST

  サーバーへの接続がクエリー中に失われました。

- CR_UNKNOWN_ERROR

  不明なエラーが発生しました。

23.7.7.42 mysql_list_fields()

MYSQL_RES *mysql_list_fields(MYSQL *mysql, const char *table, const char *wild)

説明

wild パラメータによって指定された簡単な正規表現に一致する、指定されたテーブル内のカラムに関する情報をメタデータで提供する空の結果セットを返します。wild にはワイルドカード文字 「%」 または 「_」 を含めることができ、またはすべてのフィールドに一致する NULL ポインタにできます。mysql_list_fields() の呼び出しは、クエリー SHOW COLUMNS FROM tbl_name [LIKE wild] の実行と似ています。

mysql_list_fields() の代わりに SHOW COLUMNS FROM tbl_name を使用することをお勧めします。

mysql_free_result() によって結果セットを解放する必要があります。

戻り値

成功した場合は MYSQL_RES 結果セット。エラーが発生した場合は NULL。

エラー

- CR_COMMANDS_OUT_OF_SYNC

  コマンドが正しくない順番で実行されました。

- CR_SERVER_GONE_ERROR

  MySQL サーバーが存在しなくなりました。

- CR_SERVER_LOST

  サーバーへの接続がクエリー中に失われました。

- CR_UNKNOWN_ERROR

  不明なエラーが発生しました。

例

```
int i;
MYSQL_RES *tbl_cols = mysql_list_fields(mysql, "mytbl", "f%");
```

```
unsigned int field_cnt = mysql_num_fields(tbl_cols);
printf("Number of columns: %d\n", field_cnt);

for (i=0; i < field_cnt; ++i)
{
  /* col describes i-th column of the table */
  MYSQL_FIELD *col = mysql_fetch_field_direct(tbl_cols, i);
  printf ("Column %d: %s\n", i, col->name);
}
mysql_free_result(tbl_cols);
```

23.7.7.43 mysql_list_processes()

MYSQL_RES *mysql_list_processes(MYSQL *mysql)

説明

現在のサーバースレッドについて説明する結果セットを返します。これは、mysqladmin processlist または SHOW PROCESSLIST クエリーによって報告されるものと同じ種類の情報です。

mysql_free_result() によって結果セットを解放する必要があります。

戻り値

成功した場合は MYSQL_RES 結果セット。エラーが発生した場合は NULL。

エラー

- CR_COMMANDS_OUT_OF_SYNC

  コマンドが正しくない順番で実行されました。

- CR_SERVER_GONE_ERROR

  MySQL サーバーが存在しなくなりました。

- CR_SERVER_LOST

  サーバーへの接続がクエリー中に失われました。

- CR_UNKNOWN_ERROR

  不明なエラーが発生しました。

23.7.7.44 mysql_list_tables()

MYSQL_RES *mysql_list_tables(MYSQL *mysql, const char *wild)

説明

wild パラメータによって指定された簡単な正規表現に一致する、現在のデータベース内のテーブル名から構成される結果セットを返します。wild にはワイルドカード文字 「%」 または 「_」 を含めることができ、またはすべてのテーブルに一致する NULL ポインタにできます。mysql_list_tables() の呼び出しは、クエリー SHOW TABLES [LIKE wild] の実行に似ています。

mysql_free_result() によって結果セットを解放する必要があります。

戻り値

成功した場合は MYSQL_RES 結果セット。エラーが発生した場合は NULL。

エラー

- CR_COMMANDS_OUT_OF_SYNC

  コマンドが正しくない順番で実行されました。

- CR_SERVER_GONE_ERROR

MySQL サーバーが存在しなくなりました。

- CR_SERVER_LOST

  サーバーへの接続がクエリー中に失われました。

- CR_UNKNOWN_ERROR

  不明なエラーが発生しました。

23.7.7.45 mysql_more_results()

my_bool mysql_more_results(MYSQL *mysql)

説明

この関数は、単一のステートメント文字列として指定された複数のステートメントを実行する場合、または複数の結果セットを返すことがある CALL ステートメントを実行する場合に使用します。

現在実行されているステートメントから、それ以上の結果が存在する場合、mysql_more_results() は true になり、この場合、アプリケーションは mysql_next_result() を呼び出して、結果をフェッチする必要があります。

戻り値

それ以上の結果が存在する場合は、TRUE (1)。それ以上の結果が存在しない場合は FALSE (0)。

ほとんどの場合、代わりに mysql_next_result() を呼び出して、それ以上の結果が存在しており、その場合に取得を開始するかどうかをテストできます。

セクション23.7.17「複数ステートメント実行の C API サポート」およびセクション23.7.7.46「mysql_next_result()」を参照してください。

エラー

なし。

23.7.7.46 mysql_next_result()

int mysql_next_result(MYSQL *mysql)

説明

この関数は、単一のステートメント文字列として指定された複数のステートメントを実行する場合、または複数の結果セットを返すことがある CALL ステートメントを使用して、ストアドプロシージャーを実行する場合に使用します。

mysql_next_result() は次のステートメント結果を読み取り、それ以上の結果が存在するかどうかを示すステータスを返します。mysql_next_result() がエラーを返した場合、それ以上の結果はありません。

mysql_next_result() の各呼び出しの前に、現在のステートメントが結果セットを返した (結果のステータスだけでなく) ステートメントである場合、そのステートメントに対して mysql_free_result() を呼び出す必要があります。

mysql_next_result() を呼び出したあとの接続の状態は、次のステートメントに対し、mysql_real_query() または mysql_query() を呼び出した場合のようになります。このことは、mysql_store_result()、mysql_warning_count()、mysql_affected_rows() などを呼び出すことができることを意味します。

プログラムで CALL ステートメントを使用して、ストアドプロシージャーを実行する場合、CLIENT_MULTI_RESULTS フラグが有効になっている必要があります。これは、各 CALL によって、プロシージャー内で実行されるステートメントによって返される可能性のある結果セットに加えて、呼び出しステータスを示すための結果が返されるためです。CALL は複数の結果を返すことができるため、mysql_next_result() を呼び出すループを使用して、それらを処理し、それ以上の結果があるかどうかを判断します。

CLIENT_MULTI_RESULTS は、mysql_real_connect() を呼び出すときに、CLIENT_MULTI_RESULTS フラグ自体を渡すことによって明示的に、または CLIENT_MULTI_STATEMENTS を渡すことによって暗黙的に有効にする (これによって CLIENT_MULTI_RESULTS も有効になります) ことができます。MySQL 5.6 では、CLIENT_MULTI_RESULTS はデフォルトで有効にされています。

mysql_more_results() を呼び出して、それ以上の結果があるかどうかをテストすることもできます。ただし、この関数は接続の状態を変更しないため、それが true を返した場合は、さらに mysql_next_result() を呼び出して、次の結果に進む必要があります。

mysql_next_result() の使用方法を示す例については、セクション23.7.17「複数ステートメント実行の C API サポート」を参照してください。

戻り値

戻り値	説明
0	成功し、それ以上の結果が存在します
-1	成功し、それ以上の結果が存在しません
0 より大きい	エラーが発生しました

エラー

- CR_COMMANDS_OUT_OF_SYNC

  コマンドが正しくない順番で実行されました。たとえば、前の結果セットに対して mysql_use_result() を呼び出していなかった場合。

- CR_SERVER_GONE_ERROR

  MySQL サーバーが存在しなくなりました。

- CR_SERVER_LOST

  サーバーへの接続がクエリー中に失われました。

- CR_UNKNOWN_ERROR

  不明なエラーが発生しました。

23.7.7.47 mysql_num_fields()

unsigned int mysql_num_fields(MYSQL_RES *result)

代わりに MYSQL* 引数を渡すには、unsigned int mysql_field_count(MYSQL *mysql) を使用します。

説明

結果セット内のカラムの数を返します。

結果セットまたは接続ハンドルへのいずれかのポインタからカラムの数を取得できます。接続ハンドルは、mysql_store_result() または mysql_use_result() が NULL を返した (そのため、結果セットポインタがない) 場合に使用します。この場合、mysql_field_count() を呼び出して、mysql_store_result() が空でない結果を生成しているかどうかを判断できます。これにより、クライアントプログラムは、クエリーが SELECT (または SELECT に似た) ステートメントであったかどうかを知らなくても、正しいアクションをとることができます。ここに示す例では、これを実行する方法を説明しています。

セクション23.7.15.1「mysql_query() が成功を返したあとに mysql_store_result() が NULL を返すことがあるのはなぜか」を参照してください。

戻り値

結果セット内のカラムの数を表す符号なし整数。

エラー

なし。

例

```
MYSQL_RES *result;
unsigned int num_fields;
```

```
unsigned int num_rows;

if (mysql_query(&mysql,query_string))
{
   // error
}
else // query succeeded, process any data returned by it
{
   result = mysql_store_result(&mysql);
   if (result)  // there are rows
   {
      num_fields = mysql_num_fields(result);
      // retrieve rows, then call mysql_free_result(result)
   }
   else  // mysql_store_result() returned nothing; should it have?
   {
      if (mysql_errno(&mysql))
      {
        fprintf(stderr, "Error: %s\n", mysql_error(&mysql));
      }
      else if (mysql_field_count(&mysql) == 0)
      {
         // query does not return data
         // (it was not a SELECT)
         num_rows = mysql_affected_rows(&mysql);
      }
   }
}
```

代替の方法 (クエリーが結果セットを返すべきであったことがわかっている場合) は、mysql_errno(&mysql) 呼び出しを mysql_field_count(&mysql) が 0 を返しているかどうかのチェックで置き換えることです。これは、何か異常があった場合にのみ発生します。

23.7.7.48 mysql_num_rows()

my_ulonglong mysql_num_rows(MYSQL_RES *result)

説明

結果セット内の行数を返します。

mysql_num_rows() の使用は、結果セットを返すために、mysql_store_result() を使用するか、mysql_use_result() を使用するかによって異なります。mysql_store_result() を使用すると、mysql_num_rows() はただちに呼び出されることがあります。mysql_use_result() を使用すると、mysql_num_rows() は、結果セット内のすべての行が取得されるまで、正しい値を返しません。

mysql_num_rows() は SELECT などの結果セットを返すステートメントと一緒に使用することを目的としています。INSERT、UPDATE、または DELETE などのステートメントでは、影響を受けた行の数を mysql_affected_rows() で取得できます。

戻り値

結果セット内の行数。

エラー

なし。

23.7.7.49 mysql_options()

int mysql_options(MYSQL *mysql, enum mysql_option option, const void *arg)

説明

追加の接続オプションを設定し、接続の動作に影響を与えるために使うことができます。この関数を複数回呼び出して、複数のオプションを設定できます。

mysql_init() のあとと mysql_connect() または mysql_real_connect() の前に mysql_options() を呼び出します。

option 引数は、設定するオプションです。arg 引数はそのオプションの値です。オプションが整数の場合、arg 引数として、整数の値へのポインタを指定します。

次のリストに、可能なオプション、それらの効果、および各オプションの arg の使用方法を示します。オプションのいくつかは、アプリケーションが libmysqld 組み込みサーバーライブラリに対してリンクされている場合にのみ適用され、libmysqlclient クライアントライブラリに対してリンクされているアプリケーションには使用されません。arg が使用されないことを示すオプションの説明では、その値は関係ありません。慣例として 0 を渡します。

- MYSQL_DEFAULT_AUTH (引数の型: char *)

  使用する認証プラグインの名前。

- MYSQL_ENABLE_CLEARTEXT_PLUGIN (引数の型: my_bool *)

  mysql_clear_password 平文認証プラグインを有効にします。(セクション6.3.8.7「クライアント側のクリアテキスト認証プラグイン」を参照してください。)このオプションは MySQL 5.6.7 で追加されました。

- MYSQL_INIT_COMMAND (引数の型: char *)

  MySQL サーバーへの接続時に実行する SQL ステートメント。再接続が行われる場合に自動的に再実行されます。

- MYSQL_OPT_BIND (引数: char *)

  サーバーに接続するためのネットワークインタフェース。これはクライアントホストに複数のネットワークインタフェースがある場合に使用されます。引数はホスト名または IP アドレス (文字列として指定) です。このオプションは MySQL 5.6.1 で追加されました。

- MYSQL_OPT_CAN_HANDLE_EXPIRED_PASSWORDS (引数の型: my_bool *)

  クライアントが期限切れのパスワードを処理できるかどうかを示します。詳細については、セクション6.3.6「パスワードの期限切れとサンドボックスモード」を参照してください。このオプションは MySQL 5.6.10 で追加されました。

- MYSQL_OPT_COMPRESS (引数: 使用なし)

  圧縮されたクライアント/サーバープロトコルを使用します。

- MYSQL_OPT_CONNECT_ATTR_DELETE (引数の型: char *)

  キー名を指定すると、このオプションは、接続時にサーバーに渡される現在の接続属性のセットから、キー/値ペアを削除します。引数はこのキーを指定する NULL 終端文字列へのポインタです。既存のキーとのキー名の比較は大文字と小文字が区別されます。

  mysql_options4() 関数の説明の MYSQL_OPT_CONNECT_ATTR_ADD オプションの説明に加えて、MYSQL_OPT_CONNECT_ATTR_RESET オプションの説明も参照してください。この関数の説明には、使用例も含まれています。これらのオプションは MySQL 5.6.6 で追加されました。

  接続属性は、session_connect_attrs および session_account_connect_attrs パフォーマンススキーマテーブル経由で公開されます。セクション22.9.8「パフォーマンススキーマ接続属性テーブル」を参照してください。

- MYSQL_OPT_CONNECT_ATTR_RESET (引数は使用されません)

  このオプションは、接続時にサーバーに渡される現在の接続属性のセットをリセット (クリア) します。

  mysql_options4() 関数の説明の MYSQL_OPT_CONNECT_ATTR_ADD オプションの説明に加えて、MYSQL_OPT_CONNECT_ATTR_DELETE オプションの説明も参照してください。この関数の説明には、使用例も含まれています。これらのオプションは MySQL 5.6.6 で追加されました。

  接続属性は、session_connect_attrs および session_account_connect_attrs パフォーマンススキーマテーブル経由で公開されます。セクション22.9.8「パフォーマンススキーマ接続属性テーブル」を参照してください。

- MYSQL_OPT_CONNECT_TIMEOUT (引数の型: unsigned int *)

  秒単位の接続タイムアウト。

- MYSQL_OPT_GUESS_CONNECTION (引数は使用されません)

  libmysqld 組み込みサーバーライブラリに対してリンクされたアプリケーションの場合、これにより、ライブラリは組み込みサーバーを使用するか、リモートサーバーを使用するかを推測できます。「推測」とは、ホスト名が設定されていて、localhost でない場合、リモートサーバーを使用することを

意味します。この動作はデフォルトです。MYSQL_OPT_USE_EMBEDDED_CONNECTION および MYSQL_OPT_USE_REMOTE_CONNECTION を使用して、それをオーバーライドできます。このオプションは、libmysqlclient クライアントライブラリに対してリンクされたアプリケーションに対しては無視されます。

- MYSQL_OPT_LOCAL_INFILE (引数の型: unsigned int へのオプションのポインタ)

  ポインタが指定されていないか、ポインタがゼロ以外の値を持つ unsigned int を指している場合、LOAD LOCAL INFILE ステートメントが有効にされます。

- MYSQL_OPT_NAMED_PIPE (引数: 使用されません)

  MySQL サーバーが名前付きパイプ接続を許可する場合に、Windows でサーバーに接続するために、名前付きパイプを使用します。

- MYSQL_OPT_PROTOCOL (引数の型: unsigned int *)

  使用するプロトコルの種類。mysql.h に定義されている mysql_protocol_type の列挙値のいずれかを指定します。

- MYSQL_OPT_READ_TIMEOUT (引数の型: unsigned int *)

  サーバーからの読み取りの試みごとの秒単位でのタイムアウト。必要に応じて再試行があるため、実際の合計タイムアウト値は、オプションの値の 3 倍です。10 分の TCP/IP Close_Wait_Timeout 値より早く、失われた接続を検出できるように値を設定できます。

  このタイムアウトの実装は、すべてのプラットフォームで使用できるとは限らないメカニズムを使用しています。そのようなプラットフォームでは、読み取り呼び出しを発行するクライアントが、特定の環境において、タイムアウトせずに待機することがあります。たとえば、サーバーが「ディスク領域不足」状況がクリアされるまで待機しているために応答しない場合、クライアントはタイムアウトしない可能性があります。

- MYSQL_OPT_RECONNECT (引数の型: my_bool *)

  接続が失われていることが検出された場合に、サーバーへの自動再接続を有効または無効にします。再接続はデフォルトでオフにされますが、このオプションは明示的に再接続動作を設定する方法を提供します。

- MYSQL_OPT_SSL_CA (引数の型: char *)

  信頼された SSL CA のリストを含むファイルへのパス。このオプションは MySQL 5.6.3 で追加されました。

- MYSQL_OPT_SSL_CAPATH (引数の型: char *)

  PEM 形式の信頼された SSL CA 証明書を格納するディレクトリのパス。このオプションは MySQL 5.6.3 で追加されました。

- MYSQL_OPT_SSL_CERT (引数の型: char *)

  セキュアな接続を確立するために使用する SSL 証明書ファイルの名前。このオプションは MySQL 5.6.3 で追加されました。

- MYSQL_OPT_SSL_CIPHER (引数の型: char *)

  SSL 暗号化に使用する許可されている暗号のリスト。このオプションは MySQL 5.6.3 で追加されました。

- MYSQL_OPT_SSL_CRL (引数の型: char *)

  PEM 形式での証明書失効リストを含むファイルへのパス。このオプションは MySQL 5.6.3 で追加されました。

- MYSQL_OPT_SSL_CRLPATH (引数の型: char *)

  PEM 形式での証明書失効リストを含むファイルを格納するディレクトリへのパス。このオプションは MySQL 5.6.3 で追加されました。

- MYSQL_OPT_SSL_KEY (引数の型: char *)

  セキュアな接続を確立するために使用する SSL 鍵ファイルの名前。このオプションは MySQL 5.6.3 で追加されました。

- MYSQL_OPT_SSL_VERIFY_SERVER_CERT (引数の型: my_bool *)

サーバーの証明書内のサーバーの一般名値の、サーバーへの接続時に使用されるホスト名に対する検証を有効または無効にします。不一致があった場合、接続は拒否されます。この機能は、中間者攻撃を防ぐために使用できます。検証はデフォルトで無効です。

- MYSQL_OPT_USE_EMBEDDED_CONNECTION (引数: 使用されません)

  libmysqld 組み込みサーバーライブラリに対してリンクされたアプリケーションの場合、これにより、接続に組み込みサーバーの使用が強制されます。このオプションは、libmysqlclient クライアントライブラリに対してリンクされたアプリケーションに対しては無視されます。

- MYSQL_OPT_USE_REMOTE_CONNECTION (引数: 使用されません)

  libmysqld 組み込みサーバーライブラリに対してリンクされたアプリケーションの場合、これにより、接続にリモートサーバーの使用が強制されます。このオプションは、libmysqlclient クライアントライブラリに対してリンクされたアプリケーションに対しては無視されます。

- MYSQL_OPT_USE_RESULT (引数: 使用されません)

  このオプションは使用されません。

- MYSQL_OPT_WRITE_TIMEOUT (引数の型: unsigned int *)

  サーバーへの書き込みの試みごとの秒単位でのタイムアウト。必要に応じて再試行があるため、実際の合計タイムアウト値は、オプションの値の 2 倍です。

- MYSQL_PLUGIN_DIR (引数の型: char *)

  クライアントプラグインを検索するディレクトリ。

- MYSQL_READ_DEFAULT_FILE (引数の型: char *)

  my.cnf の代わりに、指定したオプションファイルからオプションを読み取ります。

- MYSQL_READ_DEFAULT_GROUP (引数の型: char *)

  my.cnf の指定したグループまたは MYSQL_READ_DEFAULT_FILE によって指定されたファイルから、オプションを読み取ります。

- MYSQL_REPORT_DATA_TRUNCATION (引数の型: my_bool *)

  MYSQL_BIND 構造の error メンバーを使用して、プリペアドステートメントのデータ切り捨てエラーのレポートを有効または無効にします。(デフォルト: 有効。)

- MYSQL_SECURE_AUTH (引数の型: my_bool *)

  MySQL 4.1.1 以降で使用されているパスワードハッシュをサポートしていないサーバーに接続するかどうか。MySQL 5.6.7 以降、このオプションはデフォルトで有効にされています。

- MYSQL_SERVER_PUBLIC_KEY (引数の型: char *)

  サーバー RSA 公開鍵を含むファイルへのパス名。ファイルは PEM 形式である必要があります。公開鍵は、sha256_password プラグインによって認証するアカウントを使用して、サーバーへの接続を作成するためのクライアントパスワードの RSA 暗号化に使用されます。このオプションは、そのプラグインによって認証しないクライアントアカウントに対しては無視されます。さらに、クライアントが SSL 接続を使用して、サーバーに接続する場合のように、パスワード暗号化が必要でない場合も無視されます。

  サーバーは必要に応じて公開鍵をクライアントに送信するため、RSA パスワードの暗号化が行われるように、このオプションを使う必要はありません。そうすることで、サーバーは鍵を送信する必要がないため、効率が向上します。

  RSA 公開鍵の取得方法を含め、sha256_password プラグインの使用に関する追加の説明については、セクション6.3.8.4「SHA-256 認証プラグイン」を参照してください。

  このオプションは MySQL 5.6.6 で追加されました。

- MYSQL_SET_CHARSET_DIR (引数の型: char *)

  文字セット定義ファイルを格納するディレクトリのパス名。

- MYSQL_SET_CHARSET_NAME (引数の型: char *)

  デフォルトの文字セットとして使用する文字セットの名前。引数を MYSQL_AUTODETECT_CHARSET_NAME にして、オペレーティングシステム設定に基づいて、文字セットを自動検出させることができます (セクション 10.1.4「接続文字セットおよび照合順序」を参照してください)。

- MYSQL_SET_CLIENT_IP (引数の型: char *)

  libmysqld 組み込みサーバーライブラリに対してリンクされたアプリケーションの場合 (libmysqld が認証サポート付きでコンパイルされている場合)、これは、ユーザーが認証のために指定された IP アドレス (文字列として指定) から接続しているとみなされることを意味します。このオプションは、libmysqlclient クライアントライブラリに対してリンクされたアプリケーションに対しては無視されます。

- MYSQL_SHARED_MEMORY_BASE_NAME (引数の型: char *)

  サーバーが共有メモリー接続をサポートしている場合、Windows 上のサーバーとの通信のための共有メモリーオブジェクトの名前。接続先の mysqld サーバーに使用されている --shared-memory-base-name オプションと同じ値を指定します。

MYSQL_READ_DEFAULT_FILE または MYSQL_READ_DEFAULT_GROUP を使用する場合、client グループが常に読み取られます。

オプションファイル内の指定されたグループには、次のオプションが含まれることがあります。

オプション	説明
character-sets-dir=path	文字セットがインストールされているディレクトリ。
compress	圧縮されたクライアント/サーバープロトコルを使用します。
connect-timeout=seconds	秒単位の接続タイムアウト。Linux では、このタイムアウトはサーバーからの最初の回答を待機する場合にも使います。
database=db_name	接続コマンドにデータベースが指定されていなかった場合、このデータベースに接続します。
debug	デバッグオプション。
default-character-set=charset_name	使用するデフォルトの文字セット。
disable-local-infile	LOAD DATA LOCAL の使用を無効にします。
enable-cleartext-plugin	mysql_clear_password 平文認証プラグインを有効にします。MySQL 5.6.7 で追加されました。
host=host_name	デフォルトのホスト名。
init-command=stmt	MySQL サーバーへの接続時に実行するステートメント。再接続が行われる場合に自動的に再実行されます。
interactive-timeout=seconds	mysql_real_connect() に CLIENT_INTERACTIVE を指定する場合と同じ。セクション23.7.7.53「mysql_real_connect()」を参照してください。
local-infile[={0|1}]	引数がないか、ゼロ以外の引数の場合、LOAD DATA LOCAL の使用を有効にします。それ以外の場合は無効にします。
max_allowed_packet=bytes	クライアントがサーバーから読み取ることができるパケットの最大サイズ。
multi-queries、multi-results	複数ステートメントの実行またはストアドプロシージャーからの複数の結果セットを可能にします。
multi-statements	クライアントが複数のステートメントを単一の文字列 (「;」 で区切られた) で送信できるようにします。
password=password	デフォルトパスワード。
pipe	Windows で MySQL サーバーに接続するために、名前付きパイプを使用します。
port=port_num	デフォルトのポート番号。
protocol={TCP|SOCKET|PIPE|MEMORY}	サーバーに接続する場合に使用するプロトコル。
return-found-rows	UPDATE を使う場合、更新された行の代わりに見つかった行を返すように、mysql_info() に伝えます。

オプション	説明
shared-memory-base-name=name	サーバーに接続するために使用する共有メモリー名。
socket=path	デフォルトのソケットファイル。
ssl-ca=file_name	認証局ファイル。
ssl-capath=path	認証局ディレクトリ。
ssl-cert=file_name	証明書ファイル。
ssl-cipher=cipher_list	許容される SSL 暗号。
ssl-key=file_name	鍵ファイル。
timeout=seconds	connect-timeout に似ています。
user	デフォルトユーザー。

timeout は connect-timeout によって置き換えられましたが、timeout は下位互換性のため、MySQL 5.6 で引き続きサポートされています。

MySQL プログラムによって使用されるオプションファイルの詳細については、セクション4.2.6「オプションファイルの使用」を参照してください。

戻り値

成功の場合はゼロ。不明なオプションを指定した場合はゼロ以外。

例

次の mysql_options() 呼び出しは、クライアント/サーバープロトコルでの圧縮の使用を要求し、オプションファイルの [odbc] グループからオプションを読み取らせ、トランザクション自動コミットモードを無効にします。

```
MYSQL mysql;

mysql_init(&mysql);
mysql_options(&mysql,MYSQL_OPT_COMPRESS,0);
mysql_options(&mysql,MYSQL_READ_DEFAULT_GROUP,"odbc");
mysql_options(&mysql,MYSQL_INIT_COMMAND,"SET autocommit=0");
if (!mysql_real_connect(&mysql,"host","user","passwd","database",0,NULL,0))
{
   fprintf(stderr, "Failed to connect to database: Error: %s\n",
         mysql_error(&mysql));
}
```

このコードは、クライアントが圧縮されたクライアント/サーバープロトコルを使用し、my.cnf ファイルの odbc セクションから追加のオプションを読み取ります。

23.7.7.50 mysql_options4()

int mysql_options4(MYSQL *mysql, enum mysql_option option, const void *arg1, const void *arg2)

説明

mysql_options4() は mysql_options() と似ていますが、追加の 4 番目の引数があり、2 番目の引数に指定されたオプションに 2 つの値を渡せるようにしています。この関数は、MySQL 5.6.6 で追加されました。

次のリストに許可されるオプション、それらの効果、および arg1 と arg2 の使用方法について説明します。

- MYSQL_OPT_CONNECT_ATTR_ADD (引数の型: char *, char *)

  このオプションは、接続時にサーバーに渡される現在の接続属性のセットに、鍵/値ペアを追加します。どちらの引数も NULL 終端文字列へのポインタです。最初と 2 番目の文字列は、鍵と値をそれぞれ示します。現在の接続属性のセットに、鍵がすでに存在する場合、エラーが発生します。既存のキーとのキー名の比較は大文字と小文字が区別されます。

  下線 (_) で始まる鍵名は、内部使用のために予約されているため、アプリケーションプログラムで使用しないでください。

  mysql_options() 関数の説明の MYSQL_OPT_CONNECT_ATTR_RESET
  MYSQL_OPT_CONNECT_ATTR_DELETE オプションの説明も参照してください。

接続属性は、session_connect_attrs および session_account_connect_attrs パフォーマンススキーマテーブル経由で公開されます。セクション22.9.8「パフォーマンススキーマ接続属性テーブル」を参照してください。

戻り値

成功の場合はゼロ。不明なオプションを指定した場合はゼロ以外。

例

この例は、接続属性を指定する呼び出しを示しています。

```
MYSQL mysql;

mysql_init(&mysql);
mysql_options(&mysql,MYSQL_OPT_CONNECT_ATTR_RESET, 0);
mysql_options4(&mysql,MYSQL_OPT_CONNECT_ATTR_ADD, "key1", "value1");
mysql_options4(&mysql,MYSQL_OPT_CONNECT_ATTR_ADD, "key2", "value2");
mysql_options4(&mysql,MYSQL_OPT_CONNECT_ATTR_ADD, "key3", "value3");
mysql_options(&mysql,MYSQL_OPT_CONNECT_ATTR_DELETE, "key1");
if (!mysql_real_connect(&mysql,"host","user","passwd","database",0,NULL,0))
{
   fprintf(stderr, "Failed to connect to database: Error: %s\n",
      mysql_error(&mysql));
}
mysql_options(&mysql,MYSQL_OPT_CONNECT_ATTR_RESET, 0);
```

23.7.7.51 mysql_ping()

int mysql_ping(MYSQL *mysql)

説明

サーバーへの接続が機能しているかどうかをチェックします。接続が切断され、自動再接続が有効にされている場合、再接続の試みが行われます。接続が切断し、自動再接続が無効にされている場合、mysql_ping() はエラーを返します。

自動再接続はデフォルトで無効にされています。それを有効にするには、MYSQL_OPT_RECONNECT オプションを使用して mysql_options() を呼び出します。詳細については、セクション23.7.7.49「mysql_options()」を参照してください。

mysql_ping() は、長い間アイドルのままでいるクライアントで、サーバーが接続を閉じているかどうかをチェックし、必要に応じて再接続するために使用できます。

mysql_ping() が再接続させる場合、その明示的な兆候はありません。再接続が行われるかどうかを判断するには、mysql_ping() を呼び出す前に mysql_thread_id() を呼び出して、元の接続識別子を取得してから、再度 mysql_thread_id() を呼び出して、識別子が変更されているかどうかを確認します。

再接続が行われた場合、接続の一部の特性がリセットされます。これらの特性の詳細については、セクション23.7.16「自動再接続動作の制御」を参照してください。

戻り値

サーバーへの接続がアクティブである場合はゼロ。エラーが発生した場合、ゼロ以外。ゼロ以外の戻り値は、MySQL サーバー自体が停止しているかどうかを示していません。ネットワークの問題などのその他の理由で接続が切断している可能性があります。

エラー

- CR_COMMANDS_OUT_OF_SYNC

  コマンドが正しくない順番で実行されました。

- CR_SERVER_GONE_ERROR

  MySQL サーバーが存在しなくなりました。

- CR_UNKNOWN_ERROR

不明なエラーが発生しました。

23.7.7.52 mysql_query()

```
int mysql_query(MYSQL *mysql, const char *stmt_str)
```

説明

NULL 終端文字列 stmt_str によって指示される SQL ステートメントを実行します。通常、文字列は終端のセミコロン (「;」) または \g を含まない単一の SQL ステートメントから構成されている必要があります。複数ステートメントの実行が有効にされている場合、文字列には、セミコロンで区切られた複数のステートメントを含めることができます。セクション23.7.17「複数ステートメント実行の C API サポート」を参照してください。

バイナリデータを含むステートメントには、mysql_query() を使うことはできません。代わりに、mysql_real_query() を使う必要があります。(バイナリデータには、mysql_query() がステートメント文字列の終端と解釈する 「\0」 文字が含まれることがあります。)

ステートメントが結果セットを返すかどうかを知りたい場合は、mysql_field_count() を使用してこれをチェックできます。セクション23.7.7.22「mysql_field_count()」を参照してください。

戻り値

成功の場合はゼロ。エラーが発生した場合、ゼロ以外。

エラー

- CR_COMMANDS_OUT_OF_SYNC

  コマンドが正しくない順番で実行されました。

- CR_SERVER_GONE_ERROR

  MySQL サーバーが存在しなくなりました。

- CR_SERVER_LOST

  サーバーへの接続がクエリー中に失われました。

- CR_UNKNOWN_ERROR

  不明なエラーが発生しました。

23.7.7.53 mysql_real_connect()

```
MYSQL *mysql_real_connect(MYSQL *mysql, const char *host, const char *user, const char *passwd, const char *db, unsigned int port, const char *unix_socket, unsigned long client_flag)
```

説明

mysql_real_connect() はホストで実行している MySQL データベースエンジンへの接続の確立を試みます。mysql_real_connect() は、有効な MYSQL 接続ハンドル構造を必要とするほかの任意の API 関数を実行する前に、正常に完了している必要があります。

パラメータは次のように指定します。

- 最初のパラメータで、既存の MYSQL 構造のアドレスを指定します。mysql_real_connect() を呼び出す前に、mysql_init() を呼び出して、MYSQL 構造を初期化します。mysql_options() の呼び出しによって、多くの接続オプションを変更できます。セクション23.7.7.49「mysql_options()」を参照してください。

- host の値は、ホスト名または IP アドレスを指定できます。host が NULL か、文字列 "localhost" である場合、ローカルホストへの接続とみなされます。Windows では、サーバーで共有メモリー接続が有効にされている場合、クライアントは共有メモリー接続を使用して接続します。そうでない場合は、TCP/IP が使用されます。Unix では、クライアントは Unix ソケットファイルを使用して接続します。ローカル接続では、mysql_options() への MYSQL_OPT_PROTOCOL または MYSQL_OPT_NAMED_PIPE オプションで使用する接続の種類にも影響を与えることができます。接続の種類がサーバーでサポートされている必要があります。Windows での "." の host 値では、サーバーで名前付きパイプ接続が有効にされている場合、クライアント

は名前付きパイプを使用して接続します。名前付きパイプ接続が有効にされていない場合は、エラーが発生します。

- user パラメータにはユーザーの MySQL ログイン ID が含まれます。user が NULL かまたは空の文字列 "" である場合、現在のユーザーが想定されます。Unix では、これは現在のログイン名です。Windows ODBC では、現在のユーザー名を明示的に指定する必要があります。第23章「Connector および API」の Connector/ODBC のセクションを参照してください。
- passwdパラメータには、user のパスワードが含まれます。passwd が NULL である場合、ブランク (空) のパスワードフィールドを持つユーザーの user テーブル内のエントリのみが一致をチェックされます。これにより、データベース管理者は、ユーザーがパスワードを指定しているかどうかに応じて、ユーザーが異なる権限を取得するような MySQL 権限システムをセットアップできます。

  > **注記**
  >
  > mysql_real_connect() を呼び出す前に、パスワードを暗号化しようとしないでください。パスワードの暗号化はクライアント API によって自動的に処理されます。

- user および passwd パラメータは、MYSQL オブジェクトに構成されている文字セットを使用します。デフォルトでこれは latin1 ですが、接続の前に mysql_options(mysql, MYSQL_SET_CHARSET_NAME, "charset_name") を呼び出すことによって変更できます。
- db はデータベース名です。db が NULL でない場合、接続はデフォルトのデータベースをこの値に設定します。
- port が 0 でない場合、値は TCP/IP 接続のポート番号として使用されます。host パラメータによって、接続の種類が決まります。
- unix_socket が NULL でない場合、文字列は使用するソケットまたは名前付きパイプを指定します。host パラメータによって、接続の種類が決まります。
- client_flag の値は通常 0 ですが、特定の機能を有効にするために、次のフラグの組み合わせに設定できます。

フラグ名	フラグの説明
CAN_HANDLE_EXPIRED_PASSWORDS	クライアントは期限切れのパスワードを処理できます。詳細については、セクション6.3.6「パスワードの期限切れとサンドボックスモード」を参照してください。このオプションは MySQL 5.6.10 で追加されました。
CLIENT_COMPRESS	圧縮プロトコルを使用します。
CLIENT_FOUND_ROWS	変更された行の数ではなく、見つかった (一致した) 行の数を返します。
CLIENT_IGNORE_SIGPIPE	クライアントライブラリの SIGPIPE シグナルハンドラのインストールを妨げます。これは、アプリケーションがすでにインストールしているハンドラとの競合を避けるために使用できます。
CLIENT_IGNORE_SPACE	関数名のあとのスペースを許可します。すべての関数名を予約語にします。
CLIENT_INTERACTIVE	接続を閉じるまで、interactive_timeout 秒 (wait_timeout 秒ではなく) の非アクティブを許可します。クライアントのセッション wait_timeout 変数は、セッション interactive_timeout 変数の値に設定されます。
CLIENT_LOCAL_FILES	LOAD DATA LOCAL 処理を有効にします。
CLIENT_MULTI_RESULTS	サーバーに、クライアントは複数ステートメントの実行またはストアドプロシージャーからの複数の結果セットを処理できることを伝えます。CLIENT_MULTI_STATEMENTS が有効にされている場合、このフラグは自動的に有効にされます。このフラグの詳細については、この表のあとにある注記を参照してください。
CLIENT_MULTI_STATEMENTS	サーバーに、クライアントは単一の文字列 (「;」 で区切られた) で複数のステートメントを送信する可能性があることを伝えます。このフラグが設定されていない場合、複数ステートメントの実行は無効になります。このフラグの詳細については、この表のあとにある注記を参照してください。
CLIENT_NO_SCHEMA	db_name.tbl_name.col_name 構文を許可しません。これは ODBC 用です。ユーザーがその構文を使用した場合、パーサーにエラーを生成さ

フラグ名	フラグの説明
	せます。これは特定の ODBC プログラムでバグのトラップに役立ちます。
CLIENT_ODBC	使用されません。
CLIENT_SSL	SSL (暗号化プロトコル) を使います。アプリケーションプログラム内にこのオプションを設定しないでください。それはクライアントライブラリに内部で設定されます。代わりに、mysql_real_connect() を呼び出す前に、mysql_ssl_set() を使用します。
CLIENT_REMEMBER_OPTIONS	mysql_options() への呼び出しに指定したオプションを覚えておきます。このオプションを使用しないと、mysql_real_connect() が失敗した場合、再度接続を試みる前に、mysql_options() 呼び出しを繰り返す必要があります。このオプションを使用すると、mysql_options() 呼び出しを繰り返す必要がありません。

プログラムで CALL ステートメントを使用して、ストアドプロシージャーを実行する場合、CLIENT_MULTI_RESULTS フラグが有効になっている必要があります。これは、各 CALL によって、プロシージャー内で実行されるステートメントによって返される可能性のある結果セットに加えて、呼び出しステータスを示すための結果が返されるためです。CALL は複数の結果を返すことができるため、mysql_next_result() を呼び出すループを使用して、それらを処理し、それ以上の結果があるかどうかを判断します。

CLIENT_MULTI_RESULTS は、mysql_real_connect() を呼び出すときに、CLIENT_MULTI_RESULTS フラグ自体を渡すことによって明示的に、または CLIENT_MULTI_STATEMENTS を渡すことによって暗黙的に有効にする (これによって CLIENT_MULTI_RESULTS も有効になります) ことができます。MySQL 5.6 では、CLIENT_MULTI_RESULTS はデフォルトで有効にされています。

CLIENT_MULTI_STATEMENTS または CLIENT_MULTI_RESULTS を有効にした場合、それ以上の結果があるかどうかを判断するために mysql_next_result() を呼び出すループを使用して、mysql_query() または mysql_real_query() へのすべての呼び出しの結果を処理します。例については、セクション23.7.17「複数ステートメント実行の C API サポート」を参照してください。

一部のパラメータでは、mysql_real_connect() 呼び出しでの明示的な値からでなく、オプションファイルから値を取得させることができます。これを行うには、mysql_real_connect() を呼び出す前に、MYSQL_READ_DEFAULT_FILE または MYSQL_READ_DEFAULT_GROUP オプションを使用して、mysql_options() を呼び出します。その後、オプションファイルから読み取られるパラメータごとに、mysql_real_connect() の呼び出しで、「no-value」 値を指定します。

- host に対して、NULL または空の文字列 ("") の値を指定します。
- user に対して、NULL または空の文字列の値を指定します。
- passwd に対して、NULL の値を指定します。(パスワードの場合、mysql_real_connect() 呼び出しの空の文字列の値は、オプションファイルでオーバーライドできません。空の文字列は、MySQL アカウントが空のパスワードを持つ必要があることを明示的に示しているためです。)
- db に対して、NULL または空の文字列の値を指定します。
- port に対して、0 の値を指定します。
- unix_socket に対して、NULL の値を指定します。

パラメータのオプションファイルに値が見つからない場合、このセクションの前のほうの説明で示したように、そのデフォルト値が使われます。

戻り値

接続が成功した場合は MYSQL* 接続ハンドル、接続が成功しなかった場合は NULL。接続が成功した場合、戻り値は最初のパラメータの値と同じになります。

エラー

- CR_CONN_HOST_ERROR

  MySQL サーバーへの接続に失敗しました。

- CR_CONNECTION_ERROR

ローカル MySQL サーバーへの接続に失敗しました。

- CR_IPSOCK_ERROR

  IP ソケットの作成に失敗しました。

- CR_OUT_OF_MEMORY

  メモリー不足。

- CR_SOCKET_CREATE_ERROR

  Unix ソケットの作成に失敗しました。

- CR_UNKNOWN_HOST

  ホスト名の IP アドレスの検出に失敗しました。

- CR_VERSION_ERROR

  異なるプロトコルバージョンを使用するクライアントライブラリによるサーバーへの接続の試みから発生したプロトコルの不一致。

- CR_NAMEDPIPEOPEN_ERROR

  Windows で名前付きパイプの作成に失敗しました。

- CR_NAMEDPIPEWAIT_ERROR

  Windows で名前付きパイプの待機に失敗しました。

- CR_NAMEDPIPESETSTATE_ERROR

  Windows でパイプハンドラの取得に失敗しました。

- CR_SERVER_LOST

  connect_timeout > 0 で、サーバーに接続するために connect_timeout 秒より長くかかった場合、または init-command の実行中にサーバーが停止した場合。

- CR_ALREADY_CONNECTED

  MYSQL 接続ハンドルはすでに接続されています。

例

```
MYSQL mysql;

mysql_init(&mysql);
mysql_options(&mysql,MYSQL_READ_DEFAULT_GROUP,"your_prog_name");
if (!mysql_real_connect(&mysql,"host","user","passwd","database",0,NULL,0))
{
   fprintf(stderr, "Failed to connect to database: Error: %s\n",
         mysql_error(&mysql));
}
```

mysql_options() を使用することで、MySQL ライブラリは、だれかが MySQL を標準でない方法でセットアップした場合でも、プログラムが動作するようにする my.cnf ファイル内の [client] および [your_prog_name] セクションを読み取ります。

接続時に、mysql_real_connect() は、reconnect フラグ (MYSQL 構造の一部) を、5.0.3 より古い API のバージョンでは 1 の値、以降のバージョンでは 0 の値に設定してください。このフラグの 1 の値は、切断された接続のためにステートメントを実行できない場合、諦める前にサーバーに再接続しようとすることを示します。mysql_options() に MYSQL_OPT_RECONNECT オプションを使用して、再接続動作を制御できます。

23.7.7.54 mysql_real_escape_string()

unsigned long mysql_real_escape_string(MYSQL *mysql, char *to, const char *from, unsigned long length)

mysql は有効な開いている接続でなければなりません。エスケープはサーバーで使用されている文字セットによって異なるため、これが必要です。

説明

この関数は、SQL ステートメントで使用できる正当な SQL 文字列を作成するために使われます。セクション 9.1.1「文字列リテラル」を参照してください。

from 内の文字列は、接続の現在の文字セットを考慮して、エスケープされた SQL 文字列にエンコードされます。結果は to に置かれ、終端の NULL バイトが付加されます。エンコードされる文字は「\」、「'」、「"」、NUL (ASCII 0)、「\n」、「\r」、および Control+Z です。厳密に言うと、MySQL では、クエリー内の文字列に引用符を付けるために使われるバックスラッシュと引用文字がエスケープされることのみを必要とします。mysql_real_escape_string() は、ほかの文字をログファイル内で読みやすくするために引用符で囲みます。比較については、セクション9.1.1「文字列リテラル」およびセクション12.5「文字列関数」のリテラル文字列と QUOTE() SQL 関数の引用符使用のルールを参照してください。

from によって指示される文字列は、length バイトの長さである必要があります。to バッファーを少なくとも length*2+1 バイト長になるように割り当てる必要があります。(最悪の場合、各文字を 2 バイトを使用してエンコードする必要がある可能性があるため、終端の NULL バイト分の空きが必要です。)mysql_real_escape_string() が戻ったときの to の内容は NULL 終端文字列です。戻り値は、終端の NULL 文字を含まない、エンコードされた文字列の長さです。

接続の文字セットを変更する必要がある場合、SET NAMES (または SET CHARACTER SET) ステートメントを実行するのではなく、mysql_set_character_set() 関数を使用します。mysql_set_character_set() は、SET NAMES と同じように機能しますが、SET NAMES では影響を与えない mysql_real_escape_string() で使用される文字セットにも影響を与えます。

例

```
char query[1000],*end;

end = strmov(query,"INSERT INTO test_table values(");
*end++ = '\'';
end += mysql_real_escape_string(&mysql, end,"What is this",12);
*end++ = '\'';
*end++ = ',';
*end++ = '\'';
end += mysql_real_escape_string(&mysql, end,"binary data: \0\r\n",16);
*end++ = '\'';
*end++ = ')';

if (mysql_real_query(&mysql,query,(unsigned int) (end - query)))
{
  fprintf(stderr, "Failed to insert row, Error: %s\n",
          mysql_error(&mysql));
}
```

例で使用している strmov() 関数は、libmysqlclient ライブラリに含まれ、strcpy() のように機能しますが、最初のパラメータの終端の NULL へのポインタを返します。

戻り値

to に置かれた、終端の NULL 文字を含まない値の長さ。

エラー

なし。

23.7.7.55 mysql_real_query()

```
int mysql_real_query(MYSQL *mysql, const char *stmt_str, unsigned long length)
```

説明

length バイト長の文字列 stmt_str によって指示された SQL ステートメントを実行します。通常、文字列は終端のセミコロン (「;」) または \g を含まない単一の SQL ステートメントから構成されている必要があります。複数ステートメントの実行が有効にされている場合、文字列には、セミコロンで区切られた複数のステートメントを含めることができます。セクション23.7.17「複数ステートメント実行の C API サポート」を参照してください。

バイナリデータを含むステートメントには、mysql_query() を使うことはできません。代わりに、mysql_real_query() を使う必要があります。(バイナリデータには、mysql_query() がステートメント文字列の

終端と解釈する 「\0」 文字が含まれることがあります。)さらに、mysql_real_query() はステートメント文字列に対して strlen() を呼び出さないため、mysql_query() より高速です。

ステートメントが結果セットを返すかどうかを知りたい場合は、mysql_field_count() を使用してこれをチェックできます。セクション23.7.7.22「mysql_field_count()」を参照してください。

戻り値

成功の場合はゼロ。エラーが発生した場合、ゼロ以外。

エラー

- CR_COMMANDS_OUT_OF_SYNC

  コマンドが正しくない順番で実行されました。

- CR_SERVER_GONE_ERROR

  MySQL サーバーが存在しなくなりました。

- CR_SERVER_LOST

  サーバーへの接続がクエリー中に失われました。

- CR_UNKNOWN_ERROR

  不明なエラーが発生しました。

23.7.7.56 mysql_refresh()

int mysql_refresh(MYSQL *mysql, unsigned int options)

説明

この関数はテーブルまたはキャッシュをフラッシュするか、レプリケーションサーバー情報をリセットします。接続されたユーザーは RELOAD 権限を持つ必要があります。

options 引数は次の値の任意の組み合わせから構成されるビットマスクです。複数の値をまとめて OR をとり、単一の呼び出しで複数の操作を実行できます。

- REFRESH_GRANT

  FLUSH PRIVILEGES などの付与テーブルをリフレッシュします。

- REFRESH_LOG

  FLUSH LOGS のように、ログをフラッシュします。

- REFRESH_TABLES

  FLUSH TABLES のように、テーブルキャッシュをフラッシュします。

- REFRESH_HOSTS

  FLUSH HOSTS のように、ホストキャッシュをフラッシュします。

- REFRESH_STATUS

  FLUSH STATUS のように、ステータス変数をリセットします。

- REFRESH_THREADS

  スレッドキャッシュをフラッシュします。

- REFRESH_SLAVE

  スレーブレプリケーションサーバーで、RESET SLAVE のように、マスターサーバー情報をリセットし、スレーブを再起動します。

- REFRESH_MASTER

マスターレプリケーションサーバーで、RESET MASTER のように、バイナリログインデックスに示されているバイナリログファイルを削除し、インデックスファイルを切り捨てます。

戻り値

成功の場合はゼロ。エラーが発生した場合、ゼロ以外。

エラー

- CR_COMMANDS_OUT_OF_SYNC

  コマンドが正しくない順番で実行されました。

- CR_SERVER_GONE_ERROR

  MySQL サーバーが存在しなくなりました。

- CR_SERVER_LOST

  サーバーへの接続がクエリー中に失われました。

- CR_UNKNOWN_ERROR

  不明なエラーが発生しました。

23.7.7.57 mysql_reload()

int mysql_reload(MYSQL *mysql)

説明

MySQL サーバーに付与テーブルをリロードするように求めます。接続されたユーザーは RELOAD 権限を持つ必要があります。

この関数は非推奨です。代わりに、mysql_query() を使用して、SQL FLUSH PRIVILEGES ステートメントを発行することをお勧めします。

戻り値

成功の場合はゼロ。エラーが発生した場合、ゼロ以外。

エラー

- CR_COMMANDS_OUT_OF_SYNC

  コマンドが正しくない順番で実行されました。

- CR_SERVER_GONE_ERROR

  MySQL サーバーが存在しなくなりました。

- CR_SERVER_LOST

  サーバーへの接続がクエリー中に失われました。

- CR_UNKNOWN_ERROR

  不明なエラーが発生しました。

23.7.7.58 mysql_rollback()

my_bool mysql_rollback(MYSQL *mysql)

説明

現在のトランザクションをロールバックします。

この関数のアクションは、completion_type システム変数の値に影響を受けます。特に、completion_type の値が RELEASE (または 2) の場合、トランザクションの終了後、サーバーは解放を実行し、クライアント接続を閉じます。クライアントプログラムから mysql_close() を呼び出して、クライアント側から接続をクローズします。

戻り値

成功の場合はゼロ。エラーが発生した場合、ゼロ以外。

エラー

なし。

23.7.7.59 mysql_row_seek()

MYSQL_ROW_OFFSET mysql_row_seek(MYSQL_RES *result, MYSQL_ROW_OFFSET offset)

説明

クエリー結果セットの任意の行に行カーソルを設定します。offset 値は、一般に mysql_row_tell() または mysql_row_seek() から返される値である行オフセットです。この値は行番号ではありません。番号によって結果セット内の行にシークするには、代わりに mysql_data_seek() を使用します。

この関数は、結果セット構造にクエリーの結果全体を格納することを必要とするため、mysql_row_seek() は、mysql_use_result() とではなく、mysql_store_result() と一緒にのみ使用できます。

戻り値

行カーソルの前の値。この値は、mysql_row_seek() への後続の呼び出しに渡すことができます。

エラー

なし。

23.7.7.60 mysql_row_tell()

MYSQL_ROW_OFFSET mysql_row_tell(MYSQL_RES *result)

説明

最後の mysql_fetch_row() の行カーソルの現在の位置を返します。この値は、mysql_row_seek() への引数として使うことができます。

mysql_row_tell() は、mysql_use_result() のあとではなく、mysql_store_result() のあとにのみ使用します。

戻り値

行カーソルの現在のオフセット。

エラー

なし。

23.7.7.61 mysql_select_db()

int mysql_select_db(MYSQL *mysql, const char *db)

説明

db で指定されたデータベースを、mysql で指定された接続で、デフォルトの (現在の) データベースにさせます。その後のクエリーで、このデータベースは明示的なデータベース指定子を含まないテーブル参照のデフォルトになります。

データベースに使用する権限を持っているとして、接続されたユーザーを認証できないかぎり、mysql_select_db() は失敗します。

戻り値

成功の場合はゼロ。エラーが発生した場合、ゼロ以外。

エラー

- CR_COMMANDS_OUT_OF_SYNC

  コマンドが正しくない順番で実行されました。

- CR_SERVER_GONE_ERROR

  MySQL サーバーが存在しなくなりました。

- CR_SERVER_LOST

  サーバーへの接続がクエリー中に失われました。

- CR_UNKNOWN_ERROR

  不明なエラーが発生しました。

23.7.7.62 mysql_set_character_set()

int mysql_set_character_set(MYSQL *mysql, const char *csname)

説明

この関数は現在の接続のデフォルトの文字セットを設定するために使用します。文字列 csname は有効な文字セット名を指定します。接続の照合順序は文字セットのデフォルトの照合順序になります。この関数は SET NAMES ステートメントのように機能しますが、mysql->charset の値も設定するため、mysql_real_escape_string() によって使用される文字セットに影響を与えます

戻り値

成功の場合はゼロ。エラーが発生した場合、ゼロ以外。

例

```
MYSQL mysql;

mysql_init(&mysql);
if (!mysql_real_connect(&mysql,"host","user","passwd","database",0,NULL,0))
{
   fprintf(stderr, "Failed to connect to database: Error: %s\n",
         mysql_error(&mysql));
}

if (!mysql_set_character_set(&mysql, "utf8"))
{
   printf("New client character set: %s\n",
          mysql_character_set_name(&mysql));
}
```

23.7.7.63 mysql_set_local_infile_default()

void mysql_set_local_infile_default(MYSQL *mysql);

説明

LOAD LOCAL DATA INFILE ハンドラコールバック関数を、C クライアントライブラリによって内部で使用されるデフォルトに設定します。mysql_set_local_infile_handler() が呼び出されなかったか、その各コールバックに対して有効な関数を提供しない場合、ライブラリはこの関数を自動的に呼び出します。

戻り値

なし。

エラー

なし。

23.7.7.64 mysql_set_local_infile_handler()

```
void mysql_set_local_infile_handler(MYSQL *mysql, int (*local_infile_init)(void **, const char *, void *), int
(*local_infile_read)(void *, char *, unsigned int), void (*local_infile_end)(void *), int (*local_infile_error)(void *, char*,
unsigned int), void *userdata);
```

説明

この関数は LOAD DATA LOCAL INFILE ステートメントの実行中に使用されるコールバックをインストールします。それにより、アプリケーションプログラムは、ローカル (クライアント側) データファイルの読み取りを制御できます。引数は、接続ハンドラ、コールバック関数への一連のポインタ、およびコールバックが情報を共有するために使用できるデータ領域へのポインタです。

mysql_set_local_infile_handler() を使うには、次のコールバック関数を書く必要があります。

```
int
local_infile_init(void **ptr, const char *filename, void *userdata);
```

初期化関数。これは、必要な設定を実行する、データファイルを開く、データ構造を割り当てるなどのために 1 回呼び出されます。最初の void** 引数はポインタへのポインタです。ほかの各コールバックに渡される値へのポインタ (つまり、*ptr) を設定できます (void* として)。コールバックは、この指示先の値を使用して、状態情報を保守できます。userdata 引数は mysql_set_local_infile_handler() に渡される同じ値です。

初期化関数が成功の場合はゼロを返し、エラーの場合はゼロ以外を返すようにします。

```
int
local_infile_read(void *ptr, char *buf, unsigned int buf_len);
```

データ読み取り関数。これはデータファイルを読み取るために繰り返し呼び出されます。buf は、読み取られたデータを格納するバッファーを指示し、buf_len はコールバックが読み取り、バッファーに格納できる最大バイト数です。(それより少ないバイトを読み取ることはできますが、それより多くのバイトを読み取るべきではありません。)

戻り値は読み取ったバイト数で、それ以上のデータを読み取ることができなかった場合はゼロです (これは EOF を示します)。エラーが発生した場合、ゼロ未満の値を返します。

```
void
local_infile_end(void *ptr)
```

終了関数。これは、local_infile_read() がゼロ (EOF) またはエラーを返したあとに 1 回呼び出されます。この関数内では、local_infile_init() によって割り当てられたメモリーの割り当てを解除し、必要なほかのクリーンアップを実行します。それは、初期化関数がエラーを返した場合でも呼び出されます。

```
int
local_infile_error(void *ptr,
                   char *error_msg,
                   unsigned int error_msg_len);
```

エラー処理関数。これは、ほかのいずれかの関数がエラーを返した場合に、ユーザーに返すテキストのエラーメッセージを取得するために呼び出されます。error_msg はメッセージが書き込まれるバッファーを指示し、error_msg_len はバッファーの長さです。最大 error_msg_len−1 バイト長で、NULL 終端文字列としてメッセージを書き込みます。

戻り値はエラー番号です。

一般に、ほかのコールバックは ptr によって指示されるデータ構造にエラーメッセージを格納するため、local_infile_error() はそこから error_msg にメッセージをコピーできます。

C コードで mysql_set_local_infile_handler() を呼び出し、ポインタをコールバック関数に渡したあとに、LOAD DATA LOCAL INFILEステートメントを発行できます (たとえば、mysql_query() を使用して)。クライアントライブラリは自動的にコールバックを呼び出します。LOAD DATA LOCAL INFILE は、local_infile_init() コールバックに 2 番目のパラメータとして渡されます。

戻り値

なし。

エラー

なし。

23.7.7.65 mysql_set_server_option()

```
int mysql_set_server_option(MYSQL *mysql, enum enum_mysql_set_option option)
```

説明

接続のオプションを有効または無効にします。option には次のいずれかの値を指定できます。

オプション	説明
MYSQL_OPTION_MULTI_STATEMENTS_ON	複数ステートメントのサポートを有効にします
MYSQL_OPTION_MULTI_STATEMENTS_OFF	複数ステートメントのサポートを無効にします

複数ステートメントのサポートを有効にする場合、mysql_next_result() を呼び出すループを使用して、それ以上の結果があるかどうかを判断することによって、mysql_query() または mysql_real_query() への呼び出しからの結果を取得してください。例については、セクション23.7.17「複数ステートメント実行の C API サポート」を参照してください。

MYSQL_OPTION_MULTI_STATEMENTS_ON による複数ステートメントのサポートの有効化は、CLIENT_MULTI_STATEMENTS フラグを mysql_real_connect() に渡すことによって、それを有効化する場合とまったく同じ効果を持つわけではありません。CLIENT_MULTI_STATEMENTS は CLIENT_MULTI_RESULTS も有効にします。プログラムで CALL SQL ステートメントを使用している場合、複数結果のサポートを有効にする必要があります。これは、MYSQL_OPTION_MULTI_STATEMENTS_ON だけでは、CALL の使用を許可するのに不十分であることを意味します。

戻り値

成功の場合はゼロ。エラーが発生した場合、ゼロ以外。

エラー

- CR_COMMANDS_OUT_OF_SYNC

  コマンドが正しくない順番で実行されました。

- CR_SERVER_GONE_ERROR

  MySQL サーバーが存在しなくなりました。

- CR_SERVER_LOST

  サーバーへの接続がクエリー中に失われました。

- ER_UNKNOWN_COM_ERROR

  サーバーは mysql_set_server_option() をサポートしていなかった (これはサーバーが 4.1.1 より古い場合) か、または設定しようとしたオプションをサポートしていませんでした。

23.7.7.66 mysql_shutdown()

```
int mysql_shutdown(MYSQL *mysql, enum mysql_enum_shutdown_level shutdown_level)
```

説明

データベースサーバーにシャットダウンするように求めます。接続されたユーザーは SHUTDOWN 権限を持っている必要があります。MySQL 5.6 サーバーは 1 種類のシャットダウンしかサポートしていません。shutdown_level は SHUTDOWN_DEFAULT と等しい必要があります。目的のレベルを選択できるようにするため、追加のシャットダウンレベルが計画されています。古いバージョンの libmysqlclient ヘッダーや呼び出し mysql_shutdown() でコンパイルされた、ダイナミックリンクされた実行可能ファイルは、古い libmysqlclient ダイナミックライブラリと一緒に使用する必要があります。

シャットダウンプロセスについては、セクション5.1.12「シャットダウンプロセス」で説明しています。

戻り値

成功の場合はゼロ。エラーが発生した場合、ゼロ以外。

エラー

- CR_COMMANDS_OUT_OF_SYNC

  コマンドが正しくない順番で実行されました。

- CR_SERVER_GONE_ERROR

  MySQL サーバーが存在しなくなりました。

- CR_SERVER_LOST

  サーバーへの接続がクエリー中に失われました。

- CR_UNKNOWN_ERROR

  不明なエラーが発生しました。

23.7.7.67 mysql_sqlstate()

```
const char *mysql_sqlstate(MYSQL *mysql)
```

説明

最近実行した SQL ステートメントに対する SQLSTATE エラーコードを含む、NULL 終端文字列を返します。エラーコードは 5 つの文字から構成されます。'00000' は「エラーなし」を意味します。値は ANSI SQL と ODBC によって規定されています。可能な値のリストについては、付録B「エラー、エラーコード、および一般的な問題」を参照してください。

mysql_sqlstate() によって返される SQLSTATE 値は、mysql_errno() によって返される MySQL 固有のエラー番号とは異なります。たとえば、mysql クライアントプログラムは、次の形式を使用してエラーを表示します。ここで、1146 は mysql_errno() 値で、'42S02' は対応する mysql_sqlstate() 値です。

```
shell> SELECT * FROM no_such_table;
ERROR 1146 (42S02): Table 'test.no_such_table' doesn't exist
```

すべての MySQL エラー番号が SQLSTATE エラーコードにマップされるわけではありません。値 'HY000' (一般エラー) がマップされていないエラー番号に使われます。

mysql_real_connect() が失敗したあとに mysql_sqlstate() を呼び出した場合、mysql_sqlstate() は有益な値を返さないことがあります。たとえば、これは、ホストがサーバーによってブロックされ、SQLSTATE 値がクライアントに送信されることなく、接続が閉じられた場合に発生します。

戻り値

SQLSTATE エラーコードを含む NULL 終端文字列。

関連項目

セクション23.7.7.14「mysql_errno()」、セクション23.7.7.15「mysql_error()」、およびセクション23.7.11.27「mysql_stmt_sqlstate()」を参照してください。

23.7.7.68 mysql_ssl_set()

```
my_bool mysql_ssl_set(MYSQL *mysql, const char *key, const char *cert, const char *ca, const char *capath,
const char *cipher)
```

説明

mysql_ssl_set() は SSL を使ったセキュアな接続を確立するために使用します。それは mysql_real_connect() の前に呼び出す必要があります。

クライアントライブラリで SSL のサポートが有効にされていないかぎり、mysql_ssl_set() は何も行いません。

mysql は mysql_init() から返される接続ハンドラです。ほかのパラメータは次のように指定されます。

- key は鍵ファイルへのパス名です。

- cert は証明書ファイルへのパス名です。
- ca は認証局ファイルへのパス名です。
- capath は PEM 形式の信頼された SSL CA 証明書を格納するディレクトリのパス名です。
- cipher は SSL 暗号化に使用する許可される暗号のリストです。

未使用の SSL パラメータはすべて NULL として指定できます。

戻り値

この関数は常に 0 を返します。SSL の設定が正しくない場合、接続を試みると、mysql_real_connect() はエラーを返します。

23.7.7.69 mysql_stat()

const char *mysql_stat(MYSQL *mysql)

説明

mysqladmin status コマンドによって提供されているものと似た情報を含む文字列を返します。これには、秒単位での稼働時間と実行中のスレッド、質問、再ロード、および開いているテーブルの数が含まれます。

戻り値

サーバーのステータスを説明する文字列。エラーが発生した場合は NULL。

エラー

- CR_COMMANDS_OUT_OF_SYNC

  コマンドが正しくない順番で実行されました。

- CR_SERVER_GONE_ERROR

  MySQL サーバーが存在しなくなりました。

- CR_SERVER_LOST

  サーバーへの接続がクエリー中に失われました。

- CR_UNKNOWN_ERROR

  不明なエラーが発生しました。

23.7.7.70 mysql_store_result()

MYSQL_RES *mysql_store_result(MYSQL *mysql)

説明

mysql_query() または mysql_real_query() を呼び出したあとに、結果セットを正常に生成するすべてのステートメント (SELECT、SHOW、DESCRIBE、EXPLAIN、CHECK TABLE など) に対して、mysql_store_result() または mysql_use_result() を呼び出す必要があります。結果セットの処理を終了したら、mysql_free_result() も呼び出す必要があります。

ほかのステートメントに対して、mysql_store_result() または mysql_use_result() を呼び出す必要はありませんが、あらゆるケースで mysql_store_result() を呼び出しても、それは何の害も及ぼしたり、顕著なパフォーマンスの低下を招いたりすることはありません。mysql_store_result() がゼロ以外の値を返すかどうかをチェックすることによって、ステートメントに結果セットがあるかどうかを検出できます (これについてはあとで詳しく説明しています)。

複数ステートメントのサポートを有効にする場合、mysql_next_result() を呼び出すループを使用して、それ以上の結果があるかどうかを判断することによって、mysql_query() または mysql_real_query() への呼び出しからの結果を取得してください。例については、セクション23.7.17「複数ステートメント実行の C API サポート」を参照してください。

ステートメントが結果セットを返すかどうかを知る必要がある場合は、mysql_field_count() を使ってこれをチェックできます。セクション23.7.7.22「mysql_field_count()」を参照してください。

mysql_store_result() はクライアントへのクエリーの結果全体を読み取り、MYSQL_RES 構造を割り当て、この構造に結果を配置します。

ステートメントが結果セットを返さなかった場合 (たとえば、それが INSERT ステートメントであった場合)、mysql_store_result() は NULL ポインタを返します。

mysql_store_result() は結果セットの読み取りに失敗した場合も、NULL ポインタを返します。mysql_error() が空でない文字列を返すか、mysql_errno() がゼロ以外を返すか、または mysql_field_count() がゼロを返すかどうかをチェックすることによって、エラーが発生したかどうかをチェックできます。

返される行がない場合、空の結果セットが返されます。(空の結果セットは戻り値としての NULL ポインタとは異なります。)

mysql_store_result() を呼び出し、NULL ポインタでない結果が返されたあとに、mysql_num_rows() を呼び出して、結果セット内にある行数を調べることができます。

mysql_fetch_row() を呼び出して、結果セットから行をフェッチしたり、mysql_row_seek() と mysql_row_tell() を呼び出して、結果セット内の現在の行の位置を取得または設定したりできます。

セクション23.7.15.1「mysql_query() が成功を返したあとに mysql_store_result() が NULL を返すことがあるのはなぜか」を参照してください。

戻り値

結果を含む MYSQL_RES 結果構造。エラーが発生した場合は NULL (0)。

エラー

mysql_store_result() は成功した場合に、mysql_error() と mysql_errno() をリセットします。

- CR_COMMANDS_OUT_OF_SYNC

  コマンドが正しくない順番で実行されました。

- CR_OUT_OF_MEMORY

  メモリー不足。

- CR_SERVER_GONE_ERROR

  MySQL サーバーが存在しなくなりました。

- CR_SERVER_LOST

  サーバーへの接続がクエリー中に失われました。

- CR_UNKNOWN_ERROR

  不明なエラーが発生しました。

23.7.7.71 mysql_thread_id()

unsigned long mysql_thread_id(MYSQL *mysql)

説明

現在の接続のスレッド ID を返します。この値は、mysql_kill() への引数として使用して、スレッドを強制終了できます。

接続が失われ、mysql_ping() によって再接続した場合、スレッド ID が変わります。これは、スレッド ID を取得して、あとのためにそれを格納すべきでないことを意味します。必要に応じて、それを取得すべきです。

> **注記**
>
> スレッド ID が 32 ビットより大きくなった (一部のシステムで発生する可能性があります) 場合、この関数は正常に機能しません。mysql_thread_id() の問題を避けるには、そ

れを使用しないでください。接続 ID を取得するには、SELECT CONNECTION_ID() クエリーを実行して、結果を取得します。

戻り値

現在の接続のスレッド ID。

エラー

なし。

23.7.7.72 mysql_use_result()

MYSQL_RES *mysql_use_result(MYSQL *mysql)

説明

mysql_query() または mysql_real_query() を呼び出したあとに、結果セットを正常に生成するすべてのステートメント (SELECT、SHOW、DESCRIBE、EXPLAIN、CHECK TABLE など) に対して、mysql_store_result() または mysql_use_result() を呼び出す必要があります。結果セットの処理を終了したら、mysql_free_result() も呼び出す必要があります。

mysql_use_result() は結果セットの取得を開始しますが、mysql_store_result() のように、実際に結果セットをクライアントに読み込みません。代わりに、mysql_fetch_row() への呼び出しを行うことによって、各行を個別に取得する必要があります。これは、クエリーの結果を一時テーブルやローカルバッファーに保存することなく、サーバーから直接読み取ります。これは、mysql_store_result() よりいくぶん高速で、使用するメモリーがはるかに少なくなります。クライアントは現在の行と max_allowed_packet バイトまで拡大する可能性のある通信バッファーにのみメモリーを割り当てます。

一方、クライアント側で、各行について大量の処理を行う場合またはユーザーが ^S (スクロール停止) を入力する可能性のある画面に出力が送信される場合は、mysql_use_result() を使用しないでください。これはサーバーを拘束し、ほかのスレッドが、データのフェッチ元のテーブルを更新するのを妨げます。

mysql_use_result() を使う場合は、NULL 値が返されるまで mysql_fetch_row() を実行する必要があり、そうしないと、次のクエリーの結果セットの一部として、フェッチされていない行が返されます。この実行を忘れると、C API によって、エラー「コマンドは同期されていません。このコマンドは現在実行できません」が生成されます。

mysql_data_seek()、mysql_row_seek()、mysql_row_tell()、mysql_num_rows()、または mysql_affected_rows() を mysql_use_result() から返される結果と一緒に使用できません。また、mysql_use_result() が終了するまで、ほかのクエリーを発行することもできません。(ただし、すべての行をフェッチしたあとに、mysql_num_rows() はフェッチした行数を正確に返します。)

結果セットの処理を終了したら、mysql_free_result() を呼び出す必要があります。

libmysqld 組み込みサーバーを使用する場合、mysql_free_result() が呼び出されるまで、取得される各行によって、メモリーの使用量が徐々に増加するため、メモリーのメリットが本質的に失われます。

戻り値

MYSQL_RES結果構造。エラーが発生した場合は NULL。

エラー

mysql_use_result() は成功すると、mysql_error() と mysql_errno() をリセットします。

- CR_COMMANDS_OUT_OF_SYNC

  コマンドが正しくない順番で実行されました。

- CR_OUT_OF_MEMORY

  メモリー不足。

- CR_SERVER_GONE_ERROR

MySQL サーバーが存在しなくなりました。

- CR_SERVER_LOST

  サーバーへの接続がクエリー中に失われました。

- CR_UNKNOWN_ERROR

  不明なエラーが発生しました。

23.7.7.73 mysql_warning_count()

unsigned int mysql_warning_count(MYSQL *mysql)

説明

前の SQL ステートメントの実行中に生成されたエラー、警告、および注意事項の数を返します。

戻り値

警告カウント。

エラー

なし。

23.7.8 C API プリペアドステートメント

MySQL クライアント/サーバープロトコルはプリペアドステートメントの使用を規定します。この機能は、mysql_stmt_init() 初期化関数によって返される MYSQL_STMT ステートメントハンドラデータ構造を使用します。プリペアド実行はステートメントを複数回実行するための効率的な方法です。ステートメントはまずその実行を準備するために解析されます。その後、初期化関数によって返されるステートメントハンドルを使用して、1 回または複数回実行されます。

プリペアド実行は、主にクエリーが 1 回しか解析されないため、複数回実行されるステートメントの場合に、直接実行より高速になります。直接実行の場合、クエリーは実行されるたびに解析されます。さらに、プリペアド実行は、プリペアドステートメントの実行ごとに、パラメータにデータを送信する必要があるだけであるため、ネットワークトラフィックの削減も実現できます。

プリペアドステートメントは、状況によっては、パフォーマンスの向上をもたらさない場合があります。最善の結果を得るため、プリペアドステートメントと非プリペアドステートメントの両方でアプリケーションをテストし、最高のパフォーマンスを発揮するものを選択してください。

プリペアドステートメントのもう 1 つの利点は、それがクライアントとサーバー間のデータ転送の効率を向上するバイナリプロトコルを使用することです。

プリペアドステートメントとして使用できる SQL ステートメントのリストについては、セクション13.5「準備済みステートメントのための SQL 構文」を参照してください。

プリペアドステートメントによって参照されているテーブルやビューのメタデータの変更が検出され、それが次に実行されるときに、ステートメントが自動再準備されます。詳細については、セクション8.9.4「プリペアドステートメントおよびストアドプログラムのキャッシュ」を参照してください。

23.7.9 C API プリペアドステートメントデータ構造

プリペアドステートメントはいくつかのデータ構造を使用します。

- ステートメントハンドルを取得するには、MYSQL 接続ハンドラを mysql_stmt_init() に渡します。これは、MYSQL_STMT データ構造へポインタを返します。この構造は、ステートメントのその後の処理に使用されます。ステートメントの準備を指定するには、MYSQL_STMT ポインタとステートメント文字列を mysql_stmt_prepare() に渡します。
- プリペアドステートメントの入力パラメータを提供するには、MYSQL_BIND 構造を設定して、それらを mysql_stmt_bind_param() に渡します。出力カラム値を受け取るには、MYSQL_BIND 構造を設定し、それらを mysql_stmt_bind_result() に渡します。

- MYSQL_TIME 構造は両方向で時間データを転送するために使われます。

次の説明では、プリペアドステートメントのデータ型を詳しく示します。それらの使用方法の例については、セクション23.7.11.10「mysql_stmt_execute()」およびセクション23.7.11.11「mysql_stmt_fetch()」を参照してください。

- MYSQL_STMT

  この構造は、プリペアドステートメントのハンドルです。ハンドルを作成するには、mysql_stmt_init() を呼び出します。これは、MYSQL_STMT へのポインタを返します。ハンドルは、mysql_stmt_close() でクローズする (その時点でハンドルが無効になります) まで、その後のすべてのステートメントの操作に使われます。

  MYSQL_STMT 構造にはアプリケーションで使用することを意図されたメンバーがありません。アプリケーションでは、MYSQL_STMT 構造をコピーしようとしないでください。そのようなコピーが使用可能である保証はありません。

  複数のステートメントハンドルを単一の接続に関連付けることができます。ハンドル数の制限は、使用可能なシステムリソースによって異なります。

- MYSQL_BIND

  この構造は、ステートメントの入力 (サーバーに送信されるデータ値) と出力 (サーバーから返される結果値) の両方に使用されます。

  - 入力の場合は、mysql_stmt_bind_param() で MYSQL_BIND 構造を使用して、パラメータデータ値を mysql_stmt_execute() で使用するために、バッファーにバインドします。
  - 出力の場合は、mysql_stmt_bind_result() で MYSQL_BIND 構造を使用して、mysql_stmt_fetch() によって行のフェッチで使用するために、バッファーを結果セットカラムにバインドします。

  MYSQL_BIND 構造を使用するには、その内容をゼロにして初期化してから、そのメンバーを適切に設定します。たとえば、3 つの MYSQL_BIND 構造の配列を宣言し、初期化するには、このコードを使用します。

```
MYSQL_BIND bind[3];
memset(bind, 0, sizeof(bind));
```

  MYSQL_BIND 構造には、アプリケーションプログラムによって使用するための次のメンバーが含まれます。いくつかのメンバーに対して使用する方法は、構造が入力に使われるか、出力に使われるかによって異なります。

  - enum enum_field_types buffer_type

    バッファーの種類。このメンバーは、ステートメントパラメータまたは結果セットカラムにバインドされている C 言語変数のデータ型を示します。入力の場合、buffer_type はサーバーに送信される値を格納する変数の型を示します。出力の場合、それはサーバーから受け取った値を格納すべき変数の型を示します。許可される buffer_type 値については、セクション23.7.9.1「C API プリペアドステートメントタイプコード」を参照してください。

  - void *buffer

    データの転送に使用されるバッファーへのポインタ。これは C 言語変数のアドレスです。

    入力の場合、buffer はステートメントパラメータのデータ値を格納する変数へのポインタです。mysql_stmt_execute() を呼び出すと、MySQL は、ステートメント内の対応するパラメータマーカー (ステートメント文字列内に ? で指定される) の代わりに変数に格納されている値を使用します。

    出力の場合、buffer は結果セットカラム値を返すための変数へのポインタです。mysql_stmt_fetch() を呼び出すと、MySQL は結果セットの現在の行からのカラム値をこの変数に保存します。呼び出しが戻ると、値にアクセスできます。

    MySQL が、クライアント側の C 言語値とサーバー側の SQL 値間の型変換を実行する必要性を最小にするには、対応する SQL 値の型に似た型を持つ C 変数を使用します。

    - 数値データ型の場合、buffer は正しい数値 C 型の変数を指すべきです。整数変数 (これは単一バイト値の場合に char または大きな値の場合に整数型を指定できます) の場合、後述する is_unsigned メンバーを設定して、変数が unsigned 属性を持つかどうかも指定してください。
    - 文字 (非バイナリ) およびバイナリ文字列データ型の場合、buffer は文字バッファーを指すようにしてください。

- 日付および時間データ型の場合、buffer は MYSQL_TIME 構造を指すようにしてください。

C 型と SQL 型間のマッピングに関するガイドラインと型変換に関する注意事項については、セクション 23.7.9.1「C API プリペアドステートメントタイプコード」およびセクション23.7.9.2「C API プリペアドステートメント型変換」を参照してください。

- unsigned long buffer_length

*buffer のバイト単位での実際のサイズ。これはバッファーに格納できるデータの最大量を示します。文字およびバイナリ C データの場合、buffer_length 値は、入力値を指定する mysql_stmt_bind_param() と一緒に使用したときに、*buffer の長さ、または mysql_stmt_bind_result() と一緒に使用したときに、バッファーにフェッチできる出力データバイトの最大数を指定します。

- unsigned long *length

*buffer に格納されたデータの実際のバイト数を示す unsigned long 変数へのポインタ。length は文字またはバイナリ C データに対して使われます。

入力パラメータデータバインディングの場合、*buffer に格納されるパラメータ値の実際の長さを示すように *length を設定します。これは mysql_stmt_execute() で使用されます。

出力値バインディングの場合、mysql_stmt_fetch() を呼び出したときに、MySQL によって *length が設定されます。mysql_stmt_fetch() の戻り値によって、長さの解釈方法が決まります。

- 戻り値が 0 の場合、*length はパラメータ値の実際の長さを示します。
- 戻り値が MYSQL_DATA_TRUNCATED の場合、*length はパラメータ値の切り捨てされていない長さを示します。この場合、*length と buffer_length の最小は、値の実際の長さを示します。

buffer_type 値によって、データ値の長さが決まるため、length は数値および時間データ型に対しては無視されます。

戻り値の長さをフェッチする前に判断する必要がある場合は、いくつかの戦略について、セクション 23.7.11.11「mysql_stmt_fetch()」を参照してください。

- my_bool *is_null

このメンバーは、値が NULL の場合に true で、それが NULL でない場合に false である my_bool 変数を指示します。入力では、*is_null を true に設定して、ステートメントパラメータとして NULL 値を渡していることを示します。

is_null はブール型スカラーへのポインタで、NULL 値の指定方法の柔軟性を提供します。

- データ値が常に NULL である場合、カラムのバインド時に、buffer_type 値として MYSQL_TYPE_NULL を使用します。is_null を含むほかの MYSQL_BIND メンバーは重要ではありません。
- データ値が常に NOT NULL である場合は、is_null = (my_bool*) 0 を設定し、バインドする変数に適切にほかのメンバーを設定します。
- ほかのすべての場合に、ほかのメンバーを適切に設定し、is_null に my_bool 変数のアドレスを設定します。実行と実行の間に、その変数の値を true または false に適切に設定して、対応するデータ値がそれぞれ NULL であるか、NOT NULL であるかを示します。

出力の場合、行をフェッチすると、MySQL は、ステートメントから返された結果セットカラム値が NULL であるかどうかに従って、is_null で指示されている値を true または false に設定します。

- my_bool is_unsigned

このメンバーは、unsigned (char、short int、int、long long int) を指定できるデータ型を持つ C 変数に適用されます。buffer によって指示されている変数が unsigned の場合は、is_unsigned を true に設定し、それ以外の場合は false に設定します。たとえば、signed char 変数を buffer にバインドする場合、MYSQL_TYPE_TINY のタイプコードを指定し、is_unsigned を false に設定します。代わりに unsigned char をバインドする場合、タイプコードは同じですが、is_unsigned を true にしてください。(char の場合、

それは符号付きか符号なしか定義されないため、signed char や unsigned char を使用して、符号の有無を明示することをお勧めします。)

is_unsigned はクライアント側の C 言語変数のみに適用されます。それはサーバー側の対応する SQL 値の符号の有無については何も示しません。たとえば、int 変数を使用して、BIGINT UNSIGNED カラムの値を提供する場合、int は符号付きの型であるため、is_unsigned は false にすべきです。unsigned int 変数を使用して、BIGINT カラムの値を提供する場合、unsigned int は符号なしの型であるため、is_unsigned を true にすべきです。MySQL は符号付きの値と符号なしの値で、両方向で正しい変換を実行しますが、切り捨てが発生した場合、警告が生成されます。

- my_bool *error

  出力の場合、このメンバーを my_bool 変数を指すように設定し、行のフェッチ操作のあとに、そこに格納されるパラメータの切り捨て情報を保持します。切り捨てのレポートを有効にすると、mysql_stmt_fetch() は MYSQL_DATA_TRUNCATED を返し、切り捨てが発生したパラメータの MYSQL_BIND 構造で、*error が true になります。切り捨ては、符号または桁落ち、または文字列が長すぎてカラムに収まらなかったことを示します。切り捨てのレポートはデフォルトで有効ですが、MYSQL_REPORT_DATA_TRUNCATION オプションを使用して mysql_options() を呼び出すことによって制御できます。

- MYSQL_TIME

  この構造は、DATE、TIME、DATETIME、および TIMESTAMP データを直接サーバーと送受信するために使われます。buffer メンバーを MYSQL_TIME 構造を指すように設定し、MYSQL_BIND 構造の buffer_type メンバーをいずれかの時間型 (MYSQL_TYPE_TIME、MYSQL_TYPE_DATE、MYSQL_TYPE_DATETIME、MYSQL_TYPE_TIMESTAMP) に設定します。

  MYSQL_TIME 構造には、次の表に示すメンバーが格納されます。

メンバー	説明
unsigned int year	年
unsigned int month	月
unsigned int day	日
unsigned int hour	時間
unsigned int minute	分
unsigned int second	秒
my_bool neg	時間が負であるかどうかを示すブールフラグ
unsigned long second_part	ミリ秒単位での秒の小数部 (MySQL 5.6.4 より前では未使用)

  時間値の指定された型に適用する MYSQL_TIME 構造の部分のみが使われます。year、month、および day 要素は、DATE、DATETIME、および TIMESTAMP 値に使用されます。hour、minute、および second 要素は、TIME、DATETIME、および TIMESTAMP 値に使用されます。セクション23.7.19「C API プリペアドステートメントの日時値の処理」を参照してください。

23.7.9.1 C API プリペアドステートメントタイプコード

MYSQL_BIND 構造の buffer_type メンバーは、ステートメントパラメータまたは結果セットカラムにバインドされている C 言語変数のデータ型を示します。入力の場合、buffer_type はサーバーに送信される値を格納する変数の型を示します。出力の場合、それはサーバーから受け取った値を格納すべき変数の型を示します。

次の表に、サーバーに送信される入力値に対して、MYSQL_BIND 構造の buffer_type メンバーに許可される値を示します。表に、使用できる C 変数型、対応するタイプコード、および提供された値を変換なしで使用できる SQL データ型を示します。バインドする C 言語変数のデータ型に従って、buffer_type 値を選択します。整数型の場合、is_unsigned メンバーも設定して、変数が符号付きか符号なしかを示してください。

入力変数 C 型	buffer_type 値	宛先の値の SQL 型
signed char	MYSQL_TYPE_TINY	TINYINT
short int	MYSQL_TYPE_SHORT	SMALLINT
int	MYSQL_TYPE_LONG	INT
long long int	MYSQL_TYPE_LONGLONG	BIGINT

入力変数 C 型	buffer_type 値	宛先の値の SQL 型
float	MYSQL_TYPE_FLOAT	FLOAT
double	MYSQL_TYPE_DOUBLE	DOUBLE
MYSQL_TIME	MYSQL_TYPE_TIME	TIME
MYSQL_TIME	MYSQL_TYPE_DATE	DATE
MYSQL_TIME	MYSQL_TYPE_DATETIME	DATETIME
MYSQL_TIME	MYSQL_TYPE_TIMESTAMP	TIMESTAMP
char[]	MYSQL_TYPE_STRING	TEXT、CHAR、VARCHAR
char[]	MYSQL_TYPE_BLOB	BLOB、BINARY、VARBINARY
	MYSQL_TYPE_NULL	NULL

セクション23.7.9「C API プリペアドステートメントデータ構造」で、is_null メンバーの説明に示されているように、MYSQL_TYPE_NULL を使用します。

入力文字列データの場合、値が文字 (非バイナリ) であるか、バイナリ文字列であるかによって、MYSQL_TYPE_STRING または MYSQL_TYPE_BLOB を使用します。

- MYSQL_TYPE_STRING は文字入力の文字列データを示します。この値は、character_set_client システム変数によって示される文字セットに含まれるものとみなされます。サーバーが値を異なる文字セットでカラムに格納する場合、値をその文字セットに変換します。
- MYSQL_TYPE_BLOB はバイナリ入力文字列データを示します。この値は binary 文字セットを持つものとして扱われます。つまり、それは、バイト文字列として扱われ、変換は行われません。

次の表に、サーバーから受信される出力値の MYSQL_BIND 構造の buffer_type メンバーの許可される値を示します。表には、受信された値の SQL 型、そのような値の結果セットメタデータに含まれる対応するタイプコード、および変換なしで SQL 値を受信するために MYSQL_BIND 構造にバインドする推奨される C 言語データ型を示します。バインドする C 言語変数のデータ型に従って、buffer_type 値を選択します。整数型の場合、is_unsigned メンバーも設定して、変数が符号付きか符号なしかを示してください。

受信した値の SQL 型	buffer_type 値	出力変数 C 型
TINYINT	MYSQL_TYPE_TINY	signed char
SMALLINT	MYSQL_TYPE_SHORT	short int
MEDIUMINT	MYSQL_TYPE_INT24	int
INT	MYSQL_TYPE_LONG	int
BIGINT	MYSQL_TYPE_LONGLONG	long long int
FLOAT	MYSQL_TYPE_FLOAT	float
DOUBLE	MYSQL_TYPE_DOUBLE	double
DECIMAL	MYSQL_TYPE_NEWDECIMAL	char[]
YEAR	MYSQL_TYPE_SHORT	short int
TIME	MYSQL_TYPE_TIME	MYSQL_TIME
DATE	MYSQL_TYPE_DATE	MYSQL_TIME
DATETIME	MYSQL_TYPE_DATETIME	MYSQL_TIME
TIMESTAMP	MYSQL_TYPE_TIMESTAMP	MYSQL_TIME
CHAR、BINARY	MYSQL_TYPE_STRING	char[]
VARCHAR、VARBINARY	MYSQL_TYPE_VAR_STRING	char[]
TINYBLOB、TINYTEXT	MYSQL_TYPE_TINY_BLOB	char[]
BLOB、TEXT	MYSQL_TYPE_BLOB	char[]
MEDIUMBLOB、MEDIUMTEXT	MYSQL_TYPE_MEDIUM_BLOB	char[]
LONGBLOB、LONGTEXT	MYSQL_TYPE_LONG_BLOB	char[]
BIT	MYSQL_TYPE_BIT	char[]

23.7.9.2 C API プリペアドステートメント型変換

プリペアドステートメントは、クライアント側で、サーバー側の SQL 値に対応する C 言語変数を使用して、クライアントとサーバー間でデータを転送します。クライアント側の C 変数型とサーバー側の対応する SQL 値型間に不一致がある場合、MySQL は両方向で暗黙的な型変換を実行します。

MySQL はサーバー側の SQL 値に対するタイプコードを認識しています。MYSQL_BIND 構造内の buffer_type 値は、クライアント側の値を保持する C 変数のタイプコードを示します。2 つのコードはまとめて、実行する必要がある変換がある場合に、それを MySQL に伝えます。次にいくつかの例を示します。

- int 変数で MYSQL_TYPE_LONG を使用して、FLOAT カラムに格納される整数値をサーバーに渡した場合、MySQL はその値を格納する前に浮動小数点形式に変換します。
- SQL MEDIUMINT カラム値をフェッチするが、MYSQL_TYPE_LONGLONG の buffer_type 値を指定し、宛先バッファーとして、型 long long int の C 変数を使用した場合、MySQL は、long long int (8 バイト変数) に格納するために、MEDIUMINT 値 (これは 8 バイト未満を必要とする) を変換します。
- 255 の値を含む数値カラムを char[4] 文字配列にフェッチし、MYSQL_TYPE_STRING の buffer_type 値を指定した場合、配列内の結果の値は 4 バイト文字列 '255\0' になります。
- MySQL は元のサーバー側値の文字列表現として DECIMAL 値を返します。そのため、対応する C 型は char[] です。たとえば、12.345 はクライアントに '12.345' として返されます。MYSQL_TYPE_NEWDECIMAL を指定し、文字列バッファーを MYSQL_BIND 構造にバインドすると、mysql_stmt_fetch() は変換なしで値を文字列としてバッファーに格納します。代わりに、数値変数とタイプコードを指定した場合、mysql_stmt_fetch() は文字列形式 DECIMAL 値を数値形式に変換します。
- MYSQL_TYPE_BIT タイプコードでは、BIT 値が文字列バッファーに返されます。そのため、対応する C 型は char[] です。値はクライアント側での解釈を必要とするビット文字列を表します。扱いやすい型として値を返すため、次のいずれかの式の型を使用して、値を整数にキャストさせることができます。

  ```
  SELECT bit_col + 0 FROM t
  SELECT CAST(bit_col AS UNSIGNED) FROM t
  ```

  値を取得するには、値を保持するために十分な大きさの整数変数をバインドし、適切な対応する整数タイプコードを指定します。

カラム値のフェッチに使用される MYSQL_BIND 構造に変数をバインドする前に、結果セットの各カラムのタイプコードをチェックできます。これは、型変換を避けるために使用するもっともよい変数型を判断する場合、望ましいと考えられます。タイプコードを取得するには、mysql_stmt_execute() によって、プリペアドステートメントを実行したあとに、mysql_stmt_result_metadata() を呼び出します。メタデータは、セクション23.7.11.23「mysql_stmt_result_metadata()」およびセクション23.7.5「C API データ構造」で説明しているように、結果セットのタイプコードへのアクセスを提供します。

サーバーから返される結果セット内の出力文字列にバイナリデータが含まれているか、非バイナリデータが含まれているかを確認するには、結果セットメタデータの charsetnr 値が 63 であるかどうかをチェックします (セクション23.7.5「C API データ構造」を参照してください)。その場合、文字セットは binary で、これは非バイナリデータではなく、バイナリを示します。これにより、BINARY と CHAR、VARBINARY と VARCHAR、および BLOB 型と TEXT 型を区別できます。

MYSQL_FIELD カラムメタデータ構造の max_length メンバーを設定させる場合 (mysql_stmt_attr_set() を呼び出すことによって)、結果セットの max_length 値が、バイナリ表現の長さではなく、結果値の最長の文字列表現の長さを示すことに注意してください。つまり、max_length は必ずしも、プリペアドステートメントに使用されるバイナリプロトコルによって値をフェッチするために必要なバッファーのサイズに対応していません。値をフェッチする変数の型に従って、バッファーのサイズを選択します。たとえば、値 -128 を格納する TINYINT カラムは、4 の max_length 値を持つ可能性があります。ただし、TINYINT 値のバイナリ表現は、ストレージに 1 バイトしか必要としないため、値を格納する signed char 変数を提供し、値が符号付きであることを示す is_unsigned を設定できます。

プリペアドステートメントによって参照されているテーブルやビューのメタデータの変更が検出され、それが次に実行されるときに、ステートメントが自動再準備されます。詳細については、セクション8.9.4「プリペアドステートメントおよびストアドプログラムのキャッシュ」を参照してください。

23.7.10 C API プリペアドステートメント関数の概要

プリペアドステートメントの処理に使用可能な関数の概要をここで示し、あとのセクションで詳しく説明します。セクション23.7.11「C API プリペアドステートメント関数の説明」を参照してください。

関数	説明
mysql_stmt_affected_rows()	プリペアド UPDATE、DELETE、または INSERT ステートメントによって変更、削除、挿入された行数を返します

関数	説明
mysql_stmt_attr_get()	プリペアドステートメントの属性の値を取得します
mysql_stmt_attr_set()	プリペアドステートメントの属性を設定します
mysql_stmt_bind_param()	アプリケーションデータバッファーをプリペアド SQL ステートメントのパラメータマーカーに関連付けます
mysql_stmt_bind_result()	アプリケーションデータバッファーを結果セット内のカラムに関連付けます
mysql_stmt_close()	プリペアドステートメントによって使用されているメモリーを解放します
mysql_stmt_data_seek()	ステートメント結果セット内の任意の行番号にシークします
mysql_stmt_errno()	最後のステートメント実行のエラー番号を返します
mysql_stmt_error()	最後のステートメント実行のエラーメッセージを返します
mysql_stmt_execute()	プリペアドステートメントを実行します
mysql_stmt_fetch()	結果セットから次のデータの行をフェッチし、バインドされたすべてのカラムのデータを返します
mysql_stmt_fetch_column()	結果セットの現在の行の 1 つのカラムのデータをフェッチします
mysql_stmt_field_count()	最近のステートメントの結果カラムの数を返します
mysql_stmt_free_result()	ステートメントハンドルに割り当てられているリソースを解放します
mysql_stmt_init()	MYSQL_STMT 構造にメモリーを割り当て、それを初期化します
mysql_stmt_insert_id()	プリペアドステートメントによって AUTO_INCREMENT カラムに対して生成された ID を返します
mysql_stmt_next_result()	複数結果の実行での次の結果を返すか、初期化します
mysql_stmt_num_rows()	バッファーされたステートメント結果セットから行カウントを返します
mysql_stmt_param_count()	プリペアドステートメント内のパラメータの数を返します。
mysql_stmt_param_metadata()	(結果セットの形式でパラメータメタデータを返します) 現在、この関数は何も実行しません
mysql_stmt_prepare()	SQL ステートメント文字列の実行を準備します
mysql_stmt_reset()	サーバー内のステートメントバッファーをリセットします
mysql_stmt_result_metadata()	結果セットの形式でプリペアドステートメントメタデータを返します
mysql_stmt_row_seek()	mysql_stmt_row_tell() から返される値を使用して、ステートメント結果セット内の行オフセットにシークします。
mysql_stmt_row_tell()	ステートメント行カーソル位置を返します
mysql_stmt_send_long_data()	長いデータをまとめてサーバーに送信します
mysql_stmt_sqlstate()	最後のステートメント実行の SQLSTATE エラーコードを返します
mysql_stmt_store_result()	クライアントへの完全な結果セットを取得します

mysql_stmt_init() を呼び出して、ステートメントハンドルを作成し、次にmysql_stmt_prepare() を呼び出して、ステートメント文字列を準備し、mysql_stmt_bind_param() を呼び出してパラメータデータを提供し、mysql_stmt_execute() を呼び出して、ステートメントを実行します。mysql_stmt_bind_param() によって提供される各バッファー内のパラメータ値を変更することによって、mysql_stmt_execute() を繰り返すことができます。

mysql_stmt_send_long_data() を使用して、テキストやバイナリデータをまとめてサーバーに送信できます。セクション23.7.11.26「mysql_stmt_send_long_data()」を参照してください。

ステートメントが SELECT または結果セットを生成するその他のステートメントである場合、mysql_stmt_prepare() は、mysql_stmt_result_metadata() によって、MYSQL_RES 結果セットの形式で、結果セットメタデータ情報も返します。

mysql_stmt_fetch() が自動的に結果バッファーにデータを返すように、mysql_stmt_bind_result() を使用して、それらのバッファーを提供できます。これは行単位のフェッチです。

ステートメントの実行が完了したら、mysql_stmt_close() を使用してステートメントハンドルをクローズして、それに関連付けられているすべてのリソースを解放できるようにします。

mysql_stmt_result_metadata() を呼び出すことによって、SELECT ステートメントの結果セットメタデータを取得した場合、mysql_free_result() を使用してメタデータも解放してください。

実行ステップ

ステートメントを準備して実行するには、アプリケーションはこれらのステップに従います。

1. mysql_stmt_init() によってプリペアドステートメントハンドルを作成します。サーバーでステートメントを準備するには、mysql_stmt_prepare() を呼び出して、それに SQL ステートメントを格納する文字列を渡します。
2. ステートメントが結果セットを生成する場合、mysql_stmt_result_metadata() を呼び出して、結果セットメタデータを取得します。このメタデータは、クエリーによって返される行を格納するものとは別のものですが、それ自体が結果セットの形式になります。メタデータ結果セットは、結果に存在するカラム数を示し、各カラムに関する情報を含んでいます。
3. mysql_stmt_bind_param() を使用して、パラメータの値を設定します。すべてのパラメータを設定する必要があります。そうしないと、ステートメント実行がエラーを返すか、予期しない結果を生成します。
4. mysql_stmt_execute() を呼び出してステートメントを実行します。
5. ステートメントが結果セットを生成する場合、mysql_stmt_bind_result() を呼び出すことによって、行値を取得するために使用するデータバッファーをバインドします。
6. mysql_stmt_fetch() を繰り返し呼び出すことによって、それ以上の行が見つからなくなるまで、行ごとにデータをバッファーにフェッチします。
7. パラメータ値を変更してステートメントを再実行することによって、必要に応じて、ステップ 3 から 6 を繰り返します。

mysql_stmt_prepare() が呼び出されると、MySQL クライアント/サーバープロトコルはこれらのアクションを実行します。

- サーバーはステートメントを解析し、ステートメント ID を割り当てて、クライアントに正常ステータスを返送します。さらに、それが結果セット指向のステートメントである場合に、パラメータの合計数、カラムカウント、およびそのメタデータも送信します。この呼び出し中に、サーバーによって、ステートメントのすべての構文とセマンティクスがチェックされます。
- サーバーがそのステートメントのプール内からステートメントを識別できるように、クライアントはその後の操作にこのステートメント ID を使用します。

mysql_stmt_execute() が呼び出されると、MySQL クライアント/サーバープロトコルはこれらのアクションを実行します。

- クライアントはステートメントハンドルを使用して、サーバーにパラメータデータを送信します。
- サーバーは、クライアントによって提供された ID を使用してステートメントを識別し、パラメータマーカーを新しく提供されたデータで置換して、ステートメントを実行します。ステートメントが結果セットを生成する場合、サーバーはクライアントにデータを返送します。そうでない場合、正常ステータスと変更、削除、または挿入された行数を送信します。

mysql_stmt_fetch() が呼び出されると、MySQL クライアント/サーバープロトコルはこれらのアクションを実行します。

- クライアントは結果セットの現在の行からデータを読み取り、必要な変換を行なって、それをアプリケーションデータバッファーに配置します。アプリケーションバッファーの型が、サーバーから返されるフィールドの型のそれと同じである場合、変換は簡単です。

エラーが発生した場合、mysql_stmt_errno()、mysql_stmt_error()、および mysql_stmt_sqlstate() をそれぞれ使用して、ステートメントエラー番号、エラーメッセージ、および SQLSTATE コードを取得できます。

プリペアドステートメントのロギング

mysql_stmt_prepare() と mysql_stmt_execute() C API 関数によって実行されたプリペアドステートメントに対して、サーバーは一般クエリーログに Prepare および Execute 行を書き込むため、ステートメントが準備され、実行されたタイミングがわかります。

次のようにステートメントを準備して実行するものとします。

1. mysql_stmt_prepare() を呼び出して、ステートメント文字列 "SELECT ?" を準備します。

2. mysql_stmt_bind_param() を呼び出して、値 3 をプリペアドステートメント内のパラメータにバインドします。
3. mysql_stmt_execute() を呼び出して、プリペアドステートメントを実行します。

以前の呼び出しの結果として、サーバーは一般クエリーログに次の行を書き込みます。

```
Prepare  [1] SELECT ?
Execute  [1] SELECT 3
```

ログ内の各 Prepare および Execute 行は、[N] ステートメント識別子でタグ付けされるため、ログに記録されているプリペアドステートメントを追跡できます。N は正の整数です。クライアントに、同時にアクティブな複数のプリペアドステートメントがある場合、N は 1 より大きくなる可能性があります。? パラメータのデータ値の置換後、各 Execute 行にプリペアドステートメントが示されます。

23.7.11 C API プリペアドステートメント関数の説明

クエリーを準備して実行するには、次のセクションで詳しく説明している関数を使用します。

MYSQL_STMT 構造で動作するすべての関数はプリフィクス mysql_stmt_ で始まります。

MYSQL_STMT ハンドルを作成するには、mysql_stmt_init() 関数を使用します。

23.7.11.1 mysql_stmt_affected_rows()

my_ulonglong mysql_stmt_affected_rows(MYSQL_STMT *stmt)

説明

mysql_stmt_affected_rows() は、mysql_stmt_execute() によるステートメントの実行直後に呼び出すことができます。それはプリペアドステートメントを除けば mysql_affected_rows() に似ています。この関数によって返される影響を受ける行の値の説明については、セクション23.7.7.1「mysql_affected_rows()」を参照してください。

エラー

なし。

例

セクション23.7.11.10「mysql_stmt_execute()」の例を参照してください。

23.7.11.2 mysql_stmt_attr_get()

my_bool mysql_stmt_attr_get(MYSQL_STMT *stmt, enum enum_stmt_attr_type option, void *arg)

説明

ステートメント属性の現在の値を取得するために使用できます。

option 引数は取得するオプションです。arg はオプション値を含む変数を指すようにするべきです。オプションが整数の場合、arg は整数の値を指すべきです。

オプションとオプションの型のリストについては、セクション23.7.11.3「mysql_stmt_attr_set()」を参照してください。

戻り値

成功の場合はゼロ。option が不明な場合はゼロ以外。

エラー

なし。

23.7.11.3 mysql_stmt_attr_set()

my_bool mysql_stmt_attr_set(MYSQL_STMT *stmt, enum enum_stmt_attr_type option, const void *arg)

説明

プリペアドステートメントの動作に影響を与えるために使用できます。この関数を複数回呼び出して、複数のオプションを設定できます。

option 引数は、設定するオプションです。arg 引数はオプションの値です。arg は、目的の属性値に設定される変数を指すべきです。変数の型は次の表に示すようなものになります。

次の表に可能な option 値を示します。

オプション	引数型	関数
STMT_ATTR_UPDATE_MAX_LENGTH	my_bool *	1 に設定されている場合、mysql_stmt_store_result() によってメタデータ MYSQL_FIELD->max_length 値が更新されます。
STMT_ATTR_CURSOR_TYPE	unsigned long *	mysql_stmt_execute() が呼び出されたときにステートメントに対して開くカーソルの型。*arg は CURSOR_TYPE_NO_CURSOR (デフォルト) または CURSOR_TYPE_READ_ONLY を指定できます。
STMT_ATTR_PREFETCH_ROWS	unsigned long *	カーソルの使用時に、サーバーからフェッチする行数。*arg は 1 から unsigned long の最大値の範囲で指定できます。デフォルトは 1 です。

CURSOR_TYPE_READ_ONLY で STMT_ATTR_CURSOR_TYPE オプションを使用する場合、mysql_stmt_execute() を呼び出すと、ステートメントに対して、カーソルが開かれます。前の mysql_stmt_execute() 呼び出しからの開いているカーソルがすでに存在する場合、それは新しいカーソルを開く前にカーソルを閉じます。mysql_stmt_reset() もステートメントの再実行を準備する前に、開いているカーソルを閉じます。mysql_stmt_free_result() はすべての開いているカーソルを閉じます。

プリペアドステートメントに対してカーソルを開く場合、mysql_stmt_store_result() は結果セットをクライアント側にバッファーさせるため、必要ありません。

戻り値

成功の場合はゼロ。option が不明な場合はゼロ以外。

エラー

なし。

例

次の例は、プリペアドステートメントに対してカーソルを開き、一度にフェッチする行数を 5 に設定します。

```
MYSQL_STMT *stmt;
int rc;
unsigned long type;
unsigned long prefetch_rows = 5;

stmt = mysql_stmt_init(mysql);
type = (unsigned long) CURSOR_TYPE_READ_ONLY;
rc = mysql_stmt_attr_set(stmt, STMT_ATTR_CURSOR_TYPE, (void*) &type);
/* ... check return value ... */
rc = mysql_stmt_attr_set(stmt, STMT_ATTR_PREFETCH_ROWS,
                         (void*) &prefetch_rows);
/* ... check return value ... */
```

23.7.11.4 mysql_stmt_bind_param()

my_bool mysql_stmt_bind_param(MYSQL_STMT *stmt, MYSQL_BIND *bind)

説明

mysql_stmt_bind_param() は、mysql_stmt_prepare() に渡された SQL ステートメント内のパラメータマーカーに入力データをバインドするために使用します。それは MYSQL_BIND 構造を使用して、データを提供します。bind は MYSQL_BIND 構造の配列のアドレスです。クライアントライブラリは、配列に、クエリーに存在する ? パラメータマーカーごとに 1 つの要素が含まれることを期待します。

次のステートメントを準備するとします。

```
INSERT INTO mytbl VALUES(?,?,?)
```

パラメータをバインドする場合、MYSQL_BIND 構造の配列は、3 つの要素が含まれている必要があり、このように宣言できます。

```
MYSQL_BIND bind[3];
```

セクション23.7.9「C API プリペアドステートメントデータ構造」に、各 MYSQL_BIND 要素のメンバーおよび入力値を提供するためにそれらを設定する方法について説明しています。

戻り値

成功の場合はゼロ。エラーが発生した場合、ゼロ以外。

エラー

- CR_UNSUPPORTED_PARAM_TYPE

  変換はサポートされていません。buffer_type 値は無効か、サポートされるいずれの型でもない可能性があります。

- CR_OUT_OF_MEMORY

  メモリー不足。

- CR_UNKNOWN_ERROR

  不明なエラーが発生しました。

例

セクション23.7.11.10「mysql_stmt_execute()」の例を参照してください。

23.7.11.5 mysql_stmt_bind_result()

my_bool mysql_stmt_bind_result(MYSQL_STMT *stmt, MYSQL_BIND *bind)

説明

mysql_stmt_bind_result() は、結果セット内の出力カラムをデータバッファーおよび長さバッファーに関連付ける (つまりバインドする) ために使用します。mysql_stmt_fetch() がデータをフェッチするために呼び出されると、MySQL クライアント/サーバープロトコルはバインドされたカラムのデータを、指定されたバッファーに配置します。

mysql_stmt_fetch() を呼び出す前に、すべてのカラムがバッファーにバインドされている必要があります。bind は MYSQL_BIND 構造の配列のアドレスです。クライアントライブラリは、配列に、結果セットのカラムごとに 1 つの要素が含まれることを期待します。カラムを MYSQL_BIND 構造にバインドしない場合、mysql_stmt_fetch() は単にデータフェッチを無視します。プロトコルはまとめてデータ値を返さないため、バッファーはデータ値を保持できる十分な大きさがあるべきです。

結果セットが部分的に取得されたあとでも、カラムはいつでもバインドまたは再バインドできます。新しいバインドタスクは、次に mysql_stmt_fetch() が呼び出されたときに有効になります。アプリケーションが結果セット内のカラムをバインドし、mysql_stmt_fetch() を呼び出すものとします。クライアント/サーバープロトコルはバインドされたバッファーにデータを返します。その後、アプリケーションはカラムを別の一連のバッファーにバインドするものとします。mysql_stmt_fetch() への次の呼び出し時に、プロトコルは新しくバインドされたバッファーにデータを配置します。

カラムをバインドするには、アプリケーションで mysql_stmt_bind_result() を呼び出し、値を格納する出力バッファーの型、アドレス、および長さを渡します。セクション23.7.9「C API プリペアドステートメントデータ構造」に各 MYSQL_BIND 要素のメンバーおよび出力値を受け取るために設定する方法について説明します。

戻り値

成功の場合はゼロ。エラーが発生した場合、ゼロ以外。

エラー

- CR_UNSUPPORTED_PARAM_TYPE

  変換はサポートされていません。buffer_type 値は無効か、サポートされるいずれの型でもない可能性があります。

- CR_OUT_OF_MEMORY

  メモリー不足。

- CR_UNKNOWN_ERROR

  不明なエラーが発生しました。

例

セクション23.7.11.11「mysql_stmt_fetch()」の例を参照してください。

23.7.11.6 mysql_stmt_close()

my_bool mysql_stmt_close(MYSQL_STMT *)

説明

プリペアドステートメントをクローズします。mysql_stmt_close() は stmt によって指されているステートメントハンドルの割り当ても解除します。

現在のステートメントに保留中か未読の結果がある場合、この関数は、次のクエリーを実行できるように、それらをキャンセルします。

戻り値

成功の場合はゼロ。エラーが発生した場合、ゼロ以外。

エラー

- CR_SERVER_GONE_ERROR

  MySQL サーバーが存在しなくなりました。

- CR_UNKNOWN_ERROR

  不明なエラーが発生しました。

例

セクション23.7.11.10「mysql_stmt_execute()」の例を参照してください。

23.7.11.7 mysql_stmt_data_seek()

void mysql_stmt_data_seek(MYSQL_STMT *stmt, my_ulonglong offset)

説明

ステートメント結果セット内の任意の行にシークします。offset 値は行番号で、0 から mysql_stmt_num_rows(stmt)-1 までの範囲であるべきです。

この関数は、ステートメント結果セット構造に最後に実行されたクエリーの結果全体を含める必要があるため、mysql_stmt_data_seek() は、mysql_stmt_store_result() と一緒にのみ使用できます。

戻り値

なし。

エラー

なし。

23.7.11.8 mysql_stmt_errno()

unsigned int mysql_stmt_errno(MYSQL_STMT *stmt)

説明

stmt によって指定されたステートメントに対して、mysql_stmt_errno() は最近呼び出された成功または失敗した可能性のあるステートメント API 関数のエラーコードを返します。ゼロの戻り値はエラーが発生しなかったこ

とを意味します。クライアントのエラーメッセージ番号は、MySQL errmsg.h ヘッダーファイルに一覧表示されています。サーバーのエラーメッセージ番号は、mysqld_error.h に一覧表示されています。エラーは付録B「エラー、エラーコード、および一般的な問題」にも一覧表示しています。

戻り値

エラーコード値。エラーが発生しなかった場合はゼロ。

エラー

なし。

23.7.11.9 mysql_stmt_error()

```
const char *mysql_stmt_error(MYSQL_STMT *stmt)
```

説明

stmt によって指定されたステートメントに対して、mysql_stmt_error() は最近呼び出された、成功または失敗した可能性のあるステートメント API 関数のエラーメッセージを格納する NULL 終端文字列を返します。エラーが発生しなかった場合、空の文字列 ("") が返されます。エラーをチェックするために、これらの 2 つのテストのいずれかを使用できます。

```
if(*mysql_stmt_errno(stmt))
{
  // an error occurred
}

if (mysql_stmt_error(stmt)[0])
{
  // an error occurred
}
```

クライアントエラーメッセージの言語は、MySQL クライアントライブラリを再コンパイルすることによって変更できます。現在、数種類の言語のエラーメッセージを選択できます。

戻り値

エラーについて説明する文字列。エラーが発生しなかった場合は空の文字列。

エラー

なし。

23.7.11.10 mysql_stmt_execute()

```
int mysql_stmt_execute(MYSQL_STMT *stmt)
```

説明

mysql_stmt_execute() はステートメントハンドルと関連付けられたプリペアドクエリーを実行します。現在バインドされているパラメータマーカー値がこの呼び出し中にサーバーに送信され、サーバーはそのマーカーをこの新しく提供されたデータと置き換えます。

mysql_stmt_execute() のあとのステートメント処理は、ステートメントの種類によって異なります。

- UPDATE、DELETE、または INSERT では、変更、削除、または挿入された行の数を mysql_stmt_affected_rows() を呼び出して見つけることができます。
- 結果セットを生成する SELECT などのステートメントでは、クエリー処理が発生するほかのすべての関数を呼び出す前に、mysql_stmt_fetch() を呼び出して、データをフェッチする必要があります。結果をフェッチする方法の詳細については、セクション23.7.11.11「mysql_stmt_fetch()」を参照してください。

  mysql_stmt_execute() の呼び出しに続いて、mysql_store_result() または mysql_use_result() を呼び出さないでください。それらの関数は、プリペアドステートメントからの結果を処理することを意図していません。

結果セットを生成するステートメントに対して、mysql_stmt_execute() に、ステートメントを実行する前に、mysql_stmt_attr_set() を呼び出すことによって、ステートメントのカーソルを開くように要求できます。ステートメントを複数回実行する場合、mysql_stmt_execute() は新しいカーソルを開く前に、開いているカーソルを閉じます。

プリペアドステートメントによって参照されているテーブルやビューのメタデータの変更が検出され、それが次に実行されるときに、ステートメントが自動再準備されます。詳細については、セクション8.9.4「プリペアドステートメントおよびストアドプログラムのキャッシュ」を参照してください。

戻り値

成功の場合はゼロ。エラーが発生した場合、ゼロ以外。

エラー

- CR_COMMANDS_OUT_OF_SYNC

  コマンドが正しくない順番で実行されました。

- CR_OUT_OF_MEMORY

  メモリー不足。

- CR_SERVER_GONE_ERROR

  MySQL サーバーが存在しなくなりました。

- CR_SERVER_LOST

  サーバーへの接続がクエリー中に失われました。

- CR_UNKNOWN_ERROR

  不明なエラーが発生しました。

例

次の例に、mysql_stmt_init()、mysql_stmt_prepare()、mysql_stmt_param_count()、mysql_stmt_bind_param()、mysql_stmt_execut および mysql_stmt_affected_rows() を使用して、テーブルを作成し、移入する方法を示します。mysql 変数は有効な接続ハンドルであるとみなされます。データの取得方法を示す例については、セクション 23.7.11.11「mysql_stmt_fetch()」を参照してください。

```
#define STRING_SIZE 50

#define DROP_SAMPLE_TABLE "DROP TABLE IF EXISTS test_table"
#define CREATE_SAMPLE_TABLE "CREATE TABLE test_table(col1 INT,\
                                     col2 VARCHAR(40),\
                                     col3 SMALLINT,\
                                     col4 TIMESTAMP)"
#define INSERT_SAMPLE "INSERT INTO \
                    test_table(col1,col2,col3) \
                    VALUES(?,?,?)"

MYSQL_STMT    *stmt;
MYSQL_BIND    bind[3];
my_ulonglong  affected_rows;
int           param_count;
short         small_data;
int           int_data;
char          str_data[STRING_SIZE];
unsigned long str_length;
my_bool       is_null;

if (mysql_query(mysql, DROP_SAMPLE_TABLE))
{
  fprintf(stderr, " DROP TABLE failed\n");
  fprintf(stderr, " %s\n", mysql_error(mysql));
  exit(0);
}

if (mysql_query(mysql, CREATE_SAMPLE_TABLE))
{
  fprintf(stderr, " CREATE TABLE failed\n");
  fprintf(stderr, " %s\n", mysql_error(mysql));
  exit(0);
}

/* Prepare an INSERT query with 3 parameters */
/* (the TIMESTAMP column is not named; the server */
/*  sets it to the current date and time) */
```

```
stmt = mysql_stmt_init(mysql);
if (!stmt)
{
  fprintf(stderr, " mysql_stmt_init(), out of memory\n");
  exit(0);
}
if (mysql_stmt_prepare(stmt, INSERT_SAMPLE, strlen(INSERT_SAMPLE)))
{
  fprintf(stderr, " mysql_stmt_prepare(), INSERT failed\n");
  fprintf(stderr, " %s\n", mysql_stmt_error(stmt));
  exit(0);
}
fprintf(stdout, " prepare, INSERT successful\n");

/* Get the parameter count from the statement */
param_count= mysql_stmt_param_count(stmt);
fprintf(stdout, " total parameters in INSERT: %d\n", param_count);

if (param_count != 3) /* validate parameter count */
{
  fprintf(stderr, " invalid parameter count returned by MySQL\n");
  exit(0);
}

/* Bind the data for all 3 parameters */

memset(bind, 0, sizeof(bind));

/* INTEGER PARAM */
/* This is a number type, so there is no need
   to specify buffer_length */
bind[0].buffer_type= MYSQL_TYPE_LONG;
bind[0].buffer= (char *)&int_data;
bind[0].is_null= 0;
bind[0].length= 0;

/* STRING PARAM */
bind[1].buffer_type= MYSQL_TYPE_STRING;
bind[1].buffer= (char *)str_data;
bind[1].buffer_length= STRING_SIZE;
bind[1].is_null= 0;
bind[1].length= &str_length;

/* SMALLINT PARAM */
bind[2].buffer_type= MYSQL_TYPE_SHORT;
bind[2].buffer= (char *)&small_data;
bind[2].is_null= &is_null;
bind[2].length= 0;

/* Bind the buffers */
if (mysql_stmt_bind_param(stmt, bind))
{
  fprintf(stderr, " mysql_stmt_bind_param() failed\n");
  fprintf(stderr, " %s\n", mysql_stmt_error(stmt));
  exit(0);
}

/* Specify the data values for the first row */
int_data= 10;             /* integer */
strncpy(str_data, "MySQL", STRING_SIZE); /* string  */
str_length= strlen(str_data);

/* INSERT SMALLINT data as NULL */
is_null= 1;

/* Execute the INSERT statement - 1*/
if (mysql_stmt_execute(stmt))
{
  fprintf(stderr, " mysql_stmt_execute(), 1 failed\n");
  fprintf(stderr, " %s\n", mysql_stmt_error(stmt));
  exit(0);
}

/* Get the number of affected rows */
affected_rows= mysql_stmt_affected_rows(stmt);
fprintf(stdout, " total affected rows(insert 1): %lu\n",
               (unsigned long) affected_rows);

if (affected_rows != 1) /* validate affected rows */
{
```

```
  fprintf(stderr, " invalid affected rows by MySQL\n");
  exit(0);
}

/* Specify data values for second row,
   then re-execute the statement */
int_data= 1000;
strncpy(str_data, "
       The most popular Open Source database",
       STRING_SIZE);
str_length= strlen(str_data);
small_data= 1000;         /* smallint */
is_null= 0;               /* reset */

/* Execute the INSERT statement - 2*/
if (mysql_stmt_execute(stmt))
{
  fprintf(stderr, " mysql_stmt_execute, 2 failed\n");
  fprintf(stderr, " %s\n", mysql_stmt_error(stmt));
  exit(0);
}

/* Get the total rows affected */
affected_rows= mysql_stmt_affected_rows(stmt);
fprintf(stdout, " total affected rows(insert 2): %lu\n",
               (unsigned long) affected_rows);

if (affected_rows != 1) /* validate affected rows */
{
  fprintf(stderr, " invalid affected rows by MySQL\n");
  exit(0);
}

/* Close the statement */
if (mysql_stmt_close(stmt))
{
  fprintf(stderr, " failed while closing the statement\n");
  fprintf(stderr, " %s\n", mysql_stmt_error(stmt));
  exit(0);
}
```

> **注記**
>
> プリペアドステートメント関数の使用の完全な例については、ファイル tests/mysql_client_test.c を参照してください。このファイルは MySQL ソース配布または Bazaar ソースリポジトリから取得できます。

23.7.11.11 mysql_stmt_fetch()

```
int mysql_stmt_fetch(MYSQL_STMT *stmt)
```

説明

mysql_stmt_fetch() は結果セットで次の行を返します。それは結果セットが存在する間にのみ呼び出すことができます。つまり、結果セットを生成する SELECT などのステートメントに対する mysql_stmt_execute() への呼び出し後です。

mysql_stmt_fetch() は、mysql_stmt_bind_result() によってバインドされたバッファーを使用して、行データを返します。それは、現在の行セット内のすべてのカラムについて、それらのバッファー内のデータを返し、長さが length ポインタに返されます。アプリケーションは mysql_stmt_fetch() を呼び出す前に、すべてのカラムをバインドする必要があります。

デフォルトで、結果セットはサーバーから一度に行がバッファーされずにフェッチされます。クライアントで結果セット全体をバッファーするには、データバッファーをバインドしたあとの mysql_stmt_fetch() を呼び出す前に、mysql_stmt_store_result() を呼び出します。

フェッチしたデータ値が NULL 値である場合、対応する MYSQL_BIND 構造の *is_null 値には TRUE (1) が含まれます。そうでない場合、データとその長さが、アプリケーションによって指定されたバッファーの型に基づいて、*buffer および *length 要素で返されます。次の表に示すように、各数値および時間型は固定の長さを持ちます。data_length に示されるように、文字列型の長さは実際のデータ値の長さによって異なります。

型	長さ
MYSQL_TYPE_TINY	1

型	長さ
MYSQL_TYPE_SHORT	2
MYSQL_TYPE_LONG	4
MYSQL_TYPE_LONGLONG	8
MYSQL_TYPE_FLOAT	4
MYSQL_TYPE_DOUBLE	8
MYSQL_TYPE_TIME	sizeof(MYSQL_TIME)
MYSQL_TYPE_DATE	sizeof(MYSQL_TIME)
MYSQL_TYPE_DATETIME	sizeof(MYSQL_TIME)
MYSQL_TYPE_STRING	data length
MYSQL_TYPE_BLOB	data_length

場合によって、mysql_stmt_fetch() でカラム値をフェッチする前にその長さを判断したいと考えることがあります。たとえば、値が、割り当てる必要があるスペースの量を知りたいと思う長い文字列や BLOB 値である場合があります。これを実現するには、これらの戦略を使用できます。

- mysql_stmt_fetch() を呼び出して、各行を取得する前に、STMT_ATTR_UPDATE_MAX_LENGTH を mysql_stmt_attr_set() に渡してから、mysql_stmt_store_result() を呼び出して、クライアント側で結果セット全体をバッファーします。STMT_ATTR_UPDATE_MAX_LENGTH 属性を設定すると、カラム値の最大長が mysql_stmt_result_metadata() によって返される結果セットメタデータの max_length によって示されます。
- 問題のカラムに対し、ゼロの長さのバッファーで mysql_stmt_fetch() を呼び出すと、実際の長さのポインタを格納できます。その後、mysql_stmt_fetch_column() で実際の長さを使用します。

```
real_length= 0;

bind[0].buffer= 0;
bind[0].buffer_length= 0;
bind[0].length= &real_length
mysql_stmt_bind_result(stmt, bind);

mysql_stmt_fetch(stmt);
if (real_length > 0)
{
  data= malloc(real_length);
  bind[0].buffer= data;
  bind[0].buffer_length= real_length;
  mysql_stmt_fetch_column(stmt, bind, 0, 0);
}
```

戻り値

戻り値	説明
0	成功、データがアプリケーションデータバッファーにフェッチされました。
1	エラーが発生しました。エラーコードとエラーメッセージは、mysql_stmt_errno() および mysql_stmt_error() を呼び出すことによって取得できます。
MYSQL_NO_DATA	それ以上の行/データは存在しません
MYSQL_DATA_TRUNCATED	データ切り捨てが発生しました

切り捨てレポートが有効にされている場合、MYSQL_DATA_TRUNCATED が返されます。この値が返されたときに、どのカラム値が切り捨てられたか判断するには、値をフェッチするために使用された MYSQL_BIND 構造の error メンバーをチェックします。切り捨てのレポートはデフォルトで有効ですが、MYSQL_REPORT_DATA_TRUNCATION オプションを使用して mysql_options() を呼び出すことによって制御できます。

エラー

- CR_COMMANDS_OUT_OF_SYNC

  コマンドが正しくない順番で実行されました。

- CR_OUT_OF_MEMORY

メモリー不足。

- CR_SERVER_GONE_ERROR

  MySQL サーバーが存在しなくなりました。

- CR_SERVER_LOST

  サーバーへの接続がクエリー中に失われました。

- CR_UNKNOWN_ERROR

  不明なエラーが発生しました。

- CR_UNSUPPORTED_PARAM_TYPE

  バッファー型
  は、MYSQL_TYPE_DATE、MYSQL_TYPE_TIME、MYSQL_TYPE_DATETIME、MYSQL_TYPE_TIMESTAMP
  ですが、データ型は DATE、TIME、DATETIME、または TIMESTAMP ではありません。

- サポートされていないほかのすべての変換エラーは、mysql_stmt_bind_result() から返されます。

例

次の例は、mysql_stmt_result_metadata()、mysql_stmt_bind_result()、および mysql_stmt_fetch() を使用して、テーブルからデータをフェッチする方法を示します。(この例は セクション23.7.11.10「mysql_stmt_execute()」に示す例によって挿入された 2 つの行を取得することを期待します)。mysql 変数は有効な接続ハンドルであるとみなされます。

```
#define STRING_SIZE 50

#define SELECT_SAMPLE "SELECT col1, col2, col3, col4 \
                       FROM test_table"

MYSQL_STMT    *stmt;
MYSQL_BIND    bind[4];
MYSQL_RES     *prepare_meta_result;
MYSQL_TIME    ts;
unsigned long length[4];
int           param_count, column_count, row_count;
short         small_data;
int           int_data;
char          str_data[STRING_SIZE];
my_bool       is_null[4];
my_bool       error[4];

/* Prepare a SELECT query to fetch data from test_table */
stmt = mysql_stmt_init(mysql);
if (!stmt)
{
  fprintf(stderr, " mysql_stmt_init(), out of memory\n");
  exit(0);
}
if (mysql_stmt_prepare(stmt, SELECT_SAMPLE, strlen(SELECT_SAMPLE)))
{
  fprintf(stderr, " mysql_stmt_prepare(), SELECT failed\n");
  fprintf(stderr, " %s\n", mysql_stmt_error(stmt));
  exit(0);
}
fprintf(stdout, " prepare, SELECT successful\n");

/* Get the parameter count from the statement */
param_count= mysql_stmt_param_count(stmt);
fprintf(stdout, " total parameters in SELECT: %d\n", param_count);

if (param_count != 0) /* validate parameter count */
{
  fprintf(stderr, " invalid parameter count returned by MySQL\n");
  exit(0);
}

/* Fetch result set meta information */
prepare_meta_result = mysql_stmt_result_metadata(stmt);
if (!prepare_meta_result)
{
  fprintf(stderr,
```

```
        " mysql_stmt_result_metadata(), \
          returned no meta information\n");
  fprintf(stderr, " %s\n", mysql_stmt_error(stmt));
  exit(0);
}

/* Get total columns in the query */
column_count= mysql_num_fields(prepare_meta_result);
fprintf(stdout,
        " total columns in SELECT statement: %d\n",
        column_count);

if (column_count != 4) /* validate column count */
{
  fprintf(stderr, " invalid column count returned by MySQL\n");
  exit(0);
}

/* Execute the SELECT query */
if (mysql_stmt_execute(stmt))
{
  fprintf(stderr, " mysql_stmt_execute(), failed\n");
  fprintf(stderr, " %s\n", mysql_stmt_error(stmt));
  exit(0);
}

/* Bind the result buffers for all 4 columns before fetching them */

memset(bind, 0, sizeof(bind));

/* INTEGER COLUMN */
bind[0].buffer_type= MYSQL_TYPE_LONG;
bind[0].buffer= (char *)&int_data;
bind[0].is_null= &is_null[0];
bind[0].length= &length[0];
bind[0].error= &error[0];

/* STRING COLUMN */
bind[1].buffer_type= MYSQL_TYPE_STRING;
bind[1].buffer= (char *)str_data;
bind[1].buffer_length= STRING_SIZE;
bind[1].is_null= &is_null[1];
bind[1].length= &length[1];
bind[1].error= &error[1];

/* SMALLINT COLUMN */
bind[2].buffer_type= MYSQL_TYPE_SHORT;
bind[2].buffer= (char *)&small_data;
bind[2].is_null= &is_null[2];
bind[2].length= &length[2];
bind[2].error= &error[2];

/* TIMESTAMP COLUMN */
bind[3].buffer_type= MYSQL_TYPE_TIMESTAMP;
bind[3].buffer= (char *)&ts;
bind[3].is_null= &is_null[3];
bind[3].length= &length[3];
bind[3].error= &error[3];

/* Bind the result buffers */
if (mysql_stmt_bind_result(stmt, bind))
{
  fprintf(stderr, " mysql_stmt_bind_result() failed\n");
  fprintf(stderr, " %s\n", mysql_stmt_error(stmt));
  exit(0);
}

/* Now buffer all results to client (optional step) */
if (mysql_stmt_store_result(stmt))
{
  fprintf(stderr, " mysql_stmt_store_result() failed\n");
  fprintf(stderr, " %s\n", mysql_stmt_error(stmt));
  exit(0);
}

/* Fetch all rows */
row_count= 0;
fprintf(stdout, "Fetching results ...\n");
while (!mysql_stmt_fetch(stmt))
{
```

```
 row_count++;
 fprintf(stdout, "  row %d\n", row_count);

 /* column 1 */
 fprintf(stdout, "   column1 (integer)  : ");
 if (is_null[0])
  fprintf(stdout, " NULL\n");
 else
  fprintf(stdout, " %d(%ld)\n", int_data, length[0]);

 /* column 2 */
 fprintf(stdout, "   column2 (string)   : ");
 if (is_null[1])
  fprintf(stdout, " NULL\n");
 else
  fprintf(stdout, " %s(%ld)\n", str_data, length[1]);

 /* column 3 */
 fprintf(stdout, "   column3 (smallint) : ");
 if (is_null[2])
  fprintf(stdout, " NULL\n");
 else
  fprintf(stdout, " %d(%ld)\n", small_data, length[2]);

 /* column 4 */
 fprintf(stdout, "   column4 (timestamp): ");
 if (is_null[3])
  fprintf(stdout, " NULL\n");
 else
  fprintf(stdout, " %04d-%02d-%02d %02d:%02d:%02d (%ld)\n",
                  ts.year, ts.month, ts.day,
                  ts.hour, ts.minute, ts.second,
                  length[3]);
 fprintf(stdout, "\n");
}

/* Validate rows fetched */
fprintf(stdout, " total rows fetched: %d\n", row_count);
if (row_count != 2)
{
  fprintf(stderr, " MySQL failed to return all rows\n");
  exit(0);
}

/* Free the prepared result metadata */
mysql_free_result(prepare_meta_result);


/* Close the statement */
if (mysql_stmt_close(stmt))
{
  fprintf(stderr, " failed while closing the statement\n");
  fprintf(stderr, " %s\n", mysql_stmt_error(stmt));
  exit(0);
}
```

23.7.11.12 mysql_stmt_fetch_column()

int mysql_stmt_fetch_column(MYSQL_STMT *stmt, MYSQL_BIND *bind, unsigned int column, unsigned long offset)

説明

現在の結果セット行から 1 つのカラムをフェッチします。bind はデータを配置すべきバッファーを提供します。それは mysql_stmt_bind_result() の場合と同じように設定してください。column はフェッチするカラムを示します。最初のカラムは 0 と番号付けされます。offset はデータの取得を開始するデータ値内のオフセットです。これはデータ値を個々にフェッチするために使用できます。値の開始はオフセット 0 です。

戻り値

成功の場合はゼロ。エラーが発生した場合、ゼロ以外。

エラー

- CR_INVALID_PARAMETER_NO

無効なカラム番号。

- CR_NO_DATA

  結果セットの末尾にすでに達しています。

23.7.11.13 mysql_stmt_field_count()

unsigned int mysql_stmt_field_count(MYSQL_STMT *stmt)

説明

ステートメントハンドラの最近のステートメントのカラムの数を返します。この値は、結果セットを生成しない INSERT または DELETE などのステートメントの場合ゼロです。

mysql_stmt_prepare() を呼び出して、ステートメントを準備したあとに、mysql_stmt_field_count() を呼び出すことができます。

戻り値

結果セット内のカラムの数を表す符号なし整数。

エラー

なし。

23.7.11.14 mysql_stmt_free_result()

my_bool mysql_stmt_free_result(MYSQL_STMT *stmt)

説明

プリペアドステートメントの実行によって生成された結果セットに関連付けられたメモリーを解放します。ステートメントに開いているカーソルがある場合、mysql_stmt_free_result() はそれを閉じます。

戻り値

成功の場合はゼロ。エラーが発生した場合、ゼロ以外。

エラー

23.7.11.15 mysql_stmt_init()

MYSQL_STMT *mysql_stmt_init(MYSQL *mysql)

説明

MYSQL_STMT ハンドルを作成します。ハンドルは mysql_stmt_close(MYSQL_STMT *) で解放してください。

詳細については、セクション23.7.9「C API プリペアドステートメントデータ構造」も参照してください。

戻り値

成功した場合は MYSQL_STMT 構造へのポインタ。メモリー不足の場合は NULL。

エラー

- CR_OUT_OF_MEMORY

  メモリー不足。

23.7.11.16 mysql_stmt_insert_id()

my_ulonglong mysql_stmt_insert_id(MYSQL_STMT *stmt)

説明

プリペアド INSERT または UPDATE ステートメントによって、AUTO_INCREMENT カラムに生成された値を返します。AUTO_INCREMENT フィールドを含むテーブルに対して、プリペアド INSERT ステートメントを実行したあとに、この関数を使用します。

詳細については、セクション23.7.7.37「mysql_insert_id()」を参照してください。

戻り値

プリペアドステートメントの実行中に自動的に生成されたか明示的に設定された AUTO_INCREMENT カラムの値、または LAST_INSERT_ID(expr) 関数によって生成された値。ステートメントで AUTO_INCREMENT 値を設定しない場合、戻り値は不定になります。

エラー

なし。

23.7.11.17 mysql_stmt_next_result()

int mysql_stmt_next_result(MYSQL_STMT *mysql)

説明

この関数は、複数の結果セットを返すことができるプリペアド CALL ステートメントを使用して、ストアドプロシージャーを実行する場合に使用します。mysql_stmt_next_result() を呼び出すループを使用して、それ以上の結果があるかどうかを判断します。プロシージャーに OUT または INOUT パラメータがある場合、それらの値はほかの結果セットに続いて、単一の行結果セットとして返されます。値は、プロシージャーパラメータリストに宣言されている順番で表示されます。

mysql_stmt_next_result() はそれ以上の結果が存在するかどうかを示すステータスを返します。mysql_stmt_next_result() がエラーを返した場合、それ以上の結果はありません。

mysql_stmt_next_result() の各呼び出しの前に、現在の結果で結果セット (結果のステータスだけでなく) が生成された場合、現在の結果に対して mysql_stmt_free_result() を呼び出す必要があります。

mysql_stmt_next_result() を呼び出したあとの接続の状態は、mysql_stmt_execute() を呼び出した場合のようになります。このことは、mysql_stmt_bind_result()、mysql_stmt_affected_rows() などを呼び出すことができることを意味します。

mysql_more_results() を呼び出して、それ以上の結果があるかどうかをテストすることもできます。ただし、この関数は接続の状態を変更しないため、それが true を返した場合は、さらに mysql_stmt_next_result() を呼び出して、次の結果に進む必要があります。

mysql_stmt_next_result() の使用方法を示す例については、セクション23.7.20「C API のプリペアド CALL ステートメントのサポート」を参照してください。

戻り値

戻り値	説明
0	成功し、それ以上の結果が存在します
-1	成功し、それ以上の結果が存在しません
0 より大きい	エラーが発生しました

エラー

- CR_COMMANDS_OUT_OF_SYNC

  コマンドが正しくない順番で実行されました。

- CR_SERVER_GONE_ERROR

  MySQL サーバーが存在しなくなりました。

- CR_SERVER_LOST

サーバーへの接続がクエリー中に失われました。

- CR_UNKNOWN_ERROR

  不明なエラーが発生しました。

23.7.11.18 mysql_stmt_num_rows()

```
my_ulonglong mysql_stmt_num_rows(MYSQL_STMT *stmt)
```

説明

結果セット内の行数を返します。

mysql_stmt_num_rows() の使用は、ステートメントハンドルに結果セット全体をバッファーするために、mysql_stmt_store_result() を使用したどうかによって異なります。mysql_stmt_store_result() を使用した場合、mysql_stmt_num_rows() をただちに呼び出すことができます。そうでない場合、行をフェッチしながら、それをカウントしないかぎり、行カウントは使用できません。

mysql_stmt_num_rows() は SELECT などの結果セットを返すステートメントと一緒に使用することを意図しています。INSERT、UPDATE、または DELETE などのステートメントでは、影響を受けた行の数を mysql_stmt_affected_rows() で取得できます。

戻り値

結果セット内の行数。

エラー

なし。

23.7.11.19 mysql_stmt_param_count()

```
unsigned long mysql_stmt_param_count(MYSQL_STMT *stmt)
```

説明

プリペアドステートメントに存在するパラメータマーカーの数を返します。

戻り値

ステートメント内のパラメータ数を表す符号なし long 整数。

エラー

なし。

例

セクション23.7.11.10「mysql_stmt_execute()」の例を参照してください。

23.7.11.20 mysql_stmt_param_metadata()

```
MYSQL_RES *mysql_stmt_param_metadata(MYSQL_STMT *stmt)
```

この関数は現在何もしません。

説明

戻り値

エラー

23.7.11.21 mysql_stmt_prepare()

```
int mysql_stmt_prepare(MYSQL_STMT *stmt, const char *stmt_str, unsigned long length)
```

説明

mysql_stmt_init() によってステートメントハンドルが返されるとすると、文字列 stmt_str によって指示される SQL ステートメントを準備し、ステータス値を返します。文字列の長さは length 引数によって指定してください。文字列は 1 つの SQL ステートメントで構成されている必要があります。ステートメントに終端のセミコロン (「;」) または \g を追加しないでください。

アプリケーションでは、疑問符 (?) 文字を SQL 文字列の適切な位置に埋め込むことによって、1 つまたは複数のパラメータマーカーを SQL ステートメントに含めることができます。

マーカーは SQL ステートメント内の特定の場所でのみ正当です。たとえば、それらは、INSERT ステートメントの VALUES() リスト (行のカラム値を指定するため) で、または比較値を指定するために WHERE 句内のカラムとの比較で使用できます。ただし、それらは識別子 (テーブルまたはカラム名など)、または = 等号などのバイナリ演算子の両方のオペランドに指定するために使用できません。パラメータの型を判断することは不可能な場合があるため、後者の制限は必要です。一般に、パラメータは Data Manipulation Language (DML) ステートメント内でのみ正当で、Data Definition Language (DDL) ステートメント内では許可されません。

パラメータマーカーは、ステートメントを実行する前に mysql_stmt_bind_param() を使用してアプリケーション変数にバインドされている必要があります。

プリペアドステートメントによって参照されているテーブルやビューのメタデータの変更が検出され、それが次に実行されるときに、ステートメントが自動再準備されます。詳細については、セクション8.9.4「プリペアドステートメントおよびストアドプログラムのキャッシュ」を参照してください。

戻り値

成功の場合はゼロ。エラーが発生した場合、ゼロ以外。

エラー

- CR_COMMANDS_OUT_OF_SYNC

  コマンドが正しくない順番で実行されました。

- CR_OUT_OF_MEMORY

  メモリー不足。

- CR_SERVER_GONE_ERROR

  MySQL サーバーが存在しなくなりました。

- CR_SERVER_LOST

  クエリー中にサーバーへの接続が失われました。

- CR_UNKNOWN_ERROR

  不明なエラーが発生しました。

準備操作が成功しなかった (つまり、mysql_stmt_prepare() はゼロ以外を返す) 場合、mysql_stmt_error() を呼び出して、エラーメッセージを取得できます。

例

セクション23.7.11.10「mysql_stmt_execute()」の例を参照してください。

23.7.11.22 mysql_stmt_reset()

my_bool mysql_stmt_reset(MYSQL_STMT *stmt)

説明

クライアントおよびサーバーでプリペアドステートメントを準備後の状態にリセットします。それは、サーバー上のステートメント、mysql_stmt_send_long_data() を使用して送信されたデータ、バッファーされていない結果セット、および現在のエラーをリセットします。それは、バインディングまたは保存された結果セットはクリアしません。保存された結果セットは、プリペアドステートメントを実行した (またはそれを閉じた) ときにクリアされます。

別のクエリーでステートメントを再準備するには、mysql_stmt_prepare() を使用します。

戻り値

成功の場合はゼロ。エラーが発生した場合、ゼロ以外。

エラー

- CR_COMMANDS_OUT_OF_SYNC

  コマンドが正しくない順番で実行されました。

- CR_SERVER_GONE_ERROR

  MySQL サーバーが存在しなくなりました。

- CR_SERVER_LOST

  クエリー中にサーバーへの接続が失われました。

- CR_UNKNOWN_ERROR

  不明なエラーが発生しました。

23.7.11.23 mysql_stmt_result_metadata()

MYSQL_RES *mysql_stmt_result_metadata(MYSQL_STMT *stmt)

説明

mysql_stmt_prepare() に渡されたステートメントが結果セットを生成するステートメントである場合、mysql_stmt_result_metadata() は、フィールド数や個々のフィールド情報などのメタ情報を処理するために使用できる MYSQL_RES 構造へのポインタの形式で、結果セットメタデータを返します。この結果セットポインタは引数として、次のような結果セットメタデータを処理する任意のフィールドベースの API 関数に渡すことができます。

- mysql_num_fields()
- mysql_fetch_field()
- mysql_fetch_field_direct()
- mysql_fetch_fields()
- mysql_field_count()
- mysql_field_seek()
- mysql_field_tell()
- mysql_free_result()

結果セット構造は、その処理を終了したら解放してください。これは、mysql_free_result() にそれを渡すことによって実行できます。これは、mysql_store_result() への呼び出しから取得した結果セットを解放する方法に似ています。

mysql_stmt_result_metadata() によって返される結果セットにはメタデータだけが含まれます。それは行の結果を含んでいません。行は、mysql_stmt_fetch() によってステートメントハンドルを使用して取得します。

戻り値

MYSQL_RES結果構造。プリペアドクエリーにメタ情報が存在しない場合は NULL。

エラー

- CR_OUT_OF_MEMORY

  メモリー不足。

- CR_UNKNOWN_ERROR

  不明なエラーが発生しました。

例

セクション23.7.11.11「mysql_stmt_fetch()」の例を参照してください。

23.7.11.24 mysql_stmt_row_seek()

MYSQL_ROW_OFFSET mysql_stmt_row_seek(MYSQL_STMT *stmt, MYSQL_ROW_OFFSET offset)

説明

ステートメント結果セット内の任意の行に行カーソルを設定します。offset 値は、mysql_stmt_row_tell() または mysql_stmt_row_seek() から返される値になるべきである行オフセットです。この値は行番号ではありません。番号によって結果セット内の行にシークする場合は、代わりに mysql_stmt_data_seek() を使用します。

この関数は、結果セット構造にクエリーの結果全体を含める必要があるため、mysql_stmt_row_seek() は、mysql_stmt_store_result() と一緒にのみ使用できます。

戻り値

行カーソルの前の値。この値は、mysql_stmt_row_seek() への後続の呼び出しに渡すことができます。

エラー

なし。

23.7.11.25 mysql_stmt_row_tell()

MYSQL_ROW_OFFSET mysql_stmt_row_tell(MYSQL_STMT *stmt)

説明

最後の mysql_stmt_fetch() の行カーソルの現在の位置を返します。この値は、mysql_stmt_row_seek() への引数として使うことができます。

mysql_stmt_row_tell() は、mysql_stmt_store_result() のあとにのみ使用してください。

戻り値

行カーソルの現在のオフセット。

エラー

なし。

23.7.11.26 mysql_stmt_send_long_data()

my_bool mysql_stmt_send_long_data(MYSQL_STMT *stmt, unsigned int parameter_number, const char *data, unsigned long length)

説明

アプリケーションはパラメータデータをサーバーに個々に (または「まとめて」) 送信できます。この関数は mysql_stmt_bind_param() のあと、かつ mysql_stmt_execute() の前に呼び出します。それを複数回呼び出して、カラムの文字の一部またはバイナリデータ値を送信できますが、これは TEXT または BLOB データ型のいずれかである必要があります。

parameter_number はデータを関連付けるパラメータを示します。パラメータは 0 から番号付けされます。data は送信されるデータを格納するバッファーへのポインタで、length はバッファー内のバイト数を示します。

> **注記**
>
> 次の mysql_stmt_execute() 呼び出しは、最後の mysql_stmt_execute() または mysql_stmt_reset() から、mysql_stmt_send_long_data() によって使用されたすべてのパラメータのバインドバッファーを無視します。

送信されたデータをリセット/消去する場合、mysql_stmt_reset() によってそれを実行できます。セクション 23.7.11.22「mysql_stmt_reset()」を参照してください。

MySQL 5.6.3 以降、max_allowed_packet システム変数は、mysql_stmt_send_long_data() によって送信できるパラメータ値の最大サイズを制御します。

戻り値

成功の場合はゼロ。エラーが発生した場合、ゼロ以外。

エラー

- CR_INVALID_BUFFER_USE

  このパラメータは文字列またはバイナリ型は持ちません。

- CR_COMMANDS_OUT_OF_SYNC

  コマンドが正しくない順番で実行されました。

- CR_SERVER_GONE_ERROR

  MySQL サーバーが存在しなくなりました。

- CR_OUT_OF_MEMORY

  メモリー不足。

- CR_UNKNOWN_ERROR

  不明なエラーが発生しました。

例

次の例に、TEXT カラムのデータをまとめて送信する方法を示します。それはデータ値 'MySQL - The most popular Open Source database' を text_column カラムに挿入します。mysql 変数は有効な接続ハンドルであるとみなされます。

```
#define INSERT_QUERY "INSERT INTO \
                      test_long_data(text_column) VALUES(?)"

MYSQL_BIND bind[1];
long       length;

stmt = mysql_stmt_init(mysql);
if (!stmt)
{
  fprintf(stderr, " mysql_stmt_init(), out of memory\n");
  exit(0);
}
if (mysql_stmt_prepare(stmt, INSERT_QUERY, strlen(INSERT_QUERY)))
{
  fprintf(stderr, "\n mysql_stmt_prepare(), INSERT failed");
  fprintf(stderr, "\n %s", mysql_stmt_error(stmt));
  exit(0);
}
 memset(bind, 0, sizeof(bind));
 bind[0].buffer_type= MYSQL_TYPE_STRING;
 bind[0].length= &length;
 bind[0].is_null= 0;

/* Bind the buffers */
if (mysql_stmt_bind_param(stmt, bind))
{
  fprintf(stderr, "\n param bind failed");
  fprintf(stderr, "\n %s", mysql_stmt_error(stmt));
  exit(0);
}

 /* Supply data in chunks to server */
 if (mysql_stmt_send_long_data(stmt,0,"MySQL",5))
{
  fprintf(stderr, "\n send_long_data failed");
  fprintf(stderr, "\n %s", mysql_stmt_error(stmt));
```

```
  exit(0);
}

 /* Supply the next piece of data */
 if (mysql_stmt_send_long_data(stmt,0,
         " - The most popular Open Source database",40))
{
  fprintf(stderr, "\n send_long_data failed");
  fprintf(stderr, "\n %s", mysql_stmt_error(stmt));
  exit(0);
}

 /* Now, execute the query */
 if (mysql_stmt_execute(stmt))
{
  fprintf(stderr, "\n mysql_stmt_execute failed");
  fprintf(stderr, "\n %s", mysql_stmt_error(stmt));
  exit(0);
}
```

23.7.11.27 mysql_stmt_sqlstate()

const char *mysql_stmt_sqlstate(MYSQL_STMT *stmt)

説明

stmt によって指定されたステートメントに対して、mysql_stmt_sqlstate() は最近呼び出された成功または失敗した可能性のあるプリペアドステートメント API 関数の SQLSTATE エラーコードを格納する NULL 終端文字列を返します。エラーコードは 5 つの文字から構成されます。"00000" は「エラーなし」を意味します。値は ANSI SQL と ODBC によって規定されています。可能な値のリストについては、付録B「エラー、エラーコード、および一般的な問題」を参照してください。

すべての MySQL エラーがまだ SQLSTATE コードにマップされていません。値 "HY000" (一般エラー) はマップされていないエラー番号に使われます。

戻り値

SQLSTATE エラーコードを含む NULL 終端文字列。

23.7.11.28 mysql_stmt_store_result()

int mysql_stmt_store_result(MYSQL_STMT *stmt)

説明

結果セットは、SELECT、SHOW、DESCRIBE、および EXPLAIN などの SQL ステートメントの実行されたプリペアドステートメントに対し、mysql_stmt_execute() を呼び出して生成されます。デフォルトで、正常に実行されたプリペアドステートメントの結果セットは、クライアントでバッファーされず、mysql_stmt_fetch() はそれらをサーバーから一度に 1 つずつフェッチします。クライアントで完全な結果セットをバッファーさせるには、mysql_stmt_bind_result() によってデータバッファーをバインドしたあと、かつ mysql_stmt_fetch() を呼び出して行をフェッチする前に mysql_stmt_store_result() を呼び出します。(例については、セクション 23.7.11.11「mysql_stmt_fetch()」を参照してください。)

mysql_stmt_data_seek()、mysql_stmt_row_seek()、または mysql_stmt_row_tell() を呼び出さないかぎり、mysql_stmt_store_result() は結果セットの処理にオプションです。それらの関数ではシーク可能な結果セットが必要です。

結果セットを生成しない SQL ステートメントの実行後に、mysql_stmt_store_result() を呼び出す必要はありませんが、そうした場合に、損害を与えたり、目立ったパフォーマンスの問題を発生させたりすることはありません。mysql_stmt_result_metadata() が NULL を返すかどうかをチェックすることによって、ステートメントが結果セットを生成したかどうかを検出できます。詳細については、セクション 23.7.11.23「mysql_stmt_result_metadata()」を参照してください。

> **注記**
>
> MySQL はデフォルトで、mysql_stmt_store_result() ですべてのカラムの MYSQL_FIELD->max_length を計算しません。これを計算すると、mysql_stmt_store_result() がかなり遅くなることがあり、ほとんどのアプリケーションで max_length を必要としないためです。max_length を更新する必要がある場

合、mysql_stmt_attr_set(MYSQL_STMT, STMT_ATTR_UPDATE_MAX_LENGTH, &flag) を呼び出してこれを可能にできます。セクション23.7.11.3「mysql_stmt_attr_set()」を参照してください。

戻り値

成功の場合はゼロ。エラーが発生した場合、ゼロ以外。

エラー

- CR_COMMANDS_OUT_OF_SYNC

  コマンドが正しくない順番で実行されました。

- CR_OUT_OF_MEMORY

  メモリー不足。

- CR_SERVER_GONE_ERROR

  MySQL サーバーが存在しなくなりました。

- CR_SERVER_LOST

  サーバーへの接続がクエリー中に失われました。

- CR_UNKNOWN_ERROR

  不明なエラーが発生しました。

23.7.12 C API スレッド関数の説明

スレッドクライアントを作成するには、次のセクションで説明する関数を使用します。セクション23.7.4.2「C API スレッドクライアントプログラムの作成」も参照してください。

23.7.12.1 my_init()

void my_init(void)

説明

my_init() は MySQL が必要とするいくつかのグローバル変数を初期化します。それはこのスレッドに対し、mysql_thread_init() も呼び出します。

my_init() は、プログラムの MySQL ライブラリの使用の初期化フェーズの早期に呼び出される必要があります。ただし、my_init() は mysql_init()、mysql_library_init()、mysql_server_init()、および mysql_connect() によって自動的に呼び出されます。プログラムでほかの MySQL 呼び出しの前に、それらの関数のいずれかを確実に呼び出している場合は、my_init() を明示的に呼び出す必要はありません。

my_init() のプロトタイプにアクセスするには、プログラムにこれらのヘッダーファイルを含めてください。

```
#include <my_global.h>
#include <my_sys.h>
```

戻り値

なし。

23.7.12.2 mysql_thread_end()

void mysql_thread_end(void)

説明

この関数は、pthread_exit() を呼び出す前に、mysql_thread_init() によって割り当てられたメモリーを解放するために呼び出す必要があります。

mysql_thread_end() はクライアントライブラリによって自動的に呼び出されません。それはメモリーリークを避けるために明示的に呼び出す必要があります。

戻り値

なし。

23.7.12.3 mysql_thread_init()

```
my_bool mysql_thread_init(void)
```

説明

この関数は、スレッド固有変数を初期化するために、作成された各スレッド内の早期に呼び出す必要があります。ただし、必ずしもそれを明示的に呼び出す必要がないことがあります。mysql_thread_init() は my_init() によって自動的に呼び出され、それ自体も mysql_init()、mysql_library_init()、mysql_server_init()、および mysql_connect() によって自動的に呼び出されます。それらの関数のいずれかを呼び出すと、mysql_thread_init() が自動的に呼び出されます。

戻り値

成功の場合はゼロ。エラーが発生した場合、ゼロ以外。

23.7.12.4 mysql_thread_safe()

```
unsigned int mysql_thread_safe(void)
```

説明

この関数は、クライアントライブラリがスレッドセーフとしてコンパイルされるかどうかを示します。

戻り値

クライアントライブラリがスレッドセーフの場合は 1、それ以外の場合は 0。

23.7.13 C API 組み込みサーバー関数の説明

MySQL アプリケーションは、組み込みサーバーを使うように書くことができます。セクション23.6「組み込み MySQL サーバーライブラリ libmysqld」を参照してください。そのようなアプリケーションを書くには、-lmysqlclient フラグを使用して libmysqlclient クライアントライブラリに対してそれをリンクするのではなく、-lmysqld フラグを使用して libmysqld ライブラリに対してそれをリンクする必要があります。ただし、クライアントアプリケーションを書くか、組み込みサーバーを使うものを書くかに関係なく、ライブラリを初期化する呼び出しとファイナライズする呼び出しは同じです。ライブラリを初期化するには mysql_library_init() を呼び出し、その処理が終わったら、mysql_library_end() を呼び出します。セクション23.7.6「C API 関数の概要」を参照してください。

23.7.13.1 mysql_server_init()

```
int mysql_server_init(int argc, char **argv, char **groups)
```

説明

この関数は MySQL ライブラリを初期化します。これは、ほかのすべての MySQL 関数を呼び出す前に実行する必要があります。ただし、mysql_server_init() は非推奨であるため、代わりに、mysql_library_init() を呼び出してください。セクション23.7.7.40「mysql_library_init()」を参照してください。

戻り値

成功の場合はゼロ。エラーが発生した場合、ゼロ以外。

23.7.13.2 mysql_server_end()

```
void mysql_server_end(void)
```

説明

この関数は MySQL ライブラリをファイナライズします。これはライブラリの使用を終了したときに実行してください。ただし、mysql_server_end() は非推奨であるため、代わりに、mysql_library_end() を使用してください。セクション23.7.7.39「mysql_library_end()」を参照してください。

戻り値

なし。

23.7.14 C API クライアントプラグイン関数

このセクションでは、クライアント側プラグイン API に使用される関数について説明します。それらはクライアントプラグインの管理を可能にします。これらの関数によって使用される st_mysql_client_plugin 構造の説明については、クライアントプラグインディスクリプタを参照してください。

クライアントプログラムでこのセクションの関数を呼び出す必要がある可能性はないと考えられます。たとえば、認証プラグインの使用をサポートするクライアントは通常 mysql_options() を呼び出して、MYSQL_DEFAULT_AUTH および MYSQL_PLUGIN_DIR オプションを設定することによって、プラグインをロードさせます。

```
char *plugin_dir = "path_to_plugin_dir";
char *default_auth = "plugin_name";

/* ... process command-line options ... */

mysql_options(&mysql, MYSQL_PLUGIN_DIR, plugin_dir);
mysql_options(&mysql, MYSQL_DEFAULT_AUTH, default_auth);
```

一般にプログラムは、ユーザーがデフォルト値をオーバーライドできるようにする --plugin-dir および --default-auth オプションも受け付けます。

23.7.14.1 mysql_client_find_plugin()

struct st_mysql_client_plugin *mysql_client_find_plugin(MYSQL *mysql, const char *name, int type)

説明

ロードしたプラグインへのポインタを返し、必要に応じて、最初にプラグインをロードします。型が無効か、プラグインが見つからないかロードできない場合はエラーが発生します。

パラメータを次のように指定します。

- mysql: MYSQL 構造へのポインタ。プラグイン API は MySQL サーバーへの接続を必要としませんが、この構造は正しく初期化する必要があります。この構造は、接続関連情報を取得するために使用されます。
- name: プラグイン名。
- type: プラグインの型。

戻り値

成功のためのプラグインへのポインタ。エラーが発生した場合は NULL。

エラー

エラーをチェックするには、mysql_error() または mysql_errno() 関数を呼び出します。セクション23.7.7.15「mysql_error()」およびセクション23.7.7.14「mysql_errno()」を参照してください。

例

```
MYSQL mysql;
struct st_mysql_client_plugin *p;

if ((p = mysql_client_find_plugin(&mysql, "myplugin",
                                  MYSQL_CLIENT_AUTHENTICATION_PLUGIN, 0)))
{
  printf("Plugin version: %d.%d.%d\n", p->version[0], p->version[1], p->version[2]);
}
```

23.7.14.2 mysql_client_register_plugin()

```
struct st_mysql_client_plugin *mysql_client_register_plugin(MYSQL *mysql, struct st_mysql_client_plugin *plugin)
```

説明

ロードされたプラグインのリストにプラグイン構造を追加します。プラグインがすでにロードされている場合、エラーが発生します。

パラメータを次のように指定します。

- mysql: MYSQL 構造へのポインタ。プラグイン API は MySQL サーバーへの接続を必要としませんが、この構造は正しく初期化する必要があります。この構造は、接続関連情報を取得するために使用されます。
- plugin: プラグイン構造へのポインタ。

戻り値

成功のためのプラグインへのポインタ。エラーが発生した場合は NULL。

エラー

エラーをチェックするには、mysql_error() または mysql_errno() 関数を呼び出します。セクション 23.7.7.15「mysql_error()」およびセクション23.7.7.14「mysql_errno()」を参照してください。

23.7.14.3 mysql_load_plugin()

```
struct st_mysql_client_plugin *mysql_load_plugin(MYSQL *mysql, const char *name, int type, int argc, ...)
```

説明

名前と型で指定された MySQL クライアントプラグインをロードします。型が無効か、プラグインをロードできない場合はエラーが発生します。

同じ型の複数のプラグインをロードすることはできません。すでにロードされている型のプラグインをロードしようとすると、エラーが発生します。

パラメータを次のように指定します。

- mysql: MYSQL 構造へのポインタ。プラグイン API は MySQL サーバーへの接続を必要としませんが、この構造は正しく初期化する必要があります。この構造は、接続関連情報を取得するために使用されます。
- name: ロードするプラグインの名前。
- type: ロードするプラグインの型、または型チェックを無効にする場合は −1。型が −1 でない場合、型に一致するプラグインのみがロードに考慮されます。
- argc: 後続の引数の数 (何もない場合は 0)。後続の引数の解釈はプラグインの型によって異なります。

プラグインをロードさせるもう 1 つの方法は、LIBMYSQL_PLUGINS 環境変数にセミコロン区切りのプラグイン名のリストを設定することです。例:

```
shell> export LIBMYSQL_PLUGINS="myplugin1;myplugin2"
```

クライアントプログラムが mysql_library_init() を呼び出したときに、LIBMYSQL_PLUGINS によって指定されたプラグインがロードされます。これらのプラグインのロードで問題が発生した場合、エラーは報告されません。

MySQL 5.6.10 以降、LIBMYSQL_PLUGIN_DIR 環境変数には、クライアントプラグインを探すディレクトリのパス名を設定できます。この変数は 2 つの方法で使用されます。

- クライアントプラグインのプリロード中、--plugin-dir オプションの値は使用できないため、プラグインが、組み込まれたデフォルトのディレクトリに存在しない場合、クライアントプラグインのロードが失敗します。プラグインがほかの場所に存在する場合は、LIBMYSQL_PLUGIN_DIR 環境変数を正しいディレクトリに設定し、プラグインのプリロードが成功できるようにします。
- 明示的なクライアントプラグインのロードで、mysql_load_plugin() および mysql_load_plugin_v() C API 関数は LIBMYSQL_PLUGIN_DIR 値が存在し、--plugin-dir オプションが指定されていない場合は、その値を使用します。--plugin-dir が指定されている場合は、mysql_load_plugin() と mysql_load_plugin_v() は LIBMYSQL_PLUGIN_DIR を無視します。

戻り値

プラグインが正常にロードされた場合にプラグインへのポインタ。エラーが発生した場合は NULL。

エラー

エラーをチェックするには、mysql_error() または mysql_errno() 関数を呼び出します。セクション23.7.7.15「mysql_error()」およびセクション23.7.7.14「mysql_errno()」を参照してください。

例

```
MYSQL mysql;

if(!mysql_load_plugin(&mysql, "myplugin",
                      MYSQL_CLIENT_AUTHENTICATION_PLUGIN, 0))
{
   fprintf(stderr, "Error: %s\n", mysql_error(&mysql));
   exit(-1);
}
```

関連項目

セクション23.7.14.3「mysql_load_plugin()」、セクション23.7.7.15「mysql_error()」、セクション23.7.7.14「mysql_errno()」 も参照してください。

23.7.14.4 mysql_load_plugin_v()

struct st_mysql_client_plugin *mysql_load_plugin_v(MYSQL *mysql, const char *name, int type, int argc, va_list args)

説明

この関数は mysql_load_plugin() と同等ですが、それはパラメータの変数リストの代わりに va_list を受け付けます。

関連項目

セクション23.7.14.3「mysql_load_plugin()」も参照してください。

23.7.14.5 mysql_plugin_options()

int mysql_plugin_options(struct st_mysql_client_plugin *plugin, const char *option, const void *value)

説明

オプションの型と値をプラグインに渡します。この関数を複数回呼び出して、複数のオプションを設定できます。プラグインにオプションハンドラがない場合、エラーが発生します。

パラメータを次のように指定します。

- plugin: プラグイン構造へのポインタ。
- option: 設定されるオプション。
- value: オプション値へのポインタ。

戻り値

成功した場合はゼロ、エラーが発生した場合は 1。プラグインにオプションハンドラがある場合、そのハンドラも成功した場合にゼロを返し、エラーが発生した場合は 1 を返します。

23.7.15 C API を使用する場合の一般的な質問と問題

23.7.15.1 mysql_query() が成功を返したあとに mysql_store_result() が NULL を返すことがあるのはなぜか

mysql_query() への成功した呼び出しに続いて、mysql_store_result() が NULL を返すことがあります。これが発生した場合、それは次の状況のいずれかが発生したことを意味します。

- malloc() の障害が発生しました (たとえば、結果セットが大きすぎた場合)。
- データを読み取ることができませんでした (接続でエラーが発生しました)。
- クエリーがデータを返しませんでした (たとえば、それが INSERT、UPDATE、またはDELETE でした)。

mysql_field_count() を呼び出して、ステートメントが空でない結果を生成したかどうかを常にチェックできます。mysql_field_count() がゼロを返し、結果が空で、最後のクエリーが値を返さないステートメント (INSERT または DELETE など) でした。mysql_field_count() がゼロ以外を返す場合、ステートメントは空でない結果を生成しています。例については、mysql_field_count() 関数の説明を参照してください。

mysql_error() または mysql_errno() を呼び出すことによって、エラーがないかテストできます。

23.7.15.2 クエリーからどんな結果を得ることができるか

クエリーによって返される結果セットに加えて、次の情報も取得できます。

- mysql_affected_rows() は、INSERT、UPDATE、または DELETE の実行中に、最後のクエリーによって影響を受けた行の数を返します。

  高速の再作成のために、TRUNCATE TABLE を使用します。

- mysql_num_rows() は結果セット内に行数を返します。mysql_store_result() では、mysql_store_result() が戻ったらすぐに、mysql_num_rows() を呼び出すことができます。mysql_use_result() では、mysql_fetch_row() によってすべての行をフェッチしたあとにのみ、mysql_num_rows() を呼び出すことができます。
- mysql_insert_id() は、AUTO_INCREMENT インデックスでテーブル内に行を挿入した最後のクエリーによって生成された ID を返します。セクション23.7.7.37「mysql_insert_id()」を参照してください。
- いくつかのクエリー (LOAD DATA INFILE ...、INSERT INTO ... SELECT ...、UPDATE) は追加の情報を返します。結果は mysql_info() によって返されます。mysql_info() が返す文字列の形式については、その説明を参照してください。追加の情報がない場合は、mysql_info() は NULL ポインタを返します。

23.7.15.3 最後に挿入された行の一意の ID を取得する方法

AUTO_INCREMENT カラムを含むテーブルにレコードを挿入する場合、mysql_insert_id() 関数を呼び出すことによって、そのカラム内に格納された値を取得できます。

C アプリケーションから、次のコード (ステートメントが成功したことをチェックしているものとみなす) を実行することによって、値が AUTO_INCREMENT カラム内に格納されたかどうかをチェックできます。それは、クエリーが AUTO_INCREMENT インデックスによる INSERT であったかどうかを判断します。

```
if ((result = mysql_store_result(&mysql)) == 0 &&
   mysql_field_count(&mysql) == 0 &&
   mysql_insert_id(&mysql) != 0)
{
   used_id = mysql_insert_id(&mysql);
}
```

新しい AUTO_INCREMENT 値が生成されたら、mysql_query() で、SELECT LAST_INSERT_ID() ステートメントを実行し、ステートメントによって返された結果セットから値を取得することによって、それを取得することもできます。

複数の値を挿入すると、自動的に増分された最後の値が返されます。

LAST_INSERT_ID() では、最近生成された ID が接続ごとにサーバーに維持されます。それはほかのクライアントによって変更されません。別の AUTO_INCREMENT カラムを非マジック値 (つまり、NULL でなく、0 でない値) で更新した場合でも、それは変更されません。LAST_INSERT_ID() カラムと AUTO_INCREMENT カラムを複数のクライアントから同時に使うことは完全に有効です。各クライアントは、そのクライアントが実行した最後のステートメントの最後に挿入された ID を受け取ります。

1 つのテーブルに生成された ID を使用して、それを 2 番目のテーブルに挿入する場合、このような SQL ステートメントを使うことができます。

```
INSERT INTO foo (auto,text)
   VALUES(NULL,'text');        # generate ID by inserting NULL
INSERT INTO foo2 (id,text)
   VALUES(LAST_INSERT_ID(),'text');  # use ID in second table
```

mysql_insert_id() は AUTO_INCREMENT カラムに格納された値が、NULL または 0 を格納することによって自動的に生成されているか、または明示的な値として指定されたかどうかに関係なく、その値を返しま

す。LAST_INSERT_ID() は自動的に生成された AUTO_INCREMENT 値のみを返します。NULL または 0 以外の明示的な値を格納する場合、それは、LAST_INSERT_ID() によって返される値に影響を与えません。

AUTO_INCREMENT カラム内の最後の ID を取得する詳細については:

- SQL ステートメント内で使用できる LAST_INSERT_ID() に関する情報については、セクション12.14「情報関数」を参照してください。
- C API 内から使用する関数 mysql_insert_id() については、セクション23.7.7.37「mysql_insert_id()」を参照してください。
- Connector/J を使用する場合の自動インクリメントされた値の取得については、Retrieving AUTO_INCREMENT Column Values through JDBCを参照してください。
- Connector/ODBC を使用する場合の自動インクリメントされた値の取得については、Obtaining Auto-Increment Valuesを参照してください。

23.7.16 自動再接続動作の制御

実行されるステートメントをサーバーに送信しようとしたときに、MySQL クライアントライブラリが接続が停止していることを検出した場合、それはサーバーへの自動再接続を実行できます。自動再接続が有効にされている場合、ライブラリは 1 回サーバーに再接続し、ステートメントを再度送信しようとします。

MySQL 5.6 では、自動再接続はデフォルトで無効にされています。

アプリケーションで、接続が切断されたことを知ることが重要な場合 (状態情報の損失に合わせて終了したり、アクションをとったりできるように)、自動再接続が無効にされるようにしてください。これを確実にするには、MYSQL_OPT_RECONNECT オプションを使用して mysql_options() を呼び出します。

```
my_bool reconnect = 0;
mysql_options(&mysql, MYSQL_OPT_RECONNECT, &reconnect);
```

接続が停止した場合、mysql_ping() の効果は自動再接続の状態によって異なります。自動再接続が有効にされている場合、mysql_ping() は再接続を実行します。そうでない場合、それはエラーを返します。

一部のクライアントプログラムでは、自動再接続を制御する機能を提供できます。たとえば、mysql はデフォルトで再接続しますが、--skip-reconnect オプションを使用して、この動作を抑止できます。

自動再接続が行われた場合 (たとえば、mysql_ping() の呼び出しの結果として)、その明示的な兆候はありません。再接続をチェックするには、mysql_ping() を呼び出す前に mysql_thread_id() を呼び出して、元の接続識別子を取得してから、再度 mysql_thread_id() を呼び出して、識別子が変更されているかどうかを確認します。

自動再接続は、独自の再接続コードを実装する必要がないため、便利な場合がありますが、再接続が行われる場合、サーバー側で接続状態のいくつかの側面がリセットされ、アプリケーションに通知されません。

接続関連の状態は、次のように影響を受けます。

- アクティブなトランザクションがすべてロールバックされ、自動コミットモードがリセットされます。
- すべてのテーブルロックが解除されます。
- すべての TEMPORARY テーブルが閉じられ (さらに削除され) ます。
- セッションシステム変数は、SET NAMES などのステートメントによって暗黙的に設定されるシステム変数を含む、対応するグローバルシステム変数の値に再初期化されます。
- ユーザー変数設定が失われます。
- プリペアドステートメントがリリースされます。
- HANDLER 変数が閉じられます。
- LAST_INSERT_ID() の値が 0 にリセットされます。
- GET_LOCK() によって取得されたロックが解放されます。
- 接続スレッドインストゥルメンテーションを判断するパフォーマンススキーマ threads テーブル行とクライアントの関連付けが失われます。クライアントが切断後に再接続した場合、セッションは threads テーブル内の新しい行に関連付けられるため、スレッドモニタリング状態が異なることがあります。セクション22.9.10.3「スレッドテーブル」を参照してください。

接続が切断した場合、サーバーでクライアントが接続されなくなったことを検出していない場合に、サーバー側でその接続に関連付けられているセッションが引き続き実行する可能性があります。この場合、元の接続によって保持されたロックはまだそのセッションに属しているため、mysql_kill() を呼び出して、それを強制終了する必要がある可能性があります。

23.7.17 複数ステートメント実行の C API サポート

デフォルトで、mysql_query() と mysql_real_query() はそれらのステートメント文字列引数を、実行すべき単一のステートメントとして解釈し、ユーザーはステートメントが結果セット (行のセット、SELECT の場合) を生成するか、または影響を受ける行カウント (INSERT、UPDATE などの場合) を生成するかに従って、結果を処理します。

MySQL 5.6 はセミコロン (「;」) 文字によって区切られた複数のステートメントを格納する文字列の実行もサポートします。この機能は、mysql_real_connect() によってサーバーに接続するとき、または mysql_set_server_option() の呼び出しによる接続後のいずれかに指定される特別なオプションによって有効化されます。

複数ステートメント文字列を実行すると、複数の結果セットまたは行カウントインジケータを生成できます。これらの結果の処理には、単一ステートメントの場合と異なるアプローチが必要です。最初のステートメントからの結果の処理後、それ以上の結果が存在するかどうかをチェックし、存在する場合は、それらを順番に処理する必要があります。複数結果の処理をサポートするため、C API には、mysql_more_results() 関数と mysql_next_result() 関数が含まれています。これらの関数は、それ以上の結果があるかぎり反復するループの最後で使用します。結果をこのように処理できないと、サーバーへの接続が切断される可能性があります。

複数結果の処理は、ストアドプロシージャーに対して CALL ステートメントを実行する場合にも必要です。ストアドプロシージャーの結果にはこれらの特性があります。

- プロシージャー内のステートメントは結果セットを生成することがあります (たとえば、それが SELECT ステートメントを実行する場合など)。これらの結果セットは、プロシージャーの実行とともに、それらが生成された順番で返されます。

  一般に、呼び出し元はプロシージャーが返す結果セットの数を知ることができません。プロシージャーの実行は、呼び出しごとに実行パスが異なるループや条件ステートメントによって異なることがあります。そのため、複数の結果を取得するように準備しておく必要があります。

- プロシージャーからの最終結果は、結果セットを含まないステータス結果です。このステータスはプロシージャーが成功したか、エラーが発生したかを示します。

複数ステートメントおよび結果機能は、mysql_query() または mysql_real_query() と一緒にのみ使用できます。それらはプリペアドステートメントインタフェースと一緒に使用できません。プリペアドステートメントハンドルは単一のステートメントを含む文字列のみを操作するように定義されます。セクション23.7.8「C API プリペアドステートメント」を参照してください。

複数ステートメントの実行と結果の処理を有効にするには、次のオプションを使うことができます。

- mysql_real_connect() 関数には 2 つのオプション値が関連する flags 引数があります。

  - CLIENT_MULTI_RESULTS により、クライアントプログラムは複数の結果を処理できます。結果セットを生成するストアドプロシージャーに対して、CALL ステートメントを実行する場合、このオプションを有効にする必要があります。そうしないと、そのようなプロシージャーはエラー「エラー 1312 (0A000): PROCEDURE proc_name は指定されたコンテキストで結果セットを返すことができません」を生成します。MySQL 5.6 では、CLIENT_MULTI_RESULTS はデフォルトで有効にされています。

  - CLIENT_MULTI_STATEMENTS により、mysql_query() および mysql_real_query() はセミコロンで区切られた複数のステートメントを含むステートメント文字列を実行できます。このオプションにより、CLIENT_MULTI_RESULTS も暗黙的に有効になるため、mysql_real_connect() への CLIENT_MULTI_STATEMENTS の flags 引数は、CLIENT_MULTI_STATEMENTS | CLIENT_MULTI_RESULTS の引数と同等になります。つまり、CLIENT_MULTI_STATEMENTS は複数ステートメントの実行とすべての複数結果の処理を有効にするのに十分です。

- サーバーへの接続が確立されたあと、mysql_set_server_option() 関数を使用して、それを MYSQL_OPTION_MULTI_STATEMENTS_ON または MYSQL_OPTION_MULTI_STATEMENTS_OFF の引数に渡すことによって、複数ステートメントの実行を有効または無効にできます。この関数によって複数ステートメントの実行を有効にすると、複数ステートメント文字列の「簡単な」結果の処理も有効になります。そこでは、各ステートメントが単一の結果を生成しますが、結果セットを生成するストアドプロシージャーの処理を許可するには十分でありません。

次の手順に、複数ステートメントの処理の推奨される戦略の概要を示します。

1. CLIENT_MULTI_STATEMENTS を mysql_real_connect() に渡して、複数ステートメントの実行と複数結果の処理を完全に有効にします。
2. mysql_query() または mysql_real_query() を呼び出し、それが成功したことを確認したあとに、ステートメント結果を処理するループに入ります。
3. ループの反復ごとに、現在のステートメント結果を処理し、結果セットまたは影響を受けた行カウントを取得します。エラーが発生したら、ループを終了します。
4. ループの終わりに、mysql_next_result() を呼び出して、ほかの結果が存在するかどうかをチェックし、その場合は、取得を開始します。それ以上の結果がなくなったら、ループを終了します。

先述の戦略の 1 つの可能な実装を次に示します。ループの最終部分は、mysql_next_result() がゼロ以外を返すかどうかの簡単なテストに単純化できます。示されているコードは、それ以上の結果がないことと エラーを区別し、それによって、後者の発生でメッセージを出力させることができます。

```
/* connect to server with the CLIENT_MULTI_STATEMENTS option */
if (mysql_real_connect (mysql, host_name, user_name, password,
    db_name, port_num, socket_name, CLIENT_MULTI_STATEMENTS) == NULL)
{
  printf("mysql_real_connect() failed\n");
  mysql_close(mysql);
  exit(1);
}

/* execute multiple statements */
status = mysql_query(mysql,
                     "DROP TABLE IF EXISTS test_table;\
                      CREATE TABLE test_table(id INT);\
                      INSERT INTO test_table VALUES(10);\
                      UPDATE test_table SET id=20 WHERE id=10;\
                      SELECT * FROM test_table;\
                      DROP TABLE test_table");
if (status)
{
  printf("Could not execute statement(s)");
  mysql_close(mysql);
  exit(0);
}

/* process each statement result */
do {
  /* did current statement return data? */
  result = mysql_store_result(mysql);
  if (result)
  {
    /* yes; process rows and free the result set */
    process_result_set(mysql, result);
    mysql_free_result(result);
  }
  else          /* no result set or error */
  {
    if (mysql_field_count(mysql) == 0)
    {
      printf("%lld rows affected\n",
             mysql_affected_rows(mysql));
    }
    else  /* some error occurred */
    {
      printf("Could not retrieve result set\n");
      break;
    }
  }
  /* more results? -1 = no, >0 = error, 0 = yes (keep looping) */
  if ((status = mysql_next_result(mysql)) > 0)
    printf("Could not execute statement\n");
} while (status == 0);

mysql_close(mysql);
```

23.7.18 C API プリペアドステートメントの問題

次に、プリペアドステートメントの現在既知の問題のリストを示します。

- TIME、TIMESTAMP、および DATETIME は秒の部分 (DATE_FORMAT() からなど) をサポートしていません。

- 整数から文字列への変換時に、ZEROFILL は、MySQL サーバーが先頭のゼロを出力しない特定の場合に、プリペアドステートメントを使用できます。(たとえば、MIN(number-with-zerofill) によって)。
- クライアントで浮動小数点数値を文字列に変換すると、変換された値の右端の桁が、元の値のそれとわずかに異なることがあります。
- プリペアドステートメントは、セクション8.9.3.1「クエリーキャッシュの動作」に説明する状況でクエリーキャッシュを使用します。
- プリペアドステートメントは、複数ステートメント (つまり、単一の文字列内にある 「;」 文字で区切られた複数のステートメント) をサポートしません。
- プリペアド CALL ステートメントの機能については、セクション23.7.20「C API のプリペアド CALL ステートメントのサポート」で説明しています。

23.7.19 C API プリペアドステートメントの日時値の処理

バイナリ (プリペアドステートメント) プロトコルにより、MYSQL_TIME 構造を使用して、日時値 (DATE、TIME、DATETIME、および TIMESTAMP) を送受信できます。この構造のメンバーについては、セクション23.7.9「C API プリペアドステートメントデータ構造」で説明しています。

時間データ値を送信するには、mysql_stmt_prepare() を使用して、プリペアドステートメントを作成します。その後、mysql_stmt_execute() を呼び出して、ステートメントを実行する前に、次の手順を使用して、各時間パラメータを設定します。

1. データ値に関連付けられている MYSQL_BIND 構造で、buffer_type メンバーを、送信する時間値の種類を示す型に設定します。DATE、TIME、DATETIME、または TIMESTAMP 値で、buffer_type を MYSQL_TYPE_DATE、MYSQL_TYPE_TIME、MYSQL_TYPE_DATETIME、または MYSQL_TYPE_TIMESTAMP にそれぞれ設定します。
2. MYSQL_BIND 構造の buffer メンバーを、時間値を渡す MYSQL_TIME 構造のアドレスに設定します。
3. 渡す時間値の型に適切な MYSQL_TIME 構造のメンバーを入力します。

mysql_stmt_bind_param() を使用して、パラメータデータをステートメントにバインドします。これにより、mysql_stmt_execute() を呼び出すことができます。

時間値を取得する場合、buffer_type メンバーを、受け取ることを期待する値の型に設定し、buffer メンバーを、戻り値を配置させる MYSQL_TIME 構造のアドレスに設定することを除いて、手順は同じです。mysql_stmt_bind_result() を使用して、mysql_stmt_execute() を呼び出したあと、かつ結果をフェッチする前にステートメントにバッファーをバインドします。

これは DATE、TIME、および TIMESTAMP データを挿入する簡単な例です。mysql 変数は有効な接続ハンドルであるとみなされます。

```
MYSQL_TIME  ts;
MYSQL_BIND  bind[3];
MYSQL_STMT  *stmt;

strmov(query, "INSERT INTO test_table(date_field, time_field, \
                      timestamp_field) VALUES(?,?,?");

stmt = mysql_stmt_init(mysql);
if (!stmt)
{
  fprintf(stderr, " mysql_stmt_init(), out of memory\n");
  exit(0);
}
if (mysql_stmt_prepare(mysql, query, strlen(query)))
{
  fprintf(stderr, "\n mysql_stmt_prepare(), INSERT failed");
  fprintf(stderr, "\n %s", mysql_stmt_error(stmt));
  exit(0);
}

/* set up input buffers for all 3 parameters */
bind[0].buffer_type= MYSQL_TYPE_DATE;
bind[0].buffer= (char *)&ts;
bind[0].is_null= 0;
bind[0].length= 0;
...
bind[1]= bind[2]= bind[0];
...
```

```
mysql_stmt_bind_param(stmt, bind);

/* supply the data to be sent in the ts structure */
ts.year= 2002;
ts.month= 02;
ts.day= 03;

ts.hour= 10;
ts.minute= 45;
ts.second= 20;

mysql_stmt_execute(stmt);
..
```

23.7.20 C API のプリペアド CALL ステートメントのサポート

このセクションでは、CALL ステートメントを使用して実行されるストアドプロシージャーに対する C API のプリペアドステートメントのサポートについて説明します。

MySQL 5.6 では、プリペアド CALL ステートメントを使用して実行されるストアドプロシージャーを次のように使用できます。

- ストアドプロシージャーは任意の数の結果セットを生成できます。カラムの数とカラムのデータ型はすべての結果セットで同じにする必要はありません。
- OUT および INOUT パラメータの最終値は、プロシージャーが戻ったあとに、呼び出し元のアプリケーションで使用できます。これらのパラメータは、プロシージャー自体によって生成される結果セットに続いて、特別な単一行結果セットとして返されます。行には、OUT および INOUT パラメータがプロシージャーパラメータリストで宣言されている順番で含まれます。

次の説明に、プリペアドステートメントに対して、C API からこれらの機能を使用する方法を示します。PREPARE および EXECUTE ステートメントからプリペアド CALL ステートメントを使用するには、セクション13.2.1「CALL 構文」を参照してください。

アプリケーションが 5.5.3 より古い MySQL のバージョンが使用されたコンテキストでコンパイルまたは実行されている可能性がある場合、複数の結果セットのプリペアド CALL 機能および OUT または INOUT パラメータを使用できない可能性があります。

- クライアント側で、ライブラリが MySQL 5.5.3 以上のものでないと、アプリケーションはコンパイルされません (そのバージョンで導入された API 関数およびシンボルが存在しなくなります)。
- サーバーが十分に新しいことを実行時に確認するには、クライアントでこのテストを使用できます。

```
if (mysql_get_server_version(mysql) < 50503)
{
  fprintf(stderr,
          "Server does not support required CALL capabilities\n");
  mysql_close(mysql);
  exit (1);
}
```

プリペアド CALL ステートメントを実行するアプリケーションは、結果をフェッチし、次に mysql_stmt_next_result() を呼び出して、それ以上の結果があるかどうかを判断するループを使用してください。結果は、ストアドプロシージャーによって生成された結果セットと、それに続くプロシージャーが正常に終了したかどうかを示す最終ステータス値から構成されます。

プロシージャーに OUT または INOUT パラメータがある場合、最終ステータス値の前の結果セットにそれらの値が格納されます。結果セットにパラメータ値が格納されているかどうかを判断するには、MYSQL 接続ハンドラの server_status メンバーに、SERVER_PS_OUT_PARAMS ビットが設定されているかどうかをテストします。

```
mysql->server_status & SERVER_PS_OUT_PARAMS
```

次の例では、プリペアド CALL ステートメントを使用して、複数の結果セットを生成して、OUT および INOUT パラメータを使用して、呼び出し元にパラメータ値を返すストアドプロシージャーを実行しています。プロシージャーは 3 つすべての型 (IN、OUT、INOUT) のパラメータをとり、それらの初期値を表示して、新しい値を割り当て、更新された値を表示して戻ります。そのため、プロシージャーからの予期される戻り情報は、複数の結果セットと最終ステータスから構成されます。

- 初期パラメータ値 (10、NULL、30) を表示する SELECT からの 1 つの結果セット。(OUT パラメータには呼び出し元によって値が割り当てられますが、この割り当ては無効にすることが期待されます。OUT パラメータは、プロシージャー内で値を割り当てるまで、プロシージャー内で NULL とみなされます。)

- 変更されたパラメータ値 (100、200、300) を表示する SELECT からの 1 つの結果セット。
- 最終 OUT および INOUT パラメータ値 (200、300) を格納する 1 つの結果セット。
- 最終ステータスパケット。

プロシージャーを実行するコード:

```
MYSQL_STMT *stmt;
MYSQL_BIND ps_params[3];  /* input parameter buffers */
int        int_data[3];   /* input/output values */
my_bool    is_null[3];    /* output value nullability */
int        status;

/* set up stored procedure */
status = mysql_query(mysql, "DROP PROCEDURE IF EXISTS p1");
test_error(mysql, status);

status = mysql_query(mysql,
 "CREATE PROCEDURE p1("
 "  IN p_in INT, "
 "  OUT p_out INT, "
 "  INOUT p_inout INT) "
 "BEGIN "
 "  SELECT p_in, p_out, p_inout; "
 "  SET p_in = 100, p_out = 200, p_inout = 300; "
 "  SELECT p_in, p_out, p_inout; "
 "END");
test_error(mysql, status);

/* initialize and prepare CALL statement with parameter placeholders */
stmt = mysql_stmt_init(mysql);
if (!stmt)
{
  printf("Could not initialize statement\n");
  exit(1);
}
status = mysql_stmt_prepare(stmt, "CALL p1(?, ?, ?)", 16);
test_stmt_error(stmt, status);

/* initialize parameters: p_in, p_out, p_inout (all INT) */
memset(ps_params, 0, sizeof (ps_params));

ps_params[0].buffer_type = MYSQL_TYPE_LONG;
ps_params[0].buffer = (char *) &int_data[0];
ps_params[0].length = 0;
ps_params[0].is_null = 0;

ps_params[1].buffer_type = MYSQL_TYPE_LONG;
ps_params[1].buffer = (char *) &int_data[1];
ps_params[1].length = 0;
ps_params[1].is_null = 0;

ps_params[2].buffer_type = MYSQL_TYPE_LONG;
ps_params[2].buffer = (char *) &int_data[2];
ps_params[2].length = 0;
ps_params[2].is_null = 0;

/* bind parameters */
status = mysql_stmt_bind_param(stmt, ps_params);
test_stmt_error(stmt, status);

/* assign values to parameters and execute statement */
int_data[0]= 10;  /* p_in */
int_data[1]= 20;  /* p_out */
int_data[2]= 30;  /* p_inout */

status = mysql_stmt_execute(stmt);
test_stmt_error(stmt, status);

/* process results until there are no more */
do {
  int i;
  int num_fields;       /* number of columns in result */
  MYSQL_FIELD *fields;  /* for result set metadata */
  MYSQL_BIND *rs_bind;  /* for output buffers */

  /* the column count is > 0 if there is a result set */
  /* 0 if the result is only the final status packet */
  num_fields = mysql_stmt_field_count(stmt);
```

```
if (num_fields > 0)
{
  /* there is a result set to fetch */
  printf("Number of columns in result: %d\n", (int) num_fields);

  /* what kind of result set is this? */
  printf("Data: ");
  if(mysql->server_status & SERVER_PS_OUT_PARAMS)
    printf("this result set contains OUT/INOUT parameters\n");
  else
    printf("this result set is produced by the procedure\n");

  MYSQL_RES *rs_metadata = mysql_stmt_result_metadata(stmt);
  test_stmt_error(stmt, rs_metadata == NULL);

  fields = mysql_fetch_fields(rs_metadata);

  rs_bind = (MYSQL_BIND *) malloc(sizeof (MYSQL_BIND) * num_fields);
  if (!rs_bind)
  {
    printf("Cannot allocate output buffers\n");
    exit(1);
  }
  memset(rs_bind, 0, sizeof (MYSQL_BIND) * num_fields);

  /* set up and bind result set output buffers */
  for (i = 0; i < num_fields; ++i)
  {
    rs_bind[i].buffer_type = fields[i].type;
    rs_bind[i].is_null = &is_null[i];

    switch (fields[i].type)
    {
      case MYSQL_TYPE_LONG:
        rs_bind[i].buffer = (char *) &(int_data[i]);
        rs_bind[i].buffer_length = sizeof (int_data);
        break;

      default:
        fprintf(stderr, "ERROR: unexpected type: %d.\n", fields[i].type);
        exit(1);
    }
  }

  status = mysql_stmt_bind_result(stmt, rs_bind);
  test_stmt_error(stmt, status);

  /* fetch and display result set rows */
  while (1)
  {
    status = mysql_stmt_fetch(stmt);

    if (status == 1 || status == MYSQL_NO_DATA)
      break;

    for (i = 0; i < num_fields; ++i)
    {
      switch (rs_bind[i].buffer_type)
      {
        case MYSQL_TYPE_LONG:
          if (*rs_bind[i].is_null)
            printf(" val[%d] = NULL;", i);
          else
            printf(" val[%d] = %ld;",
                   i, (long) *((int *) rs_bind[i].buffer));
          break;

        default:
          printf("  unexpected type (%d)\n",
            rs_bind[i].buffer_type);
      }
    }
    printf("\n");
  }

  mysql_free_result(rs_metadata); /* free metadata */
  free(rs_bind);                  /* free output buffers */
}
else
```

```
 {
   /* no columns = final status packet */
   printf("End of procedure output\n");
 }

 /* more results? -1 = no, >0 = error, 0 = yes (keep looking) */
 status = mysql_stmt_next_result(stmt);
 if (status > 0)
   test_stmt_error(stmt, status);
} while (status == 0);

mysql_stmt_close(stmt);
```

プロシージャーを実行すると次の出力が生成されます。

```
Number of columns in result: 3
Data: this result set is produced by the procedure
 val[0] = 10; val[1] = NULL; val[2] = 30;
Number of columns in result: 3
Data: this result set is produced by the procedure
 val[0] = 100; val[1] = 200; val[2] = 300;
Number of columns in result: 2
Data: this result set contains OUT/INOUT parameters
 val[0] = 200; val[1] = 300;
End of procedure output
```

このコードでは、2 つのユーティリティールーチン test_error() および test_stmt_error() を使用して、エラーをチェックし、エラーが発生した場合は診断情報を出力したあとに終了します。

```
static void test_error(MYSQL *mysql, int status)
{
 if (status)
 {
   fprintf(stderr, "Error: %s (errno: %d)\n",
           mysql_error(mysql), mysql_errno(mysql));
   exit(1);
 }
}

static void test_stmt_error(MYSQL_STMT *stmt, int status)
{
 if (status)
 {
   fprintf(stderr, "Error: %s (errno: %d)\n",
           mysql_stmt_error(stmt), mysql_stmt_errno(stmt));
   exit(1);
 }
}
```

23.8 MySQL PHP API

MySQL PHP API のマニュアルは、MySQL リファレンスマニュアルの一部としてではなく、独立した形式で発行されるようになりました。MySQL and PHPを参照してください。

23.9 MySQL Perl API

PerlDBIモジュールはデータベースアクセスのための一般的なインタフェースを提供します。変更せずに、多くのさまざまなデータベースエンジンで動作する DBI スクリプトを書くことができます。MySQL で DBI を使用するには、次をインストールします。

1. DBI モジュール。
2. DBD::mysql モジュール。これは Perl のデータベースドライバ (DBD) モジュールです。
3. オプションで、アクセスするほかの任意の種類のデータベースサーバーの DBD モジュール。

Perl DBI は推薦される Perl インタフェースです。それは、廃止とみなされるべき mysqlperl と呼ばれる古いインタフェースを置き換えます。

これらのセクションには、MySQL と Perl の使用および Perl での MySQL アプリケーションの作成に関する情報が含まれます。

- Perl DBI サポートのインストール手順については、セクション2.13「Perl のインストールに関する注釈」を参照してください。

- オプションファイルからオプションを読み取る例については、セクション5.3.4「複数サーバー環境でのクライアントプログラムの使用」を参照してください。
- セキュアなコーディングのヒントについては、セクション6.1.1「セキュリティーガイドライン」を参照してください。
- デバッグのヒントについては、セクション24.4.1.4「gdb での mysqld のデバッグ」を参照してください。
- いくつかの Perl 固有の環境変数については、セクション2.12「環境変数」を参照してください。
- OS X 上で実行する場合の考慮事項については、セクション2.4「OS X に MySQL をインストールする」を参照してください。
- 文字列リテラルを引用する方法については、セクション9.1.1「文字列リテラル」を参照してください。

DBI 情報はコマンド行、オンライン、または印刷物で取得できます。

- DBI と DBD::mysql モジュールをインストールしたら、コマンド行で perldoc コマンドを使って、それらに関する情報を得ることができます。

```
shell> perldoc DBI
shell> perldoc DBI::FAQ
shell> perldoc DBD::mysql
```

  pod2man、pod2html などを使用して、この情報をほかのフォーマットに変換することもできます。

- Perl DBI に関するオンライン情報については、DBI Web サイト http://dbi.perl.org/ にアクセスしてください。そのサイトでは一般的な DBI メーリングリストをホストしています。Oracle Corporation では、特に DBD::mysql に関するリストをホストしています。セクション1.5.1「MySQL メーリングリスト」を参照してください。
- 印刷された情報として、公式の DBI の書籍は『Programming the Perl DBI』(Alligator Descartes および Tim Bunce 著、O'Reilly & Associates 発行、2000 年) があります。この書籍については、DBI Web サイト http://dbi.perl.org/ を参照してください。

  MySQL と DBI の使用に特に焦点を合わせた情報については、『MySQL and Perl for the Web』(Paul DuBois 著、New Riders 発行、2001 年) を参照してください。この書籍の Web サイトは http://www.kitebird.com/mysql-perl/ です。

23.10 MySQL Python API

MySQLdb は Python DB API バージョン 2.0 に準拠した MySQL の Python のサポートを提供するサードパーティードライバです。それは http://sourceforge.net/projects/mysql-python/ にあります。

新しい MySQL Connector/Python コンポーネントは同じ Python API へのインタフェースを提供し、MySQL サーバーに組み込まれ、Oracle によってサポートされます。Connector の詳細と、Python アプリケーションのコーディングガイドラインおよびサンプル Python コードについては、MySQL Connector/Python Developer Guideを参照してください。

23.11 MySQL Ruby API

MySQL アプリケーションを開発する Ruby プログラマは 2 つの API を使用できます。

- MySQL/Ruby API は libmysqlclient API ライブラリに基づいています。MySQL/Ruby API のインストールと使用については、セクション23.11.1「MySQL/Ruby API」を参照してください。
- Ruby/MySQL API はネイティブ MySQL ネットワークプロトコル (ネイティブドライバ) を使用するために書かれています。Ruby/Ruby API のインストールと使用については、セクション23.11.2「Ruby/MySQL API」を参照してください。

Ruby 言語に関するバックグラウンドと構文情報については、Ruby プログラミング言語を参照してください。

23.11.1 MySQL/Ruby API

MySQL/Ruby モジュールは、libmysqlclient 経由で Ruby を使用した MySQL データベースへのアクセスを提供します。

モジュールのインストールと公開されている関数については、「MySQL/Ruby」を参照してください。

23.11.2 Ruby/MySQL API

MySQL/Ruby モジュールは、MySQL ネットワークプロトコルを使用して、ネイティブドライバインタフェース経由で、Ruby を使用した MySQL データベースへのアクセスを提供します。

モジュールのインストールと公開されている関数については、「Ruby/MySQL」を参照してください。

23.12 MySQL Tcl API

MySQLtcl は Tcl プログラミング言語から MySQL データベースサーバーにアクセスするための簡単な API です。それは http://www.xdobry.de/mysqltcl/ にあります。

23.13 MySQL Eiffel ラッパー

Eiffel MySQL は、Michael Ravits によって書かれた Eiffel プログラミング言語を使用した MySQL データベースサーバーへのインタフェースです。それは http://efsa.sourceforge.net/archive/ravits/mysql.htm にあります。

第 24 章 MySQL の拡張

目次

24.1 MySQL の内部仕様 2767
24.1.1 MySQL のスレッド 2767
24.1.2 MySQL テストスイート 2768
24.2 MySQL プラグイン API 2768
24.2.1 プラグイン API の特徴 2769
24.2.2 プラグイン API のコンポーネント 2770
24.2.3 プラグインのタイプ 2771
24.2.4 プラグインの作成 2773
24.2.5 プラグインのための MySQL サービス 2810
24.3 MySQL への新しい関数の追加 2812
24.3.1 ユーザー定義関数インタフェースの機能 2813
24.3.2 新しいユーザー定義関数の追加 2813
24.3.3 新しいネイティブ関数の追加 2821
24.4 MySQL のデバッグおよび移植 2822
24.4.1 MySQL サーバーのデバッグ 2823
24.4.2 MySQL クライアントのデバッグ 2828
24.4.3 DBUG パッケージ 2829

24.1 MySQL の内部仕様

この章では、MySQL コードを記述するときに知っておく必要がある多くの事柄について説明します。MySQL 開発の動向を調べたり貢献したりするには、セクション2.9.3「開発ソースツリーを使用して MySQL をインストールする」の手順に従ってください。MySQL の内部仕様について興味がある場合、弊社の internals メーリングリストに登録することもお勧めします。このリストのトラフィックは比較的少なめです。登録する方法については、セクション1.5.1「MySQL メーリングリスト」を参照してください。オラクルの多くの MySQL 開発者が internals リストに登録されており、弊社は MySQL コードを記述するユーザーを支援しています。コードについて質問したり、MySQL プロジェクトに貢献すると思われるパッチを送信したりするときに、このリストを気軽に利用してください。

24.1.1 MySQL のスレッド

MySQL サーバーは次のスレッドを作成します。

- 接続マネージャースレッドは、サーバーが待機しているネットワークインタフェース上でクライアントの接続要求を処理します。どのプラットフォームでも、1 つのマネージャースレッドが TCP/IP 接続要求を処理します。Unix では、このマネージャースレッドは Unix ソケットファイルの接続要求も処理します。Windows では、1 つのマネージャースレッドが共有メモリーの接続要求を処理し、別のマネージャースレッドが名前付きパイプの接続要求を処理します。サーバーは、待機していないインタフェースを処理するためのスレッドを作成しません。たとえば、Windows サーバーで名前付きパイプ接続のサポートが有効になっていない場合、これらの接続を処理するスレッドは作成されません。

- 接続マネージャースレッドは、各クライアント接続を、その接続の認証および要求を処理する専用スレッドに関連付けます。マネージャースレッドは、必要に応じて新しいスレッドを作成しますが、まずスレッドキャッシュを調べて接続に使用できるスレッドが含まれているかどうかを確認することによって、それを回避することを試みます。接続が終了すると、スレッドキャッシュが満杯でない場合は、そのスレッドがスレッドキャッシュに返されます。

  スレッドのリソースを制御するパラメータの調整については、セクション8.11.5.1「MySQL のクライアント接続のためのスレッドの使用方法」を参照してください。

- マスターレプリケーションサーバーでは、スレーブサーバーからの接続はクライアント接続のように扱われ、接続されているスレーブごとに 1 つのスレッドがあります。

- スレーブレプリケーションサーバー上では、マスターサーバーに接続してそこから更新を読み取るための I/O スレッドが開始されます。マスターから読み取られた更新を適用するための SQL スレッドが開始されます。これらの 2 つのスレッドは別個に実行され、別個に開始および停止できます。

- シグナルスレッドはすべてのシグナルを処理します。通常、このスレッドは、アラームの処理、および長時間アイドル状態になっている接続をタイムアウトさせる process_alarm() の呼び出しも行います。

- InnoDB が使用される場合は、デフォルトでは追加の読み取りスレッドおよび書き込みスレッドが存在します。これらの数は、innodb_read_io_threads および innodb_write_io_threads パラメータによって制御されます。セクション14.12「InnoDB の起動オプションおよびシステム変数」を参照してください。
- mysqld が -DUSE_ALARM_THREAD を指定してコンパイルされている場合は、アラームを処理する専用のスレッドが作成されます。これが使用されるのは、sigwait() に問題があるシステムの場合、および専用のシグナル処理スレッドを使用せずにアプリケーションで thr_alarm() コードを使用する場合のみです。
- サーバーが --flush_time=val オプションを指定して起動された場合、val 秒ごとにすべてのテーブルをフラッシュする専用のスレッドが作成されます。
- INSERT DELAYED ステートメントが発行された各テーブルには、独自のスレッドが作成されます。セクション13.2.5.2「INSERT DELAYED 構文」を参照してください。
- イベントスケジューラがアクティブである場合は、スケジューラの 1 つスレッド、および現在実行されている各イベントのそれぞれのスレッドが存在します。セクション20.4.1「イベントスケジューラの概要」を参照してください。

mysqladmin processlist は、接続、INSERT DELAYED、レプリケーション、およびイベントスレッドのみを表示します。

24.1.2 MySQL テストスイート

Unix ソースおよびバイナリ配布に付属するテストシステムを使用することによって、ユーザーおよび開発者は MySQL コードのリグレッションテストを実行できます。これらのテストは Unix 上で実行できます。

独自のテストケースを記述することもできます。システム要件を含む MySQL テストフレームワークについては、https://dev.mysql.com/doc/index-other.html から入手できるマニュアルを参照してください。

現在のテストケースのセットは MySQL のすべてをテストするわけではありませんが、SQL 処理コードのほとんどの明らかなバグやオペレーティングシステムまたはライブラリの問題を捕捉し、レプリケーションのテストに関してはかなり徹底しています。テストによってコードの 100% がカバーされることを目標としています。弊社はテストスイートへの貢献を歓迎しています。使用しているシステムで重要な機能を検査するテストに貢献すれば、使用しているアプリケーションが将来のすべての MySQL リリースで動作することが保証されます。

テストシステムは、テスト言語インタプリタ (mysqltest)、すべてのテストを実行する Perl スクリプト (mysql-test-run.pl)、特別なテスト言語で記述された実際のテストケース、およびそれらの予期される結果で構成されています。ビルド後にシステム上でテストスイートを実行するには、ソースのルートディレクトリから make test と入力するか、場所を mysql-test ディレクトリに変更して、./mysql-test-run.pl と入力します。バイナリ配布をインストールした場合は、インストールルートディレクトリの下にある mysql-test ディレクトリ (/usr/local/mysql/mysql-test など) に場所を変更して、./mysql-test-run.pl を実行します。すべてのテストが成功します。成功しないものがあれば原因を調べ、MySQL 内のバグを示す場合は問題を報告してください。セクション1.6「質問またはバグをレポートする方法」を参照してください。

1 つのテストに失敗する場合は、--force オプションを指定して mysql-test-run.pl を実行し、ほかのテストも失敗するかどうかを確認してください。

テストスイートを実行するマシン上で mysqld のコピーが実行中である場合、それがポート 9306 または 9307 を使用していなければ、停止する必要はありません。これらのいずれかのポートが使用中である場合、環境変数 MTR_BUILD_THREAD を適切な値に設定します。こうすることで、テストスイートは、マスター、スレーブ、および NDB のポートに別のセットを使用します。例:

```
shell> export MTR_BUILD_THREAD=31
shell> ./mysql-test-run.pl [options] [test_name]
```

mysql-test ディレクトリでは、./mysql-test-run.pl test_name を指定することで個々のテストケースを実行できます。

テストスイートについて質問がある場合、または貢献するテストケースがある場合は、MySQL internals メーリングリストに電子メールメッセージを送信してください。セクション1.5.1「MySQL メーリングリスト」 を参照してください。

24.2 MySQL プラグイン API

MySQL は、サーバーコンポーネントの作成を可能にするプラグイン API をサポートします。プラグインはサーバー起動時にロードしたり、実行時にサーバーを再起動せずにロードおよびアンロードしたりできます。API は汎用的なものであり、プラグインが実行できる動作を指定しません。このインタフェースによってサポートされ

るコンポーネントには、ストレージエンジン、全文パーサープラグイン、サーバー拡張機能、およびその他のコンポーネントがあります。

たとえば、全文パーサープラグインは、組み込みの全文パーサーの代わりに使用したり、強化したりするために使用できます。プラグインは、組み込みパーサーによって使用されるものと異なるルールを使用して、テキストを単語に構文解析できます。これは、組み込みパーサーが予期する特性と異なる特性を持つテキストを構文解析する必要がある場合に役立ちます。

プラグインインタフェースは、以前のユーザー定義関数 (UDF) インタフェースよりも汎用的です。

プラグインインタフェースは mysql データベースの plugin テーブルを使用して、INSTALL PLUGIN ステートメントによって永続的にインストールされているプラグインに関する情報を記録します。このテーブルは、MySQL インストール処理の一部として作成されます。プラグインは、--plugin-load オプションを指定して単一のサーバーを起動することによってもインストールできます。この方法でインストールされたプラグインは、plugin テーブルに記録されません。セクション5.1.8.1「プラグインのインストールおよびアンインストール」を参照してください。

MySQL 5.6 では、サーバープラグイン用の API のほかにクライアントプラグイン用の API もサポートされています。これは、たとえば、サーバー側プラグインとクライアント側プラグインが連係して、クライアントがサーバーにさまざまな認証方式を使用して接続できるようにする認証プラグインによって使用されます。

追加のリソース

Sergei Golubchik および Andrew Hutchings 共著の『MySQL 5.1 Plugin Development』には、プラグイン API に関する豊富な情報が記載されています。この書籍の表題には MySQL Server 5.1 と記載されていますが、書籍内の大半の情報は以降のバージョンにも同様に当てはまります。

24.2.1 プラグイン API の特徴

サーバープラグイン API には次の特徴があります。

- すべてのプラグインにはいくつかの共通点があります。

  各プラグインには、SQL ステートメント内で参照可能な名前のほかに、ほかの情報を提供する作成者および説明などのメタデータがあります。この情報は、INFORMATION_SCHEMA.PLUGINS テーブルまたは SHOW PLUGINS ステートメントを使用して調査できます。

- プラグインフレームワークは、別の種類のプラグインに対応するために拡張可能です。

  プラグイン API の一部の特性はすべてのタイプのプラグインに共通していますが、API ではタイプ固有のインタフェース要素も使用できるため、異なるタイプのプラグインを作成できます。ある用途を持つプラグインは、それ自体の要件にもっとも適したインタフェースを持っているため、ほかのプラグインタイプの要件には適していない可能性があります。

  ストレージエンジン、全文パーサー、INFORMATION_SCHEMA テーブルなどのさまざまなタイプのプラグイン用のインタフェースが存在します。ほかのインタフェースも追加できます。

- プラグインは情報をユーザーに公開します。

  プラグインは、SHOW VARIABLES および SHOW STATUS ステートメントを使用して得られるシステム変数およびステータス変数を実装できます。

- プラグイン API にはバージョン情報が含まれています。

  プラグイン API に含まれているバージョン情報によって、プラグインライブラリおよびそれに含まれる各プラグインが、ライブラリのビルドに使用された API バージョンを自己認識できます。API がその後変更されるとバージョン番号が変更されますが、サーバーは対象となるプラグインライブラリのバージョン情報を検査して、そのライブラリ内のプラグインをサポートしているかどうかを判別できます。

  バージョン番号には 2 つのタイプがあります。1 つ目は、全般的なプラグインフレームワーク自体のバージョン番号です。各プラグインライブラリには、この種類のバージョン番号が含まれています。2 つ目のタイプのバージョンは個々のプラグインに適用されます。特定のタイプの各プラグインにはそのインタフェースのバージョンがあるため、ライブラリ内の各プラグインにはタイプ固有のバージョン番号があります。たとえば、全文パーサープラグインを含むライブラリには全般的なプラグイン API バージョン番号があり、そのプラグインには全文プラグインインタフェースに固有のバージョン番号があります。

- プラグイン API はセキュリティー制約を実装します。

プラグインライブラリは特定の専用ディレクトリにインストールする必要があり、ディレクトリの場所はサーバーによって制御され、実行時に変更することはできません。また、ライブラリには、それがプラグインライブラリであることを識別する特定のシンボルが含まれている必要があります。サーバーは、プラグインとしてビルドされていないものをプラグインとしてロードしません。

- プラグインはサーバーサービスにアクセスできます。

  サービスインタフェースは、プラグインが通常の関数呼び出しを使用してアクセスできるサーバー機能を公開します。詳細は、セクション24.2.5「プラグインのための MySQL サービス」を参照してください。

サーバープラグイン API は、それに置き換わる前のユーザー定義関数 (UDF) API といくつかの点で似ていますが、古いインタフェースよりさまざまな点で優れています。たとえば、UDF にはバージョン情報がありません。また、新しいプラグインインタフェースには、以前の UDF インタフェースでのセキュリティー問題がありません。プラグインでない UDF を記述するための以前のインタフェースでは、システムのダイナミックリンカーによって検索されたすべてのディレクトリからライブラリをロードでき、UDF ライブラリを識別するシンボルは比較的特定性の低いものでした。

クライアントプラグイン API には類似したアーキテクチャー上の特徴がありますが、クライアントプラグインはサーバープラグインのようにサーバーに直接アクセスしません。

24.2.2 プラグイン API のコンポーネント

サーバープラグインの実装は、いくつかのコンポーネントで構成されています。

SQL ステートメント:

- INSTALL PLUGIN はプラグインを mysql.plugin テーブルに登録し、プラグインコードをロードします。
- UNINSTALL PLUGIN はプラグインを mysql.plugin テーブルから登録解除し、プラグインコードをアンロードします。
- 全文インデックス作成用の WITH PARSER 句は、全文パーサープラグインを特定の FULLTEXT インデックスに関連付けます。
- SHOW PLUGINS は、サーバープラグインについての情報を表示します。

コマンド行オプションおよびシステム変数:

- --plugin-load オプションを指定すると、サーバー起動時にプラグインをロードできます。
- plugin_dir システム変数は、すべてのプラグインをインストールする必要があるディレクトリの場所を指定します。この変数の値は、サーバー起動時に --plugin_dir=path オプションで指定できます。mysql_config --plugindir を指定すると、デフォルトのプラグインディレクトリのパス名が表示されます。

プラグインのロードについての追加情報は、セクション5.1.8.1「プラグインのインストールおよびアンインストール」を参照してください。

プラグインに関連するテーブル:

- INFORMATION_SCHEMA.PLUGINS テーブルにはプラグイン情報が格納されています。
- mysql.plugin テーブルには、INSTALL PLUGIN によってインストールされた各プラグインが示され、プラグインを使用するために必要となります。新規に MySQL をインストールする場合、このテーブルはインストール処理中に作成されます。

クライアントプラグインの実装はより単純です。

- mysql_options() C API 関数の場合は、MYSQL_DEFAULT_AUTH オプションおよび MYSQL_PLUGIN_DIR オプションを指定すると、クライアントプログラムが認証プラグインをロードできます。
- クライアントプラグインを管理できる C API 関数があります。

MySQL がプラグインを実装する方法を調べるには、MySQL ソース配布内の次のソースファイルを参照してください。

- include/mysql ディレクトリの plugin.h は、パブリックなプラグイン API を公開しています。プラグインライブラリを記述するすべてのユーザーは、このファイルを調査することをお勧めします。plugin_xxx.h ファイルには

特定のタイプのプラグインに関する追加情報があります。client_plugin.h にはクライアントプラグインに固有の情報が含まれています。

- sql ディレクトリ内の sql_plugin.h および sql_plugin.cc は、内部プラグインの実装を構成しています。sql_acl.cc はサーバーが認証プラグインを使用する場所です。プラグイン開発者はこれらのファイルを参照する必要はありません。サーバーがプラグインを処理する方法について知りたい場合は、これらのファイルを参照できます。
- sql-common ディレクトリ内の、client_plugin.h は C API クライアントプラグイン関数を実装し、client.c はクライアント認証サポートを実装します。プラグイン開発者はこれらのファイルを参照する必要はありません。サーバーがプラグインを処理する方法について知りたい場合は、これらのファイルを参照できます。

24.2.3 プラグインのタイプ

プラグイン API を使用すると、さまざまな機能が実装されたプラグインを作成できます。

- ストレージエンジン
- 全文パーサー
- デーモン
- INFORMATION_SCHEMA テーブル
- 準同期レプリケーション
- 監査
- 認証

次のセクションでは、これらのプラグインタイプの概要について説明します。

24.2.3.1 ストレージエンジンプラグイン

MySQL サーバーによって使用されるプラガブルストレージエンジンアーキテクチャーにより、ストレージエンジンをプラグインとして作成し、実行中のサーバーにロードしたりそのサーバーからアンロードしたりできます。このアーキテクチャーの説明については、セクション15.11「MySQL ストレージエンジンアーキテクチャーの概要」を参照してください。

プラグイン API を使用してストレージエンジンを作成するための方法については、「MySQL Internals: Writing a Custom Storage Engine」を参照してください。

24.2.3.2 全文パーサープラグイン

MySQL には、全文操作 (インデックス付けされるテキストの構文解析、または検索に使用される用語を判別するためのクエリー文字列の構文解析) のためにデフォルトで使用される組み込みパーサーがあります。全文処理の場合、「構文解析」とは、単語を構成する文字シーケンスや単語の境界となる位置を定義するルールに基づいて、テキストまたはクエリー文字列から単語を抽出することを意味します。

インデックスを作成するために構文解析するとき、パーサーは各単語をサーバーに渡し、サーバーは単語を全文インデックスに追加します。クエリー文字列を構文解析するとき、パーサーは各単語をサーバーに渡し、サーバーは単語を蓄積して検索に使用します。

組み込みの全文パーサーの構文解析プロパティーは、セクション12.9「全文検索関数」に記載されています。これらのプロパティーには、テキストから単語を抽出する方法を決定するためのルールも含まれています。パーサーは、短い単語または長い単語を除外する ft_min_word_len、ft_max_word_len などの特定のシステム変数、および無視される一般的な単語を示すストップワードリストの影響を受けます。

プラグイン API を使用すると、独自の全文パーサーを作成して、パーサーの基本的な機能を制御できます。パーサープラグインは 2 つの役割のいずれかで動作できます。

- プラグインを組み込みパーサーの代わりに使用できます。この役割では、プラグインは構文解析する対象の入力データを読み取り、入力データを単語に分割し、単語を (インデックス設定または単語の蓄積のために) サーバーに渡します。

  この方法でパーサーを使用する 1 つの理由は、入力データを単語に分割する方法を決定するために、組み込みパーサーのルールとは異なるルールを使用する必要がある場合です。たとえば、組み込みパーサーは「case-

sensitive」というテキストを「case」および「sensitive」という 2 つの単語で構成されるものとして解釈するが、あるアプリケーションではこのテキストを単一の単語として扱う必要があることもあります。

- プラグインは組み込みパーサーのフロントエンドとして動作することによって、組み込みパーサーと連動できます。この役割では、プラグインは入力データからテキストを抽出してテキストをパーサーに渡し、パーサーはその通常の構文解析ルールを使用してテキストを単語に分割します。この構文解析は、特に、ft_xxx システム変数およびストップワードリストの影響を受けます。

  この方法でパーサーを使用する 1 つの理由は、PDF ドキュメント、XML ドキュメント、.doc ファイルなどのコンテンツにインデックスを設定する必要がある場合です。組み込みパーサーはこれらのタイプの入力データのためのものではありませんが、プラグインを使用してこれらの入力ソースからテキストを抜き出し、テキストを組み込みパーサーに渡すことができます。

パーサープラグインが両方の役割で動作することも可能です。つまり、平文でないテキスト入力からテキストを抽出し (フロントエンドの役割)、テキストを構文解析して単語を得る (組み込みパーサーの代わりとなる) ことができます。

全文プラグインは、インデックスごとに全文インデックスに関連付けられます。つまり、パーサープラグインを最初にインストールした状態では、全文操作にパーサープラグインを使用することはできません。単にパーサープラグインが利用可能になるだけです。たとえば、個々の FULLTEXT インデックスを作成するときに、全文パーサープラグインを WITH PARSER 句に指定できるようになります。テーブル作成時にそのようなインデックスを作成するには、次のステートメントを実行します。

```
CREATE TABLE t
(
  doc CHAR(255),
  FULLTEXT INDEX (doc) WITH PARSER my_parser
) ENGINE=MyISAM;
```

または、テーブルが作成されたあとにインデックスを追加できます。

```
ALTER TABLE t ADD FULLTEXT INDEX (doc) WITH PARSER my_parser;
```

パーサーをインデックスに関連付けるために SQL を変更する箇所は、WITH PARSER 句のみです。検索は以前と同様に指定し、クエリーを変更する必要はありません。

パーサープラグインを FULLTEXT インデックスに関連付けると、そのインデックスを使用するためにこのプラグインが必要になります。そのパーサープラグインが削除された場合は、パーサープラグインに関連付けられていたすべてのインデックスが使用できなくなります。プラグインを使用できないテーブルを使用しようとすると エラーになりますが、DROP TABLE は引き続き実行できます。

全文プラグインについては、セクション24.2.4.4「全文パーサープラグインの作成」を参照してください。MySQL 5.6 では、MyISAM を使用したフルインデックスプラグインのみがサポートされます。

24.2.3.3 デーモンプラグイン

デーモンプラグインは、サーバーによって実行されるがサーバーと通信しないコードに使用される単純なタイプのプラグインです。MySQL 配布には、定期的にハートビートメッセージをファイルに書き込むデーモンプラグインの例が含まれています。

デーモンプラグインについては、セクション24.2.4.5「デーモンプラグインの作成」を参照してください。

24.2.3.4 INFORMATION_SCHEMA プラグイン

INFORMATION_SCHEMA プラグインを使用すると、INFORMATION_SCHEMA データベース経由でユーザーに公開されるサーバーメタデータを含むテーブルを作成できます。たとえば、InnoDB は、INFORMATION_SCHEMA プラグインを使用して、現在のトランザクションおよびロックに関する情報が含まれているテーブルを提供しています。

INFORMATION_SCHEMA プラグインについては、セクション24.2.4.6「INFORMATION_SCHEMA プラグインの作成」を参照してください。

24.2.3.5 準同期レプリケーションプラグイン

MySQL レプリケーションはデフォルトでは非同期です。準同期レプリケーションでは、マスター側で実行されたコミットは、トランザクションを実行したセッションに戻る前に、少なくとも 1 つのスレーブがトランザクションのイベントを受け取ってログに記録したことを通知するまでブロックされます。準同期レプリケーションは、補完的なマスタープラグインおよびクライアントプラグインを介して実装されます。セクション17.3.8「準同期レプリケーション」を参照してください。

準同期レプリケーションについては、セクション24.2.4.7「準同期レプリケーションプラグインの作成」を参照してください。

24.2.3.6 監査プラグイン

MySQL 5.6 では、サーバー操作に関する情報を関係者に報告できるプラガブルな監査インタフェースがサーバーによって提供されます。現時点では、次の操作について監査通知が行われます (インタフェースは汎用的なものですが、サーバーを変更してほかの情報を報告できます)。

- 一般クエリーログへのメッセージの書き込み (ログが有効にされている場合)
- エラーログへのメッセージの書き込み
- クライアントへのクエリー結果の送信

監査プラグインは、サーバー操作についての通知を受け取るために監査インタフェースに登録できます。監査可能なイベントがサーバー内で発生すると、サーバーは通知が必要であるかどうかを判別します。登録されている各監査プラグインについて、サーバーはプラグインが対象としているイベントクラスに対してイベントを照合し、一致する場合はイベントをプラグインに渡します。

このインタフェースでは、監査プラグインが重要だとみなすイベントクラスの操作の通知のみを監査プラグインが受け取り、ほかのものを無視できます。このインタフェースは、操作をイベントクラスまで分類し、さらに各クラス内のイベントサブクラスまで分割します。

監査プラグインに監査可能なイベントが通知されると、監査プラグインは現在の THD 構造体へのポインタ、およびイベントについての情報を含む構造へのポインタを受け取ります。プラグインはイベントを検査し、適切な監査アクションを実行します。たとえば、プラグインは、結果セットを生成したステートメント、ログに記録されたステートメント、結果の行数、操作についての現在のユーザー、または失敗した操作のエラーコードを確認できます。

監査プラグインについては、セクション24.2.4.8「監査プラグインの作成」を参照してください。

24.2.3.7 認証プラグイン

MySQL 5.6 はプラガブルな認証をサポートしています。認証プラグインはサーバー側とクライアント側の両方に存在します。サーバー側のプラグインは、クライアントがサーバーに接続するときにクライアントによって使用される認証方式を実装します。クライアント側のプラグインはサーバー側のプラグインと通信して、サーバー側のプラグインが必要とする認証情報を提供します。クライアント側のプラグインは、ユーザーと対話し、サーバーに送信するパスワードやその他の認証情報の要求などのタスクを実行することもあります。セクション6.3.7「プラガブル認証」を参照してください。

プラガブルな認証では、あるユーザーが別のユーザーの ID を取得するプロキシユーザー機能も使用できます。サーバー側の認証プラグインは、接続するユーザーが使用するべき ID の所有者であるユーザーの名前をサーバーに返すことができます。セクション6.3.9「プロキシユーザー」を参照してください。

認証プラグインについては、セクション24.2.4.9「認証プラグインの作成」を参照してください。

24.2.3.8 パスワード検証プラグイン

MySQL 5.6.6 以降では、サーバーはパスワードをテストするプラグインを記述するためのインタフェースを提供しています。そのようなプラグインは、2 つの機能を実装します。

- パスワードを割り当てるステートメント (CREATE USER、GRANT、SET PASSWORD ステートメントなど) の非常に弱いパスワード、および PASSWORD() 関数および OLD_PASSWORD() 関数に引数として指定されたパスワードの拒否。
- VALIDATE_PASSWORD_STRENGTH() SQL 関数でのパスワード候補の強さの評価。

このタイプのプラグインを作成については、セクション24.2.4.10「パスワード検証プラグインの作成」を参照してください。

24.2.4 プラグインの作成

プラグインライブラリを作成するには、ライブラリファイルに含まれるプラグインを示す必要なディスクリプタ情報を提供し、各プラグインのインタフェース関数を記述する必要があります。

すべてのサーバープラグインには、プラグイン API に情報を提供する一般ディスクリプタ、および特定のタイプのプラグインにプラグインインタフェースについての情報を提供するタイプ固有のディスクリプタがある必要が

あります。一般ディスクリプタの構造体は、すべてのプラグインタイプで同じです。タイプ固有のディスクリプタの構造体はプラグインタイプによって異なり、プラグインが実行する必要がある動作の要件によって決定されます。サーバープラグインインタフェースを使用すると、プラグインはステータス変数およびシステム変数を公開することもできます。これらの変数は、SHOW STATUS ステートメントや SHOW VARIABLES ステートメント、および対応する INFORMATION_SCHEMA テーブルを介して表示できます。

クライアント側のプラグインの場合、アーキテクチャーは少し異なります。各プラグインにはディスクリプタがある必要がありますが、一般ディスクリプタとタイプ固有のディスクリプタに分割されていません。そうではなく、ディスクリプタはすべてのクライアントプラグインに共通する固定された一連のメンバーで始まり、この共通メンバーのあとに、特定のプラグインタイプを実装するために必要な追加のメンバーが続きます。

プラグインは、C または C++ (あるいは C の呼び出し規則を使用できる別の言語) で記述できます。プラグインは動的にロードおよびアンロードされるため、オペレーティングシステムが動的ロードをサポートしている必要があり、呼び出し元アプリケーションを (静的にではなく) 動的にコンパイルしている必要があります。サーバープラグインの場合、これは mysqld を動的にコンパイルする必要があることを意味します。

サーバープラグインには実行中のサーバーの一部となるコードが含まれるため、プラグインを記述するときは、サーバーコードを作成する場合に該当するすべての制約に従う必要があります。たとえば、libstdc++ ライブラリの関数を使用しようとすると、問題が生じることがあります。これらの制約はサーバーの今後のバージョンで変更される場合があるため、サーバーのアップグレードにより、古いサーバー用に作成されたプラグインの改訂が必要になることがあります。これらの制約については、セクション2.9.4「MySQL ソース構成オプション」およびセクション2.9.5「MySQL のコンパイルに関する問題」を参照してください。

どのようなアプリケーションがプラグインを使用するかはわからないため、クライアントプラグインの作成者は、呼び出し元アプリケーションが持つシンボルに依存しないようにしてください。

24.2.4.1 プラグインの作成の概要

次の手順では、プラグインライブラリの作成に必要なステップの概要を示します。以降のセクションでは、プラグインのデータ構造体の設定、および特定のタイプのプラグインの作成について詳しく説明します。

1. プラグインのソースファイルに、プラグインライブラリが必要とするヘッダーファイルをインクルードします。plugin.h ファイルは必須であり、ライブラリではほかのファイルも必要になることがあります。例:

   ```
   #include <stdlib.h>
   #include <ctype.h>
   #include <mysql/plugin.h>
   ```

2. プラグインライブラリファイル用のディスクリプタ情報をセットアップします。サーバープラグインの場合は、ライブラリディスクリプタを記述します。ライブラリディスクリプタには、ファイル内の各サーバープラグインの一般プラグインディスクリプタが含まれている必要があります。詳細は、サーバープラグインライブラリおよびプラグインディスクリプタを参照してください。また、ライブラリ内の各サーバープラグインのタイプ固有のディスクリプタをセットアップします。各プラグインの一般ディスクリプタは、タイプ固有のディスクリプタを指しています。

   クライアントプラグインの場合は、クライアントディスクリプタを記述します。詳細は、クライアントプラグインディスクリプタを参照してください。

3. 各プラグインのプラグインインタフェース関数を作成します。たとえば、各プラグインの一般プラグインディスクリプタは、サーバーがプラグインをロードおよびアンロードするときに呼び出す初期化関数および初期化解除関数を指しています。プラグインのタイプ固有のディスクリプタは、インタフェース関数を指していることもあります。

4. サーバープラグインの場合は、ステータス変数およびシステム変数を設定します (ある場合)。

5. プラグインライブラリを共有ライブラリとしてコンパイルし、プラグインディレクトリにインストールします。詳細は、セクション24.2.4.3「プラグインライブラリのコンパイルおよびインストール」を参照してください。

6. サーバープラグインの場合は、プラグインをサーバーに登録します。詳細は、セクション5.1.8.1「プラグインのインストールおよびアンインストール」を参照してください。

7. プラグインをテストして、正しく動作することを確認します。

24.2.4.2 プラグインのデータ構造体

プラグインライブラリファイルには、それに格納されているプラグインを示すディスクリプタ情報が含まれています。

プラグインライブラリにサーバープラグインが含まれている場合、プラグインライブラリには次のディスクリプタ情報が含まれている必要があります。

- ライブラリディスクリプタは、ライブラリによって使用される一般的なサーバープラグイン API バージョン番号を示し、ライブラリ内の各サーバープラグインの一般プラグインディスクリプタを含んでいます。このディスクリプタのフレームワークを提供するには、plugin.h ヘッダーファイルから 2 つのマクロを呼び出します。

  ```
  mysql_declare_plugin(name)
   ... one or more server plugin descriptors here ...
  mysql_declare_plugin_end;
  ```

  マクロが展開され、API バージョンの宣言が自動的に提供されます。プラグインディスクリプタを指定する必要があります。

- ライブラリディスクリプタ内の各一般的なサーバープラグインは、st_mysql_plugin 構造体によって記述されます。このプラグインのディスクリプタ構造体には、すべてのタイプのサーバープラグインに共通する情報 (プラグインタイプを示す値、プラグイン名、作成者、説明、ライセンスタイプ、サーバーがプラグインをロードおよびアンロードするときに呼び出す初期化関数と初期化解除関数へのポインタ、およびプラグインが実装するステータス変数またはシステム変数へのポインタ) が含まれています。

- ライブラリディスクリプタ内の各一般的なサーバープラグインディスクリプタには、タイプ固有のプラグインディスクリプタへのポインタも含まれています。プラグインはタイプごとに独自の API を持つことがあるため、タイプ固有のディスクリプタの構造体はプラグインタイプによって異なります。タイプ固有のプラグインディスクリプタには、タイプ固有の API バージョン番号およびそのプラグインタイプを実装するために必要な関数へのポインタが含まれています。たとえば、全文パーサープラグインには、初期化関数と初期化解除関数、およびメインの構文解析関数があります。サーバーはプラグインを使用してテキストを構文解析するときに、これらの関数を呼び出します。

プラグインライブラリには、ライブラリ内の各プラグインの一般ディスクリプタおよびタイプ固有のディスクリプタによって参照されるインタフェース関数も格納されています。

プラグインライブラリにクライアントプラグインが格納されている場合は、そのプラグインのディスクリプタが含まれている必要があります。そのディスクリプタはすべてのクライアントプラグインに共通する固定された一連のメンバーで始まり、そのあとにプラグインタイプ固有のメンバーが続きます。ディスクリプタフレームワークを提供するには、client_plugin.h ヘッダーファイルから 2 つのマクロを呼び出します。

```
mysql_declare_client_plugin(plugin_type)
  ... members common to all client plugins ...
  ... type-specific extra members ...
mysql_end_client_plugin;
```

プラグインライブラリには、クライアントディスクリプタによって参照されるインタフェース関数も格納されています。

mysql_declare_plugin() マクロおよび mysql_declare_client_plugin() マクロは呼び出す方法が若干異なり、これはプラグインライブラリの内容が関係しています。次のガイドラインはこのルールを要約しています。

- mysql_declare_plugin() および mysql_declare_client_plugin() は同じソースファイル内で使用できます。これは、1 つのプラグインライブラリにサーバープラグインとクライアントプラグインを両方格納できることを意味します。ただし、mysql_declare_plugin() と mysql_declare_client_plugin() はそれぞれ 1 回しか使用できません。

- mysql_declare_plugin() は複数のサーバープラグイン宣言が可能であるため、1 つのプラグインライブラリに複数のサーバープラグインを格納できます。

- mysql_declare_client_plugin() は、単一のクライアントプラグイン宣言のみを行うことができます。複数のクライアントプラグインを作成するには、別々のプラグインライブラリを使用する必要があります。

プラグインライブラリに存在して libmysqlclient に組み込まれていないクライアントプラグインをクライアントプログラムが探す場合、クライアントプログラムはプラグイン名と同じベース名を持つファイルを探します。たとえば、ライブラリのサフィクスとして .so を使用するシステムで、auth_xxx という名前のクライアント認証プラグインをプログラムで使用する必要がある場合、プログラムは auth_xxx.so という名前のファイルを探します。(OS X では、プログラムはまず auth_xxx.dylib を探し、次に auth_xxx.so を探します。)このため、プラグインライブラリにクライアントプラグインが格納されている場合、ライブラリはそのプラグインと同じベース名である必要があります。

サーバープラグインを格納しているライブラリの場合は異なります。--plugin-load オプションおよび INSTALL PLUGIN ステートメントはライブラリファイル名を明示的に指定するため、ライブラリ名とライブラリに格納されているサーバープラグインの名前に明示的な関係がある必要はありません。

サーバープラグインライブラリおよびプラグインディスクリプタ

サーバープラグインを格納するすべてのプラグインライブラリには、ファイル内の各サーバープラグインの一般プラグインディスクリプタが含まれているライブラリディスクリプタが格納されている必要があります。このセクションでは、サーバープラグイン用のライブラリおよび一般ディスクリプタを記述する方法について説明します。

ライブラリディスクリプタは 2 つのシンボルを定義する必要があります。

- _mysql_plugin_interface_version_ は全般的なプラグインフレームワークのバージョン番号を指定します。これは MYSQL_PLUGIN_INTERFACE_VERSION シンボルを使用して指定し、このシンボルは plugin.h ファイルに定義します。
- _mysql_plugin_declarations_ は、プラグイン宣言の配列を定義し、すべてのメンバーが 0 に設定された宣言で終わります。各宣言は、st_mysql_plugin 構造体のインスタンスです (plugin.h 内でも定義されます)。ライブラリ内のサーバープラグインごとに、これらのいずれかが必要となります。

サーバーがライブラリ内でこれらの 2 つのシンボルを見つけることができない場合、サーバーはこれを正当なプラグインライブラリとして受け入れず、拒否してエラーを発生させます。これにより、ライブラリをプラグインライブラリとして特別にビルドしないかぎり、プラグイン用にライブラリを使用できなくなります。

必須の 2 つのシンボルを定義する通常の方法は、plugin.h ファイルから mysql_declare_plugin() マクロおよび mysql_declare_plugin_end マクロを使用することです。

```
mysql_declare_plugin(name)
 ... one or more server plugin descriptors here ...
mysql_declare_plugin_end;
```

各サーバープラグインには、サーバープラグイン API に情報を提供する一般ディスクリプタが必要となります。一般ディスクリプタの構造体はすべてのプラグインタイプで同じです。plugin.h ファイル内の st_mysql_plugin 構造体によって、このディスクリプタが定義されます。

```
struct st_mysql_plugin
{
  int type;            /* the plugin type (a MYSQL_XXX_PLUGIN value)   */
  void *info;          /* pointer to type-specific plugin descriptor   */
  const char *name;     /* plugin name                                 */
  const char *author;   /* plugin author (for I_S.PLUGINS)             */
  const char *descr;    /* general descriptive text (for I_S.PLUGINS)   */
  int license;          /* the plugin license (PLUGIN_LICENSE_XXX)     */
  int (*init)(void *);  /* the function to invoke when plugin is loaded */
  int (*deinit)(void *);/* the function to invoke when plugin is unloaded */
  unsigned int version; /* plugin version (for I_S.PLUGINS)            */
  struct st_mysql_show_var *status_vars;
  struct st_mysql_sys_var **system_vars;
  void * __reserved1;   /* reserved for dependency checking            */
  unsigned long flags;  /* flags for plugin */
};
```

st_mysql_plugin ディスクリプタ構造体のメンバーは次のように使用します。char * メンバーは、NULL で終わる文字列として指定してください。

- type: プラグインの型。これは plugin.h のプラグインタイプ値のいずれかである必要があります。

  ```
  /*
    The allowable types of plugins
  */
  #define MYSQL_UDF_PLUGIN              0  /* User-defined function       */
  #define MYSQL_STORAGE_ENGINE_PLUGIN   1  /* Storage Engine              */
  #define MYSQL_FTPARSER_PLUGIN         2  /* Full-text parser plugin     */
  #define MYSQL_DAEMON_PLUGIN           3  /* The daemon/raw plugin type */
  #define MYSQL_INFORMATION_SCHEMA_PLUGIN  4  /* The I_S plugin type */
  #define MYSQL_AUDIT_PLUGIN            5  /* The Audit plugin type       */
  #define MYSQL_REPLICATION_PLUGIN      6  /* The replication plugin type */
  #define MYSQL_AUTHENTICATION_PLUGIN  7  /* The authentication plugin type */
  ...
  ```

  たとえば、全文パーサープラグインの場合、type 値は MYSQL_FTPARSER_PLUGIN です。

- info: プラグインのタイプ固有のディスクリプタへのポインタ。このディスクリプタの構造体は、一般プラグインディスクリプタ構造体とは異なり、プラグインの特定のタイプによって異なります。バージョンを制御するために、すべてのプラグインタイプのタイプ固有のディスクリプタの最初のメンバーは、タイプのインタ

フェースバージョンであることが予期されています。これにより、サーバーはタイプに関係なくすべてのプラグインのタイプ固有のバージョンをチェックできます。ディスクリプタには、バージョン番号のあとに、必要なほかのメンバー (サーバーがプラグインを適切に呼び出すために必要となるコールバック関数やその他の情報など) が含まれています。特定タイプのサーバープラグインの記述に関する以降のセクションでは、それらのタイプ固有のディスクリプタの構造体について説明しています。

- name: プラグイン名を指定する文字列。これは mysql.plugin テーブルに表示される名前であり、この名前を使用して、SQL ステートメント (INSTALL PLUGIN、UNINSTALL PLUGIN など) でプラグインを参照したり、--plugin-load オプションで指定してプラグインを参照したりします。この名前は、INFORMATION_SCHEMA.PLUGINS テーブルおよび SHOW PLUGINS の出力にも表示されます。

  プラグイン名は、サーバーオプション名で始まらないようにしてください。そのようにした場合、サーバーはプラグインの初期化に失敗します。たとえば、サーバーには --socket オプションがあるため、socket、socket_plugin などのプラグイン名を使用しないでください。

- author: プラグイン作成者を示す文字列。これには任意の文字列を指定できます。

- desc: プラグインの概要を説明する文字列。これには任意の文字列を指定できます。

- license: プラグインのライセンスタイプ。値には PLUGIN_LICENSE_PROPRIETARY、PLUGIN_LICENSE_GPL、または PLUGIN_LICENSE_BSD のいずれかを指定できます。

- init: 1 回だけ実行される初期化関数であり、そのような関数がない場合は NULL です。サーバーはプラグインをロードするときにこの関数を実行し、これは INSTALL PLUGIN で実行されるか、mysql.plugin テーブルにリストされているプラグインの場合はサーバー起動時に実行されます。この関数は、プラグインを識別するために使用される内部構造体を指している 1 つの引数を受け取ります。成功した場合はゼロ、および失敗した場合はゼロ以外を返します。

- deinit: 1 回だけ実行される初期化解除関数であり、そのような関数がない場合は NULL です。サーバーはプラグインをアンロードするときにこの関数を実行し、これは UNINSTALL PLUGIN で実行されるか、mysql.plugin テーブルにリストされているプラグインの場合はサーバーのシャットダウン時に実行されます。この関数は、プラグインを識別するために使用される内部構造体を指している 1 つの引数を受け取ります。成功した場合はゼロ、および失敗した場合はゼロ以外を返します。

- version: プラグインのバージョン番号。プラグインがインストールされている場合は、この値を INFORMATION_SCHEMA.PLUGINS テーブルから取得できます。この値にはメジャー番号とマイナー番号が含まれています。16 進数の定数として値を記述する場合、形式は 0xMMNN であり、ここで MM および NN はそれぞれメジャー番号とマイナー番号です。たとえば、0x0302 はバージョン 3.2 を表します。

- status_vars: プラグインに関連付けられているステータス変数の構造体へのポインタであり、そのような変数がない場合は NULL です。プラグインをインストールすると、これらの変数は SHOW STATUS ステートメントの出力として表示されます。

  status_vars のメンバーは、NULL ではない場合、ステータス変数を記述する st_mysql_show_var 構造体の配列を指しています。サーバープラグインのステータス変数およびシステム変数を参照してください。

- system_vars: プラグインに関連付けられているシステム変数の構造体へのポインタであり、そのような変数がない場合は NULL です。これらのオプションおよびシステム変数は、プラグイン内の変数を初期化するために使用できます。

  system_vars のメンバーは、NULL ではない場合、システム変数を記述する st_mysql_sys_var 構造体の配列を指しています。サーバープラグインのステータス変数およびシステム変数を参照してください。

- __reserved1: 今後利用するためのプレースホルダ。現時点では NULL を設定してください。

- flags: プラグインフラグ。個々のビットは各種のフラグに対応しています。この値には、該当するフラグの論理和を設定してください。次のフラグを使用できます。

```
#define PLUGIN_OPT_NO_INSTALL   1UL   /* Not dynamically loadable */
#define PLUGIN_OPT_NO_UNINSTALL 2UL   /* Not dynamically unloadable */
```

  PLUGIN_OPT_NO_INSTALL は、INSTALL PLUGIN ステートメントを使用してプラグインを実行時にロードできないことを示します。これは、--plugin-load オプションを使用してサーバー起動時にロードする必要があるプラグインの場合に適しています。PLUGIN_OPT_NO_UNINSTALL は、UNINSTALL PLUGIN ステートメントを使用してプラグインを実行時にアンロードできないことを示します。

  このメンバーは MySQL 5.6.3 で追加されました。

サーバーは、プラグインをロードおよびアンロードする場合にのみ、一般プラグインディスクリプタ内の init 関数および deinit 関数を呼び出します。これらは、SQL ステートメントによってプラグインが起動された場合などのプラグインの使用時には関係がありません。

たとえば、simple_parser という名前の単一の全文パーサープラグインを含むライブラリのディスクリプタ情報は、次のようになります。

```
mysql_declare_plugin(ftexample)
{
  MYSQL_FTPARSER_PLUGIN,      /* type                         */
  &simple_parser_descriptor,  /* descriptor                   */
  "simple_parser",            /* name                         */
  "Oracle Corporation",       /* author                       */
  "Simple Full-Text Parser",  /* description                  */
  PLUGIN_LICENSE_GPL,         /* plugin license               */
  simple_parser_plugin_init,  /* init function (when loaded)  */
  simple_parser_plugin_deinit,/* deinit function (when unloaded) */
  0x0001,                     /* version                      */
  simple_status,              /* status variables             */
  simple_system_variables,    /* system variables             */
  NULL,
  0
}
mysql_declare_plugin_end;
```

全文パーサープラグインの場合、タイプは MYSQL_FTPARSER_PLUGIN である必要があります。これは、FULLTEXT インデックスの作成時に WITH PARSER 句で使用する場合に、プラグインが正当なものであることを識別する値です。(ほかのプラグインタイプは、この句に対して正当ではありません。)

plugin.h では、mysql_declare_plugin() マクロおよび mysql_declare_plugin_end マクロを次のように定義します。

```
#ifndef MYSQL_DYNAMIC_PLUGIN
#define __MYSQL_DECLARE_PLUGIN(NAME, VERSION, PSIZE, DECLS) \
MYSQL_PLUGIN_EXPORT int VERSION= MYSQL_PLUGIN_INTERFACE_VERSION; \
MYSQL_PLUGIN_EXPORT int PSIZE= sizeof(struct st_mysql_plugin); \
MYSQL_PLUGIN_EXPORT struct st_mysql_plugin DECLS[]= {
#else
#define __MYSQL_DECLARE_PLUGIN(NAME, VERSION, PSIZE, DECLS) \
MYSQL_PLUGIN_EXPORT int _mysql_plugin_interface_version_= MYSQL_PLUGIN_INTERFACE_VERSION; \
MYSQL_PLUGIN_EXPORT int _mysql_sizeof_struct_st_plugin_= sizeof(struct st_mysql_plugin); \
MYSQL_PLUGIN_EXPORT struct st_mysql_plugin _mysql_plugin_declarations_[]= {
#endif

#define mysql_declare_plugin(NAME) \
__MYSQL_DECLARE_PLUGIN(NAME, \
            builtin_ ## NAME ## _plugin_interface_version, \
            builtin_ ## NAME ## _sizeof_struct_st_plugin, \
            builtin_ ## NAME ## _plugin)

#define mysql_declare_plugin_end ,{0,0,0,0,0,0,0,0,0,0,0,0,0,0}}
```

> **注記**
>
> これらの宣言では、MYSQL_DYNAMIC_PLUGIN シンボルを定義した場合にのみ、_mysql_plugin_interface_version_ シンボルを定義します。これは、プラグインを共有ライブラリとしてビルドするためのコンパイルコマンドの一部として -DMYSQL_DYNAMIC_PLUGIN を指定する必要があることを意味します。

上記のようにマクロが使用された場合は、次のコードが展開され、必要なシンボル (_mysql_plugin_interface_version_ および _mysql_plugin_declarations_) が定義されます。

```
int _mysql_plugin_interface_version_= MYSQL_PLUGIN_INTERFACE_VERSION;
int _mysql_sizeof_struct_st_plugin_= sizeof(struct st_mysql_plugin);
struct st_mysql_plugin _mysql_plugin_declarations_[]= {
{
  MYSQL_FTPARSER_PLUGIN,      /* type                         */
  &simple_parser_descriptor,  /* descriptor                   */
  "simple_parser",            /* name                         */
  "Oracle Corporation",       /* author                       */
  "Simple Full-Text Parser",  /* description                  */
  PLUGIN_LICENSE_GPL,         /* plugin license               */
  simple_parser_plugin_init,  /* init function (when loaded)  */
  simple_parser_plugin_deinit,/* deinit function (when unloaded) */
  0x0001,                     /* version                      */
  simple_status,              /* status variables             */
```

```
  simple_system_variables,    /* system variables                */
  NULL,
  0
}
  ,{0,0,0,0,0,0,0,0,0,0,0,0}}
};
```

前の例では一般ディスクリプタ内に単一のプラグインを宣言しましたが、複数のプラグインを宣言することもできます。mysql_declare_plugin() と mysql_declare_plugin_end の間に宣言をカンマで区切って順番にリストします。

MySQL サーバープラグインは、C または C++ (あるいは C の呼び出し規則を使用できる別の言語) で記述できます。C++ プラグインを記述する場合に使用するべきではない C++ 機能は、グローバル構造体を初期化するための非定数変数です。st_mysql_plugin 構造などの構造体のメンバーは、定数変数でのみ初期化してください。前に示した simple_parser ディスクリプタはこの要件を満たすため、C++ プラグインで許容されます。

```
mysql_declare_plugin(ftexample)
{
  MYSQL_FTPARSER_PLUGIN,      /* type                            */
  &simple_parser_descriptor,  /* descriptor                      */
  "simple_parser",            /* name                            */
  "Oracle Corporation",       /* author                          */
  "Simple Full-Text Parser",  /* description                     */
  PLUGIN_LICENSE_GPL,         /* plugin license                  */
  simple_parser_plugin_init,  /* init function (when loaded)     */
  simple_parser_plugin_deinit,/* deinit function (when unloaded) */
  0x0001,                     /* version                         */
  simple_status,              /* status variables                */
  simple_system_variables,    /* system variables                */
  NULL,
  0
}
mysql_declare_plugin_end;
```

一般ディスクリプタを記述するための別の有効な方法を次に示します。ここでは、プラグイン名、作成者、および説明を指定するために定数変数が使用されています。

```
const char *simple_parser_name = "simple_parser";
const char *simple_parser_author = "Oracle Corporation";
const char *simple_parser_description = "Simple Full-Text Parser";

mysql_declare_plugin(ftexample)
{
  MYSQL_FTPARSER_PLUGIN,      /* type                            */
  &simple_parser_descriptor,  /* descriptor                      */
  simple_parser_name,         /* name                            */
  simple_parser_author,       /* author                          */
  simple_parser_description,  /* description                     */
  PLUGIN_LICENSE_GPL,         /* plugin license                  */
  simple_parser_plugin_init,  /* init function (when loaded)     */
  simple_parser_plugin_deinit,/* deinit function (when unloaded) */
  0x0001,                     /* version                         */
  simple_status,              /* status variables                */
  simple_system_variables,    /* system variables                */
  NULL,
  0
}
mysql_declare_plugin_end;
```

ただし、次の一般ディスクリプタは無効です。ここではプラグイン名、作成者、および説明を指定するために構造体メンバーが使用されていますが、C++ では構造は定数イニシャライザと見なされません。

```
typedef struct
{
  const char *name;
  const char *author;
  const char *description;
} plugin_info;

plugin_info parser_info = {
  "simple_parser",
  "Oracle Corporation",
  "Simple Full-Text Parser"
};

mysql_declare_plugin(ftexample)
{
```

```
  MYSQL_FTPARSER_PLUGIN,      /* type                         */
  &simple_parser_descriptor,  /* descriptor                   */
  parser_info.name,           /* name                         */
  parser_info.author,         /* author                       */
  parser_info.description,    /* description                  */
  PLUGIN_LICENSE_GPL,         /* plugin license               */
  simple_parser_plugin_init,  /* init function (when loaded)  */
  simple_parser_plugin_deinit,/* deinit function (when unloaded) */
  0x0001,                     /* version                      */
  simple_status,              /* status variables             */
  simple_system_variables,    /* system variables             */
  NULL,
  0
}
mysql_declare_plugin_end;
```

サーバープラグインのステータス変数およびシステム変数

サーバープラグインインタフェースを使用すると、プラグインが一般プラグインディスクリプタの status_vars メンバーおよび system_vars メンバーを使用して、ステータス変数およびシステム変数を公開できます。

一般プラグインディスクリプタの status_vars メンバーは、0 でない場合、st_mysql_show_var 構造体の配列を指しており、各構造には 1 つのステータス変数が記述されていて、すべてのメンバーが 0 に設定された構造があとに続きます。st_mysql_show_var 構造体の定義は次のとおりです。

```
struct st_mysql_show_var {
  const char *name;
  char *value;
  enum enum_mysql_show_type type;
};
```

プラグインがインストールされると、プラグイン名と name 値がアンダースコアで結合されて、SHOW STATUS によって表示される名前が形成されます。

次の表は、許可されるステータス変数の type 値、および対応する変数を示しています。

表 24.1 サーバープラグインのステータス変数のタイプ

変数型	意味
SHOW_BOOL	ブール変数へのポインタ
SHOW_INT	整数変数へのポインタ
SHOW_LONG	long 整数変数へのポインタ
SHOW_LONGLONG	longlong 整数変数へのポインタ
SHOW_CHAR	文字列
SHOW_CHAR_PTR	文字列へのポインタ
SHOW_ARRAY	別の st_mysql_show_var 配列へのポインタ
SHOW_FUNC	関数へのポインタ
SHOW_DOUBLE	double へのポインタ

SHOW_FUNC タイプの場合、関数が呼び出されると関数は out パラメータに値を設定し、表示される変数に関する情報がこのパラメータによって提供されます。関数のシグネチャーは次のようになります。

```
#define SHOW_VAR_FUNC_BUFF_SIZE 1024

typedef int (*mysql_show_var_func) (void *thd,
                                    struct st_mysql_show_var *out,
                                    char *buf);
```

system_vars メンバーは、0 でない場合、st_mysql_sys_var 構造体の配列を指しており、各構造には 1 つのシステム変数が記述されていて (これはコマンド行または構成ファイルからも設定できます)、すべてのメンバーが 0 に設定された構造があとに続きます。st_mysql_sys_var 構造体は次のように定義します。

```
struct st_mysql_sys_var {
 int flags;
 const char *name, *comment;
 int (*check)(THD*, struct st_mysql_sys_var *, void*, st_mysql_value*);
```

```
 void (*update)(THD*, struct st_mysql_sys_var *, void*, const void*);
};
```

フラグによって、追加フィールドが必要に応じて付加されます。

利便性のため、プラグイン内の新しいシステム変数の作成をより簡単にするための多数のマクロが定義されています。

マクロでは次のフィールドを使用できます。

- name: システム変数の引用符で囲まれていない識別子。
- varname: 静的変数の識別子。識別子がない場合は name フィールドと同じです。
- opt: システム変数に追加で使用されるフラグ。次の表は、許容されるフラグを示しています。

表 24.2 サーバープラグインのシステム変数フラグ

フラグ値	説明
PLUGIN_VAR_READONLY	システム変数は読み取り専用である
PLUGIN_VAR_NOSYSVAR	システム変数は実行時にユーザーに表示されない
PLUGIN_VAR_NOCMDOPT	システム変数はコマンド行から構成できない
PLUGIN_VAR_NOCMDARG	コマンド行に引数は不要である (通常はブール変数に使用します)
PLUGIN_VAR_RQCMDARG	コマンド行に引数が必要である (これはデフォルトです)
PLUGIN_VAR_OPCMDARG	コマンド行の引数はオプションである
PLUGIN_VAR_MEMALLOC	文字列変数に使用され、文字列の格納にメモリーが割り当てられることを示す

- comment: サーバーヘルプメッセージに表示される説明コメント。この変数が非表示の場合は NULL です。
- check: チェック関数。デフォルトは NULL です。
- update: 更新関数。デフォルトは NULL です。
- default: 変数のデフォルト値。
- minimum: 変数の最小値。
- maximum: 変数の最大値。
- blocksize: 変数のブロックサイズ。値が設定されると、もっとも近い blocksize の倍数に丸められます。

システム変数には、静的変数を使用して直接アクセスするか、SYSVAR() アクセス機能マクロを使用してアクセスできます。完全性を期すために、SYSVAR() マクロが提供されています。通常、これはコードがベースとなる変数に直接アクセスできない場合にのみ使用してください。

例:

```
static int my_foo;
static MYSQL_SYSVAR_INT(foo_var, my_foo,
                        PLUGIN_VAR_RQCMDARG, "foo comment",
                        NULL, NULL, 0, 0, INT_MAX, 0);
...
  SYSVAR(foo_var)= value;
  value= SYSVAR(foo_var);
  my_foo= value;
  value= my_foo;
```

セッション変数には THDVAR() アクセス機能マクロを介してのみアクセスできます。例:

```
static MYSQL_THDVAR_BOOL(some_flag,
                         PLUGIN_VAR_NOCMDARG, "flag comment",
                         NULL, NULL, FALSE);
...
  if (THDVAR(thd, some_flag))
  {
    do_something();
    THDVAR(thd, some_flag)= FALSE;
  }
```

すべてのグローバルシステム変数およびセッションシステム変数は、使用する前に mysqld に公開する必要があります。これは、NULL で終わる変数の配列を作成し、プラグインパブリックインタフェースでそれにリンクすることによって行うことができます。例:

```
static struct st_mysql_sys_var *my_plugin_vars[]= {
  MYSQL_SYSVAR(foo_var),
  MYSQL_SYSVAR(some_flag),
  NULL
};
mysql_declare_plugin(fooplug)
{
  MYSQL_..._PLUGIN,
  &plugin_data,
  "fooplug",
  "foo author",
  "This does foo!",
  PLUGIN_LICENSE_GPL,
  foo_init,
  foo_fini,
  0x0001,
  NULL,
  my_plugin_vars,
  NULL,
  0
}
mysql_declare_plugin_end;
```

次の支援マクロを使用すると、さまざまなタイプのシステム変数を宣言できます。

- 1 バイトのブール値 (0 = FALSE、1 = TRUE) である my_bool 型のブールシステム変数。

  ```
  MYSQL_THDVAR_BOOL(name, opt, comment, check, update, default)
  MYSQL_SYSVAR_BOOL(name, varname, opt, comment, check, update, default)
  ```

- NULL で終わる文字列へのポインタである char* 型の文字列システム変数。

  ```
  MYSQL_THDVAR_STR(name, opt, comment, check, update, default)
  MYSQL_SYSVAR_STR(name, varname, opt, comment, check, update, default)
  ```

- 各種の整数システム変数。

  - 通常は 4 バイトの符号付きワードである int システム変数。

    ```
    MYSQL_THDVAR_INT(name, opt, comment, check, update, default, min, max, blk)
    MYSQL_SYSVAR_INT(name, varname, opt, comment, check, update, default,
                     minimum, maximum, blocksize)
    ```

  - 通常は 4 バイトの符号なしワードである unsigned int システム変数。

    ```
    MYSQL_THDVAR_UINT(name, opt, comment, check, update, default, min, max, blk)
    MYSQL_SYSVAR_UINT(name, varname, opt, comment, check, update, default,
                      minimum, maximum, blocksize)
    ```

  - 通常は 4 バイトまたは 8 バイトの符号付きワードである long システム変数。

    ```
    MYSQL_THDVAR_LONG(name, opt, comment, check, update, default, min, max, blk)
    MYSQL_SYSVAR_LONG(name, varname, opt, comment, check, update, default,
                      minimum, maximum, blocksize)
    ```

  - 通常は 4 バイトまたは 8 バイトの符号なしワードである unsigned long システム変数。

    ```
    MYSQL_THDVAR_ULONG(name, opt, comment, check, update, default, min, max, blk)
    MYSQL_SYSVAR_ULONG(name, varname, opt, comment, check, update, default,
                       minimum, maximum, blocksize)
    ```

  - 通常は 8 バイトの符号付きワードである long long システム変数。

    ```
    MYSQL_THDVAR_LONGLONG(name, opt, comment, check, update,
                          default, minimum, maximum, blocksize)
    MYSQL_SYSVAR_LONGLONG(name, varname, opt, comment, check, update,
                          default, minimum, maximum, blocksize)
    ```

  - 通常は 8 バイトの符号なしワードである unsigned long long システム変数。

    ```
    MYSQL_THDVAR_ULONGLONG(name, opt, comment, check, update,
                           default, minimum, maximum, blocksize)
    MYSQL_SYSVAR_ULONGLONG(name, varname, opt, comment, check, update,
    ```

```
                default, minimum, maximum, blocksize)
```

- 通常は 4 バイトまたは 8 バイトの符号なしワードである unsigned long システム変数。設定可能な値の範囲は typelib 内の要素の数の序数であり、0 から始まります。

```
MYSQL_THDVAR_ENUM(name, opt, comment, check, update, default, typelib)
MYSQL_SYSVAR_ENUM(name, varname, opt, comment, check, update,
                  default, typelib)
```

- 通常は 8 バイトの符号なしワードである unsigned long long システム変数。各ビットは typelib 内の要素を表します。

```
MYSQL_THDVAR_SET(name, opt, comment, check, update, default, typelib)
MYSQL_SYSVAR_SET(name, varname, opt, comment, check, update,
                 default, typelib)
```

内部的には、変更されることがあるすべてのプラグインシステム変数は HASH 構造体に格納されます。

サーバーのコマンド行でのヘルプテキストの表示は、コマンド行オプションに関係するすべての変数の DYNAMIC_ARRAY をコンパイルしてそれらをソートし、それを繰り返して各オプションを表示することによって処理されます。

コマンド行オプションが処理されると、handle_option() 関数 (my_getopt.c) によって argv から削除され、実質的にその役割が終わります。

サーバーはプラグインのインストール処理中 (プラグインが正常にロードされた直後、かつプラグインの初期化関数が呼び出される前) にコマンド行オプションを処理します。

実行時にロードされるプラグインは構成オプションを利用できないため、使用可能なデフォルト値がある必要があります。プラグインをインストールすると、mysqld の初期化時にロードされ、構成オプションをコマンド行または my.cnf 内に設定できます。

プラグインで thd パラメータを読み取り専用にすることを検討してください。

クライアントプラグインディスクリプタ

各クライアントプラグインには、クライアントプラグイン API に情報を提供するディスクリプタが必要となります。ディスクリプタの構造体は、すべてのクライアントプラグインに共通する固定された一連のメンバーで始まり、プラグインタイプ固有のメンバーがあとに続きます。

client_plugin.h ファイル内の st_mysql_client_plugin 構造体は、共通メンバーが含まれている「一般的な」ディスクリプタを定義します。

```
struct st_mysql_client_plugin
{
  int type;
  unsigned int interface_version;
  const char *name;
  const char *author;
  const char *desc;
  unsigned int version[3];
  const char *license;
  void *mysql_api;
  int (*init)(char *, size_t, int, va_list);
  int (*deinit)();
  int (*options)(const char *option, const void *);
};
```

st_mysql_client_plugin ディスクリプタ構造体の共通メンバーは、次のように使用します。char * メンバーは、NULL で終わる文字列として指定してください。

- type: プラグインの型。これは client_plugin.h のプラグインタイプ値 (MYSQL_CLIENT_AUTHENTICATION_PLUGIN など) のいずれかである必要があります。
- interface_version: プラグインインタフェースのバージョン。たとえば、認証プラグインの場合、これは MYSQL_CLIENT_AUTHENTICATION_PLUGIN_INTERFACE_VERSION です。
- name: プラグイン名を指定する文字列。これは、MYSQL_DEFAULT_AUTH オプションを指定して mysql_options() を呼び出すか、MySQL クライアントプログラムに --default-auth オプションを指定するときにプラグインを参照するための名前です。

- author: プラグイン作成者を示す文字列。これには任意の文字列を指定できます。
- desc: プラグインの概要を説明する文字列。これには任意の文字列を指定できます。
- version: メジャー、マイナー、およびその下のバージョンを示す 3 つの整数の配列であるプラグインバージョン。たとえば、{1,2,3} はバージョン 1.2.3 を示します。
- license: ライセンスタイプを指定する文字列。
- mysql_api: 内部で使用されます。プラグインディスクリプタでは NULL に指定します。
- init: 1 回だけ実行される初期化関数であり、そのような関数がない場合は NULL です。クライアントライブラリはプラグインをロードするときにこの関数を実行します。関数は、成功した場合はゼロ、および失敗した場合はゼロ以外を返します。

  エラーが発生した場合、init 関数は最初の 2 つの引数を使用してエラーメッセージを返します。最初の引数は char バッファーへのポインタであり、2 番目の引数はバッファー長を示します。init 関数によって返されるすべてのメッセージは NULL で終わる必要があるため、最大のメッセージ長はバッファー長から 1 を引いた長さです。それに続く引数が mysql_load_plugin() に渡されます。先頭の引数は、以降に存在する引数の数を示し (ない場合は 0)、そのあとに残りの引数が続きます。
- deinit: 1 回だけ実行される初期化解除関数であり、そのような関数がない場合は NULL です。クライアントライブラリは、プラグインをアンロードするときにこの関数を実行します。この関数は引数を受け取りません。成功した場合はゼロ、および失敗した場合はゼロ以外を返します。
- options: プラグインに渡されるオプションを処理するための関数であり、そのような関数がない場合は NULL です。この関数は、オプション名およびその値へのポインタを表す 2 つの引数を受け取ります。関数は、成功した場合はゼロ、および失敗した場合はゼロ以外を返します。

特定のクライアントプラグインタイプでは、共通のディスクリプタメンバーのあとに、そのタイプのプラグインを実装するために必要な追加メンバーが続くことがあります。たとえば、認証プラグインの st_mysql_client_plugin_AUTHENTICATION 構造体には、認証を実行するためにクライアントライブラリが呼び出す関数が最後にあります。

プラグインを宣言するには、mysql_declare_client_plugin() マクロおよび mysql_end_client_plugin マクロを使用します。

```
mysql_declare_client_plugin(plugin_type)
  ... members common to all client plugins ...
  ... type-specific extra members ...
mysql_end_client_plugin;
```

type または interface_version メンバーを明示的に指定しないでください。mysql_declare_client_plugin() マクロは plugin_type 引数を使用して、これらの値を自動的に生成します。たとえば、認証クライアントプラグインは次のように宣言します。

```
mysql_declare_client_plugin(AUTHENTICATION)
 "my_auth_plugin",
 "Author Name",
 "My Client Authentication Plugin",
 {1,0,0},
 "GPL",
 NULL,
 my_auth_init,
 my_auth_deinit,
 my_auth_options,
 my_auth_main
mysql_end_client_plugin;
```

この宣言では、AUTHENTICATION 引数を使用して、type および interface_version メンバーに MYSQL_CLIENT_AUTHENTICATION_PLUGIN および MYSQL_CLIENT_AUTHENTICATION_PLUGIN_INTERFACE_VERSION を設定しています。

プラグインタイプによっては、ディスクリプタの共通メンバーに続いてほかのメンバーがあることがあります。たとえば、認証プラグインの場合、サーバーとの通信を処理する関数 (上記のディスクリプタでは my_auth_main()) があります。セクション24.2.4.9「認証プラグインの作成」を参照してください。

通常、認証プラグインの使用をサポートしているクライアントプログラムは、mysql_options() を呼び出して MYSQL_DEFAULT_AUTH オプションおよび MYSQL_PLUGIN_DIR オプションを設定することによって、プラグインをロードします。

```
char *plugin_dir = "path_to_plugin_dir";
char *default_auth = "plugin_name";

/* ... process command-line options ... */

mysql_options(&mysql, MYSQL_PLUGIN_DIR, plugin_dir);
mysql_options(&mysql, MYSQL_DEFAULT_AUTH, default_auth);
```

一般にプログラムは、ユーザーがデフォルト値をオーバーライドできるようにする --plugin-dir および --default-auth オプションも受け付けます。

クライアントプログラムで低レベルのプラグイン管理が必要である場合は、クライアントライブラリに st_mysql_client_plugin 引数を受け取る関数を含めます。セクション23.7.14「C API クライアントプラグイン関数」を参照してください。

24.2.4.3 プラグインライブラリのコンパイルおよびインストール

プラグインを記述したら、プラグインをコンパイルしてインストールする必要があります。共有オブジェクトをコンパイルする手順はシステムによって異なります。CMake を使用してライブラリをビルドする場合は、それがシステムの正しいコンパイルコマンドを生成できる必要があります。ライブラリに somepluglib という名前を指定した場合、共有オブジェクトファイルの名前は somepluglib.so のようになります。(使用しているシステムではファイル名のサフィクスが異なる場合もあります。)

CMake を使用する場合は、構成ファイルをセットアップして、プラグインがコンパイルおよびインストールされるようにする必要があります。MySQL ソース配布の plugin ディレクトリの下にあるプラグインの例をガイドとして使用してください。

次のような CMakeLists.txt を作成します。

```
MYSQL_ADD_PLUGIN(somepluglib somepluglib.c
  MODULE_ONLY MODULE_OUTPUT_NAME "somepluglib")
```

CMake が Makefile を生成するとき、コンパイルコマンドに -DMYSQL_DYNAMIC_PLUGIN フラグを渡し、リンカーに -lmysqlservices フラグを渡します。これは、プラグインサービスインタフェースを介して提供されるサービスの関数でリンクするために必要となります。セクション24.2.5「プラグインのための MySQL サービス」を参照してください。

CMake を実行してから make を実行します。

```
shell> cmake .
shell> make
```

CMake の構成オプションを指定する必要がある場合、リストについてはセクション2.9.4「MySQL ソース構成オプション」を参照してください。たとえば、プラグインのインストール先となる MySQL のベースディレクトリを指定するために CMAKE_INSTALL_PREFIX を指定する場合があります。このオプションに使用する値は SHOW VARIABLES で確認できます。

```
mysql> SHOW VARIABLES LIKE 'basedir';
+---------------+-----------------+
| Variable_name | Value           |
+---------------+-----------------+
| base          | /usr/local/mysql |
+---------------+-----------------+
```

ライブラリをインストールするプラグインディレクトリの場所は、plugin_dir システム変数によって指定されます。例:

```
mysql> SHOW VARIABLES LIKE 'plugin_dir';
+---------------+---------------------------------+
| Variable_name | Value                           |
+---------------+---------------------------------+
| plugin_dir    | /usr/local/mysql/lib/mysql/plugin |
+---------------+---------------------------------+
```

プラグインライブラリをインストールするには、make を使用します。

```
shell> make install
```

make install によってプラグインライブラリが適切なディレクトリにインストールされたことを確認します。インストール後、サーバーが実行するためのアクセス権がライブラリに設定されていることを確認してください。

24.2.4.4 全文パーサープラグインの作成

MySQL 5.6 では、MyISAM でのみ全文パーサープラグインがサポートされます。全文パーサープラグインについての概要は、セクション24.2.3.2「全文パーサープラグイン」を参照してください。

全文パーサーサーバープラグインは、組み込みの全文パーサーの代わりにするか、それを変更するために使用できます。このセクションでは、simple_parser という名前の全文パーサープラグインを記述する方法について説明します。このプラグインは、MySQL の組み込み全文パーサーによって使用されるものよりも簡単なルールに基づいて構文解析を実行し、単語は空白文字を含まない文字列です。

この手順では、MySQL ソース配布の plugin/fulltext ディレクトリ内のソースコードを使用しているため、このディレクトリに場所を変更してください。次の手順では、プラグインライブラリを作成する方法について説明します。

1. 全文パーサープラグインを記述するために、プラグインソースファイルで次のヘッダーファイルをインクルードします。プラグインの機能および要件によっては、ほかの MySQL のヘッダーファイルまたは一般的なヘッダーファイルが必要になることもあります。

   ```
   #include <mysql/plugin.h>
   ```

   plugin.h は、MYSQL_FTPARSER_PLUGIN サーバープラグインタイプ、およびプラグインを宣言するために必要なデータ構造体を定義します。

2. プラグインライブラリファイルのライブラリディスクリプタをセットアップします。

   このディスクリプタには、サーバープラグインの一般プラグインディスクリプタが含まれています。全文パーサープラグインの場合、タイプは MYSQL_FTPARSER_PLUGIN である必要があります。これは、FULLTEXT インデックスの作成時に WITH PARSER 句で使用する場合に、プラグインが正当なものであることを識別する値です。(ほかのプラグインタイプは、この句に対して正当ではありません。)

   たとえば、simple_parser という名前の単一の全文パーサープラグインを格納するライブラリのライブラリディスクリプタは、次のようになります。

   ```
   mysql_declare_plugin(ftexample)
   {
     MYSQL_FTPARSER_PLUGIN,      /* type                         */
     &simple_parser_descriptor,  /* descriptor                   */
     "simple_parser",            /* name                         */
     "Oracle Corporation",       /* author                       */
     "Simple Full-Text Parser",  /* description                  */
     PLUGIN_LICENSE_GPL,         /* plugin license               */
     simple_parser_plugin_init,  /* init function (when loaded)  */
     simple_parser_plugin_deinit,/* deinit function (when unloaded) */
     0x0001,                     /* version                      */
     simple_status,              /* status variables             */
     simple_system_variables,    /* system variables             */
     NULL,
     0
   }
   mysql_declare_plugin_end;
   ```

   name メンバー (simple_parser) は、INSTALL PLUGIN、UNINSTALL PLUGIN などのステートメントでプラグインを参照するために使用する名前を示しています。これは、SHOW PLUGINS または INFORMATION_SCHEMA.PLUGINS によって表示される名前でもあります。

   詳細は、サーバープラグインライブラリおよびプラグインディスクリプタを参照してください。

3. タイプ固有のプラグインディスクリプタをセットアップします。

   ライブラリディスクリプタの各一般プラグインディスクリプタは、タイプ固有のディスクリプタを指しています。全文パーサープラグインの場合、タイプ固有のディスクリプタは plugin.h ファイル内の st_mysql_ftparser 構造体のインスタンスです。

   ```
   struct st_mysql_ftparser
   {
     int interface_version;
     int (*parse)(MYSQL_FTPARSER_PARAM *param);
     int (*init)(MYSQL_FTPARSER_PARAM *param);
     int (*deinit)(MYSQL_FTPARSER_PARAM *param);
   };
   ```

構造体定義に示されているように、ディスクリプタにはインタフェースバージョン番号があり、3 つの関数へのポインタが含まれています。

インタフェースのバージョン番号はシンボルを使用して指定し、MYSQL_xxx_INTERFACE_VERSION という形式です。全文パーサープラグインの場合、シンボルは「MYSQL_FTPARSER_INTERFACE_VERSION」です。ソースコードには、include/mysql/plugin_ftparser.h に定義されている全文パーサープラグインの実際のインタフェースバージョン番号があります。

init メンバーおよび deinit メンバーは、関数を指すようにするか、関数が不要な場合は 0 に設定します。parse メンバーは、構文解析を実行する関数を指している必要があります。

simple_parser 宣言では、そのディスクリプタは &simple_parser_descriptor によって示されています。ディスクリプタは、全文プラグインインタフェースのバージョン番号 (MYSQL_FTPARSER_INTERFACE_VERSION によって指定されます)、プラグインの構文解析関数、初期化関数、および初期化解除関数を指定します。

```
static struct st_mysql_ftparser simple_parser_descriptor=
{
  MYSQL_FTPARSER_INTERFACE_VERSION, /* interface version      */
  simple_parser_parse,              /* parsing function      */
  simple_parser_init,               /* parser init function  */
  simple_parser_deinit              /* parser deinit function */
};
```

全文パーサーサーバープラグインは、インデックス設定と検索の 2 つの異なるコンテキストで使用されます。両方のコンテキストで、サーバーはプラグインを呼び出した各 SQL ステートメントの処理の開始時と終了時に初期化関数および初期化解除関数を呼び出します。ただし、ステートメントの処理中に、サーバーはコンテキストに固有の方法でメインの構文解析関数を呼び出します。

- インデックス設定の場合、サーバーはインデックス設定される各カラム値に対してパーサーを呼び出します。
- 検索の場合、サーバーは検索文字列を構文解析するためにパーサーを呼び出します。パーサーはステートメントによって処理される行に対して呼び出されることもあります。自然言語モードでは、サーバーがパーサーを呼び出す必要はありません。クエリー拡張を使用したブールモードフレーズ検索または自然言語検索の場合、インデックスにない情報についてカラム値を構文解析するためにパーサーが使用されます。また、FULLTEXT インデックスのないカラムに対してブールモード検索が実行された場合は、組み込みパーサーが呼び出されます。(プラグインは特定のインデックスに関連付けられます。インデックスがない場合、プラグインは使用されません。)

一般プラグインディスクリプタのプラグイン宣言には、初期化関数および初期化解除関数を指している init メンバーおよび deinit メンバーがあり、指している先のタイプ固有のプラグインディスクリプタにもこれらがあります。ただし、これらの関数のペアは目的が異なり、異なる理由で呼び出されます。

- 一般プラグインディスクリプタのプラグイン宣言の場合、初期化関数および初期化解除関数は、プラグインがロードおよびアンロードされるときに呼び出されます。
- タイプ固有のプラグインディスクリプタの場合、初期化関数および初期化解除関数は、プラグインが使用される SQL ステートメントごとに呼び出されます。

プラグインディスクリプタに指定されている各インタフェース関数は、成功の場合はゼロ、および失敗の場合はゼロ以外を返し、構文解析するコンテキストが含まれている MYSQL_FTPARSER_PARAM 構造体を指している引数を受け取ります。この構造体の定義は次のとおりです。

```
typedef struct st_mysql_ftparser_param
{
  int (*mysql_parse)(struct st_mysql_ftparser_param *,
                     char *doc, int doc_len);
  int (*mysql_add_word)(struct st_mysql_ftparser_param *,
                        char *word, int word_len,
                        MYSQL_FTPARSER_BOOLEAN_INFO *boolean_info);
  void *ftparser_state;
  void *mysql_ftparam;
  struct charset_info_st *cs;
  char *doc;
  int length;
  int flags;
  enum enum_ftparser_mode mode;
```

```
} MYSQL_FTPARSER_PARAM;
```

構造体メンバーは次のように使用されます。

- mysql_parse: サーバーの組み込みパーサーを呼び出すコールバック関数へのポインタ。プラグインが組み込みパーサーのフロントエンドとして動作する場合は、このコールバックを使用します。つまり、プラグインの構文解析関数が呼び出されたとき、その関数は入力を処理してテキストを抽出し、テキストを mysql_parse コールバックに渡します。

  このコールバック関数の最初のパラメータは、param 値自体となります。

  ```
  param->mysql_parse(param, ...);
  ```

  フロントエンドプラグインは、テキストを抽出して一度にすべてのテキストを組み込みパーサーに渡すか、テキストを抽出して一度にテキストの一部分ずつを組み込みパーサーに渡すことができます。ただし、この場合、組み込みサーバーはテキストの一部分同士の間に暗黙的な単語の分割があるかのように処理します。

- mysql_add_word: 全文インデックスまたは検索語のリストに単語を追加するコールバック関数へのポインタ。このコールバックは、組み込みパーサーをパーサープラグインに置き換えた場合に使用します。つまり、プラグインの構文解析関数が呼び出されたとき、その関数が入力を単語に構文解析し、各単語について mysql_add_word コールバックが呼び出されます。

  このコールバック関数の最初のパラメータは、param 値自体となります。

  ```
  param->mysql_add_word(param, ...);
  ```

- ftparser_state: これは一般的なポインタです。プラグインは独自の目的のために内部で使用される情報を指すようにこれを設定できます。

- mysql_ftparam: これはサーバーによって設定されます。これは最初の引数として mysql_parse または mysql_add_word コールバックに渡されます。

- cs: テキストの文字セットについての情報へのポインタであり、情報がない場合は 0 です。

- doc: 構文解析されるテキストへのポインタ。

- length: 構文解析されるテキストのバイト単位の長さ。

- flags: パーサーのフラグ。特別なフラグがない場合、これはゼロです。現時点で、ゼロ以外のフラグは MYSQL_FTFLAGS_NEED_COPY のみであり、これは mysql_add_word() が単語のコピーを保存する必要がある (つまり、単語は上書きされるバッファー内にあるため、単語へのポインタを使用できない) ことを意味します。このメンバーは MySQL 5.1.12 で追加されました。

  このフラグは、MySQL (パーサープラグインを呼び出す前)、パーサープラグイン自体、または mysql_parse() 関数が設定またはリセットできます

- mode: 構文解析のモード。この値は次の定数のいずれかです。

  - MYSQL_FTPARSER_SIMPLE_MODE: 単純で高速なモードで構文解析し、インデックス設定および自然言語クエリーに使用されます。パーサーは、インデックスを設定する単語のみをサーバーに渡します。パーサーが長さ制限またはストップワードリストを使用して無視する単語を判別する場合、パーサーはそのような単語をサーバーに渡しません。

  - MYSQL_FTPARSER_WITH_STOPWORDS: ストップワードモードで構文解析します。これはフレーズ照合のためのブール検索で使用されます。パーサーは、ストップワードや通常の長さ制限の範囲外にある単語であっても、すべての単語をサーバーに渡します。

  - MYSQL_FTPARSER_FULL_BOOLEAN_INFO: ブールモードで構文解析します。これはブールクエリー文字列の構文解析に使用されます。パーサーは単語だけでなくブールモード演算子も認識し、mysql_add_word コールバックを使用してこれらをトークンとしてサーバーに渡します。渡されるトークンの種類をサーバーに通知するために、プラグインは MYSQL_FTPARSER_BOOLEAN_INFO 構造体に値を設定して、この構造体へのポインタを渡す必要があります。

パーサーがブールモードで呼び出された場合、param->mode 値は MYSQL_FTPARSER_FULL_BOOLEAN_INFO になります。サーバーにトークン情報を渡すためにパーサーが使用する MYSQL_FTPARSER_BOOLEAN_INFO 構造体は次のようになります。

```
typedef struct st_mysql_ftparser_boolean_info
{
```

```
  enum enum_ft_token_type type;
  int yesno;
  int weight_adjust;
  char wasign;
  char trunc;
  /* These are parser state and must be removed. */
  char prev;
  char *quot;
} MYSQL_FTPARSER_BOOLEAN_INFO;
```

パーサーは構造体メンバーに次のように値を設定します。

- type: トークンのタイプ。許容されるタイプを次の表に示します。

  表 24.3 全文パーサーのトークンタイプ

トークン値	意味
FT_TOKEN_EOF	データの終わり
FT_TOKEN_WORD	通常の単語
FT_TOKEN_LEFT_PAREN	グループまたは部分式の始まり
FT_TOKEN_RIGHT_PAREN	グループまたは部分式の終わり
FT_TOKEN_STOPWORD	ストップワード

- yesno: 一致が発生するために単語が存在する必要があるかどうか。0 の場合、単語はオプションでですが、存在する場合は一致の関連性が増加することを意味します。0 より大きい値は、単語が存在する必要があることを意味します。0 より小さい値は、単語が存在していない必要があることを意味します。
- weight_adjust: 単語の一致の重要度を決定する重み付けの要素。これは、関連性計算での単語の重要性を増加または減少させるために使用できます。値がゼロの場合は、重み付けによる調整がないことを示します。ゼロより大きい値または小さい値は、重みが大きいことまたは小さいことをそれぞれ意味します。< および > 演算子を使用するセクション12.9.2「ブール全文検索」の例は、重み付けがどのように機能するかを示しています。
- wasign: 重み付け要素の符号。負の値は ~ ブール検索演算子のように作用し、これによって、関連性に対する単語の貢献度がマイナスになります。
- trunc: ブールモードの * 切り捨て演算子が指定された場合のように、照合を実行するかどうか。

プラグインで、MYSQL_FTPARSER_BOOLEAN_INFO 構造体の prev メンバーおよび quot メンバーを使用しないでください。

> **注記**
>
> ブール演算子 @ distance は現在のプラグインパーサーフレームワークによってサポートされません。ブール全文検索の演算子については、セクション12.9.2「ブール全文検索」を参照してください。

4. プラグインインタフェース関数をセットアップします。

   ライブラリディスクリプタ内の一般プラグインディスクリプタは、サーバーがプラグインをロードおよびアンロードするときに呼び出す初期化関数および初期化解除関数を指定します。simple_parser の場合、これらの関数は何も行いませんが、成功したことを示すためにゼロを返します。

```
static int simple_parser_plugin_init(void *arg __attribute__((unused)))
{
  return(0);
}

static int simple_parser_plugin_deinit(void *arg __attribute__((unused)))
{
  return(0);
```

```
}
```

これらの関数は実際に何も行わないため、これらを省略して、プラグイン宣言でそれぞれに 0 を指定できます。

simple_parser のタイプ固有のプラグインディスクリプタは、プラグインが使用されるときにサーバーが呼び出す初期化関数、初期化解除関数、および構文解析関数を指定します。simple_parser の場合、初期化関数および初期化解除関数は何も行いません。

```
static int simple_parser_init(MYSQL_FTPARSER_PARAM *param
                              __attribute__((unused)))
{
  return(0);
}

static int simple_parser_deinit(MYSQL_FTPARSER_PARAM *param
                                __attribute__((unused)))
{
  return(0);
}
```

ここでも、これらの関数は何も行わないため、これらを省略して、プラグインディスクリプタでそれぞれに 0 を指定できます。

メインの構文解析関数 simple_parser_parse() は組み込みの全文パーサーの代わりに動作するため、テキストを単語に分割して各単語をサーバーに渡す必要があります。構文解析関数の最初の引数は、構文解析するコンテキストが含まれている構造体へのポインタです。この構造体には、構文解析されるテキストを指している doc メンバー、およびテキストの長さを示す length メンバーがあります。このプラグインによって実行される単純な構文解析では、空白文字を含まない文字列を単語とみなすため、次のように単語を識別します。

```
static int simple_parser_parse(MYSQL_FTPARSER_PARAM *param)
{
  char *end, *start, *docend= param->doc + param->length;

  for (end= start= param->doc;; end++)
  {
    if (end == docend)
    {
      if (end > start)
        add_word(param, start, end - start);
      break;
    }
    else if (isspace(*end))
    {
      if (end > start)
        add_word(param, start, end - start);
      start= end + 1;
    }
  }
  return(0);
}
```

パーサーは単語を検出するごとに、関数 add_word() を呼び出して単語をサーバーに渡します。add_word() はヘルパー関数にすぎず、プラグインインタフェースの一部ではありません。パーサーは、構文解析されるコンテキストへのポインタ、単語へのポインタ、および長さの値を add_word() に渡します。

```
static void add_word(MYSQL_FTPARSER_PARAM *param, char *word, size_t len)
{
  MYSQL_FTPARSER_BOOLEAN_INFO bool_info=
    { FT_TOKEN_WORD, 0, 0, 0, 0, ' ', 0 };

  param->mysql_add_word(param, word, len, &bool_info);
}
```

ブールモード構文解析の場合、前に st_mysql_ftparser_boolean_info 構造体の説明で示したように、add_word() は bool_info 構造体のメンバーに値を設定します。

5. ステータス変数をセットアップします。simple_parser プラグインの場合、次のステータス変数の配列は、1 つのステータス変数を静的テキストの値で設定し、別のステータス変数を long 整数変数に格納されている値で設定しています。

```
long number_of_calls= 0;

struct st_mysql_show_var simple_status[]=
{
```

```
  {"static", (char *)"just a static text", SHOW_CHAR},
  {"called", (char *)&number_of_calls,     SHOW_LONG},
  {0,0,0}
};
```

プラグインがインストールされると、プラグイン名と name 値がアンダースコアで結合されて、SHOW STATUS によって表示される名前が形成されます。上記の配列の場合、生成されるステータス変数名は simple_parser_static および simple_parser_called です。この規則は、プラグインの変数をプラグイン名を使用して簡単に表示できることを意味します。

```
mysql> SHOW STATUS LIKE 'simple_parser%';
+----------------------+--------------------+
| Variable_name        | Value              |
+----------------------+--------------------+
| simple_parser_static | just a static text |
| simple_parser_called | 0                  |
+----------------------+--------------------+
```

6. プラグインライブラリのオブジェクトファイルをコンパイルおよびインストールするには、セクション 24.2.4.3「プラグインライブラリのコンパイルおよびインストール」の手順を使用します。ライブラリファイルを使用するには、ライブラリファイルがプラグインディレクトリ (plugin_dir システム変数で指定されているディレクトリ) にインストールされている必要があります。simple_parser プラグインの場合、ソースから MySQL をビルドするときにコンパイルおよびインストールされます。これはバイナリ配布にも含められます。ビルド処理では、mypluglib.so という名前の共有オブジェクトライブラリが生成されます (サフィクスはプラットフォームによって異なる場合があります)。

7. プラグインを使用するには、プラグインをサーバーに登録します。たとえば、プラグインを実行時に登録するには次のステートメントを使用します (必要に応じてサフィクスを変更します)。

   ```
   mysql> INSTALL PLUGIN simple_parser SONAME 'mypluglib.so';
   ```

   プラグインのロードについての追加情報は、セクション5.1.8.1「プラグインのインストールおよびアンインストール」を参照してください。

8. プラグインのインストールを検証するには、INFORMATION_SCHEMA.PLUGINS テーブルを調査するか、SHOW PLUGINS ステートメントを使用します。

9. プラグインをテストして、正しく動作することを確認します。

   文字列カラムを含むテーブルを作成し、パーサープラグインをそのカラムの FULLTEXT インデックスに関連付けます。

   ```
   mysql> CREATE TABLE t (c VARCHAR(255),
     ->   FULLTEXT (c) WITH PARSER simple_parser
     -> ) ENGINE=MyISAM;
   Query OK, 0 rows affected (0.01 sec)
   ```

   このテーブルにいくつかのテキストを挿入して検索を実行します。これらにより、パーサープラグインは空白文字でないすべての文字を単語文字として処理することが確認されます。

   ```
   mysql> INSERT INTO t VALUES
     ->   ('latin1_general_cs is a case-sensitive collation'),
     ->   ('I\'d like a case of oranges'),
     ->   ('this is sensitive information'),
     ->   ('another row'),
     ->   ('yet another row');
   Query OK, 5 rows affected (0.02 sec)
   Records: 5  Duplicates: 0  Warnings: 0

   mysql> SELECT c FROM t;
   +-------------------------------------------------+
   | c                                               |
   +-------------------------------------------------+
   | latin1_general_cs is a case-sensitive collation |
   | I'd like a case of oranges                      |
   | this is sensitive information                   |
   | another row                                     |
   | yet another row                                 |
   +-------------------------------------------------+
   5 rows in set (0.00 sec)

   mysql> SELECT MATCH(c) AGAINST('case') FROM t;
   +-------------------------+
   | MATCH(c) AGAINST('case') |
   +-------------------------+
   ```

```
|                     0 |
|        1.2968142032623 |
|                     0 |
|                     0 |
|                     0 |
+------------------------+
5 rows in set (0.00 sec)

mysql> SELECT MATCH(c) AGAINST('sensitive') FROM t;
+-------------------------------+
| MATCH(c) AGAINST('sensitive') |
+-------------------------------+
|                             0 |
|                             0 |
|             1.3253291845322 |
|                             0 |
|                             0 |
+-------------------------------+
5 rows in set (0.01 sec)

mysql> SELECT MATCH(c) AGAINST('case-sensitive') FROM t;
+-----------------------------------+
| MATCH(c) AGAINST('case-sensitive') |
+-----------------------------------+
|                 1.3109166622162 |
|                                 0 |
|                                 0 |
|                                 0 |
|                                 0 |
+-----------------------------------+
5 rows in set (0.01 sec)

mysql> SELECT MATCH(c) AGAINST('l\'d') FROM t;
+------------------------+
| MATCH(c) AGAINST('l\'d') |
+------------------------+
|                     0 |
|        1.2968142032623 |
|                     0 |
|                     0 |
|                     0 |
+------------------------+
5 rows in set (0.01 sec)
```

組み込みパーサーとは異なり、「case」および「insensitive」は「case-insensitive」と一致していません。

24.2.4.5 デーモンプラグインの作成

デーモンプラグインは、サーバーによって実行されるがサーバーと通信しないコードに使用される単純なタイプのプラグインです。このセクションでは、MySQL ソース配布の plugin/daemon_example ディレクトリにあるプラグインの例を使用して、デーモンサーバープラグインを作成する方法について説明します。このディレクトリには、daemon_example という名前のデーモンプラグイン用の daemon_example.cc というソースファイルが格納されており、このデーモンプラグインは、データディレクトリ内の mysql-heartbeat.log という名前のファイルにハートビート文字列を定期的な間隔で書き込みます。

デーモンプラグインを作成するには、プラグインのソースファイルに次のヘッダーファイルをインクルードします。プラグインの機能および要件によっては、ほかの MySQL のヘッダーファイルまたは一般的なヘッダーファイルが必要になることもあります。

```
#include <mysql/plugin.h>
```

plugin.h は MYSQL_DAEMON_PLUGIN サーバープラグインタイプおよびプラグインの宣言に必要なデータ構造体を定義します。

daemon_example.cc ファイルはライブラリディスクリプタを次のようにセットアップします。このライブラリディスクリプタには、単一の一般サーバープラグインディスクリプタが含まれています。

```
mysql_declare_plugin(daemon_example)
{
  MYSQL_DAEMON_PLUGIN,
  &daemon_example_plugin,
  "daemon_example",
  "Brian Aker",
  "Daemon example, creates a heartbeat beat file in mysql-heartbeat.log",
  PLUGIN_LICENSE_GPL,
  daemon_example_plugin_init, /* Plugin Init */
```

```
 daemon_example_plugin_deinit, /* Plugin Deinit */
 0x0100 /* 1.0 */,
 NULL,                       /* status variables                */
 NULL,                       /* system variables                */
 NULL,                       /* config options                  */
 0,                          /* flags                           */
}
mysql_declare_plugin_end;
```

name メンバー (daemon_example) は、INSTALL PLUGIN や UNINSTALL PLUGIN などのステートメント内でプラグインを参照するために使用する名前を指定します。これは、SHOW PLUGINS または INFORMATION_SCHEMA.PLUGINS によって表示される名前でもあります。

プラグインディスクリプタの 2 番目のメンバー daemon_example_plugin は、タイプ固有のデーモンプラグインディスクリプタを指しています。この構造体は、タイプ固有の API バージョン番号のみで構成されています。

```
struct st_mysql_daemon daemon_example_plugin=
{ MYSQL_DAEMON_INTERFACE_VERSION  };
```

タイプ固有の構造体にはインタフェース機能がありません。サーバーとプラグインの間の通信はありませんが、サーバーは一般プラグインディスクリプタから初期化関数および初期化解除関数を呼び出してプラグインを開始および停止します。

- daemon_example_plugin_init() はハートビートファイルを開き、定期的に動作するスレッドを生成し、次のメッセージをファイルに書き込みます。
- daemon_example_plugin_deinit() はファイルを閉じて、ほかのクリーンアップ処理を実行します。

プラグインライブラリのオブジェクトファイルをコンパイルおよびインストールするには、セクション 24.2.4.3「プラグインライブラリのコンパイルおよびインストール」の手順を使用します。ライブラリファイルを使用するには、ライブラリファイルがプラグインディレクトリ (plugin_dir システム変数で指定されているディレクトリ) にインストールされている必要があります。daemon_example プラグインの場合、ソースから MySQL をビルドするときにコンパイルおよびインストールされます。これはバイナリ配布にも含められます。ビルド処理では、libdaemon_example.so という名前の共有オブジェクトライブラリが生成されます (サフィクスはプラットフォームによって異なる場合があります)。

プラグインを使用するには、プラグインをサーバーに登録します。たとえば、プラグインを実行時に登録するには、次のステートメントを使用します (必要に応じてサフィクスを変更します)。

```
mysql> INSTALL PLUGIN daemon_example SONAME 'libdaemon_example.so';
```

プラグインのロードについての追加情報は、セクション5.1.8.1「プラグインのインストールおよびアンインストール」を参照してください。

プラグインのインストールを検証するには、INFORMATION_SCHEMA.PLUGINS テーブルを調査するか、SHOW PLUGINS ステートメントを使用します。

プラグインがロードされると、プラグインはデータディレクトリ内の mysql-heartbeat.log という名前のファイルにハートビート文字列を定期的な間隔で書き込みます。このファイルは無制限に大きくなるため、プラグインが正しく動作することを確認したら、ファイルをアンロードしてください。

```
mysql> UNINSTALL PLUGIN daemon_example;
```

24.2.4.6 INFORMATION_SCHEMA プラグインの作成

このセクションでは、INFORMATION_SCHEMA テーブルサーバープラグインを作成する方法について説明します。このようなプラグインを実装するためのコード例については、MySQL ソース配布の sql/sql_show.cc ファイルを参照してください。InnoDB ソースにあるプラグインの例を参照することもできます。InnoDB ソースツリーの handler/i_s.cc ファイルおよび handler/ha_innodb.cc ファイルを参照してください (storage/innobase ディレクトリ内)。

INFORMATION_SCHEMA テーブルプラグインを作成するには、プラグインのソースファイルに次のヘッダーファイルをインクルードします。プラグインの機能および要件によっては、ほかの MySQL のヘッダーファイルまたは一般的なヘッダーファイルが必要になることもあります。

```
#include <sql_class.h>
#include <table.h>
```

これらのヘッダーファイルは、MySQL ソース配布の sql ディレクトリにあります。これらは C++ 構造体を含んでいるため、INFORMATION_SCHEMA プラグインのソースファイルは (C ではなく) C++ コードとしてコンパイルする必要があります。

ここで作成するプラグインの例のソースファイルは、simple_i_s_table.cc という名前です。これは、SIMPLE_I_S_TABLE という名前の単純な INFORMATION_SCHEMA テーブルを作成し、NAME および VALUE という名前の 2 つのカラムがあります。テーブルを実装するプラグインライブラリの一般ディスクリプタは、次のようになります。

```
mysql_declare_plugin(simple_i_s_library)
{
  MYSQL_INFORMATION_SCHEMA_PLUGIN,
  &simple_table_info,              /* type-specific descriptor */
  "SIMPLE_I_S_TABLE",              /* table name */
  "Author Name",                   /* author */
  "Simple INFORMATION_SCHEMA table", /* description */
  PLUGIN_LICENSE_GPL,              /* license type */
  simple_table_init,               /* init function */
  NULL,
  0x0100,                          /* version = 1.0 */
  NULL,                            /* no status variables */
  NULL,                            /* no system variables */
  NULL,                            /* no reserved information */
  0                              /* no flags */
}
mysql_declare_plugin_end;
```

name メンバー (SIMPLE_I_S_TABLE) は、INSTALL PLUGIN、UNINSTALL PLUGIN などのステートメントでプラグインを参照するために使用する名前を指定します。これは、SHOW PLUGINS または INFORMATION_SCHEMA.PLUGINS によって表示される名前でもあります。

一般ディスクリプタの simple_table_info メンバーは、タイプ固有の API バージョン番号のみで構成されるタイプ固有のディスクリプタを指しています。

```
static struct st_mysql_information_schema simple_table_info =
{ MYSQL_INFORMATION_SCHEMA_INTERFACE_VERSION };
```

一般ディスクリプタは、初期化関数および初期化解除関数を指しています。

- 初期化関数は、テーブル構造体およびテーブルにデータを移入する関数についての情報を提供します。
- 初期化解除関数は、必要なすべてのクリーンアップ処理を実行します。クリーンアップが不要な場合、このディスクリプタメンバーには (例に示されているように) NULL を指定できます。

初期化関数は、成功した場合は 0、および工ラーが発生した場合は 1 を返します。この関数は一般ポインタを受け取り、テーブル構造体へのポインタとして解釈します。

```
static int table_init(void *ptr)
{
  ST_SCHEMA_TABLE *schema_table= (ST_SCHEMA_TABLE*)ptr;

  schema_table->fields_info= simple_table_fields;
  schema_table->fill_table= simple_fill_table;
  return 0;
}
```

この関数は、テーブル構造体の次の 2 つのメンバーを設定します。

- fields_info: 各カラムについての情報を格納する ST_FIELD_INFO 構造体の配列。
- fill_table: テーブルにデータを移入する関数。

fields_info によって指し示されている配列には、INFORMATION_SCHEMA のカラムごとの 1 つの要素および終端の要素が含まれるようにします。プラグイン例の次の simple_table_fields 配列は、SIMPLE_I_S_TABLE に 2 つのカラムがあることを示しています。NAME は長さ 10 バイトの文字列値であり、VALUE は表示幅が 20 バイトの整数値です。最後の構造体は、配列の終わりを示しています。

```
static ST_FIELD_INFO simple_table_fields[]=
{
  {"NAME", 10, MYSQL_TYPE_STRING, 0, 0 0, 0},
  {"VALUE", 6, MYSQL_TYPE_LONG, 0, MY_I_S_UNSIGNED, 0, 0},
  {0, 0, MYSQL_TYPE_NULL, 0, 0, 0, 0}
};
```

カラムの情報構造体については、table.h ヘッダーファイルの ST_FIELD_INFO の定義を参照してください。許容される MYSQL_TYPE_xxx タイプ値は、C API で使用される値です。セクション23.7.5「C API データ構造」を参照してください。

fill_table メンバーには、テーブルにデータを移入して、成功した場合は 0、およびエラーが発生した場合は 1 を返す関数を設定します。プラグインの例の場合、simple_fill_table() 関数は次のようになっています。

```
static int simple_fill_table(THD *thd, TABLE_LIST *tables, Item *cond)
{
  TABLE *table= tables->table;

  table->field[0]->store("Name 1", 6, system_charset_info);
  table->field[1]->store(1);
  if (schema_table_store_record(thd, table))
    return 1;
  table->field[0]->store("Name 2", 6, system_charset_info);
  table->field[1]->store(2);
  if (schema_table_store_record(thd, table))
    return 1;
  return 0;
}
```

この関数は、INFORMATION_SCHEMA テーブルの各行の各カラムを初期化し、schema_table_store_record() を呼び出して行を生成します。store() メソッドの引数は、格納される値のタイプによって異なります。カラム 0 (NAME、文字列) の場合、store() は、文字列へのポインタ、その長さ、およびその文字列の文字セットに関する情報を受け取ります。

```
store(const char *to, uint length, CHARSET_INFO *cs);
```

カラム 1 (VALUE、整数) の場合、store() は値およびその値が符号なしであるかどうかを示すフラグを受け取ります。

```
store(longlong nr, bool unsigned_value);
```

INFORMATION_SCHEMA テーブルにデータを移入する方法に関するほかの例については、sql_show.cc で schema_table_store_record() のインスタンスを検索してください。

プラグインライブラリのオブジェクトファイルをコンパイルおよびインストールするには、セクション 24.2.4.3「プラグインライブラリのコンパイルおよびインストール」の手順を参照してください。ライブラリファイルを使用するには、ライブラリファイルがプラグインディレクトリ (plugin_dir システム変数で指定されているディレクトリ) にインストールされている必要があります。

プラグインをテストするには、プラグインをインストールします。

```
mysql> INSTALL PLUGIN SIMPLE_I_S_TABLE SONAME 'simple_i_s_table.so';
```

テーブルが存在することを確認します。

```
mysql> SELECT TABLE_NAME FROM INFORMATION_SCHEMA.TABLES
    -> WHERE TABLE_NAME = 'SIMPLE_I_S_TABLE';
+-----------------+
| TABLE_NAME      |
+-----------------+
| SIMPLE_I_S_TABLE |
+-----------------+
```

選択を試行します。

```
mysql> SELECT * FROM INFORMATION_SCHEMA.SIMPLE_I_S_TABLE;
+--------+-------+
| NAME   | VALUE |
+--------+-------+
| Name 1 |     1 |
| Name 2 |     2 |
+--------+-------+
```

アンインストールします。

```
mysql> UNINSTALL PLUGIN SIMPLE_I_S_TABLE;
```

24.2.4.7 準同期レプリケーションプラグインの作成

このセクションでは、MySQL ソース配布の plugin/semisync ディレクトリにあるプラグイン例を使用して、準同期レプリケーションサーバープラグインを作成する方法について説明します。このディレクトリには、rpl_semi_sync_master および rpl_semi_sync_slave という名前のマスタープラグインおよびスレーブプラグインのソースファイルが格納されています。ここでは、プラグインフレームワークをセットアップする方法についてのみ説明します。プラグインがレプリケーション機能を実装する方法については、ソースを参照してください。

準同期レプリケーションプラグインを作成するには、プラグインソースファイルに次のヘッダーファイルをインクルードします。プラグインの機能および要件によっては、ほかの MySQL のヘッダーファイルまたは一般的なヘッダーファイルが必要になることもあります。

```
#include <mysql/plugin.h>
```

plugin.h は、MYSQL_REPLICATION_PLUGIN サーバープラグインタイプ、およびプラグインを宣言するために必要なデータ構造体を定義します。

マスター側の場合、semisync_master_plugin.cc には rpl_semi_sync_master という名前のプラグインの次の一般ディスクリプタが含まれています。

```
mysql_declare_plugin(semi_sync_master)
{
  MYSQL_REPLICATION_PLUGIN,
  &semi_sync_master_plugin,
  "rpl_semi_sync_master",
  "He Zhenxing",
  "Semi-synchronous replication master",
  PLUGIN_LICENSE_GPL,
  semi_sync_master_plugin_init, /* Plugin Init */
  semi_sync_master_plugin_deinit, /* Plugin Deinit */
  0x0100 /* 1.0 */,
  semi_sync_master_status_vars, /* status variables */
  semi_sync_master_system_vars, /* system variables */
  NULL,                         /* config options */
  0,                          /* flags */
}
mysql_declare_plugin_end;
```

スレーブ側の場合、semisync_slave_plugin.cc には rpl_semi_sync_slave という名前のプラグインの次の一般ディスクリプタが含まれています。

```
mysql_declare_plugin(semi_sync_slave)
{
  MYSQL_REPLICATION_PLUGIN,
  &semi_sync_slave_plugin,
  "rpl_semi_sync_slave",
  "He Zhenxing",
  "Semi-synchronous replication slave",
  PLUGIN_LICENSE_GPL,
  semi_sync_slave_plugin_init, /* Plugin Init */
  semi_sync_slave_plugin_deinit, /* Plugin Deinit */
  0x0100 /* 1.0 */,
  semi_sync_slave_status_vars,  /* status variables */
  semi_sync_slave_system_vars,  /* system variables */
  NULL,                         /* config options */
  0,                          /* flags */
}
mysql_declare_plugin_end;
```

マスタープラグインおよびスレーブプラグインのどちらの場合も、一般ディスクリプタには、タイプ固有のディスクリプタ、初期化関数と初期化解除関数、およびプラグインによって実装されるステータス変数とシステム変数へのポインタがあります。変数の設定については、サーバープラグインのステータス変数およびシステム変数を参照してください。以降の説明では、マスタープラグインのタイプ固有のディスクリプタ、初期化関数および初期化解除関数について記述していますが、スレーブプラグインにも同様に当てはまります。

マスターの一般ディスクリプタの semi_sync_master_plugin メンバーは、タイプ固有のディスクリプタを指しており、これはタイプ固有の API バージョン番号のみで構成されています。

```
struct Mysql_replication semi_sync_master_plugin= {
  MYSQL_REPLICATION_INTERFACE_VERSION
};
```

初期化関数および初期化解除関数の宣言は、次のようになります。

```
static int semi_sync_master_plugin_init(void *p);
static int semi_sync_master_plugin_deinit(void *p);
```

初期化関数はポインタを使用して、トランザクションおよびバイナリロギングの「オブザーバー」をサーバーに登録します。初期化が成功したあとに、サーバーは適切な時期にオブザーバーを呼び出します。(オブザーバーについては、ソースファイルを参照してください。)初期化解除関数は、オブザーバーを登録解除することによってクリーンアップ処理を実行します。各関数は、成功した場合は 0、およびエラーが発生した場合は 1 を返します。

プラグインライブラリのオブジェクトファイルをコンパイルおよびインストールするには、セクション 24.2.4.3「プラグインライブラリのコンパイルおよびインストール」の手順を使用します。ライブラリファイルを使用するには、ライブラリファイルがプラグインディレクトリ (plugin_dir システム変数で指定されているディレクトリ) にインストールされている必要があります。rpl_semi_sync_master プラグインおよび rpl_semi_sync_slave プラグインの場合、これらはソースから MySQL をビルドするときにコンパイルおよびインストールされます。これらはバイナリ配布にも含まれています。ビルド処理では、semisync_master.so および semisync_slave.so という名前の共有オブジェクトライブラリが生成されます (サフィクスはプラットフォームによって異なる場合があります)。

24.2.4.8 監査プラグインの作成

このセクションでは、MySQL ソース配布の plugin/audit_null ディレクトリにあるプラグインの例を使用して、監査サーバープラグインを作成する方法について説明します。このディレクトリにある audit_null.c ソースファイルは、NULL_AUDIT という名前の簡単な監査プラグインの例を実装します。

サーバー内では、プラガブルな監査インタフェースは MySQL ソース配布の sql ディレクトリの sql_audit.h ファイルおよび sql_audit.cc ファイル内に実装されています。また、サーバー内のいくつかの場所は、監査可能なイベントが発生したときに監査インタフェースを呼び出すように変更されているため、登録された監査プラグインは必要に応じてイベントに関する通知を受けることができます。そのような呼び出しが行われる場所を確認するには、mysql_audit_xxx() という形式の名前を持つ関数の呼び出しを探してください。監査通知は、次のようなサーバーの動作に対して発生します。

- 一般クエリーログへのメッセージの書き込み (ログが有効にされている場合)
- エラーログへのメッセージの書き込み
- クライアントへのクエリー結果の送信
- クライアントの接続イベントおよび接続解除イベント

監査プラグインを作成するには、プラグインのソースファイルに次のヘッダーファイルをインクルードします。プラグインの機能および要件によっては、ほかの MySQL のヘッダーファイルまたは一般的なヘッダーファイルが必要になることもあります。

```
#include <mysql/plugin_audit.h>
```

plugin_audit.h は plugin.h をインクルードするため、後者のファイルを明示的にインクルードする必要はありません。plugin.h は MYSQL_AUDIT_PLUGIN サーバープラグインタイプおよびプラグインを宣言するために必要なデータ構造体を定義します。plugin_audit.h は、監査プラグインに固有のデータ構造体を定義します。

監査プラグインには、ほかの MySQL サーバープラグインと同様に、一般プラグインディスクリプタがあります (サーバープラグインライブラリおよびプラグインディスクリプタを参照してください)。audit_null.c では、一般ディスクリプタは次のようになります。

```
mysql_declare_plugin(audit_null)
{
  MYSQL_AUDIT_PLUGIN,        /* type                         */
  &audit_null_descriptor,    /* descriptor                   */
  "NULL_AUDIT",              /* name                         */
  "Oracle Corp",             /* author                       */
  "Simple NULL Audit",       /* description                  */
  PLUGIN_LICENSE_GPL,
  audit_null_plugin_init,    /* init function (when loaded)     */
  audit_null_plugin_deinit,  /* deinit function (when unloaded) */
  0x0003,                    /* version                      */
  simple_status,             /* status variables             */
  NULL,                      /* system variables             */
  NULL,
  0,
}
mysql_declare_plugin_end;
```

name メンバー (NULL_AUDIT) は、INSTALL PLUGIN、UNINSTALL PLUGIN などのステートメントでプラグインを参照するために使用する名前を指定します。これは INFORMATION_SCHEMA.PLUGINS または SHOW PLUGINS によって表示される名前でもあります。

一般ディスクリプタは、SHOW STATUS ステートメントに対して各種のステータス変数を公開する構造体である simple_status も参照しています。

```
static struct st_mysql_show_var simple_status[]=
```

```
{
 { "Audit_null_called",
   (char *) &number_of_calls,
   SHOW_INT },
 { "Audit_null_general_log",
   (char *) &number_of_calls_general_log,
   SHOW_INT },
 { "Audit_null_general_error",
   (char *) &number_of_calls_general_error,
   SHOW_INT },
 { "Audit_null_general_result",
   (char *) &number_of_calls_general_result,
   SHOW_INT },
 { "Audit_null_general_status",
   (char *) &number_of_calls_general_status,
   SHOW_INT },
 { "Audit_null_connection_connect",
   (char *) &number_of_calls_connection_connect,
   SHOW_INT },
 { "Audit_null_connection_disconnect",
   (char *) &number_of_calls_connection_disconnect,
   SHOW_INT },
 { "Audit_null_connection_change_user",
   (char *) &number_of_calls_connection_change_user,
   SHOW_INT },
 { 0, 0, 0}
};
```

audit_null_plugin_init 初期化関数は、プラグインがロードされたときにステータス変数をゼロに設定します。audit_null_plugin_deinit 関数は、プラグインがアンロードされるときにクリーンアップ処理を実行します。プラグインは動作中に、通知を受け取るたびに最初のステータス変数を増分します。また、イベントクラスおよびサブクラスに応じてほかのものも増分します。つまり、最初の変数はイベントサブクラスのカウントの総計となります。

一般ディスクリプタの audit_null_descriptor 値は、タイプ固有のディスクリプタを指しています。監査プラグインの場合、このディスクリプタの構造体は次のようになります。

```
struct st_mysql_audit
{
  int interface_version;
  void (*release_thd)(MYSQL_THD);
  void (*event_notify)(MYSQL_THD, unsigned int, const void *);
  unsigned long class_mask[MYSQL_AUDIT_CLASS_MASK_SIZE];
};
```

タイプ固有のディスクリプタには次のメンバーがあります。

- interface_version: 規則により、タイプ固有のプラグインディスクリプタは、特定のプラグインタイプのインタフェースバージョンで始まります。サーバーはプラグインをロードするときに interface_version を検査し、プラグインと互換性があるかどうかを確認します。監査プラグインの場合、interface_version メンバーの値は MYSQL_AUDIT_INTERFACE_VERSION (plugin_audit.h に定義されています) です。
- release_thd: スレッドコンテキストとの関連付けが解除されていることをプラグインに通知するためにサーバーが呼び出す関数。そのような関数がない場合は、NULL を設定します。
- event_notify: 監査可能イベントが発生したことをプラグインに通知するためにサーバーが呼び出す関数。この関数は NULL にはできません。監査が行われなくては意味がないためです。
- class_mask: プラグインが通知を受け取るイベントクラスを示すビットマスク。この値が 0 の場合、サーバーはプラグインにイベントを渡しません。

サーバーは event_notify 関数および release_thd 関数を一緒に使用します。これらは特定のスレッドのコンテキストで呼び出され、スレッドはいくつかのイベント通知が生成されるアクティビティーを実行する場合があります。サーバーは、スレッドに対して event_notify を最初に呼び出すときに、プラグインからスレッドへのバインドを作成します。このバインドが存在する間はプラグインをアンインストールできません。スレッドに対してイベントが発生しなくなると、サーバーは release_thd 関数を呼び出すことによってプラグインにこれを通知してからバインドを破棄します。たとえば、クライアントがステートメントを発行したとき、ステートメントを処理するスレッドは、ステートメントによって生成された結果セットとログに記録されたステートメントについて、監査プラグインに通知することがあります。これらの通知が実行されたあと、クライアントが別のステートメントを発行するまでの間、サーバーはスレッドをスリープ状態にする前にプラグインを解放します。

この設計により、プラグインは event_notify 関数への最初の呼び出しで対象となるスレッドに必要なリソースを割り当て、release_thd 関数でリソースを解放します。

```
event_notify function:
  if memory is needed to service the thread
    allocate memory
  ... rest of notification processing ...

release_thd function:
  if memory was allocated
    release memory
  ... rest of release processing ...
```

これは、通知関数でメモリーの割り当てと解放を繰り返すよりも効率的です。

NULL_AUDIT 監査プラグインの例では、タイプ固有のディスクリプタは次のようになります。

```
static struct st_mysql_audit audit_null_descriptor=
{
  MYSQL_AUDIT_INTERFACE_VERSION,                  /* interface version    */
  NULL,                                           /* release_thd function */
  audit_null_notify,                              /* notify function      */
  { (unsigned long) MYSQL_AUDIT_GENERAL_CLASSMASK |
                    MYSQL_AUDIT_CONNECTION_CLASSMASK } /* class mask        */
};
```

サーバーは audit_null_notify を呼び出して監査イベント情報をプラグインに渡します。release_thd 関数はありません。

イベントクラスマスクは、「general」クラスおよび「connection」クラスのすべてのイベントを対象とすることを示しています。plugin_audit.h は、これらのクラスとそれらに対応するクラスマスクのシンボルを定義しています。

```
#define MYSQL_AUDIT_GENERAL_CLASS 0
#define MYSQL_AUDIT_GENERAL_CLASSMASK (1 << MYSQL_AUDIT_GENERAL_CLASS)

#define MYSQL_AUDIT_CONNECTION_CLASS 1
#define MYSQL_AUDIT_CONNECTION_CLASSMASK (1 << MYSQL_AUDIT_CONNECTION_CLASS)
```

タイプ固有のディスクリプタでは、event_notify 関数プロトタイプの 2 番目と 3 番目のパラメータは、イベントクラス、およびイベント構造体への一般的なポインタを表しています。

```
void (*event_notify)(MYSQL_THD, unsigned int, const void *);
```

異なるクラスのイベントは異なる構造体を持つ場合があるため、通知関数はイベントクラス値を使用して、イベント構造へのポインタを解釈する方法を判断します。

イベントクラスが MYSQL_AUDIT_GENERAL_CLASS である通知関数をサーバーが呼び出す場合、サーバーは mysql_event_general 構造へのポインタとしてイベント構造体を渡します。

```
struct mysql_event_general
{
  unsigned int event_subclass;
  int general_error_code;
  unsigned long general_thread_id;
  const char *general_user;
  unsigned int general_user_length;
  const char *general_command;
  unsigned int general_command_length;
  const char *general_query;
  unsigned int general_query_length;
  struct charset_info_st *general_charset;
  unsigned long long general_time;
  unsigned long long general_rows;
};
```

監査プラグインは mysql_event_general メンバーを次のように解釈できます。

- event_subclass: 次のいずれかの値を持つイベントサブクラス。

  ```
  #define MYSQL_AUDIT_GENERAL_LOG 0
  #define MYSQL_AUDIT_GENERAL_ERROR 1
  #define MYSQL_AUDIT_GENERAL_RESULT 2
  #define MYSQL_AUDIT_GENERAL_STATUS 3
  ```

- general_error_code: エラーコード。これは mysql_errno() C API 関数によって返されるものと似た値であり、0 は「エラーなし」を意味します。

- general_thread_id: イベントが発生したスレッドの ID。

- general_user: イベントの現在のユーザー。
- general_user_length: general_user のバイト単位の長さ。
- general_command: 一般クエリーログイベントの場合、操作の種類。たとえば、Connect、Query、Shutdown です。エラーログイベントの場合、エラーメッセージ。これは mysql_error() C API 関数によって返されるものと似た値であり、空の文字列は「エラーなし」を意味します。結果イベントの場合、これは空です。
- general_command_length: general_command のバイト単位の長さ。
- general_query: ログに記録されたか結果を生成した SQL ステートメント。
- general_query_length: general_query のバイト単位の長さ。
- general_charset: イベントの文字セット情報。
- general_time: 通知関数が呼び出された直前の時間を示す TIMESTAMP 値。
- general_rows: 一般クエリーログイベントの場合、ゼロ。エラーログイベントの場合、エラーが発生した行番号。結果イベントの場合、結果の行数に 1 を加えた数値。結果セットを生成しないステートメントの場合、値は 0 です。このエンコードによって、結果セットを生成しないステートメントを、空の結果セットを生成するステートメントと区別できます。たとえば、DELETE ステートメントの場合、この値は 0 です。SELECT の場合、結果は常に 1 以上であり、1 は空の結果セットを表します。
- general_host: 一般クエリーログイベントの場合、クライアントのホスト名を表す文字列。
- general_sql_command: 一般クエリーログイベントの場合、connect、drop_table などの実行されるアクションの種類を示す文字列。
- general_external_user: 一般クエリーログイベントの場合、外部ユーザーを表す文字列 (ない場合は空)。
- general_ip: 一般クエリーログイベントの場合、クライアントの IP アドレスを表す文字列。

general_host、general_sql_command、general_external_user、および general_ip メンバーは、MySQL 5.6.14 で新しく導入されました。これらは、文字列とその長さがペアになった MYSQL_LEX_STRING 構造体です。たとえば、event_general が一般イベントへのポインタである場合、次のようにして general_host 値のメンバーにアクセスできます。

```
event_general->general_host.length
event_general->general_host.str
```

イベントクラスが MYSQL_AUDIT_CONNECTION_CLASS である通知関数をサーバーが呼び出す場合、サーバーは mysql_event_connection 構造へのポインタとしてイベント構造体を渡します。これは mysql_event_general 構造と似ており、ほぼ同じように解釈されます。

NULL_AUDIT プラグインの通知関数はきわめて単純です。これは、グローバルイベントカウンタを増分して、イベントクラスを判別し、イベントサブクラスを参照して増分するサブクラスカウンタを判別します。

```
static void audit_null_notify(MYSQL_THD thd __attribute__((unused)),
                              unsigned int event_class,
                              const void *event)
{
  /* prone to races, oh well */
  number_of_calls++;
  if (event_class == MYSQL_AUDIT_GENERAL_CLASS)
  {
    const struct mysql_event_general *event_general=
      (const struct mysql_event_general *) event;
    switch (event_general->event_subclass)
    {
    case MYSQL_AUDIT_GENERAL_LOG:
      number_of_calls_general_log++;
      break;
    case MYSQL_AUDIT_GENERAL_ERROR:
      number_of_calls_general_error++;
      break;
    case MYSQL_AUDIT_GENERAL_RESULT:
      number_of_calls_general_result++;
      break;
    case MYSQL_AUDIT_GENERAL_STATUS:
      number_of_calls_general_status++;
      break;
    default:
      break;
```

```
  }
 }
 else if (event_class == MYSQL_AUDIT_CONNECTION_CLASS)
 {
   const struct mysql_event_connection *event_connection=
     (const struct mysql_event_connection *) event;
   switch (event_connection->event_subclass)
   {
   case MYSQL_AUDIT_CONNECTION_CONNECT:
     number_of_calls_connection_connect++;
     break;
   case MYSQL_AUDIT_CONNECTION_DISCONNECT:
     number_of_calls_connection_disconnect++;
     break;
   case MYSQL_AUDIT_CONNECTION_CHANGE_USER:
     number_of_calls_connection_change_user++;
     break;
   default:
     break;
   }
 }
}
```

プラグインライブラリのオブジェクトファイルをコンパイルおよびインストールするには、セクション 24.2.4.3「プラグインライブラリのコンパイルおよびインストール」の手順を使用します。ライブラリファイルを使用するには、ライブラリファイルがプラグインディレクトリ (plugin_dir システム変数で指定されているディレクトリ) にインストールされている必要があります。AUDIT_NULL プラグインの場合、ソースから MySQL をビルドするときにコンパイルおよびインストールされます。これはバイナリ配布にも含められます。ビルド処理では、adt_null.so という名前の共有オブジェクトライブラリが生成されます (サフィクスはプラットフォームによって異なる場合があります)。

プラグインを実行時に登録するには、次のステートメントを使用します (必要に応じてサフィクスを変更します)。

```
mysql> INSTALL PLUGIN NULL_AUDIT SONAME 'adt_null.so';
```

プラグインのロードについての追加情報は、セクション5.1.8.1「プラグインのインストールおよびアンインストール」を参照してください。

プラグインのインストールを検証するには、INFORMATION_SCHEMA.PLUGINS テーブルを調査するか、SHOW PLUGINS ステートメントを使用します。

監査プラグインがインストールされている間、監査プラグインはプラグインが呼び出されたイベントを示すステータス変数を公開します。

```
mysql> SHOW STATUS LIKE 'Audit_null%';
+-----------------------------------+-------+
| Variable_name                     | Value |
+-----------------------------------+-------+
| Audit_null_called                 | 1388  |
| Audit_null_connection_change_user | 0     |
| Audit_null_connection_connect     | 22    |
| Audit_null_connection_disconnect  | 21    |
| Audit_null_general_error          | 1     |
| Audit_null_general_log            | 513   |
| Audit_null_general_result         | 415   |
| Audit_null_general_status         | 416   |
+-----------------------------------+-------+
```

Audit_null_called はすべてのイベントをカウントし、ほかの変数はイベントサブクラスのインスタンスをカウントします。たとえば、上記の SHOW STATUS ステートメントによって、サーバーは結果をクライアントに送信し、ログが有効にされている場合は一般クエリーログにメッセージを書き込みます。このため、クライアントがステートメントを繰り返し発行すると、Audit_null_called および Audit_null_general_result が毎回増分され、ログが有効にされている場合は Audit_null_general_log が増分されます。

プラグインをテスト後に無効にするには、次のステートメントを使用してプラグインをアンロードします。

```
mysql> UNINSTALL PLUGIN NULL_AUDIT;
```

24.2.4.9 認証プラグインの作成

MySQL ではプラガブルな認証がサポートされており、プラグインはクライアント接続を認証するために呼び出されます。認証プラグインを使用すると、mysql.user テーブルに格納されているパスワードによる組み込み方式ではない認証方式を使用できます。たとえば、外部認証方式にアクセスするプラグインを作成できます。また、認

証プラグインはプロキシユーザー機能をサポートできるため、接続するユーザーを別のユーザーのプロキシにして、(アクセス制御のために) 別のユーザーの権限を持つユーザーとして扱うことが可能になります。詳細は、セクション6.3.7「プラガブル認証」およびセクション6.3.9「プロキシユーザー」を参照してください。

認証プラグインでは、サーバー側およびクライアント側のプラグインを作成します。サーバー側プラグインは、全文パーサープラグイン、監査プラグインなどのほかの種類のサーバープラグインで使用されるものと同じプラグイン API を使用します (ただし、タイプ固有のディスクリプタは異なります)。クライアント側プラグインはクライアントプラグイン API を使用します。

いくつかのヘッダーファイルに、認証プラグインに関する情報が格納されています。

- plugin.h: MYSQL_AUTHENTICATION_PLUGIN サーバープラグインタイプを定義します。
- client_plugin.h: クライアントプラグインの API を定義します。これには、クライアントプラグインディスクリプタ、およびクライアントプラグイン C API を呼び出すための関数プロトタイプが含まれます (セクション23.7.14「C API クライアントプラグイン関数」を参照してください)。
- plugin_auth.h: 認証プラグインに固有のサーバープラグイン API の一部を定義します。これにはサーバー側認証プラグインのタイプ固有のディスクリプタと MYSQL_SERVER_AUTH_INFO 構造体が含まれます。
- plugin_auth_common.h: クライアント認証プラグインとサーバー認証プラグインの共通要素が格納されます。これには戻り値の定義と MYSQL_PLUGIN_VIO 構造体が含まれます。

認証プラグインを作成するには、プラグインのソースファイルに次のヘッダーファイルをインクルードします。プラグインの機能および要件によっては、ほかの MySQL のヘッダーファイルまたは一般的なヘッダーファイルが必要になることもあります。

- サーバー認証プラグインを実装するソースファイルの場合、次のファイルをインクルードします。

```
#include <mysql/plugin_auth.h>
```

- クライアント認証プラグインを実装するか、クライアントプラグインとサーバープラグインの両方を実装するソースファイルの場合、次のファイルをインクルードします。

```
#include <mysql/plugin_auth.h>
#include <mysql/client_plugin.h>
#include <mysql.h>
```

plugin_auth.h は plugin.h および plugin_auth_common.h をインクルードするため、後者のファイルを明示的にインクルードする必要はありません。

このセクションでは、連係して動作するサーバー認証プラグインとクライアント認証プラグインのペアを作成する方法について説明します。

> **警告**
>
> これらのプラグインは空でないあらゆるパスワードを受け入れ、パスワードは平文で送信されます。これは安全でないため、このプラグインは本番環境で使用しないでください。

ここで作成するサーバー側プラグインおよびクライアント側プラグインの名前は、両方とも auth_simple です。セクション24.2.4.2「プラグインのデータ構造体」で説明したように、プラグインライブラリファイルはクライアントプラグインと同じベース名を持つ必要があるため、ソースファイル名は auth_simple.c であり、auth_simple.so という名前のライブラリが生成されます (システムで、ライブラリファイルのサフィクスとして .so が使用されていることを想定しています)。

MySQL ソース配布では、認証プラグインのソースが plugin/auth ディレクトリにあり、ほかの認証プラグインを作成するときのガイドとして参考にできます。また、組み込み認証プラグインが実装される方法を確認するには、MySQL サーバーに組み込まれるプラグインについては sql/sql_acl.cc、およびlibmysqlclient クライアントライブラリに組み込まれるプラグインについては sql-common/client.c を参照してください。(組み込みクライアントプラグインの場合、使用される auth_plugin_t 構造体は、通常のクライアントプラグイン宣言マクロで使用される構造と異なります。特に、最初の 2 つのメンバーは、宣言マクロによって設定されるのではなく、明示的に指定されています。)

サーバー側認証プラグインの作成

すべてのサーバープラグインタイプに使用される通常の一般ディスクリプタ形式を使用して、サーバー側プラグインを宣言します (サーバープラグインライブラリおよびプラグインディスクリプタを参照してください)。auth_simple プラグインの場合、ディスクリプタは次のようになります。

```
mysql_declare_plugin(auth_simple)
{
  MYSQL_AUTHENTICATION_PLUGIN,
  &auth_simple_handler,               /* type-specific descriptor */
  "auth_simple",                      /* plugin name */
  "Author Name",                      /* author */
  "Any-password authentication plugin", /* description */
  PLUGIN_LICENSE_GPL,                 /* license type */
  NULL,                               /* no init function */
  NULL,                               /* no deinit function */
  0x0100,                             /* version = 1.0 */
  NULL,                               /* no status variables */
  NULL,                               /* no system variables */
  NULL,                               /* no reserved information */
  0                                   /* no flags */
}
mysql_declare_plugin_end;
```

name メンバー (auth_simple) は、INSTALL PLUGIN、UNINSTALL PLUGIN などのステートメントでプラグインを参照するために使用する名前を指定します。これは、SHOW PLUGINS または INFORMATION_SCHEMA.PLUGINS によって表示される名前でもあります。

一般ディスクリプタの auth_simple_handler メンバーはタイプ固有のディスクリプタを指しています。認証プラグインの場合、タイプ固有のディスクリプタは st_mysql_auth 構造体 (plugin_auth.h で定義されます) のインスタンスです。

```
struct st_mysql_auth
{
  int interface_version;
  const char *client_auth_plugin;
  int (*authenticate_user)(MYSQL_PLUGIN_VIO *vio, MYSQL_SERVER_AUTH_INFO *info);
};
```

st_mysql_auth 構造体には 3 つのメンバー (タイプ固有の API バージョン番号、クライアントプラグイン名、およびクライアントと通信するメインのプラグイン関数へのポインタ) があります。client_auth_plugin メンバーは、特定のプラグインが必要な場合にクライアントプラグインの名前を示すようにします。NULL 値は「任意のプラグイン」を意味します。後者の場合、クライアントは任意のプラグインを使用できます。これは、クライアントプラグイン、およびクライアントプラグインが送信するユーザー名またはパスワードが、サーバープラグインの処理に関与しない場合に役立ちます。たとえば、サーバープラグインがローカルのクライアントのみを認証し、クライアントプラグインによって送信される情報ではなくオペレーティングシステムのいくつかのプロパティーを使用する場合です。

auth_simple の場合、タイプ固有のディスクリプタは次のようになります。

```
static struct st_mysql_auth auth_simple_handler =
{
  MYSQL_AUTHENTICATION_INTERFACE_VERSION,
  "auth_simple",          /* required client-side plugin name */
  auth_simple_server      /* server-side plugin main function */
};
```

メインの関数は、I/O 構造体および MYSQL_SERVER_AUTH_INFO 構造体を表す 2 つの引数を受け取ります。この構造体の定義は plugin_auth.h にあります。これは次のような定義です。

```
typedef struct st_mysql_server_auth_info
{
  char *user_name;
  unsigned int user_name_length;
  const char *auth_string;
  unsigned long auth_string_length;
  char authenticated_as[MYSQL_USERNAME_LENGTH+1];
  char external_user[512];
  int  password_used;
  const char *host_or_ip;
  unsigned int host_or_ip_length;
} MYSQL_SERVER_AUTH_INFO;
```

文字列メンバーの文字セットは UTF-8 です。文字列に関連付けられた _length メンバーがある場合、これはバイト単位の文字列の長さを示します。文字列も NULL で終わります。

サーバーによって認証プラグインが呼び出されると、MYSQL_SERVER_AUTH_INFO 構造体メンバーが次のように解釈されます。これらの一部は、説明されているようにクライアントセッション内の SQL 関数またはシステム変数の値を設定するために使用されます。

- user_name: クライアントによって送信されたユーザー名。この値は USER() 関数の値になります。

- user_name_length: user_name のバイト単位の長さ。
- auth_string: mysql.user テーブルの一致するアカウント名の行 (つまり、クライアントユーザー名およびホスト名が一致し、クライアントを認証する方法を判別するためにサーバーによって使用される行) の authentication_string カラムの値。

  次のステートメントを使用してアカウントを作成するとします。

```
CREATE USER 'my_user'@'localhost'
  IDENTIFIED WITH my_plugin AS 'my_auth_string';
```

  my_user がローカルホストから接続する場合、サーバーは my_plugin を呼び出し、'my_auth_string' を auth_string 値として渡します。

- auth_string_length: auth_string のバイト単位の長さ。
- authenticated_as: サーバーはこれをユーザー名 (user_name の値) に設定します。プラグインは、クライアントが別のユーザーの権限を持つことを示すように変更できます。たとえば、プラグインでプロキシユーザーがサポートされる場合、初期値は接続する (プロキシ) ユーザーの名前ですが、プラグインがこのメンバーをプロキシ設定されているユーザー名に変更できます。その後、サーバーはプロキシ設定されているユーザーの権限を持つものとしてプロキシユーザーを扱います (プロキシユーザーのサポートのためのほかの条件が満たされていることを想定しています。認証プラグインでのプロキシユーザーサポートの実装を参照してください)。この値は、最大 MYSQL_USER_NAME_LENGTH バイトの長さを持つ文字列、および終端の NULL として表されます。この値は CURRENT_USER() 関数の値になります。
- external_user: サーバーは空の文字列 (NULL で終わります) をこれに設定します。この値は external_user システム変数の値になります。プラグインでこのシステム変数が異なる値を持つようにするには、プラグインでそのようにこのメンバーを設定してください (たとえば、接続しているユーザー名を設定します)。この値は、最大 511 バイトの長さの文字列、および終端の NULL として表されます。
- password_used: このメンバーは、認証に失敗したときに適用されます。プラグインは、これを設定するか、無視できます。この値は、Authentication fails. Password used: %s という失敗のエラーメッセージを作成するために使用されます。password_used の値によって、次の表に示すように、%s の処理方法が決まります。

password_used	%s の処理
0	行わない
1	行う
2	%s がない

- host_or_ip: 解決できる場合はクライアントのホスト名、または解決できない場合は IP アドレス。
- host_or_ip_length: host_or_ip のバイト単位の長さ。

auth_simple のメイン関数 auth_simple_server() は、クライアントからパスワード (NULL で終了する文字列) を読み取り、パスワードが空でない (先頭バイトが NULL でない) 場合に成功します。

```
static int auth_simple_server (MYSQL_PLUGIN_VIO *vio,
                               MYSQL_SERVER_AUTH_INFO *info)
{
  unsigned char *pkt;
  int pkt_len;

  /* read the password as null-terminated string, fail on error */
  if ((pkt_len= vio->read_packet(vio, &pkt)) < 0)
    return CR_ERROR;

  /* fail on empty password */
  if (!pkt_len || *pkt == '\0')
  {
    info->password_used= PASSWORD_USED_NO;
    return CR_ERROR;
  }

  /* accept any nonempty password */
  info->password_used= PASSWORD_USED_YES;

  return CR_OK;
}
```

メイン関数は、次の表に示すいずれかのエラーコードを返します。

エラーコード	意味
CR_OK	成功
CR_OK_HANDSHAKE_COMPLETE	ハンドシェイクステータスパケットをクライアントに送信しない
CR_ERROR	エラー
CR_AUTH_USER_CREDENTIALS	認証エラー
CR_AUTH_HANDSHAKE	認証ハンドシェイクエラー
CR_AUTH_PLUGIN_ERROR	内部プラグインエラー

ハンドシェイクが動作する方法の例については、plugin/auth/dialog.c ソースファイルを参照してください。

CR_ERROR に続くエラーコードは MySQL 5.6.5 から使用できます。サーバーは、MySQL 5.6.5 以降で使用可能なパフォーマンススキーマ host_cache テーブル内のプラグインエラーをカウントします。

auth_simple_server_main() は非常に基本的なものであるため、パスワードを受け取ったかどうかを示すメンバーを設定することを除き、認証情報構造体を使用しません。

プロキシユーザーをサポートするプラグインは、プロキシ設定されているユーザー (クライアントユーザーが取得する権限を持っている MySQL ユーザー) の名前をサーバーに返す必要があります。これを行うには、プラグインはプロキシ設定されているユーザー名を info->authenticated_as メンバーに設定する必要があります。プロキシ設定については、セクション6.3.9「プロキシユーザー」および認証プラグインでのプロキシユーザーサポートの実装を参照してください。

クライアント側認証プラグインの作成

mysql_declare_client_plugin() マクロおよび mysql_end_client_plugin マクロを使用して、クライアント側プラグインディスクリプタを宣言します (クライアントプラグインディスクリプタを参照してください)。auth_simple プラグインの場合、ディスクリプタは次のようになります。

```
mysql_declare_client_plugin(AUTHENTICATION)
  "auth_simple",                    /* plugin name */
  "Author Name",                    /* author */
  "Any-password authentication plugin", /* description */
  {1,0,0},                          /* version = 1.0.0 */
  "GPL",                            /* license type */
  NULL,                             /* for internal use */
  NULL,                             /* no init function */
  NULL,                             /* no deinit function */
  NULL,                             /* no option-handling function */
  auth_simple_client                /* main function */
mysql_end_client_plugin;
```

プラグイン名からオプション処理関数までのディスクリプタメンバーは、すべてのクライアントプラグインタイプで共通しています。(説明については、クライアントプラグインディスクリプタを参照してください。)共通メンバーに続いて、このディスクリプタには認証プラグインに固有の追加メンバーがあります。これは「メイン」の関数であり、サーバーとの通信を処理します。この関数は、I/O 構造体と接続ハンドラを表す 2 つの引数を取ります。任意のパスワードを指定できるこの単純なプラグインでは、メイン関数はユーザーが指定したパスワードをサーバーに書き込むだけです。

```
static int auth_simple_client (MYSQL_PLUGIN_VIO *vio, MYSQL *mysql)
{
  int res;

  /* send password as null-terminated string in clear text */
  res= vio->write_packet(vio, (const unsigned char *) mysql->passwd,
                         strlen(mysql->passwd) + 1);

  return res ? CR_ERROR : CR_OK;
}
```

メイン関数は、次の表に示すいずれかのエラーコードを返します。

エラーコード	意味
CR_OK	成功
CR_OK_HANDSHAKE_COMPLETE	成功、クライアント完了
CR_ERROR	エラー

CR_OK_HANDSHAKE_COMPLETE は、クライアントが自分の役割を正常に完了し、最後のパケットを読み取ったことを示します。認証プロトコルのラウンドトリップ数が事前に知られておらず、認証が完了したかどうかを判別するためにプラグインが別のパケットを読み取る必要がある場合、クライアントプラグインが CR_OK_HANDSHAKE_COMPLETE を返すことがあります。

認証プラグインの使用

プラグインライブラリのオブジェクトファイルをコンパイルおよびインストールするには、セクション 24.2.4.3「プラグインライブラリのコンパイルおよびインストール」の手順を参照してください。ライブラリファイルを使用するには、ライブラリファイルがプラグインディレクトリ (plugin_dir システム変数で指定されているディレクトリ) にインストールされている必要があります。

サーバー側プラグインをサーバーに登録します。たとえば、プラグインをサーバー起動時にロードするには、--plugin-load=auth_simple.so オプションを使用します (システムで必要な場合はライブラリサフィクスを変更します)。

サーバーが認証のために auth_simple プラグインを使用する対象となるユーザーを作成します。

```
mysql> CREATE USER 'x'@'localhost'
    -> IDENTIFIED WITH auth_simple;
```

クライアントプログラムを使用して、ユーザー x としてサーバーに接続します。サーバー側の auth_simple プラグインはクライアント側の auth_simple プラグインを使用するクライアントプログラムと通信し、後者はサーバーにパスワードを送信します。サーバーは空のパスワードを送信する接続を拒否し、空ではないパスワードを送信する接続を受け入れます。これを検証するために、それぞれの方法でクライアントプログラムを起動します。

```
shell> mysql --user=x --skip-password
ERROR 1045 (28000): Access denied for user 'x'@'localhost' (using password: NO)

shell> mysql --user=x --password=abc
mysql>
```

このサーバープラグインは空ではないパスワードをすべて受け入れるため、安全ではないとみなされます。プラグインをテストしてプラグインが機能することを検証したら、気づかずに安全でない認証プラグインがロードされたままの状態でサーバーが実行され続けないように、--plugin-load オプションを指定せずにサーバーを再起動します。また、DROP USER 'x'@'localhost' を使用してユーザーを削除します。

認証プラグインのロードおよび使用に関する追加情報については、セクション5.1.8.1「プラグインのインストールおよびアンインストール」およびセクション6.3.7「プラガブル認証」を参照してください。

認証プラグインの使用をサポートするクライアントプログラムを作成する場合、一般的にそのようなプログラムは、mysql_options() を呼び出して MYSQL_DEFAULT_AUTH オプションおよび MYSQL_PLUGIN_DIR オプションを設定することによって、プラグインをロードします。

```
char *plugin_dir = "path_to_plugin_dir";
char *default_auth = "plugin_name";

/* ... process command-line options ... */

mysql_options(&mysql, MYSQL_PLUGIN_DIR, plugin_dir);
mysql_options(&mysql, MYSQL_DEFAULT_AUTH, default_auth);
```

一般にプログラムは、ユーザーがデフォルト値をオーバーライドできるようにする --plugin-dir および --default-auth オプションも受け付けます。

クライアントプログラムで低レベルのプラグイン管理が必要である場合は、クライアントライブラリに st_mysql_client_plugin 引数を受け取る関数を含めます。セクション23.7.14「C API クライアントプラグイン関数」を参照してください。

認証プラグインでのプロキシユーザーサポートの実装

プラガブルな認証によって使用できるようになる機能の 1 つはプロキシユーザーです (セクション6.3.9「プロキシユーザー」を参照してください)。サーバー側の認証プラグインでプロキシユーザーがサポートされるようにするには、次の条件を満たしている必要があります。

- 接続するクライアントをプロキシユーザーとして扱う場合、プラグインは MYSQL_SERVER_AUTH_INFO 構造体の authenticated_as メンバー内に異なる名前を返して、プロキシ設定されているユーザー名を示す必要があります。external_user システム変数の値を設定するために、必要に応じて external_user メンバーを設定することもできます。

- プロキシユーザーのアカウントは、プラグインによって認証されるようにセットアップする必要があります。アカウントをプラグインに関連付けるには、CREATE USER ステートメントまたは GRANT ステートメントを使用します。
- プロキシユーザーアカウントには、プロキシ設定されているアカウントに対する PROXY 権限がある必要があります。この権限を付与するには、GRANT ステートメントを使用します。

言い換えると、プラグインに必要なプロキシユーザーサポートの特性は、プロキシ設定されているユーザー名を authenticated_as に設定することのみです。そのほかは、オプションであるか (external_user の設定)、SQL ステートメントを使用して DBA が行います。

認証プラグインはプロキシユーザーが接続したときに返すプロキシ設定されているユーザーをどのように決定するのでしょうか。これはプラグインによって異なります。通常、プラグインはサーバーから渡された認証文字列に基づいて、クライアントをプロキシ設定されているユーザーにマップします。この文字列は、認証にプラグインを使用することを指定する CREATE USER ステートメントの IDENTIFIED WITH 句の AS 部分に指定されます。

プラグイン開発者は認証文字列の構文ルールを決定し、これらのルールに従ってプラグインを実装します。外部ユーザーを MySQL ユーザーにマップするカンマ区切りのリストのペアをプラグインが受け取るとします。例:

```
CREATE USER ''@'%.example.com'
  IDENTIFIED WITH my_plugin AS 'extuser1=mysqlusera, extuser2=mysqluserb'
CREATE USER ''@'%.example.org'
  IDENTIFIED WITH my_plugin AS 'extuser1=mysqluserc, extuser2=mysqluserd'
```

サーバーがプラグインを起動してクライアントを認証するときに、適切な認証文字列をプラグインに渡します。プラグインには次の役割があります。

1. 文字列を構文解析して構成要素に分解し、使用するマッピングを決定する
2. クライアントユーザー名とマッピングを比較する
3. 適切な MySQL ユーザー名を返す

たとえば、example.com ホストから extuser2 が接続する場合、サーバーは 'extuser1=mysqlusera, extuser2=mysqluserb' をプラグインに渡し、プラグインは mysqluserb に終端の NULL バイトを付加して authenticated_as にコピーします。example.org ホストから extuser2 が接続する場合、サーバーは 'extuser1=mysqluserc, extuser2=mysqluserd' を渡し、プラグインは代わりに mysqluserd をコピーします。

マッピングに一致するものがない場合のアクションはプラグインによって異なります。一致するものがある必要があるとき、多くの場合、プラグインはエラーを返します。または、プラグインが単純にクライアント名を返すこともあり、この場合、プラグインは authenticated_as を変更せず、サーバーはクライアントをプロキシとして扱いません。

次の例では、auth_simple_proxy という名前のプラグインを使用して、プロキシユーザーを処理する方法を示しています。前に示した auth_simple プラグインのように、auth_simple_proxy は空ではないすべてのパスワードを有効なものとして受け入れます (このため、本番環境で使用しないでください)。また、これは auth_string 認証文字列メンバーを検査し、次の非常に単純なルールを使用してこれを解釈します。

- 文字列が空の場合、プラグインは指定されたユーザー名を返し、プロキシ設定は行われません。つまり、プラグインは authenticated_as の値を変更せずにそのままにします。
- 文字列が空ではない場合、プラグインはこれをプロキシ設定されているユーザーの名前として扱い、プロキシ設定が行われるようにこれを authenticated_as にコピーします。

テストを行うために、上記のルールに従ってプロキシ設定されていない 1 つのアカウントとプロキシ設定されている 1 つのアカウントをセットアップします。これは、一方のアカウントには AS 句がなく、他方のアカウントにはプロキシ設定されているユーザーを指定する AS 句があることを意味します。

```
CREATE USER 'plugin_user1'@'localhost'
  IDENTIFIED WITH auth_simple_proxy;
CREATE USER 'plugin_user2'@'localhost'
  IDENTIFIED WITH auth_simple_proxy AS 'proxied_user';
```

また、プロキシ設定されているユーザーのアカウントを作成し、そのアカウントに対する PROXY 権限を plugin_user2 に付与します。

```
CREATE USER 'proxied_user'@'localhost'
  IDENTIFIED BY 'proxied_user_pass';
GRANT PROXY
```

```
 ON 'proxied_user'@'localhost'
 TO 'plugin_user2'@'localhost';
```

サーバーは、認証プラグインを呼び出す前に、authenticated_as にクライアントユーザー名を設定します。ユーザーがプロキシであることを示すために、プラグインはプロキシ設定されているユーザー名を authenticated_as に設定します。auth_simple_proxy の場合、これは、auth_string 値を検査し、値が空でない場合は値を authenticated_as メンバーにコピーして、プロキシ設定されているユーザーの名前としてそれを返す必要があることを意味します。また、プロキシ設定が行われる場合、プラグインは external_user メンバーにクライアントユーザー名を設定し、これが external_user システム変数の値になります。

```
static int auth_simple_proxy_server (MYSQL_PLUGIN_VIO *vio,
                                     MYSQL_SERVER_AUTH_INFO *info)
{
  unsigned char *pkt;
  int pkt_len;

  /* read the password as null-terminated string, fail on error */
  if ((pkt_len= vio->read_packet(vio, &pkt)) < 0)
    return CR_ERROR;

  /* fail on empty password */
  if (!pkt_len || *pkt == '\0')
  {
    info->password_used= PASSWORD_USED_NO;
    return CR_ERROR;
  }

  /* accept any nonempty password */
  info->password_used= PASSWORD_USED_YES;

  /* if authentication string is nonempty, use as proxied user name */
  /* and use client name as external_user value */
  if (info->auth_string_length > 0)
  {
    strcpy (info->authenticated_as, info->auth_string);
    strcpy (info->external_user, info->user_name);
  }

  return CR_OK;
}
```

接続が成功すると、USER() 関数は接続しているクライアントユーザーとホスト名を示し、CURRENT_USER() はセッション中に適用される権限を持つアカウントを示します。後者の値は、プロキシ設定が行われない場合は接続するユーザーアカウント、およびプロキシ設定が行われた場合はプロキシ設定されているアカウントになります。

プラグインをコンパイルおよびインストールして、テストします。最初に、plugin_user1 として接続します。

```
shell> mysql --user=plugin_user1 --password=x
```

この場合、プロキシ設定は行われません。

```
mysql> SELECT USER(), CURRENT_USER(), @@proxy_user, @@external_user\G
*************************** 1. row ***************************
       USER(): plugin_user1@localhost
CURRENT_USER(): plugin_user1@localhost
  @@proxy_user: NULL
@@external_user: NULL
```

次に、plugin_user2 として接続します。

```
shell> mysql --user=plugin_user2 --password=x
```

この場合、plugin_user2 は proxied_user に対してプロキシ設定されます。

```
mysql> SELECT USER(), CURRENT_USER(), @@proxy_user, @@external_user\G
*************************** 1. row ***************************
       USER(): plugin_user2@localhost
CURRENT_USER(): proxied_user@localhost
  @@proxy_user: 'plugin_user2'@'localhost'
@@external_user: 'plugin_user2'@'localhost'
```

24.2.4.10 パスワード検証プラグインの作成

このセクションでは、パスワード検証サーバープラグインを作成する方法について説明します。この手順は、MySQL ソース配布の plugin/password_validation ディレクトリにあるソースコードに基づいています。その

ディレクトリ内にある validate_password.cc ソースファイルは、validate_password という名前のプラグインを実装します。

パスワード検証プラグインを作成するには、プラグインのソースファイルに次のヘッダーファイルをインクルードします。プラグインの機能および要件によっては、ほかの MySQL のヘッダーファイルまたは一般的なヘッダーファイルが必要になることもあります。

```
#include <mysql/plugin_validate_password.h>
```

plugin_validate_password.h は plugin.h をインクルードするため、後者のファイルを明示的にインクルードする必要はありません。plugin.h は MYSQL_VALIDATE_PASSWORD_PLUGIN サーバープラグインタイプとプラグインを宣言するために必要なデータ構造体を定義します。plugin_validate_password.h は、パスワード検証プラグインに固有のデータ構造体を定義します。

パスワード検証プラグインには、ほかの MySQL サーバープラグインと同じように一般プラグインディスクリプタがあります (サーバープラグインライブラリおよびプラグインディスクリプタを参照してください)。validate_password.cc では、一般ディスクリプタは次のようになります。

```
mysql_declare_plugin(validate_password)
{
  MYSQL_VALIDATE_PASSWORD_PLUGIN,    /*   type                         */
  &validate_password_descriptor,     /*   descriptor                   */
  "validate_password",               /*   name                         */
  "Oracle Corporation",              /*   author                       */
  "check password strength",         /*   description                  */
  PLUGIN_LICENSE_GPL,
  validate_password_init,            /*   init function (when loaded)     */
  validate_password_deinit,          /*   deinit function (when unloaded) */
  0x0100,                            /*   version                      */
  NULL,
  validate_password_system_variables, /*   system variables             */
  NULL,
  0,
}
mysql_declare_plugin_end;
```

name メンバー (validate_password) は、INSTALL PLUGIN、UNINSTALL PLUGIN などのステートメントでプラグインを参照するために使用する名前を指定します。これは INFORMATION_SCHEMA.PLUGINS または SHOW PLUGINS によって表示される名前でもあります。

一般ディスクリプタは、SHOW VARIABLES ステートメントに対してさまざまなステータス変数を公開する構造体である validate_password_system_variables も参照します。

```
static struct st_mysql_sys_var* validate_password_system_variables[]= {
  MYSQL_SYSVAR(length),
  MYSQL_SYSVAR(number_count),
  MYSQL_SYSVAR(mixed_case_count),
  MYSQL_SYSVAR(special_char_count),
  MYSQL_SYSVAR(policy),
  MYSQL_SYSVAR(dictionary_file),
  NULL
};
```

validate_password_init 初期化関数は辞書ファイルを読み取り (指定されている場合)、validate_password_deinit 関数はそのファイルに関連付けられているデータ構造体を解放します。

一般ディスクリプタの validate_password_descriptor 値はタイプ固有のディスクリプタを指しています。パスワード検証プラグインの場合、このディスクリプタの構造体は次のようになります。

```
struct st_mysql_validate_password
{
  int interface_version;
  /*
    This function returns TRUE for passwords which satisfy the password
    policy (as chosen by plugin variable) and FALSE for all other
    password
  */
  int (*validate_password)(mysql_string_handle password);
  /*
    This function returns the password strength (0-100) depending
    upon the policies
  */
  int (*get_password_strength)(mysql_string_handle password);
};
```

タイプ固有のディスクリプタには次のメンバーがあります。

- interface_version: 規則により、タイプ固有のプラグインディスクリプタは、特定のプラグインタイプのインタフェースバージョンで始まります。サーバーはプラグインをロードするときに interface_version を検査し、プラグインと互換性があるかどうかを確認します。パスワード検証プラグインの場合、interface_version メンバーの値は MYSQL_VALIDATE_PASSWORD_INTERFACE_VERSION (plugin_validate_password.h で定義されます) です。

- validate_password: パスワードが現在のパスワードポリシーを満たしているかどうかをテストするためにサーバーが呼び出す関数。パスワードが妥当な場合は 1 を返し、そうでない場合は 0 を返します。引数はパスワードであり、mysql_string_handle 値として渡されます。このデータ型は mysql_string サーバーサービスによって実装されます。詳細は、sql ディレクトリ内の string_service.h および string_service.cc のソースファイルを参照してください。

- get_password_strength: パスワードの強さを評価するためにサーバーが呼び出す関数。これは 0 (弱い) から 100 (強い) までの値を返します。引数はパスワードであり、mysql_string_handle 値として渡されます。

validate_password プラグインの場合、タイプ固有のディスクリプタは次のようになります。

```
static struct st_mysql_validate_password validate_password_descriptor=
{
  MYSQL_VALIDATE_PASSWORD_INTERFACE_VERSION,
  validate_password,                    /* validate function         */
  get_password_strength                   /* validate strength function */
};
```

プラグインライブラリのオブジェクトファイルをコンパイルおよびインストールするには、セクション 24.2.4.3「プラグインライブラリのコンパイルおよびインストール」の手順を使用します。ライブラリファイルを使用するには、ライブラリファイルがプラグインディレクトリ (plugin_dir システム変数で指定されているディレクトリ) にインストールされている必要があります。validate_password プラグインの場合、MySQL をソースからビルドするときにコンパイルおよびインストールされます。これはバイナリ配布にも含められます。ビルド処理では、validate_password.so という名前の共有オブジェクトライブラリが生成されます (拡張子はプラットフォームによって異なる場合があります)。

プラグインを実行時に登録するには、次のステートメントを使用します (必要に応じて拡張子を変更します)。

```
mysql> INSTALL PLUGIN validate_password SONAME 'validate_password.so';
```

プラグインのロードについての追加情報は、セクション5.1.8.1「プラグインのインストールおよびアンインストール」を参照してください。

プラグインのインストールを検証するには、INFORMATION_SCHEMA.PLUGINS テーブルを調査するか、SHOW PLUGINS ステートメントを使用します。

validate_password プラグインがインストールされている間、プラグインはパスワード検査パラメータを示すシステム変数を公開します。

```
mysql> SHOW VARIABLES LIKE 'validate_password%';
+--------------------------------------+--------+
| Variable_name                        | Value  |
+--------------------------------------+--------+
| validate_password_dictionary_file    |        |
| validate_password_length             | 8      |
| validate_password_mixed_case_count   | 1      |
| validate_password_number_count       | 1      |
| validate_password_policy             | MEDIUM |
| validate_password_special_char_count | 1      |
+--------------------------------------+--------+
```

これらの変数の説明については、パスワード検証プラグインのオプションおよび変数を参照してください。

プラグインをテスト後に無効にするには、次のステートメントを使用してプラグインをアンロードします。

```
mysql> UNINSTALL PLUGIN validate_password;
```

24.2.5 プラグインのための MySQL サービス

MySQL サーバープラグインは、サーバーの「サービス」にアクセスできます。サービスインタフェースは、プラグインが呼び出すことができるサーバー機能を公開します。これはプラグイン API を補完し、次の特徴があります。

- サービスによって、プラグインは通常の関数呼び出しを使用して、サーバー内部のコードにアクセスできます。
- サービスには移植性があり、複数のプラットフォームで動作します。
- インタフェースにはバージョン管理メカニズムがあるため、サーバーによってサポートされるサービスバージョンを、プラグインバージョンに対してロード時にチェックできます。バージョン管理により、サーバーが提供するサービスのバージョン、およびプラグインが予期または必要とするサービスのバージョンの間の互換性の問題が防止されます。

現在のサービスには次のものが含まれており、ほかのサービスも実装できます。

- my_plugin_log_service: プラグインがエラーの報告およびエラーメッセージの指定を行うことができるサービス。サーバーはエラーログにメッセージを書き込みます。
- my_snprintf: プラットフォーム間で一貫性のある結果を生成する、文字列の書式設定サービス。
- my_thd_scheduler: プラグインがスレッドスケジューラを選択するためのサービス。
- mysql_string: 文字列操作のためのサービス。
- thd_alloc: メモリー割り当てサービス。
- thd_wait: プラグインがスリープまたは停止することを報告するためのサービス。

プラグインサービスインタフェースとプラグイン API の違いは次のとおりです。

- プラグイン API では、サーバーからプラグインを使用できます。呼び出しを開始するのは、プラグインを呼び出すサーバーです。これにより、サーバー機能の拡張、またはサーバーの処理に関する通知を受け取るための登録をプラグインで行うことができます。
- プラグインサービスインタフェースでは、プラグインはサーバー内部のコードを呼び出すことができます。呼び出しを開始するのは、サービス関数を呼び出すプラグインです。これにより、多くのプラグインがサーバーにすでに実装されている機能を使用できるようになり、プラグインに個別に機能を実装する必要がなくなります。

サーバーを変更して新しいサービスを追加する開発者は、プラグインのための MySQL サービスを参照してください。

このセクションの残りの部分では、サービスとして使用可能なサーバー機能をプラグインで使用する方法について説明します。my_snprintf サービスを使用する「デーモン」プラグインの例のソースも参照してください。このプラグインは、MySQL ソース配布の plugin/daemon_example ディレクトリにあります。

存在するサービスおよびサービスが提供する関数を確認するには、MySQL ソース配布の include/mysql ディレクトリ内を参照してください。関連するファイルは次のとおりです。

- plugin.h は services.h をインクルードします。
- services.h は、使用可能なすべてのサービス固有のヘッダーファイルをインクルードする「包括的な」ヘッダーです。
- サービス固有のヘッダーには、service_my_snprintf.h または service_thd_alloc.h のような名前が付いています。

各サービス固有ヘッダーには、対象となるサービスの完全な使用法を示すドキュメントを提供するコメントがあり、使用可能なサービス関数、それらの呼び出しシーケンス、および戻り値が含まれています。

プラグイン内からサービスを使用するには、サービス関連の情報にアクセスするための plugin.h ヘッダーファイルをプラグインのソースファイルでインクルードする必要があります。

```
#include <mysql/plugin.h>
```

これによってセットアップのコストが増えるわけではありません。このファイルにはすべてのプラグインで必要となる定義および構造体が含まれているため、プラグインはいずれにしてもこのファイルをインクルードする必要があります。

サービスにアクセスするために、プラグインはほかの関数と同様にサービス関数を呼び出します。たとえば、出力のために文字列を書式設定してバッファーに入れるには、同じ名前のサービスによって提供されている my_snprintf() 関数を呼び出します。

```
char buffer[BUFFER_SIZE];

my_snprintf(buffer, sizeof(buffer), format_string, argument_to_format, ...);
```

サーバーがエラーログに書き込むエラーを報告するには、最初にエラーレベルを選択します。mysql/service_my_plugin_log.h はこれらのレベルを定義しています。

```
enum plugin_log_level
{
  MY_ERROR_LEVEL,
  MY_WARNING_LEVEL,
  MY_INFORMATION_LEVEL
};
```

次に、my_plugin_log_message() を呼び出します。

```
int my_plugin_log_message(MYSQL_PLUGIN *plugin, enum plugin_log_level level,
                          const char *format, ...);
```

例:

```
my_plugin_log_message(plugin_ptr, MY_ERROR_LEVEL, "Cannot initialize plugin");
```

プラグインをビルドするときに、libmysqlservices ライブラリでリンクする必要があります。リンク時には -lmysqlservices フラグを使用します。たとえば CMake の場合、最上位レベルの CMakeLists.txt ファイルに次の内容を指定します。

```
FIND_LIBRARY(MYSQLSERVICES_LIB mysqlservices
 PATHS "${MYSQL_SRCDIR}/libservices" NO_DEFAULT_PATH)
```

プラグインのソースが格納されているディレクトリの CMakeLists.txt ファイルに次の内容を指定します。

```
# the plugin needs the mysql services library for error logging
TARGET_LINK_LIBRARIES (your_plugin_library_name ${MYSQLSERVICES_LIB})
```

24.3 MySQL への新しい関数の追加

MySQL に新しい関数を追加する方法は 3 つあります。

- ユーザー定義関数 (UDF) インタフェースを使用して関数を追加できます。ユーザー定義関数はオブジェクトファイルとしてコンパイルされ、CREATE FUNCTION ステートメントおよび DROP FUNCTION ステートメントを使用して、サーバーに対して動的に追加および削除されます。セクション13.7.3.1「ユーザー定義関数のための CREATE FUNCTION 構文」を参照してください。
- 関数をネイティブ (組み込み) MySQL 関数として追加できます。ネイティブ関数はコンパイルされて mysqld サーバー内に組み込まれ、永続的に使用できます。
- 関数を追加するもう 1 つの方法は、ストアドファンクションを作成することです。これらは、オブジェクトコードをコンパイルするのではなく、SQL ステートメントを使用して記述します。ストアドファンクションを記述するための構文は、ここでは説明しません。セクション20.2「ストアドルーチン (プロシージャーと関数) の使用」を参照してください。

コンパイルされた関数を作成するための各方法には、長所と短所があります。

- ユーザー定義関数を作成する場合、サーバー自体のほかにオブジェクトファイルをインストールする必要があります。関数をコンパイルしてサーバーに配置する場合、それを行う必要はありません。
- ネイティブ関数の場合は、ソース配布を変更する必要があります。UDF の場合はその必要はありません。UDF をバイナリの MySQL 配布に追加できます。MySQL のソースにアクセスする必要はありません。
- MySQL の配布をアップグレードする場合、UDF インタフェースが変更される新しいバージョンにアップグレードしないかぎり、以前インストールした UDF を使用し続けることができます。ネイティブ関数の場合、アップグレードするたびに変更を繰り返す必要があります。

どのような方法を使用して新しい関数を追加したかにかかわらず、これらの関数は、ABS()、SOUNDEX() などのネイティブ関数と同じように SQL ステートメントから呼び出すことができます。

各種の関数への参照をサーバーが解釈する方法を記述したルールについては、セクション9.2.4「関数名の構文解析と解決」を参照してください。

以降のセクションでは、UDF インタフェースの機能、UDF を作成するための手順、UDF の誤用を防ぐために MySQL が行うセキュリティー予防措置、およびネイティブな MySQL 関数を追加する方法について説明します。

UDF を作成する方法を示すソースコードの例については、MySQL ソース配布に提供されている sql/udf_example.cc ファイルを参照してください。

24.3.1 ユーザー定義関数インタフェースの機能

ユーザー定義関数のための MySQL インタフェースには次の機能があります。

- 関数は、文字列値、整数値、または実数値を返すことができ、同じタイプの引数を受け入れることができます。
- 一度に単一の行を操作する単純な関数、または行のグループを操作する集約関数を定義できます。
- 関数に渡される引数の数、タイプ、および名前をチェックできる情報が関数に渡されます。
- 引数が関数に渡される前に引数を特定のタイプに強制的に変更するように MySQL に指示できます。
- 関数が NULL を返すこと、またはエラーが発生したことを示すことができます。

24.3.2 新しいユーザー定義関数の追加

UDF のメカニズムが機能するためには、関数を C または C++ で記述し、オペレーティングシステムが動的ロードをサポートしている必要があります。MySQL ソース配布には、5 つの UDF 関数を定義している sql/udf_example.cc ファイルが含まれています。UDF の呼び出し規則のしくみを理解するには、このファイルを参照してください。include/mysql_com.h ヘッダーファイルには UDF 関連のシンボルおよびデータ構造体が定義されていますが、このヘッダーファイルを直接インクルードする必要はなく、mysql.h によってインクルードされます。

UDF に含まれているコードは実行中のサーバーの一部になるため、UDF を記述するときは、サーバーコードを記述するときに適用されるすべての制約に従う必要があります。たとえば、libstdc++ ライブラリの関数を使用しようとすると、問題が生じることがあります。これらの制約は将来のサーバーのバージョンで変更されることがあるため、サーバーをアップグレードするときに、古いサーバー用にもともと作成された UDF を改訂する必要がある場合があります。これらの制約については、セクション2.9.4「MySQL ソース構成オプション」およびセクション2.9.5「MySQL のコンパイルに関する問題」を参照してください。

UDF を使用できるようにするには、mysqld を動的にリンクする必要があります。mysqld からシンボルにアクセスする必要がある UDF を使用する場合 (たとえば、sql/udf_example.cc の metaphone 関数は default_charset_info を使用します) は、-rdynamic を指定してプログラムをリンクする必要があります (man dlopen を参照してください)。

SQL ステートメントで使用する各関数について、対応する C (または C++) 関数を定義します。以降の説明では、「xxx」という名前を関数名の例として使用します。SQL と C/C++ での使用を区別するために、XXX() (大文字) は SQL 関数の呼び出しを示し、xxx() (小文字) は C/C++ 関数の呼び出しを示します。

> **注記**
>
> C++ を使用する場合は、C 関数を次のようにカプセル化できます。
>
> ```
> extern "C" { ... }
> ```
>
> これにより、完成した UDF 内で C++ 関数名が読み取ることができる状態に維持されます。

次のリストは、XXX() という名前の関数のインタフェースを実装するために記述する C/C++ 関数について説明しています。メインの関数 xxx() は必須です。また、セクション24.3.2.6「ユーザー定義関数のセキュリティー上の予防措置」で説明している理由により、UDF にはここで説明している少なくとも 1 つのほかの関数が必要となります。

- xxx()

  メイン関数。ここで関数の結果が計算されます。SQL 関数のデータ型と C/C++ 関数の戻り型との対応を次に示します。

SQL の型	C/C++ の型
STRING	char *

SQL の型	C/C++ の型
INTEGER	long long
REAL	double

DECIMAL 関数を宣言することも可能ですが、現時点では値は文字列として返されるため、STRING 関数の場合と同様に UDF を作成します。ROW 関数は実装されていません。

- xxx_init()

  xxx() の初期化関数。存在する場合は次の目的で使用できます。

  - XXX() への引数の数をチェックする。
  - 引数が目的の型であることを確認するか、メインの関数が呼び出されたときに引数を目的の型に強制的に変更するように MySQL に指示する。
  - メインの関数が必要とするメモリーを割り当てる。
  - 結果の最大長を指定する。
  - 結果の小数点以下の最大の桁数を指定する (REAL 関数の場合)。
  - 結果として NULL を許容するかどうかを指定する。

- xxx_deinit()

  xxx() の初期化解除関数。存在する場合、初期化関数によって割り当てられたすべてのメモリーを割り当て解除します。

SQL ステートメントによって XXX() が呼び出されると、MySQL は初期化関数 xxx_init() を呼び出して、引数のチェック、メモリーの割り当てなどの必要なセットアップを実行させます。xxx_init() が エラーを返した場合、MySQL はエラーメッセージを出力して SQL ステートメントを中止し、メイン関数または初期化解除関数を呼び出しません。それ以外の場合、MySQL はメイン関数 xxx() を行ごとに 1 回ずつ呼び出します。すべての行が処理されると、MySQL は初期化解除関数 xxx_deinit() を呼び出し、必要なクリーンアップ処理が実行されます。

SUM() のように動作する集約関数の場合は、次の関数も作成する必要があります。

- xxx_clear()

  現在の集約値をリセットしますが、新しいグループに対する初期集計値として引数を挿入しません。

- xxx_add()

  現在の集計値に引数を追加します。

MySQL は集計 UDF を次のように処理します。

1. xxx_init() を呼び出して、集約関数が結果を格納するために必要なメモリーを割り当てます。
2. GROUP BY 式に従ってテーブルをソートします。
3. 新しいグループになるたびに先頭行で xxx_clear() を呼び出します。
4. 同じグループに属する各行に対して xxx_add() を呼び出します。
5. グループが変更されたとき、または最後の行が処理されたあとに、xxx() を呼び出して集約の結果を取得します。
6. すべての行が処理されるまでステップ 3 から 5 までを繰り返します。
7. xxx_deinit() を呼び出して、UDF が割り当てたメモリーを解放します。

すべての関数はスレッドセーフである必要があります。これにはメイン関数だけでなく、初期化関数および初期化解除関数のほか、集約関数によって必要とされる追加の関数も含まれます。この要件により、変化するグローバル変数または静的変数を割り当てることができなくなります。メモリーが必要な場合は、メモリーを xxx_init() で割り当て、xxx_deinit() で解放するようにしてください。

24.3.2.1 単純な関数のための UDF の呼び出しシーケンス

このセクションでは単純な UDF を作成するときに定義する必要があるさまざまな関数について説明します。セクション24.3.2「新しいユーザー定義関数の追加」には、MySQL がこれらの関数を呼び出す順序が記載されています。

メインの xxx() 関数は、このセクションに示すように宣言してください。CREATE FUNCTION ステートメントで SQL 関数 XXX() が STRING、INTEGER、または REAL のいずれを返すかに応じて、戻り型およびパラメータは異なります。

STRING 関数の場合:

```
char *xxx(UDF_INIT *initid, UDF_ARGS *args,
          char *result, unsigned long *length,
          char *is_null, char *error);
```

INTEGER 関数の場合:

```
long long xxx(UDF_INIT *initid, UDF_ARGS *args,
              char *is_null, char *error);
```

REAL 関数の場合:

```
double xxx(UDF_INIT *initid, UDF_ARGS *args,
           char *is_null, char *error);
```

DECIMAL 関数は文字列値を返すため、STRING 関数と同様に宣言してください。ROW 関数は実装されていません。

初期化関数および初期化解除関数は、次のように宣言します。

```
my_bool xxx_init(UDF_INIT *initid, UDF_ARGS *args, char *message);

void xxx_deinit(UDF_INIT *initid);
```

initid パラメータは 3 つの関数すべてに渡されます。これは、関数間で情報をやり取りするために使用される UDF_INIT 構造体を指しています。UDF_INIT 構造体メンバーを次に示します。初期化関数で、変更するメンバーを設定してください。(メンバーのデフォルトを使用する場合は、変更せずにそのままにします。)

- my_bool maybe_null

  xxx() が NULL を返すことができる場合、xxx_init() は maybe_null を 1 に設定します。いずれかの引数が maybe_null として宣言されている場合、デフォルト値は 1 です。

- unsigned int decimals

  小数点の右側の桁数。デフォルト値は、メイン関数に渡された引数の小数点以下の桁数の最大値です。たとえば、関数に 1.34、1.345、および 1.3 が渡された場合、1.345 の小数点以下の桁数が 3 であるため、デフォルトは 3 になります。

  引数で小数点以下の桁数が固定されていない場合、decimals 値は 31 に設定され、これは DECIMAL、FLOAT、および DOUBLE データ型に許可される最大の小数点以下の桁数に 1 を加えた数です。MySQL 5.6 では、この値は mysql_com.h ヘッダーファイルの定数 NOT_FIXED_DEC として利用できます。

  decimals 値の 31 は、FLOAT カラムまたは DOUBLE カラムが小数点以下の桁数を明示せずに宣言された (たとえば、FLOAT(10,3) でなく FLOAT) 場合の引数、または 1345E-3 などの浮動小数点の定数に使用されます。これは、関数内で数値形式に変換される可能性がある、文字列およびその他の数値以外の引数に対しても使用されます。

  decimals メンバーを初期化する値は、デフォルトにすぎません。これは、実行される実際の計算が反映されるように関数内で変更できます。デフォルトは、引数の最大の小数点以下の桁数が使用されるように決定します。いずれかの引数の小数点以下の桁数が NOT_FIXED_DEC である場合、その値が decimals に使用されます。

- unsigned int max_length

  結果の最大長。デフォルトの max_length 値は、関数の結果の型に応じて異なります。文字列関数の場合、デフォルトはもっとも長い引数の長さです。整数関数の場合、デフォルトは 21 桁です。実数関数の場合、デフォルトは initid->decimals によって示されている小数点以下の桁数に 13 を加えた値です。(数値関数の場合、長さには符号文字または小数点文字が含まれています。)

BLOB 値を返す場合は、max_length を 65K バイトまたは 16M バイトに設定できます。このメモリーは割り当てられませんが、この値は、データを一時的に格納する必要がある場合に使用するデータ型を決定するために使用されます。

- char *ptr

  関数が独自の用途に使用できるポインタ。たとえば、複数の関数間で initid->ptr を使用して、割り当て済みのメモリーをやり取りできます。xxx_init() でこのメモリーを割り当てて、それをこのポインタに割り当てます。

```
initid->ptr = allocated_memory;
```

  xxx() および xxx_deinit() では、initid->ptr を参照して、メモリーを使用または割り当て解除します。

- my_bool const_item

  xxx_init() は、xxx() が常に同じ値を返す場合は const_item を 1 に設定し、それ以外の場合は 0 に設定します。

24.3.2.2 集約関数のための UDF の呼び出しシーケンス

このセクションでは集約 UDF を作成するときに定義する必要があるさまざまな関数について説明します。セクション24.3.2「新しいユーザー定義関数の追加」には、MySQL がこれらの関数を呼び出す順序が記載されています。

- xxx_reset()

  この関数は、MySQL が新しいグループ内で最初の行を見つけたときに呼び出されます。これはすべての内部サマリー変数をリセットし、指定された UDF_ARGS 引数をグループの内部サマリー値の最初の値として使用します。xxx_reset() は次のように宣言します。

```
void xxx_reset(UDF_INIT *initid, UDF_ARGS *args,
               char *is_null, char *error);
```

  MySQL 5.6 では、xxx_reset() は必要がないか使用されず、UDF インタフェースでは代わりに xxx_clear() が使用されます。ただし、古いバージョンのサーバーで UDF を動作させる場合、xxx_reset() と xxx_clear() を両方定義できます。(両方の関数を含める場合、xxx_reset() 関数は、すべての変数をリセットする xxx_clear() を呼び出してから、xxx_add() を呼び出して UDF_ARGS 引数をグループの最初の値として追加することによって、多くの場合内部的に実装できます。)

- xxx_clear()

  この関数は、MySQL でサマリー結果をリセットする必要がある場合に呼び出されます。これは新しいグループになるたびに最初に呼び出されますが、一致する行のないクエリーの値をリセットするために呼び出されることもあります。xxx_clear() は次のように宣言します。

```
void xxx_clear(UDF_INIT *initid, char *is_null, char *error);
```

  is_null は、xxx_clear() を呼び出す前に、CHAR(0) を指すように設定されます。

  処理に問題があった場合は、error 引数が指している変数に値を格納できます。error は文字列バッファーでなく単一バイト変数を指しています。

  xxx_clear() は MySQL 5.6 で必要となります。

- xxx_add()

  この関数は、同じグループに属するすべての行に対して呼び出されます。これは、UDF_ARGS 引数内の値を内部サマリー変数に追加するために使用します。

```
void xxx_add(UDF_INIT *initid, UDF_ARGS *args,
             char *is_null, char *error);
```

集約 UDF の xxx() 関数は、非集約 UDF と同様に宣言してください。セクション24.3.2.1「単純な関数のための UDF の呼び出しシーケンス」を参照してください。

集約 UDF の場合、MySQL はグループ内のすべての行が処理されたあとに xxx() 関数を呼び出します。通常はここで UDF_ARGS 引数にアクセスすることはなく、内部サマリー変数に基づいて値を返します。

xxx() での戻り値の処理は、非集約 UDF と同様に行います。セクション24.3.2.4「UDF の戻り値およびエラー処理」を参照してください。

xxx_reset() 関数および xxx_add() 関数は、UDF_ARGS 引数を非集約 UDF の関数と同様に処理します。セクション24.3.2.3「UDF 引数の処理」を参照してください。

is_null および error へのポインタ引数は、xxx_reset()、xxx_clear()、xxx_add()、および xxx() へのすべての呼び出しで同じです。これを使用すると、エラーが発生したこと、または xxx() 関数が NULL を返すかどうかを記憶できます。文字列を *error に格納しないでください。error は文字列バッファーでなく単一バイト変数を指しています。

*is_null は (xxx_clear() を呼び出す前に) グループごとにリセットされます。*error がリセットされることはありません。

xxx() が戻るときに *is_null または *error が設定されていた場合、MySQL はグループ関数の結果として NULL を返します。

24.3.2.3 UDF 引数の処理

args パラメータは、次に示すメンバーを持つ UDF_ARGS 構造体を指しています。

- unsigned int arg_count

  引数の数。特定の数の引数を使用して関数を呼び出す必要がある場合は、初期化関数でこの値をチェックします。例:

```
if (args->arg_count != 2)
{
  strcpy(message,"XXX() requires two arguments");
  return 1;
}
```

  配列であるその他の UDF_ARGS メンバー値の場合、配列参照はゼロベースです。つまり、0 から args->arg_count − 1 までのインデックス値を使用して配列メンバーを参照します。

- enum Item_result *arg_type

  各引数の型を格納する配列へのポインタ。指定可能な型の値は、STRING_RESULT、INT_RESULT、REAL_RESULT、および DECIMAL_RESULT です。

  引数が予期している型であることを確認し、そうではない場合にエラーを返すには、初期化関数で arg_type 配列をチェックします。例:

```
if (args->arg_type[0] != STRING_RESULT ||
  args->arg_type[1] != INT_RESULT)
{
  strcpy(message,"XXX() requires a string and an integer");
  return 1;
}
```

  DECIMAL_RESULT 型の引数は文字列として渡されるため、STRING_RESULT 値と同様にこれらを処理する必要があります。

  関数の引数が特定の型であることを要求する代わりに、初期化関数を使用して、arg_type 要素を必要な型に設定できます。これにより、MySQL が xxx() の各呼び出しで引数をそれらの型に強制的に変更します。たとえば、最初の 2 つの引数がそれぞれ文字列および整数になるように強制的に指定するには、xxx_init() で次の操作を実行します。

```
args->arg_type[0] = STRING_RESULT;
args->arg_type[1] = INT_RESULT;
```

  1.3、DECIMAL カラム値などの正確値型の 10 進数引数は、DECIMAL_RESULT 型で渡されます。ただし、値は文字列として渡されます。数値を受け取るには、初期化関数を使用して、引数が REAL_RESULT 値に強制的に変更されるように指定します。

```
args->arg_type[2] = REAL_RESULT;
```

- char **args

  args->args は、関数に渡される引数の一般的な性質についての情報を初期化関数に通知します。定数引数 i の場合、args->args[i] は引数値を指しています。(値に正しくアクセスする方法については、以降の説明を参照してください。)非定数引数の場合、args->args[i] は 0 です。定数引数は、3、4*7-2、SIN(3.14) などの定数のみを使用した式です。非定数引数は、行によって変化することがある値を参照する式のことであり、カラム名、非定数引数を指定して呼び出される関数などがあります。

メイン関数への各呼び出しで、args->args には、現在処理されている行に関して渡される実際の引数が含まれています。

引数 i が NULL の場合、args->args[i] はゼロポインタ (0) です。引数が NULL ではない場合、関数は引数を次のように参照できます。

- STRING_RESULT 型の引数は、文字列ポインタおよび長さとして指定し、バイナリデータまたは任意の長さのデータを処理できます。文字列の内容は args->args[i] として取得でき、文字列の長さは args->lengths[i] です。文字列がゼロで終わると想定しないでください。
- INT_RESULT 型の引数の場合は、args->args[i] を long long 値にキャストする必要があります。

```
long long int_val;
int_val = *((long long*) args->args[i]);
```

- REAL_RESULT 型の引数の場合は、args->args[i] を double 値にキャストする必要があります。

```
double    real_val;
real_val = *((double*) args->args[i]);
```

- DECIMAL_RESULT 型の引数の場合、値は文字列として渡さるため、STRING_RESULT 値と同様に処理されるようにしてください。
- ROW_RESULT 引数は実装されていません。

- unsigned long *lengths

  初期化関数の場合、lengths 配列は各引数の最大文字列長を示しています。これらは変更しないでください。メイン関数の各呼び出しで、lengths には、現在処理されている行に関して渡されるすべての文字列引数の実際の長さが含まれています。INT_RESULT 型または REAL_RESULT 型の引数の場合は、lengths には (初期化関数の場合と同様に) 引数の最大長が含まれています。

- char *maybe_null

  初期化関数において、maybe_null 配列は、各引数について引数値が NULL の場合があるかどうかを示します (ない場合は 0、およびある場合は 1)。

- char **attributes

  args->attributes は UDF 引数の名前についての情報を通知します。引数 i の場合、属性名は args->attributes[i] の文字列として取得でき、属性の長さは args->attribute_lengths[i] です。文字列がゼロで終わると想定しないでください。

  デフォルトでは、UDF 引数の名前は引数を指定するために使用される式のテキストです。UDF の場合、引数にはオプションの [AS] alias_name 句が含まれていることもあり、この場合の引数名は alias_name です。このため、各引数の attributes 値は、エイリアスが指定されているかどうかに応じて異なります。

  UDF my_udf() が次のように呼び出されたとします。

```
SELECT my_udf(expr1, expr2 AS alias1, expr3 alias2);
```

  この場合、attributes 配列および attribute_lengths 配列の値は次のようになります。

```
args->attributes[0] = "expr1"
args->attribute_lengths[0] = 5

args->attributes[1] = "alias1"
args->attribute_lengths[1] = 6

args->attributes[2] = "alias2"
args->attribute_lengths[2] = 6
```

- unsigned long *attribute_lengths

  attribute_lengths 配列は、各属性名の長さを示しています。

24.3.2.4 UDF の戻り値およびエラー処理

初期化関数は、エラーが発生しなかった場合は 0、およびそうでない場合は 1 を返します。エラーが発生した場合、xxx_init() は message パラメータに NULL で終わるエラーメッセージを格納します。メッセージはクライア

ントに返されます。メッセージバッファーの長さは MYSQL_ERRMSG_SIZE 文字ですが、標準の端末画面の幅にメッセージが収まるように、メッセージを 80 文字未満に維持するようにしてください。

long long 関数および double 関数の場合、メイン関数 xxx() の戻り値は関数値です。文字列関数は結果へのポインタを返し、*length を戻り値の (バイト単位の) 長さに設定します。例:

```
memcpy(result, "result string", 13);
*length = 13;
```

MySQL は、result パラメータを使用してバッファーを xxx() 関数に渡します。このバッファーは 255 文字を保持するための十分な長さがあり、マルチバイト文字も保持できます。xxx() 関数はこのバッファーに結果を格納でき (結果が収まる場合)、その場合は戻り値にそのバッファーへのポインタを設定してください。関数が別のバッファーに結果を格納する場合は、そのバッファーへのポインタを返します。

文字列関数で提供されたバッファーを使用しない場合 (たとえば、255 文字より長い文字列を返す必要がある場合)、xxx_init() 関数または xxx() 関数で malloc() を使用して独自のバッファーの領域を割り当て、xxx_deinit() 関数でその領域を解放する必要があります。以降の xxx() の呼び出しで再利用できるように、割り当てられたメモリーを UDF_INIT 構造体の ptr スロットに格納できます。セクション24.3.2.1「単純な関数のための UDF の呼び出しシーケンス」を参照してください。

メイン関数で戻り値が NULL であることを示すには、*is_null を 1 に設定します。

```
*is_null = 1;
```

メイン関数でエラーを返すことを示すには、*error を 1 に設定します。

```
*error = 1;
```

xxx() がいずれかの行で *error に 1 を設定した場合、XXX() が呼び出されたステートメントによって処理される現在の行および後続の行の関数値は NULL です。(後続の行では xxx() は呼び出されません。)

24.3.2.5 ユーザー定義関数のコンパイルおよびインストール

UDF を実装するファイルは、サーバーが実行されるホストでコンパイルおよびインストールする必要があります。MySQL ソース配布に含まれている UDF ファイルの例 sql/udf_example.cc を使用して、このプロセスについて説明します。

スレーブサーバーにレプリケーションされるステートメントで UDF が参照される場合は、すべてのスレーブでその関数が使用可能である必要があります。そうしないと、スレーブでその関数を呼び出そうとしたときに、レプリケーションがスレーブで失敗します。

次の手順は Unix の場合です。Windows での手順は、このセクションの以降で説明します。

udf_example.cc ファイルには次の関数が含まれています。

- metaphon() は、文字列引数の metaphon 文字列を返します。これは soundex 文字列に似ていますが、英語向けに調整されています。
- myfunc_double() は、引数内の文字の ASCII 値の合計をその引数の長さの合計で割ったものを返します。
- myfunc_int() は、その引数の長さの合計を返します。
- sequence([const int]) は、指定された数値または 1 (数値が指定されなかった場合) から始まるシーケンスを返します。
- lookup() は、ホスト名の IP アドレスを返します。
- reverse_lookup() は、IP アドレスに対するホスト名を返します。この関数は、'xxx.xxx.xxx.xxx' という形式の単一の文字列引数または 4 つの数値のいずれかを使用して呼び出すことができます。
- avgcost() は、平均コストを返します。これは集約関数です。

動的にロード可能なファイルは、次のようなコマンドを使用して、共有可能なオブジェクトファイルとしてコンパイルします。

```
shell> gcc -shared -o udf_example.so udf_example.cc
```

CMake に gcc を使用している場合 (MySQL はそのように構成されています) は、より簡単なコマンドで udf_example.so を作成できます。

```
shell> make udf_example
```

UDF が含まれている共有オブジェクトをコンパイルしたら、共有オブジェクトをインストールして MySQL に通知する必要があります。gcc を使用して udf_example.cc から共有オブジェクトをコンパイルすると、udf_example.so という名前のファイルが直接生成されます。共有オブジェクトをサーバーのプラグインディレクトリにコピーし、udf_example.so という名前を付けます。このディレクトリは、plugin_dir システム変数の値から取得できます。

一部のシステムでは、ダイナミックリンカーを構成する ldconfig プログラムは、共有オブジェクトの名前が lib で始まっていない場合、共有オブジェクトを認識しません。この場合は、たとえば、ファイル名を udf_example.so から libudf_example.so に名前変更してください。

Windows では、次の手順を使用してユーザー定義関数をコンパイルできます。

1. MySQL ソース配布を取得します。セクション2.1.3「MySQL の取得方法」を参照してください。
2. 必要な場合は CMake ビルドユーティリティーを http://www.cmake.org から取得します。(バージョン 2.6 以降が必要となります)。
3. ソースツリーで sql ディレクトリを参照します。その場所には udf_example.def udf_example.cc という名前のファイルがあります。両方のファイルをこのディレクトリから作業ディレクトリにコピーします。
4. 次の内容を持つ CMake の makefile (CMakeLists.txt) を作成します。

```
PROJECT(udf_example)

# Path for MySQL include directory
INCLUDE_DIRECTORIES("c:/mysql/include")

ADD_DEFINITIONS("-DHAVE_DLOPEN")
ADD_LIBRARY(udf_example MODULE udf_example.cc udf_example.def)
TARGET_LINK_LIBRARIES(udf_example wsock32)
```

5. VC プロジェクトファイルおよびソリューションファイルを作成します。

```
cmake -G "<Generator>"
```

   cmake --help を呼び出すと、有効なジェネレータのリストが表示されます。

6. udf_example.dll を作成します。

```
devenv udf_example.sln /build Release
```

共有オブジェクトファイルがインストールされたら、次のステートメントを使用して、新しい関数について mysqld に通知します。使用しているシステムでオブジェクトファイルのサフィクスが .so ではない場合、正しいサフィクスにすべて置き換えます (たとえば、Windows の場合は .dll)。

```
mysql> CREATE FUNCTION metaphon RETURNS STRING SONAME 'udf_example.so';
mysql> CREATE FUNCTION myfunc_double RETURNS REAL SONAME 'udf_example.so';
mysql> CREATE FUNCTION myfunc_int RETURNS INTEGER SONAME 'udf_example.so';
mysql> CREATE FUNCTION sequence RETURNS INTEGER SONAME 'udf_example.so';
mysql> CREATE FUNCTION lookup RETURNS STRING SONAME 'udf_example.so';
mysql> CREATE FUNCTION reverse_lookup
    ->       RETURNS STRING SONAME 'udf_example.so';
mysql> CREATE AGGREGATE FUNCTION avgcost
    ->       RETURNS REAL SONAME 'udf_example.so';
```

関数を削除するには、DROP FUNCTION を使用します。

```
mysql> DROP FUNCTION metaphon;
mysql> DROP FUNCTION myfunc_double;
mysql> DROP FUNCTION myfunc_int;
mysql> DROP FUNCTION sequence;
mysql> DROP FUNCTION lookup;
mysql> DROP FUNCTION reverse_lookup;
mysql> DROP FUNCTION avgcost;
```

CREATE FUNCTION ステートメントおよび DROP FUNCTION ステートメントは、mysql データベースの func システムテーブルを更新します。関数の名前、型、および共有ライブラリ名は、このテーブルに保存されます。関数を作成または削除するには、mysql データベースの INSERT 権限または DELETE 権限をそれぞれ持っている必要があります。

すでに作成されている関数を CREATE FUNCTION を使用して追加しないでください。関数を再インストールする必要がある場合は、DROP FUNCTION を使用して関数を削除してから、CREATE FUNCTION を使用して再インストールする必要があります。これを行う必要があるのは、たとえば、mysqld が関数の新しいバージョンを取

得するように、新しいバージョンを再コンパイルする場合です。そうしないと、サーバーは古いバージョンを使用し続けます。

アクティブ関数とは、CREATE FUNCTION を使用してロードされていて、DROP FUNCTION を使用して削除されていない関数です。すべてのアクティブ関数は、サーバーが起動するたびにリロードされますが、--skip-grant-tables オプションを指定して mysqld を起動した場合は異なります。この場合は、UDF の初期化がスキップされ、UDF は使用できません。

24.3.2.6 ユーザー定義関数のセキュリティー上の予防措置

MySQL では、ユーザー定義関数の誤用を防ぐためのさまざまな対策が講じられています。

UDF オブジェクトファイルは任意のディレクトリに配置できません。これらはサーバーのプラグインディレクトリに配置する必要があります。このディレクトリは、plugin_dir システム変数の値から取得できます。

CREATE FUNCTION または DROP FUNCTION を使用するには、mysql データベースの INSERT 権限または DELETE 権限をそれぞれ持っている必要があります。これが必要となるのは、これらのステートメントが mysql.func テーブルに対して行の追加および削除を行うためです。

UDF には、メインの xxx() 関数に対応する xxx シンボルのほかに、少なくとも 1 つのシンボルが定義してください。これらの補助シンボルは、xxx_init()、xxx_deinit()、xxx_reset()、xxx_clear()、および xxx_add() 関数に対応します。mysqld では、xxx シンボルのみを持つ UDF をロードできるかどうかを制御する --allow-suspicious-udfs オプションもサポートされています。デフォルトでは、このオプションはオフになっており、正当な UDF が含まれている共有オブジェクトファイル以外から関数がロードされることを防いでいます。xxx シンボルのみが含まれていて、補助シンボルを含めるように再コンパイルできない古い UDF がある場合は、--allow-suspicious-udfs オプションを指定する必要がある場合があります。それ以外の場合は、この機能を有効にしないようにしてください。

24.3.3 新しいネイティブ関数の追加

新しいネイティブ MySQL 関数を追加するには、ここで説明している手順を使用します。この手順では、ソース配布を使用する必要があります。ネイティブ関数をバイナリ配布に追加することはできません。その場合、MySQL ソースコードを変更して、変更されたソースから MySQL をコンパイルする必要があるためです。別のバージョンの MySQL に移行する場合 (たとえば、新しいバージョンがリリースされたとき) は、新しいバージョンを使用してこの手順を繰り返す必要があります。

スレーブサーバーにレプリケートされるステートメントで新しいネイティブ関数が参照される場合は、すべてのスレーブサーバーで関数を使用できるようにする必要があります。そうしないと、スレーブでその関数を呼び出そうとしたときに、レプリケーションがスレーブで失敗します。

新しいネイティブ関数を追加するには、次のステップに従って、sql ディレクトリ内のソースファイルを変更します。

1. item_create.cc に関数のためのサブクラスを作成します。
   - 関数が固定された数の引数を受け取る場合は、関数が受け取る引数の数 (0、1、2、または 3 個) に応じて、Create_func_arg0、Create_func_arg1、Create_func_arg2、または Create_func_arg3 のサブクラスをそれぞれ作成します。たとえば、Create_func_uuid、Create_func_abs、Create_func_pow、および Create_func_lpad クラスを参照してください。
   - 関数が受け取る引数の数が可変の場合は、Create_native_func のサブクラスを作成します。たとえば、Create_func_concat を参照してください。
2. SQL ステートメントで関数を参照できる名前を指定するには、次の配列に行を追加することによって item_create.cc に名前を登録します。

   ```
   static Native_func_registry func_array[]
   ```

   同じ関数に対して複数の名前を登録できます。たとえば、"LCASE" および "LOWER" の行を参照してください。これらは Create_func_lcase のエイリアスです。
3. item_func.h に、関数が数値または文字列のいずれを返すかに応じて、Item_num_func または Item_str_func から継承するクラスを宣言します。
4. item_func.cc に、数値関数または文字列関数のいずれを定義するかに応じて、次のいずれかの宣言を追加します。

   ```
   double   Item_func_newname::val()
   longlong Item_func_newname::val_int()
   ```

```
String  *Item_func_newname::Str(String *str)
```

標準的な項目 (Item_num_func など) からオブジェクトを継承する場合は、それらの関数のいずれかを定義するだけで、多くの場合、親オブジェクトによってほかの関数が提供されます。たとえば、Item_str_func クラスは、::str() によって返された値に対して atof() を実行する val() 関数を定義しています。

5. 関数に決定性がない場合は、関数の結果をキャッシュしないことを示すために、次のステートメントを項目のコンストラクタに含めます。

```
current_thd->lex->safe_to_cache_query=0;
```

関数に決定性がない状態とは、引数の値が固定されていても、呼び出しごとに異なる結果が返されることがある場合です。

6. 多くの場合、次のオブジェクト関数も定義します。

```
void Item_func_newname::fix_length_and_dec()
```

この関数は、指定された引数に基づいて、少なくとも max_length を計算します。max_length は関数が返すことができる最大文字数です。メイン関数が NULL 値を返せない場合は、この関数で maybe_null = 0 も設定します。この関数は、引数の maybe_null 変数をチェックすることによって、関数のいずれかの引数が NULL を返すことがあるかどうかをチェックできます。これを実行する方法の典型的な例については、Item_func_mod::fix_length_and_dec を参照してください。

すべての関数はスレッドセーフである必要があります。言い換えると、関数内で相互排他ロックで保護せずにグローバル変数または静的変数を使用しないでください。

::val()、::val_int()、または ::str() から NULL を返す場合は、null_value を 1 に設定して 0 を返します。

::str() オブジェクト関数については、注意すべき追加の考慮事項があります。

- String *str 引数は、結果を保持するために使用できる文字列バッファーを提供します。(String 型の詳細は、sql_string.h ファイルを参照してください。)
- ::str() 関数では、結果が保持されている文字列を返すか、結果が NULL の場合は (char*) 0 を返します。
- 現在のすべての文字列関数は、絶対に必要である場合を除き、メモリーの割り当てを回避しようとします。

24.4 MySQL のデバッグおよび移植

このセクションは、MySQL をほかのオペレーティングシステムに移植する場合に役立ちます。まず、現在サポートされているオペレーティングシステムのリストを確認してください。https://www.mysql.com/support/supportedplatforms/database.html を参照してください。MySQL の新しい移植版を作成した場合は、オラクルにお知らせください。このドキュメントおよび Web サイト (http://www.mysql.com/) にそれを示して、ほかのユーザーに推奨します。

> **注記**
>
> MySQL の新しい移植版を作成した場合は、GPL ライセンスに基づいて自由にコピーおよび配布できますが、MySQL の著作権所有者になるわけではありません。

動作する POSIX スレッドライブラリがサーバーで必要となります。

MySQL をソースからビルドするには、セクション2.9「ソースから MySQL をインストールする」に示されているツール要件をシステムが満たしている必要があります。

> **重要**
>
> IA64 プラットフォームで icc を使用して MySQL 5.6 をビルドしようとしていて、MySQL Cluster のサポートが必要な場合は、まず icc バージョン 9.1.043 以降を使用していることを確認してください。(詳細は、Bug #21875 を参照してください。)

新しい移植版で問題が発生した場合は、MySQL のデバッグを行う必要があることがあります。セクション24.4.1「MySQL サーバーのデバッグ」を参照してください。

> **注記**
>
> mysqld のデバッグを開始する前に、まずテストプログラム mysys/thr_alarm および mysys/thr_lock を使用できるようにします。これにより、スレッドがわずかながらも動作する可能性があります。

24.4.1 MySQL サーバーのデバッグ

MySQL で非常に新しい一部の機能を使用している場合は、--skip-new (安全でない可能性がある新しい機能がすべて無効にされます) を指定して mysqld の実行を試みることができます。セクションB.5.4.2「MySQL が繰り返しクラッシュする場合の対処方法」を参照してください。

mysqld が起動しない場合は、セットアップを妨げている my.cnf ファイルがないことを確認してください。my.cnf の引数をチェックするには mysqld --print-defaults を使用します。mysqld --no-defaults ... を指定して起動するとそれらの引数は使用されません。

mysqld が CPU やメモリーを使い尽くしてしまうようになった場合、または「ハングアップ」する場合は、mysqladmin processlist status を使用すると、ユーザーが長時間かかるクエリーを実行しているかどうかを確認できます。パフォーマンスの問題、または新しいクライアントが接続できないことに関する問題がある場合は、ほかのウィンドウで mysqladmin -i10 processlist status を実行すると役に立つことがあります。

mysqladmin debug コマンドは、使用されているロック、使用されているメモリー、およびクエリーの使用状況に関する情報を MySQL ログファイルにダンプします。これは一部の問題の解決に役立つことがあります。また、このコマンドでは、MySQL をデバッグ用にまだコンパイルしていなくても、役立つ情報が提供されます。

一部のテーブルが徐々に遅くなるという問題がある場合は、OPTIMIZE TABLE または myisamchk を使用してテーブルを最適化することを試みてください。第5章「MySQL サーバーの管理」 を参照してください。また、遅いクエリーを EXPLAIN でチェックしてください。

また、使用している環境に特有である可能性がある問題については、このマニュアルの OS 固有のセクションをお読みください。セクション2.1「一般的なインストールガイド」を参照してください。

24.4.1.1 デバッグのための MySQL のコンパイル

非常に特定された問題がある場合は、MySQL のデバッグを試みることができます。これを実行するには、-DWITH_DEBUG=1 オプションを指定して MySQL を構成する必要があります。mysqld --help を実行すると、MySQL がデバッグ付きでコンパイルされたかどうかを確認できます。--debug フラグがオプションとともに示される場合は、デバッグが有効にされています。この場合は、mysqladmin ver でも mysqld バージョンが mysql ... --debug として示されます。

mysqld に -DWITH_DEBUG=1 CMake オプションを指定して構成するとクラッシュして停止する場合は、MySQL 内のコンパイラのバグまたはタイミングのバグが検出された可能性があります。この場合は、-DWITH_DEBUG=1 を使用せずに、CMAKE_C_FLAGS および CMAKE_CXX_FLAGS CMake オプションを使用して -g の追加を試みることができます。mysqld が異常終了した場合は、少なくとも gdb を使用してそれに接続するか、コアファイルに対して gdb を使用して、発生した事象を確認できます。

MySQL をデバッグ用に構成すると、mysqld のヘルスをモニターする多くの追加の安全チェック機能が自動的に有効になります。「予期しない」現象が検出された場合、stderr にエントリが書き込まれ、これは mysqld_safe によってエラーログに送られます。また、これは、ソース配布を使用していて、MySQL に予期しない問題があった場合、最初に行うことは MySQL をデバッグ用に構成することであることを意味します。(2 番目にすることは、MySQL のメーリングリストにメールを送信して支援を求めることです。セクション1.5.1「MySQL メーリングリスト」 を参照してください。バグを見つけたと思われる場合は、セクション1.6「質問またはバグをレポートする方法」の手順を使用してください。

Windows の MySQL 配布では、mysqld.exe はデフォルトでトレースファイルをサポートするようにコンパイルされています。

24.4.1.2 トレースファイルの作成

mysqld サーバーが起動しない場合、またはサーバーがすぐにクラッシュしてしまう場合は、問題を見つけるためにトレースファイルの作成を試みることができます。

これを行うには、デバッグサポート付きでコンパイルされている mysqld がある必要があります。これは mysqld -V を実行することによって確認できます。バージョン番号が -debug で終わっている場合は、トレースファイルのサポート付きでコンパイルされています。(Windows の場合、MySQL 4.1 以降では、デバッグサーバーの名前は mysqld ではなく mysqld-debug です。)

Unix の場合は /tmp/mysqld.trace、Windows の場合は \mysqld.trace にあるトレースログを使用して、mysqld サーバーを起動します。

```
shell> mysqld --debug
```

Windows では、mysqld をサービスとして起動しないようにするために、--standalone フラグも使用してください。コンソールウィンドウで、次のコマンドを使用します。

```
C:\> mysqld-debug --debug --standalone
```

このあと、2 つ目のコンソールウィンドウで mysql.exe コマンド行ツールを使用して、問題を再現できます。mysqld サーバーを停止するには、mysqladmin shutdown を使用します。

トレースファイルは非常に大きくなることがあります。より小さいトレースファイルが生成されるようにするには、次のようなデバッグオプションを使用できます。

```
mysqld --debug=d,info,error,query,general,where:O,/tmp/mysqld.trace
```

これにより、もっとも関心があるタグの情報のみがトレースファイルに出力されます。

これに関するバグレポートを作成する場合は、動作に問題があると思われるトレースファイルの行のみを、該当するメーリングリストに送信してください。問題のある場所を見つけることができない場合は、バグレポートを開いて、MySQL の開発者が調査できるように、トレースファイルをレポートにアップロードしてください。手順については、セクション1.6「質問またはバグをレポートする方法」を参照してください。

トレースファイルは、Fred Fish が作成した DBUG パッケージによって生成されます。セクション24.4.3「DBUG パッケージ」を参照してください。

24.4.1.3 pdb を使用した Windows のクラッシュダンプの作成

プログラムデータベースファイル (拡張子は pdb) は、MySQL の非インストール配布に含まれています。これらのファイルには、問題が発生したときに MySQL インストール環境をデバッグするための情報が含まれています。

PDB ファイルには、より詳細なトレースファイルおよびダンプファイルを作成できる、mysqld およびその他のツールについての詳細な情報が含まれています。これらをワトソン博士、WinDbg、および Visual Studio とともに使用して、mysqld をデバッグできます。

PDB ファイルについては、Microsoft サポート技術情報の記事 121366 を参照してください。使用可能なデバッグオプションについては、Windows のデバッグツールを参照してください。

ワトソン博士はすべての Windows 配布でインストールされますが、Windows 開発ツールをインストールした場合、ワトソン博士は Visual Studio に付属しているデバッガである WinDbg、または Borland や Delphi で提供されているデバッグツールに置き換えられている可能性があります。

ワトソン博士を使用してクラッシュファイルを生成するには、次の手順に従います。

1. -i オプションを使用して drwtsn32.exe を対話的に実行することによって、ワトソン博士を起動します。

   ```
   C:\> drwtsn32 -i
   ```

2. 「ログ ファイルのパス」にトレースファイルを格納するディレクトリを設定します。
3. 「すべてのスレッド コンテキストをダンプ」および「既存のログ ファイルに追加する」が選択されていることを確認します。
4. 「ダンプ シンボル テーブル」、「メッセージ ボックスによる通知」、「音による通知」、および「クラッシュ ダンプ ファイルの作成」のチェックを外します。
5. 「インストラクションの数」に、スタックトレース内の呼び出しを必要なだけ取得できる値を設定します。値は 25 にすれば十分です。

生成されるファイルは、非常に大きくなる可能性があります。

24.4.1.4 gdb での mysqld のデバッグ

ほとんどのシステムでは、mysqld がクラッシュした場合に詳細な情報を取得するために、gdb からも mysqld を起動できます。

Linux の一部の古い gdb バージョンでは、mysqld のスレッドをデバッグできるようにするには、run --one-thread を使用する必要があります。この場合、一度にアクティブにできるのは 1 つのスレッドのみです。スレッドのデバッグは gdb 5.1 のほうがより良く機能するため、このバージョンにアップグレードすると最適です。

gdb で mysqld を実行すると、NPTL スレッド (Linux の新しいスレッドライブラリ) に起因する問題が発生することがあります。次のような現象が発生します。

- mysqld が起動中 (「接続準備完了」と出力される前) にハングアップする。
- mysqld が pthread_mutex_lock() または pthread_mutex_unlock() の呼び出し中にクラッシュする。

この場合、gdb を起動する前に、シェルで次の環境変数を設定してください。

```
LD_ASSUME_KERNEL=2.4.1
export LD_ASSUME_KERNEL
```

gdb で mysqld を実行するときは、--skip-stack-trace を使用してスタックトレースを無効にし、gdb 内でセグメンテーション違反を捕捉できるようにする必要があります。

MySQL 4.0.14 以降では、mysqld に --gdb オプションを使用してください。これにより、SIGINT 用の割り込みハンドラ (mysqld を ^C で停止してブレークポイントを設定するために必要となります) がインストールされ、スタックトレースおよびコアファイル処理が無効になります。

新しい接続が常時多数発生する場合、gdb は古いスレッドのメモリーを解放しないため、gdb で MySQL をデバッグすることは非常に困難です。この問題を回避するには、thread_cache_size を max_connections + 1 に等しい値に設定して mysqld を起動します。ほとんどの場合、--thread_cache_size=5' を使用するだけでもかなり改善されます。

SIGSEGV シグナルが発生して mysqld が異常終了したときに Linux でコアダンプを取得する場合は、--core-file オプションを指定して mysqld を起動します。このコアファイルは、mysqld が異常終了した理由を見つけるために役立つことがあるバックトレースを作成するために使用できます。

```
shell> gdb mysqld core
gdb>   backtrace full
gdb>   quit
```

セクションB.5.4.2「MySQL が繰り返しクラッシュする場合の対処方法」を参照してください。

Linux で gdb 4.17.x 以降を使用している場合は、次の情報を持つ .gdb ファイルを現在のディレクトリにインストールしてください。

```
set print sevenbit off
handle SIGUSR1 nostop noprint
handle SIGUSR2 nostop noprint
handle SIGWAITING nostop noprint
handle SIGLWP nostop noprint
handle SIGPIPE nostop
handle SIGALRM nostop
handle SIGHUP nostop
handle SIGTERM nostop noprint
```

gdb を使用したスレッドのデバッグに問題がある場合、gdb 5.x をダウンロードし、これを代わりに試してみてください。新しいバージョンの gdb ではスレッド処理が非常に改善されています。

mysqld をデバッグする方法の例を次に示します。

```
shell> gdb /usr/local/libexec/mysqld
gdb> run
...
backtrace full # Do this when mysqld crashes
```

上記の出力をバグレポートに含め、セクション1.6「質問またはバグをレポートする方法」の手順を使用してバグレポートを提出できます。

mysqld がハングアップした場合は、strace、/usr/proc/bin/pstack などのシステムツールを使用して、mysqld がハングアップした場所を調査できます。

```
strace /tmp/log libexec/mysqld
```

Perl の DBI インタフェースを使用している場合は、trace メソッドを使用するか、DBI_TRACE 環境変数を設定することによって、デバッグ情報をオンにできます。

24.4.1.5 スタックトレースの使用

オペレーティングシステムによっては、mysqld が予期せずに異常終了した場合に、エラーログにスタックトレースが含まれています。これを使用して、mysqld が異常終了した場所 (および多くの場合その理由) を見つけることができます。セクション5.2.2「エラーログ」 を参照してください。スタックトレースを取得するには、-fomit-

frame-pointer オプションを gcc に指定して mysqld をコンパイルしないでください。セクション24.4.1.1「デバッグのための MySQL のコンパイル」を参照してください。

エラーログのスタックトレースは次のように出力されます。

```
mysqld got signal 11;
Attempting backtrace. You can use the following information
to find out where mysqld died. If you see no messages after
this, something went terribly wrong...

stack_bottom = 0x41fd0110 thread_stack 0x40000
mysqld(my_print_stacktrace+0x32)[0x9da402]
mysqld(handle_segfault+0x28a)[0x6648e9]
/lib/libpthread.so.0[0x7f1a5af000f0]
/lib/libc.so.6(strcmp+0x2)[0x7f1a5a10f0f2]
mysqld(_Z21check_change_passwordP3THDPKcS2_Pcj+0x7c)[0x7412cb]
mysqld(_ZN16set_var_password5checkEP3THD+0xd0)[0x688354]
mysqld(_Z17sql_set_variablesP3THDP4ListI12set_var_baseE+0x68)[0x688494]
mysqld(_Z21mysql_execute_commandP3THD+0x41a0)[0x67a170]
mysqld(_Z11mysql_parseP3THDPKcjPS2_+0x282)[0x67f0ad]
mysqld(_Z16dispatch_command19enum_server_commandP3THDPcj+0xbb7[0x67fdf8]
mysqld(_Z10do_commandP3THD+0x24d)[0x6811b6]
mysqld(handle_one_connection+0x11c)[0x66e05e]
```

トレースの関数名の解決に失敗した場合、トレースに格納される情報が少なくなります。

```
mysqld got signal 11;
Attempting backtrace. You can use the following information
to find out where mysqld died. If you see no messages after
this, something went terribly wrong...

stack_bottom = 0x41fd0110 thread_stack 0x40000
[0x9da402]
[0x6648e9]
[0x7f1a5af000f0]
[0x7f1a5a10f0f2]
[0x7412cb]
[0x688354]
[0x688494]
[0x67a170]
[0x67f0ad]
[0x67fdf8]
[0x6811b6]
[0x66e05e]
```

後者の場合は、resolve_stack_dump ユーティリティーを使用して、次の手順を使用することによって、mysqld が異常終了した場所を判別できます。

1. スタックトレースから mysqld.stack などのファイルに数値をコピーします。この数値には、囲んでいた角括弧を含めないでください。

   ```
   0x9da402
   0x6648e9
   0x7f1a5af000f0
   0x7f1a5a10f0f2
   0x7412cb
   0x688354
   0x688494
   0x67a170
   0x67f0ad
   0x67fdf8
   0x6811b6
   0x66e05e
   ```

2. mysqld サーバーのシンボルファイルを作成します。

   ```
   shell> nm -n libexec/mysqld > /tmp/mysqld.sym
   ```

   mysqld が静的にリンクされていない場合は、代わりに次のコマンドを使用します。

   ```
   shell> nm -D -n libexec/mysqld > /tmp/mysqld.sym
   ```

   C++ のシンボルをデコードする場合は、nm に対して --demangle を使用します (使用できる場合)。使用しているバージョンの nm にこのオプションがない場合は、スタックダンプが生成されたあとに c++filt コマンドを使用して、C++ 名をデマングルする必要があります。

3. 次のコマンドを実行します。

```
shell> resolve_stack_dump -s /tmp/mysqld.sym -n mysqld.stack
```

デマングルされた C++ 名をシンボルファイルに含めることができなかった場合は、c++filt を使用して resolve_stack_dump の出力を処理します。

```
shell> resolve_stack_dump -s /tmp/mysqld.sym -n mysqld.stack | c++filt
```

これにより、mysqld が異常終了した場所が出力されます。この出力が、mysqld が異常終了した理由を見つけるために役立たない場合は、バグレポートを作成して、上記のコマンドの出力をバグレポートに含めてください。

ただし、多くの場合、スタックトレースだけがあっても問題の原因を見つけるために役立ちません。バグを見つけたり回避策を提供したりするためには、ほとんどの場合、mysqld が強制終了されたステートメントを知る必要があり、問題を再現できるテストケースがあれば役に立ちます。セクション1.6「質問またはバグをレポートする方法」を参照してください。

新しいバージョンの glibc スタックトレース関数では、オブジェクトへの相対アドレスも出力されます。glibc ベースのシステム (Linux) では、プラグイン内でのクラッシュのトレースは次のようになります。

```
plugin/auth/auth_test_plugin.so(+0x9a6)[0x7ff4d11c29a6]
```

相対アドレス (+0x9a6) をファイル名および行番号に変換するには、次のコマンドを使用します。

```
shell> addr2line -fie auth_test_plugin.so 0x9a6
auth_test_plugin
mysql-trunk/plugin/auth/test_plugin.c:65
```

addr2line ユーティリティーは Linux の binutils パッケージの一部です。

Solaris でも手順は同様です。Solaris の printstack() では、相対アドレスがすでに出力されています。

```
plugin/auth/auth_test_plugin.so:0x1510
```

これを変換するには、次のコマンドを使用します。

```
shell> gaddr2line -fie auth_test_plugin.so 0x1510
mysql-trunk/plugin/auth/test_plugin.c:88
```

Windows では、アドレス、関数名、および行がすでに出力されています。

```
000007FEF07E10A4 auth_test_plugin.dll!auth_test_plugin()[test_plugin.c:72]
```

24.4.1.6 mysqld でのエラーの原因を見つけるためのサーバーログの使用

一般クエリーログを有効にして mysqld を起動する前に、myisamchk を使用してすべてのテーブルをチェックしてください。第5章「MySQL サーバーの管理」 を参照してください。

mysqld が異常終了またはハングアップする場合は、一般クエリーログを有効にして mysqld を起動してください。セクション5.2.3「一般クエリーログ」を参照してください。mysqld がふたたび異常終了したら、ログファイルの最後の部分を調査して、mysqld が強制終了されたクエリーを見つけることができます。

デフォルトの一般クエリーログファイルを使用した場合、ログはデータベースディレクトリに host_name.log として格納されます。ほとんどの場合、mysqld が強制終了されたのはログファイル内の最後のクエリーですが、可能であれば、mysqld を再起動して、見つかったクエリーを mysql コマンド行ツールから実行することによって、このことを検証してください。これが動作する場合は、完了しなかった複雑なクエリーもすべてテストしてください。

また、長い時間がかかるすべての SELECT ステートメントに対して EXPLAIN コマンドを試すことで、mysqld がインデックスを適切に使用していることを確認できます。セクション13.8.2「EXPLAIN 構文」を参照してください。

実行に長い時間がかかるクエリーを見つけるには、スロークエリーログを有効にして mysqld を起動します。セクション5.2.5「スロークエリーログ」を参照してください。

エラーログファイル (通常は hostname.err という名前) に mysqld restarted というテキストがあった場合は、mysqld でエラーが発生した原因であるクエリーが見つかった可能性があります。これが発生した場合、myisamchk を使用してすべてのテーブルをチェックし (第5章「MySQL サーバーの管理」を参照してください)、MySQL ログファイル内のクエリーをテストして、失敗するかどうかを確認します。そのようなクエリーが

見つかった場合は、まず最新バージョンの MySQL にアップグレードすることを試してください。これで解決されず、mysql のメールアーカイブで参考になる回答が見つからない場合は、MySQL メーリングリストにバグを報告してください。メーリングリストについては http://lists.mysql.com/ で説明されており、アーカイブのオンラインリストへのリンクもあります。

--myisam-recover-options を指定して mysqld を起動した場合、MySQL は「not closed properly」または「crashed」としてマークされている MyISAM テーブルを自動的にチェックして修復しようとします。これが発生した場合、MySQL は hostname.err ファイルに「警告: テーブル ... をチェックしています」と書き込み、テーブルを修復する必要がある場合は、「警告: テーブルを修復しています」がそのあとに書き込まれます。これらのエラーを多数受け取り、その直前に予期しない mysqld の停止がなかった場合は、何らかの問題があるため、さらに調査する必要があります。セクション5.1.3「サーバーコマンドオプション」を参照してください。

MySQL 5.6 では、サーバーが MyISAM テーブルの破損を検出すると、追加の情報 (ソースファイルの名前と行番号、テーブルにアクセスしていたスレッドのリストなど) をエラーログに書き込みます。たとえば、「thread_id=1 からエラーを受け取りました。mi_dynrec.c:368」です。これは、バグレポートに含めると役に立つ情報です。

mysqld が予期せず異常終了することは良い兆候ではありませんが、この場合は Checking table... メッセージを調査するのではなく、mysqld が異常終了した原因を見つけるようにしてください。

24.4.1.7 テーブルが破損した場合のテストケースの作成

テーブルが破損した場合、または一部の更新コマンドのあとに mysqld で常に障害が発生する場合は、次の手順を行うことによってこのバグが再現可能かどうかをテストできます。

- MySQL デーモンを停止します (mysqladmin shutdown を使用します)。
- テーブルのバックアップを作成します (非常にまれですが、修復によって何らかの障害が発生する場合に対して保護するため)。
- myisamchk -s database/*.MYI を使用してすべてのテーブルをチェックします。不正なテーブルがあった場合は、myisamchk -r database/table.MYI を使用して修復します。
- テーブルの 2 番目のバックアップを作成します。
- より多くの領域が必要な場合は、MySQL データディレクトリから古いログファイルを削除 (または移動) します。
- バイナリログを有効にして mysqld を起動します。mysqld がクラッシュするクエリーを見つける場合は、一般クエリーログも有効にしてサーバーを起動します。セクション5.2.3「一般クエリーログ」およびセクション5.2.4「バイナリログ」を参照してください。
- テーブルがクラッシュしたら、mysqld サーバーを停止します。
- バックアップをリストアします。
- バイナリログを有効にせずに、mysqld サーバーを再起動します。
- mysqlbinlog binary-log-file | mysql を指定してコマンドを再実行します。バイナリログは、hostname-bin.NNNNNN という名前で MySQL データベースディレクトリに保存されます。
- テーブルがふたたび破損したか、上記のコマンドで mysqld が異常終了する場合は、簡単に修正できる可能性がある再現可能なバグが見つかりました。セクション1.6「質問またはバグをレポートする方法」の手順を使用して、テーブルおよびバイナリログをバグデータベースに FTP で送信してください。サポートのお客様の場合は、MySQL カスタマサポートセンター (http://www.mysql.com/support/) を使用して MySQL チームにその問題を通知し、可能な限り早く修正してもらうことができます。

24.4.2 MySQL クライアントのデバッグ

統合デバッグパッケージを使用して MySQL クライアントをデバッグできるようにするには、-DWITH_DEBUG=1 を指定して MySQL を構成します。セクション2.9.4「MySQL ソース構成オプション」を参照してください。

クライアントを実行する前に、MYSQL_DEBUG 環境変数を設定します。

```
shell> MYSQL_DEBUG=d:t:O,/tmp/client.trace
shell> export MYSQL_DEBUG
```

これにより、クライアントは /tmp/client.trace にトレースファイルを生成します。

独自のクライアントコードに問題がある場合は、動作することがわかっているクライアントを使用してサーバーに接続し、クエリーを実行してください。これを行うには、mysql をデバッグモードで実行します (デバッグを有効にして MySQL をコンパイルしたことを想定しています)。

```
shell> mysql --debug=d:t:O,/tmp/client.trace
```

これにより、バグレポートをメール送信するときに役立つ情報が得られます。セクション1.6「質問またはバグをレポートする方法」を参照してください。

クライアントが「正しい」ように見えるコードでクラッシュしている場合は、mysql.h インクルードファイルが MySQL のライブラリファイルと一致していることを確認してください。非常によくある間違いは、古い MySQL インストール環境にある古い mysql.h ファイルを新しい MySQL ライブラリとともに使用していることです。

24.4.3 DBUG パッケージ

MySQL サーバーおよびほとんどの MySQL クライアントは、もともと Fred Fish によって作成された DBUG パッケージとともにコンパイルされます。MySQL をデバッグ用に構成した場合は、このパッケージによって、プログラムが実行している内容のトレースファイルを取得できるようになります。セクション24.4.1.2「トレースファイルの作成」を参照してください。

このセクションでは、デバッグサポート付きでビルドされた MySQL プログラムのコマンド行のデバッグオプションに指定できる引数値をまとめています。DBUG パッケージを使用したプログラミングについては、MySQL ソース配布の dbug ディレクトリにある DBUG マニュアルを参照してください。最新の DBUG マニュアルを入手するには、最新の配布を使用してください。

DBUG パッケージは、--debug[=debug_options] または -# [debug_options] オプションを指定してプログラムを起動することによって使用できます。--debug または -# オプションを指定して、debug_options 値を指定しない場合、ほとんどの MySQL プログラムではデフォルト値が使用されます。サーバーのデフォルトは、Unix の場合は d:t:i:o,/tmp/mysqld.trace、Windows の場合は d:t:i:O,\mysqld.trace です。このデフォルトには次のような効果があります。

- d: すべてのデバッグマクロの出力を有効にします
- t: 関数の呼び出しおよび終了をトレースします
- i: 出力行に PID を追加します
- o,/tmp/mysqld.trace、O,\mysqld.trace: デバッグ出力ファイルを設定します

ほとんどのクライアントプログラムでは、プラットフォームにかかわらず、デフォルトの debug_options 値である d:t:o,/tmp/program_name.trace が使用されます。

シェルのコマンド行で指定されることがある、デバッグ制御文字列のいくつかの例を次に示します。

```
--debug=d:t
--debug=d:f,main,subr1:F:L:t,20
--debug=d,input,output,files:n
--debug=d:t:i:O,\\mysqld.trace
```

mysqld の場合は、debug システム変数を設定することによって、DBUG 設定を実行時に変更することもできます。この変数にはグローバル値とセッション値があります。

```
mysql> SET GLOBAL debug = 'debug_options';
mysql> SET SESSION debug = 'debug_options';
```

実行時に変更するには、セッション値であっても SUPER 権限が必要となります。

debug_options 値は、コロンで区切られた一連のフィールドです。

```
field_1:field_2:...:field_N
```

この値内の各フィールドは必須のフラグ文字で構成され、フラグ文字の前に + 文字または - 文字、およびフラグ文字の後ろにカンマ区切りの修飾子のリストがオプションで付加されることがあります。

```
[+|-]flag[,modifier,modifier,...,modifier]
```

次の表は、許可されるフラグ文字を示しています。認識されないフラグ文字は暗黙のうちに無視されます。

フラグ	説明

d	DBUG_XXX マクロからの現在の状態に関する出力を有効にします。キーワードのリストがあとに続くことがあり、そのキーワードを使用する DBUG マクロの出力のみが有効になります。キーワードのリストが空の場合は、すべてのマクロの出力が有効になります。 MySQL では、一般的に有効にされるデバッグマクロのキーワードは、enter、exit、error、warning、info、および loop です。
D	各デバッガの出力行のあとに待機します。引数は 0.1 秒単位の待機時間であり、マシンの能力の影響を受けます。たとえば、D,20 は 2 秒の待機を指定します。
f	デバッグ、トレース、およびプロファイリングの対象を指定された関数のリストに制限します。空のリストの場合はすべての関数が有効になります。適切な d フラグまたは t フラグを指定する必要があり、それらのフラグが有効な場合にのみ、このフラグはそれらのフラグのアクションを制限します。
F	デバッグ出力またはトレース出力の各行にソースファイル名を示します。
i	デバッグ出力またはトレース出力の各行に PID またはスレッド ID でプロセスを示します。
L	デバッグ出力またはトレース出力の各行にソースファイルの行番号を示します。
n	デバッグ出力またはトレース出力の各行に現在の関数のネストの深さを出力します。
N	デバッグ出力の各行に番号を付けます。
o	デバッガの出力ストリームを指定されたファイルにリダイレクトします。デフォルトの出力先は stderr です。
O	o と似ていますが、ファイルは書き込みごとに実際にはフラッシュされます。必要な場合、ファイルが書き込みごとに閉じられてふたたび開きます。
p	デバッガアクションを指定されたプロセスに限定します。デバッガアクションが実行されるためには、プロセスが DBUG_PROCESS マクロで識別され、リスト内のプロセスと一致する必要があります。
P	デバッグ出力またはトレース出力の各行に現在のプロセス名を出力します。
r	新しい状態をプッシュするときに、前の状態の関数のネストレベルを継承しません。出力を左マージンから開始する場合に便利です。
S	_sanity() から 0 以外が返されるまで、デバッグされる各関数で関数 _sanity(_file_,_line_) を実行します。
t	関数の呼び出し/終了のトレース行を有効にします。最大のトレースレベルを示す数値を指定するリスト (修飾子が 1 つだけ含まれています) があとに続く場合があり、それを超えるとデバッグマクロまたはトレースマクロの出力は行われません。デフォルトはコンパイル時のオプションです。

フラグの前の + 文字や - 文字、およびフラグの後ろに続く修飾子のリストは、d、f などのフラグ文字に対して使用して、該当するすべての修飾子または一部の修飾子に対してデバッグ操作を有効にできます。

- フラグの前に + または - がない場合、フラグ値は指定された修飾子リストのとおりに設定されます。
- フラグの前に + または - がある場合は、リスト内の修飾子が現在の修飾子リストに対して追加または削除されます。

次の例は、d フラグでのこの動作を示しています。d のリストが空の場合は、すべてのデバッグマクロの出力が有効になります。リストが空でない場合は、リスト内のマクロキーワードの出力のみが有効になります。

次のステートメントでは、指定されたとおりに d 値が修飾子リストに設定されます。

```
mysql> SET debug = 'd';
mysql> SELECT @@debug;
+---------+
| @@debug |
+---------+
| d       |
+---------+
mysql> SET debug = 'd,error,warning';
mysql> SELECT @@debug;
+-----------------+
| @@debug         |
+-----------------+
| d,error,warning |
+-----------------+
```

フラグの前の + または - は、現在の d 値に対して追加または削除を行います。

```
mysql> SET debug = '+d,loop';
mysql> SELECT @@debug;
```

```
+---------------------+
| @@debug             |
+---------------------+
| d,error,warning,loop |
+---------------------+
mysql> SET debug = '-d,error,loop';
mysql> SELECT @@debug;
+-----------+
| @@debug   |
+-----------+
| d,warning |
+-----------+
```

「すべてのマクロが有効な状態」に対して追加した場合は、何も変更されません。

```
mysql> SET debug = 'd';
mysql> SELECT @@debug;
+---------+
| @@debug |
+---------+
| d       |
+---------+
mysql> SET debug = '+d,loop';
mysql> SELECT @@debug;
+---------+
| @@debug |
+---------+
| d       |
+---------+
```

有効なすべてのマクロを無効にすると、d フラグは完全に無効になります。

```
mysql> SET debug = 'd,error,loop';
mysql> SELECT @@debug;
+--------------+
| @@debug      |
+--------------+
| d,error,loop |
+--------------+
mysql> SET debug = '-d,error,loop';
mysql> SELECT @@debug;
+---------+
| @@debug |
+---------+
|         |
+---------+
```

> **注記**
>
> MySQL 5.6.12 より前では、+ および - の修飾子が正しく処理されない場合があり、フラグ値が不正な状態のままになる可能性があります。debug の設定順序を事前に確認するか、+ または - を使用せずに設定してください。

第 25 章 MySQL Enterprise Edition

目次

25.1 MySQL Enterprise Monitor 2833
25.2 MySQL Enterprise Backup 2834
25.3 MySQL Enterprise Security 2834
25.4 MySQL Enterprise Encryption 2835
25.5 MySQL Enterprise Audit 2835
25.6 MySQL Enterprise Thread Pool 2835

MySQL Enterprise Edition は商用製品です。MySQL Community Edition と同様に、MySQL Enterprise Edition には MySQL Server が含まれており、これは、完全なコミット、ロールバック、クラッシュリカバリ、および行レベルのロックの機能を持つ、完全に統合されたトランザクションセーフな ACID 準拠のデータベースです。また、MySQL Enterprise Edition には、モニタリングおよびオンラインバックアップに加えて、改良されたセキュリティーおよびスケーラビリティーを提供するように設計された次のコンポーネントが含まれています。

以降のセクションでは、これらの各コンポーネントについて簡単に説明し、より詳細な情報がある場所を示します。商用製品については、https://www.mysql.com/products/ を参照してください。

- MySQL Enterprise Monitor
- MySQL Enterprise Backup
- MySQL Enterprise Security
- MySQL Enterprise Encryption
- MySQL Enterprise Audit
- MySQL Enterprise Thread Pool

25.1 MySQL Enterprise Monitor

MySQL Enterprise Monitor は MySQL のエンタープライズモニタリングシステムであり、MySQL サーバーを監視して、潜在的な問題を通知し、問題を修正する方法をアドバイスします。MySQL Enterprise Monitor は、ビジネスで重要な単一の MySQL サーバーから、アクセスの多い Web サイトをささえる大規模な施設の MySQL サーバーまでのすべての種類の構成をモニターできます。

以降では、MySQL Enterprise Monitor 製品を構成する基本的なコンポーネントについて簡単に概要を示します。詳細は、https://dev.mysql.com/doc/mysql-monitor/en/ で閲覧できる MySQL Enterprise Monitor のマニュアルを参照してください。

MySQL Enterprise Monitor コンポーネントは、データベースおよびネットワークのトポロジに応じてさまざまな構成でインストールでき、データベースサーバーのマシンのオーバーヘッドを最小限にして、信頼できる応答のよいモニタリングデータの最適な組み合わせが提供されます。MySQL Enterprise Monitor の一般的なインストール環境は次のもので構成されています。

- モニターする 1 つ以上の MySQL サーバー。MySQL Enterprise Monitor は MySQL サーバーの Community リリースおよび Enterprise リリースの両方をモニターできます。
- モニターされる各ホストの MySQL Enterprise Monitor Agent。
- エージェントからの情報を照合して、収集されたデータへのユーザーインタフェースを提供する単一の MySQL Enterprise Service Manager。

MySQL Enterprise Monitor は 1 つ以上の MySQL サーバーをモニターするように設計されています。モニタリング情報は、エージェント MySQL Enterprise Monitor Agent を使用して収集されます。このエージェントは、モニターするホストおよび MySQL サーバーとやり取りして、変数、ステータス、およびヘルス情報を収集し、この情報を MySQL Enterprise Service Manager に送信します。

エージェントによって収集されたモニタリングしている各 MySQL サーバーおよびホストの情報は、MySQL Enterprise Service Manager に送信されます。このサーバーは、エージェントからのすべての情報を照合します。

エージェントによって送信された情報を照合するときに、MySQL Enterprise Service Manager はサーバーのステータスを適切な値と比較して、収集されたデータを頻繁にテストします。しきい値に達すると、サーバーはイベント (アラームおよび通知を含む) をトリガーして潜在的な問題 (メモリー不足、高い CPU 使用率、より複雑な状況 (バッファーサイズの不足、ステータス情報など) など) を強調できます。各テストは、それに関連付けられているしきい値と合わせて、ルールと呼ばれます。

これらのルール、アラーム、および通知は、それぞれ MySQL Enterprise Advisor と呼ばれます。Advisors は MySQL Enterprise Service Manager の重要な部分を形成しており、警告情報および潜在的な問題に関するトラブルシューティングの推奨事項を提供します。

MySQL Enterprise Service Manager には Web サーバーが含まれており、ユーザーは Web ブラウザを使用してやり取りします。このインタフェース (MySQL Enterprise Monitor User Interface) には、エージェントによって収集されたすべての情報が表示され、すべてのサーバーおよびそれらの現在のステータスをグループで表示したり、個別に表示したりできます。サービスのすべての特性を MySQL Enterprise Monitor User Interface を使用して制御および構成します。

MySQL Enterprise Monitor Agent の処理によって提供される情報には、グラフ形式で表示できる統計およびクエリーの情報も含まれています。たとえば、サーバーの負荷、クエリーの数、インデックスの使用状況の情報などの特性をグラフとして時間の経過に従って表示できます。このグラフを使用すると、サーバー上での問題や潜在的な問題が正確に特定され、特定の期間のデータを検査することによって、データベースの問題または外部の問題 (外部システムまたはネットワーク障害など) からの影響を診断するために役立ちます。

MySQL Enterprise Monitor Agent は、サーバーで実行されたクエリーに関する詳細な情報 (行数および各クエリーを実行するためにかかった実行時間を含む) を収集するように構成することもできます。詳細なクエリーデータをグラフィカルな情報に関連付けることによって、著しく高い負荷やインデックスなどの問題が発生したときに実行されていたクエリーを識別できます。クエリーデータはクエリーアナライザと呼ばれるシステムによってサポートされており、ニーズに応じて異なる方法でデータを表示できます。

25.2 MySQL Enterprise Backup

MySQL Enterprise Backup は、MySQL データベースに対してホットバックアップ操作を実行します。この製品は、InnoDB ストレージエンジンによって作成されたテーブルの信頼できる効率的なバックアップが行われるように設計されています。補完するために、MyISAM およびほかのストレージエンジンのテーブルをバックアップすることもできます。

以降では、MySQL Enterprise Backup の概要を簡単に示します。詳細は、https://dev.mysql.com/doc/mysql-enterprise-backup/en/ で閲覧できる MySQL Enterprise Backup のマニュアルを参照してください。

ホットバックアップは、データベースが実行されていて、アプリケーションがそれに対して読み取りおよび書き込みを行なっている間に実行します。このタイプのバックアップは、通常のデータベースの操作をブロックせず、バックアップが行われているときに発生したすべての変更を取得します。これらの理由で、ホットバックアップが望ましいのは、データベースが「拡張」され、データが大きくなってバックアップ処理に長い時間がかかるようになったとき、およびデータがビジネスにとって重要になって、アプリケーション、Web サイト、または Web サービスをオフラインにすることなく、最後に行われたすべての変更を取得する必要があるようになったときです。

MySQL Enterprise Backup は、InnoDB ストレージエンジンを使用するすべてのテーブルのホットバックアップを実行します。MyISAM またはその他の InnoDB 以外のストレージエンジンを使用しているテーブルの場合、データベースの実行は継続されるがバックアップされている間はテーブルを変更できない「ウォーム」バックアップが行われます。バックアップ操作を効率的にするために、InnoDB を新しいテーブルのデフォルトのストレージエンジンに指定するか、InnoDB ストレージエンジンを使用するように既存のテーブルを変更できます。

25.3 MySQL Enterprise Security

MySQL Enterprise Edition は、外部サービスを使用した認証を実装するプラグインを提供しています。

- MySQL Enterprise Edition には、MySQL サーバーが PAM (Pluggable Authentication Modules) を使用して MySQL ユーザーを認証できる認証プラグインが含まれています。PAM を使用すると、システムは標準インタフェースを使用して、さまざまな種類の認証方式 (Unix パスワードや LDAP ディレクトリなど) にアクセスできます。
- MySQL Enterprise Edition には、Windows で外部認証を実行する認証プラグインが含まれており、MySQL サーバーが Windows のネイティブサービスを使用してクライアントの接続を認証できます。Windows にログインしたユーザーは、追加のパスワードを指定せずに、自分の環境内の情報に基づいて MySQL クライアントプログラムからサーバーに接続できます。

詳細は、セクション6.3.8.5「PAM 認証プラグイン」およびセクション6.3.8.6「Windows ネイティブ認証プラグイン」を参照してください。

ほかの関連するエンタープライズセキュリティー機能については、セクション25.4「MySQL Enterprise Encryption」を参照してください。

25.4 MySQL Enterprise Encryption

MySQL 5.6 では、MySQL Enterprise Edition には SQL レベルで OpenSSL 機能を提供する OpenSSL ライブラリに基づく一連の暗号化関数が含まれています。これらの関数を使用することによって、エンタープライズアプリケーションが次の操作を実行できるようになります。

- 公開鍵非対称暗号方式を使用した、追加のデータ保護の実装
- 公開鍵、秘密鍵、およびデジタル署名の作成
- 非対称暗号化および非対称復号化の実行
- デジタル署名およびデータの検証や妥当性検査に対する暗号化ハッシュの使用

詳細は、セクション12.17「MySQL Enterprise Encryption の関数」を参照してください。

25.5 MySQL Enterprise Audit

MySQL Enterprise Edition には、サーバープラグインを使用して実装される MySQL Enterprise Audit が含まれています。MySQL Enterprise Audit はオープンな MySQL Audit API を使用し、特定の MySQL サーバーに対して実行された接続およびクエリーのアクティビティーについて、標準のポリシーベースのモニタリングおよびロギングを行うことができます。MySQL Enterprise Audit は、オラクルの監査仕様を満たすように設計されており、内部および外部の規制ガイドラインの両方に管理されているアプリケーションに対して、そのままで簡単に使用できる監査およびコンプライアンスのソリューションを提供しています。

インストール時に監査プラグインを使用すると、MySQL サーバーはサーバーアクティビティーの監査レコードを含むログファイルを生成できます。ログの内容には、クライアントが接続および切断した時間、接続中に実行したアクション (アクセスしたデータベースおよびテーブルなど) が含まれます。

詳細は、セクション6.3.12「MySQL Enterprise Audit ログプラグイン」を参照してください。

25.6 MySQL Enterprise Thread Pool

MySQL Enterprise Edition には、サーバープラグインを使用して実装される MySQL Thread Pool が含まれています。MySQL サーバーのデフォルトのスレッド処理モデルでは、クライアント接続ごとに 1 つのスレッドを使用してステートメントが実行されます。より多くのクライアントがサーバーに接続してステートメントを実行すると、全体的なパフォーマンスが低下します。MySQL Enterprise Edition では、オーバーヘッドを減らしてパフォーマンスを向上させる代替のスレッド処理モデルが、スレッドプールプラグインによって提供されています。このプラグインは、多数のクライアント接続に対してステートメント実行スレッドを効率的に管理することによってサーバーのパフォーマンスを向上させるスレッドプールを実装します。

詳細は、セクション8.11.6「スレッドプールプラグイン」を参照してください。

第 26 章 MySQL Workbench

MySQL Workbench は MySQL Server およびデータベースを操作するためのグラフィカルツールを提供します。MySQL Workbench は MySQL Server バージョン 5.1 以降を完全にサポートしています。MySQL Server 5.0 とも互換性がありますが、5.0 のすべての機能がサポートされるとは限りません。MySQL Server バージョン 4.x はサポートされません。

以降では、MySQL Workbench の機能について簡単に説明します。詳細は、https://dev.mysql.com/doc/workbench/en/で閲覧できる MySQL Workbench のマニュアルを参照してください。

MySQL Workbench は 5 つの主な機能領域を提供しています。

- SQL の開発: データベースサーバーへの接続を作成および管理できます。接続パラメータを構成できるほかに、MySQL Workbench は組み込みの SQL Editor を使用してデータベース接続で SQL クエリーを実行する機能を提供しています。この機能は、以前は Query Browser スタンドアロンアプリケーションによって提供されていた機能と置き換わるものです。
- データモデリング: データベーススキーマのモデルのグラフィカルな作成、スキーマとライブデータベースの間のリバースエンジニアリングとフォワードエンジニアリング、および包括的な Table Editor を使用したデータベースのすべての特性の編集を行うことができます。Table Editor は、テーブル、カラム、インデックス、トリガー、パーティション化、オプション、挿入と権限、ルーチン、およびビューを編集するための使いやすい機能を提供しています。
- サーバー管理: サーバーインスタンスを作成および管理できます。
- データ移行: Microsoft SQL Server、Sybase ASE、SQLite、SQL Anywhere、PostreSQL、およびその他の RDBMS のテーブル、オブジェクト、およびデータを MySQL に移行できます。移行では、以前のバージョンの MySQL から最新のリリースへの移行もサポートされます。
- MySQL エンタープライズサポート: MySQL Enterprise Backup および MySQL Audit などのエンタープライズ製品をサポートします。

MySQL Workbench は Community Edition と Commercial Edition の 2 つのエディションで使用できます。Community Edition は無料で使用できます。Commercial Edition では、データベースドキュメントの生成などの追加のエンタープライズ機能が低価格で提供されています。

付録 A MySQL 5.6 のよくある質問

目次

A.1 MySQL 5.6 FAQ: 全般 2839
A.2 MySQL 5.6 FAQ: ストレージエンジン 2840
A.3 MySQL 5.6 FAQ: サーバー SQL モード 2841
A.4 MySQL 5.6 FAQ: ストアドプロシージャーおよびストアドファンクション 2842
A.5 MySQL 5.6 FAQ: トリガー 2845
A.6 MySQL 5.6 FAQ: ビュー 2847
A.7 MySQL 5.6 FAQ: INFORMATION_SCHEMA 2848
A.8 MySQL 5.6 FAQ: 移行 2848
A.9 MySQL 5.6 FAQ: セキュリティー 2849
A.10 MySQL 5.6 FAQ: MySQL Cluster 2850
A.11 MySQL 5.6 FAQ: MySQL の中国語、日本語、および韓国語の文字セット 2861
A.12 MySQL 5.6 FAQ: コネクタおよび API 2871
A.13 MySQL 5.6 FAQ: レプリケーション 2871
A.14 MySQL 5.6 FAQ: MySQL エンタープライズスケーラビリティースレッドプール 2874

A.1 MySQL 5.6 FAQ: 全般

A.1.1 本番で使用 (GA) できる MySQL はどのバージョンですか。 2839
A.1.2 開発 (GA ではない) バージョンの状況を教えてください。 2839
A.1.3 MySQL 5.6 ではサブクエリーを実行できますか。 2839
A.1.4 MySQL 5.6 では複数テーブルへの挿入、更新、および削除を実行できますか。 2840
A.1.5 MySQL 5.6 にはクエリーキャッシュはありますか。それはサーバー、インスタンス、またはデータベースで動作しますか。 2840
A.1.6 MySQL 5.6 にはシーケンスはありますか。 2840
A.1.7 MySQL 5.6 には秒の小数部が返される NOW() 関数はありますか。 2840
A.1.8 MySQL 5.6 はマルチコアプロセッサで動作しますか。 2840
A.1.9 mysqld の複数のプロセスが表示されるのはなぜですか。 2840
A.1.10 MySQL 5.6 は ACID トランザクションを実行できますか。 2840

A.1.1. 本番で使用 (GA) できる MySQL はどのバージョンですか。

MySQL 5.6、MySQL 5.5、MySQL 5.1、および MySQL 5.0 は、本番での使用がサポートされます。

MySQL 5.6 は MySQL 5.6.10 で一般提供 (GA) ステータスとなり、2013 年 2 月 5 日に本番使用のためにリリースされました。

MySQL 5.5 は MySQL 5.5.8 で一般提供 (GA) ステータスとなり、2010 年 12 月 3 日に本番使用のためにリリースされました。

MySQL 5.1 は MySQL 5.1.30 で一般提供 (GA) ステータスとなり、2008 年 11 月 14 日に本番使用のためにリリースされました。

MySQL 5.0 は MySQL 5.0.15 で一般提供 (GA) ステータスとなり、2005 年 10 月 19 日に本番使用のためにリリースされました。MySQL 5.0 のアクティブな開発は終了しました。

A.1.2. 開発 (GA ではない) バージョンの状況を教えてください。

MySQL は、本番前品質機能を導入してリリース品質に安定化させるマイルストーンリリースモデルに従っています (http://dev.mysql.com/doc/mysql-development-cycle/en/index.html を参照してください)。このプロセスは繰り返され、各リリースが本番前ステータスおよびリリース品質ステータスの間を循環します。特定のリリースのステータスを確認するには、変更ログをチェックしてください。

MySQL 5.4 は開発シリーズでした。このシリーズへの作業は終了しました。

MySQL 5.7 は、前述のマイルストーンリリース方式を使用して、アクティブに開発されています。

MySQL 6.0 は開発シリーズでした。このシリーズへの作業は終了しました。

A.1.3. MySQL 5.6 ではサブクエリーを実行できますか。

はい。セクション13.2.10「サブクエリー構文」を参照してください。

A.1.4. MySQL 5.6 では複数テーブルへの挿入、更新、および削除を実行できますか。

はい。複数テーブルへの更新の実行に必要な構文については、セクション13.2.11「UPDATE 構文」を参照してください。複数テーブルでの削除の実行に必要な構文については、セクション13.2.2「DELETE 構文」を参照してください。

複数テーブルへの挿入は、FOR EACH ROW 句の BEGIN ... END ブロック内に複数の INSERT ステートメントが含まれているトリガーを使用して実行できます。セクション20.3「トリガーの使用」を参照してください。

A.1.5. MySQL 5.6 にはクエリーキャッシュはありますか。それはサーバー、インスタンス、またはデータベースで動作しますか。

はい。クエリーキャッシュはサーバーレベルで動作し、元のクエリー文字列と一致する完全な結果セットがキャッシュされます。まったく同じクエリーが発行された (これは特に Web アプリケーションでよく行われます) 場合、解析または実行を行う必要はありません。結果がキャッシュから直接送信されます。さまざまなチューニングオプションを使用できます。セクション8.9.3「MySQL クエリーキャッシュ」を参照してください。

A.1.6. MySQL 5.6 にはシーケンスはありますか。

いいえ。ただし、MySQL には AUTO_INCREMENT システムがあり、MySQL 5.6 のマルチマスターレプリケーションセットアップで挿入を処理することもできます。auto_increment_increment および auto_increment_offset システム変数を使用して、各サーバーがほかのサーバーと競合しない自動インクリメント値を生成するように設定できます。auto_increment_increment にはサーバーの数より大きい値を指定し、各サーバーが一意のオフセットを持つようにします。

A.1.7. MySQL 5.6 には秒の小数部が返される NOW() 関数はありますか。

いいえ。これは MySQL のロードマップに「ローリング機能」として挙がっています。これは、フラッグシップ機能ではありませんが、開発時間が許すかぎり今後実装されることを意味します。特定の顧客の要求によって、このスケジュールが変更されることがあります。

ただし、MySQL では小数部がある時間文字列が解析されます。セクション11.3.2「TIME 型」を参照してください。

A.1.8. MySQL 5.6 はマルチコアプロセッサで動作しますか。

はい。MySQL は完全にマルチスレッド化されており、オペレーティングシステムでそれらがサポートされている場合、複数の CPU が使用されます。

A.1.9. mysqld の複数のプロセスが表示されるのはなぜですか。

LinuxThreads を使用している場合は、少なくとも 3 つの mysqld プロセスが実行されていることが表示されます。これらは実際にはスレッドです。LinuxThreads マネージャーのための 1 つのスレッド、接続を処理するための 1 つのスレッド、およびアラームおよびシグナルを処理するための 1 つのスレッドがあります。

A.1.10. MySQL 5.6 は ACID トランザクションを実行できますか。

はい。MySQL の現在のすべてのバージョンでトランザクションがサポートされます。InnoDB ストレージエンジンは、行レベルのロック、マルチバージョン、非ロックの反復可能読み取り、および 4 つのすべての SQL 標準分離レベルを持つ、完全な ACID トランザクションを提供しています。

NDB ストレージエンジンは、READ COMMITTED トランザクション分離レベルのみをサポートしています。

A.2 MySQL 5.6 FAQ: ストレージエンジン

A.2.1 MySQL のストレージエンジンの完全なドキュメントはどこで入手できますか。 2840
A.2.2 MySQL 5.6 に新しいストレージエンジンはありますか。 2841
A.2.3 MySQL 5.6 で削除されたストレージエンジンはありますか。 2841
A.2.4 ARCHIVE ストレージエンジンに固有の利点は何ですか。 2841

A.2.1. MySQL のストレージエンジンの完全なドキュメントはどこで入手できますか。

第15章「代替ストレージエンジン」を参照してください。この章には、MySQL Cluster に使用される NDB ストレージエンジンを除く、MySQL のすべてのストレージエンジンについての情報が含まれていま

す。NDB については、第18章「MySQL Cluster NDB 7.3 および MySQL Cluster NDB 7.4」で説明しています。

A.2.2. MySQL 5.6 に新しいストレージエンジンはありますか。

MySQL 5.1 ではオプションの InnoDB Plugin の機能が組み込みの InnoDB ストレージエンジンに取り込まれたため、Barracuda ファイル形式、InnoDB テーブルの圧縮、パフォーマンスのための新しい構成オプションなどの機能を活用できます。詳細は、第14章「InnoDB ストレージエンジン」を参照してください。また、InnoDB は新しいテーブルのデフォルトのストレージエンジンになりました。詳細は、セクション14.1.1「デフォルトの MySQL ストレージエンジンとしての InnoDB」を参照してください。

A.2.3. MySQL 5.6 で削除されたストレージエンジンはありますか。

いいえ。

A.2.4. ARCHIVE ストレージエンジンに固有の利点は何ですか。

ARCHIVE ストレージエンジンは、インデックスのない大量のデータを格納するのにもっとも適しています。フットプリントが非常に小さく、テーブルスキャンを使用して選択が実行されます。詳細は、セクション15.5「ARCHIVE ストレージエンジン」を参照してください。

A.3 MySQL 5.6 FAQ: サーバー SQL モード

A.3.1 サーバー SQL モードとは何ですか。 2841
A.3.2 サーバー SQL モードはいくつありますか。 2841
A.3.3 サーバー SQL モードを判別するにはどうすればよいですか。 2841
A.3.4 モードはデータベースまたは接続に依存していますか。 2841
A.3.5 厳密モードにルールを追加できますか。 2841
A.3.6 厳密モードはパフォーマンスに影響しますか。 2841
A.3.7 MySQL 5.6 をインストールしたときのデフォルトのサーバー SQL モードは何ですか。 2842

A.3.1. サーバー SQL モードとは何ですか。

サーバー SQL モードは、MySQL でサポートされる SQL 構文、および実行されるデータ妥当性チェックの種類を定義します。これにより、MySQL をさまざまな環境で使用したり、MySQL をほかのデータベースサーバーと一緒に使用したりすることが、さらに容易になります。MySQL サーバーは、これらのモードを各クライアントに個別に適用します。詳細は、セクション5.1.7「サーバー SQL モード」を参照してください。

A.3.2. サーバー SQL モードはいくつありますか。

各モードは、個別にオン/オフを切り替えることができます。使用可能なモードの完全なリストについては、セクション5.1.7「サーバー SQL モード」を参照してください。

A.3.3. サーバー SQL モードを判別するにはどうすればよいですか。

--sql-mode オプションを使用すると、デフォルトの SQL モード (mysqld を起動する場合) を設定できます。SET [GLOBAL|SESSION] sql_mode='modes' ステートメントを使用すると、ローカルに接続に対して、またはグローバルに適用されるように、接続内から設定を変更できます。現在のモードを取得するには、SELECT @@sql_mode ステートメントを発行します。

A.3.4. モードはデータベースまたは接続に依存していますか。

モードは特定のデータベースにリンクされていません。モードは、ローカルにセッション (接続) に対して設定するか、グローバルにサーバーに対して設定できます。これらの設定は、SET [GLOBAL|SESSION] sql_mode='modes' を使用して変更できます。

A.3.5. 厳密モードにルールを追加できますか。

厳密モードと呼ぶ場合は、TRADITIONAL、STRICT_TRANS_TABLES、または STRICT_ALL_TABLES モードの少なくとも 1 つが有効にされているモードを意味します。オプションは組み合わせることができるため、モードに制約を追加できます。詳細は、セクション5.1.7「サーバー SQL モード」を参照してください。

A.3.6. 厳密モードはパフォーマンスに影響しますか。

一部の設定での入力データの厳密な検証では、検証を行わない場合より時間がかかります。パフォーマンスへの影響はそれほど大きくありませんが、そのような検証が必要ない場合 (アプリケーションでそのすべてをすでに処理している場合)、MySQL には厳密モードを無効にするオプションがあります。ただし、必要な場合は、厳密モードでそのような検証を行うことができます。

A.3.7. MySQL 5.6 をインストールしたときのデフォルトのサーバー SQL モードは何ですか。

MySQL 5.6.6 の時点では、デフォルトの SQL モードは NO_ENGINE_SUBSTITUTION です。5.6.6 より前は、デフォルトのモードは空でした (どのモードも有効にされません)。使用可能なすべてのモードおよび MySQL のデフォルトの動作については、セクション5.1.7「サーバー SQL モード」を参照してください。

A.4 MySQL 5.6 FAQ: ストアドプロシージャーおよびストアドファンクション

A.4.1 MySQL 5.6 はストアドプロシージャーおよびストアドファンクションをサポートしていますか。 2842
A.4.2 MySQL のストアドプロシージャーおよびストアドファンクションについてのドキュメントはどこにありますか。 2842
A.4.3 MySQL のストアドプロシージャーのディスカッションフォーラムはありますか。 2842
A.4.4 ストアドプロシージャーの ANSI SQL 2003 仕様はどこにありますか。 2842
A.4.5 ストアドルーチンを管理するにはどうすればよいですか。 2843
A.4.6 特定のデータベースのすべてのストアドプロシージャーおよびストアドファンクションを表示する方法はありますか。 2843
A.4.7 ストアドプロシージャーはどこに格納されますか。 2843
A.4.8 ストアドプロシージャーまたはストアドファンクションをパッケージにグループ化することはできますか。 2843
A.4.9 ストアドプロシージャーは別のストアドプロシージャーを呼び出すことができますか。 2843
A.4.10 ストアドプロシージャーはトリガーを呼び出すことができますか。 2843
A.4.11 ストアドプロシージャーはテーブルにアクセスできますか。 2843
A.4.12 ストアドプロシージャーには、アプリケーションエラーを発生させるステートメントはありますか。 2843
A.4.13 ストアドプロシージャーには例外処理はありますか。 2843
A.4.14 MySQL 5.6 のストアドルーチンは結果セットを返すことができますか。 2843
A.4.15 ストアドプロシージャーで WITH RECOMPILE はサポートされますか。 2844
A.4.16 mod_plsql を Apache のゲートウェイとして使用してデータベース内のストアドプロシージャーと直接やり取りするのと同等の機能は MySQL にありますか。 2844
A.4.17 ストアドプロシージャーに入力として配列を渡すことはできますか。 2844
A.4.18 ストアドプロシージャーに IN パラメータとしてカーソルを渡すことはできますか。 2844
A.4.19 ストアドプロシージャーの OUT パラメータとしてカーソルを返すことはできますか。 2844
A.4.20 デバッグのために、ストアドルーチン内の変数の値を出力できますか。 2844
A.4.21 ストアドプロシージャー内でトランザクションをコミットまたはロールバックできますか。 2844
A.4.22 MySQL 5.6 のストアドプロシージャーおよびストアドファンクションはレプリケーションで動作しますか。 2844
A.4.23 マスターサーバーで作成されたストアドプロシージャーおよびストアドファンクションはスレーブにレプリケートされますか。 2844
A.4.24 ストアドプロシージャーおよびストアドファンクション内で実行されたアクションはどのようにレプリケートされますか。 2844
A.4.25 レプリケーションでストアドプロシージャーおよびストアドファンクションを使用するための特別なセキュリティー要件はありますか。 2844
A.4.26 ストアドプロシージャーおよびストアドファンクションのアクションをレプリケートする場合の制限は何ですか。 2845
A.4.27 前述の制限は MySQL のポイントインタイムリカバリを行う機能に影響しますか。 2845
A.4.28 前述の制限を修正するために何が行われていますか。 2845

A.4.1. MySQL 5.6 はストアドプロシージャーおよびストアドファンクションをサポートしていますか。

はい。MySQL 5.6 は 2 種類のストアドルーチン (ストアドプロシージャーおよびストアドファンクション) をサポートしています。

A.4.2. MySQL のストアドプロシージャーおよびストアドファンクションについてのドキュメントはどこにありますか。

セクション20.2「ストアドルーチン (プロシージャーと関数) の使用」を参照してください。

A.4.3. MySQL のストアドプロシージャーのディスカッションフォーラムはありますか。

はい。https://forums.mysql.com/list.php?98 を参照してください。

A.4.4. ストアドプロシージャーの ANSI SQL 2003 仕様はどこにありますか。

残念ながら、正式な仕様は無料では入手できません (ANSI は有料で販売しています)。ただし、標準の包括的な概要を説明した (ストアドプロシージャーの説明を含む)、Peter Gulutzan および Trudy Pelzer 著の『SQL-99 Complete, Really』などの本があります。

A.4.5. ストアドルーチンを管理するにはどうすればよいですか。

ストアドルーチンに明快な命名スキームを使用することはよい管理方法です。ストアドプロシージャーは、CREATE [FUNCTION|PROCEDURE]、ALTER [FUNCTION|PROCEDURE]、DROP [FUNCTION|PROCEDURE]、および SHOW CREATE [FUNCTION|PROCEDURE] を使用して管理できます。既存のストアドプロシージャーに関する情報を取得するには、INFORMATION_SCHEMA データベースの ROUTINES テーブル (セクション21.18「INFORMATION_SCHEMA ROUTINES テーブル」を参照してください) を使用します。

A.4.6. 特定のデータベースのすべてのストアドプロシージャーおよびストアドファンクションを表示する方法はありますか。

はい。dbname という名前のデータベースの場合、INFORMATION_SCHEMA.ROUTINES テーブルに対して次のクエリーを使用します。

```
SELECT ROUTINE_TYPE, ROUTINE_NAME
  FROM INFORMATION_SCHEMA.ROUTINES
  WHERE ROUTINE_SCHEMA='dbname';
```

詳細は、セクション21.18「INFORMATION_SCHEMA ROUTINES テーブル」を参照してください。

ストアドルーチンの本体は、SHOW CREATE FUNCTION (ストアドファンクションの場合) または SHOW CREATE PROCEDURE (ストアドプロシージャーの場合) を使用して表示できます。詳細は、セクション13.7.5.11「SHOW CREATE PROCEDURE 構文」を参照してください。

A.4.7. ストアドプロシージャーはどこに格納されますか。

mysql システムデータベースの proc テーブルに格納されます。ただし、システムデータベースのテーブルに直接アクセスしないでください。代わりに、ストアドファンクションに関する情報を取得するには SHOW CREATE FUNCTION、およびストアドプロシージャーに関する情報を取得するには SHOW CREATE PROCEDURE を使用します。これらのステートメントについては、セクション13.7.5.11「SHOW CREATE PROCEDURE 構文」を参照してください。

INFORMATION_SCHEMA データベースの ROUTINES テーブルにクエリーすることもできます。このテーブルの情報については、セクション21.18「INFORMATION_SCHEMA ROUTINES テーブル」を参照してください。

A.4.8. ストアドプロシージャーまたはストアドファンクションをパッケージにグループ化することはできますか。

いいえ。これは MySQL 5.6 ではサポートされません。

A.4.9. ストアドプロシージャーは別のストアドプロシージャーを呼び出すことができますか。

はい。

A.4.10. ストアドプロシージャーはトリガーを呼び出すことができますか。

ストアドプロシージャーでは、トリガーが実行される UPDATE などの SQL ステートメントを実行できます。

A.4.11. ストアドプロシージャーはテーブルにアクセスできますか。

はい。ストアドプロシージャーは、必要に応じて 1 つ以上のテーブルにアクセスできます。

A.4.12. ストアドプロシージャーには、アプリケーションエラーを発生させるステートメントはありますか。

はい。MySQL 5.6 には、SQL 標準の SIGNAL ステートメントおよび RESIGNAL ステートメントが実装されています。セクション13.6.7「条件の処理」を参照してください。

A.4.13. ストアドプロシージャーには例外処理はありますか。

MySQL には、SQL 標準に従った HANDLER 定義が実装されています。詳細は、セクション13.6.7.2「DECLARE ... HANDLER 構文」を参照してください。

A.4.14. MySQL 5.6 のストアドルーチンは結果セットを返すことができますか。

ストアドプロシージャーは返すことができますが、ストアドファンクションは返すことができません。ストアドプロシージャー内で通常の SELECT を実行すると、結果セットがクライアントに直接返されます。これが動作するようにするには、MySQL 4.1 (以降) のクライアント/サーバープロトコルを使用する必要が

あります。これは、たとえば PHP では、古い mysql 拡張ではなく mysqli 拡張を使用する必要があることを意味します。

A.4.15.ストアドプロシージャーで WITH RECOMPILE はサポートされますか。

MySQL 5.6 にはありません。

A.4.16.mod_plsql を Apache のゲートウェイとして使用してデータベース内のストアドプロシージャーと直接やり取りするのと同等の機能は MySQL にありますか。

MySQL 5.6 には同等の機能はありません。

A.4.17.ストアドプロシージャーに入力として配列を渡すことはできますか。

MySQL 5.6 にはありません。

A.4.18.ストアドプロシージャーに IN パラメータとしてカーソルを渡すことはできますか。

MySQL 5.6 では、カーソルはストアドプロシージャーの内部でのみ使用できます。

A.4.19.ストアドプロシージャーの OUT パラメータとしてカーソルを返すことはできますか。

MySQL 5.6 では、カーソルはストアドプロシージャーの内部でのみ使用できます。ただし、SELECT でカーソルを開かない場合、結果はクライアントに直接送信されます。変数に対して SELECT INTO を発行することもできます。セクション13.2.9「SELECT 構文」を参照してください。

A.4.20.デバッグのために、ストアドルーチン内の変数の値を出力できますか。

はい。これは、ストアドプロシージャーでは行うことができますが、ストアドファンクションでは行うことはできません。ストアドプロシージャー内で通常の SELECT を実行すると、結果セットがクライアントに直接返されます。これが動作するようにするには、MySQL 4.1 (以降) のクライアント/サーバープロトコルを使用する必要があります。これは、たとえば PHP では、古い mysql 拡張ではなく mysqli 拡張を使用する必要があることを意味します。

A.4.21.ストアドプロシージャー内でトランザクションをコミットまたはロールバックできますか。

はい。ただし、ストアドファンクション内でトランザクション操作を実行することはできません。

A.4.22.MySQL 5.6 のストアドプロシージャーおよびストアドファンクションはレプリケーションで動作しますか。

はい。ストアドプロシージャーおよびストアドファンクションで実行された通常のアクションは、マスター MySQL サーバーからスレーブサーバーにレプリケートされます。セクション20.7「ストアドプログラムのバイナリロギング」で詳しく説明されているいくつかの制限があります。

A.4.23.マスターサーバーで作成されたストアドプロシージャーおよびストアドファンクションはスレーブにレプリケートされますか。

はい。マスターサーバーで通常の DDL ステートメントを使用して実行されたストアドプロシージャーおよびストアドファンクションの作成はスレーブにレプリケートされるため、オブジェクトは両方のサーバーに存在します。ストアドプロシージャーおよびストアドファンクションに対する ALTER ステートメントおよび DROP ステートメントもレプリケートされます。

A.4.24.ストアドプロシージャーおよびストアドファンクション内で実行されたアクションはどのようにレプリケートされますか。

MySQL は、ストアドプロシージャーで行われた各 DML イベントを記録し、それらの個々のアクションをスレーブサーバーにレプリケートします。ストアドプロシージャーを実行するために行われた実際の呼び出しはレプリケートされません。

データを変更するストアドファンクションは、各ファンクション内で行われた DML イベントとしてではなく、関数呼び出しとして口グ記録されます。

A.4.25.レプリケーションでストアドプロシージャーおよびストアドファンクションを使用するための特別なセキュリティー要件はありますか。

はい。スレーブサーバーにはマスターのバイナリログから読み取ったステートメントを実行する権限があるため、レプリケーションでストアドファンクションを使用するための特殊なセキュリティー制約が存在します。レプリケーションまたは一般のバイナリロギング (ポイントインタイムリカバリのための) がアクティブである場合、MySQL の DBA には選択できるセキュリティーオプションが 2 つあります。

1. ストアドファンクションを作成するユーザーに、SUPER 権限を付与する必要があります。
2. または、DBA は log_bin_trust_function_creators システム変数に 1 を設定できます。これにより、標準の CREATE ROUTINE 権限を持ったユーザーがストアドファンクションを作成できます。

A.4.26.ストアドプロシージャーおよびストアドファンクションのアクションをレプリケートする場合の制限は何ですか。

ストアドプロシージャーに埋め込まれている決定性のない (ランダムな) アクションまたは時間ベースのアクションは、正しくレプリケートされないことがあります。その特性により、ランダムに生成された結果は予測できず、正確に再現できません。このため、スレーブにレプリケートされたランダムアクションは、マスターで実行されたアクションとは異なります。ストアドファンクションを DETERMINISTIC として宣言した場合、または log_bin_trust_function_creators システム変数に 0 を設定した場合は、ランダムに値が設定される操作を呼び出すことはできません。

また、時間ベースのアクションはスレーブで再現できません。ストアドプロシージャーのそのようなアクションのタイミングは、レプリケーションに使用されるバイナリログを介して再現できないためです。バイナリログには、DML イベントのみが記録され、タイミング制約は含まれていません。

最後に、非トランザクションテーブルで大きい DML アクション (一括挿入など) 中にエラーが発生した場合、マスターは DML アクティビティーによって部分的に更新されたがスレーブが更新されず、レプリケーションの問題が発生することがあります。回避策は、関数の DML アクションを IGNORE キーワードを指定して実行することであり、マスターでエラーが発生した更新が無視され、エラーが発生しなかった更新がスレーブにレプリケートされるようにします。

A.4.27.前述の制限は MySQL のポイントインタイムリカバリを行う機能に影響しますか。

レプリケーションに影響する制限が、ポイントインタイムリカバリに同様に影響します。

A.4.28.前述の制限を修正するために何が行われていますか。

ステートメントベースのレプリケーションまたは行ベースのレプリケーションのいずれかを選択できます。元のレプリケーションの実装は、ステートメントベースのバイナリロギングに基づいています。行ベースのバイナリロギングによって、前述の制限が解決されます。

複合レプリケーションも使用できます (--binlog-format=mixed を指定してサーバーを起動します)。このハイブリッドの「賢い」形式のレプリケーションは、ステートメントレベルのレプリケーションを安全に使用できるかどうか、または行レベルのレプリケーションが必要であるかを「判断」します。

追加情報については、セクション17.1.2「レプリケーション形式」を参照してください。

A.5 MySQL 5.6 FAQ: トリガー

A.5.1 MySQL 5.6 のトリガーについてのドキュメントはどこにありますか。 2845
A.5.2 MySQL のトリガーについてのディスカッションフォーラムはありますか。 2845
A.5.3 MySQL 5.6 にはステートメントレベルまたは行レベルのトリガーはありますか。 2846
A.5.4 デフォルトのトリガーはありますか。 2846
A.5.5 MySQL でトリガーを管理するにはどうすればよいですか。 2846
A.5.6 特定のデータベースのすべてのトリガーを表示する方法はありますか。 2846
A.5.7 トリガーはどこに格納されますか。 2846
A.5.8 トリガーはストアドプロシージャーを呼び出すことができますか。 2846
A.5.9 トリガーはテーブルにアクセスできますか。 2846
A.5.10 1 つのテーブルに同じトリガーイベントおよびアクション時間のトリガーを複数定義することはできますか。 2846
A.5.11 トリガーでは UDF を使用して外部アプリケーションを呼び出すことができますか。 2846
A.5.12 トリガーはリモートサーバー上のテーブルを更新できますか。 2846
A.5.13 トリガーはレプリケーションで動作しますか。 2846
A.5.14 マスターでトリガーによって実行されたアクションはどのようにスレーブにレプリケートされますか。 2847

A.5.1. MySQL 5.6 のトリガーについてのドキュメントはどこにありますか。

セクション20.3「トリガーの使用」を参照してください。

A.5.2. MySQL のトリガーについてのディスカッションフォーラムはありますか。

はい。https://forums.mysql.com/list.php?99 にあります。

A.5.3. MySQL 5.6 にはステートメントレベルまたは行レベルのトリガーはありますか。

MySQL 5.6 では、すべてのトリガーは FOR EACH ROW です。つまり、挿入、更新、または削除される各行に対してトリガーが実行されます。MySQL 5.6 は FOR EACH STATEMENT を使用したトリガーをサポートしていません。

A.5.4. デフォルトのトリガーはありますか。

明示的にはありません。MySQL は、一部の TIMESTAMP カラム、および AUTO_INCREMENT を使用して定義されたカラムに特殊な動作を割り当てています。

A.5.5. MySQL でトリガーを管理するにはどうすればよいですか。

MySQL 5.6 では、トリガーを作成する場合は CREATE TRIGGER ステートメントを使用し、削除する場合は DROP TRIGGER を使用します。これらのステートメントについては、セクション13.1.19「CREATE TRIGGER 構文」およびセクション13.1.30「DROP TRIGGER 構文」を参照してください。

トリガーに関する情報は、INFORMATION_SCHEMA.TRIGGERS テーブルをクエリーすることによって取得できます。セクション21.26「INFORMATION_SCHEMA TRIGGERS テーブル」を参照してください。

A.5.6. 特定のデータベースのすべてのトリガーを表示する方法はありますか。

はい。データベース dbname に定義されているすべてのトリガーのリストを取得するには、INFORMATION_SCHEMA.TRIGGERS テーブルに対して次のようなクエリーを使用します。

```
SELECT TRIGGER_NAME, EVENT_MANIPULATION, EVENT_OBJECT_TABLE, ACTION_STATEMENT
  FROM INFORMATION_SCHEMA.TRIGGERS
  WHERE TRIGGER_SCHEMA='dbname';
```

このテーブルについては、セクション21.26「INFORMATION_SCHEMA TRIGGERS テーブル」を参照してください。

MySQL に固有の SHOW TRIGGERS ステートメントを使用することもできます。セクション13.7.5.39「SHOW TRIGGERS 構文」を参照してください。

A.5.7. トリガーはどこに格納されますか。

テーブルのトリガーは、現在、.TRG ファイルに格納され、テーブルごとにそのようなファイルが 1 つあります。

A.5.8. トリガーはストアドプロシージャーを呼び出すことができますか。

はい。

A.5.9. トリガーはテーブルにアクセスできますか。

トリガーは、それ自体が定義されているテーブルの古いデータおよび新しいデータの両方にアクセスできます。トリガーはほかのテーブルに影響を与えることもできますが、その関数またはトリガーを呼び出したステートメントによってすでに使用されている (読み取りまたは書き込みのために) テーブルを変更することはできません。

A.5.10.1 つのテーブルに同じトリガーイベントおよびアクション時間のトリガーを複数定義することはできますか。

MySQL 5.6 では、同じトリガーイベントおよびアクション時間を持つ複数のトリガーを特定のテーブルに定義することはできません。たとえば、1 つのテーブルに対して 2 つの BEFORE UPDATE トリガーを定義することはできません。この制限は MySQL 5.7 で解除されました。

A.5.11.トリガーでは UDF を使用して外部アプリケーションを呼び出すことができますか。

はい。たとえば、トリガーは sys_exec() という UDF を呼び出すことができます。

A.5.12.トリガーはリモートサーバー上のテーブルを更新できますか。

はい。リモートサーバー上のテーブルは、FEDERATED ストレージエンジンを使用して更新できます。(セクション15.8「FEDERATED ストレージエンジン」を参照してください)。

A.5.13.トリガーはレプリケーションで動作しますか。

はい。ただし、それらが動作する仕組みは、MySQL のすべてのバージョンで使用できる「標準」のステートメントベースのレプリケーション、または MySQL 5.1 で導入された行ベースのレプリケーション形式のいずれを使用しているかによって異なります。

ステートメントベースのレプリケーションを使用している場合、スレーブのトリガーは、マスターで実行されてスレーブにレプリケートされたステートメントによって実行されます。

行ベースのレプリケーションを使用している場合、スレーブのトリガーは、マスターで実行されてスレーブにレプリケートされたステートメントによって実行されません。代わりに、行ベースのレプリケーションを使用した場合は、マスターでトリガーが実行されたことによる変更がスレーブに適用されます。

詳細は、セクション17.4.1.32「レプリケーションとトリガー」を参照してください。

A.5.14.マスターでトリガーによって実行されたアクションはどのようにスレーブにレプリケートされますか。

これは、ステートメントベースのレプリケーションまたは行ベースのレプリケーションのいずれを使用しているかによって異なります。

ステートメントベースのレプリケーション　まず、マスターに存在するトリガーをスレーブサーバーに再作成する必要があります。これが行われると、レプリケーションに関与するほかの標準の DML ステートメントと同様に、レプリケーションフローが動作します。たとえば、マスター MySQL サーバーに存在する、AFTER 挿入トリガーを持つテーブル EMP について考えてみましょう。同じ EMP テーブルおよび AFTER 挿入トリガーが、スレーブサーバーにも存在します。レプリケーションフローは次のようになります。

1. INSERT ステートメントが EMP に発行されます。
2. EMP の AFTER トリガーが実行されます。
3. INSERT ステートメントがバイナリログに書き込まれます。
4. レプリケーションのスレーブは、EMP に対するその INSERT ステートメントを取得してそれを実行します。
5. スレーブに存在する EMP の AFTER トリガーが実行されます。

行ベースのレプリケーション　行ベースのレプリケーションを使用している場合は、マスターでトリガーが実行されたことによる変更がスレーブに適用されます。ただし、行ベースのレプリケーションでは、トリガー自体は実際にはスレーブで実行されません。これは、マスターとスレーブの両方にマスターの変更が適用され、さらにそれらの変更を行なったトリガーがスレーブで実行された場合、その変更は実際には 2 回スレーブに適用され、マスターとスレーブのデータが異なるデータとなってしまうためです。

ほとんどの場合、行ベースのレプリケーションおよびステートメントベースのレプリケーションで結果は同じです。ただし、マスターとスレーブで異なるトリガーを使用している場合、行ベースのレプリケーションは使用できません。(これは、行ベース形式は、トリガーが実行されるきっかけとなったステートメントではなく、マスターで実行されたトリガーによる変更をスレーブにレプリケートし、スレーブの対応するトリガーは実行されないためです。)代わりに、そのようなトリガーが実行されるきっかけとなったステートメントを、ステートメントベースのレプリケーションを使用してレプリケートする必要があります。

詳細は、セクション17.4.1.32「レプリケーションとトリガー」を参照してください。

A.6 MySQL 5.6 FAQ: ビュー

A.6.1 MySQL のビューについて説明しているドキュメントはどこにありますか。 2847
A.6.2 MySQL のビューについてのディスカッションフォーラムはありますか。 2847
A.6.3 基礎テーブルが削除または名前変更された場合、ビューはどうなりますか。 2847
A.6.4 MySQL 5.6 にはテーブルのスナップショットはありますか。 2848
A.6.5 MySQL 5.6 にはマテリアライズドビューはありますか。 2848
A.6.6 結合に基づいているビューに挿入できますか。 2848

A.6.1. MySQL のビューについて説明しているドキュメントはどこにありますか。

セクション20.5「ビューの使用」を参照してください。

A.6.2. MySQL のビューについてのディスカッションフォーラムはありますか。

はい。https://forums.mysql.com/list.php?100 を参照してください。

A.6.3. 基礎テーブルが削除または名前変更された場合、ビューはどうなりますか。

ビューが作成されたあとに、その定義が参照しているテーブルまたはビューを削除または変更できます。この種類の問題に関してビュー定義を確認するには、CHECK TABLE ステートメントを使用します。(セクション13.7.2.2「CHECK TABLE 構文」を参照してください。)

A.6.4. MySQL 5.6 にはテーブルのスナップショットはありますか。

いいえ。

A.6.5. MySQL 5.6 にはマテリアライズドビューはありますか。

いいえ。

A.6.6. 結合に基づいているビューに挿入できますか。

INSERT ステートメントに、関連するテーブルが 1 つのみであることを明確にするカラムリストがある場合は、実行できます。

ビューに対する単一の挿入で、複数のテーブルに挿入することはできません。

A.7 MySQL 5.6 FAQ: INFORMATION_SCHEMA

A.7.1 MySQL の INFORMATION_SCHEMA データベースについてのドキュメントはどこにありますか。 2848
A.7.2 INFORMATION_SCHEMA についてのディスカッションフォーラムはありますか。 2848
A.7.3 INFORMATION_SCHEMA の ANSI SQL 2003 仕様はどこにありますか。 2848
A.7.4 Oracle のデータディクショナリと MySQL の INFORMATION_SCHEMA の違いは何ですか。 2848
A.7.5 INFORMATION_SCHEMA データベースのテーブルに挿入または変更を行うことはできますか。 2848

A.7.1. MySQL の INFORMATION_SCHEMA データベースについてのドキュメントはどこにありますか。

第21章「INFORMATION_SCHEMA テーブル」を参照してください。

A.7.2. INFORMATION_SCHEMA についてのディスカッションフォーラムはありますか。

https://forums.mysql.com/list.php?101 を参照してください。

A.7.3. INFORMATION_SCHEMA の ANSI SQL 2003 仕様はどこにありますか。

残念ながら、正式な仕様は無料では入手できません。(ANSI は有料で販売しています。)ただし、標準の包括的な概要を説明した (INFORMATION_SCHEMA を含む)、Peter Gulutzan および Trudy Pelzer 著の『SQL-99 Complete, Really』などの本があります。

A.7.4. Oracle のデータディクショナリと MySQL の INFORMATION_SCHEMA の違いは何ですか。

Oracle および MySQL は両方とも、テーブルにメタデータを提供しています。ただし、Oracle と MySQL では異なるテーブル名およびカラム名が使用されています。MySQL の実装は、SQL 標準に定義されている INFORMATION_SCHEMA をサポートしている DB2 および SQL Server により似ています。

A.7.5. INFORMATION_SCHEMA データベースのテーブルに挿入または変更を行うことはできますか。

いいえ。アプリケーションが特定の標準構造に依存していることがあるため、変更しないでください。このため、INFORMATION_SCHEMA テーブルまたはデータを変更したことによるバグまたはその他の問題はサポートできません。

A.8 MySQL 5.6 FAQ: 移行

A.8.1 MySQL 5.5 から MySQL 5.6 に移行する方法に関する情報はどこにありますか。 2848
A.8.2 MySQL 5.6 のストレージエンジン (テーブルタイプ) のサポートは、以前のバージョンからどのように変更されましたか。 2848

A.8.1. MySQL 5.5 から MySQL 5.6 に移行する方法に関する情報はどこにありますか。

アップグレード情報については、セクション2.11.1「MySQL のアップグレード」を参照してください。アップグレード時はメジャーバージョンをスキップせずに、各手順でメジャーバージョンから次のメジャーバージョンにアップグレードして、手順の操作を完了してください。これはより複雑に見えますが、時間を節約してトラブルを避けることができます。アップグレード中に問題が発生した場合に、ユーザー自身または (MySQL Enterprise サブスクリプションがある場合は) MySQL サポートがその原因を識別しやすくなります。

A.8.2. MySQL 5.6 のストレージエンジン (テーブルタイプ) のサポートは、以前のバージョンからどのように変更されましたか。

ストレージエンジンのサポートは次のように変更されました。

- ISAM テーブルのサポートは MySQL 5.0 で削除されたため、ISAM の代わりに MyISAM ストレージエンジンを使用してください。テーブル tblname を ISAM から MyISAM に変更するには、次のようなステートメントを発行します。

```
ALTER TABLE tblname ENGINE=MYISAM;
```

- MyISAM テーブルの内部 RAID も MySQL 5.0 で削除されました。これは、2G バイトを超えるファイルサイズをサポートしていないファイルシステムで大きいテーブルを許容するために以前使用されていました。現在のすべてのファイルシステムではより大きいテーブルが許容されます。MERGE テーブル、ビューなどの新しいほかのソリューションもあります。
- すべてのストレージエンジンの VARCHAR カラム型で、末尾のスペースが維持されるようになりました。
- MEMORY テーブル (以前は HEAP テーブルと呼ばれました) にも VARCHAR カラムを含めることができます。

A.9 MySQL 5.6 FAQ: セキュリティー

A.9.1 MySQL のセキュリティーの問題に関するドキュメントはどこにありますか。 2849
A.9.2 MySQL 5.6 には SSL に対するネイティブなサポートはありますか。 2849
A.9.3 SSL のサポートは MySQL バイナリに組み込まれていますか。それとも、バイナリを再コンパイルして有効にする必要がありますか。 2849
A.9.4 MySQL 5.6 には LDAP ディレクトリに対する組み込みの認証はありますか。 2850
A.9.5 MySQL 5.6 にはロールベースのアクセス制御 (RBAC) のサポートは含まれていますか。 2850

A.9.1. MySQL のセキュリティーの問題に関するドキュメントはどこにありますか。

最初に第6章「セキュリティー」をお読みください。

特定のセキュリティーの問題に関して役に立つことがある MySQL ドキュメントのほかの部分としては、次のセクションがあります。

- セクション6.1.1「セキュリティーガイドライン」。
- セクション6.1.3「攻撃者に対する MySQL のセキュアな状態の維持」。
- セクションB.5.4.1「root のパスワードをリセットする方法」。
- セクション6.1.5「MySQL を通常ユーザーとして実行する方法」。
- セクション24.3.2.6「ユーザー定義関数のセキュリティー上の予防措置」。
- セクション6.1.4「セキュリティー関連の mysqld オプションおよび変数」。
- セクション6.1.6「LOAD DATA LOCAL のセキュリティーの問題」。
- セクション2.10「インストール後のセットアップとテスト」。
- セクション6.3.10.1「基本的な SSL の概念」。

A.9.2. MySQL 5.6 には SSL に対するネイティブなサポートはありますか。

ほとんどの 5.6 バイナリには、クライアントおよびサーバー間の SSL 接続のサポートがあります。セクション6.3.10「セキュアな接続のための SSL の使用」を参照してください。

(たとえば) クライアントアプリケーションで SSL 接続がサポートされない場合は、SSH を使用して接続をトンネルすることもできます。例については、セクション6.3.11「SSH を使用した Windows から MySQL へのリモート接続」を参照してください。

A.9.3. SSL のサポートは MySQL バイナリに組み込まれていますか。それとも、バイナリを再コンパイルして有効にする必要がありますか。

ほとんどの 5.6 バイナリでは、セキュアなクライアントとサーバーの接続、認証が行われるクライアントとサーバーの接続、またはその両方に対して SSL が有効にされます。セクション6.3.10「セキュアな接続のための SSL の使用」を参照してください。

A.9.4. MySQL 5.6 には LDAP ディレクトリに対する組み込みの認証はありますか。

現在はありません。

A.9.5. MySQL 5.6 にはロールベースのアクセス制御 (RBAC) のサポートは含まれていますか。

現在はありません。

A.10 MySQL 5.6 FAQ: MySQL Cluster

次のセクションでは、MySQL Cluster および NDB ストレージエンジンに関するよくある質問に回答します。

A.10.1 クラスタがサポートされる MySQL ソフトウェアのバージョンはどれですか。ソースからコンパイルする必要がありますか。 2851
A.10.2 「NDB」 および 「NDBCLUSTER」 は何を意味していますか。 2851
A.10.3 MySQL Cluster を使用した場合と MySQL Replication を使用した場合の違いは何ですか。 2851
A.10.4 MySQL Cluster を実行するには、特殊なネットワーク環境が必要となりますか。クラスタ内のコンピュータはどのように通信しますか。 2851
A.10.5 MySQL Cluster を実行するために必要なコンピュータは何台ですか。その理由は何ですか。 2851
A.10.6 MySQL Cluster 内の各種のコンピュータにはどのような役割がありますか。 2852
A.10.7 MySQL Cluster の管理クライアントで SHOW コマンドを実行すると、次のような出力行が表示されます。 2852
A.10.8 MySQL Cluster を使用できるオペレーティングシステムはどれですか。 2853
A.10.9 MySQL Cluster を実行するためのハードウェア要件は何ですか。 2853
A.10.10 MySQL Cluster を使用するために必要な RAM のサイズはどれくらいですか。ディスクメモリーを使用することはできますか。 2853
A.10.11 MySQL Cluster に使用できるファイルシステムは何ですか。ネットワークファイルシステムまたはネットワーク共有は使用できますか。 2854
A.10.12 MySQL Cluster のノードは、仮想マシン (VMWare、Parallels、Xen によって作成された仮想マシンなど) 内で実行できますか。 2854
A.10.13 MySQL Cluster データベースにデータを移入しようとしています。ロード処理が予期せずに終了し、次のようなエラーメッセージが表示されます。 2854
A.10.14 MySQL Cluster は TCP/IP を使用しています。これは、1 つ以上のノードをリモートの場所に配置して、インターネット経由で実行できることを意味しますか。 2855
A.10.15 MySQL Cluster を使用するために、新しいプログラミング言語またはクエリー言語を学習する必要がありますか。 2855
A.10.16 MySQL Cluster によってサポートされるプログラミング言語および API は何ですか。 2855
A.10.17 MySQL Cluster には管理ツールが含まれていますか。 2856
A.10.18 MySQL Cluster を使用しているときに、エラーメッセージまたは警告メッセージの意味を確認するにはどうすればよいですか。 2856
A.10.19 MySQL Cluster はトランザクションセーフですか。どのような分離レベルがサポートされますか。 2856
A.10.20 MySQL Cluster によってサポートされるストレージエンジンは何ですか。 2856
A.10.21 壊滅的な障害が発生した場合 (たとえば、街全体が停電かつUPS の障害が発生した場合)、すべてのデータが失われますか。 2856
A.10.22 MySQL Cluster では FULLTEXT インデックスを使用できますか。 2856
A.10.23 単一のコンピュータ上で複数のノードを実行できますか。 2856
A.10.24 MySQL Cluster を再起動せずにデータノードを追加できますか。 2857
A.10.25 MySQL Cluster を使用するときに、注意するべき制限はありますか。 2857
A.10.26 MySQL Cluster では外部キーはサポートされますか。 2857
A.10.27 既存の MySQL データベースを MySQL Cluster にインポートするにはどうすればよいですか。 2857
A.10.28 MySQL Cluster のノードはどのようにして相互にやり取りしますか。 2858
A.10.29 アービトレータとは何ですか。 2858
A.10.30 MySQL Cluster によってサポートされるデータ型を何ですか。 2858
A.10.31 MySQL Cluster を起動および停止するにはどうすればよいですか。 2858
A.10.32 MySQL Cluster をシャットダウンすると、MySQL Cluster のデータはどうなりますか。 2859
A.10.33 MySQL Cluster に複数の管理ノードを使用することはよい考えですか。 2859
A.10.34 単一の MySQL Cluster に、異なる種類のハードウェアおよびオペレーティングシステムを混在させることはできますか。 2859
A.10.35 単一のホスト上で 2 つのデータノードを実行できますか。2 つの SQL ノードは実行できますか。 2859
A.10.36 MySQL Cluster ではホスト名を使用できますか。 2860
A.10.37 MySQL Cluster では IPv6 はサポートされますか。 2860
A.10.38 複数の MySQL サーバーを持つ MySQL Cluster で MySQL ユーザーを管理するにはどうすればよいですか。 2860
A.10.39 SQL ノードの 1 つで障害が発生した場合に、クエリーの送信を続けるにはどうすればよいですか。 2860
A.10.40 MySQL Cluster をバックアップおよびリストアするにはどうすればよいですか。 2860

A.10.41 「エンジェルプロセス」とは何ですか。 2860

A.10.1.クラスタがサポートされる MySQL ソフトウェアのバージョンはどれですか。ソースからコンパイルする必要がありますか。

MySQL Cluster は標準の MySQL Server 5.6 リリースではサポートされません。そうではなく、MySQL Cluster は個別の製品として提供されています。現在、次の MySQL Cluster リリースを本番で使用できます。

- MySQL Cluster NDB 7.3　このシリーズは、MySQL Cluster の最新の一般提供 (GA) バージョンであり、NDB ストレージエンジンのバージョン 7.3 および MySQL Server 5.6 に基づいています。新しい配備には、このシリーズの最新のリリースを使用してください。最新の MySQL Cluster NDB 7.3 リリースは https://dev.mysql.com/downloads/cluster/ から入手できます。
- MySQL Cluster NDB 7.4　これは、MySQL Cluster の現在の開発バージョンであり、NDB ストレージエンジンのバージョン 7.4 および MySQL Server 5.6 に基づいています。現在、MySQL Cluster NDB 7.4 はテストおよび評価のために使用できます。最新の MySQL Cluster NDB 7.4 リリースは https://dev.mysql.com/downloads/cluster/ から入手できます。

  MySQL Cluster NDB 7.4 での改善点の概要については、セクション18.1.4.2「MySQL Cluster NDB 7.4 での MySQL Cluster の開発」を参照してください。

使用している MySQL Server で NDB がサポートされるかどうかを判別するには、SHOW VARIABLES LIKE 'have_%'、SHOW ENGINES、または SHOW PLUGINS ステートメントのいずれかを使用します。

A.10.2.「NDB」 および 「NDBCLUSTER」 は何を意味していますか。

「NDB」 は 「Network Database」 を意味しています。NDB および NDBCLUSTER は、MySQL でクラスタがサポートされるストレージエンジンの名前です。開発者は NDB を好んでいますが、どちらの名前も正しい名前です。両方の名前がドキュメントに記述され、どちらの名前も MySQL Cluster テーブルを作成するための CREATE TABLE ステートメントの ENGINE オプションに使用できます。

A.10.3.MySQL Cluster を使用した場合と MySQL Replication を使用した場合の違いは何ですか。

従来の MySQL レプリケーションでは、マスター MySQL サーバーが 1 つ以上のスレーブを更新します。トランザクションは順次的にコミットされ、遅いトランザクションの場合、スレーブでの処理がマスターより遅れることがあります。これは、マスターで障害が発生した場合に、スレーブで最後のいくつかのトランザクションが記録されないことがあることを意味します。InnoDB などのトランザクションセーフなエンジンを使用している場合、トランザクションはスレーブで完了するかまったく適用されないかのいずれかであり、レプリケーションではマスターおよびスレーブのすべてのデータが常時整合性がとれていることは保証されません。MySQL Cluster では、すべてのデータノードの同期が維持され、あるデータノードでコミットされたトランザクションはすべてのデータノードでコミットされます。データノードの障害が発生した場合、ほかのすべてのデータノードでは整合性のある状態が維持されます。

簡単に言うと、標準の MySQL レプリケーションは非同期で実行され、MySQL Cluster は同期を維持して実行されます。

非同期レプリケーションは MySQL Cluster でも使用できます。MySQL Cluster Replication (「ジオレプリケーション」を呼ばれることもあります) には、2 つの MySQL Cluster 間のレプリケーション、および MySQL Cluster からクラスタ化されていない MySQL サーバーへのレプリケーションの両方を行う機能が含まれています。セクション18.6「MySQL Cluster レプリケーション」を参照してください。

A.10.4.MySQL Cluster を実行するには、特殊なネットワーク環境が必要となりますか。クラスタ内のコンピュータはどのように通信しますか。

MySQL Cluster は、TCP/IP を使用してコンピュータが接続されている高帯域幅の環境で使用することが意図されています。そのパフォーマンスは、クラスタ内のコンピュータ間の接続速度に直接関係しています。MySQL Cluster の最低限の接続要件には、一般的な 100 メガビット Ethernet ネットワークまたは同等のものが含まれます。可能な場合はギガビット Ethernet を使用することをお勧めします。

より速い SCI プロトコルもサポートされますが、特殊なハードウェアが必要となります。SCI については、セクション18.3.5「MySQL Cluster での高速インターコネクトの使用」を参照してください。

A.10.5.MySQL Cluster を実行するために必要なコンピュータは何台ですか。その理由は何ですか。

実行可能なクラスタの実行には、少なくとも 3 台のコンピュータが必要となります。ただし、MySQL Cluster の推奨される最低限のコンピュータの数は 4 台であり、管理ノードと SQL ノードを実行するためにそれぞれ 1 台ずつ、およびデータノードとして使用される 2 台のコンピュータです。データノードを 2 台にする目的は冗長性を持たせるためです。管理ノードは別個のマシンで実行して、いずれかのデー

タノードで障害が発生した場合にアービトレーションサービスが継続されることを保証する必要があります。

スループットおよび高可用性を向上させるには、複数の SQL ノード (クラスタに接続された MySQL サーバー) を使用してください。複数の管理サーバーを実行することもできます (必須ではありません)。

A.10.6 MySQL Cluster 内の各種のコンピュータにはどのような役割がありますか。

MySQL Cluster には、コンピュータを物理的要素とする、物理組織および論理組織の両方があります。クラスタの論理要素または機能要素はノードと呼ばれ、クラスタのノードを収容するコンピュータはクラスタホストと呼ばれることがあります。クラスタ内の特定のロールにそれぞれ対応する 3 つのタイプのノードがあります。これらを次に示します。

- 管理ノード　このノードはクラスタ全体の管理サービス (起動、シャットダウン、バックアップ、およびほかのノードの構成データを含む) を提供します。管理ノードサーバーはアプリケーション ndb_mgmd として実装されます。MySQL Cluster を制御するために使用される管理クライアントは ndb_mgm です。これらのプログラムについては、セクション18.4.4「ndb_mgmd — MySQL Cluster 管理サーバーデーモン」およびセクション18.4.5「ndb_mgm — MySQL Cluster 管理クライアント」を参照してください。

- データノード　このタイプのノードは、データを格納およびレプリケートします。データノードの機能は、NDB データノードプロセスndbd のインスタンスによって処理されます。詳細は、セクション 18.4.1「ndbd — MySQL Cluster データノードデーモン」を参照してください。

- SQL ノード　これは、単純に MySQL サーバーのインスタンス (mysqld) であり、NDBCLUSTER ストレージエンジンのサポートを組み込んでビルドされていて、--ndb-cluster オプションを指定して起動することによってこのエンジンを有効にし、--ndb-connectstring オプションを指定して MySQL Cluster の管理サーバーに接続できるようにします。これらのオプションについては、セクション 18.3.4.2「MySQL Cluster 用の MySQL Server オプション」を参照してください。

  > 注記
  >
  > API ノードは、データの格納および取得のためにクラスタのデータノードを直接使用するアプリケーションです。このため、SQL ノードは MySQL サーバーを使用してクラスタへの SQL インタフェースを提供する API ノードの一種であると考えることができます。直接的でオブジェクト指向のトランザクションおよび MySQL Cluster のデータへのスキャンインタフェースを提供するそのようなアプリケーション (MySQL サーバーに依存しない) は、NDB API を使用して記述できます。詳細は、NDB Cluster API Overview: The NDB APIを参照してください。

A.10.7 MySQL Cluster の管理クライアントで SHOW コマンドを実行すると、次のような出力行が表示されます。

```
id=2    @10.100.10.32  (Version: 5.6.22-ndb-7.3.9 Nodegroup: 0, *)
```

「*」は何を意味していますか。このノードはほかのノードとどのように異なりますか。

もっとも簡単な回答は、「ユーザーが制御できるものではなく、MySQL Cluster のソースコードを記述または分析するソフトウェアエンジニアではないかぎり、どのような場合でも憂慮する必要はないものです」となります。

この回答に満足できない場合、より長くテクニカルなバージョンは次のとおりです。

MySQL Cluster の多数のメカニズムでは、データノードの分散調整が必要となります。これらの分散アルゴリズムおよびプロトコルには、グローバルチェックポイント、DDL (スキーマ) の変更、およびノードの再起動処理が含まれます。この調整を単純にするために、それらのメンバーのいずれかがリーダーとして動作するようにデータノードで「選出」されます。(このノードは「マスター」と呼ばれていましたが、MySQL Replication のマスターサーバーと混乱しないように、この用語は削除されました。)この選択は完全に自動的であり、それに影響するメカニズムをユーザーが目にすることはありません。自動的であることが、MySQL Cluster の内部アーキテクチャーの重要な部分です。

ノードがこれらのメカニズムの「リーダー」として動作する場合、通常、それがアクティビティーの調整の中心となり、「フォロワー」として動作するほかのノードは、リーダーによって指示されたアクティビティーの各自の担当分を実行します。リーダーとして動作するノードで障害が発生すると、残りのノードによって新しいリーダーが選出されます。古いリーダーによって調整されていた進行中のタスクは、実際に関係するメカニズムに従って、失敗するか、新しいリーダーによって続行されます。

これらの各種のメカニズムおよびプロトコルの一部で別のリーダーノードが使用されることがありますが、一般的には、それらのすべてで同じリーダーが選択されます。管理クライアントの SHOW の出力でリーダーとして示されるノードは、DDL およびメタデータのアクティビティーの調整を担当する DICT マネージャーと内部では呼ばれています (詳細は、『MySQL Cluster API Developer Guide』のThe DBDICT Blockを参照してください)。

MySQL Cluster は、リーダーの選択がクラスタの外部で認識できるような影響を及ぼさないように設計されています。たとえば、現在のリーダーの CPU またはリソースの使用量がほかのデータノードより著しく高いことはなく、リーダーで障害が発生した場合にほかのデータノードで障害が発生した場合よりクラスタに対して特別に大きい影響があるということはありません。

A.10.8 MySQL Cluster を使用できるオペレーティングシステムはどれですか。

MySQL Cluster はほとんどの Unix 系のオペレーティングシステムでサポートされています。MySQL Cluster NDB 7.2 は、Microsoft Windows オペレーティングシステムの本番設定でもサポートされます。

さまざまなオペレーティングシステムのバージョン、オペレーティングシステムの配布、およびハードウェアプラットフォームで MySQL Cluster に提供されるサポートのレベルについては、https://www.mysql.com/support/supportedplatforms/cluster.htmlを参照してください。

A.10.9 MySQL Cluster を実行するためのハードウェア要件は何ですか。

MySQL Cluster は NDB 対応のバイナリを使用できるプラットフォームで実行してください。データノードおよび API ノードの場合、より速い CPU およびより多くのメモリーによって、パフォーマンスが向上することがあり、64 ビットの CPU は 32 ビットのプロセッサよりも効率的である場合があります。各ノードのデータベースの分担を保持するために、データノードに使用されるマシンに十分なメモリーがある必要があります (詳細は、「必要な RAM のサイズはどれくらいですか」を参照してください)。MySQL Cluster の管理サーバーを実行するためにのみ使用されるコンピュータの場合、要件は最低限のものです。通常、このタスクには一般的なデスクトップ PC (または同等のもの) で十分です。ノードは標準の TCP/IP ネットワークおよびハードウェアを使用して通信できます。高速の SCI プロトコルを使用することもできます。ただし、SCI を使用するには、特殊なネットワークハードウェアおよびソフトウェアが必要となります (セクション18.3.5「MySQL Cluster での高速インターコネクトの使用」を参照してください)。

A.10.10 MySQL Cluster を使用するために必要な RAM のサイズはどれくらいですか。ディスクメモリーを使用することはできますか。

以前は、MySQL Cluster はインメモリーのみを使用していました。MySQL 5.1 以降では、MySQL Cluster をディスクに格納する機能も提供されています。(この機能を以前のリリースにバックポートする計画はありません。)詳細は、セクション18.5.12「MySQL Cluster ディスクデータテーブル」を参照してください。

インメモリーの NDB テーブルの場合は、クラスタ内の各データノードで必要となる RAM の容量のおおよその見積もりを取得するために次の式を使用できます。

```
(SizeofDatabase × NumberOfReplicas × 1.1 ) / NumberOfDataNodes
```

メモリー要件をより正確に計算するには、クラスタデータベースの各テーブルで行ごとに必要となる格納領域 (詳細は、セクション11.7「データ型のストレージ要件」を参照してください) を判別して、それを行数で乗算する必要があります。カラムインデックスについては、次のことを考慮する必要もあります。

- NDBCLUSTER テーブルに作成される各主キーまたはハッシュインデックスには、レコードごとに 21–25 バイトが必要となります。これらのインデックスは IndexMemory を使用します。
- 順序付けされた各インデックスでは、DataMemory を使用して、レコードごとに 10 バイトのストレージが必要となります。
- また、主キーまたは一意のインデックスを作成すると、このインデックスが USING HASH を指定して作成された場合を除き、順序付けされたインデックスが作成されます。言い換えると、次のようになります。
  - 通常、クラスタテーブルの主キーまたは一意のインデックスには、レコードごとに 31 - 35 バイトが使用されます。
  - ただし、主キーまたは一意のインデックスが USING HASH を指定して作成されている場合は、レコードごとに 21 バイトから 25 バイトのみが必要となります。

すべての主キーおよび一意のインデックスに USING HASH を指定して MySQL Cluster のテーブルを作成すると、通常、テーブルの更新がより迅速に実行されます。主キーおよびユニークキーの作成に USING

HASH が使用されなかったテーブルへの更新より 20 - 30% 速いことがあります。これは、必要なメモリーが少なくなり (順序付けされたインデックスが作成されないため)、使用される CPU が少なくなる (読み取りおよび (場合によっては) 更新する必要があるインデックスが少なくなるため) ためです。ただし、これは、本来範囲スキャンを使用するクエリーをほかの方法で実行する必要があることも意味し、選択が低速になることがあります。

クラスタのメモリー要件を計算するときに、最新の MySQL 5.6 リリースに付属している ndb_size.pl ユーティリティーが役に立つことがあります。この Perl スクリプトは、現在の (クラスタではない) MySQL データベースに接続して、NDBCLUSTER ストレージエンジンを使用した場合にデータベースで必要となる容量に関する報告を作成します。詳細は、セクション18.4.25「ndb_size.pl — NDBCLUSTER サイズ要件エスティメータ」を参照してください。

すべての MySQL Cluster テーブルには主キーがある必要があることに留意することが非常に重要です。NDB ストレージエンジンは、主キーが定義されていない場合、主キーを自動的に作成します。この主キーは USING HASH を指定せずに作成されます。

ある時点で MySQL Cluster のデータおよびインデックスの格納に使用されているメモリー量を判別するには、ndb_mgm クライアントで REPORT MEMORYUSAGE コマンドを使用します。詳細は、セクション18.5.2「MySQL Cluster 管理クライアントのコマンド」を参照してください。また、使用可能な DataMemory または IndexMemory の 80% が使用されている場合、および使用率が 85%、90% などに達した場合に、警告がクラスタのログに書き込まれます。

A.10.11. MySQL Cluster に使用できるファイルシステムは何ですか。ネットワークファイルシステムまたはネットワーク共有は使用できますか。

通常、ホストのオペレーティングシステムにネイティブなファイルシステムは、MySQL Cluster で動作します。特定のファイルシステムが MySQL Cluster と特に良好に動作する (またはそれほどよくない) ことがわかった場合は、その知見をMySQL Cluster フォーラムで共有してください。

Windows の場合は、通常の MySQL と同様に、MySQL Cluster に NTFS ファイルシステムを使用することをお勧めします。FAT ファイルシステムまたは VFAT ファイルシステムでは、MySQL Cluster はテストされていません。このため、それらを MySQL または MySQL Cluster に使用することはお勧めしません。

MySQL Cluster はシェアードナッシングソリューションとして実装されています。この背景にある考え方は、1 つのハードウェアの障害によって、複数のクラスタノードの障害、または場合によってはクラスタ全体の障害が引き起こされないようにすることです。このため、ネットワーク共有またはネットワークファイルシステムの使用は、MySQL Cluster ではサポートされません。これは、SAN などの共有ストレージデバイスにも当てはまります。

A.10.12. MySQL Cluster のノードは、仮想マシン (VMWare、Parallels、Xen によって作成された仮想マシンなど) 内で実行できますか。

MySQL Cluster は MySQL Cluster NDB 7.2 以降で仮想マシンでの使用がサポートされます。現在、Oracle VM を使用してサポートおよびテストが行われています。

一部の MySQL Cluster ユーザーは、ほかの仮想化製品を使用して MySQL Cluster の配備に成功しました。そのような場合、オラクルは MySQL Cluster のサポートを提供できますが、仮想環境に固有の問題についてはその製品のベンダーに問い合わせる必要があります。

A.10.13. MySQL Cluster データベースにデータを移入しようとしています。ロード処理が予期せずに終了し、次のようなエラーメッセージが表示されます。

「ERROR 1114: The table 'my_cluster_table' is full」

これが発生するのはなぜですか。

原因はすべてのテーブルデータおよびすべてのインデックス (テーブル定義に主キーの定義が含まれていない場合に自動的に作成される、NDB ストレージエンジンによって要求される主キーを含む) のための十分な RAM が、使用しているセットアップで提供されていないことである可能性があります。

すべてのデータノードで同じ容量の RAM が使用されることにも注意してください。クラスタ内のデータノードは、データノードの中で使用可能なメモリーの容量がもっとも少ないノードより多いメモリーを使用することはできないためです。たとえば、クラスタのデータノードをホストしているコンピュータが 4 台あり、そのうちの 3 台がクラスタのデータの格納に 3G バイトの RAM を使用でき、残りのデータノードが 1G バイトの RAM のみを使用できる場合、各データノードは最大 1G バイトを MySQL Cluster のデータおよびインデックスに使用できます。

場合によっては、ndb_mgm -e "ALL REPORT MEMORYUSAGE" で DataMemory に大きい空き領域が示されていても、Table is fullというエラーが MySQL クライアントアプリケーションに表示されることがあります。CREATE TABLE に MAX_ROWS オプションを使用すると、MySQL Cluster のテーブルのために追加のパーティションを作成して、ハッシュインデックスに使用できるメモリーを増やすように NDB に強制できます。通常、テーブルに格納することが予期されている行数の 2 倍の数値を MAX_ROWS に設定すれば十分です。

同様の理由で、データが大量にロードされたノードで、データノードが再起動される問題が発生することもあります。MySQL Cluster NDB 7.1 以降では、MinFreePct パラメータが追加され、DataMemory および IndexMemory の一部 (デフォルトでは 5%) を再起動で使用するために予約することによって、この問題に対処しています。この予約されたメモリーは、NDB テーブルまたはデータの格納には使用できません。

A.10.14 MySQL Cluster は TCP/IP を使用しています。これは、1 つ以上のノードをリモートの場所に配置して、インターネット経由で実行できることを意味しますか。

そのような状況でクラスタが確実に実行される可能性はほとんどありません。MySQL Cluster は、100M ビット/秒またはギガビット Ethernet (後者が望ましい) を使用した LAN 環境のような専用の高速接続が保証された状況で実行されることを想定して設計および実装されているためです。これより遅い環境で使用したときのパフォーマンスはテストされておらず保証されません。

また、MySQL Cluster のノード間の通信はセキュアではないことに注意することも重要です。それらは暗号化されておらず、ほかの保護メカニズムによる保護手段も講じられていません。クラスタのもっともセキュアな構成は、外部からクラスタのデータまたは管理ノードに直接アクセスされない、ファイアウォールの内側のプライベートネットワークです。(SQL ノードの場合は、MySQL サーバーのほかのインスタンスの場合と同様の予防措置を取るようにしてください。)詳細は、セクション18.5.11「MySQL Cluster のセキュリティー上の問題」を参照してください。

A.10.15 MySQL Cluster を使用するために、新しいプログラミング言語またはクエリー言語を学習する必要がありますか。

いいえ。クラスタ自体の管理および構成にはいくつかの特殊なコマンドが使用されますが、次の操作には標準の (My)SQL ステートメントのみが必要となります。

- テーブルの作成、変更、および削除
- テーブルデータの挿入、更新、および削除
- プライマリインデックスおよび一意のインデックスの作成、変更、および削除

MySQL Cluster を設定するには、いくつかの特殊な構成パラメータおよびファイルが必要となります。これらの詳細は、セクション18.3.2「MySQL Cluster の構成ファイル」を参照してください。

クラスタノードの起動、停止などのタスクのために、いくつかの簡単なコマンドが MySQL Cluster 管理クライアント (ndb_mgm) で使用されます。セクション18.5.2「MySQL Cluster 管理クライアントのコマンド」を参照してください。

A.10.16 MySQL Cluster によってサポートされるプログラミング言語および API は何ですか。

MySQL Cluster は、標準の MySQL サーバーと同じプログラミング API および言語 (ODBC、.Net、MySQL C API、および一般的なスクリプト言語 (PHP、Perl、Python など) のための多数のドライバを含む) をサポートしています。これらの API を使用して記述された MySQL Cluster アプリケーションは、ほかの MySQL アプリケーションと同様に動作します。SQL ステートメントを MySQL サーバー (MySQL Cluster の場合は SQL ノード) に送信して、データの行が含まれている応答を受け取ります。これらの API については、第23章「Connector および API」を参照してください。

MySQL Cluster は、NDB API を使用したアプリケーションプログラミングもサポートしています。これは、MySQL サーバーにアクセスする必要のない、MySQL Cluster のデータへの低レベルの C++ インタフェースを提供します。The NDB APIを参照してください。また、多くの NDBCLUSTER 管理関数が C 言語の MGM API によって提供されています。詳細は、The MGM APIを参照してください。

MySQL Cluster (NDB 7.1 以降) は、ClusterJ を使用した Java アプリケーションプログラミングもサポートしています。これは、セッションおよびトランザクションを使用して、データのドメインオブジェクトモデルをサポートします。詳細は、Java and NDB Clusterを参照してください。

MySQL Cluster (NDB 7.2 以降) には、memcached のサポートが追加され、開発者は memcached インタフェースを使用して MySQL Cluster に格納されているデータにアクセスできます。詳細は、ndbmemcache—Memcache API for NDB Cluster (DEPRECATED)を参照してください。

MySQL Cluster NDB 7.3 には、MySQL Cluster をデータストアとして使用する、Node.js に対して記述された NoSQL アプリケーションをサポートするアダプタが追加されました。詳細は、MySQL NoSQL Connector for JavaScriptを参照してください。

A.10.17 MySQL Cluster には管理ツールが含まれていますか。

MySQL Cluster には、基本的な管理機能を実行するためのコマンド行クライアントが含まれています。セクション18.4.5「ndb_mgm — MySQL Cluster 管理クライアント」およびセクション18.5.2「MySQL Cluster 管理クライアントのコマンド」を参照してください。

MySQL Cluster NDB 7.0 以降は MySQL Cluster Manager によってもサポートされます。これは別個の製品であり、ローリング再起動、構成の変更などの MySQL Cluster の多くの管理タスクを自動化できる拡張されたコマンド行インタフェースが提供されます。MySQL Cluster Manager については、MySQL™ Cluster Manager 1.3.6 User Manualを参照してください。

MySQL Cluster NDB 7.3 では、MySQL Cluster ソフトウェア配布の一部として、MySQL Cluster を設定および配備するためのブラウザベースのグラフィカルな Auto-Installer が導入されました。詳細は、セクション18.2.1「MySQL Cluster Auto-Installer」を参照してください。

A.10.18 MySQL Cluster を使用しているときに、エラーメッセージまたは警告メッセージの意味を確認するにはどうすればよいですか。

これを行うことができる方法は 2 つあります。

- エラー状態または警告状態を通知された直後に、mysql クライアント内で SHOW ERRORS または SHOW WARNINGS を使用します。
- システムのシェルプロンプトで、perror --ndb error_code を使用します。

A.10.19 MySQL Cluster はトランザクションセーフですか。どのような分離レベルがサポートされますか。

はい。NDB ストレージエンジンを指定して作成されたテーブルの場合は、トランザクションがサポートされます。現在、MySQL Cluster では READ COMMITTED トランザクション分離レベルのみがサポートされます。

A.10.20 MySQL Cluster によってサポートされるストレージエンジンは何ですか。

MySQL でのクラスタリングは、NDB ストレージエンジンによってのみサポートされます。つまり、MySQL Cluster のノード間でテーブルを共有するには、ENGINE=NDB (または同等のオプション ENGINE=NDBCLUSTER) を使用してテーブルを作成する必要があります。

MySQL Cluster で使用されている MySQL サーバーに、ほかのストレージエンジン (InnoDB、MyISAM など) を使用するテーブルを作成することはできますが、それらのテーブルは NDB を使用していないため、クラスタリングに参加しません。そのような各テーブルはそれが作成された MySQL サーバーインスタンスに厳密にローカルなテーブルとなります。

A.10.21 壊滅的な障害が発生した場合 (たとえば、街全体が停電かつUPS の障害が発生した場合)、すべてのデータが失われますか。

コミットされたすべてのトランザクションはログ記録されます。このため、大災害が起こった場合に一部のデータが失われることはありますが、それはごく限定的なものとなります。データの損失は、トランザクションごとの操作の数を最小限にすることによって、さらに減らすことができます。(どのような場合でも、1 つのトランザクションで多数の操作を実行するのはよい考えではありません。)

A.10.22 MySQL Cluster では FULLTEXT インデックスを使用できますか。

現在、FULLTEXT インデックスは、InnoDB ストレージエンジン (MySQL 5.6.4 以降) および MyISAM ストレージエンジンのみによってサポートされています。詳細は、セクション12.9「全文検索関数」を参照してください。

A.10.23 単一のコンピュータ上で複数のノードを実行できますか。

実行できますが常に推奨できるとは限りません。クラスタを実行する主な理由の 1 つは冗長性を持たせるためです。この冗長性の利点を完全に享受するには、各ノードを別個のマシン上に配置してください。単一のマシンに複数のノードを配置した場合、そのマシンで障害が発生したときに、それらのすべてのノードが失われます。このため、単一のマシンで複数のデータノードを実行する場合は、そのマシンの障害によってノードグループのすべてのデータノードが失われることがないように設定することが非常に重要です。

MySQL Cluster は低コスト (または無料) のオペレーティングシステムがロードされた一般的なハードウェアで実行できるため、追加のマシンを 1 台または 2 台購入する出費は、ミッションクリティカルなデータを保護するためには十分に価値があります。管理ノードを実行するクラスタホストの要件は最低限のものであることにも注意してください。このタスクは、300 MHz の Pentium または同等の CPU、オペレーティングシステムのための十分な RAM、および ndb_mgmd プロセスと ndb_mgm プロセスのための少量のオーバーヘッドに対応した装備があれば実行できます。

複数の CPU、コア、またはその両方を持つ単一のホストで、複数のクラスタデータノードを実行することは容認できます。MySQL Cluster NDB 7.0 以降では、そのようなシステムで使用することが意図された、マルチスレッドバージョンのデータノードのバイナリも提供されています。詳細は、セクション18.4.3「ndbmtd — MySQL Cluster データノードデーモン (マルチスレッド)」を参照してください。

同じマシン上でデータノードと SQL ノードを同時に実行できる場合もあります。そのような配置での実行状態は、さまざまな要因 (コアおよび CPU の数、データノードおよび SQL ノードのプロセスが使用できるディスクおよびメモリーの容量など) によって異なります。そのような構成を計画する場合は、これらの要因を考慮する必要があります。

A.10.24. MySQL Cluster を再起動せずにデータノードを追加できますか。

クラスタをオフラインにせずに、実行されている MySQL Cluster に新しいデータノードを追加できます。詳細は、セクション18.5.13「MySQL Cluster データノードのオンライン追加」を参照してください。

ほかのタイプの MySQL Cluster ノードの場合は、ローリング再起動を行えばよいだけです (セクション18.5.5「MySQL Cluster のローリング再起動の実行」を参照してください)。

A.10.25. MySQL Cluster を使用するときに、注意するべき制限はありますか。

MySQL Cluster NDB 7.3 以降の NDB テーブルの制限には次のものがあります。

- 一時テーブルはサポートされません。ENGINE=NDB または ENGINE=NDBCLUSTER を使用した CREATE TEMPORARY TABLE ステートメントは、エラーが発生して失敗します。
- NDBCLUSTER テーブルでサポートされるユーザー定義のパーティション化のタイプは、KEY および LINEAR KEY のみです。ほかのパーティショニングタイプを使用して NDB テーブルを作成しようとすると、エラーが発生して失敗します。
- FULLTEXT インデックスはサポートされません。
- インデックスのプリフィクスはサポートされません。インデックスを作成できるのは完全なカラムのみです。
- 空間インデックスはサポートされません (空間カラムは使用できます)。セクション11.5「空間データの拡張」を参照してください。
- 部分的なトランザクションおよび部分的なロールバックのサポートは、ほかのトランザクションストレージエンジン (個別のステートメントをロールバックできる InnoDB など) と同等です。
- テーブルごとに許可される属性の最大数は 512 個です。属性名は 31 文字以内である必要があります。各テーブルで、テーブル名とデータベース名を合わせた最大長は 122 文字です。
- テーブルの行の最大サイズは、BLOB 値をカウントせずに 14K バイトです。

  NDB テーブルごとの行数の制限はありません。テーブルサイズの制限は多くの要因によって異なり、各データノードで使用できる RAM の容量に特に関係しています。

MySQL Cluster の制限の完全なリストについては、セクション18.1.6「MySQL Cluster の既知の制限」を参照してください。セクション18.1.6.11「MySQL Cluster NDB 7.3 で解決された以前の MySQL Cluster の問題」も参照してください。

A.10.26. MySQL Cluster では外部キーはサポートされますか。

MySQL Cluster NDB 7.3 では、InnoDB ストレージエンジンと同等の外部キー制約のサポートが追加されました。詳細は、セクション1.7.3.2「FOREIGN KEY の制約」およびセクション13.1.17.2「外部キー制約の使用」を参照してください。外部キーのサポートが必要なアプリケーションの場合は、MySQL Cluster NDB 7.3 以降を使用してください。

A.10.27. 既存の MySQL データベースを MySQL Cluster にインポートするにはどうすればよいですか。

ほかのバージョンの MySQL の場合と同様に、データベースを MySQL Cluster にインポートできます。この FAQ の各所で説明されている制限以外の唯一の特別な要件は、クラスタに含まれるテーブルが NDB ストレージエンジンを使用する必要があることです。これは、ENGINE=NDB または ENGINE=NDBCLUSTER を指定してテーブルを作成する必要があることを意味します。

ほかのストレージエンジンを使用しているテーブルを、1 つ以上の ALTER TABLE ステートメントを使用して、NDBCLUSTER に変更することもできます。ただし、変更を行う前に、テーブルの定義が NDBCLUSTER ストレージエンジンと互換性がある必要があります。MySQL 5.6 では、追加の回避策も必要となります。詳細は、セクション18.1.6「MySQL Cluster の既知の制限」を参照してください。

A.10.28. MySQL Cluster のノードはどのようにして相互にやり取りしますか。

クラスタのノードは、3 種類の転送メカニズム (TCP/IP、SHM (共有メモリー)、および SCI (スケーラブルコヒーラントインタフェース)) を使用して通信できます。使用可能な場合、同じクラスタホスト上にあるノード間では SHM がデフォルトで使用されます。ただし、これは実験的とみなされます。SCI は、スケーラブルなマルチプロセッサシステムを構築するために使用される高速 (1 Gbps 以上) で高可用性のプロトコルであり、特殊なハードウェアおよびドライバが必要となります。MySQL Cluster の転送メカニズムとして SCI を使用する方法については、セクション18.3.5「MySQL Cluster での高速インターコネクトの使用」を参照してください。

A.10.29. アービトレータとは何ですか。

クラスタ内の 1 つ以上のデータノードで障害が発生した場合、相互に「認識」できないクラスタデータノードが発生することがあります。実際に、2 つのセットのデータノードがネットワークのパーティション化で別々に分離されることがあります (「スプリットブレイン」シナリオとも呼ばれます)。各セットのデータノードがクラスタ全体のように動作しようとするため、このタイプの状況は好ましくありません。競合するデータノードのセットのいずれかを選択するために、アービトレータが必要となります。

少なくとも 1 つのノードグループのすべてのデータノードが存続して場合、クラスタの単一のサブセットが独自に機能するクラスタを形成できないため、ネットワークのパーティション化は問題とはなりません。本当の問題はすべてのノードが存続している単一のノードグループがない場合に発生し、その場合はネットワークのパーティション化 (「スプリットブレイン」シナリオ) が発生する可能性があります。そしてアービトレータが必要となります。すべてのクラスタノードは、同じノードをアービトレータとして認識しますが、通常、これは管理サーバーです。ただし、クラスタ内の MySQL サーバーのいずれかを代わりにアービトレータとして動作するように構成できます。アービトレータは、クラスタノードの最初のセットがアクセスすることを受け入れ、残りのセットにシャットダウンするように指示します。アービトレータの選択は、MySQL サーバーおよび管理サーバーノードの ArbitrationRank 構成パラメータによって制御されます。MySQL Cluster NDB 7.0.7 以降では、ArbitrationRank 構成パラメータを使用してアービトレータの選択処理を制御することもできます。これらのパラメータについては、セクション18.3.2.5「MySQL Cluster 管理サーバーの定義」を参照してください。

アービトレータの役割によって、指定されたホストに重い要求が課されることはないため、アービトレータのホストはこの目的のために特別に処理が速いマシン、または追加のメモリーがあるマシンである必要はありません。

A.10.30. MySQL Cluster によってサポートされるデータ型を何ですか。

MySQL Cluster は MySQL の通常のデータ型 (MySQL の空間拡張に関連付けられている型を含む) をすべてサポートしています。ただし、NDB ストレージエンジンは空間インデックスをサポートしていません。(空間インデックスは MyISAM によってのみサポートされます。詳細は、セクション11.5「空間データの拡張」を参照してください。)また、NDB テーブルとともに使用された場合、インデックスに関していくつかの違いがあります。

> **注記**
>
> MySQL Cluster のディスクデータテーブル (つまり、TABLESPACE ... STORAGE DISK ENGINE=NDB または TABLESPACE ... STORAGE DISK ENGINE=NDBCLUSTER を指定して作成されたテーブル) には固定長の行のみが格納されています。これは、(たとえば) VARCHAR(255) カラムが含まれている各ディスクデータテーブルレコードで、実際に格納される文字数に関係なく、255 文字 (テーブルに使用されている文字セットおよび照合順序に必要となります) の領域が必要となることを意味します。

これらの問題については、セクション18.1.6「MySQL Cluster の既知の制限」を参照してください。

A.10.31. MySQL Cluster を起動および停止するにはどうすればよいですか。

クラスタ内の各ノードを次の順序で個別に起動する必要があります。

1. ndb_mgmd コマンドを使用して、管理ノードを起動します。

   -f オプションまたは --config-file オプションを指定して、構成ファイルのある場所を管理ノードに通知する必要があります。

2. ndbd コマンドを使用して、各データノードを起動します。

   データノードが管理サーバーに接続する方法を認識するように、-c オプションまたは --ndb-connectstring オプションを指定して各データノードを起動する必要があります。

3. 任意の起動スクリプト (mysqld_safe など) を使用して各 MySQL サーバー (SQL ノード) を起動します。

   各 MySQL サーバーは、--ndbcluster オプションおよび --ndb-connectstring オプションを指定して起動する必要があります。これらのオプションによって、NDBCLUSTER ストレージエンジンのサポート、および管理サーバーに接続する方法が mysqld で有効になります。

影響を受けるノードがあるマシンのシステムシェルで、これらの各コマンドを実行する必要があります。(そのマシンを物理的にその場で操作する必要はありません。リモートログインシェルをこの目的に使用できます。)クラスタが実行されていることを確認するには、管理ノードが収容されているマシンで NDB 管理クライアント ndb_mgm を起動して、SHOW コマンドまたは ALL STATUS コマンドを発行します。

実行されているクラスタをシャットダウンするには、管理クライアントで SHUTDOWN コマンドを発行します。または、システムシェルに次のコマンドを入力できます。

```
shell> ndb_mgm -e "SHUTDOWN"
```

(この例の引用符はオプションです。-e オプションの後ろのコマンド文字列にスペースがないためです。また、管理クライアントのほかのコマンドと同様に、SHUTDOWN コマンドでは大文字/小文字が区別されません。)

これらのコマンドのいずれかによって、ndb_mgm、ndb_mgm、および ndbd プロセスが正常に終了します。SQL ノードとして実行されている MySQL サーバーは、mysqladmin shutdown を使用して停止できます。

詳細は、セクション18.5.2「MySQL Cluster 管理クライアントのコマンド」およびセクション18.2.7「MySQL Cluster の安全なシャットダウンと再起動」を参照してください。

A.10.32. MySQL Cluster をシャットダウンすると、MySQL Cluster のデータはどうなりますか。

クラスタのデータノードによってメモリーに保持されていたデータがディスクに書き込まれ、次回クラスタが起動されたときにメモリーにリロードされます。

A.10.33. MySQL Cluster に複数の管理ノードを使用することはよい考えですか。

フェイルセーフとして役に立つことがあります。特定の時点でクラスタを制御しているのは 1 つの管理ノードのみですが、1 つの管理ノードをプライマリとして構成し、プライマリ管理ノードで障害が発生した場合に、1 つ以上の追加の管理ノードが引き継ぐようにすることができます。

MySQL Cluster の管理ノードを構成する方法については、セクション18.3.2「MySQL Cluster の構成ファイル」を参照してください。

A.10.34. 単一の MySQL Cluster に、異なる種類のハードウェアおよびオペレーティングシステムを混在させることはできますか。

はい。すべてのマシンおよびオペレーティングシステムが同じ「エンディアン」(すべてがビッグエンディアンまたはすべてがリトルエンディアン) であれば可能です。

異なる MySQL Cluster リリースのソフトウェアを各ノードに使用することもできます。ただし、ローリングアップグレードの手順の一部としてのみこれをサポートしています (セクション18.5.5「MySQL Cluster のローリング再起動の実行」を参照してください)。

A.10.35. 単一のホスト上で 2 つのデータノードを実行できますか。2 つの SQL ノードは実行できますか。

はい。実行できます。複数のデータノードの場合は、各ノードで別のデータディレクトリを使用することをお勧めします (必須ではありません)。単一のマシン上で複数の SQL ノードを実行する場合は、mysqld の各インスタンスで別の TCP/IP ポートを使用する必要があります。ただし、MySQL 5.6 では、1 台のマ

シンで特定のタイプの複数のクラスタノードを実行することは推奨されず、本番での使用はサポートされません。

また、ndbd プロセスおよび mysqld プロセスはメモリーで競合することがあるため、同じホスト上でデータノードおよび SQL ノードを一緒に実行しないことをお勧めします。

A.10.36 MySQL Cluster ではホスト名を使用できますか。

はい。クラスタのホストに DNS および DHCP を使用できます。ただし、アプリケーションが「ファイブナイン」可用性を要求する場合は、固定 (数値) IP アドレスを使用してください。クラスタホスト間通信を DNS、DHCP などのサービスに依存させると、潜在的な障害点が増えるためです。

A.10.37 MySQL Cluster では IPv6 はサポートされますか。

IPv6 は SQL ノード (MySQL サーバー) 間の接続ではサポートされますが、ほかのすべてのタイプの MySQL Cluster ノード間の接続には IPv4 を使用する必要があります。

具体的には、これは、MySQL Cluster 間のレプリケーションには IPv6 を使用できるが、同じ MySQL Cluster 内のノード間の接続には IPv4 を使用する必要があることを意味します。詳細は、セクション 18.6.3「MySQL Cluster レプリケーションの既知の問題」を参照してください。

A.10.38 複数の MySQL サーバーを持つ MySQL Cluster で MySQL ユーザーを管理するにはどうすればよいですか。

通常、MySQL のユーザーアカウントおよび権限は、同じ MySQL Cluster にアクセスする異なる MySQL サーバー間で自動的に伝播されません。MySQL Cluster NDB 7.2 以降では、MySQL Cluster は権限の配布のサポートを提供しています。権限の配布は自動的に有効になりませんが、MySQL Cluster のドキュメントで説明されている次の手順を使用して有効にできます。詳細は、セクション18.5.14「MySQL Cluster の配布された MySQL 権限」を参照してください。

A.10.39 SQL ノードの 1 つで障害が発生した場合に、クエリーの送信を続けるにはどうすればよいですか。

MySQL Cluster は SQL ノード間でどのような種類の自動フェイルオーバーも提供していません。SQL ノードを失ったときの処理、およびそれらの間のフェイルオーバーを行うように、アプリケーションで準備している必要があります。

A.10.40 MySQL Cluster をバックアップおよびリストアするにはどうすればよいですか。

MySQL Cluster 管理クライアントおよび ndb_restore プログラムの NDB にネイティブなバックアップおよびリストア機能を使用できます。セクション18.5.3「MySQL Cluster のオンラインバックアップ」およびセクション18.4.20「ndb_restore — MySQL Cluster バックアップのリストア」を参照してください。

このために mysqldump および MySQL サーバーで提供されている従来の機能を使用することもできます。詳細は、セクション4.5.4「mysqldump — データベースバックアッププログラム」を参照してください。

A.10.41 「エンジェルプロセス」とは何ですか。

このプロセスは、データノードプロセスをモニターし、必要に応じて再起動を試みます。ndbd を起動したあとに、システムでアクティブなプロセスのリストをチェックすると、その名前で実行されているプロセスが実際には 2 つあることを確認できます (簡潔にするために、出力では ndb_mgmd および ndbd を省略しました)。

```
shell> ./ndb_mgmd

shell> ps aux | grep ndb
me     23002  0.0  0.0 122948  3104 ?        Ssl  14:14   0:00 ./ndb_mgmd
me     23025  0.0  0.0   5284   820 pts/2    S+   14:14   0:00 grep ndb

shell> ./ndbd -c 127.0.0.1 --initial

shell> ps aux | grep ndb
me     23002  0.0  0.0 123080  3356 ?        Ssl  14:14   0:00 ./ndb_mgmd
me     23096  0.0  0.0  35876  2036 ?        Ss   14:14   0:00 ./ndbd -c 127.0.0.1 --initial
me     23097  1.0  2.4 524116 91096 ?        Sl   14:14   0:00 ./ndbd -c 127.0.0.1 --initial
me     23168  0.0  0.0   5284   812 pts/2    R+   14:15   0:00 grep ndb
```

メモリーおよび CPU の使用量が 0 と表示されている ndbd プロセスがエンジェルプロセスです。実際には、もちろんそれぞれにごく少量が使用されます。これは、メインの ndbd プロセス (データを実際に処理するプライマリデータノードプロセス) が実行されているかどうかを単純にチェックします。許可されている場合 (たとえば、StopOnError 構成パラメータが false に設定されている場合。セクション 18.3.3.1「MySQL Cluster データノードの構成パラメータ」を参照してください)、エンジェルプロセスはプライマリデータノードプロセスを再起動しようとします。

A.11 MySQL 5.6 FAQ: MySQL の中国語、日本語、および韓国語の文字セット

この一連のよくある質問は、CJK (中国語、日本語、韓国語) の問題に関する多くの問い合わせに対応している MySQL のサポートグループおよび開発グループの経験から得られたものです。

A.11.1 MySQL で使用できる CJK 文字セットはどの文字セットですか。 2861
A.11.2 CJK 文字をテーブルに挿入しました。SELECT でそれらが「?」文字として表示されるのはなぜですか。 2862
A.11.3 Big5 中国語文字セットを使用する場合は、どのような問題に注意するべきですか。 2863
A.11.4 日本語文字セットの変換が失敗するのはなぜですか。 2863
A.11.5 SJIS の 81CA を cp932 に変換する場合はどうすればよいですか。 2864
A.11.6 MySQL では円 (¥) 記号をどのように表しますか。 2864
A.11.7 MySQL で韓国語文字セットを使用する場合に注意する問題はありますか。 2864
A.11.8 なぜ「Incorrect string value」というエラーメッセージが表示されるのですか。 2865
A.11.9 Access、PHP、またはその他の API を使用したアプリケーションの GUI フロントエンドまたはブラウザで、CJK 文字が正しく表示されないのはなぜですか。 2865
A.11.10 MySQL 5.6 にアップグレードしました。文字セットに関して、MySQL 4.0 の動作に戻すにはどうすればよいですか。 2866
A.11.11 CJK 文字での LIKE 検索および FULLTEXT 検索が失敗することがあるのはなぜですか。 2867
A.11.12 文字 X がすべての文字セットで使用可能であるかどうかを判別するにはどうすればよいですか。 2868
A.11.13 CJK 文字列が Unicode で間違ってソートされるのはなぜですか。(I) 2869
A.11.14 CJK 文字列が Unicode で間違ってソートされるのはなぜですか。(II) 2869
A.11.15 補助文字が MySQL で拒否されるのはなぜですか。 2870
A.11.16 「CJKV」にするべきではありませんか。 2870
A.11.17 MySQL では CJK 文字をデータベース名およびテーブル名に使用できますか。 2870
A.11.18 MySQL マニュアルの中国語版、日本語版、および韓国語版はどこにありますか。 2870
A.11.19 MySQL での CJK および関連する問題についての支援はどこで受けることができますか。 2871

A.11.1 MySQL で使用できる CJK 文字セットはどの文字セットですか。

CJK 文字セットのリストは、MySQL のバージョンによって異なることがあります。たとえば、gb18030 文字セットは MySQL 5.7.4 より前はサポートされていませんでした。ただし、INFORMATION_SCHEMA.CHARACTER_SETS 表のすべてのエントリの DESCRIPTION カラムには適用可能な言語の名前が表示されるため、次のクエリーを使用して Unicode 以外のすべての CJK 文字セットの最新のリストを取得できます。

```
mysql> SELECT CHARACTER_SET_NAME, DESCRIPTION
    -> FROM INFORMATION_SCHEMA.CHARACTER_SETS
    -> WHERE DESCRIPTION LIKE '%Chin%'
    -> OR DESCRIPTION LIKE '%Japanese%'
    -> OR DESCRIPTION LIKE '%Korean%'
    -> ORDER BY CHARACTER_SET_NAME;
+--------------------+---------------------------------+
| CHARACTER_SET_NAME | DESCRIPTION                     |
+--------------------+---------------------------------+
| big5               | Big5 Traditional Chinese        |
| cp932              | SJIS for Windows Japanese       |
| eucjpms            | UJIS for Windows Japanese       |
| euckr              | EUC-KR Korean                   |
| gb18030            | China National Standard GB18030 |
| gb2312             | GB2312 Simplified Chinese       |
| gbk                | GBK Simplified Chinese          |
| sjis               | Shift-JIS Japanese              |
| ujis               | EUC-JP Japanese                 |
+--------------------+---------------------------------+
9 rows in set (0.01 sec)
```

(詳細は、セクション21.1「INFORMATION_SCHEMA CHARACTER_SETS テーブル」を参照してください。)

MySQL は中華人民共和国の正式な文字セットである GB 文字セット (Guojia Biaozhun、National Standard、または Simplified Chinese) の 3 つのバリアント (gb2312、gbk、および gb18030 (MySQL 5.7.4 で追加されました)) をサポートしています。

ユーザーが gbk の文字を gb2312 に挿入しようとすることがありますが、これはほとんどの場合動作します。gbk は gb2312 のスーパーセットであるためです。ただし、より珍しい漢字を挿入しようとすると動作しません。(例については、Bug #16072 を参照してください)。

ここでは、gb2312 または gbk で正当な文字を明らかにして、正式なドキュメントへの参照を示します。gb2312 または gbk のバグを報告する前に、これらのリファレンスを確認してください。

- MySQL の gbk は、実際には「Microsoft コードページ 936」です。これは、正式な gbk とは文字 A1A4 (中黒)、A1AA (エムダッシュ)、A6E0-A6F5、および A8BB-A8C0 が異なります。
- gbk/Unicode マッピングのリストについては、http://www.unicode.org/Public/MAPPINGS/VENDORS/MICSFT/WINDOWS/CP936.TXT を参照してください。

A.11.2. CJK 文字をテーブルに挿入しました。SELECT でそれらが「?」文字として表示されるのはなぜですか。

この問題は、通常、アプリケーションプログラムまたはオペレーティングシステムの設定と MySQL の設定が一致していないことが原因です。これらのタイプの問題を修正するための一般的な手順を次に示します。

- 使用している MySQL のバージョンを確認します。

  これを判別するには、SELECT VERSION(); ステートメントを使用します。

- 意図した文字セットがデータベースで実際に使用されていることを確認します。

  ユーザーは多くの場合、クライアントの文字セットはサーバーの文字セットまたは表示のために使用される文字セットと常に同じであると考えます。ただし、これらは両方とも間違った想定です。SHOW CREATE TABLE tablename の結果をチェックするか、次のステートメントを使用することによって確認できます。

```
SELECT character_set_name, collation_name
  FROM information_schema.columns
  WHERE table_schema = your_database_name
    AND table_name = your_table_name
    AND column_name = your_column_name;
```

- 正しく表示されない文字の 16 進値を判別します。

  テーブル table_name のカラム column_name のこの情報を取得するには、次のクエリーを使用します。

```
SELECT HEX(column_name)
FROM table_name;
```

  3F は ? 文字のエンコードです。これは、? が実際にカラムに格納される文字であることを意味します。これは、ほとんどの場合、特定の文字をクライアントの文字セットからターゲットの文字セットに変換するときの問題のために発生します。

- ラウンドトリップが可能であることを確認します (つまり、literal (または _introducer hexadecimal-value) を選択すると、結果として literal が取得される)。

  たとえば、日本語のカタカナ文字 Pe (ペ) はすべての CJK 文字セットに存在し、コードポイント値 (16 進コーディング) は 0x30da です。この文字のラウンドトリップをテストするには、次のクエリーを使用します。

```
SELECT 'ペ' AS `ペ`;        /* or SELECT _ucs2 0x30da; */
```

  結果も「ペ」ではない場合、ラウンドトリップは失敗しました。

  そのような失敗に関するバグレポートの場合は、SELECT HEX('ペ'); も試すように要請されることがあります。これにより、クライアントのエンコードが正しいかどうかを判断できます。

- 問題が MySQL 以外のブラウザまたはほかのアプリケーションの問題ではないことを確認します。

  このタスクを行うには、mysql クライアントプログラム (Windows の場合: mysql.exe) を使用します。mysql では正しく表示されるが、アプリケーションでは表示されない場合、問題はシステム設定が原因である可能性があります。

  設定を確認するには、次のような出力が表示される SHOW VARIABLES ステートメントを使用します。

```
mysql> SHOW VARIABLES LIKE 'char%';
+--------------------------+----------------------------------------+
| Variable_name            | Value                                  |
+--------------------------+----------------------------------------+
| character_set_client     | utf8                                   |
```

```
| character_set_connection | utf8                                  |
| character_set_database   | latin1                                |
| character_set_filesystem | binary                                |
| character_set_results    | utf8                                  |
| character_set_server     | latin1                                |
| character_set_system     | utf8                                  |
| character_sets_dir       | /usr/local/mysql/share/mysql/charsets/ |
+--------------------------+---------------------------------------+
8 rows in set (0.03 sec)
```

これらは、西洋諸国のサーバー (latin1 は西ヨーロッパ文字セットであり、MySQL のデフォルトです) に接続される国際業務用のクライアントの典型的な文字セット設定です (utf8 Unicode が使用されています)。

Latin よりも Unicode (通常、Unix での utf8 バリアント、および Windows での ucs2 バリアント) の方が望ましいですが、オペレーティングシステムのユーティリティーで最適にサポートされる文字セットではないことがあります。多くの Windows ユーザーは、Microsoft の文字セット (日本語 Windows 用の cp932 など) が適当であると判断します。

サーバーの設定を制御できず、基盤となっているコンピュータがわからない場合は、自分の国の一般的な文字セットに変更することを試してください (euckr = 韓国、gb18030、gb2312、または gbk = 中華人民共和国、big5 = 台湾、sjis、ujis、cp932、または eucjpms = 日本、ucs2 または utf8 = すべての国)。通常、クライアント、接続、および結果の設定のみを変更する必要があります。この 3 つを一度にすべて変更する簡単なステートメントがSET NAMES です。例:

```
SET NAMES 'big5';
```

設定が正しい場合は、my.cnf または my.ini を編集することによってそれを永続的なものにできます。たとえば、次のような行を追加できます。

```
[mysqld]
character-set-server=big5
[client]
default-character-set=big5
```

アプリケーションで使用されている API 構成の設定に問題があることもあります。詳細は、「GUI フロントエンドまたはブラウザで CJK 文字が正しく表示されないのはなぜですか」を参照してください。

A.11.3 Big5 中国語文字セットを使用する場合は、どのような問題に注意するべきですか。

MySQL は、香港および台湾 (中華民国) で一般的な Big5 文字セットをサポートしています。MySQL の big5 は、実際には、本来の big5 文字セットと非常に似ている Microsoft コードページ 950 です。この文字セットは MySQL バージョン 4.1.16 / 5.0.16 以降で変更されています (Bug #12476 のため)。たとえば、次のステートメントは最新のバージョンの MySQL では動作しますが、古いバージョンでは動作しません。

```
mysql> CREATE TABLE big5 (BIG5 CHAR(1) CHARACTER SET BIG5);
Query OK, 0 rows affected (0.13 sec)

mysql> INSERT INTO big5 VALUES (0xf9dc);
Query OK, 1 row affected (0.00 sec)

mysql> SELECT * FROM big5;
+------+
| big5 |
+------+
| 嫺 |
+------+
1 row in set (0.02 sec)
```

HKSCS 拡張を追加する機能要求は申請されています。この拡張を必要とするユーザーの場合、Bug #13577 のための推奨パッチが役に立つことがあります。

A.11.4 日本語文字セットの変換が失敗するのはなぜですか。

MySQL は、sjis、ujis、cp932、および eucjpms 文字セットと Unicode をサポートしています。一般的なニーズは文字セット間で変換を行うことです。たとえば、Unix のサーバー (通常は sjis または ujis が使用されます) と Windows のクライアント (通常は cp932 が使用されます) を使用している場合があります。

次の変換表で、ucs2 カラムは変換前を表し、sjis、cp932、ujis、および eucjpms カラムは変換後を表しています。つまり、後ろの 4 つのカラムは、CONVERT(ucs2) を使用するか、値が含まれている ucs2 カラムを sjis、cp932、ujis、または eucjpms カラムに割り当てたときの 16 進数の結果を示しています。

文字名	ucs2	sjis	cp932	ujis	eucjpms
BROKEN BAR (破断線)	00A6	3F	3F	8FA2C3	3F
FULLWIDTH BROKEN BAR (全角破断線)	FFE4	3F	FA55	3F	8FA2
YEN SIGN (円記号)	00A5	3F	3F	20	3F
FULLWIDTH YEN SIGN (全角円記号)	FFE5	818F	818F	A1EF	3F
TILDE (チルダ)	007E	7E	7E	7E	7E
OVERLINE (オーバーライン)	203E	3F	3F	20	3F
HORIZONTAL BAR (水平線)	2015	815C	815C	A1BD	A1BD
EM DASH (エムダッシュ)	2014	3F	3F	3F	3F
REVERSE SOLIDUS (リバースソリダス)	005C	815F	5C	5C	5C
FULLWIDTH "" (全角リバースソリダス)	FF3C	3F	815F	3F	A1C0
WAVE DASH (波ダッシュ)	301C	8160	3F	A1C1	3F
FULLWIDTH TILDE (全角チルダ)	FF5E	3F	8160	3F	A1C1
DOUBLE VERTICAL LINE (双柱)	2016	8161	3F	A1C2	3F
PARALLEL TO (平行)	2225	3F	8161	3F	A1C2
MINUS SIGN (マイナス記号)	2212	817C	3F	A1DD	3F
FULLWIDTH HYPHEN-MINUS (全角ハイフンマイナス)	FF0D	3F	817C	3F	A1DD
CENT SIGN (セント記号)	00A2	8191	3F	A1F1	3F
FULLWIDTH CENT SIGN (全角セント記号)	FFE0	3F	8191	3F	A1F1
POUND SIGN (ポンド記号)	00A3	8192	3F	A1F2	3F
FULLWIDTH POUND SIGN (全角ポンド記号)	FFE1	3F	8192	3F	A1F2
NOT SIGN (否定記号)	00AC	81CA	3F	A2CC	3F
FULLWIDTH NOT SIGN (全角否定記号)	FFE2	3F	81CA	3F	A2CC

では、この表の次の部分について考えてみましょう。

	ucs2	sjis	cp932
NOT SIGN (否定記号)	00AC	81CA	3F
FULLWIDTH NOT SIGN (全角否定記号)	FFE2	3F	81CA

これは、MySQL が NOT SIGN (Unicode の U+00AC) を sjis のコードポイント 0x81CA および cp932 のコードポイント 3F に変換することを意味します (3F は疑問符 (「?」) であり、変換を実行できないときに常に使用される文字です)。

A.11.5 SJIS の 81CA を cp932 に変換する場合はどうすればよいですか。

答えは「?」です。これについては深刻な苦情が寄せられています。多くのユーザーは、sjis の 81CA (NOT SIGN) が cp932 の 81CA (FULLWIDTH NOT SIGN) になるような「緩い」変換を望んでいます。この動作に変更することが検討されています。

A.11.6 MySQL では円 (¥) 記号をどのように表しますか。

いくつかのバージョンの日本語文字セット (sjis および euc の両方) では 5C がリバースソリダス (\。バックスラッシュとも呼ばれます) として扱われ、ほかの文字セットでは円記号 (¥) として扱われるため、問題が発生します。

MySQL は JIS (Japanese Industrial Standards) 標準に記述されている 1 つのバージョンのみに従っています。MySQL では 5C は常にリバースソリダス (\) です。

A.11.7 MySQL で韓国語文字セットを使用する場合に注意する問題はありますか。

理論的には、複数のバージョンの euckr (Extended Unix Code Korea) 文字セットがありますが、認識されている問題は 1 つだけです。

コードポイント 0x5c がウォン記号 (₩) である EUC-KR の「KS-Roman」バリアントではなく、コードポイント 0x5c がリバースソリダス (つまり、\) である EUC-KR の「ASCII」バリアントが使用されています。これは、Unicode の U+20A9 を euckr に変換できないことを意味します。

```
mysql> SELECT
    ->     CONVERT('₩' USING euckr) AS euckr,
    ->     HEX(CONVERT('₩' USING euckr)) AS hexeuckr;
+-------+----------+
| euckr | hexeuckr |
+-------+----------+
| ?     | 3F       |
+-------+----------+
1 row in set (0.00 sec)
```

A.11.8なぜ「Incorrect string value」というエラーメッセージが表示されるのですか。

説明のために、1 つの Unicode (ucs2) カラムおよび 1 つの中国語 (gb2312) カラムを持つテーブルを作成します。

```
mysql> CREATE TABLE ch
    -> (ucs2 CHAR(3) CHARACTER SET ucs2,
    -> gb2312 CHAR(3) CHARACTER SET gb2312);
Query OK, 0 rows affected (0.05 sec)
```

両方のカラムに珍しい文字「㑖」を挿入することを試みます。

```
mysql> INSERT INTO ch VALUES ('A㑖B','A㑖B');
Query OK, 1 row affected, 1 warning (0.00 sec)
```

警告が報告されました。次のステートメントを使用してその内容を確認します。

```
mysql> SHOW WARNINGS\G
*************************** 1. row ***************************
  Level: Warning
   Code: 1366
Message: Incorrect string value: '\xE6\xB1\x8CB' for column 'gb2312' at row 1
1 row in set (0.00 sec)
```

gb2312 カラムのみに関する警告でした。

```
mysql> SELECT ucs2,HEX(ucs2),gb2312,HEX(gb2312) FROM ch;
+-------+--------------+--------+-------------+
| ucs2  | HEX(ucs2)    | gb2312 | HEX(gb2312) |
+-------+--------------+--------+-------------+
| A㑖B | 00416C4C0042 | A?B    | 413F42      |
+-------+--------------+--------+-------------+
1 row in set (0.00 sec)
```

いくつかのことを説明する必要があります。

1. 「エラー」ではなく「警告」であることが MySQL の特徴です。あきらめずにできることをして最善の結果を得ることが試みられます。
2. 「㑖」という文字は gb2312 文字セットにありません。この問題については前に説明しました。
3. 古いバージョンの MySQL を使用している場合は、別のメッセージが表示される可能性があります。
4. sql_mode=TRADITIONAL を設定している場合は、警告ではなくエラーメッセージが表示されます。

A.11.9Access、PHP、またはその他の API を使用したアプリケーションの GUI フロントエンドまたはブラウザで、CJK 文字が正しく表示されないのはなぜですか。

mysql クライアント (Windows: mysql.exe) を使用してサーバーに直接接続し、同じクエリーをそこで試してください。mysql が正常に応答する場合、問題はアプリケーションインタフェースで初期化が必要であることである可能性があります。mysql で SHOW VARIABLES LIKE 'char%'; ステートメント使用して、使用される文字セットを確認します。Access を使用している場合は、Connector/ODBC を使用して接続することがほとんどです。この場合は、Configuring Connector/ODBCを確認してください。たとえば、big5 を使用している場合は、SET NAMES 'big5' と入力します。(この場合「;」は必要ありません)。ASP を使用している場合は、コードに SET NAMES を追加する必要があることがあります。過去に作成された例を次に示します。

```
<%
Session.CodePage=0
Dim strConnection
Dim Conn
strConnection="driver={MySQL ODBC 3.51 Driver};server=server;uid=username;" \
            & "pwd=password;database=database;stmt=SET NAMES 'big5';"
Set Conn = Server.CreateObject("ADODB.Connection")
Conn.Open strConnection
%>
```

同様に、Connector/Net で latin1 以外の文字セットを使用している場合は、接続文字列に文字セットを指定する必要があります。詳細は、Connector/NET Connectionsを参照してください。

PHP を使用している場合は、次のコードを試してください。

```
<?php
 $link = mysql_connect($host, $usr, $pwd);

 mysql_select_db($db);

 if( mysql_error() ) { print "Database ERROR: " . mysql_error(); }
 mysql_query("SET NAMES 'utf8'", $link);
?>
```

この場合、SET NAMES を使用して character_set_client、character_set_connection、および character_set_results を変更しています。

mysql ではなく、新しい mysqli 拡張を使用することをお勧めします。mysqli を使用する場合は、前の例を次のように書き直すことができます。

```
<?php
 $link = new mysqli($host, $usr, $pwd, $db);

 if( mysqli_connect_errno() )
 {
   printf("Connect failed: %s\n", mysqli_connect_error());
   exit();
 }

 $link->query("SET NAMES 'utf8'");
?>
```

PHP アプリケーションで頻繁に発生する別の問題は、ブラウザによって行われる想定に関連しています。<meta> タグを追加または変更するだけでこの問題が修正されることがあります。たとえば、ユーザーエージェントがページの内容を UTF-8 として解釈するようにするには、HTML ページの <head> に <meta http-equiv="Content-Type" content="text/html; charset=utf-8"> を含めてください。

Connector/J を使用している場合は、Using Character Sets and Unicodeを参照してください。

A.11.10. MySQL 5.6 にアップグレードしました。文字セットに関して、MySQL 4.0 の動作に戻すにはどうすればよいですか。

MySQL バージョン 4.0 では、サーバーおよびクライアントの両方のための単一の「グローバル」文字セットがあり、使用する文字の決定はサーバー管理者が行なっていました。これは MySQL バージョン 4.1 以降で変更されました。現在行われるのは、セクション10.1.4「接続文字セットおよび照合順序」に説明されているように、「ハンドシェイク」です。

> クライアントが接続するときに、使用する文字セットの名前をサーバーに送信します。サーバーはこの名前を使用して、character_set_client、character_set_results、および character_set_connection システム変数を設定します。実際には、サーバーは文字セット名を使用して SET NAMES 操作を実行します。

このことの影響は、--character-set-server=utf8 を指定して mysqld を開始することによって、クライアントの文字セットを制御できないことです。ただし、アジア地域には MySQL 4.0 の動作が望ましいという顧客がいます。この動作を維持できるように、--skip-character-set-client-handshake を使用してオフにできる mysqld スイッチ --character-set-client-handshake が追加されました。--skip-character-set-client-handshake を指定して mysqld を起動すると、クライアントが接続するときに、使用する文字セットの名前がサーバーに送信されますが、サーバーはクライアントからのこの要求を無視します。

例として、望ましいサーバーの文字セットが latin1 であるとします (CJK 地域ではあまりないことですが、これはデフォルト値です)。クライアントのオペレーティングシステムでサポートされている文字セット

であるため、クライアントが utf8 を使用しているとします。latin1 をデフォルトの文字セットとしてサーバーを起動します。

```
mysqld --character-set-server=latin1
```

そのあと、クライアントをデフォルトの文字セット utf8 で起動します。

```
mysql --default-character-set=utf8
```

現在の設定は、SHOW VARIABLES の出力を表示することによって確認できます。

```
mysql> SHOW VARIABLES LIKE 'char%';
+--------------------------+----------------------------------------+
| Variable_name            | Value                                  |
+--------------------------+----------------------------------------+
| character_set_client     | utf8                                   |
| character_set_connection | utf8                                   |
| character_set_database   | latin1                                 |
| character_set_filesystem | binary                                 |
| character_set_results    | utf8                                   |
| character_set_server     | latin1                                 |
| character_set_system     | utf8                                   |
| character_sets_dir       | /usr/local/mysql/share/mysql/charsets/ |
+--------------------------+----------------------------------------+
8 rows in set (0.01 sec)
```

クライアントを停止し、mysqladmin を使用してサーバーを停止します。今度はハンドシェイクをスキップするように通知して、サーバーをふたたび起動します。

```
mysqld --character-set-server=utf8 --skip-character-set-client-handshake
```

クライアントをデフォルトの文字セットとして utf8 をふたたび指定して起動し、現在の設定を表示します。

```
mysql> SHOW VARIABLES LIKE 'char%';
+--------------------------+----------------------------------------+
| Variable_name            | Value                                  |
+--------------------------+----------------------------------------+
| character_set_client     | latin1                                 |
| character_set_connection | latin1                                 |
| character_set_database   | latin1                                 |
| character_set_filesystem | binary                                 |
| character_set_results    | latin1                                 |
| character_set_server     | latin1                                 |
| character_set_system     | utf8                                   |
| character_sets_dir       | /usr/local/mysql/share/mysql/charsets/ |
+--------------------------+----------------------------------------+
8 rows in set (0.01 sec)
```

SHOW VARIABLES の異なる結果を比較することによって確認できるように、--skip-character-set-client-handshake が使用された場合、サーバーはクライアントの初期設定を無視します。

A.11.1 CJK 文字での LIKE 検索および FULLTEXT 検索が失敗することがあるのはなぜですか。

BINARY カラムおよび BLOB カラムに対する LIKE 検索には非常に単純な問題があります。文字の終わりを判別する必要があることです。マルチバイト文字セットでは、各文字のオクテット長が異なることがあります。たとえば、utf8 の場合、次に示すように A は 1 バイトを必要としますが、ぺ は 3 バイトを必要とします。

```
+-------------------------+--------------------------+
| OCTET_LENGTH(_utf8 'A') | OCTET_LENGTH(_utf8 'ぺ') |
+-------------------------+--------------------------+
|                       1 |                        3 |
+-------------------------+--------------------------+
1 row in set (0.00 sec)
```

最初の文字の終わりを判別できなければ、2 番目の文字が始まる位置を判別できません。この場合、LIKE '_A%' などの非常に単純な検索が失敗します。解決策は、通常の CJK 文字セットを最初から使用するか、比較する前に CJK 文字を変換することです。

これは、存在しない文字のエンコードを MySQL が許可できない理由の 1 つです。不正な入力を厳密に拒否しなければ、文字の終わりを判別する方法がありません。

FULLTEXT 検索の場合は、単語の始まりと終わりを判別する必要があります。西洋の言語の場合、それらのほとんど (すべてでなければ) が簡単に識別できる単語の境界 (スペース文字) を使用しているため、これが問題となることはほとんどありません。ただし、通常、アジアの言語ではこれは異なります。独自の判断で不完全な方法を使用して、すべての漢字が単語を表すと想定したり、(日本語の場合) 文法的な終わりに従ったカタカナからひらがなへの変化に応じるようにしたりすることもできます。ただし、唯一の確実な解決策では、包括的な単語のリストが必要となります。これは、サポートされる各アジア言語のサーバーに辞書を含める必要があることを意味します。これは簡単に言って実現可能ではありません。

A.11.12 文字 X がすべての文字セットで使用可能であるかどうかを判別するにはどうすればよいですか。

簡体字中国語および半角ではない基本的な日本語のかな文字のほとんどは、すべての CJK 文字セットで表示されます。次のストアドプロシージャーは、UCS-2 Unicode 文字を受け入れて、それをほかのすべての文字セットに変換し、結果を 16 進数で表示します。

```
DELIMITER //

CREATE PROCEDURE p_convert(ucs2_char CHAR(1) CHARACTER SET ucs2)
BEGIN

CREATE TABLE tj
           (ucs2 CHAR(1) character set ucs2,
            utf8 CHAR(1) character set utf8,
            big5 CHAR(1) character set big5,
            cp932 CHAR(1) character set cp932,
            eucjpms CHAR(1) character set eucjpms,
            euckr CHAR(1) character set euckr,
            gb2312 CHAR(1) character set gb2312,
            gbk CHAR(1) character set gbk,
            sjis CHAR(1) character set sjis,
            ujis CHAR(1) character set ujis);

INSERT INTO tj (ucs2) VALUES (ucs2_char);

UPDATE tj SET utf8=ucs2,
            big5=ucs2,
            cp932=ucs2,
            eucjpms=ucs2,
            euckr=ucs2,
            gb2312=ucs2,
            gbk=ucs2,
            sjis=ucs2,
            ujis=ucs2;

/* If there is a conversion problem, UPDATE will produce a warning. */

SELECT hex(ucs2) AS ucs2,
      hex(utf8) AS utf8,
      hex(big5) AS big5,
      hex(cp932) AS cp932,
      hex(eucjpms) AS eucjpms,
      hex(euckr) AS euckr,
      hex(gb2312) AS gb2312,
      hex(gbk) AS gbk,
      hex(sjis) AS sjis,
      hex(ujis) AS ujis
FROM tj;

DROP TABLE tj;

END//
```

入力は、単一の ucs2 文字、またはその文字のコードポイント値 (16 進表記) です。たとえば、ucs2 のエンコードおよび名前についての Unicode のリスト (http://www.unicode.org/Public/UNIDATA/UnicodeData.txt) から、カタカナ文字 Pe はすべての CJK 文字セットにあり、コードポイント値が 0x30da であることがわかります。この値を p_convert() の引数として使用すると、次のような結果となります。

```
mysql> CALL p_convert(0x30da)//
+------+--------+------+-------+---------+-------+--------+------+------+------+
| ucs2 | utf8   | big5 | cp932 | eucjpms | euckr | gb2312 | gbk  | sjis | ujis |
+------+--------+------+-------+---------+-------+--------+------+------+------+
| 30DA | E3839A | C772 | 8379  | A5DA    | ABDA  | A5DA   | A5DA | 8379 | A5DA |
+------+--------+------+-------+---------+-------+--------+------+------+------+
1 row in set (0.04 sec)
```

カラム値が 3F (つまり、疑問符文字 (?)) であるカラムがないため、すべての変換が正常に行われたことがわかります。

A.11.13. CJK 文字列が Unicode で間違ってソートされるのはなぜですか。(I)

utf8_unicode_ci 検索や ucs2_unicode_ci 検索、または ORDER BY ソートの結果が、母国語のユーザーが予期する結果ではないことがあります。バグである可能性を除外するわけではありませんが、多くのユーザーは Unicode Collation Algorithm の標準表のウェイトを正しく読んでいないことが過去に判明しました。MySQL は http://www.unicode.org/Public/UCA/4.0.0/allkeys-4.0.0.txt にある表を使用しています。これは、unicode.org のホームページからナビゲートすることによって見つかる最初の表ではありません。MySQL はより新しい 4.1.0 の 「allkeys」 表ではなく、古い 4.0.0 の表を使用しているためです。(MySQL 5.6 の新しい '520' 照合順序では、5.2 の 「allkeys」 表が使用されています。)これは、次に示した Bug #16526 で報告されたような状況が発生しないように、インデックスに影響する順序付けの変更に非常に慎重であるためです。

```
mysql< CREATE TABLE tj (s1 CHAR(1) CHARACTER SET utf8 COLLATE utf8_unicode_ci);
Query OK, 0 rows affected (0.05 sec)

mysql> INSERT INTO tj VALUES ('が'),('か');
Query OK, 2 rows affected (0.00 sec)
Records: 2  Duplicates: 0  Warnings: 0

mysql> SELECT * FROM tj WHERE s1 = 'か';
+------+
| s1   |
+------+
| が   |
| か   |
+------+
2 rows in set (0.00 sec)
```

最初の結果行の文字は、検索した文字ではありません。MySQL でその文字が取得されたのはなぜでしょうか。まず、Unicode のコードポイント値を確認します。これを行うには、ucs2 バージョンの文字の 16 進数を表示します。

```
mysql> SELECT s1, HEX(CONVERT(s1 USING ucs2)) FROM tj;
+------+-----------------------------+
| s1   | HEX(CONVERT(s1 USING ucs2)) |
+------+-----------------------------+
| が   | 304C                        |
| か   | 304B                        |
+------+-----------------------------+
2 rows in set (0.03 sec)
```

4.0.0 allkeys 表で 304B および 304C を探すと、これらの行が見つかります。

```
304B  ; [.1E57.0020.000E.304B] # HIRAGANA LETTER KA
304C  ; [.1E57.0020.000E.304B][.0000.0140.0002.3099] # HIRAGANA LETTER GA; QQCM
```

正式な Unicode 名 (「#」 マークの後ろにあります) は、日本語の五十音 (ひらがな)、非公式な区分 (文字、数字、または句読点)、西洋言語での識別子 (KA または GA。これは、たまたま同じ文字のペアの有声音および無声音となっています) を示しています。さらに重要なのは、プライマリウェイト (角括弧の中の最初の 16 進数) が両方の行で 1E57 となっていることです。検索とソートの両方の比較で、MySQL はプライマリウェイトのみに注意を払って、ほかのすべての数値を無視します。これは、「が」および「か」が Unicode の仕様に従って正しくソートされていることを意味します。これらを区別するには、UCA (Unicode Collation Algorithm) 以外の照合順序 (utf8_bin または utf8_general_ci) を使用するか、HEX() 値を比較するか、ORDER BY CONVERT(s1 USING sjis) を使用する必要があります。もちろん、「Unicode に従えば」正しいだけでは十分ではなく、バグを報告したユーザーも同様に正しいと言えます。KA/GA のような有声音/無声音文字のペアを順序付けのために区別できる、JIS X 4061 標準に従った別の日本語の照合順序を追加する計画があります。

A.11.14. CJK 文字列が Unicode で間違ってソートされるのはなぜですか。(II)

Unicode (ucs2 または utf8) を使用していて、Unicode のソート順がわかっているが (セクション A.11「MySQL 5.6 FAQ: MySQL の中国語、日本語、および韓国語の文字セット」を参照してください)、MySQL でテーブルが間違ってソートされているように見える場合は、まずテーブルの文字セットを確認してください。

```
mysql> SHOW CREATE TABLE t\G
*************************** 1. row ***************************
Table: t
Create Table: CREATE TABLE `t` (
```

```
`s1` char(1) CHARACTER SET ucs2 DEFAULT NULL
) ENGINE=MyISAM DEFAULT CHARSET=latin1
1 row in set (0.00 sec)
```

文字セットは正しいように見えるため、このカラムに関して INFORMATION_SCHEMA.COLUMNS テーブルで表示できる情報を確認します。

```
mysql> SELECT COLUMN_NAME, CHARACTER_SET_NAME, COLLATION_NAME
    -> FROM INFORMATION_SCHEMA.COLUMNS
    -> WHERE COLUMN_NAME = 's1'
    -> AND TABLE_NAME = 't';
+-------------+--------------------+-----------------+
| COLUMN_NAME | CHARACTER_SET_NAME | COLLATION_NAME  |
+-------------+--------------------+-----------------+
| s1          | ucs2               | ucs2_general_ci |
+-------------+--------------------+-----------------+
1 row in set (0.01 sec)
```

(詳細は、セクション21.4「INFORMATION_SCHEMA COLUMNS テーブル」を参照してください。)

照合順序が ucs2_unicode_ci ではなく ucs2_general_ci であることがわかります。このようになる理由は、次のように SHOW CHARSET を使用して見つけることができます。

```
mysql> SHOW CHARSET LIKE 'ucs2%';
+---------+---------------+--------------------+--------+
| Charset | Description   | Default collation | Maxlen |
+---------+---------------+--------------------+--------+
| ucs2    | UCS-2 Unicode | ucs2_general_ci   |      2 |
+---------+---------------+--------------------+--------+
1 row in set (0.00 sec)
```

ucs2 および utf8 の場合、デフォルトの照合順序は 「general」 です。Unicode の照合順序を指定するには、COLLATE ucs2_unicode_ci を使用します。

A.11.15. 補助文字が MySQL で拒否されるのはなぜですか。

MySQL 5.5.3 より前では、MySQL は補助文字 (つまり、UTF-8 で 3 バイトを超えるサイズを必要とする文字) をサポートしていません。Unicode で基本多言語面/第 0 面と呼ばれているもののみがサポートされます。いくつかの非常に珍しい漢字のみが補助文字となっています。それらに対するサポートは一般的ではありません。このことが Bug #12600 などで報告され、「バグではありません」として拒否されました。utf8 では、理解できないバイトが出現した場合に、入力文字列を切り捨てる必要があります。そうしないと、不正なマルチバイト文字の長さがわかりません。

考えられる 1 つの回避策は、utf8 の代わりに ucs2 を使用することです。この場合、「不正な」文字は疑問符に置き換えられますが、切り捨ては行われません。妥当性チェックが行われない BLOB または BINARY にデータ型を変更することもできます。

MySQL 5.5.3 の時点では、追加の Unicode 文字セット utf16、utf32、および 4 バイトの utf8mb4 によって Unicode のサポートが拡張され、補助文字が含まれるようになりました。これらの文字セットでは、基本多言語面 (BMP) の外にある補助 Unicode 文字がサポートされます。

A.11.16. 「CJKV」にするべきではありませんか。

いいえ。用語「CJKV」(Chinese Japanese Korean Vietnamese) は漢字 (もともとは中国語) が含まれているベトナム文字セットを指しています。MySQL では、漢字を使用した古いベトナム文字をサポートする計画はありません。MySQL では、西洋文字を使用する現代のベトナム文字はもちろんサポートされます。

MySQL 5.6 の時点では、セクション10.1.14.1「Unicode 文字セット」で説明しているように、Unicode 文字セットにベトナム語の照合順序があります。

A.11.17. MySQL では CJK 文字をデータベース名およびテーブル名に使用できますか。

この問題は、対応するディレクトリおよびファイルの名前を自動的に書き換えることによって、MySQL 5.1 で修正されました。

たとえば、オペレーティングシステムでディレクトリ名に CJK がサポートされないサーバーに「楮」という名前のデータベースを作成した場合、MySQL は E6A5AE (つまり、「楮」文字の Unicode の 16 進表現) を特殊な方法でエンコードした @0w@00a5@00ae という名前のディレクトリが作成されます。ただし、SHOW DATABASES ステートメントを実行すると、データベースが「楮」として示されます。

A.11.18. MySQL マニュアルの中国語版、日本語版、および韓国語版はどこにありますか。

MySQL 5.1.12 の簡体字中国語版のマニュアルは、http://dev.mysql.com/doc/にあります。MySQL 4.1 マニュアルの日本語版は、http://dev.mysql.com/doc/ からダウンロードできます。

A.11.19 MySQL での CJK および関連する問題についての支援はどこで受けることができますか。

次のリソースを利用できます。

- MySQL ユーザーグループのリストは、https://wikis.oracle.com/display/mysql/List+of+MySQL+User+Groups にあります。
- 文字セットの問題に関連する機能要求については、http://tinyurl.com/y6xcuf を参照してください。
- MySQL の Character Sets, Collation, Unicode Forum にアクセスしてください。また、http://forums.mysql.com/ に外国語フォーラムを追加する作業が行われています。

A.12 MySQL 5.6 FAQ: コネクタおよび API

MySQL Connector およびその他の API に関する一般的な質問、問題、および回答については、マニュアルの次の部分を参照してください。

- セクション23.7.15「C API を使用する場合の一般的な質問と問題」
- Common Problems with MySQL and PHP
- Connector/ODBC Notes and Tips
- Connector/NET Programming
- MySQL Connector/J 5.1 Developer Guide

A.13 MySQL 5.6 FAQ: レプリケーション

次のセクションでは、MySQL レプリケーションに関するよくある質問に回答しています。

A.13.1 スレーブはマスターに常時接続されている必要がありますか。 2871
A.13.2 レプリケーションを有効にするには、マスターとスレーブのネットワーク接続を有効にする必要がありますか。 2872
A.13.3 マスターと比較してスレーブがどれくらい遅れているかを判別するにはどのようにすればよいですか。言い換えると、スレーブによってレプリケートされた最後のステートメントの日付はどのように判別できますか。 2872
A.13.4 スレーブが追いつくまで、マスターで更新をブロックするにはどうすればよいですか。 2872
A.13.5 二方向レプリケーションを設定する場合に注意する問題はありますか。 2872
A.13.6 レプリケーションを使用してシステムのパフォーマンスを改善するにはどうすればよいですか。 2872
A.13.7 パフォーマンスがより高いレプリケーションを使用するには、アプリケーションのクライアントコードを準備するときに何を行えばよいですか。 2873
A.13.8 MySQL レプリケーションによってシステムのパフォーマンスが向上するのは、どのような場合でどの程度向上しますか。 2873
A.13.9 冗長性または高可用性を持たせるには、レプリケーションをどのように使用すればよいですか。 2874
A.13.10 マスターサーバーがステートメントベースのバイナリロギング形式または行ベースのバイナリロギング形式のどちらを使用しているかを判別するにはどうすればよいですか。 2874
A.13.11 行ベースのレプリケーションを使用するようにスレーブに指示するにはどうすればよいですか。 ... 2874
A.13.12 GRANT ステートメントおよび REVOKE ステートメントがスレーブマシンにレプリケートされないようにするにはどうすればよいですか。 2874
A.13.13 オペレーティングシステムが混在している場合、レプリケーションは動作しますか (たとえば、マスターは Linux で実行され、スレーブは OS X および Windows で実行されている場合)。 2874
A.13.14 ハードウェアのアーキテクチャーが混在している場合、レプリケーションは動作しますか (たとえば、マスターは 64 ビットマシンで実行され、スレーブは 32 ビットマシンで実行されている場合)。 2874

A.13.1 スレーブはマスターに常時接続されている必要がありますか。

いいえ、その必要はありません。スレーブは、シャットダウンされるか、数時間または数日間切断されても、再接続してマスターの更新に追いつくことができます。たとえば、リンクが散発的に短時間接続されるダイヤルアップリンクを使用して、マスターとスレーブの関係を設定できます。これは、特殊な対応が行われていないかぎり、特定の時点でスレーブがマスターと同期されていることは保証されないことを意味します。

切断されているスレーブが追いつくことができるようにするには、スレーブにまだレプリケートされていない情報が含まれているマスターからのバイナリログファイルを削除しないでください。非同期レプリケーションは、スレーブが最後にイベントを読み取った位置からバイナリログを継続して読み取ることができる場合にのみ動作できます。

A.13.2. レプリケーションを有効にするには、マスターとスレーブのネットワーク接続を有効にする必要がありますか。

はい。マスターとスレーブのネットワーク接続を有効にする必要があります。ネットワーク接続を有効にしないと、スレーブはマスターに接続できず、バイナリログを転送できません。両方のサーバーの構成ファイルで skip-networking オプションが有効にされていないことを確認します。

A.13.3. マスターと比較してスレーブがどれくらい遅れているかを判別するにはどのようにすればよいですか。言い換えると、スレーブによってレプリケートされた最後のステートメントの日付はどのように判別できますか。

SHOW SLAVE STATUS の出力の Seconds_Behind_Master カラムを確認してください。セクション 17.1.5.1「レプリケーションステータスの確認」を参照してください。

スレーブの SQL スレッドでマスターから読み取ったイベントを実行するときに、このスレッドは実行時の時間をイベントのタイムスタンプに変更します。(これにより、TIMESTAMP が正常にレプリケートされます。)SHOW PROCESSLIST の出力で、スレーブの SQL スレッドの Time カラムに表示される秒数は、最後にレプリケートされたイベントのタイムスタンプとスレーブマシンの実際の時間の差異の秒数です。これを使用して、最後にレプリケートされたイベントの日付を判別できます。スレーブがマスターから切断されて 1 時間たってから再接続された直後の場合は、SHOW PROCESSLIST のスレーブ SQL スレッドに大きい Time 値 (3600 など) が表示されることがあります。これは、スレーブが 1 時間前のステートメントを実行しているためです。セクション17.2.1「レプリケーション実装の詳細」を参照してください。

A.13.4. スレーブが追いつくまで、マスターで更新をブロックするにはどうすればよいですか。

次の手順を使用します。

1. マスターで次のステートメントを実行します。

```
mysql> FLUSH TABLES WITH READ LOCK;
mysql> SHOW MASTER STATUS;
```

SHOW ステートメントの出力から、レプリケーション座標 (現在のバイナリログファイル名および位置) を記録します。

2. スレーブで次のステートメントを発行します。ここで、MASTER_POS_WAIT() 関数への引数は、前のステップで取得したレプリケーション座標値です。

```
mysql> SELECT MASTER_POS_WAIT('log_name', log_pos);
```

この SELECT ステートメントは、スレーブが指定されたログファイルおよび位置に達するまで更新をブロックします。その時点で、スレーブがマスターと同期され、ステートメントが戻ります。

3. マスターで次のステートメントを発行して、更新処理がマスターでふたたび開始されるようにします。

```
mysql> UNLOCK TABLES;
```

A.13.5. 二方向レプリケーションを設定する場合に注意する問題はありますか。

現在、MySQL レプリケーションでは、分散 (サーバー間) 更新の原子性を保証するためのマスターおよびスレーブ間のロックプロトコルはサポートされていません。言い換えると、クライアント A が co-master 1 を更新し、それを co-master 2 に伝播する前にクライアント B が co-master 2 を更新した場合、クライアント A の更新が co-master 1 への更新と異なる更新になる可能性があります。このため、クライアント A の更新が co-master 2 に行われたときに、co-master 2 からの更新がすべて伝播されたあとでも、co-master 1 とは異なるテーブルとなります。これは、どのような順序でも更新が安全に行われるという確証がある場合、またはクライアントコードで更新順序の間違いに対処できる場合を除いて、2 つのサーバーを二方向レプリケーション関係にするべきではないことを意味します。

二方向レプリケーションでは、実際には、更新に関してパフォーマンスはそれほど向上しません。各サーバーは、単一のサーバーで更新を行うときと同量の更新を行う必要があります。唯一の違いは、別のサーバーで発生した更新が 1 つのスレーブスレッドに直列化されるため、ロック競合がやや少ないことです。この利点さえも、ネットワーク遅延によって相殺されてしまうことがあります。

A.13.6. レプリケーションを使用してシステムのパフォーマンスを改善するにはどうすればよいですか。

1 つのサーバーをマスターとして設定し、すべての書き込みをそれに指示します。そして、予算とラックスペースが許すかぎり多くのスレーブを構成し、マスターと複数のスレーブに読み取りを分散します。スレーブ側の速度を改善するには、--skip-innodb、--low-priority-updates、および --delay-key-write=ALL オプションを指定してスレーブを起動することもできます。この場合、スレーブは InnoDB テーブルの代わりに非トランザクションの MyISAM テーブルを使用し、トランザクションのオーバーヘッドを排除して速度を上げます。

A.13.7 パフォーマンスがより高いレプリケーションを使用するには、アプリケーションのクライアントコードを準備するときに何を行えばよいですか。

スケールアウトソリューションとしてレプリケーションを使用するためのガイドについては、セクション 17.3.3「スケールアウトのためにレプリケーションを使用する」を参照してください。

A.13.8 MySQL レプリケーションによってシステムのパフォーマンスが向上するのは、どのような場合でどの程度向上しますか。

MySQL レプリケーションは、読み取りが頻繁に行われ、書き込みはそれほどないシステムにもっとも適しています。理論的には、単一のマスターおよび複数のスレーブのセットアップを使用する場合、ネットワーク帯域幅がなくなるか、マスターで更新のロードを処理できないポイントに到達するまでは、スレーブを追加することによってシステムを拡張できます。

処理能力の上積みが横ばいになる前のスレーブの数、およびサイトのパフォーマンスを改善できる度合いを判別するには、クエリーパターンを知り、典型的なマスターおよび典型的なスレーブでの読み取りおよび書き込みのスループットの関係をベンチマークすることによって経験的に判別する必要があります。ここでは、架空のシステムのレプリケーションの構成を単純に計算した例を示します。reads および writes は、1 秒当たりの読み取りおよび書き込みの数をそれぞれ示しています。

システムのロードは 10% の書き込みと 90% の読み取りで構成されていて、ベンチマークによって reads が 1200 - 2 * writes であることが判別されたとします。つまり、書き込みがない場合、システムは毎秒 1,200 回の読み取りを実行できます。書き込みの平均は読み取りの平均の 2 倍の時間がかかり、この関係はリニア (直線的) です。マスターと各スレーブの容量は同じであり、1 つのマスターと N 個のスレーブがあるとします。この場合、各サーバー (マスターまたはスレーブ) は次のようになります。

reads = 1200 - 2 * writes

reads = 9 * writes / (N + 1) (読み取りは分散されますが、書き込みはすべてのスレーブにレプリケートされます)

9 * writes / (N + 1) + 2 * writes = 1200

writes = 1200 / (2 + 9/(N + 1))

最後の等式は、実行可能な最大読み取り回数が毎秒 1200 回で、書き込み 1 回当たり 9 回の読み取りがある場合の N 個のスレーブの書き込みの最大数を示しています。

この分析によって、次の結論が導かれます。

- N = 0 (レプリケーションがないことを意味します) の場合、システムは毎秒 1200/11 = 109 回の書き込みを処理できます。
- N = 1 の場合、毎秒最大 184 回の書き込みを実行できます。
- N = 8 の場合、毎秒最大 400 回の書き込みを実行できます。
- N = 17 の場合、毎秒最大 480 回の書き込みを実行できます。
- N が無限に近づくと (予算も負の無限大になり)、毎秒 600 回の書き込みに近くなり、システムスループットは 5.5 倍になります。ただし、8 サーバーのみでも 4 倍近くに増えます。

これらの計算ではネットワーク帯域幅を無限と想定しており、システムにおいて重要である可能性があるさまざまなほかの要因が無視されています。多くの場合、N 個のレプリケーションスレーブを追加した場合のシステムの状態を正確に予測する前述と同様の計算を行うことはできません。ただし、次の質問に答えると、レプリケーションによってシステムのパフォーマンスが改善されるかどうか、およびどれくらい改善されるかを判別するのに役立ちます。

- システムの読み取りと書き込みの比率はどれくらいですか。
- 読み取りを減らした場合、単一のサーバーで書き込みのロードをどれくらい処理できますか。

- ネットワークで帯域幅を使用できるスレーブはいくつですか。

A.13.9 冗長性または高可用性を持たせるには、レプリケーションをどのように使用すればよいですか。

冗長性の実装方法は、アプリケーションおよび状況によってまったく異なります。高可用性ソリューション (自動フェイルオーバーを使用する) では、アクティブなモニタリング、および元の MySQL サーバーからスレーブへのフェイルオーバーのサポートを提供する、カスタムスクリプトまたはサードパーティーのツールが必要となります。

この処理を手動で行うには、新しいサーバーとやり取りするようにアプリケーションを変更するか、MySQL サーバーの DNS を障害が発生したサーバーから新しいサーバーに変更することによって、障害が発生したマスターから事前構成されているスレーブに切り替えることができる必要があります。

詳細およびソリューションの例については、セクション17.3.6「フェイルオーバー中にマスターを切り替える」を参照してください。

A.13.10 マスターサーバーがステートメントベースのバイナリロギング形式または行ベースのバイナリロギング形式のどちらを使用しているかを判別するにはどうすればよいですか。

binlog_format システム変数の値を確認します。

```
mysql> SHOW VARIABLES LIKE 'binlog_format';
```

表示される値は、STATEMENT、ROW、または MIXED のいずれかです。MIXED モードの場合、行ベースのロギングが望ましいのですが、特定の状況では、レプリケーションがステートメントベースのロギングに自動的に切り替わります。これが発生する場合については、セクション5.2.4.3「混合形式のバイナリロギング形式」を参照してください。

A.13.11 行ベースのレプリケーションを使用するようにスレーブに指示するにはどうすればよいですか。

スレーブは使用する形式を自動的に識別します。

A.13.12 GRANT ステートメントおよび REVOKE ステートメントがスレーブマシンにレプリケートされないようにするにはどうすればよいですか。

--replicate-wild-ignore-table=mysql.% オプションを指定してサーバーを起動し、mysql データベースのテーブルのレプリケーションが無視されるようにします。

A.13.13 オペレーティングシステムが混在している場合、レプリケーションは動作しますか (たとえば、マスターは Linux で実行され、スレーブは OS X および Windows で実行されている場合)。

はい。

A.13.14 ハードウェアのアーキテクチャーが混在している場合、レプリケーションは動作しますか (たとえば、マスターは 64 ビットマシンで実行され、スレーブは 32 ビットマシンで実行されている場合)。

はい。

A.14 MySQL 5.6 FAQ: MySQL エンタープライズスケーラビリティースレッドプール

A.14.1 スレッドプールとは何ですか。これはどのような問題を解決しますか。 2874
A.14.2 スレッドプールは、最適なパフォーマンスおよびスループットのために、並列セッションおよびトランザクションをどのように制限および管理しますか。 2875
A.14.3 スレッドプールは、クライアント側の接続プールとどのように異なりますか。 2875
A.14.4 スレッドプールはどのような場合に使用しますか。 2875
A.14.5 推奨されるスレッドプールの構成はありますか。 2875

A.14.1 スレッドプールとは何ですか。これはどのような問題を解決しますか。

MySQL スレッドプールは、MySQL サーバーのデフォルトの接続処理機能を拡張する MySQL サーバーのプラグインであり、ステートメント/クエリーおよびトランザクションの並列実行数を制限して、タスクを完了するための CPU およびメモリーのリソースがそれぞれに十分に確保されるようにします。MySQL 5.5 および 5.6 の商用配布には、スレッドプールプラグインが含まれています。

MySQL サーバーのデフォルトのスレッド処理モデルでは、クライアント接続ごとに 1 つのスレッドを使用してステートメントが実行されます。より多くのクライアントがサーバーに接続してステートメントを

実行すると、全体的なパフォーマンスが低下します。スレッドプールプラグインは、オーバーヘッドを軽減してパフォーマンスを向上させるように設計された代替のスレッド処理モデルを提供します。スレッドプールプラグインは、特に現在のマルチ CPU/コアシステムでの多数のクライアント接続において、ステートメント実行スレッドを効率的に管理することによって、サーバーのパフォーマンスを向上させます。

詳細は、セクション8.11.6「スレッドプールプラグイン」を参照してください。

A.14.2 スレッドプールは、最適なパフォーマンスおよびスループットのために、並列セッションおよびトランザクションをどのように制限および管理しますか。

スレッドプールは、「分割統治」手法を使用して並列性を制限および調整します。MySQL サーバーのデフォルトの接続処理と異なり、スレッドプールでは接続とスレッドが分離され、接続およびそれらの接続から受け取ったステートメントを実行するスレッドが固定された関係を持たないようになっています。スレッドプールは、構成可能なスレッドグループ内でクライアント接続を管理します。スレッドグループでは、実行される処理の性質に基づいて優先順位が付けられ、キューに入れられます。

詳細は、セクション8.11.6.2「スレッドプール操作」を参照してください。

A.14.3 スレッドプールは、クライアント側の接続プールとどのように異なりますか。

MySQL 接続プールはクライアント側で動作し、MySQL サーバーに対して MySQL クライアントで接続および切断が繰り返されないようにします。MySQL クライアントのアイドル状態の接続をキャッシュし、必要に応じてほかのユーザーが使用できるように設計されています。これにより、クエリーが MySQL サーバーに発行されたときに、接続の確立および切断のオーバーヘッドおよびコストが最小化されます。MySQL 接続プールは、クエリー処理機能、またはバックエンドの MySQL サーバーの負荷を認識できません。それとは対照的に、スレッドプールは MySQL サーバー側で動作し、インバウンドの並列接続の実行、およびバックエンドの MySQL データベースにアクセスするクライアント接続から受け取ったクエリーの実行を管理するように設計されています。機能が分離されているため、MySQL 接続プールとスレッドプールは次元の異なるものであり、相互に独立して使用できます。

MySQL Connector を使用した MySQL の接続プールについては、第23章「Connector および API」で説明しています。

A.14.4 スレッドプールはどのような場合に使用しますか。

スレッドプールを最適に使用するには、考慮するべきいくつかの経験則があります。

MySQL の Threads_running 変数は、MySQL サーバーで現在実行されている並列ステートメントの数を追跡します。この変数がサーバーが最適に動作しない範囲に常時ある場合 (通常、InnoDB のワークロードの場合、40 を超えている場合)、特にきわめて並列負荷が高い状況では、スレッドプールを使用することは効果があります。

innodb_thread_concurrency を使用して同時に実行されるステートメントの数を制限している場合、スレッドプールを使用すると、トランザクションの内容、ユーザー定義の指定などに基づいて、接続をスレッドグループに割り当てて実行をキューに入れることによって、同じ問題が少しだけ良好に解決されます。

最後に、ワークロードが主に短いクエリーで構成されている場合、スレッドプールを使用することは効果があります。

詳細は、セクション8.11.6.3「スレッドプールのチューニング」を参照してください。

A.14.5 推奨されるスレッドプールの構成はありますか。

スレッドプールには、パフォーマンスに影響する、ユーザー事例から追加された多数の構成パラメータがあります。これらの詳細、およびチューニングのヒントについては、セクション8.11.6.3「スレッドプールのチューニング」を参照してください。

付録 B エラー、エラーコード、および一般的な問題

目次

B.1 エラー情報のソース 2877
B.2 エラー値のタイプ 2877
B.3 サーバーのエラーコードおよびメッセージ 2878
B.4 クライアントのエラーコードおよびメッセージ 2937
B.5 問題および一般的なエラー 2941
B.5.1 問題の原因を判別する方法 2941
B.5.2 MySQL プログラム使用時の一般的なエラー 2942
B.5.3 インストール関連の問題 2953
B.5.4 管理関連の問題 2954
B.5.5 クエリー関連の問題 2960
B.5.6 オプティマイザ関連の問題 2966
B.5.7 テーブル定義関連の問題 2966
B.5.8 MySQL の既知の問題 2967

この付録では、一般的な問題、発生する可能性があるエラー、および考えられる解決策を一覧表示することに加えて、ホスト言語から MySQL を呼び出したときに発生する可能性があるエラーも一覧表示しています。最初のセクションは問題と解決策について説明しています。エラーに関する詳細な情報があります。1 つのリストには、サーバーのエラーメッセージが示されています。もう 1 つのリストには、クライアントプログラムのメッセージが示されています。

B.1 エラー情報のソース

MySQL にはエラー情報のソースがいくつかあります。

- 各 SQL ステートメントを実行すると、セクションB.2「エラー値のタイプ」で説明されているように、エラーコード、SQLSTATE 値、およびエラーメッセージが返されます。これらのエラーはサーバー側から返されます (セクションB.3「サーバーのエラーコードおよびメッセージ」を参照してください)。
- クライアント側でエラーが発生することがあり、通常、サーバーと通信するプログラムが関係しています (セクションB.4「クライアントのエラーコードおよびメッセージ」を参照してください)。
- SQL ステートメントの警告およびエラーの情報は、SHOW WARNINGS ステートメントおよび SHOW ERRORS ステートメントを使用して表示できます。warning_count システム変数は、エラー、警告、および注意の数を示します。error_count システム変数はエラー数を示します。この値には警告および注意は含まれていません。
- GET DIAGNOSTICS ステートメントは、診断領域の診断情報を調査するために使用できます。セクション13.6.7.3「GET DIAGNOSTICS 構文」を参照してください。
- SHOW SLAVE STATUS ステートメントの出力には、スレーブ側で発生したレプリケーションエラーに関する情報が含まれています。
- InnoDB テーブルに対するCREATE TABLE ステートメントが失敗した場合、SHOW ENGINE INNODB STATUS ステートメントの出力には、最後の外部キーエラーに関する情報が含まれています。
- perror プログラムは、コマンド行からエラー番号に関する情報を表示します。セクション4.8.1「perror — エラーコードの説明」を参照してください。

サーバーエラーとクライアントエラーについては、この付録の以降のセクションで説明します。InnoDB に関連するエラーについては、セクション14.19.4「InnoDB のエラー処理」を参照してください。

B.2 エラー値のタイプ

MySQL でエラーが発生すると、サーバーは 2 種類のエラー値を返します。

- MySQL 固有のエラーコード。この値は数字です。これはほかのデータベースシステムには移植できません。
- SQLSTATE 値。値は 5 文字の文字列です (たとえば、'42S02')。この値は、ANSI SQL および ODBC から採用されており、より標準化されています。

エラーの説明を示すメッセージ文字列も利用できます。

エラーが発生した場合は、C API 関数を使用して、MySQL のエラーコード、SQLSTATE 値、およびメッセージ文字列を利用できます。

- MySQL エラーコード: Call mysql_errno()
- SQLSTATE 値: Call mysql_sqlstate()
- エラーメッセージ: Call mysql_error()

準備済みステートメントの場合、対応するエラー関数は mysql_stmt_errno()、mysql_stmt_sqlstate()、および mysql_stmt_error() です。すべてのエラー関数については、セクション23.7「MySQL C API」で説明しています。

最後のステートメントのエラー、警告、および注意の数は、mysql_warning_count() を呼び出すことによって取得できます。セクション23.7.7.73「mysql_warning_count()」を参照してください。

SQLSTATE 値の最初の 2 文字はエラークラスを示しています。

- クラス = '00' は成功を示しています。
- クラス = '01' は警告を示しています。
- クラス = '02' は 「Not Found」 を示しています。これは、カーソルのコンテキストに関係しており、カーソルがデータセットの最後に達したときの動作を制御するために使用します。この状況は、行が取得されない SELECT ... INTO var_list ステートメントでも発生します。
- クラス > '02' は例外を示しています。

B.3 サーバーのエラーコードおよびメッセージ

MySQL プログラムは、サーバーがエラーを返したときに、いくつかのタイプのエラー情報にアクセスできます。たとえば、mysql クライアントプログラムは、次の形式を使用してエラーを表示します。

```
shell> SELECT * FROM no_such_table;
ERROR 1146 (42S02): Table 'test.no_such_table' doesn't exist
```

表示されるメッセージには、3 つのタイプの情報が含まれています。

- 数字のエラーコード (1146)。この数字は MySQL 固有であり、ほかのデータベースシステムに移植できません。
- 5 文字の SQLSTATE 値 ('42S02')。この値は、ANSI SQL および ODBC から採用されており、より標準化されています。すべての MySQL エラー番号に、対応する SQLSTATE 値があるわけではありません。それらの場合は、'HY000' (一般エラー) が使用されます。
- エラーの説明を示すメッセージ文字列。

エラーを確認する場合は、エラーメッセージではなくエラーコードを使用してください。エラーメッセージが頻繁に変更されることはありませんが、変更される可能性はあります。また、データベース管理者が言語設定を変更した場合は、エラーメッセージの言語に影響します。

エラーコードは、特定の MySQL シリーズの GA リリースにおいて固定されています。シリーズが GA ステータスとなる前は、新しいコードはまだ開発中であることがあり、変更されることがあります。

サーバーのエラー情報では、次のソースファイルが使用されています。エラー情報が定義される仕組みについては、「MySQL Internals Manual」を参照してください。

- エラーメッセージ情報は share/errmsg.txt ファイルに一覧表示されています。%d および %s は数字および文字列をそれぞれ表しており、表示されるときにメッセージ値に置換されます。
- share/errmsg.txt に一覧表示されているエラー値は、MySQL ソースファイル (include/mysqld_error.h および include/mysqld_ername.h) の定義を生成するために使用されます。
- share/errmsg.txt に一覧表示されている SQLSTATE 値は、MySQL ソースファイル (include/sql_state.h) の定義を生成するために使用されます。

更新は頻繁に行われるため、ここに一覧表示されていない追加のエラー情報がこれらのファイルに含まれている可能性があります。

- エラー: 1000 SQLSTATE: HY000 (ER_HASHCHK)

  メッセージ: hashchk

使用されません。

- エラー: 1001 SQLSTATE: HY000 (ER_NISAMCHK)

  メッセージ: isamchk

  使用されません。

- エラー: 1002 SQLSTATE: HY000 (ER_NO)

  メッセージ: NO

  その他のメッセージの構築に使われます。

- エラー: 1003 SQLSTATE: HY000 (ER_YES)

  メッセージ: YES

  その他のメッセージの構築に使われます。

  拡張された EXPLAIN 形式は注記メッセージを生成します。ER_YES は、後続の SHOW WARNINGS 出力にあるメッセージの Code カラムに使用されます。

- エラー: 1004 SQLSTATE: HY000 (ER_CANT_CREATE_FILE)

  メッセージ: ファイル '%s' を作成できません (エラー番号: %d - %s)

  サーバーがソーステーブルの .frm ファイルをコピー先テーブルの .frm ファイルにコピーしようとして、CREATE TABLE dst LIKE src ステートメントを実行したときに、新しい場所への .frm ファイルのコピーに失敗した場合に発生します。

  考えられる原因: ソース .frm ファイルのアクセス権の問題。コピー先 .frm ファイルがすでに存在するが、書き込みできない。

- エラー: 1005 SQLSTATE: HY000 (ER_CANT_CREATE_TABLE)

  メッセージ: 表 '%s' を作成できません (エラー番号: %d)

- エラー: 1006 SQLSTATE: HY000 (ER_CANT_CREATE_DB)

  メッセージ: データベース '%s' を作成できません (エラー番号: %d)

- エラー: 1007 SQLSTATE: HY000 (ER_DB_CREATE_EXISTS)

  メッセージ: データベース '%s' を作成できません。データベースはすでに存在します

  データベースがすでに存在するため、データベースの作成が失敗しました。

  既存のデータベースを置き換える場合は、最初にデータベースをドロップします。既存のデータベースを維持して、ステートメントでエラーが発生しないようにする場合は、CREATE DATABASE ステートメントに IF NOT EXISTS 句を追加します。

- エラー: 1008 SQLSTATE: HY000 (ER_DB_DROP_EXISTS)

  メッセージ: データベース '%s' を削除できません。データベースは存在しません

- エラー: 1009 SQLSTATE: HY000 (ER_DB_DROP_DELETE)

  メッセージ: データベース削除エラー ('%s' を削除できません。エラー番号: %d)

- エラー: 1010 SQLSTATE: HY000 (ER_DB_DROP_RMDIR)

  メッセージ: データベース削除エラー (ディレクトリ '%s' を削除できません。エラー番号: %d)

- エラー: 1011 SQLSTATE: HY000 (ER_CANT_DELETE_FILE)

  メッセージ: ファイル '%s' の削除エラー (エラー番号: %d - %s)

- エラー: 1012 SQLSTATE: HY000 (ER_CANT_FIND_SYSTEM_REC)

メッセージ: システムテーブルのレコードを読み取れません

InnoDB を使用できないときに、InnoDB の INFORMATION_SCHEMA テーブルにアクセスしようとすると、InnoDB によって返されます。

- エラー: 1013 SQLSTATE: HY000 (ER_CANT_GET_STAT)

  メッセージ: '%s' のステータスを取得できません (エラー番号: %d - %s)

- エラー: 1014 SQLSTATE: HY000 (ER_CANT_GET_WD)

  メッセージ: 作業ディレクトリを取得できません (エラー番号: %d - %s)

- エラー: 1015 SQLSTATE: HY000 (ER_CANT_LOCK)

  メッセージ: ファイルをロックできません (エラー番号: %d - %s)

- エラー: 1016 SQLSTATE: HY000 (ER_CANT_OPEN_FILE)

  メッセージ: ファイル '%s' をオープンできません (エラー番号: %d - %s)

- エラー: 1017 SQLSTATE: HY000 (ER_FILE_NOT_FOUND)

  メッセージ: ファイル '%s' が見つかりません (エラー番号: %d - %s)

- エラー: 1018 SQLSTATE: HY000 (ER_CANT_READ_DIR)

  メッセージ: ディレクトリ '%s' を読み取れません (エラー番号: %d - %s)

- エラー: 1019 SQLSTATE: HY000 (ER_CANT_SET_WD)

  メッセージ: ディレクトリ '%s' に移動できません (エラー番号: %d - %s)

- エラー: 1020 SQLSTATE: HY000 (ER_CHECKREAD)

  メッセージ: 表 '%s' の最後の読み取り時点から、レコードが変化しました

- エラー: 1021 SQLSTATE: HY000 (ER_DISK_FULL)

  メッセージ: ディスク領域不足です (%s)。領域が解放されるのを待機しています... (エラー番号: %d - %s)

- エラー: 1022 SQLSTATE: 23000 (ER_DUP_KEY)

  メッセージ: 書き込めません。表 '%s' に重複するキーがあります

- エラー: 1023 SQLSTATE: HY000 (ER_ERROR_ON_CLOSE)

  メッセージ: '%s' のクローズ時エラー (エラー番号: %d - %s)

- エラー: 1024 SQLSTATE: HY000 (ER_ERROR_ON_READ)

  メッセージ: ファイル '%s' の読み取りエラー (エラー番号: %d - %s)

- エラー: 1025 SQLSTATE: HY000 (ER_ERROR_ON_RENAME)

  メッセージ: '%s' の名前を '%s' に変更できません (エラー番号: %d - %s)

- エラー: 1026 SQLSTATE: HY000 (ER_ERROR_ON_WRITE)

  メッセージ: ファイル '%s' の書き込みエラー (エラー番号: %d - %s)

- エラー: 1027 SQLSTATE: HY000 (ER_FILE_USED)

  メッセージ: '%s' はロックされています

- エラー: 1028 SQLSTATE: HY000 (ER_FILSORT_ABORT)

  メッセージ: ソート処理を中断しました

- エラー: 1029 SQLSTATE: HY000 (ER_FORM_NOT_FOUND)

  メッセージ: ビュー '%s' は '%s' に存在しません

- エラー: 1030 SQLSTATE: HY000 (ER_GET_ERRNO)

  メッセージ: ストレージエンジンがエラー %d を返しました

  OS エラーの意味を確認するには、%d 値をチェックします。たとえば、28 はディスクに空き領域がないことを示します。

- エラー: 1031 SQLSTATE: HY000 (ER_ILLEGAL_HA)

  メッセージ: 表 '%s' のストレージエンジンでは提供されないオプションです

- エラー: 1032 SQLSTATE: HY000 (ER_KEY_NOT_FOUND)

  メッセージ: '%s' にレコードが見つかりません

- エラー: 1033 SQLSTATE: HY000 (ER_NOT_FORM_FILE)

  メッセージ: ファイル '%s' 内の情報が不正です

- エラー: 1034 SQLSTATE: HY000 (ER_NOT_KEYFILE)

  メッセージ: 表 '%s' のキーファイル (key file) の内容が不正です。修復を試行してください

- エラー: 1035 SQLSTATE: HY000 (ER_OLD_KEYFILE)

  メッセージ: 表 '%s' のキーファイル (key file) は古い形式です。修復してください。

- エラー: 1036 SQLSTATE: HY000 (ER_OPEN_AS_READONLY)

  メッセージ: 表 '%s' は読み取り専用です

- エラー: 1037 SQLSTATE: HY001 (ER_OUTOFMEMORY)

  メッセージ: メモリーが不足しています。サーバーを再起動してみてください (%d バイトの割り当てに失敗)

- エラー: 1038 SQLSTATE: HY001 (ER_OUT_OF_SORTMEMORY)

  メッセージ: ソートメモリーが不足しています。ソートバッファーサイズ (sort buffer size) の増加を検討してください

- エラー: 1039 SQLSTATE: HY000 (ER_UNEXPECTED_EOF)

  メッセージ: ファイル '%s' を読み取り中に予期せずファイルの終端に達しました (エラー番号: %d - %s)

- エラー: 1040 SQLSTATE: 08004 (ER_CON_COUNT_ERROR)

  メッセージ: 接続が多すぎます

- エラー: 1041 SQLSTATE: HY000 (ER_OUT_OF_RESOURCES)

  メッセージ: メモリーが不足しています。mysqld やその他のプロセスがメモリーを使い切っていないか確認してください。メモリーを使い切っていない場合、'ulimit' の設定などで mysqld のメモリー使用最大量を多くするか、スワップ領域を増やす必要があるかもしれません

- エラー: 1042 SQLSTATE: 08S01 (ER_BAD_HOST_ERROR)

  メッセージ: IP アドレスからホスト名を解決できません

- エラー: 1043 SQLSTATE: 08S01 (ER_HANDSHAKE_ERROR)

  メッセージ: ハンドシェイクエラー

- エラー: 1044 SQLSTATE: 42000 (ER_DBACCESS_DENIED_ERROR)

  メッセージ: ユーザー '%s'@'%s によるデータベース '%s' へのアクセスは拒否されました

- エラー: 1045 SQLSTATE: 28000 (ER_ACCESS_DENIED_ERROR)

  メッセージ: ユーザー '%s'@'%s' のアクセスは拒否されました (使用パスワード: %s)

- エラー: 1046 SQLSTATE: 3D000 (ER_NO_DB_ERROR)

メッセージ: データベースが選択されていません

- エラー: 1047 SQLSTATE: 08S01 (ER_UNKNOWN_COM_ERROR)

  メッセージ: 不明なコマンドです

- エラー: 1048 SQLSTATE: 23000 (ER_BAD_NULL_ERROR)

  メッセージ: カラム '%s' は null にできません

- エラー: 1049 SQLSTATE: 42000 (ER_BAD_DB_ERROR)

  メッセージ: '%s' は不明なデータベースです

- エラー: 1050 SQLSTATE: 42S01 (ER_TABLE_EXISTS_ERROR)

  メッセージ: 表 '%s' はすでに存在します

- エラー: 1051 SQLSTATE: 42S02 (ER_BAD_TABLE_ERROR)

  メッセージ: '%s' は不明な表です

- エラー: 1052 SQLSTATE: 23000 (ER_NON_UNIQ_ERROR)

  メッセージ: カラム '%s' は %s 内であいまいです

  ```
  %s = column name
  %s = location of column (for example, "field list")
  ```

  考えられる原因: クエリーに記述されたカラム (選択リスト内、ON 句内など) が適切に修飾されていません。

  例:

  ```
  mysql> SELECT i FROM t INNER JOIN t AS t2;
  ERROR 1052 (23000): Column 'i' in field list is ambiguous

  mysql> SELECT * FROM t LEFT JOIN t AS t2 ON i = i;
  ERROR 1052 (23000): Column 'i' in on clause is ambiguous
  ```

  解決方法:

  - カラムを該当するテーブル名で修飾します。

    ```
    mysql> SELECT t2.i FROM t INNER JOIN t AS t2;
    ```

  - 修飾が不要になるようにクエリーを変更します。

    ```
    mysql> SELECT * FROM t LEFT JOIN t AS t2 USING (i);
    ```

- エラー: 1053 SQLSTATE: 08S01 (ER_SERVER_SHUTDOWN)

  メッセージ: サーバーをシャットダウン中です

- エラー: 1054 SQLSTATE: 42S22 (ER_BAD_FIELD_ERROR)

  メッセージ: カラム '%s' は '%s' にはありません

- エラー: 1055 SQLSTATE: 42000 (ER_WRONG_FIELD_WITH_GROUP)

  メッセージ: '%s' は GROUP BY 句で指定されていません

- エラー: 1056 SQLSTATE: 42000 (ER_WRONG_GROUP_FIELD)

  メッセージ: '%s' でのグループ化はできません

- エラー: 1057 SQLSTATE: 42000 (ER_WRONG_SUM_SELECT)

  メッセージ: 集計関数と通常のカラムが同時に指定されています

- エラー: 1058 SQLSTATE: 21S01 (ER_WRONG_VALUE_COUNT)

  メッセージ: カラム数が値の個数と一致しません

- エラー: 1059 SQLSTATE: 42000 (ER_TOO_LONG_IDENT)

  メッセージ: 識別子名 '%s' は長すぎます

- エラー: 1060 SQLSTATE: 42S21 (ER_DUP_FIELDNAME)

  メッセージ: カラム名 '%s' は重複しています

- エラー: 1061 SQLSTATE: 42000 (ER_DUP_KEYNAME)

  メッセージ: キー名 '%s' は重複しています

- エラー: 1062 SQLSTATE: 23000 (ER_DUP_ENTRY)

  メッセージ: '%s' はキー %d で重複しています

  このエラーメッセージで返されるメッセージでは、ER_DUP_ENTRY_WITH_KEY_NAME の書式文字列が使用されます。

- エラー: 1063 SQLSTATE: 42000 (ER_WRONG_FIELD_SPEC)

  メッセージ: カラム '%s' のカラム指定子が不正です

- エラー: 1064 SQLSTATE: 42000 (ER_PARSE_ERROR)

  メッセージ: %s : '%s' 付近 %d 行目

- エラー: 1065 SQLSTATE: 42000 (ER_EMPTY_QUERY)

  メッセージ: クエリーが空です

- エラー: 1066 SQLSTATE: 42000 (ER_NONUNIQ_TABLE)

  メッセージ: 表名/エイリアス名 '%s' は一意ではありません

- エラー: 1067 SQLSTATE: 42000 (ER_INVALID_DEFAULT)

  メッセージ: '%s' へのデフォルト値が無効です

- エラー: 1068 SQLSTATE: 42000 (ER_MULTIPLE_PRI_KEY)

  メッセージ: PRIMARY KEY が複数定義されています

- エラー: 1069 SQLSTATE: 42000 (ER_TOO_MANY_KEYS)

  メッセージ: キーの数が多すぎます。最大 %d 個までです

- エラー: 1070 SQLSTATE: 42000 (ER_TOO_MANY_KEY_PARTS)

  メッセージ: 索引のキー列指定が多すぎます。最大 %d 個までです

- エラー: 1071 SQLSTATE: 42000 (ER_TOO_LONG_KEY)

  メッセージ: 索引のキーが長すぎます。最大 %d バイトまでです

- エラー: 1072 SQLSTATE: 42000 (ER_KEY_COLUMN_DOES_NOT_EXITS)

  メッセージ: キーカラム '%s' は表にありません

- エラー: 1073 SQLSTATE: 42000 (ER_BLOB_USED_AS_KEY)

  メッセージ: 指定されたテーブルタイプでは、BLOB カラム '%s' をキーに指定できません

- エラー: 1074 SQLSTATE: 42000 (ER_TOO_BIG_FIELDLENGTH)

  メッセージ: カラム '%s' のサイズ定義が大きすぎます (最大 %lu まで)。代わりに BLOB または TEXT を使用してください

- エラー: 1075 SQLSTATE: 42000 (ER_WRONG_AUTO_KEY)

  メッセージ: 不正な表定義です。AUTO_INCREMENT カラムは 1 個までで、キーを定義する必要があります

- エラー: 1076 SQLSTATE: HY000 (ER_READY)

メッセージ: %s: 接続準備完了。バージョン: '%s' ソケット: '%s' ポート: %d

- エラー: 1077 SQLSTATE: HY000 (ER_NORMAL_SHUTDOWN)

  メッセージ: %s: 通常シャットダウン

- エラー: 1078 SQLSTATE: HY000 (ER_GOT_SIGNAL)

  メッセージ: %s: シグナル %d を受信しました。強制終了します！

- エラー: 1079 SQLSTATE: HY000 (ER_SHUTDOWN_COMPLETE)

  メッセージ: %s: シャットダウン完了

- エラー: 1080 SQLSTATE: 08S01 (ER_FORCING_CLOSE)

  メッセージ: %s: スレッド %ld を強制終了します (ユーザー: '%s')

- エラー: 1081 SQLSTATE: 08S01 (ER_IPSOCK_ERROR)

  メッセージ: IP ソケットを作成できません

- エラー: 1082 SQLSTATE: 42S12 (ER_NO_SUCH_INDEX)

  メッセージ: 表 '%s' に以前 CREATE INDEX で作成されたインデックスがありません。表を作り直してください

- エラー: 1083 SQLSTATE: 42000 (ER_WRONG_FIELD_TERMINATORS)

  メッセージ: フィールド区切り文字引数が予期しない使われ方をしています。マニュアルを確認してください

- エラー: 1084 SQLSTATE: 42000 (ER_BLOBS_AND_NO_TERMINATED)

  メッセージ: BLOB には固定長レコードが使用できません。'FIELDS TERMINATED BY' 句を使用してください

- エラー: 1085 SQLSTATE: HY000 (ER_TEXTFILE_NOT_READABLE)

  メッセージ: ファイル '%s' はデータベースディレクトリにあるか、すべてのユーザーから読める必要があります

- エラー: 1086 SQLSTATE: HY000 (ER_FILE_EXISTS_ERROR)

  メッセージ: ファイル '%s' はすでに存在します

- エラー: 1087 SQLSTATE: HY000 (ER_LOAD_INFO)

  メッセージ: レコード数: %ld 削除: %ld スキップ: %ld 警告: %ld

- エラー: 1088 SQLSTATE: HY000 (ER_ALTER_INFO)

  メッセージ: レコード数: %ld 重複: %ld

- エラー: 1089 SQLSTATE: HY000 (ER_WRONG_SUB_KEY)

  メッセージ: キーのプレフィックスが不正です。キーが文字列ではないか、プレフィックス長がキーよりも長いか、ストレージエンジンが一意索引のプレフィックス指定をサポートしていません

- エラー: 1090 SQLSTATE: 42000 (ER_CANT_REMOVE_ALL_FIELDS)

  メッセージ: ALTER TABLE ではすべてのカラムの削除はできません。DROP TABLE を使用してください

- エラー: 1091 SQLSTATE: 42000 (ER_CANT_DROP_FIELD_OR_KEY)

  メッセージ: '%s' を削除できません。カラム/キーの存在を確認してください

- エラー: 1092 SQLSTATE: HY000 (ER_INSERT_INFO)

  メッセージ: レコード数: %ld 重複数: %l 警告: %ld

- エラー: 1093 SQLSTATE: HY000 (ER_UPDATE_TABLE_USED)

  メッセージ: FROM 句にある表 '%s' は UPDATE の対象にできません

- エラー: 1094 SQLSTATE: HY000 (ER_NO_SUCH_THREAD)

  メッセージ: 不明なスレッド ID です: %lu

- エラー: 1095 SQLSTATE: HY000 (ER_KILL_DENIED_ERROR)

  メッセージ: スレッド %lu のオーナーではありません

- エラー: 1096 SQLSTATE: HY000 (ER_NO_TABLES_USED)

  メッセージ: 表が指定されていません

- エラー: 1097 SQLSTATE: HY000 (ER_TOO_BIG_SET)

  メッセージ: SET 型のカラム '%s' の文字列の数が多すぎます

- エラー: 1098 SQLSTATE: HY000 (ER_NO_UNIQUE_LOGFILE)

  メッセージ: 一意なログファイル名 %s.(1-999) を生成できません

- エラー: 1099 SQLSTATE: HY000 (ER_TABLE_NOT_LOCKED_FOR_WRITE)

  メッセージ: 表 '%s' は READ ロックされていて、更新できません

- エラー: 1100 SQLSTATE: HY000 (ER_TABLE_NOT_LOCKED)

  メッセージ: 表 '%s' は LOCK TABLES でロックされていません

- エラー: 1101 SQLSTATE: 42000 (ER_BLOB_CANT_HAVE_DEFAULT)

  メッセージ: BLOB/TEXT カラム '%s' にはデフォルト値を指定できません

- エラー: 1102 SQLSTATE: 42000 (ER_WRONG_DB_NAME)

  メッセージ: データベース名 '%s' は不正です

- エラー: 1103 SQLSTATE: 42000 (ER_WRONG_TABLE_NAME)

  メッセージ: 表名 '%s' は不正です

- エラー: 1104 SQLSTATE: 42000 (ER_TOO_BIG_SELECT)

  メッセージ: SELECT が MAX_JOIN_SIZE を超える行数を処理しました。WHERE 句を確認し、SELECT 文に問題がなければ、SET SQL_BIG_SELECTS=1 または SET MAX_JOIN_SIZE=# を使用してください

- エラー: 1105 SQLSTATE: HY000 (ER_UNKNOWN_ERROR)

  メッセージ: 不明なエラー

- エラー: 1106 SQLSTATE: 42000 (ER_UNKNOWN_PROCEDURE)

  メッセージ: '%s' は不明なプロシージャーです

- エラー: 1107 SQLSTATE: 42000 (ER_WRONG_PARAMCOUNT_TO_PROCEDURE)

  メッセージ: プロシージャー '%s' へのパラメータ数が不正です

- エラー: 1108 SQLSTATE: HY000 (ER_WRONG_PARAMETERS_TO_PROCEDURE)

  メッセージ: プロシージャー '%s' へのパラメータが不正です

- エラー: 1109 SQLSTATE: 42S02 (ER_UNKNOWN_TABLE)

  メッセージ: '%s' は %s では不明な表です

- エラー: 1110 SQLSTATE: 42000 (ER_FIELD_SPECIFIED_TWICE)

  メッセージ: カラム '%s' は 2 回指定されています

- エラー: 1111 SQLSTATE: HY000 (ER_INVALID_GROUP_FUNC_USE)

  メッセージ: 集計関数の使用方法が不正です

- エラー: 1112 SQLSTATE: 42000 (ER_UNSUPPORTED_EXTENSION)

  メッセージ: 表 '%s' は、この MySQL バージョンには無い機能を使用しています

- エラー: 1113 SQLSTATE: 42000 (ER_TABLE_MUST_HAVE_COLUMNS)

  メッセージ: 表には最低でも 1 個のカラムが必要です

- エラー: 1114 SQLSTATE: HY000 (ER_RECORD_FILE_FULL)

  メッセージ: 表 '%s' は満杯です。

- エラー: 1115 SQLSTATE: 42000 (ER_UNKNOWN_CHARACTER_SET)

  メッセージ: 不明な文字コードセット: '%s'

- エラー: 1116 SQLSTATE: HY000 (ER_TOO_MANY_TABLES)

  メッセージ: 表が多すぎます。MySQL が JOIN できる表は %d 個までです。

- エラー: 1117 SQLSTATE: HY000 (ER_TOO_MANY_FIELDS)

  メッセージ: カラムが多すぎます。

- エラー: 1118 SQLSTATE: 42000 (ER_TOO_BIG_ROWSIZE)

  メッセージ: 行サイズが大きすぎます。この表の最大行サイズは BLOB を含まずに %ld です。格納時のオーバーヘッドも含まれます (マニュアルを確認してください)。カラムを TEXT または BLOB に変更する必要があります。

- エラー: 1119 SQLSTATE: HY000 (ER_STACK_OVERRUN)

  メッセージ: スレッドスタック不足です (使用: %ld ; サイズ: %ld)。必要に応じて、より大きい値で 'mysqld --thread_stack=#' の指定をしてください。

- エラー: 1120 SQLSTATE: 42000 (ER_WRONG_OUTER_JOIN)

  メッセージ: OUTER JOIN に相互依存が見つかりました。ON 句の条件を確認してください。

- エラー: 1121 SQLSTATE: 42000 (ER_NULL_COLUMN_IN_INDEX)

  メッセージ: テーブルハンドラは、指定されたインデックスで NULL をサポートしていません。カラム '%s' を NOT NULL に変更するか、別のハンドラを使用してください

- エラー: 1122 SQLSTATE: HY000 (ER_CANT_FIND_UDF)

  メッセージ: 関数 '%s' をロードできません。

- エラー: 1123 SQLSTATE: HY000 (ER_CANT_INITIALIZE_UDF)

  メッセージ: 関数 '%s' を初期化できません。%s

- エラー: 1124 SQLSTATE: HY000 (ER_UDF_NO_PATHS)

  メッセージ: 共有ライブラリにはパスを指定できません。

- エラー: 1125 SQLSTATE: HY000 (ER_UDF_EXISTS)

  メッセージ: 関数 '%s' はすでに定義されています。

- エラー: 1126 SQLSTATE: HY000 (ER_CANT_OPEN_LIBRARY)

  メッセージ: 共有ライブラリ '%s' を開く事ができません。(エラー番号: %d %s)

- エラー: 1127 SQLSTATE: HY000 (ER_CANT_FIND_DL_ENTRY)

  メッセージ: 関数 '%s' は共有ライブラリ中にありません

- エラー: 1128 SQLSTATE: HY000 (ER_FUNCTION_NOT_DEFINED)

  メッセージ: 関数 '%s' は定義されていません。

- エラー: 1129 SQLSTATE: HY000 (ER_HOST_IS_BLOCKED)

  メッセージ: 接続エラーが多いため、ホスト '%s' は拒否されました。'mysqladmin flush-hosts' で解除できます。

- エラー: 1130 SQLSTATE: HY000 (ER_HOST_NOT_PRIVILEGED)

  メッセージ: ホスト '%s' からのこの MySQL Server への接続は許可されていません

- エラー: 1131 SQLSTATE: 42000 (ER_PASSWORD_ANONYMOUS_USER)

  メッセージ: MySQL を匿名ユーザーで使用しているので、パスワードの変更はできません。

- エラー: 1132 SQLSTATE: 42000 (ER_PASSWORD_NOT_ALLOWED)

  メッセージ: 他のユーザーのパスワードを変更するためには、mysql データベースの表を更新する権限が必要です。

- エラー: 1133 SQLSTATE: 42000 (ER_PASSWORD_NO_MATCH)

  メッセージ: ユーザーテーブルに該当するレコードが見つかりません。

- エラー: 1134 SQLSTATE: HY000 (ER_UPDATE_INFO)

  メッセージ: 該当した行: %ld 変更: %ld 警告: %ld

- エラー: 1135 SQLSTATE: HY000 (ER_CANT_CREATE_THREAD)

  メッセージ: 新規にスレッドを作成できません。(エラー番号 %d) もしも使用可能メモリーの不足でなければ、OS 依存のバグである可能性があります。

- エラー: 1136 SQLSTATE: 21S01 (ER_WRONG_VALUE_COUNT_ON_ROW)

  メッセージ: %ld 行目で、カラムの数が値の数と一致しません。

- エラー: 1137 SQLSTATE: HY000 (ER_CANT_REOPEN_TABLE)

  メッセージ: 表を再オープンできません。: '%s'

- エラー: 1138 SQLSTATE: 22004 (ER_INVALID_USE_OF_NULL)

  メッセージ: NULL 値の使用方法が不適切です。

- エラー: 1139 SQLSTATE: 42000 (ER_REGEXP_ERROR)

  メッセージ: regexp がエラー '%s' を返しました。

- エラー: 1140 SQLSTATE: 42000 (ER_MIX_OF_GROUP_FUNC_AND_FIELDS)

  メッセージ: GROUP BY 句が無い場合、集計関数 (MIN(),MAX(),COUNT(),...) と通常のカラムを同時に使用できません。

- エラー: 1141 SQLSTATE: 42000 (ER_NONEXISTING_GRANT)

  メッセージ: ユーザー '%s' (ホスト '%s' 上) は許可されていません。

- エラー: 1142 SQLSTATE: 42000 (ER_TABLEACCESS_DENIED_ERROR)

  メッセージ: コマンド %s はユーザー '%s'@'%s' の表 '%s' の使用に関して許可されていません。

- エラー: 1143 SQLSTATE: 42000 (ER_COLUMNACCESS_DENIED_ERROR)

  メッセージ: コマンド %s はユーザー '%s'@'%s' のカラム '%s'(表 '%s') の利用に関して許可されていません。

- エラー: 1144 SQLSTATE: 42000 (ER_ILLEGAL_GRANT_FOR_TABLE)

  メッセージ: 不正な GRANT/REVOKE コマンドです。どの権限で利用可能かはマニュアルを参照してください。

- エラー: 1145 SQLSTATE: 42000 (ER_GRANT_WRONG_HOST_OR_USER)

  メッセージ: GRANT コマンドへの、ホスト名やユーザー名が長すぎます。

- エラー: 1146 SQLSTATE: 42S02 (ER_NO_SUCH_TABLE)

  メッセージ: 表 '%s.%s' は存在しません。

- エラー: 1147 SQLSTATE: 42000 (ER_NONEXISTING_TABLE_GRANT)

  メッセージ: ユーザー '%s' (ホスト '%s' 上) の表 '%s' への権限は定義されていません。

- エラー: 1148 SQLSTATE: 42000 (ER_NOT_ALLOWED_COMMAND)

  メッセージ: この MySQL バージョンでは利用できないコマンドです。

- エラー: 1149 SQLSTATE: 42000 (ER_SYNTAX_ERROR)

  メッセージ: SQL 構文エラーです。バージョンに対応するマニュアルを参照して正しい構文を確認してください。

- エラー: 1150 SQLSTATE: HY000 (ER_DELAYED_CANT_CHANGE_LOCK)

  メッセージ: 'Delayed insert' スレッドが表 '%s' のロックを取得できませんでした。

- エラー: 1151 SQLSTATE: HY000 (ER_TOO_MANY_DELAYED_THREADS)

  メッセージ: 'Delayed insert' スレッドが多すぎます。

- エラー: 1152 SQLSTATE: 08S01 (ER_ABORTING_CONNECTION)

  メッセージ: 接続 %ld が中断されました。データベース: '%s' ユーザー: '%s' (%s)

- エラー: 1153 SQLSTATE: 08S01 (ER_NET_PACKET_TOO_LARGE)

  メッセージ: 'max_allowed_packet' よりも大きなパケットを受信しました。

- エラー: 1154 SQLSTATE: 08S01 (ER_NET_READ_ERROR_FROM_PIPE)

  メッセージ: 接続パイプの読み取りエラーです。

- エラー: 1155 SQLSTATE: 08S01 (ER_NET_FCNTL_ERROR)

  メッセージ: fcntl() がエラーを返しました。

- エラー: 1156 SQLSTATE: 08S01 (ER_NET_PACKETS_OUT_OF_ORDER)

  メッセージ: 不正な順序のパケットを受信しました。

- エラー: 1157 SQLSTATE: 08S01 (ER_NET_UNCOMPRESS_ERROR)

  メッセージ: 圧縮パケットの展開ができませんでした。

- エラー: 1158 SQLSTATE: 08S01 (ER_NET_READ_ERROR)

  メッセージ: パケットの受信でエラーが発生しました。

- エラー: 1159 SQLSTATE: 08S01 (ER_NET_READ_INTERRUPTED)

  メッセージ: パケットの受信でタイムアウトが発生しました。

- エラー: 1160 SQLSTATE: 08S01 (ER_NET_ERROR_ON_WRITE)

  メッセージ: パケットの送信でエラーが発生しました。

- エラー: 1161 SQLSTATE: 08S01 (ER_NET_WRITE_INTERRUPTED)

  メッセージ: パケットの送信でタイムアウトが発生しました。

- エラー: 1162 SQLSTATE: 42000 (ER_TOO_LONG_STRING)

  メッセージ: 結果の文字列が 'max_allowed_packet' よりも大きいです。

- エラー: 1163 SQLSTATE: 42000 (ER_TABLE_CANT_HANDLE_BLOB)

  メッセージ: 指定されたテーブルタイプでは、BLOB/TEXT 型のカラムを使用できません。

- エラー: 1164 SQLSTATE: 42000 (ER_TABLE_CANT_HANDLE_AUTO_INCREMENT)

  メッセージ: 指定されたテーブルタイプでは、AUTO_INCREMENT のカラムを使用できません。

- エラー: 1165 SQLSTATE: HY000 (ER_DELAYED_INSERT_TABLE_LOCKED)

  メッセージ: 表 '%s' は LOCK TABLES でロックされているため、INSERT DELAYED を使用できません。

- エラー: 1166 SQLSTATE: 42000 (ER_WRONG_COLUMN_NAME)

  メッセージ: カラム名 '%s' は不正です。

- エラー: 1167 SQLSTATE: 42000 (ER_WRONG_KEY_COLUMN)

  メッセージ: 使用のストレージエンジンはカラム '%s' のインデックスを作成できません。

- エラー: 1168 SQLSTATE: HY000 (ER_WRONG_MRG_TABLE)

  メッセージ: MERGE 表の構成表がオープンできません。列定義が異なるか、MyISAM 表ではないか、存在しません。

- エラー: 1169 SQLSTATE: 23000 (ER_DUP_UNIQUE)

  メッセージ: 一意制約違反のため、表 '%s' に書き込めません。

- エラー: 1170 SQLSTATE: 42000 (ER_BLOB_KEY_WITHOUT_LENGTH)

  メッセージ: BLOB/TEXT カラム '%s' をキーに使用するには長さ指定が必要です

- エラー: 1171 SQLSTATE: 42000 (ER_PRIMARY_CANT_HAVE_NULL)

  メッセージ: PRIMARY KEY の列はすべて NOT NULL でなければいけません。UNIQUE インデックスであれば NULL を含むことが可能です。

- エラー: 1172 SQLSTATE: 42000 (ER_TOO_MANY_ROWS)

  メッセージ: 結果が 2 行以上です。

- エラー: 1173 SQLSTATE: 42000 (ER_REQUIRES_PRIMARY_KEY)

  メッセージ: 使用のテーブルタイプでは、主キーが必要です

- エラー: 1174 SQLSTATE: HY000 (ER_NO_RAID_COMPILED)

  メッセージ: このバージョンの MySQL は RAID サポートを含めてコンパイルされていません。

- エラー: 1175 SQLSTATE: HY000 (ER_UPDATE_WITHOUT_KEY_IN_SAFE_MODE)

  メッセージ: 'safe update mode' で、索引を利用する WHERE 句の無い更新処理を実行しようとしました。

- エラー: 1176 SQLSTATE: 42000 (ER_KEY_DOES_NOT_EXITS)

  メッセージ: キー '%s' は表 '%s' には存在しません。

- エラー: 1177 SQLSTATE: 42000 (ER_CHECK_NO_SUCH_TABLE)

  メッセージ: 表をオープンできません。

- エラー: 1178 SQLSTATE: 42000 (ER_CHECK_NOT_IMPLEMENTED)

  メッセージ: この表のストレージエンジンは '%s' を利用できません。

- エラー: 1179 SQLSTATE: 25000 (ER_CANT_DO_THIS_DURING_AN_TRANSACTION)

  メッセージ: このコマンドはトランザクション内で実行できません。

- エラー: 1180 SQLSTATE: HY000 (ER_ERROR_DURING_COMMIT)

  メッセージ: COMMIT 中にエラー %d が発生しました。

- エラー: 1181 SQLSTATE: HY000 (ER_ERROR_DURING_ROLLBACK)

  メッセージ: ROLLBACK 中にエラー %d が発生しました。

- エラー: 1182 SQLSTATE: HY000 (ER_ERROR_DURING_FLUSH_LOGS)

  メッセージ: FLUSH_LOGS 中にエラー %d が発生しました。

- エラー: 1183 SQLSTATE: HY000 (ER_ERROR_DURING_CHECKPOINT)

  メッセージ: CHECKPOINT 中にエラー %d が発生しました。

- エラー: 1184 SQLSTATE: 08S01 (ER_NEW_ABORTING_CONNECTION)

  メッセージ: 接続 %ld が中断されました。データベース: '%s' ユーザー: '%s' ホスト: '%s' (%s)

- エラー: 1185 SQLSTATE: HY000 (ER_DUMP_NOT_IMPLEMENTED)

  メッセージ: この表のストレージエンジンはバイナリ形式の表ダンプを利用できません。

- エラー: 1186 SQLSTATE: HY000 (ER_FLUSH_MASTER_BINLOG_CLOSED)

  メッセージ: バイナリログがクローズされています。RESET MASTER を実行できません。

- エラー: 1187 SQLSTATE: HY000 (ER_INDEX_REBUILD)

  メッセージ: ダンプ表 '%s' のインデックス再構築に失敗しました。

- エラー: 1188 SQLSTATE: HY000 (ER_MASTER)

  メッセージ: マスターでエラーが発生: '%s'

- エラー: 1189 SQLSTATE: 08S01 (ER_MASTER_NET_READ)

  メッセージ: マスターからのデータ受信中のネットワークエラー

- エラー: 1190 SQLSTATE: 08S01 (ER_MASTER_NET_WRITE)

  メッセージ: マスターへのデータ送信中のネットワークエラー

- エラー: 1191 SQLSTATE: HY000 (ER_FT_MATCHING_KEY_NOT_FOUND)

  メッセージ: カラムリストに対応する全文インデックス (FULLTEXT) が見つかりません。

- エラー: 1192 SQLSTATE: HY000 (ER_LOCK_OR_ACTIVE_TRANSACTION)

  メッセージ: すでにアクティブな表ロックやトランザクションがあるため、コマンドを実行できません。

- エラー: 1193 SQLSTATE: HY000 (ER_UNKNOWN_SYSTEM_VARIABLE)

  メッセージ: '%s' は不明なシステム変数です。

- エラー: 1194 SQLSTATE: HY000 (ER_CRASHED_ON_USAGE)

  メッセージ: 表 '%s' は壊れています。修復が必要です。

- エラー: 1195 SQLSTATE: HY000 (ER_CRASHED_ON_REPAIR)

  メッセージ: 表 '%s' は壊れています。修復 (自動 ?) にも失敗しています。

- エラー: 1196 SQLSTATE: HY000 (ER_WARNING_NOT_COMPLETE_ROLLBACK)

  メッセージ: トランザクション対応ではない表への変更はロールバックされません。

- エラー: 1197 SQLSTATE: HY000 (ER_TRANS_CACHE_FULL)

  メッセージ: 複数ステートメントから成るトランザクションが 'max_binlog_cache_size' 以上の容量を必要としました。このシステム変数を増加して、再試行してください。

- エラー: 1198 SQLSTATE: HY000 (ER_SLAVE_MUST_STOP)

  メッセージ: この処理は、稼働中のスレーブでは実行できません。あらかじめ STOP SLAVE コマンドを実行してください。

- エラー: 1199 SQLSTATE: HY000 (ER_SLAVE_NOT_RUNNING)

メッセージ: この処理は、稼働中のスレーブでなければ実行できません。スレーブの構成をして START SLAVE コマンドを実行してください。

- エラー: 1200 SQLSTATE: HY000 (ER_BAD_SLAVE)

  メッセージ: このサーバーはスレーブとして構成されていません。コンフィグファイルか CHANGE MASTER TO コマンドで設定してください。

- エラー: 1201 SQLSTATE: HY000 (ER_MASTER_INFO)

  メッセージ: 'master info' 構造体の初期化ができませんでした。MySQL エラーログでエラーメッセージを確認してください。

- エラー: 1202 SQLSTATE: HY000 (ER_SLAVE_THREAD)

  メッセージ: スレーブスレッドを作成できません。システムリソースを確認してください。

- エラー: 1203 SQLSTATE: 42000 (ER_TOO_MANY_USER_CONNECTIONS)

  メッセージ: ユーザー '%s' はすでに 'max_user_connections' 以上のアクティブな接続を行っています。

- エラー: 1204 SQLSTATE: HY000 (ER_SET_CONSTANTS_ONLY)

  メッセージ: SET 処理が失敗しました。

- エラー: 1205 SQLSTATE: HY000 (ER_LOCK_WAIT_TIMEOUT)

  メッセージ: ロック待ちがタイムアウトしました。トランザクションを再試行してください。

- エラー: 1206 SQLSTATE: HY000 (ER_LOCK_TABLE_FULL)

  メッセージ: ロックの数が多すぎます。

- エラー: 1207 SQLSTATE: 25000 (ER_READ_ONLY_TRANSACTION)

  メッセージ: 読み取り専用トランザクションです。

- エラー: 1208 SQLSTATE: HY000 (ER_DROP_DB_WITH_READ_LOCK)

  メッセージ: グローバルリードロックを保持している間は、DROP DATABASE を実行できません。

- エラー: 1209 SQLSTATE: HY000 (ER_CREATE_DB_WITH_READ_LOCK)

  メッセージ: グローバルリードロックを保持している間は、CREATE DATABASE を実行できません。

- エラー: 1210 SQLSTATE: HY000 (ER_WRONG_ARGUMENTS)

  メッセージ: %s の引数が不正です

- エラー: 1211 SQLSTATE: 42000 (ER_NO_PERMISSION_TO_CREATE_USER)

  メッセージ: '%s'@'%s' は新しいユーザーを作成できません。

- エラー: 1212 SQLSTATE: HY000 (ER_UNION_TABLES_IN_DIFFERENT_DIR)

  メッセージ: 不正な表定義です。MERGE 表の構成表はすべて同じデータベース内になければなりません。

- エラー: 1213 SQLSTATE: 40001 (ER_LOCK_DEADLOCK)

  メッセージ: ロック取得中にデッドロックが検出されました。トランザクションを再試行してください。

- エラー: 1214 SQLSTATE: HY000 (ER_TABLE_CANT_HANDLE_FT)

  メッセージ: 使用の表は全文インデックスを利用できません。

- エラー: 1215 SQLSTATE: HY000 (ER_CANNOT_ADD_FOREIGN)

  メッセージ: 外部キー制約を追加できません。

- エラー: 1216 SQLSTATE: 23000 (ER_NO_REFERENCED_ROW)

  メッセージ: 親キーがありません。外部キー制約違反です。

- エラー: 1217 SQLSTATE: 23000 (ER_ROW_IS_REFERENCED)

  メッセージ: 子レコードがあります。外部キー制約違反です。

- エラー: 1218 SQLSTATE: 08S01 (ER_CONNECT_TO_MASTER)

  メッセージ: マスターへの接続エラー: %s

- エラー: 1219 SQLSTATE: HY000 (ER_QUERY_ON_MASTER)

  メッセージ: マスターでのクエリー実行エラー: %s

- エラー: 1220 SQLSTATE: HY000 (ER_ERROR_WHEN_EXECUTING_COMMAND)

  メッセージ: %s コマンドの実行エラー: %s

- エラー: 1221 SQLSTATE: HY000 (ER_WRONG_USAGE)

  メッセージ: %s の %s に関する不正な使用法です。

- エラー: 1222 SQLSTATE: 21000 (ER_WRONG_NUMBER_OF_COLUMNS_IN_SELECT)

  メッセージ: 使用の SELECT 文が返すカラム数が違います。

- エラー: 1223 SQLSTATE: HY000 (ER_CANT_UPDATE_WITH_READLOCK)

  メッセージ: 競合するリードロックを保持しているので、クエリーを実行できません。

- エラー: 1224 SQLSTATE: HY000 (ER_MIXING_NOT_ALLOWED)

  メッセージ: トランザクション対応の表と非対応の表の同時使用は無効化されています。

- エラー: 1225 SQLSTATE: HY000 (ER_DUP_ARGUMENT)

  メッセージ: オプション '%s' が 2 度使用されています。

- エラー: 1226 SQLSTATE: 42000 (ER_USER_LIMIT_REACHED)

  メッセージ: ユーザー '%s' はリソースの上限 '%s' に達しました。(現在値: %ld)

- エラー: 1227 SQLSTATE: 42000 (ER_SPECIFIC_ACCESS_DENIED_ERROR)

  メッセージ: アクセスは拒否されました。この操作には %s 権限が (複数の場合はどれか 1 つ) 必要です。

- エラー: 1228 SQLSTATE: HY000 (ER_LOCAL_VARIABLE)

  メッセージ: 変数 '%s' はセッション変数です。SET GLOBAL では使用できません。

- エラー: 1229 SQLSTATE: HY000 (ER_GLOBAL_VARIABLE)

  メッセージ: 変数 '%s' はグローバル変数です。SET GLOBAL を使用してください。

- エラー: 1230 SQLSTATE: 42000 (ER_NO_DEFAULT)

  メッセージ: 変数 '%s' にはデフォルト値がありません。

- エラー: 1231 SQLSTATE: 42000 (ER_WRONG_VALUE_FOR_VAR)

  メッセージ: 変数 '%s' に値 '%s' を設定できません。

- エラー: 1232 SQLSTATE: 42000 (ER_WRONG_TYPE_FOR_VAR)

  メッセージ: 変数 '%s' への引数の型が不正です

- エラー: 1233 SQLSTATE: HY000 (ER_VAR_CANT_BE_READ)

  メッセージ: 変数 '%s' は書き込み専用です。読み取りはできません。

- エラー: 1234 SQLSTATE: 42000 (ER_CANT_USE_OPTION_HERE)

  メッセージ: '%s' の使用法または場所が不正です。

- エラー: 1235 SQLSTATE: 42000 (ER_NOT_SUPPORTED_YET)

  メッセージ: このバージョンの MySQL では、まだ '%s' を利用できません。

- エラー: 1236 SQLSTATE: HY000 (ER_MASTER_FATAL_ERROR_READING_BINLOG)

  メッセージ: 致命的なエラー %d: '%s' がマスターでバイナリログ読み取り中に発生しました。

- エラー: 1237 SQLSTATE: HY000 (ER_SLAVE_IGNORED_TABLE)

  メッセージ: replicate-*-table ルールに従って、スレーブ SQL スレッドはクエリーを無視しました。

- エラー: 1238 SQLSTATE: HY000 (ER_INCORRECT_GLOBAL_LOCAL_VAR)

  メッセージ: 変数 '%s' は %s 変数です。

- エラー: 1239 SQLSTATE: 42000 (ER_WRONG_FK_DEF)

  メッセージ: 外部キー '%s' の定義の不正: %s

- エラー: 1240 SQLSTATE: HY000 (ER_KEY_REF_DO_NOT_MATCH_TABLE_REF)

  メッセージ: 外部キーの参照表と定義が一致しません。

- エラー: 1241 SQLSTATE: 21000 (ER_OPERAND_COLUMNS)

  メッセージ: オペランドに %d 個のカラムが必要です。

- エラー: 1242 SQLSTATE: 21000 (ER_SUBQUERY_NO_1_ROW)

  メッセージ: サブクエリーが 2 行以上の結果を返します。

- エラー: 1243 SQLSTATE: HY000 (ER_UNKNOWN_STMT_HANDLER)

  メッセージ: '%.*s' はプリペアドステートメントの不明なハンドルです (%s で指定されました)

- エラー: 1244 SQLSTATE: HY000 (ER_CORRUPT_HELP_DB)

  メッセージ: ヘルプデータベースは壊れているか存在しません。

- エラー: 1245 SQLSTATE: HY000 (ER_CYCLIC_REFERENCE)

  メッセージ: サブクエリーの参照がループしています。

- エラー: 1246 SQLSTATE: HY000 (ER_AUTO_CONVERT)

  メッセージ: カラム '%s' を %s から %s へ変換します。

- エラー: 1247 SQLSTATE: 42S22 (ER_ILLEGAL_REFERENCE)

  メッセージ: '%s' の参照はできません。(%s)

- エラー: 1248 SQLSTATE: 42000 (ER_DERIVED_MUST_HAVE_ALIAS)

  メッセージ: 導出表にはエイリアスが必須です。

- エラー: 1249 SQLSTATE: 01000 (ER_SELECT_REDUCED)

  メッセージ: Select %u は最適化によって減らされました。

- エラー: 1250 SQLSTATE: 42000 (ER_TABLENAME_NOT_ALLOWED_HERE)

  メッセージ: 特定の SELECT のみで使用の表 '%s' は %s では使用できません。

- エラー: 1251 SQLSTATE: 08004 (ER_NOT_SUPPORTED_AUTH_MODE)

  メッセージ: クライアントはサーバーが要求する認証プロトコルに対応できません。MySQL クライアントのアップグレードを検討してください。

- エラー: 1252 SQLSTATE: 42000 (ER_SPATIAL_CANT_HAVE_NULL)

  メッセージ: SPATIAL インデックスのキー列は NOT NULL でなければいけません

- エラー: 1253 SQLSTATE: 42000 (ER_COLLATION_CHARSET_MISMATCH)

  メッセージ: COLLATION '%s' は CHARACTER SET '%s' に適用できません。

- エラー: 1254 SQLSTATE: HY000 (ER_SLAVE_WAS_RUNNING)

  メッセージ: スレーブはすでに稼働中です。

- エラー: 1255 SQLSTATE: HY000 (ER_SLAVE_WAS_NOT_RUNNING)

  メッセージ: スレーブはすでに停止しています。

- エラー: 1256 SQLSTATE: HY000 (ER_TOO_BIG_FOR_UNCOMPRESS)

  メッセージ: 展開後のデータが大きすぎます。最大サイズは %d です。(展開後データの長さ情報が壊れている可能性もあります。)

- エラー: 1257 SQLSTATE: HY000 (ER_ZLIB_Z_MEM_ERROR)

  メッセージ: ZLIB: メモリー不足です。

- エラー: 1258 SQLSTATE: HY000 (ER_ZLIB_Z_BUF_ERROR)

  メッセージ: ZLIB: 出力バッファーに十分な空きがありません。(展開後データの長さ情報が壊れている可能性もあります。)

- エラー: 1259 SQLSTATE: HY000 (ER_ZLIB_Z_DATA_ERROR)

  メッセージ: ZLIB: 入力データが壊れています。

- エラー: 1260 SQLSTATE: HY000 (ER_CUT_VALUE_GROUP_CONCAT)

  メッセージ: 行 %u は GROUP_CONCAT() によって切り捨てられました

- エラー: 1261 SQLSTATE: 01000 (ER_WARN_TOO_FEW_RECORDS)

  メッセージ: 行 %ld はすべてのカラムへのデータを含んでいません。

- エラー: 1262 SQLSTATE: 01000 (ER_WARN_TOO_MANY_RECORDS)

  メッセージ: 行 %ld はデータを切り捨てられました。カラムよりも多いデータを含んでいました。

- エラー: 1263 SQLSTATE: 22004 (ER_WARN_NULL_TO_NOTNULL)

  メッセージ: カラムにデフォルト値が設定されました。NOT NULL のカラム '%s' に行 %ld で NULL が与えられました。

- エラー: 1264 SQLSTATE: 22003 (ER_WARN_DATA_OUT_OF_RANGE)

  メッセージ: カラム '%s' 行 %ld の値が範囲外です

- エラー: 1265 SQLSTATE: 01000 (WARN_DATA_TRUNCATED)

  メッセージ: カラム '%s' の行 %ld でデータが切り捨てられました。

- エラー: 1266 SQLSTATE: HY000 (ER_WARN_USING_OTHER_HANDLER)

  メッセージ: ストレージエンジン %s が表 '%s' に利用されています。

- エラー: 1267 SQLSTATE: HY000 (ER_CANT_AGGREGATE_2COLLATIONS)

  メッセージ: 照合順序 (%s,%s) と (%s,%s) の混在は操作 '%s' では不正です。

- エラー: 1268 SQLSTATE: HY000 (ER_DROP_USER)

  メッセージ: 要求された 1 つ以上のユーザーをドロップできません

- エラー: 1269 SQLSTATE: HY000 (ER_REVOKE_GRANTS)

  メッセージ: 指定されたユーザーから指定されたすべての権限を剥奪することができませんでした。

- エラー: 1270 SQLSTATE: HY000 (ER_CANT_AGGREGATE_3COLLATIONS)

メッセージ: 照合順序 (%s,%s), (%s,%s), (%s,%s) の混在は操作 '%s' では不正です。

- エラー: 1271 SQLSTATE: HY000 (ER_CANT_AGGREGATE_NCOLLATIONS)

  メッセージ: 操作 '%s' では不正な照合順序の混在です。

- エラー: 1272 SQLSTATE: HY000 (ER_VARIABLE_IS_NOT_STRUCT)

  メッセージ: 変数 '%s' は構造変数の構成要素ではありません。(XXXX.変数名という指定はできません。)

- エラー: 1273 SQLSTATE: HY000 (ER_UNKNOWN_COLLATION)

  メッセージ: 不明な照合順序: '%s'

- エラー: 1274 SQLSTATE: HY000 (ER_SLAVE_IGNORED_SSL_PARAMS)

  メッセージ: この MySQL スレーブは SSL サポートを含めてコンパイルされていないので、CHANGE MASTER の SSL パラメータは無視されました。今後 SSL サポートを持つ MySQL スレーブを起動する際に利用されます。

- エラー: 1275 SQLSTATE: HY000 (ER_SERVER_IS_IN_SECURE_AUTH_MODE)

  メッセージ: サーバーは --secure-auth モードで稼働しています。しかし '%s'@'%s' は古い形式のパスワードを使用しています。新しい形式のパスワードに変更してください。

- エラー: 1276 SQLSTATE: HY000 (ER_WARN_FIELD_RESOLVED)

  メッセージ: フィールドまたは参照 '%s%s%s%s%s' は SELECT #%d ではなく、SELECT #%d で解決されました。

- エラー: 1277 SQLSTATE: HY000 (ER_BAD_SLAVE_UNTIL_COND)

  メッセージ: START SLAVE UNTIL へのパラメータまたはその組み合わせが不正です。

- エラー: 1278 SQLSTATE: HY000 (ER_MISSING_SKIP_SLAVE)

  メッセージ: START SLAVE UNTIL で段階的にレプリケーションを行う際には、--skip-slave-start オプションを使うことを推奨します。使わない場合、スレーブの mysqld が不慮の再起動をすると問題が発生します。

- エラー: 1279 SQLSTATE: HY000 (ER_UNTIL_COND_IGNORED)

  メッセージ: スレーブ SQL スレッドが開始されないため、UNTIL オプションは無視されました。

- エラー: 1280 SQLSTATE: 42000 (ER_WRONG_NAME_FOR_INDEX)

  メッセージ: インデックス名 '%s' は不正です。

- エラー: 1281 SQLSTATE: 42000 (ER_WRONG_NAME_FOR_CATALOG)

  メッセージ: カタログ名 '%s' は不正です。

- エラー: 1282 SQLSTATE: HY000 (ER_WARN_QC_RESIZE)

  メッセージ: クエリーキャッシュのサイズを %lu にできませんでした。サイズは %lu になりました。

- エラー: 1283 SQLSTATE: HY000 (ER_BAD_FT_COLUMN)

  メッセージ: カラム '%s' は全文インデックスのキーにはできません。

- エラー: 1284 SQLSTATE: HY000 (ER_UNKNOWN_KEY_CACHE)

  メッセージ: '%s' は不明なキーキャッシュです。

- エラー: 1285 SQLSTATE: HY000 (ER_WARN_HOSTNAME_WONT_WORK)

  メッセージ: MySQL は --skip-name-resolve モードで起動しています。このオプションを外して再起動しなければ、この権限操作は機能しません。

- エラー: 1286 SQLSTATE: 42000 (ER_UNKNOWN_STORAGE_ENGINE)

  メッセージ: '%s' は不明なストレージエンジンです。

- エラー: 1287 SQLSTATE: HY000 (ER_WARN_DEPRECATED_SYNTAX)

  メッセージ: '%s' は将来のリリースで廃止予定です。代わりに %s を使用してください

- エラー: 1288 SQLSTATE: HY000 (ER_NON_UPDATABLE_TABLE)

  メッセージ: 対象表 %s は更新可能ではないので、%s を行えません

- エラー: 1289 SQLSTATE: HY000 (ER_FEATURE_DISABLED)

  メッセージ: 機能 '%s' は無効です。利用するためには '%s' を含めてビルドした MySQL が必要です

- エラー: 1290 SQLSTATE: HY000 (ER_OPTION_PREVENTS_STATEMENT)

  メッセージ: MySQL Server が %s オプションで実行されているので、このステートメントは実行できません

- エラー: 1291 SQLSTATE: HY000 (ER_DUPLICATED_VALUE_IN_TYPE)

  メッセージ: カラム '%s' で、重複する値 '%s' が %s に指定されています

- エラー: 1292 SQLSTATE: 22007 (ER_TRUNCATED_WRONG_VALUE)

  メッセージ: 不正な %s の値が切り捨てられました: '%s'

- エラー: 1293 SQLSTATE: HY000 (ER_TOO_MUCH_AUTO_TIMESTAMP_COLS)

  メッセージ: 不正な表定義です。DEFAULT 句または ON UPDATE 句に CURRENT_TIMESTAMP をともなう TIMESTAMP 型のカラムは 1 つまでです

- エラー: 1294 SQLSTATE: HY000 (ER_INVALID_ON_UPDATE)

  メッセージ: カラム '%s' に ON UPDATE 句は無効です

- エラー: 1295 SQLSTATE: HY000 (ER_UNSUPPORTED_PS)

  メッセージ: このコマンドは、準備済みステートメントプロトコルではまだサポートされていません

- エラー: 1296 SQLSTATE: HY000 (ER_GET_ERRMSG)

  メッセージ: エラー %d '%s' が %s から返されました

- エラー: 1297 SQLSTATE: HY000 (ER_GET_TEMPORARY_ERRMSG)

  メッセージ: 一時エラー %d '%s' が %s から返されました

- エラー: 1298 SQLSTATE: HY000 (ER_UNKNOWN_TIME_ZONE)

  メッセージ: 不明なタイムゾーンまたは不正なタイムゾーン: '%s'

- エラー: 1299 SQLSTATE: HY000 (ER_WARN_INVALID_TIMESTAMP)

  メッセージ: カラム '%s' 行 %ld の TIMESTAMP 値が無効です

- エラー: 1300 SQLSTATE: HY000 (ER_INVALID_CHARACTER_STRING)

  メッセージ: 無効な %s 文字列: '%s'

- エラー: 1301 SQLSTATE: HY000 (ER_WARN_ALLOWED_PACKET_OVERFLOWED)

  メッセージ: %s() の結果は max_allowed_packet (%ld) よりも大きいサイズです- 切り捨てられました

- エラー: 1302 SQLSTATE: HY000 (ER_CONFLICTING_DECLARATIONS)

  メッセージ: 宣言が競合しています: '%s%s' と '%s%s'

- エラー: 1303 SQLSTATE: 2F003 (ER_SP_NO_RECURSIVE_CREATE)

  メッセージ: 別のストアドルーチン内からは %s を作成できません

- エラー: 1304 SQLSTATE: 42000 (ER_SP_ALREADY_EXISTS)

  メッセージ: %s %s はすでに存在します

- エラー: 1305 SQLSTATE: 42000 (ER_SP_DOES_NOT_EXIST)

  メッセージ: %s %s は存在しません
- エラー: 1306 SQLSTATE: HY000 (ER_SP_DROP_FAILED)

  メッセージ: DROP %s %s が失敗しました
- エラー: 1307 SQLSTATE: HY000 (ER_SP_STORE_FAILED)

  メッセージ: CREATE %s %s が失敗しました
- エラー: 1308 SQLSTATE: 42000 (ER_SP_LILABEL_MISMATCH)

  メッセージ: 一致するラベルのない %s: %s
- エラー: 1309 SQLSTATE: 42000 (ER_SP_LABEL_REDEFINE)

  メッセージ: ラベル %s を再定義しています
- エラー: 1310 SQLSTATE: 42000 (ER_SP_LABEL_MISMATCH)

  メッセージ: 終了ラベル %s は一致するものがありません
- エラー: 1311 SQLSTATE: 01000 (ER_SP_UNINIT_VAR)

  メッセージ: 初期化されていない変数 %s を参照しています
- エラー: 1312 SQLSTATE: 0A000 (ER_SP_BADSELECT)

  メッセージ: PROCEDURE %s は指定されたコンテキストで結果セットを返すことができません
- エラー: 1313 SQLSTATE: 42000 (ER_SP_BADRETURN)

  メッセージ: RETURN は FUNCTION でのみ許可されます
- エラー: 1314 SQLSTATE: 0A000 (ER_SP_BADSTATEMENT)

  メッセージ: %s はストアドプロシージャーでは許可されません
- エラー: 1315 SQLSTATE: 42000 (ER_UPDATE_LOG_DEPRECATED_IGNORED)

  メッセージ: 更新ログは非推奨となり、バイナリログに置き換えられました。SET SQL_LOG_UPDATE は無視されました。
- エラー: 1316 SQLSTATE: 42000 (ER_UPDATE_LOG_DEPRECATED_TRANSLATED)

  メッセージ: 更新ログは非推奨となり、バイナリログに置き換えられました。SET SQL_LOG_UPDATE は SET SQL_LOG_BIN に変換されました。
- エラー: 1317 SQLSTATE: 70100 (ER_QUERY_INTERRUPTED)

  メッセージ: クエリーの実行が中断されました
- エラー: 1318 SQLSTATE: 42000 (ER_SP_WRONG_NO_OF_ARGS)

  メッセージ: %s %s の引数の数が不正です。%u 個を予期していましたが %u 個を受け取りました
- エラー: 1319 SQLSTATE: 42000 (ER_SP_COND_MISMATCH)

  メッセージ: 未定義の CONDITION: %s
- エラー: 1320 SQLSTATE: 42000 (ER_SP_NORETURN)

  メッセージ: FUNCTION %s の RETURN がありません
- エラー: 1321 SQLSTATE: 2F005 (ER_SP_NORETURNEND)

  メッセージ: FUNCTION %s が RETURN を返さずに終了しました
- エラー: 1322 SQLSTATE: 42000 (ER_SP_BAD_CURSOR_QUERY)

  メッセージ: カーソルステートメントは SELECT である必要があります

- エラー: 1323 SQLSTATE: 42000 (ER_SP_BAD_CURSOR_SELECT)

  メッセージ: カーソル SELECT には INTO を指定できません
- エラー: 1324 SQLSTATE: 42000 (ER_SP_CURSOR_MISMATCH)

  メッセージ: 未定義の CURSOR: %s
- エラー: 1325 SQLSTATE: 24000 (ER_SP_CURSOR_ALREADY_OPEN)

  メッセージ: カーソルはすでにオープンされています
- エラー: 1326 SQLSTATE: 24000 (ER_SP_CURSOR_NOT_OPEN)

  メッセージ: カーソルがオープンされていません
- エラー: 1327 SQLSTATE: 42000 (ER_SP_UNDECLARED_VAR)

  メッセージ: 宣言されていない変数: %s
- エラー: 1328 SQLSTATE: HY000 (ER_SP_WRONG_NO_OF_FETCH_ARGS)

  メッセージ: FETCH 変数の数が不正です
- エラー: 1329 SQLSTATE: 02000 (ER_SP_FETCH_NO_DATA)

  メッセージ: データなし - ゼロ行がフェッチ、選択、または処理されました
- エラー: 1330 SQLSTATE: 42000 (ER_SP_DUP_PARAM)

  メッセージ: 重複したパラメータ: %s
- エラー: 1331 SQLSTATE: 42000 (ER_SP_DUP_VAR)

  メッセージ: 重複した変数: %s
- エラー: 1332 SQLSTATE: 42000 (ER_SP_DUP_COND)

  メッセージ: 重複した条件: %s
- エラー: 1333 SQLSTATE: 42000 (ER_SP_DUP_CURS)

  メッセージ: 重複したカーソル: %s
- エラー: 1334 SQLSTATE: HY000 (ER_SP_CANT_ALTER)

  メッセージ: ALTER %s %s が失敗しました
- エラー: 1335 SQLSTATE: 0A000 (ER_SP_SUBSELECT_NYI)

  メッセージ: サブクエリー値はサポートされていません
- エラー: 1336 SQLSTATE: 0A000 (ER_STMT_NOT_ALLOWED_IN_SF_OR_TRG)

  メッセージ: ストアドファンクションまたはトリガーでは %s は許可されません
- エラー: 1337 SQLSTATE: 42000 (ER_SP_VARCOND_AFTER_CURSHNDLR)

  メッセージ: カーソルまたはハンドラの宣言のあとに変数または条件の宣言があります
- エラー: 1338 SQLSTATE: 42000 (ER_SP_CURSOR_AFTER_HANDLER)

  メッセージ: ハンドラの宣言のあとにカーソルの宣言があります
- エラー: 1339 SQLSTATE: 20000 (ER_SP_CASE_NOT_FOUND)

  メッセージ: CASE ステートメントにケースがありません
- エラー: 1340 SQLSTATE: HY000 (ER_FPARSER_TOO_BIG_FILE)

  メッセージ: 構成ファイル '%s' は大きすぎます

- エラー: 1341 SQLSTATE: HY000 (ER_FPARSER_BAD_HEADER)

  メッセージ: ファイル '%s' のファイルタイプヘッダーは誤った形式です

- エラー: 1342 SQLSTATE: HY000 (ER_FPARSER_EOF_IN_COMMENT)

  メッセージ: コメント '%s' を解析中に予期せずファイルの終端に達しました

- エラー: 1343 SQLSTATE: HY000 (ER_FPARSER_ERROR_IN_PARAMETER)

  メッセージ: パラメータ '%s' の解析中にエラーが発生しました (行: '%s')

- エラー: 1344 SQLSTATE: HY000 (ER_FPARSER_EOF_IN_UNKNOWN_PARAMETER)

  メッセージ: 不明なパラメータ '%s' をスキップしたときに、予期せずファイルの終端に達しました

- エラー: 1345 SQLSTATE: HY000 (ER_VIEW_NO_EXPLAIN)

  メッセージ: EXPLAIN/SHOW を発行できません。ベースとなるテーブルに対するアクセス権がありません

- エラー: 1346 SQLSTATE: HY000 (ER_FRM_UNKNOWN_TYPE)

  メッセージ: ファイル '%s' のヘッダーは不明なタイプ '%s' です

- エラー: 1347 SQLSTATE: HY000 (ER_WRONG_OBJECT)

  メッセージ: '%s.%s' は %s ではありません

- エラー: 1348 SQLSTATE: HY000 (ER_NONUPDATEABLE_COLUMN)

  メッセージ: カラム '%s' は更新できません

- エラー: 1349 SQLSTATE: HY000 (ER_VIEW_SELECT_DERIVED)

  メッセージ: ビューの SELECT の FROM 句にサブクエリーが含まれています

- エラー: 1350 SQLSTATE: HY000 (ER_VIEW_SELECT_CLAUSE)

  メッセージ: ビューの SELECT に '%s' 句が含まれています

- エラー: 1351 SQLSTATE: HY000 (ER_VIEW_SELECT_VARIABLE)

  メッセージ: ビューの SELECT に変数またはパラメータが含まれています

- エラー: 1352 SQLSTATE: HY000 (ER_VIEW_SELECT_TMPTABLE)

  メッセージ: ビューの SELECT が一時テーブル '%s' を参照しています

- エラー: 1353 SQLSTATE: HY000 (ER_VIEW_WRONG_LIST)

  メッセージ: ビューの SELECT とビューのフィールドリストのカラム数が異なります

- エラー: 1354 SQLSTATE: HY000 (ER_WARN_VIEW_MERGE)

  メッセージ: ビューのマージアルゴリズムはここでは使用できません (未定義のアルゴリズムと見なされました)

- エラー: 1355 SQLSTATE: HY000 (ER_WARN_VIEW_WITHOUT_KEY)

  メッセージ: 更新しているビューには、ベースとなるテーブルの完全なキーがありません

- エラー: 1356 SQLSTATE: HY000 (ER_VIEW_INVALID)

  メッセージ: ビュー '%s.%s' は、無効なテーブル、カラム、関数、または定義者を参照しています/ビューの呼び出し元にそれらを使用するアクセス権がありません

- エラー: 1357 SQLSTATE: HY000 (ER_SP_NO_DROP_SP)

  メッセージ: 別のストアドルーチン内から %s をドロップまたは変更することはできません

- エラー: 1358 SQLSTATE: HY000 (ER_SP_GOTO_IN_HNDLR)

  メッセージ: ストアドプロシージャーハンドラでは GOTO は許可されません

- エラー: 1359 SQLSTATE: HY000 (ER_TRG_ALREADY_EXISTS)

  メッセージ: トリガーがすでに存在します
- エラー: 1360 SQLSTATE: HY000 (ER_TRG_DOES_NOT_EXIST)

  メッセージ: トリガーが存在しません
- エラー: 1361 SQLSTATE: HY000 (ER_TRG_ON_VIEW_OR_TEMP_TABLE)

  メッセージ: トリガーの '%s' はビューまたは一時テーブルです
- エラー: 1362 SQLSTATE: HY000 (ER_TRG_CANT_CHANGE_ROW)

  メッセージ: %s 行の更新は %strigger では許可されません
- エラー: 1363 SQLSTATE: HY000 (ER_TRG_NO_SUCH_ROW_IN_TRG)

  メッセージ: %s 行は %s トリガー内にありません
- エラー: 1364 SQLSTATE: HY000 (ER_NO_DEFAULT_FOR_FIELD)

  メッセージ: フィールド '%s' にはデフォルト値がありません
- エラー: 1365 SQLSTATE: 22012 (ER_DIVISION_BY_ZERO)

  メッセージ: ゼロによる除算
- エラー: 1366 SQLSTATE: HY000 (ER_TRUNCATED_WRONG_VALUE_FOR_FIELD)

  メッセージ: 不正な %s 値: '%s' (カラム '%s' 行 %ld)
- エラー: 1367 SQLSTATE: 22007 (ER_ILLEGAL_VALUE_FOR_TYPE)

  メッセージ: 解析中に不正な %s '%s' 値が見つかりました
- エラー: 1368 SQLSTATE: HY000 (ER_VIEW_NONUPD_CHECK)

  メッセージ: 更新できないビュー '%s.%s' に CHECK OPTION があります
- エラー: 1369 SQLSTATE: HY000 (ER_VIEW_CHECK_FAILED)

  メッセージ: CHECK OPTION が失敗しました '%s.%s'
- エラー: 1370 SQLSTATE: 42000 (ER_PROCACCESS_DENIED_ERROR)

  メッセージ: %s コマンドがユーザー '%s'@'%s' (ルーチン '%s') を拒否しました
- エラー: 1371 SQLSTATE: HY000 (ER_RELAY_LOG_FAIL)

  メッセージ: 古いリレーログのパージに失敗しました: %s
- エラー: 1372 SQLSTATE: HY000 (ER_PASSWD_LENGTH)

  メッセージ: パスワードハッシュは %d 桁の 16 進数であるはずです
- エラー: 1373 SQLSTATE: HY000 (ER_UNKNOWN_TARGET_BINLOG)

  メッセージ: Binlog インデックスでターゲットログが見つかりません
- エラー: 1374 SQLSTATE: HY000 (ER_IO_ERR_LOG_INDEX_READ)

  メッセージ: ログインデックスファイルの読み取りで I/O エラーが発生しました
- エラー: 1375 SQLSTATE: HY000 (ER_BINLOG_PURGE_PROHIBITED)

  メッセージ: サーバー構成によって Binlog のパージが許可されません
- エラー: 1376 SQLSTATE: HY000 (ER_FSEEK_FAIL)

  メッセージ: fseek() に失敗しました

- エラー: 1377 SQLSTATE: HY000 (ER_BINLOG_PURGE_FATAL_ERR)

  メッセージ: ログのパージ中に致命的なエラーが発生しました
- エラー: 1378 SQLSTATE: HY000 (ER_LOG_IN_USE)

  メッセージ: パージ可能なファイルは使用されており、パージされません
- エラー: 1379 SQLSTATE: HY000 (ER_LOG_PURGE_UNKNOWN_ERR)

  メッセージ: ログのパージ中に不明なエラーが発生しました
- エラー: 1380 SQLSTATE: HY000 (ER_RELAY_LOG_INIT)

  メッセージ: リレーログの位置の初期化に失敗しました: %s
- エラー: 1381 SQLSTATE: HY000 (ER_NO_BINARY_LOGGING)

  メッセージ: バイナリロギングを使用していません
- エラー: 1382 SQLSTATE: HY000 (ER_RESERVED_SYNTAX)

  メッセージ: '%s' 構文は MySQL Server の内部で使用するために予約されています
- エラー: 1383 SQLSTATE: HY000 (ER_WSAS_FAILED)

  メッセージ: WSAStartup が失敗しました
- エラー: 1384 SQLSTATE: HY000 (ER_DIFF_GROUPS_PROC)

  メッセージ: 異なるグループのプロシージャーはまだ処理できません
- エラー: 1385 SQLSTATE: HY000 (ER_NO_GROUP_FOR_PROC)

  メッセージ: このプロシージャーでは Select にグループがある必要があります
- エラー: 1386 SQLSTATE: HY000 (ER_ORDER_WITH_PROC)

  メッセージ: このプロシージャーでは ORDER 句は使用できません
- エラー: 1387 SQLSTATE: HY000 (ER_LOGGING_PROHIBIT_CHANGING_OF)

  メッセージ: バイナリロギングおよびレプリケーションによって、グローバルサーバー %s の変更が禁止されています
- エラー: 1388 SQLSTATE: HY000 (ER_NO_FILE_MAPPING)

  メッセージ: ファイルをマップできません: %s、エラー番号: %d
- エラー: 1389 SQLSTATE: HY000 (ER_WRONG_MAGIC)

  メッセージ: %s のマジックが間違っています
- エラー: 1390 SQLSTATE: HY000 (ER_PS_MANY_PARAM)

  メッセージ: 準備済みステートメントにプレースホルダが多すぎます
- エラー: 1391 SQLSTATE: HY000 (ER_KEY_PART_0)

  メッセージ: キーパート '%s' の長さは 0 にできません
- エラー: 1392 SQLSTATE: HY000 (ER_VIEW_CHECKSUM)

  メッセージ: ビューテキストのチェックサムに失敗しました
- エラー: 1393 SQLSTATE: HY000 (ER_VIEW_MULTIUPDATE)

  メッセージ: 結合ビュー '%s.%s' を使用して複数のベーステーブルを変更することはできません
- エラー: 1394 SQLSTATE: HY000 (ER_VIEW_NO_INSERT_FIELD_LIST)

  メッセージ: フィールドリストを指定せずに結合ビュー '%s.%s' に挿入することはできません

- エラー: 1395 SQLSTATE: HY000 (ER_VIEW_DELETE_MERGE_VIEW)

  メッセージ: 結合ビュー '%s.%s' から削除できません

- エラー: 1396 SQLSTATE: HY000 (ER_CANNOT_USER)

  メッセージ: 操作 %s が %s で失敗しました

- エラー: 1397 SQLSTATE: XAE04 (ER_XAER_NOTA)

  メッセージ: XAER_NOTA: 不明な XID

- エラー: 1398 SQLSTATE: XAE05 (ER_XAER_INVAL)

  メッセージ: XAER_INVAL: 無効な引数 (またはサポートされていないコマンド)

- エラー: 1399 SQLSTATE: XAE07 (ER_XAER_RMFAIL)

  メッセージ: XAER_RMFAIL: グローバルトランザクションが %s 状態のときは、コマンドを実行できません

- エラー: 1400 SQLSTATE: XAE09 (ER_XAER_OUTSIDE)

  メッセージ: XAER_OUTSIDE: グローバルトランザクションの外部で一部の処理が行われました

- エラー: 1401 SQLSTATE: XAE03 (ER_XAER_RMERR)

  メッセージ: XAER_RMERR: トランザクションブランチで致命的なエラーが発生しました - データの一貫性をチェックしてください

- エラー: 1402 SQLSTATE: XA100 (ER_XA_RBROLLBACK)

  メッセージ: XA_RBROLLBACK: トランザクションブランチがロールバックされました

- エラー: 1403 SQLSTATE: 42000 (ER_NONEXISTING_PROC_GRANT)

  メッセージ: ユーザー '%s' (ホスト '%s'、ルーチン '%s') にはそのような権限は定義されていません

- エラー: 1404 SQLSTATE: HY000 (ER_PROC_AUTO_GRANT_FAIL)

  メッセージ: EXECUTE および ALTER ROUTINE 権限の付与に失敗しました

- エラー: 1405 SQLSTATE: HY000 (ER_PROC_AUTO_REVOKE_FAIL)

  メッセージ: ドロップされたルーチンのすべての権限の取り消しに失敗しました

- エラー: 1406 SQLSTATE: 22001 (ER_DATA_TOO_LONG)

  メッセージ: カラム '%s' 行 %ld のデータが長すぎます

- エラー: 1407 SQLSTATE: 42000 (ER_SP_BAD_SQLSTATE)

  メッセージ: 不良な SQLSTATE: '%s'

- エラー: 1408 SQLSTATE: HY000 (ER_STARTUP)

  メッセージ: %s: 接続準備完了。バージョン: '%s' ソケット: '%s' ポート: %d %s

- エラー: 1409 SQLSTATE: HY000 (ER_LOAD_FROM_FIXED_SIZE_ROWS_TO_VAR)

  メッセージ: 固定長の行を持つファイルから変数に値をロードできません

- エラー: 1410 SQLSTATE: 42000 (ER_CANT_CREATE_USER_WITH_GRANT)

  メッセージ: ユーザーを GRANT 付きで作成することは許可されていません

- エラー: 1411 SQLSTATE: HY000 (ER_WRONG_VALUE_FOR_TYPE)

  メッセージ: 不正な %s 値: '%s' (関数 %s)

- エラー: 1412 SQLSTATE: HY000 (ER_TABLE_DEF_CHANGED)

  メッセージ: テーブル定義が変更されました。トランザクションを再試行してください

- エラー: 1413 SQLSTATE: 42000 (ER_SP_DUP_HANDLER)

  メッセージ: 同じブロックに重複したハンドラが宣言されています

- エラー: 1414 SQLSTATE: 42000 (ER_SP_NOT_VAR_ARG)

  メッセージ: OUT または INOUT の引数 %d (ルーチン %s) は、BEFORE トリガーの変数または NEW 疑似変数ではありません

- エラー: 1415 SQLSTATE: 0A000 (ER_SP_NO_RETSET)

  メッセージ: %s から結果セットを返すことは許可されません

- エラー: 1416 SQLSTATE: 22003 (ER_CANT_CREATE_GEOMETRY_OBJECT)

  メッセージ: GEOMETRY フィールドに送信したデータから幾何オブジェクトを取得できません

- エラー: 1417 SQLSTATE: HY000 (ER_FAILED_ROUTINE_BREAK_BINLOG)

  メッセージ: ルーチンが失敗し、NO SQL および READS SQL DATA のどちらも宣言されておらず、バイナリロギングが有効にされています。非トランザクションテーブルが更新された場合は、バイナリログにそれらの変更が書き込まれていません。

- エラー: 1418 SQLSTATE: HY000 (ER_BINLOG_UNSAFE_ROUTINE)

  メッセージ: この関数では、DETERMINISTIC、NO SQL、または READS SQL DATA のいずれも宣言されておらず、バイナリロギングが有効にされています (安全性の劣る log_bin_trust_function_creators 変数を使用することもできます)

- エラー: 1419 SQLSTATE: HY000 (ER_BINLOG_CREATE_ROUTINE_NEED_SUPER)

  メッセージ: SUPER 権限がなく、バイナリロギングが有効にされています (安全性の劣る log_bin_trust_function_creators 変数を使用することもできます)

- エラー: 1420 SQLSTATE: HY000 (ER_EXEC_STMT_WITH_OPEN_CURSOR)

  メッセージ: オープンカーソルが関連付けられている準備済みステートメントは実行できません。ステートメントをリセットして、再実行してください。

- エラー: 1421 SQLSTATE: HY000 (ER_STMT_HAS_NO_OPEN_CURSOR)

  メッセージ: ステートメント (%lu) にはオープンカーソルがありません。

- エラー: 1422 SQLSTATE: HY000 (ER_COMMIT_NOT_ALLOWED_IN_SF_OR_TRG)

  メッセージ: ストアドファンクションまたはトリガーでは、明示的または暗黙的なコミットは許可されません。

- エラー: 1423 SQLSTATE: HY000 (ER_NO_DEFAULT_FOR_VIEW_FIELD)

  メッセージ: ビュー '%s.%s' のベースとなるテーブルのフィールドにデフォルト値がありません

- エラー: 1424 SQLSTATE: HY000 (ER_SP_NO_RECURSION)

  メッセージ: 再帰的なストアドファンクションおよびトリガーは許可されません。

- エラー: 1425 SQLSTATE: 42000 (ER_TOO_BIG_SCALE)

  メッセージ: スケール %d (カラム '%s' に指定されました) は大きすぎます。最大は %lu です。

- エラー: 1426 SQLSTATE: 42000 (ER_TOO_BIG_PRECISION)

  メッセージ: 精度 %d (カラム '%s' に指定されました) は大きすぎます。最大は %lu です。

- エラー: 1427 SQLSTATE: 42000 (ER_M_BIGGER_THAN_D)

  メッセージ: float(M,D)、double(M,D)、または decimal(M,D) の場合、M は >= D である必要があります (カラム '%s')。

- エラー: 1428 SQLSTATE: HY000 (ER_WRONG_LOCK_OF_SYSTEM_TABLE)

  メッセージ: システムテーブルの書き込みロックとその他のテーブルまたはロックタイプを組み合わせることはできません

- エラー: 1429 SQLSTATE: HY000 (ER_CONNECT_TO_FOREIGN_DATA_SOURCE)

  メッセージ: 外部データソースに接続できません: %s

- エラー: 1430 SQLSTATE: HY000 (ER_QUERY_ON_FOREIGN_DATA_SOURCE)

  メッセージ: 外部データソースに対するクエリーの処理に問題がありました。データソースエラー: %s

- エラー: 1431 SQLSTATE: HY000 (ER_FOREIGN_DATA_SOURCE_DOESNT_EXIST)

  メッセージ: 参照しようとしている外部データソースは存在しません。データソースエラー: %s

- エラー: 1432 SQLSTATE: HY000 (ER_FOREIGN_DATA_STRING_INVALID_CANT_CREATE)

  メッセージ: FEDERATED テーブルを作成できません。データソース接続文字列 '%s' は正しい形式ではありません

- エラー: 1433 SQLSTATE: HY000 (ER_FOREIGN_DATA_STRING_INVALID)

  メッセージ: データソース接続文字列 '%s' は正しい形式ではありません

- エラー: 1434 SQLSTATE: HY000 (ER_CANT_CREATE_FEDERATED_TABLE)

  メッセージ: FEDERATED テーブルを作成できません。外部データソースエラー: %s

- エラー: 1435 SQLSTATE: HY000 (ER_TRG_IN_WRONG_SCHEMA)

  メッセージ: 間違ったスキーマにトリガーがあります

- エラー: 1436 SQLSTATE: HY000 (ER_STACK_OVERRUN_NEED_MORE)

  メッセージ: スレッドスタック不足です (使用: %ld ; サイズ: %ld ; 要求: %ld)。より大きい値で 'mysqld --thread_stack=#' の指定をしてください。

- エラー: 1437 SQLSTATE: 42000 (ER_TOO_LONG_BODY)

  メッセージ: '%s' のルーチンの本体が長すぎます

- エラー: 1438 SQLSTATE: HY000 (ER_WARN_CANT_DROP_DEFAULT_KEYCACHE)

  メッセージ: デフォルトのキーキャッシュはドロップできません

- エラー: 1439 SQLSTATE: 42000 (ER_TOO_BIG_DISPLAYWIDTH)

  メッセージ: カラム '%s' の表示幅が範囲外です (最大 = %lu)

- エラー: 1440 SQLSTATE: XAE08 (ER_XAER_DUPID)

  メッセージ: XAER_DUPID: XID はすでに存在します

- エラー: 1441 SQLSTATE: 22008 (ER_DATETIME_FUNCTION_OVERFLOW)

  メッセージ: 日付時間関数: %s フィールドがオーバーフローしました

- エラー: 1442 SQLSTATE: HY000 (ER_CANT_UPDATE_USED_TABLE_IN_SF_OR_TRG)

  メッセージ: ストアドファンクション/トリガーでテーブル '%s' を更新できません。このストアドファンクション/トリガーを呼び出したステートメントによってすでに使用されているためです。

- エラー: 1443 SQLSTATE: HY000 (ER_VIEW_PREVENT_UPDATE)

  メッセージ: テーブル '%s' の定義が操作 %s (対象テーブル '%s') を妨げています。

- エラー: 1444 SQLSTATE: HY000 (ER_PS_NO_RECURSION)

  メッセージ: 準備済みステートメントに、その同じステートメントを参照するストアドルーチンの呼び出しが含まれています。そのような再帰的な形式で準備済みステートメントを実行することはできません

- エラー: 1445 SQLSTATE: HY000 (ER_SP_CANT_SET_AUTOCOMMIT)

  メッセージ: ストアドファンクションまたはトリガーから自動コミットを設定することはできません

- エラー: 1446 SQLSTATE: HY000 (ER_MALFORMED_DEFINER)

  メッセージ: Definer が完全修飾されていません

- エラー: 1447 SQLSTATE: HY000 (ER_VIEW_FRM_NO_USER)

  メッセージ: ビュー '%s'.'%s' に定義者情報がありません (古いテーブル形式)。現在のユーザーが定義者として使用されます。ビューを再作成してください。

- エラー: 1448 SQLSTATE: HY000 (ER_VIEW_OTHER_USER)

  メッセージ: 定義者 '%s'@'%s' でビューを作成するには、SUPER 権限が必要となります

- エラー: 1449 SQLSTATE: HY000 (ER_NO_SUCH_USER)

  メッセージ: 定義者 ('%s'@'%s') として指定されたユーザーは存在しません

- エラー: 1450 SQLSTATE: HY000 (ER_FORBID_SCHEMA_CHANGE)

  メッセージ: スキーマを '%s' から '%s' に変更することは許可されません。

- エラー: 1451 SQLSTATE: 23000 (ER_ROW_IS_REFERENCED_2)

  メッセージ: 親の行を削除または更新できません。外部キー制約に違反しています (%s)

- エラー: 1452 SQLSTATE: 23000 (ER_NO_REFERENCED_ROW_2)

  メッセージ: 子の行を追加または更新できません。外部キー制約に違反しています (%s)

- エラー: 1453 SQLSTATE: 42000 (ER_SP_BAD_VAR_SHADOW)

  メッセージ: 変数 '%s' は、`...` で囲むか、名前変更する必要があります。

- エラー: 1454 SQLSTATE: HY000 (ER_TRG_NO_DEFINER)

  メッセージ: トリガー '%s'.'%s' に定義者属性がありません。トリガーは呼び出し元の権限に基づいてアクティブ化されます。呼び出し元の権限が不十分である可能性があります。トリガーを再作成してください。

- エラー: 1455 SQLSTATE: HY000 (ER_OLD_FILE_FORMAT)

  メッセージ: '%s' は古い形式です。'%s' オブジェクトを再作成してください

- エラー: 1456 SQLSTATE: HY000 (ER_SP_RECURSION_LIMIT)

  メッセージ: 再帰の制限 %d (max_sp_recursion_depth 変数で設定します) をルーチン %s が超えました

- エラー: 1457 SQLSTATE: HY000 (ER_SP_PROC_TABLE_CORRUPT)

  メッセージ: ルーチン %s のロードに失敗しました。テーブル mysql.proc が、存在しないか、破損しているか、不良なデータが含まれています (内部コード %d)

- エラー: 1458 SQLSTATE: 42000 (ER_SP_WRONG_NAME)

  メッセージ: 不正なルーチン名 '%s'

- エラー: 1459 SQLSTATE: HY000 (ER_TABLE_NEEDS_UPGRADE)

  メッセージ: テーブルをアップグレードする必要があります。"REPAIR TABLE `%s`" を実行するか、ダンプ/リロードして修復してください。

- エラー: 1460 SQLSTATE: 42000 (ER_SP_NO_AGGREGATE)

  メッセージ: AGGREGATE はストアドファンクションではサポートされません

- エラー: 1461 SQLSTATE: 42000 (ER_MAX_PREPARED_STMT_COUNT_REACHED)

  メッセージ: max_prepared_stmt_count 個を超えるステートメントは作成できません (現在の値: %lu)

- エラー: 1462 SQLSTATE: HY000 (ER_VIEW_RECURSIVE)

  メッセージ: `%s`.`%s` にはビューの再帰が含まれています

- エラー: 1463 SQLSTATE: 42000 (ER_NON_GROUPING_FIELD_USED)

  メッセージ: グループフィールドではない '%s' が %s 句で使用されました

- エラー: 1464 SQLSTATE: HY000 (ER_TABLE_CANT_HANDLE_SPKEYS)

  メッセージ: 使用されたテーブルタイプは SPATIAL インデックスをサポートしていません

- エラー: 1465 SQLSTATE: HY000 (ER_NO_TRIGGERS_ON_SYSTEM_SCHEMA)

  メッセージ: システムテーブルに対してトリガーは作成できません

- エラー: 1466 SQLSTATE: HY000 (ER_REMOVED_SPACES)

  メッセージ: 名前 '%s' から先頭の空白文字が削除されました

- エラー: 1467 SQLSTATE: HY000 (ER_AUTOINC_READ_FAILED)

  メッセージ: ストレージエンジンからの自動インクリメント値の読み取りに失敗しました

- エラー: 1468 SQLSTATE: HY000 (ER_USERNAME)

  メッセージ: ユーザー名

- エラー: 1469 SQLSTATE: HY000 (ER_HOSTNAME)

  メッセージ: ホスト名

- エラー: 1470 SQLSTATE: HY000 (ER_WRONG_STRING_LENGTH)

  メッセージ: 文字列 '%s' は %s に対して長すぎます (%d より短くなるようにしてください)

- エラー: 1471 SQLSTATE: HY000 (ER_NON_INSERTABLE_TABLE)

  メッセージ: 対象表 %s は挿入可能ではないので、%s を行えません。

- エラー: 1472 SQLSTATE: HY000 (ER_ADMIN_WRONG_MRG_TABLE)

  メッセージ: テーブル '%s' の定義が異なるか、非 MyISAM タイプであるか、存在しません

- エラー: 1473 SQLSTATE: HY000 (ER_TOO_HIGH_LEVEL_OF_NESTING_FOR_SELECT)

  メッセージ: Select のネストが深すぎます

- エラー: 1474 SQLSTATE: HY000 (ER_NAME_BECOMES_EMPTY)

  メッセージ: 名前 '%s' は '' になりました

- エラー: 1475 SQLSTATE: HY000 (ER_AMBIGUOUS_FIELD_TERM)

  メッセージ: FIELDS TERMINATED 文字列の最初の文字があいまいです。オプションではなく、空ではない FIELDS ENCLOSED BY を使用してください

- エラー: 1476 SQLSTATE: HY000 (ER_FOREIGN_SERVER_EXISTS)

  メッセージ: 作成しようとしている外部サーバー %s はすでに存在します。

- エラー: 1477 SQLSTATE: HY000 (ER_FOREIGN_SERVER_DOESNT_EXIST)

  メッセージ: 参照しようとしている外部サーバー名はすでに存在しません。データソースエラー: %s

- エラー: 1478 SQLSTATE: HY000 (ER_ILLEGAL_HA_CREATE_OPTION)

  メッセージ: テーブルのストレージエンジン '%s' は、作成オプション '%s' をサポートしていません

- エラー: 1479 SQLSTATE: HY000 (ER_PARTITION_REQUIRES_VALUES_ERROR)

  メッセージ: 構文エラー: %s PARTITIONING では、各パーティションの VALUES %s の定義が必要となります

- エラー: 1480 SQLSTATE: HY000 (ER_PARTITION_WRONG_VALUES_ERROR)

  メッセージ: %s PARTITIONING のみがパーティション定義に VALUES %s を使用できます

- エラー: 1481 SQLSTATE: HY000 (ER_PARTITION_MAXVALUE_ERROR)

  メッセージ: MAXVALUE は最後のパーティション定義にのみ使用できます

- エラー: 1482 SQLSTATE: HY000 (ER_PARTITION_SUBPARTITION_ERROR)

  メッセージ: サブパーティションは、キーを使用したハッシュパーティションとしてのみ作成できます

- エラー: 1483 SQLSTATE: HY000 (ER_PARTITION_SUBPART_MIX_ERROR)

  メッセージ: 1 つのパーティション上にある場合は、すべてのパーティションにサブパーティションを定義する必要があります

- エラー: 1484 SQLSTATE: HY000 (ER_PARTITION_WRONG_NO_PART_ERROR)

  メッセージ: 定義されたパーティションの数が間違っており、以前の設定と一致しません

- エラー: 1485 SQLSTATE: HY000 (ER_PARTITION_WRONG_NO_SUBPART_ERROR)

  メッセージ: 定義されたサブパーティションの数が間違っており、以前の設定と一致しません

- エラー: 1486 SQLSTATE: HY000 (ER_WRONG_EXPR_IN_PARTITION_FUNC_ERROR)

  メッセージ: (サブ) パーティショニング関数の定数、ランダム、またはタイムゾーンに依存した式は許可されません

- エラー: 1487 SQLSTATE: HY000 (ER_NO_CONST_EXPR_IN_RANGE_OR_LIST_ERROR)

  メッセージ: RANGE/LIST VALUES の式は定数である必要があります

- エラー: 1488 SQLSTATE: HY000 (ER_FIELD_NOT_FOUND_PART_ERROR)

  メッセージ: パーティション関数のフィールドリストのフィールドがテーブルにありません

- エラー: 1489 SQLSTATE: HY000 (ER_LIST_OF_FIELDS_ONLY_IN_HASH_ERROR)

  メッセージ: フィールドリストは KEY パーティションでのみ許可されます

- エラー: 1490 SQLSTATE: HY000 (ER_INCONSISTENT_PARTITION_INFO_ERROR)

  メッセージ: frm ファイル内のパーティション情報が、その frm ファイルに書き込まれる可能性がある情報と整合性がありません

- エラー: 1491 SQLSTATE: HY000 (ER_PARTITION_FUNC_NOT_ALLOWED_ERROR)

  メッセージ: %s 関数が間違った型を返しました

- エラー: 1492 SQLSTATE: HY000 (ER_PARTITIONS_MUST_BE_DEFINED_ERROR)

  メッセージ: %s パーティションの各パーティションを定義する必要があります

- エラー: 1493 SQLSTATE: HY000 (ER_RANGE_NOT_INCREASING_ERROR)

  メッセージ: VALUES LESS THAN 値は、各パーティションで厳密に増加している必要があります

- エラー: 1494 SQLSTATE: HY000 (ER_INCONSISTENT_TYPE_OF_FUNCTIONS_ERROR)

  メッセージ: VALUES 値は、パーティション関数と同じ型である必要があります

- エラー: 1495 SQLSTATE: HY000 (ER_MULTIPLE_DEF_CONST_IN_LIST_PART_ERROR)

  メッセージ: リストパーティショニングに同じ定数の定義が複数あります

- エラー: 1496 SQLSTATE: HY000 (ER_PARTITION_ENTRY_ERROR)

  メッセージ: パーティショニングは、クエリー内でスタンドアロンで使用できません

- エラー: 1497 SQLSTATE: HY000 (ER_MIX_HANDLER_ERROR)

  メッセージ: パーティション内でのハンドラの混在は、このバージョンの MySQL では許可されません

- エラー: 1498 SQLSTATE: HY000 (ER_PARTITION_NOT_DEFINED_ERROR)

メッセージ: パーティション化されたエンジンの場合は、すべての %s を定義する必要があります

- エラー: 1499 SQLSTATE: HY000 (ER_TOO_MANY_PARTITIONS_ERROR)

  メッセージ: 定義されているパーティション (サブパーティションを含む) が多すぎます

- エラー: 1500 SQLSTATE: HY000 (ER_SUBPARTITION_ERROR)

  メッセージ: サブパーティショニングの場合は、RANGE/LIST パーティショニングと HASH/KEY パーティショニングの混在のみが可能です

- エラー: 1501 SQLSTATE: HY000 (ER_CANT_CREATE_HANDLER_FILE)

  メッセージ: 特定のハンドラファイルの作成に失敗しました

- エラー: 1502 SQLSTATE: HY000 (ER_BLOB_FIELD_IN_PART_FUNC_ERROR)

  メッセージ: BLOB フィールドはパーティション関数では許可されません

- エラー: 1503 SQLSTATE: HY000 (ER_UNIQUE_KEY_NEED_ALL_FIELDS_IN_PF)

  メッセージ: %s はテーブルのパーティショニング関数にすべてのカラムを含める必要があります

- エラー: 1504 SQLSTATE: HY000 (ER_NO_PARTS_ERROR)

  メッセージ: %s の数 = 0 は許可される値ではありません

- エラー: 1505 SQLSTATE: HY000 (ER_PARTITION_MGMT_ON_NONPARTITIONED)

  メッセージ: パーティション化されていないテーブルでパーティション管理を行うことはできません

- エラー: 1506 SQLSTATE: HY000 (ER_FOREIGN_KEY_ON_PARTITIONED)

  メッセージ: 外部キー句をパーティショニングと一緒に使用することはまだサポートされていません

- エラー: 1507 SQLSTATE: HY000 (ER_DROP_PARTITION_NON_EXISTENT)

  メッセージ: %s のパーティションリストにエラーがあります

- エラー: 1508 SQLSTATE: HY000 (ER_DROP_LAST_PARTITION)

  メッセージ: すべてのパーティションを削除することはできません。代わりに DROP TABLE を使用してください

- エラー: 1509 SQLSTATE: HY000 (ER_COALESCE_ONLY_ON_HASH_PARTITION)

  メッセージ: COALESCE PARTITION は HASH/KEY パーティションでのみ使用できます

- エラー: 1510 SQLSTATE: HY000 (ER_REORG_HASH_ONLY_ON_SAME_NO)

  メッセージ: REORGANIZE PARTITION は、番号が変更されないようにパーティションを再編成するためにのみ使用できます

- エラー: 1511 SQLSTATE: HY000 (ER_REORG_NO_PARAM_ERROR)

  メッセージ: パラメータを指定しない REORGANIZE PARTITION は、HASH PARTITION を使用して自動的にパーティション化されたテーブルに対してのみ使用できます

- エラー: 1512 SQLSTATE: HY000 (ER_ONLY_ON_RANGE_LIST_PARTITION)

  メッセージ: %s PARTITION は RANGE/LIST パーティションに対してのみ使用できます

- エラー: 1513 SQLSTATE: HY000 (ER_ADD_PARTITION_SUBPART_ERROR)

  メッセージ: 間違ったサブパーティション番号を使用してパーティションを追加しようとしています

- エラー: 1514 SQLSTATE: HY000 (ER_ADD_PARTITION_NO_NEW_PARTITION)

  メッセージ: 少なくとも 1 つのパーティションを追加する必要があります

- エラー: 1515 SQLSTATE: HY000 (ER_COALESCE_PARTITION_NO_PARTITION)

メッセージ: 少なくとも 1 つのパーティションを合体する必要があります

- エラー: 1516 SQLSTATE: HY000 (ER_REORG_PARTITION_NOT_EXIST)

  メッセージ: 実際のパーティション数より多いパーティションを再編成しようとしています

- エラー: 1517 SQLSTATE: HY000 (ER_SAME_NAME_PARTITION)

  メッセージ: 重複したパーティション名 %s

- エラー: 1518 SQLSTATE: HY000 (ER_NO_BINLOG_ERROR)

  メッセージ: このコマンドで Binlog を停止することは許可されません

- エラー: 1519 SQLSTATE: HY000 (ER_CONSECUTIVE_REORG_PARTITIONS)

  メッセージ: 一連のパーティションを再編成する場合、それらは連続した順序である必要があります

- エラー: 1520 SQLSTATE: HY000 (ER_REORG_OUTSIDE_RANGE)

  メッセージ: 範囲パーティションの再編成では、範囲を拡張できる最後のパーティションを除いて、合計の範囲は変更できません

- エラー: 1521 SQLSTATE: HY000 (ER_PARTITION_FUNCTION_FAILURE)

  メッセージ: パーティション関数は、このバージョンではこのハンドラに対してサポートされません

- エラー: 1522 SQLSTATE: HY000 (ER_PART_STATE_ERROR)

  メッセージ: パーティションの状態は CREATE/ALTER TABLE から定義できません

- エラー: 1523 SQLSTATE: HY000 (ER_LIMITED_PART_RANGE)

  メッセージ: %s ハンドラは、VALUES で 32 ビットの整数のみをサポートしています

- エラー: 1524 SQLSTATE: HY000 (ER_PLUGIN_IS_NOT_LOADED)

  メッセージ: プラグイン '%s' はロードされません

- エラー: 1525 SQLSTATE: HY000 (ER_WRONG_VALUE)

  メッセージ: 不正な %s 値: '%s'

- エラー: 1526 SQLSTATE: HY000 (ER_NO_PARTITION_FOR_GIVEN_VALUE)

  メッセージ: テーブルには値 %s のパーティションがありません

- エラー: 1527 SQLSTATE: HY000 (ER_FILEGROUP_OPTION_ONLY_ONCE)

  メッセージ: %s を複数回指定することは許可されません

- エラー: 1528 SQLSTATE: HY000 (ER_CREATE_FILEGROUP_FAILED)

  メッセージ: %s の作成に失敗しました

- エラー: 1529 SQLSTATE: HY000 (ER_DROP_FILEGROUP_FAILED)

  メッセージ: %s のドロップに失敗しました

- エラー: 1530 SQLSTATE: HY000 (ER_TABLESPACE_AUTO_EXTEND_ERROR)

  メッセージ: ハンドラはテーブルスペースの自動拡張をサポートしていません

- エラー: 1531 SQLSTATE: HY000 (ER_WRONG_SIZE_NUMBER)

  メッセージ: サイズパラメータが不正に指定されました (数値または 10M という形式)

- エラー: 1532 SQLSTATE: HY000 (ER_SIZE_OVERFLOW_ERROR)

  メッセージ: サイズの数値は正しい形式ですが、数字部分が 20 億を超えることは許可されません

- エラー: 1533 SQLSTATE: HY000 (ER_ALTER_FILEGROUP_FAILED)

メッセージ: 変更に失敗しました: %s

- エラー: 1534 SQLSTATE: HY000 (ER_BINLOG_ROW_LOGGING_FAILED)

  メッセージ: 行ベースのバイナリログへの 1 行の書き込みが失敗しました

- エラー: 1535 SQLSTATE: HY000 (ER_BINLOG_ROW_WRONG_TABLE_DEF)

  メッセージ: マスターとスレーブのテーブル定義が一致しません: %s

- エラー: 1536 SQLSTATE: HY000 (ER_BINLOG_ROW_RBR_TO_SBR)

  メッセージ: --log-slave-updates を指定して実行されているスレーブは、行ベースのバイナリログイベントをレプリケートできるように、行ベースのバイナリロギングを使用する必要があります

- エラー: 1537 SQLSTATE: HY000 (ER_EVENT_ALREADY_EXISTS)

  メッセージ: イベント '%s' はすでに存在します

- エラー: 1538 SQLSTATE: HY000 (ER_EVENT_STORE_FAILED)

  メッセージ: イベント %s の格納に失敗しました。ストレージエンジンからエラーコード %d が返されました。

- エラー: 1539 SQLSTATE: HY000 (ER_EVENT_DOES_NOT_EXIST)

  メッセージ: 不明なイベント '%s'

- エラー: 1540 SQLSTATE: HY000 (ER_EVENT_CANT_ALTER)

  メッセージ: イベント '%s' の変更に失敗しました

- エラー: 1541 SQLSTATE: HY000 (ER_EVENT_DROP_FAILED)

  メッセージ: %s のドロップに失敗しました

- エラー: 1542 SQLSTATE: HY000 (ER_EVENT_INTERVAL_NOT_POSITIVE_OR_TOO_BIG)

  メッセージ: INTERVAL が正数ではないか、大きすぎます

- エラー: 1543 SQLSTATE: HY000 (ER_EVENT_ENDS_BEFORE_STARTS)

  メッセージ: ENDS が無効であるか、STARTS の前にあります

- エラー: 1544 SQLSTATE: HY000 (ER_EVENT_EXEC_TIME_IN_THE_PAST)

  メッセージ: イベントの実行時間が過去になっています。イベントが無効にされました

- エラー: 1545 SQLSTATE: HY000 (ER_EVENT_OPEN_TABLE_FAILED)

  メッセージ: mysql.event のオープンに失敗しました

- エラー: 1546 SQLSTATE: HY000 (ER_EVENT_NEITHER_M_EXPR_NOR_M_AT)

  メッセージ: 日付時間式が指定されていません

- エラー: 1547 SQLSTATE: HY000 (ER_OBSOLETE_COL_COUNT_DOESNT_MATCH_CORRUPTED)

  メッセージ: mysql.%s のカラム数が間違っています。%d 個を予期しましたが、%d 個が見つかりました。テーブルが破損している可能性があります

- エラー: 1548 SQLSTATE: HY000 (ER_OBSOLETE_CANNOT_LOAD_FROM_TABLE)

  メッセージ: mysql.%s からロードできません。テーブルが破損している可能性があります

- エラー: 1549 SQLSTATE: HY000 (ER_EVENT_CANNOT_DELETE)

  メッセージ: mysql.event からのイベントの削除に失敗しました

- エラー: 1550 SQLSTATE: HY000 (ER_EVENT_COMPILE_ERROR)

  メッセージ: イベントの本体のコンパイル中にエラーが発生しました

- エラー: 1551 SQLSTATE: HY000 (ER_EVENT_SAME_NAME)

  メッセージ: 古いイベント名と新しいイベント名が同じです

- エラー: 1552 SQLSTATE: HY000 (ER_EVENT_DATA_TOO_LONG)

  メッセージ: カラム '%s' のデータが大きすぎます

- エラー: 1553 SQLSTATE: HY000 (ER_DROP_INDEX_FK)

  メッセージ: インデックス '%s' をドロップできません。外部キー制約で必要とされています

- エラー: 1554 SQLSTATE: HY000 (ER_WARN_DEPRECATED_SYNTAX_WITH_VER)

  メッセージ: 構文 '%s' は非推奨となり、MySQL %s で削除されます。代わりに %s を使用してください

- エラー: 1555 SQLSTATE: HY000 (ER_CANT_WRITE_LOCK_LOG_TABLE)

  メッセージ: ログテーブルを書き込みロックすることはできません。読み取りアクセスのみを行うことができます

- エラー: 1556 SQLSTATE: HY000 (ER_CANT_LOCK_LOG_TABLE)

  メッセージ: ログテーブルに対してロックは使用できません。

- エラー: 1557 SQLSTATE: 23000 (ER_FOREIGN_DUPLICATE_KEY)

  メッセージ: テーブル '%s'、エントリ '%s'、キー %d の外部キー制約を維持すると、重複したエントリが発生する可能性があります

  ER_FOREIGN_DUPLICATE_KEY は 5.6.3 で削除されました。

  5.6.4 で、ER_FOREIGN_DUPLICATE_KEY が ER_FOREIGN_DUPLICATE_KEY_OLD_UNUSED に名前変更されました。

- エラー: 1557 SQLSTATE: 23000 (ER_FOREIGN_DUPLICATE_KEY_OLD_UNUSED)

  メッセージ: テーブル '%s'、エントリ '%s'、キー %d の外部キー制約を維持すると、重複したエントリが発生する可能性があります

  ER_FOREIGN_DUPLICATE_KEY_OLD_UNUSED は 5.6.4 で導入されました。

  5.6.4 で、ER_FOREIGN_DUPLICATE_KEY が ER_FOREIGN_DUPLICATE_KEY_OLD_UNUSED に名前変更されました。

- エラー: 1558 SQLSTATE: HY000 (ER_COL_COUNT_DOESNT_MATCH_PLEASE_UPDATE)

  メッセージ: mysql.%s のカラム数が間違っています。%d 個を予期しましたが、%d 個が見つかりました。MySQL %d で作成され、現在は %d を実行しています。mysql_upgrade を使用して、このエラーを修正してください。

- エラー: 1559 SQLSTATE: HY000 (ER_TEMP_TABLE_PREVENTS_SWITCH_OUT_OF_RBR)

  メッセージ: セッションが一時テーブルをオープンしているときに、行ベースのバイナリログ形式を切り替えることはできません

- エラー: 1560 SQLSTATE: HY000 (ER_STORED_FUNCTION_PREVENTS_SWITCH_BINLOG_FORMAT)

  メッセージ: ストアドファンクションまたはトリガー内ではバイナリロギング形式は変更できません

- エラー: 1561 SQLSTATE: HY000 (ER_NDB_CANT_SWITCH_BINLOG_FORMAT)

  メッセージ: NDB クラスタエンジンは、実行中の Binlog 形式の変更をまだサポートしていません

- エラー: 1562 SQLSTATE: HY000 (ER_PARTITION_NO_TEMPORARY)

  メッセージ: パーティションを持つ一時テーブルは作成できません

- エラー: 1563 SQLSTATE: HY000 (ER_PARTITION_CONST_DOMAIN_ERROR)

  メッセージ: パーティション定数がパーティション関数ドメインの範囲外にあります

- エラー: 1564 SQLSTATE: HY000 (ER_PARTITION_FUNCTION_IS_NOT_ALLOWED)

  メッセージ: このパーティション関数は許可されません

- エラー: 1565 SQLSTATE: HY000 (ER_DDL_LOG_ERROR)

  メッセージ: DDL ログにエラーがあります

- エラー: 1566 SQLSTATE: HY000 (ER_NULL_IN_VALUES_LESS_THAN)

  メッセージ: VALUES LESS THAN に NULL 値を使用することは許可されません

- エラー: 1567 SQLSTATE: HY000 (ER_WRONG_PARTITION_NAME)

  メッセージ: 不正なパーティション名

- エラー: 1568 SQLSTATE: 25001 (ER_CANT_CHANGE_TX_ISOLATION)

  メッセージ: トランザクションの進行中に、トランザクションの分離レベルは変更できません

  ER_CANT_CHANGE_TX_ISOLATION は 5.6.4 で削除されました。

  5.6.5 で、ER_CANT_CHANGE_TX_ISOLATION が ER_CANT_CHANGE_TX_CHARACTERISTICS に名前変更されました。

- エラー: 1568 SQLSTATE: 25001 (ER_CANT_CHANGE_TX_CHARACTERISTICS)

  メッセージ: トランザクションの進行中に、トランザクションの特性は変更できません

  ER_CANT_CHANGE_TX_CHARACTERISTICS は 5.6.5 で導入されました。

  5.6.5 で、ER_CANT_CHANGE_TX_ISOLATION が ER_CANT_CHANGE_TX_CHARACTERISTICS に名前変更されました。

- エラー: 1569 SQLSTATE: HY000 (ER_DUP_ENTRY_AUTOINCREMENT_CASE)

  メッセージ: ALTER TABLE によって auto_increment が初期化され、重複したエントリ '%s' がキー '%s' に作成されました

- エラー: 1570 SQLSTATE: HY000 (ER_EVENT_MODIFY_QUEUE_ERROR)

  メッセージ: 内部スケジューラエラー %d

- エラー: 1571 SQLSTATE: HY000 (ER_EVENT_SET_VAR_ERROR)

  メッセージ: スケジューラの開始中/停止中にエラーが発生しました。エラーコード %u

- エラー: 1572 SQLSTATE: HY000 (ER_PARTITION_MERGE_ERROR)

  メッセージ: パーティション化されたテーブルではエンジンは使用できません

- エラー: 1573 SQLSTATE: HY000 (ER_CANT_ACTIVATE_LOG)

  メッセージ: '%s' ログをアクティブ化できません

- エラー: 1574 SQLSTATE: HY000 (ER_RBR_NOT_AVAILABLE)

  メッセージ: サーバーは行ベースのレプリケーションで構築されていません

- エラー: 1575 SQLSTATE: HY000 (ER_BASE64_DECODE_ERROR)

  メッセージ: Base64 文字列のデコードに失敗しました

- エラー: 1576 SQLSTATE: HY000 (ER_EVENT_RECURSION_FORBIDDEN)

  メッセージ: 本体が存在する場合、EVENT DDL ステートメントの再帰は禁止されています

- エラー: 1577 SQLSTATE: HY000 (ER_EVENTS_DB_ERROR)

  メッセージ: サーバー起動時に、イベントスケジューラによって使用されるシステムテーブルの損傷が見つかったため、続行できません

- エラー: 1578 SQLSTATE: HY000 (ER_ONLY_INTEGERS_ALLOWED)

  メッセージ: ここでは整数のみが数値として許可されます

- エラー: 1579 SQLSTATE: HY000 (ER_UNSUPORTED_LOG_ENGINE)

  メッセージ: このストレージエンジンはログテーブル '' には使用できません

- エラー: 1580 SQLSTATE: HY000 (ER_BAD_LOG_STATEMENT)

  メッセージ: ロギングが有効にされている場合は、ログテーブルを '%s' できません

- エラー: 1581 SQLSTATE: HY000 (ER_CANT_RENAME_LOG_TABLE)

  メッセージ: '%s' を名前変更できません。ロギングが有効にされている場合、ログテーブルの名前変更では 2 つのテーブルを名前変更する必要があります。ログテーブルをアーカイブテーブルに名前変更し、別のテーブルを '%s' に戻します

- エラー: 1582 SQLSTATE: 42000 (ER_WRONG_PARAMCOUNT_TO_NATIVE_FCT)

  メッセージ: ネイティブ関数 '%s' の呼び出しのパラメータ数が不正です

- エラー: 1583 SQLSTATE: 42000 (ER_WRONG_PARAMETERS_TO_NATIVE_FCT)

  メッセージ: ネイティブ関数 '%s' の呼び出しのパラメータが不正です

- エラー: 1584 SQLSTATE: 42000 (ER_WRONG_PARAMETERS_TO_STORED_FCT)

  メッセージ: ストアドファンクション '%s' の呼び出しのパラメータが不正です

- エラー: 1585 SQLSTATE: HY000 (ER_NATIVE_FCT_NAME_COLLISION)

  メッセージ: この関数 '%s' はネイティブ関数と同じ名前です

- エラー: 1586 SQLSTATE: 23000 (ER_DUP_ENTRY_WITH_KEY_NAME)

  メッセージ: '%s' はキー '%s' で重複しています。

  このエラーの書式文字列は、ER_DUP_ENTRY でも使用されます。

- エラー: 1587 SQLSTATE: HY000 (ER_BINLOG_PURGE_EMFILE)

  メッセージ: オープンしているファイルが多すぎます。コマンドを再実行してください

- エラー: 1588 SQLSTATE: HY000 (ER_EVENT_CANNOT_CREATE_IN_THE_PAST)

  メッセージ: イベント実行時間が過去であり、ON COMPLETION NOT PRESERVE が設定されています。イベントは作成後すぐにドロップされました。

- エラー: 1589 SQLSTATE: HY000 (ER_EVENT_CANNOT_ALTER_IN_THE_PAST)

  メッセージ: イベント実行時間が過去であり、ON COMPLETION NOT PRESERVE が設定されています。イベントは変更されませんでした。現在より先の時間を指定してください。

- エラー: 1590 SQLSTATE: HY000 (ER_SLAVE_INCIDENT)

  メッセージ: インシデント %s がマスターで発生しました。メッセージ: %s

- エラー: 1591 SQLSTATE: HY000 (ER_NO_PARTITION_FOR_GIVEN_VALUE_SILENT)

  メッセージ: いくつかの既存の値のパーティションがテーブルにありません

- エラー: 1592 SQLSTATE: HY000 (ER_BINLOG_UNSAFE_STATEMENT)

  メッセージ: BINLOG_FORMAT = STATEMENT のときに、ステートメント形式を使用して安全ではないステートメントがバイナリログに書き込まれました。%s

- エラー: 1593 SQLSTATE: HY000 (ER_SLAVE_FATAL_ERROR)

  メッセージ: 致命的なエラー: %s

- エラー: 1594 SQLSTATE: HY000 (ER_SLAVE_RELAY_LOG_READ_FAILURE)

メッセージ: リレーログの読み取りの失敗: %s

- エラー: 1595 SQLSTATE: HY000 (ER_SLAVE_RELAY_LOG_WRITE_FAILURE)

  メッセージ: リレーログの書き込みの失敗: %s

- エラー: 1596 SQLSTATE: HY000 (ER_SLAVE_CREATE_EVENT_FAILURE)

  メッセージ: %s の作成に失敗しました

- エラー: 1597 SQLSTATE: HY000 (ER_SLAVE_MASTER_COM_FAILURE)

  メッセージ: マスターのコマンド %s が失敗しました: %s

- エラー: 1598 SQLSTATE: HY000 (ER_BINLOG_LOGGING_IMPOSSIBLE)

  メッセージ: バイナリロギングを行うことができません。メッセージ: %s

- エラー: 1599 SQLSTATE: HY000 (ER_VIEW_NO_CREATION_CTX)

  メッセージ: ビュー `%s`.`%s` には作成コンテキストがありません

- エラー: 1600 SQLSTATE: HY000 (ER_VIEW_INVALID_CREATION_CTX)

  メッセージ: ビュー `%s`.`%s' の作成コンテキストが無効です

- エラー: 1601 SQLSTATE: HY000 (ER_SR_INVALID_CREATION_CTX)

  メッセージ: ストアドルーチン `%s`.`%s` の作成コンテキストが無効です

- エラー: 1602 SQLSTATE: HY000 (ER_TRG_CORRUPTED_FILE)

  メッセージ: テーブル `%s`.`%s` の TRG ファイルが破損しています

- エラー: 1603 SQLSTATE: HY000 (ER_TRG_NO_CREATION_CTX)

  メッセージ: テーブル `%s`.`%s` のトリガーに作成コンテキストがありません

- エラー: 1604 SQLSTATE: HY000 (ER_TRG_INVALID_CREATION_CTX)

  メッセージ: テーブル `%s`.`%s` のトリガーの作成コンテキストが無効です

- エラー: 1605 SQLSTATE: HY000 (ER_EVENT_INVALID_CREATION_CTX)

  メッセージ: イベント `%s`.`%s` の作成コンテキストが無効です

- エラー: 1606 SQLSTATE: HY000 (ER_TRG_CANT_OPEN_TABLE)

  メッセージ: トリガー `%s`.`%s` のテーブルをオープンできません

- エラー: 1607 SQLSTATE: HY000 (ER_CANT_CREATE_SROUTINE)

  メッセージ: ストアドルーチン `%s` を作成できません。警告をチェックしてください

- エラー: 1608 SQLSTATE: HY000 (ER_NEVER_USED)

  メッセージ: スレーブモードの組み合わせがあいまいです。%s

- エラー: 1609 SQLSTATE: HY000 (ER_NO_FORMAT_DESCRIPTION_EVENT_BEFORE_BINLOG_STATEMENT)

  メッセージ: タイプ `%s` の BINLOG ステートメントの前に、形式を記述する BINLOG ステートメントがありませんでした。

- エラー: 1610 SQLSTATE: HY000 (ER_SLAVE_CORRUPT_EVENT)

  メッセージ: 不正なレプリケーションイベントが検出されました

- エラー: 1611 SQLSTATE: HY000 (ER_LOAD_DATA_INVALID_COLUMN)

  メッセージ: LOAD DATA に無効なカラム参照 (%s) があります

- エラー: 1612 SQLSTATE: HY000 (ER_LOG_PURGE_NO_FILE)

  メッセージ: パージするログ %s が見つかりませんでした

- エラー: 1613 SQLSTATE: XA106 (ER_XA_RBTIMEOUT)

  メッセージ: XA_RBTIMEOUT: トランザクションブランチがロールバックされました: 時間がかかりすぎました

- エラー: 1614 SQLSTATE: XA102 (ER_XA_RBDEADLOCK)

  メッセージ: XA_RBDEADLOCK: トランザクションブランチがロールバックされました: デッドロックが検出されました

- エラー: 1615 SQLSTATE: HY000 (ER_NEED_REPREPARE)

  メッセージ: 準備済みステートメントを再度準備する必要があります

- エラー: 1616 SQLSTATE: HY000 (ER_DELAYED_NOT_SUPPORTED)

  メッセージ: DELAYED オプションはテーブル '%s' ではサポートされません

- エラー: 1617 SQLSTATE: HY000 (WARN_NO_MASTER_INFO)

  メッセージ: マスター情報構造体が存在しません

- エラー: 1618 SQLSTATE: HY000 (WARN_OPTION_IGNORED)

  メッセージ: <%s> オプションは無視されました

- エラー: 1619 SQLSTATE: HY000 (WARN_PLUGIN_DELETE_BUILTIN)

  メッセージ: 組み込みプラグインは削除できません

- エラー: 1620 SQLSTATE: HY000 (WARN_PLUGIN_BUSY)

  メッセージ: プラグインがビジー状態であり、シャットダウン時にアンインストールされます

- エラー: 1621 SQLSTATE: HY000 (ER_VARIABLE_IS_READONLY)

  メッセージ: %s 変数 '%s' は読み取り専用です。SET %s を使用して、値を割り当ててください

- エラー: 1622 SQLSTATE: HY000 (ER_WARN_ENGINE_TRANSACTION_ROLLBACK)

  メッセージ: ストレージエンジン %s は、このステートメントのロールバックをサポートしていません。トランザクションがロールバックされたため、再度開始する必要があります

- エラー: 1623 SQLSTATE: HY000 (ER_SLAVE_HEARTBEAT_FAILURE)

  メッセージ: 予期しないマスターのハートビートデータ: %s

- エラー: 1624 SQLSTATE: HY000 (ER_SLAVE_HEARTBEAT_VALUE_OUT_OF_RANGE)

  メッセージ: ハートビート間隔に要求された値は、負数であるか、許容される最大値を超えています (%s 秒)。

- エラー: 1625 SQLSTATE: HY000 (ER_NDB_REPLICATION_SCHEMA_ERROR)

  メッセージ: mysql.ndb_replication テーブルのスキーマが不良です。メッセージ: %s

- エラー: 1626 SQLSTATE: HY000 (ER_CONFLICT_FN_PARSE_ERROR)

  メッセージ: 競合関数の解析でエラーが発生しました。メッセージ: %s

- エラー: 1627 SQLSTATE: HY000 (ER_EXCEPTIONS_WRITE_ERROR)

  メッセージ: 例外テーブルへの書き込みが失敗しました。メッセージ: %s"

- エラー: 1628 SQLSTATE: HY000 (ER_TOO_LONG_TABLE_COMMENT)

  メッセージ: テーブル '%s' のコメントが長すぎます (最大 = %lu)

- エラー: 1629 SQLSTATE: HY000 (ER_TOO_LONG_FIELD_COMMENT)

  メッセージ: フィールド '%s' のコメントが長すぎます (最大 = %lu)

- エラー: 1630 SQLSTATE: 42000 (ER_FUNC_INEXISTENT_NAME_COLLISION)

  メッセージ: FUNCTION %s は存在しません。リファレンスマニュアルの「関数名の構文解析と解決」セクションを確認してください

- エラー: 1631 SQLSTATE: HY000 (ER_DATABASE_NAME)

  メッセージ: データベース

- エラー: 1632 SQLSTATE: HY000 (ER_TABLE_NAME)

  メッセージ: テーブル

- エラー: 1633 SQLSTATE: HY000 (ER_PARTITION_NAME)

  メッセージ: パーティション

- エラー: 1634 SQLSTATE: HY000 (ER_SUBPARTITION_NAME)

  メッセージ: サブパーティション

- エラー: 1635 SQLSTATE: HY000 (ER_TEMPORARY_NAME)

  メッセージ: 一時的

- エラー: 1636 SQLSTATE: HY000 (ER_RENAMED_NAME)

  メッセージ: 名前変更済み

- エラー: 1637 SQLSTATE: HY000 (ER_TOO_MANY_CONCURRENT_TRXS)

  メッセージ: アクティブな並列トランザクションが多すぎます

- エラー: 1638 SQLSTATE: HY000 (WARN_NON_ASCII_SEPARATOR_NOT_IMPLEMENTED)

  メッセージ: 非 ASCII 区切り文字引数は完全にサポートされていません

- エラー: 1639 SQLSTATE: HY000 (ER_DEBUG_SYNC_TIMEOUT)

  メッセージ: デバッグ同期ポイントの待機がタイムアウトしました

- エラー: 1640 SQLSTATE: HY000 (ER_DEBUG_SYNC_HIT_LIMIT)

  メッセージ: デバッグ同期ポイントの制限に達しました

- エラー: 1641 SQLSTATE: 42000 (ER_DUP_SIGNAL_SET)

  メッセージ: 重複した条件情報項目 '%s'

- エラー: 1642 SQLSTATE: 01000 (ER_SIGNAL_WARN)

  メッセージ: ユーザー定義の警告条件の処理がありません

- エラー: 1643 SQLSTATE: 02000 (ER_SIGNAL_NOT_FOUND)

  メッセージ: ユーザー定義の Not Found 条件の処理がありません

- エラー: 1644 SQLSTATE: HY000 (ER_SIGNAL_EXCEPTION)

  メッセージ: ユーザー定義の例外条件の処理がありません

- エラー: 1645 SQLSTATE: 0K000 (ER_RESIGNAL_WITHOUT_ACTIVE_HANDLER)

  メッセージ: ハンドラがアクティブではないときに RESIGNAL が実行されました

- エラー: 1646 SQLSTATE: HY000 (ER_SIGNAL_BAD_CONDITION_TYPE)

  メッセージ: SIGNAL/RESIGNAL では、SQLSTATE とともに定義された 1 つの CONDITION のみを使用できます

- エラー: 1647 SQLSTATE: HY000 (WARN_COND_ITEM_TRUNCATED)

  メッセージ: 条件項目 '%s' のデータが切り捨てられました

- エラー: 1648 SQLSTATE: HY000 (ER_COND_ITEM_TOO_LONG)

  メッセージ: 条件項目 '%s' のデータが長すぎます

- エラー: 1649 SQLSTATE: HY000 (ER_UNKNOWN_LOCALE)

  メッセージ: 不明なロケール: '%s'

- エラー: 1650 SQLSTATE: HY000 (ER_SLAVE_IGNORE_SERVER_IDS)

  メッセージ: 要求されたサーバー ID %d は、スレーブスタートアップオプション --replicate-same-server-id でクラッシュしました

- エラー: 1651 SQLSTATE: HY000 (ER_QUERY_CACHE_DISABLED)

  メッセージ: クエリーキャッシュが無効にされています。サーバーを query_cache_type=1 で再起動して有効にしてください

- エラー: 1652 SQLSTATE: HY000 (ER_SAME_NAME_PARTITION_FIELD)

  メッセージ: 重複したパーティションフィールド名 '%s'

- エラー: 1653 SQLSTATE: HY000 (ER_PARTITION_COLUMN_LIST_ERROR)

  メッセージ: パーティショニングのカラムリストの使用方法に一貫性がありません

- エラー: 1654 SQLSTATE: HY000 (ER_WRONG_TYPE_COLUMN_VALUE_ERROR)

  メッセージ: 不正な型のパーティションカラム値です

- エラー: 1655 SQLSTATE: HY000 (ER_TOO_MANY_PARTITION_FUNC_FIELDS_ERROR)

  メッセージ: '%s' のフィールドが多すぎます

- エラー: 1656 SQLSTATE: HY000 (ER_MAXVALUE_IN_VALUES_IN)

  メッセージ: MAXVALUE は VALUES IN の値として使用できません

- エラー: 1657 SQLSTATE: HY000 (ER_TOO_MANY_VALUES_ERROR)

  メッセージ: このタイプの %s パーティショニングには複数の値を指定できません

- エラー: 1658 SQLSTATE: HY000 (ER_ROW_SINGLE_PARTITION_FIELD_ERROR)

  メッセージ: VALUES IN の行の式は、複数フィールドカラムのパーティショニングにのみ許可されます

- エラー: 1659 SQLSTATE: HY000 (ER_FIELD_TYPE_NOT_ALLOWED_AS_PARTITION_FIELD)

  メッセージ: フィールド '%s' は、このタイプのパーティショニングに許可される型ではありません

- エラー: 1660 SQLSTATE: HY000 (ER_PARTITION_FIELDS_TOO_LONG)

  メッセージ: パーティショニングフィールドの合計長が長すぎます

- エラー: 1661 SQLSTATE: HY000 (ER_BINLOG_ROW_ENGINE_AND_STMT_ENGINE)

  メッセージ: ステートメントを実行できません: 行対応ではないエンジンおよびステートメント対応ではないエンジンの両方が呼び出されたため、バイナリログに書き込むことができません。

- エラー: 1662 SQLSTATE: HY000 (ER_BINLOG_ROW_MODE_AND_STMT_ENGINE)

  メッセージ: ステートメントを実行できません: BINLOG_FORMAT = ROW ですが、少なくとも 1 つのテーブルがステートメントベースのロギングに制限されているストレージエンジンを使用しているため、バイナリログに書き込むことができません。

- エラー: 1663 SQLSTATE: HY000 (ER_BINLOG_UNSAFE_AND_STMT_ENGINE)

  メッセージ: ステートメントを実行できません: ストレージエンジンがステートメントベースのロギングに制限されていて、BINLOG_FORMAT = MIXED であり、ステートメントが安全ではないため、バイナリログに書き込むことができません。%s

- エラー: 1664 SQLSTATE: HY000 (ER_BINLOG_ROW_INJECTION_AND_STMT_ENGINE)

メッセージ: ステートメントを実行できません: ステートメントが行フォーマットであり、少なくとも 1 つのテーブルがステートメントベースのロギングに制限されているストレージエンジンを使用しているため、バイナリログに書き込むことができません。

- エラー: 1665 SQLSTATE: HY000 (ER_BINLOG_STMT_MODE_AND_ROW_ENGINE)

  メッセージ: ステートメントを実行できません: BINLOG_FORMAT = STATEMENT ですが、少なくとも 1 つのテーブルが行ベースのロギングに制限されているストレージエンジンを使用しているため、バイナリログに書き込むことができません。%s

- エラー: 1666 SQLSTATE: HY000 (ER_BINLOG_ROW_INJECTION_AND_STMT_MODE)

  メッセージ: ステートメントを実行できません: ステートメントが行フォーマットであり、BINLOG_FORMAT = STATEMENT であるため、バイナリログに書き込むことができません。

- エラー: 1667 SQLSTATE: HY000 (ER_BINLOG_MULTIPLE_ENGINES_AND_SELF_LOGGING_ENGINE)

  メッセージ: ステートメントを実行できません: 複数のエンジンが関係していて、少なくとも 1 つのエンジンがセルフロギングであるため、バイナリログに書き込むことができません。

- エラー: 1668 SQLSTATE: HY000 (ER_BINLOG_UNSAFE_LIMIT)

  メッセージ: このステートメントは LIMIT 句を使用しているため安全ではありません。これは、含められる一連の行を予測できないため、安全ではありません。

- エラー: 1669 SQLSTATE: HY000 (ER_BINLOG_UNSAFE_INSERT_DELAYED)

  メッセージ: このステートメントは INSERT DELAYED を使用しているため安全ではありません。これは、行がいつ挿入されるかを予測できないため、安全ではありません。

- エラー: 1670 SQLSTATE: HY000 (ER_BINLOG_UNSAFE_SYSTEM_TABLE)

  メッセージ: このステートメントは、一般的なログ、スロークエリーログ、または performance_schema テーブルを使用しているため、安全ではありません。これは、システムテーブルがスレーブ間で異なる可能性があるため、安全ではありません。

- エラー: 1671 SQLSTATE: HY000 (ER_BINLOG_UNSAFE_AUTOINC_COLUMNS)

  メッセージ: ステートメントは、AUTO_INCREMENT カラムに挿入するトリガーまたはストアドファンクションを呼び出すため、安全ではありません。挿入される値を正しくログに記録できません。

- エラー: 1672 SQLSTATE: HY000 (ER_BINLOG_UNSAFE_UDF)

  メッセージ: ステートメントは、スレーブに同じ値を返さない可能性がある UDF を使用しているため、安全ではありません。

- エラー: 1673 SQLSTATE: HY000 (ER_BINLOG_UNSAFE_SYSTEM_VARIABLE)

  メッセージ: ステートメントは、スレーブで異なる値が保持されている可能性があるシステム変数を使用しているため、安全ではありません。

- エラー: 1674 SQLSTATE: HY000 (ER_BINLOG_UNSAFE_SYSTEM_FUNCTION)

  メッセージ: ステートメントは、スレーブに異なる値を返す可能性があるシステム関数を使用しているため、安全ではありません。

- エラー: 1675 SQLSTATE: HY000 (ER_BINLOG_UNSAFE_NONTRANS_AFTER_TRANS)

  メッセージ: ステートメントは、同じトランザクション内でトランザクションテーブルにアクセスしたあとに非トランザクションテーブルにアクセスするため、安全ではありません。

- エラー: 1676 SQLSTATE: HY000 (ER_MESSAGE_AND_STATEMENT)

  メッセージ: %s ステートメント: %s

- エラー: 1677 SQLSTATE: HY000 (ER_SLAVE_CONVERSION_FAILED)

  メッセージ: カラム %d (テーブル '%s.%s') は '%s' 型から '%s' 型に変換できません

- エラー: 1678 SQLSTATE: HY000 (ER_SLAVE_CANT_CREATE_CONVERSION)

メッセージ: テーブル '%s.%s' の変換テーブルを作成できません

- エラー: 1679 SQLSTATE: HY000 (ER_INSIDE_TRANSACTION_PREVENTS_SWITCH_BINLOG_FORMAT)

  メッセージ: トランザクション内では @@session.binlog_format は変更できません

- エラー: 1680 SQLSTATE: HY000 (ER_PATH_LENGTH)

  メッセージ: %s に指定されたパスが長すぎます。

- エラー: 1681 SQLSTATE: HY000 (ER_WARN_DEPRECATED_SYNTAX_NO_REPLACEMENT)

  メッセージ: '%s' は将来のリリースで廃止予定です。

- エラー: 1682 SQLSTATE: HY000 (ER_WRONG_NATIVE_TABLE_STRUCTURE)

  メッセージ: ネイティブテーブル '%s'.'%s' の構造は間違っています

- エラー: 1683 SQLSTATE: HY000 (ER_WRONG_PERFSCHEMA_USAGE)

  メッセージ: performance_schema の使用方法が無効です。

- エラー: 1684 SQLSTATE: HY000 (ER_WARN_I_S_SKIPPED_TABLE)

  メッセージ: 定義が並列 DDL ステートメントによって変更されているため、テーブル '%s'.'%s' はスキップされました。

- エラー: 1685 SQLSTATE: HY000 (ER_INSIDE_TRANSACTION_PREVENTS_SWITCH_BINLOG_DIRECT)

  メッセージ: トランザクション内では @@session.binlog_direct_non_transactional_updates は変更できません

- エラー: 1686 SQLSTATE: HY000 (ER_STORED_FUNCTION_PREVENTS_SWITCH_BINLOG_DIRECT)

  メッセージ: ストアドファンクションまたはトリガー内では Binlog ダイレクトフラグは変更できません

- エラー: 1687 SQLSTATE: 42000 (ER_SPATIAL_MUST_HAVE_GEOM_COL)

  メッセージ: SPATIAL インデックスには、幾何型カラムのみを含めることができます

- エラー: 1688 SQLSTATE: HY000 (ER_TOO_LONG_INDEX_COMMENT)

  メッセージ: インデックス '%s' のコメントが長すぎます (最大 = %lu)

- エラー: 1689 SQLSTATE: HY000 (ER_LOCK_ABORTED)

  メッセージ: 保留中の排他ロックのために、ロックの待機が中止されました

- エラー: 1690 SQLSTATE: 22003 (ER_DATA_OUT_OF_RANGE)

  メッセージ: %s 値は '%s' で範囲外となっています

- エラー: 1691 SQLSTATE: HY000 (ER_WRONG_SPVAR_TYPE_IN_LIMIT)

  メッセージ: LIMIT 句に整数ベースではない型の変数があります

- エラー: 1692 SQLSTATE: HY000 (ER_BINLOG_UNSAFE_MULTIPLE_ENGINES_AND_SELF_LOGGING_ENGINE)

  メッセージ: 1 つのステートメントにセルフロギングと非セルフロギングのエンジンを混在させることは、安全ではありません。

- エラー: 1693 SQLSTATE: HY000 (ER_BINLOG_UNSAFE_MIXED_STATEMENT)

  メッセージ: ステートメントが非トランザクションテーブルのほかにトランザクションテーブルまたは一時テーブルにアクセスしており、それらのいずれかに書き込んでいます。

- エラー: 1694 SQLSTATE: HY000 (ER_INSIDE_TRANSACTION_PREVENTS_SWITCH_SQL_LOG_BIN)

  メッセージ: トランザクション内では @@session.sql_log_bin は変更できません

- エラー: 1695 SQLSTATE: HY000 (ER_STORED_FUNCTION_PREVENTS_SWITCH_SQL_LOG_BIN)

メッセージ: ストアドファンクションまたはトリガー内では sql_log_bin は変更できません

- エラー: 1696 SQLSTATE: HY000 (ER_FAILED_READ_FROM_PAR_FILE)

  メッセージ: .par ファイルからの読み取りに失敗しました

- エラー: 1697 SQLSTATE: HY000 (ER_VALUES_IS_NOT_INT_TYPE_ERROR)

  メッセージ: パーティション '%s' の VALUES 値は INT 型である必要があります

  ER_VALUES_IS_NOT_INT_TYPE_ERROR は 5.6.1 で導入されました。

- エラー: 1698 SQLSTATE: 28000 (ER_ACCESS_DENIED_NO_PASSWORD_ERROR)

  メッセージ: ユーザー '%s'@'%s' のアクセスは拒否されました。

  ER_ACCESS_DENIED_NO_PASSWORD_ERROR は 5.6.1 で導入されました。

- エラー: 1699 SQLSTATE: HY000 (ER_SET_PASSWORD_AUTH_PLUGIN)

  メッセージ: プラグインを使用して認証を行うユーザーの場合、SET PASSWORD は意味を持ちません

  ER_SET_PASSWORD_AUTH_PLUGIN は 5.6.1 で導入されました。

- エラー: 1700 SQLSTATE: HY000 (ER_GRANT_PLUGIN_USER_EXISTS)

  メッセージ: IDENTIFIED WITH を指定した GRANT は、ユーザー %-.*s がすでに存在するため不正です

  ER_GRANT_PLUGIN_USER_EXISTS は 5.6.1 で導入されました。

- エラー: 1701 SQLSTATE: 42000 (ER_TRUNCATE_ILLEGAL_FK)

  メッセージ: 外部キー制約 (%s) で参照されているテーブルを切り捨てることはできません

  ER_TRUNCATE_ILLEGAL_FK は 5.6.1 で導入されました。

- エラー: 1702 SQLSTATE: HY000 (ER_PLUGIN_IS_PERMANENT)

  メッセージ: プラグイン '%s' は force_plus_permanent であるため、アンロードできません

  ER_PLUGIN_IS_PERMANENT は 5.6.1 で導入されました。

- エラー: 1703 SQLSTATE: HY000 (ER_SLAVE_HEARTBEAT_VALUE_OUT_OF_RANGE_MIN)

  メッセージ: ハートビート間隔に要求された値は 1 ミリ秒より小さい値です。値は 0 にリセットされ、ハートビートが実質的に無効にされることを意味します。

  ER_SLAVE_HEARTBEAT_VALUE_OUT_OF_RANGE_MIN は 5.6.1 で導入されました。

- エラー: 1704 SQLSTATE: HY000 (ER_SLAVE_HEARTBEAT_VALUE_OUT_OF_RANGE_MAX)

  メッセージ: ハートビート間隔に要求された値は、`slave_net_timeout' 秒を超えています。この間隔に適切な値は、タイムアウトより小さい値です。

  ER_SLAVE_HEARTBEAT_VALUE_OUT_OF_RANGE_MAX は 5.6.1 で導入されました。

- エラー: 1705 SQLSTATE: HY000 (ER_STMT_CACHE_FULL)

  メッセージ: 複数行のステートメントが、ストレージに 'max_binlog_stmt_cache_size' バイトより大きい領域を要求しました。この mysqld 変数の値を増やして再試行してください

  ER_STMT_CACHE_FULL は 5.6.1 で導入されました。

- エラー: 1706 SQLSTATE: HY000 (ER_MULTI_UPDATE_KEY_CONFLICT)

  メッセージ: このテーブルは '%s' および '%s' として更新されるため、主キー/パーティションキーの更新は許可されません。

  ER_MULTI_UPDATE_KEY_CONFLICT は 5.6.2 で導入されました。

- エラー: 1707 SQLSTATE: HY000 (ER_TABLE_NEEDS_REBUILD)

メッセージ: テーブルを再構築する必要があります。"ALTER TABLE `%s` FORCE" を実行するか、ダンプ/リロードして修復してください。

ER_TABLE_NEEDS_REBUILD は 5.6.3 で導入されました。

- エラー: 1708 SQLSTATE: HY000 (WARN_OPTION_BELOW_LIMIT)

  メッセージ: '%s' の値は '%s' の値より小さくなるようにしてください

  WARN_OPTION_BELOW_LIMIT は 5.6.3 で導入されました。

- エラー: 1709 SQLSTATE: HY000 (ER_INDEX_COLUMN_TOO_LONG)

  メッセージ: インデックスのカラムサイズが大きすぎます。最大のカラムサイズは %lu バイトです。

  ER_INDEX_COLUMN_TOO_LONG は 5.6.3 で導入されました。

- エラー: 1710 SQLSTATE: HY000 (ER_ERROR_IN_TRIGGER_BODY)

  メッセージ: トリガー '%s' は本体にエラーがあります: '%s'

  ER_ERROR_IN_TRIGGER_BODY は 5.6.3 で導入されました。

- エラー: 1711 SQLSTATE: HY000 (ER_ERROR_IN_UNKNOWN_TRIGGER_BODY)

  メッセージ: 不明なトリガーの本体にエラーがあります: '%s'

  ER_ERROR_IN_UNKNOWN_TRIGGER_BODY は 5.6.3 で導入されました。

- エラー: 1712 SQLSTATE: HY000 (ER_INDEX_CORRUPT)

  メッセージ: インデックス %s は破損しています

  ER_INDEX_CORRUPT は 5.6.3 で導入されました。

- エラー: 1713 SQLSTATE: HY000 (ER_UNDO_RECORD_TOO_BIG)

  メッセージ: Undo ログレコードが大きすぎます。

  ER_UNDO_RECORD_TOO_BIG は 5.6.4 で導入されました。

- エラー: 1714 SQLSTATE: HY000 (ER_BINLOG_UNSAFE_INSERT_IGNORE_SELECT)

  メッセージ: SELECT によって取得される行の順序によって無視される行 (ある場合) が決まるため、INSERT IGNORE... SELECT は安全ではありません。この順序は予測できず、マスターとスレーブで異なる可能性があります。

  ER_BINLOG_UNSAFE_INSERT_IGNORE_SELECT は 5.6.4 で導入されました。

- エラー: 1715 SQLSTATE: HY000 (ER_BINLOG_UNSAFE_INSERT_SELECT_UPDATE)

  メッセージ: SELECT によって取得される行の順序によって更新される行 (ある場合) が決まるため、INSERT... SELECT... ON DUPLICATE KEY UPDATE は安全ではありません。この順序は予測できず、マスターとスレーブで異なる可能性があります。

  ER_BINLOG_UNSAFE_INSERT_SELECT_UPDATE は 5.6.4 で導入されました。

- エラー: 1716 SQLSTATE: HY000 (ER_BINLOG_UNSAFE_REPLACE_SELECT)

  メッセージ: SELECT によって取得される行の順序によって置換される行 (ある場合) が決まるため、REPLACE... SELECT は安全ではありません。この順序は予測できず、マスターとスレーブで異なる可能性があります。

  ER_BINLOG_UNSAFE_REPLACE_SELECT は 5.6.4 で導入されました。

- エラー: 1717 SQLSTATE: HY000 (ER_BINLOG_UNSAFE_CREATE_IGNORE_SELECT)

  メッセージ: SELECT によって取得される行の順序によって無視される行 (ある場合) が決まるため、CREATE... IGNORE SELECT は安全ではありません。この順序は予測できず、マスターとスレーブで異なる可能性があります。

ER_BINLOG_UNSAFE_CREATE_IGNORE_SELECT は 5.6.4 で導入されました。

- エラー: 1718 SQLSTATE: HY000 (ER_BINLOG_UNSAFE_CREATE_REPLACE_SELECT)

  メッセージ: SELECT によって取得される行の順序によって置換される行 (ある場合) が決まるため、CREATE... REPLACE SELECT は安全ではありません。この順序は予測できず、マスターとスレーブで異なる可能性があります。

  ER_BINLOG_UNSAFE_CREATE_REPLACE_SELECT は 5.6.4 で導入されました。

- エラー: 1719 SQLSTATE: HY000 (ER_BINLOG_UNSAFE_UPDATE_IGNORE)

  メッセージ: 行が更新される順序によって無視される行 (ある場合) が決まるため、UPDATE IGNORE は安全ではありません。この順序は予測できず、マスターとスレーブで異なる可能性があります。

  ER_BINLOG_UNSAFE_UPDATE_IGNORE は 5.6.4 で導入されました。

- エラー: 1720 SQLSTATE: HY000 (ER_PLUGIN_NO_UNINSTALL)

  メッセージ: プラグイン '%s' は動的にアンインストールできないプラグインとしてマークされています。サーバーを停止してからアンインストールする必要があります。

  ER_PLUGIN_NO_UNINSTALL は 5.6.4 で導入されました。

- エラー: 1721 SQLSTATE: HY000 (ER_CANT_LOCK_RPL_INFO_TABLE)

  メッセージ: rpl 情報テーブルに対してロックは使用できません。

  ER_CANT_LOCK_RPL_INFO_TABLE は 5.6.1 で導入され、5.6.3 で削除されました。

- エラー: 1721 SQLSTATE: HY000 (ER_PLUGIN_NO_INSTALL)

  メッセージ: プラグイン '%s' は動的にインストールできないプラグインとしてマークされています。サーバーを停止してからインストールする必要があります。

  ER_PLUGIN_NO_INSTALL は 5.6.4 で導入されました。

- エラー: 1722 SQLSTATE: HY000 (ER_BINLOG_UNSAFE_WRITE_AUTOINC_SELECT)

  メッセージ: 行が取得される順序によって書き込まれる行 (ある場合) が決まるため、別のテーブルから選択したあとに自動インクリメントカラムを持つテーブルに書き込むステートメントは安全ではありません。この順序は予測できず、マスターとスレーブで異なる可能性があります。

  ER_BINLOG_UNSAFE_WRITE_AUTOINC_SELECT は 5.6.5 で導入されました。

- エラー: 1723 SQLSTATE: HY000 (ER_BINLOG_UNSAFE_CREATE_SELECT_AUTOINC)

  メッセージ: SELECT によって取得される行の順序によって挿入される行 (ある場合) が決まるため、自動インクリメントカラムを持つテーブルに対して CREATE TABLE... SELECT... を実行することは安全ではありません。この順序は予測できず、マスターとスレーブで異なる可能性があります。

  ER_BINLOG_UNSAFE_CREATE_SELECT_AUTOINC は 5.6.5 で導入されました。

- エラー: 1724 SQLSTATE: HY000 (ER_BINLOG_UNSAFE_INSERT_TWO_KEYS)

  メッセージ: 複数の UNIQUE KEY を持つテーブルに対して INSERT... ON DUPLICATE KEY UPDATE を実行するのは安全ではありません

  ER_BINLOG_UNSAFE_INSERT_TWO_KEYS は 5.6.6 で導入されました。

- エラー: 1725 SQLSTATE: HY000 (ER_TABLE_IN_FK_CHECK)

  メッセージ: テーブルは外部キーチェックに使用されています。

  ER_TABLE_IN_FK_CHECK は 5.6.6 で導入されました。

- エラー: 1726 SQLSTATE: HY000 (ER_UNSUPPORTED_ENGINE)

  メッセージ: ストレージエンジン '%s' はシステムテーブルをサポートしていません。[%s.%s]

ER_UNSUPPORTED_ENGINE は 5.6.6 で導入されました。

- エラー: 1727 SQLSTATE: HY000 (ER_BINLOG_UNSAFE_AUTOINC_NOT_FIRST)

  メッセージ: 構成された主キーの最初の部分ではない自動インクリメントフィールドに INSERT を実行するのは安全ではありません。

  ER_BINLOG_UNSAFE_AUTOINC_NOT_FIRST は 5.6.6 で導入されました。

- エラー: 1728 SQLSTATE: HY000 (ER_CANNOT_LOAD_FROM_TABLE_V2)

  メッセージ: %s.%s からロードできません。テーブルが破損している可能性があります

- エラー: 1729 SQLSTATE: HY000 (ER_NO_SUCH_PARTITION)

  メッセージ: パーティション '%s' は存在しません。

  ER_NO_SUCH_PARTITION は 5.6.2 で導入され、5.6.5 で削除されました。

- エラー: 1729 SQLSTATE: HY000 (ER_MASTER_DELAY_VALUE_OUT_OF_RANGE)

  メッセージ: マスター遅延に要求された値 %s が最大値 %u を超えています

- エラー: 1730 SQLSTATE: HY000 (ER_ONLY_FD_AND_RBR_EVENTS_ALLOWED_IN_BINLOG_STATEMENT)

  メッセージ: BINLOG ステートメントでは、Format_description_log_event および行イベントのみが許可されます (ただし、%s が提供されました)

- エラー: 1731 SQLSTATE: HY000 (ER_PARTITION_EXCHANGE_DIFFERENT_OPTION)

  メッセージ: パーティションとテーブルで属性 '%s' が一致しません

- エラー: 1732 SQLSTATE: HY000 (ER_PARTITION_EXCHANGE_PART_TABLE)

  メッセージ: パーティションとやり取りするテーブルがパーティション化されています: '%s'

- エラー: 1733 SQLSTATE: HY000 (ER_PARTITION_EXCHANGE_TEMP_TABLE)

  メッセージ: パーティションとやり取りするテーブルが一時テーブルです: '%s'

- エラー: 1734 SQLSTATE: HY000 (ER_PARTITION_INSTEAD_OF_SUBPARTITION)

  メッセージ: サブパーティション化されたテーブルです。パーティションではなくサブパーティションを使用してください

- エラー: 1735 SQLSTATE: HY000 (ER_UNKNOWN_PARTITION)

  メッセージ: 不明なパーティション '%s' がテーブル '%s' に存在します

- エラー: 1736 SQLSTATE: HY000 (ER_TABLES_DIFFERENT_METADATA)

  メッセージ: テーブルの定義が異なります

- エラー: 1737 SQLSTATE: HY000 (ER_ROW_DOES_NOT_MATCH_PARTITION)

  メッセージ: パーティションと一致しない行が見つかりました

- エラー: 1738 SQLSTATE: HY000 (ER_BINLOG_CACHE_SIZE_GREATER_THAN_MAX)

  メッセージ: オプション binlog_cache_size (%lu) に max_binlog_cache_size (%lu) よりも大きい値が設定されています。binlog_cache_size に max_binlog_cache_size と等しい値が設定されました。

  ER_BINLOG_CACHE_SIZE_GREATER_THAN_MAX は 5.6.1 で導入されました。

- エラー: 1739 SQLSTATE: HY000 (ER_WARN_INDEX_NOT_APPLICABLE)

  メッセージ: %s アクセスはインデックス '%s' に対して使用できません。これは、フィールド '%s' の型または照合順序の変換が原因です

  ER_WARN_INDEX_NOT_APPLICABLE は 5.6.1 で導入されました。

- エラー: 1740 SQLSTATE: HY000 (ER_PARTITION_EXCHANGE_FOREIGN_KEY)

  メッセージ: パーティションとやり取りするテーブルに外部キー参照があります: '%s'

  ER_PARTITION_EXCHANGE_FOREIGN_KEY は 5.6.1 で導入されました。

- エラー: 1741 SQLSTATE: HY000 (ER_NO_SUCH_KEY_VALUE)

  メッセージ: キー値 '%s' がテーブル '%s.%s' で見つかりませんでした

  ER_NO_SUCH_KEY_VALUE は 5.6.1 で導入されました。

- エラー: 1742 SQLSTATE: HY000 (ER_RPL_INFO_DATA_TOO_LONG)

  メッセージ: カラム '%s' のデータが大きすぎます

  ER_RPL_INFO_DATA_TOO_LONG は 5.6.1 で導入されました。

- エラー: 1743 SQLSTATE: HY000 (ER_NETWORK_READ_EVENT_CHECKSUM_FAILURE)

  メッセージ: ネットワークから読み取り中に、レプリケーションイベントのチェックサム検証が失敗しました。

  ER_NETWORK_READ_EVENT_CHECKSUM_FAILURE は 5.6.1 で導入されました。

- エラー: 1744 SQLSTATE: HY000 (ER_BINLOG_READ_EVENT_CHECKSUM_FAILURE)

  メッセージ: ログファイルから読み取り中に、レプリケーションイベントのチェックサム検証が失敗しました。

  ER_BINLOG_READ_EVENT_CHECKSUM_FAILURE は 5.6.1 で導入されました。

- エラー: 1745 SQLSTATE: HY000 (ER_BINLOG_STMT_CACHE_SIZE_GREATER_THAN_MAX)

  メッセージ: オプション binlog_stmt_cache_size (%lu) に max_binlog_stmt_cache_size (%lu) より大きい値が設定されています。binlog_stmt_cache_size に max_binlog_stmt_cache_size と等しい値が設定されました。

  ER_BINLOG_STMT_CACHE_SIZE_GREATER_THAN_MAX は 5.6.1 で導入されました。

- エラー: 1746 SQLSTATE: HY000 (ER_CANT_UPDATE_TABLE_IN_CREATE_TABLE_SELECT)

  メッセージ: テーブル '%s' は '%s' の作成中に更新することはできません。

  ER_CANT_UPDATE_TABLE_IN_CREATE_TABLE_SELECT は 5.6.2 で導入されました。

- エラー: 1747 SQLSTATE: HY000 (ER_PARTITION_CLAUSE_ON_NONPARTITIONED)

  メッセージ: パーティション化されていないテーブルに PARTITION () 句があります

  ER_PARTITION_CLAUSE_ON_NONPARTITIONED は 5.6.2 で導入されました。

- エラー: 1748 SQLSTATE: HY000 (ER_ROW_DOES_NOT_MATCH_GIVEN_PARTITION_SET)

  メッセージ: 指定されたパーティションセットと一致しない行が見つかりました

  ER_ROW_DOES_NOT_MATCH_GIVEN_PARTITION_SET は 5.6.2 で導入されました。

- エラー: 1749 SQLSTATE: HY000 (ER_NO_SUCH_PARTITION__UNUSED)

  メッセージ: パーティション '%s' は存在しません。

  ER_NO_SUCH_PARTITION__UNUSED は 5.6.6 で導入されました。

- エラー: 1750 SQLSTATE: HY000 (ER_CHANGE_RPL_INFO_REPOSITORY_FAILURE)

  メッセージ: レプリケーションリポジトリのタイプの変更中に障害が発生しました: %s。

  ER_CHANGE_RPL_INFO_REPOSITORY_FAILURE は 5.6.3 で導入されました。

- エラー: 1751 SQLSTATE: HY000 (ER_WARNING_NOT_COMPLETE_ROLLBACK_WITH_CREATED_TEMP_TABLE)

  メッセージ: 一部の一時テーブルの作成をロールバックできませんでした。

ER_WARNING_NOT_COMPLETE_ROLLBACK_WITH_CREATED_TEMP_TABLE は 5.6.3 で導入されました。

- エラー: 1752 SQLSTATE: HY000 (ER_WARNING_NOT_COMPLETE_ROLLBACK_WITH_DROPPED_TEMP_TABLE)

  メッセージ: 一部の一時テーブルがドロップされましたが、それらの操作をロールバックできませんでした。

  ER_WARNING_NOT_COMPLETE_ROLLBACK_WITH_DROPPED_TEMP_TABLE は 5.6.3 で導入されました。

- エラー: 1753 SQLSTATE: HY000 (ER_MTS_FEATURE_IS_NOT_SUPPORTED)

  メッセージ: %s はマルチスレッドスレーブモードではサポートされません。%s

  ER_MTS_FEATURE_IS_NOT_SUPPORTED は 5.6.3 で導入されました。

- エラー: 1754 SQLSTATE: HY000 (ER_MTS_UPDATED_DBS_GREATER_MAX)

  メッセージ: 変更されたデータベース数が最大値の %d を超えました。データベース名はレプリケーションイベントのメタデータに含められません。

  ER_MTS_UPDATED_DBS_GREATER_MAX は 5.6.3 で導入されました。

- エラー: 1755 SQLSTATE: HY000 (ER_MTS_CANT_PARALLEL)

  メッセージ: 並列モードでは現在のイベントグループを実行できません。イベント %s、リレーログ名 %s、位置 %s が発生し、並列モードでのこのイベントグループの実行が妨げられています。理由: %s。

  ER_MTS_CANT_PARALLEL は 5.6.3 で導入されました。

- エラー: 1756 SQLSTATE: HY000 (ER_MTS_INCONSISTENT_DATA)

  メッセージ: %s

  ER_MTS_INCONSISTENT_DATA は 5.6.3 で導入されました。

- エラー: 1757 SQLSTATE: HY000 (ER_FULLTEXT_NOT_SUPPORTED_WITH_PARTITIONING)

  メッセージ: FULLTEXT インデックスは、パーティション化されたテーブルではサポートされません。

  ER_FULLTEXT_NOT_SUPPORTED_WITH_PARTITIONING は 5.6.4 で導入されました。

- エラー: 1758 SQLSTATE: 35000 (ER_DA_INVALID_CONDITION_NUMBER)

  メッセージ: 無効な条件数

  ER_DA_INVALID_CONDITION_NUMBER は 5.6.4 で導入されました。

- エラー: 1759 SQLSTATE: HY000 (ER_INSECURE_PLAIN_TEXT)

  メッセージ: SSL/TLS を使用せずに平文でパスワードを送信することは、セキュリティー保護に大きな問題があります。

  ER_INSECURE_PLAIN_TEXT は 5.6.4 で導入されました。

- エラー: 1760 SQLSTATE: HY000 (ER_INSECURE_CHANGE_MASTER)

  メッセージ: MySQL のユーザー名またはパスワードの情報をマスター情報リポジトリに格納するとセキュリティー保護されないためお勧めしません。START SLAVE に USER および PASSWORD 接続オプションを使用することを検討してください。詳細は、MySQL マニュアルの「START SLAVE 構文」を参照してください。

  ER_INSECURE_CHANGE_MASTER は 5.6.4 で導入されました。

- エラー: 1761 SQLSTATE: 23000 (ER_FOREIGN_DUPLICATE_KEY_WITH_CHILD_INFO)

  メッセージ: テーブル '%s'、レコード '%s' の外部キー制約によって、テーブル '%s'、キー '%s' に重複したエントリが作成されることがあります

  ER_FOREIGN_DUPLICATE_KEY_WITH_CHILD_INFO は 5.6.4 で導入されました。

- エラー: 1762 SQLSTATE: 23000 (ER_FOREIGN_DUPLICATE_KEY_WITHOUT_CHILD_INFO)

メッセージ: テーブル '%s'、レコード '%s' の外部キー制約によって、子テーブルに重複したエントリが作成されることがあります

ER_FOREIGN_DUPLICATE_KEY_WITHOUT_CHILD_INFO は 5.6.4 で導入されました。

- エラー: 1763 SQLSTATE: HY000 (ER_SQLTHREAD_WITH_SECURE_SLAVE)

  メッセージ: スレーブ SQL スレッドのみが開始されている場合、認証オプションは設定できません。

  ER_SQLTHREAD_WITH_SECURE_SLAVE は 5.6.4 で導入されました。

- エラー: 1764 SQLSTATE: HY000 (ER_TABLE_HAS_NO_FT)

  メッセージ: このクエリーをサポートするための FULLTEXT インデックスがテーブルにありません

  ER_TABLE_HAS_NO_FT は 5.6.4 で導入されました。

- エラー: 1765 SQLSTATE: HY000 (ER_VARIABLE_NOT_SETTABLE_IN_SF_OR_TRIGGER)

  メッセージ: システム変数 %s は、ストアドファンクションまたはトリガーでは設定できません。

  ER_VARIABLE_NOT_SETTABLE_IN_SF_OR_TRIGGER は 5.6.5 で導入されました。

- エラー: 1766 SQLSTATE: HY000 (ER_VARIABLE_NOT_SETTABLE_IN_TRANSACTION)

  メッセージ: 実行中のトランザクションがある場合、システム変数 %s は設定できません。

  ER_VARIABLE_NOT_SETTABLE_IN_TRANSACTION は 5.6.5 で導入されました。

- エラー: 1767 SQLSTATE: HY000 (ER_GTID_NEXT_IS_NOT_IN_GTID_NEXT_LIST)

  メッセージ: システム変数 @@SESSION.GTID_NEXT の値が、@@SESSION.GTID_NEXT_LIST にリストされていない値 %s です。

  ER_GTID_NEXT_IS_NOT_IN_GTID_NEXT_LIST は 5.6.5 で導入されました。

- エラー: 1768 SQLSTATE: HY000 (ER_CANT_CHANGE_GTID_NEXT_IN_TRANSACTION_WHEN_GTID_NEXT_LIST_IS_NULL)

  メッセージ: システム変数 @@SESSION.GTID_NEXT は、トランザクション内で変更できません。

  ER_CANT_CHANGE_GTID_NEXT_IN_TRANSACTION_WHEN_GTID_NEXT_LIST_IS_NULL は 5.6.5 で導入されました。

- エラー: 1769 SQLSTATE: HY000 (ER_SET_STATEMENT_CANNOT_INVOKE_FUNCTION)

  メッセージ: ステートメント 'SET %s' は、ストアドファンクションを呼び出すことはできません。

  ER_SET_STATEMENT_CANNOT_INVOKE_FUNCTION は 5.6.5 で導入されました。

- エラー: 1770 SQLSTATE: HY000 (ER_GTID_NEXT_CANT_BE_AUTOMATIC_IF_GTID_NEXT_LIST_IS_NON_NULL)

  メッセージ: @@SESSION.GTID_NEXT_LIST が NULL ではない場合、システム変数 @@SESSION.GTID_NEXT を 'AUTOMATIC' にすることはできません。

  ER_GTID_NEXT_CANT_BE_AUTOMATIC_IF_GTID_NEXT_LIST_IS_NON_NULL は 5.6.5 で導入されました。

- エラー: 1771 SQLSTATE: HY000 (ER_SKIPPING_LOGGED_TRANSACTION)

  メッセージ: すでに実行されて口グに記録されているため、トランザクション %s はスキップされました。

  ER_SKIPPING_LOGGED_TRANSACTION は 5.6.5 で導入されました。

- エラー: 1772 SQLSTATE: HY000 (ER_MALFORMED_GTID_SET_SPECIFICATION)

  メッセージ: GTID セットの指定 '%s' は誤った形式です。

  ER_MALFORMED_GTID_SET_SPECIFICATION は 5.6.5 で導入されました。

- エラー: 1773 SQLSTATE: HY000 (ER_MALFORMED_GTID_SET_ENCODING)

  メッセージ: GTID セットのエンコードが誤った形式です。

  ER_MALFORMED_GTID_SET_ENCODING は 5.6.5 で導入されました。

- エラー: 1774 SQLSTATE: HY000 (ER_MALFORMED_GTID_SPECIFICATION)

  メッセージ: 誤った形式の GTID 指定 '%s'。

  ER_MALFORMED_GTID_SPECIFICATION は 5.6.5 で導入されました。

- エラー: 1775 SQLSTATE: HY000 (ER_GNO_EXHAUSTED)

  メッセージ: グローバルトランザクション識別子を生成できません: 整数コンポーネントが最大値に達しました。新しい server_uuid でサーバーを再起動してください。

  ER_GNO_EXHAUSTED は 5.6.5 で導入されました。

- エラー: 1776 SQLSTATE: HY000 (ER_BAD_SLAVE_AUTO_POSITION)

  メッセージ: MASTER_AUTO_POSITION がアクティブである場合、パラメータ MASTER_LOG_FILE、MASTER_LOG_POS、RELAY_LOG_FILE、および RELAY_LOG_POS は設定できません。

  ER_BAD_SLAVE_AUTO_POSITION は 5.6.5 で導入されました。

- エラー: 1777 SQLSTATE: HY000 (ER_AUTO_POSITION_REQUIRES_GTID_MODE_ON)

  メッセージ: CHANGE MASTER TO MASTER_AUTO_POSITION = 1 は、@@GLOBAL.GTID_MODE = ON のときにのみ実行できます。

  ER_AUTO_POSITION_REQUIRES_GTID_MODE_ON は 5.6.5 で導入されました。

- エラー: 1778 SQLSTATE: HY000 (ER_CANT_DO_IMPLICIT_COMMIT_IN_TRX_WHEN_GTID_NEXT_IS_SET)

  メッセージ: @@SESSION.GTID_NEXT != AUTOMATIC である場合、暗黙的なコミットを行うステートメントはトランザクション内で実行できません。

  ER_CANT_DO_IMPLICIT_COMMIT_IN_TRX_WHEN_GTID_NEXT_IS_SET は 5.6.5 で導入されました。

- エラー: 1779 SQLSTATE: HY000 (ER_GTID_MODE_2_OR_3_REQUIRES_DISABLE_GTID_UNSAFE_STATEMENTS_ON)

  メッセージ: GTID_MODE = ON または GTID_MODE = UPGRADE_STEP_2 を指定する場合は、DISABLE_GTID_UNSAFE_STATEMENTS = 1 である必要があります。

  ER_GTID_MODE_2_OR_3_REQUIRES_DISABLE_GTID_UNSAFE_STATEMENTS_ON は 5.6.5 で導入され、5.6.8 で削除されました。

  5.6.9 で、ER_GTID_MODE_2_OR_3_REQUIRES_DISABLE_GTID_UNSAFE_STATEMENTS_ON が ER_GTID_MODE_2_OR_3_REQUIRES_ENFORCE_GTID_CONSISTENCY_ON に名前変更されました。

- エラー: 1779 SQLSTATE: HY000 (ER_GTID_MODE_2_OR_3_REQUIRES_ENFORCE_GTID_CONSISTENCY_ON)

  メッセージ: @@GLOBAL.GTID_MODE = ON または UPGRADE_STEP_2 を指定する場合は、@@GLOBAL.ENFORCE_GTID_CONSISTENCY = 1 である必要があります。

  ER_GTID_MODE_2_OR_3_REQUIRES_ENFORCE_GTID_CONSISTENCY_ON は 5.6.9 で導入されました。

  5.6.9 で、ER_GTID_MODE_2_OR_3_REQUIRES_DISABLE_GTID_UNSAFE_STATEMENTS_ON が ER_GTID_MODE_2_OR_3_REQUIRES_ENFORCE_GTID_CONSISTENCY_ON に名前変更されました。

- エラー: 1780 SQLSTATE: HY000 (ER_GTID_MODE_REQUIRES_BINLOG)

  メッセージ: @@GLOBAL.GTID_MODE = ON、UPGRADE_STEP_1、または UPGRADE_STEP_2 を指定する場合は、--log-bin および --log-slave-updates である必要があります。

  ER_GTID_MODE_REQUIRES_BINLOG は 5.6.5 で導入されました。

- エラー: 1781 SQLSTATE: HY000 (ER_CANT_SET_GTID_NEXT_TO_GTID_WHEN_GTID_MODE_IS_OFF)

  メッセージ: @@GLOBAL.GTID_MODE = OFF の場合、@@SESSION.GTID_NEXT に UUID:NUMBER を設定できません。

  ER_CANT_SET_GTID_NEXT_TO_GTID_WHEN_GTID_MODE_IS_OFF は 5.6.5 で導入されました。

- エラー: 1782 SQLSTATE: HY000 (ER_CANT_SET_GTID_NEXT_TO_ANONYMOUS_WHEN_GTID_MODE_IS_ON)

  メッセージ: @@GLOBAL.GTID_MODE = ON の場合、@@SESSION.GTID_NEXT に ANONYMOUS を設定できません。

  ER_CANT_SET_GTID_NEXT_TO_ANONYMOUS_WHEN_GTID_MODE_IS_ON は 5.6.5 で導入されました。

- エラー: 1783 SQLSTATE: HY000 (ER_CANT_SET_GTID_NEXT_LIST_TO_NON_NULL_WHEN_GTID_MODE_IS_OFF)

  メッセージ: @@GLOBAL.GTID_MODE = OFF の場合、@@SESSION.GTID_NEXT_LIST に NULL ではない値は設定できません。

  ER_CANT_SET_GTID_NEXT_LIST_TO_NON_NULL_WHEN_GTID_MODE_IS_OFF は 5.6.5 で導入されました。

- エラー: 1784 SQLSTATE: HY000 (ER_FOUND_GTID_EVENT_WHEN_GTID_MODE_IS_OFF)

  メッセージ: @@GLOBAL.GTID_MODE = OFF のときに、Gtid_log_event または Previous_gtids_log_event が見つかりました。

  ER_FOUND_GTID_EVENT_WHEN_GTID_MODE_IS_OFF は 5.6.5 で導入されました。

- エラー: 1785 SQLSTATE: HY000 (ER_GTID_UNSAFE_NON_TRANSACTIONAL_TABLE)

  メッセージ: @@GLOBAL.ENFORCE_GTID_CONSISTENCY = 1 の場合、非トランザクションテーブルへの更新は、自動コミットステートメントまたは単一ステートメントのトランザクションでのみ行うことができ、トランザクションテーブルの更新と同じステートメントで行うことはできません。

  ER_GTID_UNSAFE_NON_TRANSACTIONAL_TABLE は 5.6.5 で導入されました。

- エラー: 1786 SQLSTATE: HY000 (ER_GTID_UNSAFE_CREATE_SELECT)

  メッセージ: @@GLOBAL.ENFORCE_GTID_CONSISTENCY = 1 の場合、CREATE TABLE ... SELECT は禁止されています。

  ER_GTID_UNSAFE_CREATE_SELECT は 5.6.5 で導入されました。

- エラー: 1787 SQLSTATE: HY000 (ER_GTID_UNSAFE_CREATE_DROP_TEMPORARY_TABLE_IN_TRANSACTION)

  メッセージ: @@GLOBAL.ENFORCE_GTID_CONSISTENCY = 1 の場合、ステートメント CREATE TEMPORARY TABLE および DROP TEMPORARY TABLE は非トランザクションコンテキストでのみ実行でき、AUTOCOMMIT = 1 である必要があります。

  ER_GTID_UNSAFE_CREATE_DROP_TEMPORARY_TABLE_IN_TRANSACTION は 5.6.5 で導入されました。

- エラー: 1788 SQLSTATE: HY000 (ER_GTID_MODE_CAN_ONLY_CHANGE_ONE_STEP_AT_A_TIME)

  メッセージ: @@GLOBAL.GTID_MODE の値は一度に 1 ステップのみ変更できます: OFF <-> UPGRADE_STEP_1 <-> UPGRADE_STEP_2 <-> ON。また、この値はすべてのサーバーで同時に変更する必要があります。手順については、マニュアルを参照してください。

  ER_GTID_MODE_CAN_ONLY_CHANGE_ONE_STEP_AT_A_TIME は 5.6.5 で導入されました。

- エラー: 1789 SQLSTATE: HY000 (ER_MASTER_HAS_PURGED_REQUIRED_GTIDS)

  メッセージ: スレーブは CHANGE MASTER TO MASTER_AUTO_POSITION = 1 を使用して接続していますが、スレーブが必要とする GTID を含むバイナリログがマスターによってパージされました。

  ER_MASTER_HAS_PURGED_REQUIRED_GTIDS は 5.6.5 で導入されました。

- エラー: 1790 SQLSTATE: HY000 (ER_CANT_SET_GTID_NEXT_WHEN_OWNING_GTID)

  メッセージ: GTID を所有するクライアントは @@SESSION.GTID_NEXT を変更できません。クライアントは %s を所有しています。所有権は COMMIT または ROLLBACK によって解放されます。

  ER_CANT_SET_GTID_NEXT_WHEN_OWNING_GTID は 5.6.5 で導入されました。

- エラー: 1791 SQLSTATE: HY000 (ER_UNKNOWN_EXPLAIN_FORMAT)

  メッセージ: 不明な EXPLAIN 形式名: '%s'

  ER_UNKNOWN_EXPLAIN_FORMAT は 5.6.5 で導入されました。

- エラー: 1792 SQLSTATE: 25006 (ER_CANT_EXECUTE_IN_READ_ONLY_TRANSACTION)

  メッセージ: READ ONLY トランザクション内ではステートメントを実行できません。

  ER_CANT_EXECUTE_IN_READ_ONLY_TRANSACTION は 5.6.5 で導入されました。

- エラー: 1793 SQLSTATE: HY000 (ER_TOO_LONG_TABLE_PARTITION_COMMENT)

  メッセージ: テーブルパーティション '%s' のコメントが長すぎます (最大 = %lu)

  ER_TOO_LONG_TABLE_PARTITION_COMMENT は 5.6.6 で導入されました。

- エラー: 1794 SQLSTATE: HY000 (ER_SLAVE_CONFIGURATION)

  メッセージ: スレーブが構成されていないか、正常に初期化されませんでした。マスターまたはスレーブを有効にするには、少なくとも --server-id を設定する必要があります。追加のエラーメッセージは MySQL エラーログにあります。

  ER_SLAVE_CONFIGURATION は 5.6.6 で導入されました。

- エラー: 1795 SQLSTATE: HY000 (ER_INNODB_FT_LIMIT)

  メッセージ: InnoDB は、現在、一度に 1 つの FULLTEXT インデックスの作成をサポートしています

  ER_INNODB_FT_LIMIT は 5.6.4 で導入されました。

- エラー: 1796 SQLSTATE: HY000 (ER_INNODB_NO_FT_TEMP_TABLE)

  メッセージ: InnoDB の一時テーブルには FULLTEXT インデックスを作成できません

  ER_INNODB_NO_FT_TEMP_TABLE は 5.6.4 で導入されました。

- エラー: 1797 SQLSTATE: HY000 (ER_INNODB_FT_WRONG_DOCID_COLUMN)

  メッセージ: カラム '%s' は InnoDB の FULLTEXT インデックスでは間違った型です

  ER_INNODB_FT_WRONG_DOCID_COLUMN は 5.6.6 で導入されました。

- エラー: 1798 SQLSTATE: HY000 (ER_INNODB_FT_WRONG_DOCID_INDEX)

  メッセージ: インデックス '%s' は InnoDB の FULLTEXT インデックスでは間違った型です

  ER_INNODB_FT_WRONG_DOCID_INDEX は 5.6.6 で導入されました。

- エラー: 1799 SQLSTATE: HY000 (ER_INNODB_ONLINE_LOG_TOO_BIG)

  メッセージ: インデックス '%s' を作成するには、変更ログに 'innodb_online_alter_log_max_size' バイトより大きい領域が必要となります。再試行してください。

  ER_INNODB_ONLINE_LOG_TOO_BIG は 5.6.6 で導入されました。

- エラー: 1800 SQLSTATE: HY000 (ER_UNKNOWN_ALTER_ALGORITHM)

  メッセージ: 不明な ALGORITHM '%s'

  ER_UNKNOWN_ALTER_ALGORITHM は 5.6.6 で導入されました。

- エラー: 1801 SQLSTATE: HY000 (ER_UNKNOWN_ALTER_LOCK)

メッセージ: 不明な LOCK タイプ '%s'

ER_UNKNOWN_ALTER_LOCK は 5.6.6 で導入されました。

- エラー: 1802 SQLSTATE: HY000 (ER_MTS_CHANGE_MASTER_CANT_RUN_WITH_GAPS)

  メッセージ: スレーブがエラーで停止されたか、MTS モードで強制終了された場合、CHANGE MASTER は実行できません。RESET SLAVE または START SLAVE UNTIL の使用を検討してください。

  ER_MTS_CHANGE_MASTER_CANT_RUN_WITH_GAPS は 5.6.6 で導入されました。

- エラー: 1803 SQLSTATE: HY000 (ER_MTS_RECOVERY_FAILURE)

  メッセージ: 並列実行モードのときに SLAVE でエラーが発生した場合は、リカバリできません。追加のエラーメッセージは MySQL エラーログにあります。

  ER_MTS_RECOVERY_FAILURE は 5.6.6 で導入されました。

- エラー: 1804 SQLSTATE: HY000 (ER_MTS_RESET_WORKERS)

  メッセージ: ワーカー情報テーブルをクリーンアップできません。追加のエラーメッセージは MySQL エラーログにあります。

  ER_MTS_RESET_WORKERS は 5.6.6 で導入されました。

- エラー: 1805 SQLSTATE: HY000 (ER_COL_COUNT_DOESNT_MATCH_CORRUPTED_V2)

  メッセージ: %s.%s のカラム数が間違っています。%d 個を予期しましたが、%d 個が見つかりました。テーブルが破損している可能性があります

- エラー: 1806 SQLSTATE: HY000 (ER_SLAVE_SILENT_RETRY_TRANSACTION)

  メッセージ: スレーブは現在のトランザクションを暗黙のうちに再試行する必要があります

  ER_SLAVE_SILENT_RETRY_TRANSACTION は 5.6.6 で導入されました。

- エラー: 1807 SQLSTATE: HY000 (ER_DISCARD_FK_CHECKS_RUNNING)

  メッセージ: テーブル '%s' で外部キーチェックが実行されています。テーブルを破棄できません。

  ER_DISCARD_FK_CHECKS_RUNNING は 5.6.6 で導入されました。

- エラー: 1808 SQLSTATE: HY000 (ER_TABLE_SCHEMA_MISMATCH)

  メッセージ: スキーマの不一致 (%s)

  ER_TABLE_SCHEMA_MISMATCH は 5.6.6 で導入されました。

- エラー: 1809 SQLSTATE: HY000 (ER_TABLE_IN_SYSTEM_TABLESPACE)

  メッセージ: テーブル '%s' はシステムテーブルスペースにあります

  ER_TABLE_IN_SYSTEM_TABLESPACE は 5.6.6 で導入されました。

- エラー: 1810 SQLSTATE: HY000 (ER_IO_READ_ERROR)

  メッセージ: IO 読み取りエラー: (%lu、%s) %s

  ER_IO_READ_ERROR は 5.6.6 で導入されました。

- エラー: 1811 SQLSTATE: HY000 (ER_IO_WRITE_ERROR)

  メッセージ: IO 書き込みエラー: (%lu、%s) %s

  ER_IO_WRITE_ERROR は 5.6.6 で導入されました。

- エラー: 1812 SQLSTATE: HY000 (ER_TABLESPACE_MISSING)

  メッセージ: テーブル '%s' のテーブルスペースがありません

  ER_TABLESPACE_MISSING は 5.6.6 で導入されました。

- エラー: 1813 SQLSTATE: HY000 (ER_TABLESPACE_EXISTS)

  メッセージ: テーブル '%s' のテーブルスペースは存在します。IMPORT を行う前に、そのテーブルスペースを DISCARD してください。

  ER_TABLESPACE_EXISTS は 5.6.6 で導入されました。

- エラー: 1814 SQLSTATE: HY000 (ER_TABLESPACE_DISCARDED)

  メッセージ: テーブル '%s' のテーブルスペースは破棄されています

  ER_TABLESPACE_DISCARDED は 5.6.6 で導入されました。

- エラー: 1815 SQLSTATE: HY000 (ER_INTERNAL_ERROR)

  メッセージ: 内部エラー: %s

  ER_INTERNAL_ERROR は 5.6.6 で導入されました。

- エラー: 1816 SQLSTATE: HY000 (ER_INNODB_IMPORT_ERROR)

  メッセージ: ALTER TABLE '%s' IMPORT TABLESPACE がエラー %lu で失敗しました: '%s'

  ER_INNODB_IMPORT_ERROR は 5.6.6 で導入されました。

- エラー: 1817 SQLSTATE: HY000 (ER_INNODB_INDEX_CORRUPT)

  メッセージ: インデックスが破損しています: %s

  ER_INNODB_INDEX_CORRUPT は 5.6.6 で導入されました。

- エラー: 1818 SQLSTATE: HY000 (ER_INVALID_YEAR_COLUMN_LENGTH)

  メッセージ: YEAR(%lu) カラム型は非推奨です。代わりに YEAR(4) カラムを作成してください。

  ER_INVALID_YEAR_COLUMN_LENGTH は 5.6.6 で導入されました。

- エラー: 1819 SQLSTATE: HY000 (ER_NOT_VALID_PASSWORD)

  メッセージ: パスワードが現在のポリシー要件を満たしていません

  ER_NOT_VALID_PASSWORD は 5.6.6 で導入されました。

- エラー: 1820 SQLSTATE: HY000 (ER_MUST_CHANGE_PASSWORD)

  メッセージ: このステートメントを実行する前に、SET PASSWORD を実行する必要があります

  ER_MUST_CHANGE_PASSWORD は 5.6.6 で導入されました。

- エラー: 1821 SQLSTATE: HY000 (ER_FK_NO_INDEX_CHILD)

  メッセージ: 外部キー制約の追加に失敗しました。制約 '%s' (外部テーブル '%s') にインデックスがありません

  ER_FK_NO_INDEX_CHILD は 5.6.6 で導入されました。

- エラー: 1822 SQLSTATE: HY000 (ER_FK_NO_INDEX_PARENT)

  メッセージ: 外部キー制約の追加に失敗しました。制約 '%s' (参照されるテーブル '%s') にインデックスがありません

  ER_FK_NO_INDEX_PARENT は 5.6.6 で導入されました。

- エラー: 1823 SQLSTATE: HY000 (ER_FK_FAIL_ADD_SYSTEM)

  メッセージ: システムテーブルへの外部キー制約 '%s' の追加に失敗しました

  ER_FK_FAIL_ADD_SYSTEM は 5.6.6 で導入されました。

- エラー: 1824 SQLSTATE: HY000 (ER_FK_CANNOT_OPEN_PARENT)

  メッセージ: 参照テーブル '%s' のオープンに失敗しました

ER_FK_CANNOT_OPEN_PARENT は 5.6.6 で導入されました。

- エラー: 1825 SQLSTATE: HY000 (ER_FK_INCORRECT_OPTION)

  メッセージ: テーブル '%s' への外部キー制約の追加に失敗しました。FOREIGN KEY 制約 '%s' のオプションが不正です

  ER_FK_INCORRECT_OPTION は 5.6.6 で導入されました。

- エラー: 1826 SQLSTATE: HY000 (ER_FK_DUP_NAME)

  メッセージ: 重複した外部キー制約名 '%s'

  ER_FK_DUP_NAME は 5.6.6 で導入されました。

- エラー: 1827 SQLSTATE: HY000 (ER_PASSWORD_FORMAT)

  メッセージ: パスワードハッシュが予期した形式ではありません。PASSWORD() 関数で正しいパスワードアルゴリズムが使用されていることを確認してください。

  ER_PASSWORD_FORMAT は 5.6.6 で導入されました。

- エラー: 1828 SQLSTATE: HY000 (ER_FK_COLUMN_CANNOT_DROP)

  メッセージ: カラム '%s' をドロップできません: 外部キー制約 '%s' で必要とされています

  ER_FK_COLUMN_CANNOT_DROP は 5.6.6 で導入されました。

- エラー: 1829 SQLSTATE: HY000 (ER_FK_COLUMN_CANNOT_DROP_CHILD)

  メッセージ: カラム '%s' をドロップできません: 外部キー制約 '%s' (テーブル '%s') で必要とされています

  ER_FK_COLUMN_CANNOT_DROP_CHILD は 5.6.6 で導入されました。

- エラー: 1830 SQLSTATE: HY000 (ER_FK_COLUMN_NOT_NULL)

  メッセージ: カラム '%s' は NOT NULL にできません: 外部キー制約 '%s' SET NULL で必要とされています

  ER_FK_COLUMN_NOT_NULL は 5.6.6 で導入されました。

- エラー: 1831 SQLSTATE: HY000 (ER_DUP_INDEX)

  メッセージ: 重複したインデックス '%s' がテーブル '%s.%s' に定義されています。これは非推奨であり、将来のリリースでは許可されなくなります。

  ER_DUP_INDEX は 5.6.7 で導入されました。

- エラー: 1832 SQLSTATE: HY000 (ER_FK_COLUMN_CANNOT_CHANGE)

  メッセージ: カラム '%s' を変更できません: 外部キー制約 '%s' で使用されています

  ER_FK_COLUMN_CANNOT_CHANGE は 5.6.7 で導入されました。

- エラー: 1833 SQLSTATE: HY000 (ER_FK_COLUMN_CANNOT_CHANGE_CHILD)

  メッセージ: カラム '%s' を変更できません: 外部キー制約 '%s' (テーブル '%s') で使用されています

  ER_FK_COLUMN_CANNOT_CHANGE_CHILD は 5.6.7 で導入されました。

- エラー: 1834 SQLSTATE: HY000 (ER_FK_CANNOT_DELETE_PARENT)

  メッセージ: 外部キー制約 '%s' (テーブル '%s') の親であるテーブルからは行を削除できません

  ER_FK_CANNOT_DELETE_PARENT は 5.6.7 で導入されました。

- エラー: 1835 SQLSTATE: HY000 (ER_MALFORMED_PACKET)

  メッセージ: 誤った形式のパケットです。

  ER_MALFORMED_PACKET は 5.6.7 で導入されました。

- エラー: 1836 SQLSTATE: HY000 (ER_READ_ONLY_MODE)

  メッセージ: 読み取り専用モードで実行されています

  ER_READ_ONLY_MODE は 5.6.8 で導入されました。

- エラー: 1837 SQLSTATE: HY000 (ER_GTID_NEXT_TYPE_UNDEFINED_GROUP)

  メッセージ: @@SESSION.GTID_NEXT に GTID が設定されている場合は、COMMIT または ROLLBACK のあとに、明示的に別の値を設定する必要があります。詳細は、GTID_NEXT 変数のマニュアルページを参照してください。現在の @@SESSION.GTID_NEXT は '%s' です。

  ER_GTID_NEXT_TYPE_UNDEFINED_GROUP は 5.6.9 で導入されました。

- エラー: 1838 SQLSTATE: HY000 (ER_VARIABLE_NOT_SETTABLE_IN_SP)

  メッセージ: システム変数 %s はストアドプロシージャーで設定できません。

  ER_VARIABLE_NOT_SETTABLE_IN_SP は 5.6.9 で導入されました。

- エラー: 1839 SQLSTATE: HY000 (ER_CANT_SET_GTID_PURGED_WHEN_GTID_MODE_IS_OFF)

  メッセージ: @@GLOBAL.GTID_PURGED は @@GLOBAL.GTID_MODE = ON のときにのみ設定できます。

  ER_CANT_SET_GTID_PURGED_WHEN_GTID_MODE_IS_OFF は 5.6.9 で導入されました。

- エラー: 1840 SQLSTATE: HY000
  (ER_CANT_SET_GTID_PURGED_WHEN_GTID_EXECUTED_IS_NOT_EMPTY)

  メッセージ: @@GLOBAL.GTID_PURGED は @@GLOBAL.GTID_EXECUTED が空であるときにのみ設定できます。

  ER_CANT_SET_GTID_PURGED_WHEN_GTID_EXECUTED_IS_NOT_EMPTY は 5.6.9 で導入されました。

- エラー: 1841 SQLSTATE: HY000
  (ER_CANT_SET_GTID_PURGED_WHEN_OWNED_GTIDS_IS_NOT_EMPTY)

  メッセージ: @@GLOBAL.GTID_PURGED は、実行中のトランザクションがない (ほかのクライアントでも) ときにのみ設定できます。

  ER_CANT_SET_GTID_PURGED_WHEN_OWNED_GTIDS_IS_NOT_EMPTY は 5.6.9 で導入されました。

- エラー: 1842 SQLSTATE: HY000 (ER_GTID_PURGED_WAS_CHANGED)

  メッセージ: @@GLOBAL.GTID_PURGED が '%s' から '%s' に変更されました。

  ER_GTID_PURGED_WAS_CHANGED は 5.6.9 で導入されました。

- エラー: 1843 SQLSTATE: HY000 (ER_GTID_EXECUTED_WAS_CHANGED)

  メッセージ: @@GLOBAL.GTID_EXECUTED が '%s' から '%s' に変更されました。

  ER_GTID_EXECUTED_WAS_CHANGED は 5.6.9 で導入されました。

- エラー: 1844 SQLSTATE: HY000 (ER_BINLOG_STMT_MODE_AND_NO_REPL_TABLES)

  メッセージ: ステートメントを実行できません: BINLOG_FORMAT = STATEMENT であり、レプリケートされるテーブルとレプリケートされないテーブルの両方に書き込んでいるため、バイナリログに書き込むことができません。

  ER_BINLOG_STMT_MODE_AND_NO_REPL_TABLES は 5.6.9 で導入されました。

- エラー: 1845 SQLSTATE: 0A000 (ER_ALTER_OPERATION_NOT_SUPPORTED)

  メッセージ: %s はこの操作ではサポートされません。%s を試してください。

  ER_ALTER_OPERATION_NOT_SUPPORTED は 5.6.10 で導入されました。

- エラー: 1846 SQLSTATE: 0A000 (ER_ALTER_OPERATION_NOT_SUPPORTED_REASON)

  メッセージ: %s はサポートされていません。理由: %s。%s を試してください。

ER_ALTER_OPERATION_NOT_SUPPORTED_REASON は 5.6.10 で導入されました。

- エラー: 1847 SQLSTATE: HY000 (ER_ALTER_OPERATION_NOT_SUPPORTED_REASON_COPY)

  メッセージ: COPY アルゴリズムではロックが必要となります

  ER_ALTER_OPERATION_NOT_SUPPORTED_REASON_COPY は 5.6.10 で導入されました。

- エラー: 1848 SQLSTATE: HY000 (ER_ALTER_OPERATION_NOT_SUPPORTED_REASON_PARTITION)

  メッセージ: パーティション固有の操作では、LOCK/ALGORITHM はまだサポートされていません

  ER_ALTER_OPERATION_NOT_SUPPORTED_REASON_PARTITION は 5.6.10 で導入されました。

- エラー: 1849 SQLSTATE: HY000 (ER_ALTER_OPERATION_NOT_SUPPORTED_REASON_FK_RENAME)

  メッセージ: 外部キーのカラムパーティショニングが名前変更されました

  ER_ALTER_OPERATION_NOT_SUPPORTED_REASON_FK_RENAME は 5.6.10 で導入されました。

- エラー: 1850 SQLSTATE: HY000 (ER_ALTER_OPERATION_NOT_SUPPORTED_REASON_COLUMN_TYPE)

  メッセージ: カラム型 INPLACE は変更できません

  ER_ALTER_OPERATION_NOT_SUPPORTED_REASON_COLUMN_TYPE は 5.6.10 で導入されました。

- エラー: 1851 SQLSTATE: HY000 (ER_ALTER_OPERATION_NOT_SUPPORTED_REASON_FK_CHECK)

  メッセージ: 外部キーを追加するには、foreign_key_checks=OFF にする必要があります

  ER_ALTER_OPERATION_NOT_SUPPORTED_REASON_FK_CHECK は 5.6.10 で導入されました。

- エラー: 1852 SQLSTATE: HY000 (ER_ALTER_OPERATION_NOT_SUPPORTED_REASON_IGNORE)

  メッセージ: IGNORE を指定して一意のインデックスを作成するには、重複した行を削除する COPY アルゴリズムが必要となります

  ER_ALTER_OPERATION_NOT_SUPPORTED_REASON_IGNORE は 5.6.10 で導入されました。

- エラー: 1853 SQLSTATE: HY000 (ER_ALTER_OPERATION_NOT_SUPPORTED_REASON_NOPK)

  メッセージ: 新しい主キーを追加せずに主キーをドロップすることは許可されません

  ER_ALTER_OPERATION_NOT_SUPPORTED_REASON_NOPK は 5.6.10 で導入されました。

- エラー: 1854 SQLSTATE: HY000 (ER_ALTER_OPERATION_NOT_SUPPORTED_REASON_AUTOINC)

  メッセージ: 自動インクリメントカラムを追加する場合は、ロックする必要があります

  ER_ALTER_OPERATION_NOT_SUPPORTED_REASON_AUTOINC は 5.6.10 で導入されました。

- エラー: 1855 SQLSTATE: HY000 (ER_ALTER_OPERATION_NOT_SUPPORTED_REASON_HIDDEN_FTS)

  メッセージ: 非表示の FTS_DOC_ID をユーザーが表示できるものに置き換えることはできません

  ER_ALTER_OPERATION_NOT_SUPPORTED_REASON_HIDDEN_FTS は 5.6.10 で導入されました。

- エラー: 1856 SQLSTATE: HY000 (ER_ALTER_OPERATION_NOT_SUPPORTED_REASON_CHANGE_FTS)

  メッセージ: FTS_DOC_ID をドロップまたは名前変更できません

  ER_ALTER_OPERATION_NOT_SUPPORTED_REASON_CHANGE_FTS は 5.6.10 で導入されました。

- エラー: 1857 SQLSTATE: HY000 (ER_ALTER_OPERATION_NOT_SUPPORTED_REASON_FTS)

  メッセージ: FULLTEXT インデックスを作成する場合は、ロックする必要があります

  ER_ALTER_OPERATION_NOT_SUPPORTED_REASON_FTS は 5.6.10 で導入されました。

- エラー: 1858 SQLSTATE: HY000 (ER_SQL_SLAVE_SKIP_COUNTER_NOT_SETTABLE_IN_GTID_MODE)

メッセージ: サーバーが @@GLOBAL.GTID_MODE = ON で実行されている場合、sql_slave_skip_counter は設定できません。代わりに、スキップするトランザクションごとに、そのトランザクションと同じ GTID を持つ空のトランザクションを生成してください

ER_SQL_SLAVE_SKIP_COUNTER_NOT_SETTABLE_IN_GTID_MODE は 5.6.10 で導入されました。

- エラー: 1859 SQLSTATE: 23000 (ER_DUP_UNKNOWN_IN_INDEX)

  メッセージ: キー '%s' で重複しています

  ER_DUP_UNKNOWN_IN_INDEX は 5.6.10 で導入されました。

- エラー: 1860 SQLSTATE: HY000 (ER_IDENT_CAUSES_TOO_LONG_PATH)

  メッセージ: データベース名およびオブジェクトの識別子が長いため、パスの長さが %d 文字を超えました。パス: '%s'。

  ER_IDENT_CAUSES_TOO_LONG_PATH は 5.6.10 で導入されました。

- エラー: 1861 SQLSTATE: HY000 (ER_ALTER_OPERATION_NOT_SUPPORTED_REASON_NOT_NULL)

  メッセージ: この SQL_MODE で要求されているように NULL 値を暗黙のうちに変換することはできません

  ER_ALTER_OPERATION_NOT_SUPPORTED_REASON_NOT_NULL は 5.6.10 で導入されました。

- エラー: 1862 SQLSTATE: HY000 (ER_MUST_CHANGE_PASSWORD_LOGIN)

  メッセージ: パスワードの期限が切れました。ログインするには、期限切れのパスワードをサポートするクライアントを使用して、それを変更する必要があります。

  ER_MUST_CHANGE_PASSWORD_LOGIN は 5.6.11 で導入されました。

- エラー: 1863 SQLSTATE: HY000 (ER_ROW_IN_WRONG_PARTITION)

  メッセージ: 間違ったパーティション %s で行が見つかりました

  ER_ROW_IN_WRONG_PARTITION は 5.6.11 で導入されました。

- エラー: 1864 SQLSTATE: HY000 (ER_MTS_EVENT_BIGGER_PENDING_JOBS_SIZE_MAX)

  メッセージ: イベント %s、リレーログ名 %s、位置 %s をワーカースレッドにスケジュールできません。サイズ %lu が slave_pending_jobs_size_max の %lu を超えているためです。

  ER_MTS_EVENT_BIGGER_PENDING_JOBS_SIZE_MAX は 5.6.12 で導入されました。

- エラー: 1865 SQLSTATE: HY000 (ER_INNODB_NO_FT_USES_PARSER)

  メッセージ: InnoDB テーブルに対して CREATE FULLTEXT INDEX WITH PARSER は実行できません

  ER_INNODB_NO_FT_USES_PARSER は 5.6.12 で導入されました。

- エラー: 1866 SQLSTATE: HY000 (ER_BINLOG_LOGICAL_CORRUPTION)

  メッセージ: バイナリログファイル '%s' は論理的に破損しています: %s

  ER_BINLOG_LOGICAL_CORRUPTION は 5.6.12 で導入されました。

- エラー: 1867 SQLSTATE: HY000 (ER_WARN_PURGE_LOG_IN_USE)

  メッセージ: ファイル %s は %d スレッドによって読み取られていたため、パージされませんでした。%d ファイル (%d ファイル中) のみがパージされました。

  ER_WARN_PURGE_LOG_IN_USE は 5.6.12 で導入されました。

- エラー: 1868 SQLSTATE: HY000 (ER_WARN_PURGE_LOG_IS_ACTIVE)

  メッセージ: ファイル %s はアクティブなログファイルであるため、パージされませんでした。

  ER_WARN_PURGE_LOG_IS_ACTIVE は 5.6.12 で導入されました。

- エラー: 1869 SQLSTATE: HY000 (ER_AUTO_INCREMENT_CONFLICT)

  メッセージ: UPDATE の自動インクリメント値が、内部で生成された値と競合しています

  ER_AUTO_INCREMENT_CONFLICT は 5.6.12 で導入されました。

- エラー: 1870 SQLSTATE: HY000 (WARN_ON_BLOCKHOLE_IN_RBR)

  メッセージ: BLACKHOLE テーブルを行フォーマットで変更する %s ステートメントについて、行イベントはログに記録されませんでした。テーブル: '%s'

  WARN_ON_BLOCKHOLE_IN_RBR は 5.6.12 で導入されました。

- エラー: 1871 SQLSTATE: HY000 (ER_SLAVE_MI_INIT_REPOSITORY)

  メッセージ: スレーブがリポジトリからのマスター情報構造体の初期化に失敗しました

  ER_SLAVE_MI_INIT_REPOSITORY は 5.6.12 で導入されました。

- エラー: 1872 SQLSTATE: HY000 (ER_SLAVE_RLI_INIT_REPOSITORY)

  メッセージ: スレーブがリポジトリからのリレーログ情報構造体の初期化に失敗しました

  ER_SLAVE_RLI_INIT_REPOSITORY は 5.6.12 で導入されました。

- エラー: 1873 SQLSTATE: 28000 (ER_ACCESS_DENIED_CHANGE_USER_ERROR)

  メッセージ: ユーザー '%s'@'%s' (使用したパスワード: %s) に変更しようとして、アクセスを拒否されました。切断します。

  ER_ACCESS_DENIED_CHANGE_USER_ERROR は 5.6.13 で導入されました。

- エラー: 1874 SQLSTATE: HY000 (ER_INNODB_READ_ONLY)

  メッセージ: InnoDB は読み取り専用モードです。

  ER_INNODB_READ_ONLY は 5.6.13 で導入されました。

- エラー: 1875 SQLSTATE: HY000 (ER_STOP_SLAVE_SQL_THREAD_TIMEOUT)

  メッセージ: STOP SLAVE コマンドの実行は完了していません: スレーブ SQL スレッドが停止シグナルを受け取りました。スレッドがビジー状態です。現在のタスクが完了したあとに SQL スレッドが停止します。

  ER_STOP_SLAVE_SQL_THREAD_TIMEOUT は 5.6.13 で導入されました。

- エラー: 1876 SQLSTATE: HY000 (ER_STOP_SLAVE_IO_THREAD_TIMEOUT)

  メッセージ: STOP SLAVE コマンドの実行は完了していません: スレーブ IO スレッドが停止シグナルを受け取りました。スレッドがビジー状態です。現在のタスクが完了したあとに IO スレッドが停止します。

  ER_STOP_SLAVE_IO_THREAD_TIMEOUT は 5.6.13 で導入されました。

- エラー: 1877 SQLSTATE: HY000 (ER_TABLE_CORRUPT)

  メッセージ: 操作を実行できません。テーブル '%s.%s' が存在しないか、破損しているか、不良なデータが含まれています。

  ER_TABLE_CORRUPT は 5.6.14 で導入されました。

- エラー: 1878 SQLSTATE: HY000 (ER_TEMP_FILE_WRITE_FAILURE)

  メッセージ: 一時ファイルへの書き込みに失敗しました。

  ER_TEMP_FILE_WRITE_FAILURE は 5.6.15 で導入されました。

- エラー: 1879 SQLSTATE: HY000 (ER_INNODB_FT_AUX_NOT_HEX_ID)

  メッセージ: インデックス名のアップグレードに失敗しました。CREATE INDEX (ALTER TABLE) ALGORITHM COPY を使用してインデックスを再構築してください。

  ER_INNODB_FT_AUX_NOT_HEX_ID は 5.6.16 で導入されました。

- エラー: 1880 SQLSTATE: HY000 (ER_OLD_TEMPORALS_UPGRADED)

  メッセージ: 古い形式の TIME/TIMESTAMP/DATETIME カラムが、新しい形式にアップグレードされました。

  ER_OLD_TEMPORALS_UPGRADED は 5.6.16 で導入されました。

- エラー: 1881 SQLSTATE: HY000 (ER_INNODB_FORCED_RECOVERY)

  メッセージ: innodb_forced_recovery > 0 である場合、操作は許可されません。

  ER_INNODB_FORCED_RECOVERY は 5.6.16 で導入されました。

- エラー: 1882 SQLSTATE: HY000 (ER_AES_INVALID_IV)

  メッセージ: %s に指定された初期化ベクトルが短すぎます。少なくとも %d バイト以上である必要があります

  ER_AES_INVALID_IV は 5.6.17 で導入されました。

- エラー: 1883 SQLSTATE: HY000 (ER_PLUGIN_CANNOT_BE_UNINSTALLED)

  メッセージ: 現在、プラグイン '%s' をアンインストールできません。%s

  ER_PLUGIN_CANNOT_BE_UNINSTALLED は 5.6.20 で導入されました。

- エラー: 1884 SQLSTATE: HY000 (ER_GTID_UNSAFE_BINLOG_SPLITTABLE_STATEMENT_AND_GTID_GROUP)

  メッセージ: ステートメントを実行できません。複数のステートメントとしてバイナリログに書き込む必要があり、これは @@SESSION.GTID_NEXT == 'UUID:NUMBER' の場合は許可されないためです。

  ER_GTID_UNSAFE_BINLOG_SPLITTABLE_STATEMENT_AND_GTID_GROUP は 5.6.20 で導入されました。

- エラー: 1885 SQLSTATE: HY000 (ER_SLAVE_HAS_MORE_GTIDS_THAN_MASTER)

  メッセージ: スレーブにマスターよりも多い GTID があります。マスターの SERVER_UUID を使用します。これは、バイナリログの終わりが切り捨てられたか、最後のバイナリログファイルが失われたことを示している可能性があります。たとえば、sync_binlog != 1 のときに、電源またはディスクの障害が発生したあとの状態です。スレーブにすでにレプリケートされたトランザクションを、マスターがロールバックしている場合としていない場合があります。マスターによってスレーブからマスターにロールバックされたトランザクションをレプリケートし、マスター上の空のトランザクションをコミットして、マスターでコミットされたが GTID_EXECUTED に含められていないトランザクションを反映させることをお勧めします。

  ER_SLAVE_HAS_MORE_GTIDS_THAN_MASTER は 5.6.23 で導入されました。

B.4 クライアントのエラーコードおよびメッセージ

クライアントのエラー情報では、次のソースファイルが使用されています。

- 括弧内のエラー値およびシンボルは、MySQL ソースファイル (include/errmsg.h) の定義に対応しています。
- メッセージ値は、libmysql/errmsg.c ファイルに一覧表示されているエラーメッセージに対応しています。%d および %s は数字および文字列をそれぞれ表しており、表示されるときにメッセージに置換されます。

更新は頻繁に行われるため、ここに一覧表示されていない追加のエラー情報がこれらのファイルに含まれている可能性があります。

- エラー: 2000 (CR_UNKNOWN_ERROR)

  メッセージ: 不明な MySQL エラー

- エラー: 2001 (CR_SOCKET_CREATE_ERROR)

  メッセージ: UNIX のソケット (%d) を作成できません

- エラー: 2002 (CR_CONNECTION_ERROR)

  メッセージ: ソケット '%s' (%d) を使用してローカルの MySQL Server に接続できません

- エラー: 2003 (CR_CONN_HOST_ERROR)

メッセージ: '%s' (%d) の MySQL Server に接続できません

- エラー: 2004 (CR_IPSOCK_ERROR)

  メッセージ: TCP/IP ソケット (%d) を作成できません

- エラー: 2005 (CR_UNKNOWN_HOST)

  メッセージ: 不明な MySQL Server ホスト '%s' (%d)

- エラー: 2006 (CR_SERVER_GONE_ERROR)

  メッセージ: MySQL Server が存在しなくなりました

- エラー: 2007 (CR_VERSION_ERROR)

  メッセージ: プロトコルの不一致。サーバーのバージョン = %d、クライアントのバージョン = %d

- エラー: 2008 (CR_OUT_OF_MEMORY)

  メッセージ: MySQL クライアントに空きメモリーがありません

- エラー: 2009 (CR_WRONG_HOST_INFO)

  メッセージ: 間違ったホスト情報

- エラー: 2010 (CR_LOCALHOST_CONNECTION)

  メッセージ: UNIX ソケット経由の Localhost

- エラー: 2011 (CR_TCP_CONNECTION)

  メッセージ: TCP/IP 経由の %s

- エラー: 2012 (CR_SERVER_HANDSHAKE_ERR)

  メッセージ: サーバーのハンドシェイクでエラーが発生しました

- エラー: 2013 (CR_SERVER_LOST)

  メッセージ: クエリー中に MySQL Server への接続が失われました

- エラー: 2014 (CR_COMMANDS_OUT_OF_SYNC)

  メッセージ: コマンドは同期されていません。このコマンドは現在実行できません

- エラー: 2015 (CR_NAMEDPIPE_CONNECTION)

  メッセージ: 名前付きパイプ: %s

- エラー: 2016 (CR_NAMEDPIPEWAIT_ERROR)

  メッセージ: ホストへの名前付きパイプを待機できません。ホスト: %s パイプ: %s (%lu)

- エラー: 2017 (CR_NAMEDPIPEOPEN_ERROR)

  メッセージ: ホストへの名前付きパイプをオープンできません。ホスト: %s パイプ: %s (%lu)

- エラー: 2018 (CR_NAMEDPIPESETSTATE_ERROR)

  メッセージ: ホストへの名前付きパイプの状態を設定できません。ホスト: %s パイプ: %s (%lu)

- エラー: 2019 (CR_CANT_READ_CHARSET)

  メッセージ: 文字セット %s を初期化できません (パス: %s)

- エラー: 2020 (CR_NET_PACKET_TOO_LARGE)

  メッセージ: 'max_allowed_packet' バイトより大きいパケットを受け取りました

- エラー: 2021 (CR_EMBEDDED_CONNECTION)

メッセージ: 組み込みサーバー

- エラー: 2022 (CR_PROBE_SLAVE_STATUS)

  メッセージ: SHOW SLAVE STATUS でエラーが発生しました。

- エラー: 2023 (CR_PROBE_SLAVE_HOSTS)

  メッセージ: SHOW SLAVE HOSTS でエラーが発生しました。

- エラー: 2024 (CR_PROBE_SLAVE_CONNECT)

  メッセージ: スレーブへの接続でエラーが発生しました。

- エラー: 2025 (CR_PROBE_MASTER_CONNECT)

  メッセージ: マスターへの接続でエラーが発生しました。

- エラー: 2026 (CR_SSL_CONNECTION_ERROR)

  メッセージ: SSL 接続エラー: %s

- エラー: 2027 (CR_MALFORMED_PACKET)

  メッセージ: 誤った形式のパケット

- エラー: 2028 (CR_WRONG_LICENSE)

  メッセージ: このクライアントライブラリは、'%s' ライセンスを持つ MySQL Server で使用するためにのみライセンス契約されています

- エラー: 2029 (CR_NULL_POINTER)

  メッセージ: ゼロポインタの使用方法が無効です

- エラー: 2030 (CR_NO_PREPARE_STMT)

  メッセージ: ステートメントが準備されていません

- エラー: 2031 (CR_PARAMS_NOT_BOUND)

  メッセージ: 準備済みステートメントのパラメータにデータが指定されていません

- エラー: 2032 (CR_DATA_TRUNCATED)

  メッセージ: データが切り捨てられました

- エラー: 2033 (CR_NO_PARAMETERS_EXISTS)

  メッセージ: ステートメントにパラメータが存在しません

- エラー: 2034 (CR_INVALID_PARAMETER_NO)

  メッセージ: 無効なパラメータ数

- エラー: 2035 (CR_INVALID_BUFFER_USE)

  メッセージ: 文字列/バイナリ以外のデータ型には長いデータを送信できません (パラメータ: %d)

- エラー: 2036 (CR_UNSUPPORTED_PARAM_TYPE)

  メッセージ: サポートされていないバッファータイプを使用しています: %d (パラメータ: %d)

- エラー: 2037 (CR_SHARED_MEMORY_CONNECTION)

  メッセージ: 共有メモリー: %s

- エラー: 2038 (CR_SHARED_MEMORY_CONNECT_REQUEST_ERROR)

  メッセージ: 共有メモリーをオープンできません。クライアントが要求イベント (%lu) を作成できませんでした

- エラー: 2039 (CR_SHARED_MEMORY_CONNECT_ANSWER_ERROR)

  メッセージ: 共有メモリーをオープンできません。サーバー (%lu) から応答イベントを受け取っていません

- エラー: 2040 (CR_SHARED_MEMORY_CONNECT_FILE_MAP_ERROR)

  メッセージ: 共有メモリーをオープンできません。サーバーがファイルマッピング (%lu) を割り当てできませんでした

- エラー: 2041 (CR_SHARED_MEMORY_CONNECT_MAP_ERROR)

  メッセージ: 共有メモリーをオープンできません。サーバーがファイルマッピング (%lu) へのポインタを取得できませんでした

- エラー: 2042 (CR_SHARED_MEMORY_FILE_MAP_ERROR)

  メッセージ: 共有メモリーをオープンできません。クライアントがファイルマッピング (%lu) を割り当てできませんでした

- エラー: 2043 (CR_SHARED_MEMORY_MAP_ERROR)

  メッセージ: 共有メモリーをオープンできません。クライアントがファイルマッピング (%lu) へのポインタを取得できませんでした

- エラー: 2044 (CR_SHARED_MEMORY_EVENT_ERROR)

  メッセージ: 共有メモリーをオープンできません。クライアントが %s イベント (%lu) を作成できませんでした

- エラー: 2045 (CR_SHARED_MEMORY_CONNECT_ABANDONED_ERROR)

  メッセージ: 共有メモリーをオープンできません。サーバー (%lu) から応答がありません

- エラー: 2046 (CR_SHARED_MEMORY_CONNECT_SET_ERROR)

  メッセージ: 共有メモリーをオープンできません。サーバー (%lu) に要求イベントを送信できません

- エラー: 2047 (CR_CONN_UNKNOW_PROTOCOL)

  メッセージ: 間違ったプロトコルまたは不明なプロトコル

- エラー: 2048 (CR_INVALID_CONN_HANDLE)

  メッセージ: 無効な接続ハンドル

- エラー: 2049 (CR_SECURE_AUTH)

  メッセージ: 古い (4.1.1 より前) 認証プロトコルを使用した接続が拒否されました (クライアントオプション 'secure_auth' が有効にされています)

- エラー: 2050 (CR_FETCH_CANCELED)

  メッセージ: 行の取得が mysql_stmt_close() 呼び出しによって取り消されました

- エラー: 2051 (CR_NO_DATA)

  メッセージ: 最初に行をフェッチせずに、カラムを読み取ろうとしました

- エラー: 2052 (CR_NO_STMT_METADATA)

  メッセージ: 準備済みステートメントにメタデータが含まれていません

- エラー: 2053 (CR_NO_RESULT_SET)

  メッセージ: ステートメントに結果セットが関連付けられていない状態で、行を読み取ろうとしました

- エラー: 2054 (CR_NOT_IMPLEMENTED)

  メッセージ: この機能はまだ実装されていません

- エラー: 2055 (CR_SERVER_LOST_EXTENDED)

メッセージ: '%s' の MySQL Server への接続が失われました。システムエラー: %d

- エラー: 2056 (CR_STMT_CLOSED)

  メッセージ: 先行する %s() 呼び出しが原因で、ステートメントが間接的にクローズされました

- エラー: 2057 (CR_NEW_STMT_METADATA)

  メッセージ: 結果セットのカラム数が、バインドされたバッファーの数と異なります。ステートメントをリセットして、結果セットのカラムを再度バインドし、ステートメントを再実行する必要があります

- エラー: 2058 (CR_ALREADY_CONNECTED)

  メッセージ: このハンドルはすでに接続されています。各接続に個別のハンドルを使用してください。

- エラー: 2059 (CR_AUTH_PLUGIN_CANNOT_LOAD)

  メッセージ: 認証プラグイン '%s' をロードできません: %s

  CR_AUTH_PLUGIN_CANNOT_LOAD は 5.6.1 で導入されました。

- エラー: 2060 (CR_DUPLICATE_CONNECTION_ATTR)

  メッセージ: 同じ名前の属性がすでにあります

  CR_DUPLICATE_CONNECTION_ATTR は 5.6.6 で導入されました。

- エラー: 2061 (CR_AUTH_PLUGIN_ERR)

  メッセージ: 認証プラグイン '%s' がエラーを報告しました: %s

  CR_AUTH_PLUGIN_ERR は 5.6.10 で導入されました。

B.5 問題および一般的なエラー

このセクションでは、発生する可能性がある一般的な問題およびエラーメッセージを一覧表示しています。問題の原因を判別する方法およびそれらを解決するための手順について説明しています。

B.5.1 問題の原因を判別する方法

問題が発生したときに最初にするべきことは、原因となっているプログラムまたはユニットを特定することです。

- 次のいずれかの症状が発生している場合は、ハードウェアの問題 (メモリー、マザーボード、CPU、ハードディスクなど) またはカーネルの問題である可能性があります。
  - キーボードが動作しない。通常、これは Caps Lock キーを押すことによって確認できます。Caps Lock のランプが変わらない場合は、キーボードを交換する必要があります。(これを行う前に、コンピュータの再起動を試みて、キーボードのすべてのケーブルを確認する必要があります。)
  - マウスポインタが動かない。
  - リモートマシンからの ping にマシンが応答しない。
  - MySQL に関連しないその他のプログラムが正常に動作しない。
  - システムが突然再起動される。(ユーザーレベルの欠陥のあるプログラムがシステムを停止できないようにしてください。)

  この場合は、すべてのケーブルを確認し、診断ツールを実行してハードウェアをチェックすることから開始してください。問題を解決できる可能性があるオペレーティングシステムのパッチ、アップデート、またはサービスパックがあるかどうかも確認してください。すべてのライブラリ (glibc など) が最新であることも確認してください。

  メモリーの問題を早期に発見するために、ECC メモリーを持つマシンを使用することは良いことです。

- キーボードがロックアップした場合は、別のマシンから自分のマシンにログインして kbd_mode -a を実行することによってリカバリできることがあります。

- システムのログファイル (/var/log/messages または同様のログファイル) で問題の原因を調べてください。問題の原因が MySQL にあると思われる場合は、MySQL のログファイルも調べてください。セクション 5.2「MySQL Server ログ」を参照してください。
- ハードウェアに問題がないと思われる場合は、問題の原因となっているプログラムを見つけてください。top、ps、タスクマネージャー、または同様のプログラムを使用して、すべての CPU を使用しているプログラムまたはマシンをロックしているプログラムを確認します。
- top、df、または同様のプログラムを使用して、メモリー、ディスク領域、ファイルディスクリプタ、またはその他の重要なリソースが不足しているかどうかを確認します。
- 問題の原因が暴走したプロセスにある場合は、そのプロセスの強制終了を試みることができます。プロセスが停止しない場合は、オペレーティングシステムにバグがある可能性があります。

ほかのすべての可能性を検査して、MySQL サーバーまたは MySQL クライアントが問題の原因であると判断した場合は、メーリングリストまたはサポートチームに対してバグレポートを作成してください。バグレポートには、システムの動作、および発生している事象についての詳細な説明を含めてください。MySQL が問題の原因であると考える理由も記述してください。この章のすべての状況を考慮に入れてください。システムを検査したときの問題の状況を正確に記述します。プログラムおよびログファイルの出力やエラーメッセージを「コピー&ペースト」します。

動作していないプログラムおよびすべての症状について詳しく記述してください。「システムが動作しない」とのみ記述されたバグレポートを過去に多数受け取りました。これでは、問題の原因に関する情報が提供されません。

プログラムで障害が発生した場合は、次の情報を知ることが常に役に立ちます。

- 問題のプログラムでセグメンテーション違反が発生したかどうか (コアがダンプされたかどうか)。
- プログラムが使用可能なすべての CPU 時間を使用しているかどうか。top を使用して確認してください。プログラムをしばらく実行したままにしてみてください。計算の多い処理が行われているだけである可能性があります。
- mysqld サーバーが問題の原因である場合は、mysqladmin -u root ping または mysqladmin -u root processlist で応答を取得できますか。
- MySQL サーバーに接続しようとしたときに、クライアントプログラムはどのように動作しますか。(たとえば、mysql を実行します。)クライアントが動作しなくなりますか。プログラムから出力はありますか。

バグレポートを送信する場合は、セクション1.6「質問またはバグをレポートする方法」に説明されている手順に従ってください。

B.5.2 MySQL プログラム使用時の一般的なエラー

このセクションでは、MySQL プログラムを実行したときによく発生するエラーを一覧表示しています。問題はクライアントプログラムを実行しようとしたときに発生しますが、多くの問題の解決策では MySQL サーバーの構成を変更します。

B.5.2.1 アクセスは拒否されました

「アクセスは拒否されました」というエラーには多くの原因があります。多くの場合、問題はサーバーが接続時にクライアントプログラムに使用を許可する MySQL アカウントに関連しています。セクション6.2「MySQL アクセス権限システム」およびセクション6.2.7「アクセス拒否エラーの原因」を参照してください。

B.5.2.2 [ローカルの] MySQL サーバーに接続できません

UNIX 上の MySQL クライアントは mysqld サーバーに 2 つの方法で接続できます。UNIX ソケットファイルを使用してファイルシステム内のファイル (デフォルトは /tmp/mysql.sock) を介して接続するか、TCP/IP を使用してポート番号を介して接続します。UNIX ソケットファイルでの接続は TCP/IP よりも高速ですが、同じコンピュータ上にあるサーバーに接続するときにのみ使用できます。UNIX ソケットファイルは、ホスト名を指定しない場合、または特殊なホスト名 localhost を指定する場合に使用されます。

MySQL サーバーが Windows 上で実行されている場合は、TCP/IP を使用して接続できます。サーバーが --enable-named-pipe オプションを指定して起動されていて、サーバーが実行されているホスト上でクライアントを実行する場合は、名前付きパイプを使用して接続することもできます。デフォルトでは、名前付きパイプの名前は MySQL です。mysqld に接続するときにホスト名を指定しない場合、MySQL クライアントは最初に名前付きパイプに接続しようとします。接続できない場合は、TCP/IP ポートに接続します。Windows で名前付きパイプの使用を強制するには、ホスト名として . を使用します。

エラー (2002) 「... に接続できません」 は、通常、サーバーに接続しようとしたときに、システムで MySQL サーバーが実行されていなかったこと、あるいは間違った UNIX ソケットファイル名または TCP/IP ポート番号を使用していることを意味しています。使用している TCP/IP ポートがファイアウォールまたはポートブロックサービスによってブロックされていないことも確認してください。

エラー (2003) 「'server' の MySQL サーバーに接続できません (10061)」 は、ネットワーク接続が拒否されたことを示しています。MySQL サーバーが実行されていること、ネットワーク接続が有効にされていること、および指定したネットワークポートがサーバーに構成されていることを確認してください。

サーバーのホストで mysqld という名前のプロセスが実行されているかどうかを確認することから始めます。(UNIX では ps xa | grep mysqld、Windows ではタスクマネージャーを使用します。)プロセスがない場合は、サーバーを起動してください。セクション2.10.1.3「MySQL サーバーの起動とトラブルシューティング」を参照してください。

mysqld プロセスが実行されている場合は、次のコマンドを使用してチェックできます。ポート番号または UNIX ソケットファイル名は、使用している環境では異なる場合があります。host_ip は、サーバーが実行されているマシンの IP アドレスを表しています。

```
shell> mysqladmin version
shell> mysqladmin variables
shell> mysqladmin -h `hostname` version variables
shell> mysqladmin -h `hostname` --port=3306 version
shell> mysqladmin -h host_ip version
shell> mysqladmin --protocol=SOCKET --socket=/tmp/mysql.sock version
```

hostname コマンドでは通常の引用符ではなく逆引用符が使用されています。これにより、hostname の出力 (つまり、現在のホスト名) が mysqladmin コマンドに渡されます。hostname コマンドがないか、Windows 上で実行している場合は、-h オプションに続けて、マシンのホスト名を手動で入力できます (逆引用符なし)。-h 127.0.0.1 を使用して、TCP/IP でローカルホストへの接続を試みることもできます。

サーバーがネットワーク接続を無視するように構成されていないこと、または (リモート側から接続しようとする場合に) サーバーのネットワークインタフェース上でローカル側でのみ待機するように構成されていないことを確認します。サーバーが --skip-networking を指定して起動された場合、サーバーは TCP/IP 接続を受け入れません。サーバーが --bind-address=127.0.0.1 を指定して起動された場合、サーバーはローカルのループバックインタフェース上でのみ TCP/IP 接続を待機するためリモート接続を受け入れません。

ファイアウォールが MySQL へのアクセスをブロックしていないか確認します。ファイアウォールは、実行中のアプリケーションまたは MySQL によって通信用に使用されるポート番号 (デフォルトは 3306) を基準として構成されることがあります。Linux または Unix の場合、IP テーブル (または同様の機能の) 構成を調べてポートがブロックされていないことを確認します。Windows の場合、ZoneAlarm や Windows XP パーソナルファイアウォールなどのアプリケーションが MySQL ポートをブロックしないようにこれらを構成することが必要な場合があります。

「ローカルの MySQL サーバーに接続できません」というエラーが発生する可能性があるいくつかの原因を次に示します。

- mysqld がローカルホストで実行されていない。オペレーティングシステムのプロセスリストをチェックして、mysqld プロセスが存在することを確認します。
- 多数の TCP/IP 接続がある Windows 上で MySQL サーバーを実行している。クライアントで頻繁にそのエラーが発生している場合は、Windows で MySQL サーバーへの接続に失敗するに回避策があります。
- mysqld が使用する UNIX ソケットファイル (デフォルトでは /tmp/mysql.sock) をほかのユーザーが削除した。たとえば、/tmp ディレクトリから古いファイルを削除する cron ジョブがある可能性があります。mysqladmin version を実行すると、mysqladmin が使用する UNIX ソケットファイルが実際に存在するかどうかを確認できます。この場合の対処方法は、mysql.sock を削除しないように cron ジョブを変更するか、ソケットファイルを別の場所に配置することです。セクションB.5.4.5「MySQL の UNIX ソケットファイルを保護または変更する方法」を参照してください。
- --socket=/path/to/socket オプションを指定して mysqld サーバーを起動したが、クライアントプログラムにソケットファイルの新しい名前を指定し忘れた。サーバーのソケットパス名を変更したら、MySQL クライアントにも通知する必要があります。これを行うには、クライアントプログラムを実行するときに同じ --socket オプションを指定します。mysql.sock ファイルにアクセスするための権限がクライアントにあることも確認する必要があります。ソケットファイルがある場所を見つけるには、次のコマンドを実行します。

  ```
  shell> netstat -ln | grep mysql
  ```

  セクションB.5.4.5「MySQL の UNIX ソケットファイルを保護または変更する方法」を参照してください。

- Linux を使用していて、1 つのサーバースレッドが停止した (コアがダンプされた)。この場合は、MySQL サーバーを再起動する前に、ほかの mysqld スレッドを強制終了する (たとえば、kill を使用します) 必要があります。セクションB.5.4.2「MySQL が繰り返しクラッシュする場合の対処方法」を参照してください。

- UNIX ソケットファイルが保持されているディレクトリまたはソケットファイル自体に対する適切なアクセス権限が、サーバーまたはクライアントプログラムにない可能性がある。この場合は、ディレクトリまたはソケットファイルのアクセス権限を変更して、サーバーおよびクライアントがそれらにアクセスできるようにするか、サーバーがソケットファイルを作成でき、クライアントプログラムがアクセスできるディレクトリのソケットファイル名を指定する --socket オプションを指定して mysqld を再起動します。

「some_host 上の MySQL サーバーに接続できません」というエラーメッセージが表示された場合は、次のことを行なって問題の原因を見つけてください。

- telnet some_host 3306 を実行して Enter キーを何度か押すことによって、サーバーがそのホスト上で実行されているかどうかを確認します (3306 は MySQL のデフォルトのポート番号です。サーバーが別のポートで待機している場合は、値を変更します。)MySQL サーバーが実行されていて、そのポートで待機している場合は、サーバーのバージョン番号を含む応答を受け取ります。「telnet: リモートホストに接続できません: 接続は拒否されました」などのエラーを受け取った場合、指定したポートでサーバーは実行されていません。

- サーバーがローカルホストで実行されている場合は、mysqladmin -h localhost variables を使用して UNIX ソケットファイルを使用して接続することを試みてください。サーバーが待機するように構成されている TCP/IP ポート番号を確認します (これは port 変数の値です。)

- Linux 上で実行していて、Security-Enhanced Linux (SELinux) が有効にされている場合は、mysqld プロセスに対して SELinux の保護が無効にされていることを確認します。

Windows で MySQL サーバーへの接続に失敗する

多数の TCP/IP 接続のある Windows 上で MySQL サーバーを実行していて、「MySQL サーバーに接続できません」というエラーが頻繁に発生している場合は、それらの接続に対応するための十分なエフェメラル (一時的な) ポートを Windows が許可していないことが原因である可能性があります。

TIME_WAIT の目的は、接続が閉じられたあとでも、接続がパケットを受け入れるようにすることです。これは、インターネットのルーティングによってパケットに低速なルートが割り当てられ、双方がクローズに同意したあとにパケットが受信されることがあるためです。そのポートが新しい接続に使用された場合は、古い接続からのパケットによってプロトコルが断絶したり、元の接続からの個人情報が漏えいしたりすることがあります。TIME_WAIT による遅延は、それらの遅延したパケットを受信するために一定の時間を猶予してからポートが再使用されるようにすることによってこれを防ぎます。

LAN 接続では、距離と遅延が比較的に大きいインターネットとは異なり、パケットが大きく遅延して受信されることはほとんどないため、TIME_WAIT を大幅に減らしたほうが安全です。

Windows はエフェメラル (一時的な) TCP ポートをユーザーに許可しています。ポートが閉じられると、TIME_WAIT ステータスとして 120 秒間維持されます。ポートは、この時間が経過するまで再び使用可能になりません。ポート番号のデフォルトの範囲は、Windows のバージョンによって異なります。古いバージョンではポートの数がより制限されます。

- Windows Server 2003 まで: 1025–5000 の範囲のポート

- Windows Vista、Server 2008 以降: 49152–65535 の範囲のポート

使用可能な TCP ポートが少なく (5000)、TIME_WAIT ステータスで多数の TCP ポートが短い期間にオープンおよびクローズされると、ポートが不足する可能性が高くなります。この問題に対処する方法は 2 つあります。

- 原因となっている可能性のある接続プールまたは永続的な接続を調査することによって、急速に消費される TCP ポートの数を減らします

- Windows レジストリのいくつかの設定をチューニングします (次の手順を参照してください)

重要: 次の手順では Windows のレジストリを変更しています。レジストリを変更する前にバックアップを取得し、問題が発生したときにレジストリを復旧する方法を理解してください。レジストリをバックアップ、復旧、および編集する方法については、Microsoft サポート技術情報の記事 (http://support.microsoft.com/kb/256986/EN-US/) を参照してください。

1. レジストリエディタ (Regedt32.exe) を開始します。

2. レジストリの次のキーを見つけます。

```
HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Services\Tcpip\Parameters
```

3. 「編集」メニューで、「値の追加」をクリックして次のレジストリ値を追加します。

   ```
   Value Name: MaxUserPort
   Data Type: REG_DWORD
   Value: 65534
   ```

   これにより、ユーザーが使用できるエフェメラルポートの番号が設定されます。有効な範囲は 5000 から 65534 (10 進数) までです。デフォルト値は 0x1388 (10 進数の 5000) です。

4. 「編集」メニューで、「値の追加」をクリックして次のレジストリ値を追加します。

   ```
   Value Name: TcpTimedWaitDelay
   Data Type: REG_DWORD
   Value: 30
   ```

   これにより、閉じる前に TIME_WAIT 状態で TCP ポートの接続を保持する秒数が設定されます。有効な範囲は 0 (ゼロ) から 300 (10 進数) までです。デフォルト値は 0x78 (10 進数の 120) です。

5. レジストリエディタを終了します。

6. マシンをリブートします。

注意: 前述した設定を元に戻すには、作成したレジストリのエントリを削除します。

B.5.2.3 MySQL サーバーへの接続が失われました

このエラーメッセージには 3 つの原因が考えられます。

通常、これはネットワーク接続の問題を示しており、このエラーが頻繁に発生する場合は、ネットワークの状態を確認してください。エラーメッセージに「クエリー中」が含まれている場合、発生している問題はこれに該当する可能性があります。

「クエリー中」の状態は、1 つ以上のクエリーの一部として数百万件の行が送信されているときに発生することがあります。この問題が発生していることがわかった場合は、net_read_timeout をデフォルトの 30 秒から 60 秒またはデータ転送が完了するのに十分な時間に増やすことを試みてください。

まれに、クライアントがサーバーに初期接続を試みるときに発生することがあります。この場合、connect_timeout 値の設定が数秒であるときは、10 秒 (距離が非常に遠い場合、または低速な接続の場合はさらに長く) に増やすことによって、この問題を解決できることがあります。より一般的ではない原因によってこの問題が発生しているかどうかを判別するには、SHOW GLOBAL STATUS LIKE 'Aborted_connects' を使用します。これは、各初期接続の試行をサーバーが中止するたびに 1 が加算されます。エラーメッセージの一部として「認証パケットを読み込んでいます」が表示されることがあります。その場合も、必要となる解決策がこの方法であることを示しています。

前述の原因ではない場合は、一部のクライアントで発生することがある max_allowed_packet より大きい BLOB 値に関する問題が発生している可能性があります。ER_NET_PACKET_TOO_LARGE エラーが発生することがありますが、それは max_allowed_packet を増やす必要があることを示しています。

B.5.2.4 クライアントは認証プロトコルに対応できません

認証プロトコルの現在の実装は、古い (4.1 より前) クライアントによって使用されるアルゴリズムと互換性がないパスワードハッシュアルゴリズムを使用しています。古いクライアントを使用して 4.1 以降のサーバーに接続しようとすると、次のメッセージが表示されて失敗することがあります。

```
shell> mysql
Client does not support authentication protocol requested
by server; consider upgrading MySQL client
```

この問題に対処する場合、推奨される解決方法はすべてのクライアントプログラムをアップグレードして、4.1.1 以降のクライアントライブラリが使用されるようにすることです。これを行うことができない場合は、次のいずれかの方法を使用します。

- 4.1 より前のクライアントプログラムを使用してサーバーに接続する場合に、4.1 より前のスタイルのパスワードを持つアカウントを使用します。
- 4.1 より前のクライアントプログラムを使用する必要がある各ユーザーのパスワードを 4.1 より前のスタイルにリセットします。これは、SET PASSWORD ステートメントおよび OLD_PASSWORD() 関数を使用して行うことができます。MySQL 5.6.6 の時点では、アカウントの認証プラグインが mysql_old_password であることも最初に確認する必要があります。

```
mysql> UPDATE mysql.user SET plugin = 'mysql_old_password'
mysql> WHERE User = 'some_user' AND Host = 'some_host';
mysql> FLUSH PRIVILEGES;
mysql> SET PASSWORD FOR
    -> 'some_user'@'some_host' = OLD_PASSWORD('newpwd');
```

上記の例では、「newpwd」 を使用するパスワードに置き換えます。MySQL は元のパスワードを教えることはできないため、新しいパスワードを指定する必要があります。

- 古いパスワードハッシュアルゴリズムをデフォルトで使用することをサーバーに通知します。

  1. old_passwords システム変数に 1 を設定して、mysqld を起動します。
  2. 長い 4.1 形式のパスワードにアップデートされた各アカウントに、古い形式のパスワードを割り当てます。これらのアカウントは次のクエリーを使用して識別できます。

     ```
     mysql> SELECT Host, User, Password FROM mysql.user
         -> WHERE LENGTH(Password) > 16;
     ```

     そのクエリーによって表示された各アカウントレコードについて、Host 値および User 値を使用し、前述したいずれかの方法を使用してパスワードを割り当てます。

「クライアントは認証プロトコルに対応できません」というエラーは、複数のバージョンの MySQL がインストールされているが、クライアントプログラムが動的にリンクされ、古いライブラリにリンクされる場合にも発生することがあります。クライアントが互換性のある最新のライブラリバージョンを使用していることを確認してください。これを行う手順はシステムによって異なります。

> 注記
>
> PHP の mysql 拡張は、MySQL 4.1.1 以降の認証プロトコルをサポートしていません。これは使用している PHP バージョンに関係なく当てはまります。mysql 拡張を MySQL 4.1 以降とともに使用する場合は、古いクライアントとともに動作するように MySQL を構成するために、前述のいずれかの方法を使用する必要があることがあります。mysqli 拡張 (PHP 5 で追加された「MySQL Improved」を意味します) は MySQL 4.1 以降で採用されている改善されたパスワードハッシュと互換性があり、この MySQL クライアントライブラリを使用するために MySQL で特別な構成を行う必要はありません。mysqli 拡張については、http://php.net/mysqliを参照してください。

パスワードハッシュおよび認証の背景情報については、セクション6.1.2.4「MySQL でのパスワードハッシュ」を参照してください。

B.5.2.5 パスワードをインタラクティブに入力すると失敗する

--password オプションまたは -p オプションをパスワード値を指定せずに使用して起動すると、MySQL クライアントプログラムはパスワードの入力を求めます。

```
shell> mysql -u user_name -p
Enter password:
```

一部のシステムでは、オプションファイルまたはコマンド行で指定するとパスワードが機能するが、Enter password: プロンプトでインタラクティブに入力すると機能しないことがあります。これは、パスワードを読み取るためにシステムによって提供されたライブラリが、パスワード値を少ない文字数 (通常、8 文字) に制限したときに発生します。これはシステムライブラリの問題であり、MySQL の問題ではありません。これを回避するには、MySQL のパスワードを 8 文字以下の値に変更するか、パスワードをオプションファイルに格納します。

B.5.2.6 ホスト 'host_name' は拒否されました

次のエラーが発生する場合は、mysqld が途中で中断された多数の接続要求を特定のホストから受け取ったことを意味します。

```
Host 'host_name' is blocked because of many connection errors.
Unblock with 'mysqladmin flush-hosts'
```

max_connect_errors システム変数の値は、連続して接続要求が中断された場合にそれを許可する回数を決定します。(セクション5.1.4「サーバーシステム変数」を参照してください。)接続が成功せずに max_connect_errors 回要求が失敗すると、mysqld は何らかの問題があると見なし (たとえば、何者かが侵入しようとしている)、FLUSH HOSTS ステートメントを発行するか、mysqladmin flush-hosts コマンドを実行するまで、そのホストが接続できないようにします。

デフォルトでは、mysqld は接続エラーが 100 回発生したあとにホストをブロックします (MySQL 5.6.6 より前では 10 回)。この値を調整するには、サーバーの起動時に max_connect_errors を設定します。

```
shell> mysqld_safe --max_connect_errors=10000 &
```

この値は実行時にも設定できます。

```
mysql> SET GLOBAL max_connect_errors=10000;
```

特定のホストで「ホスト 'host_name' は拒否されました」というエラーメッセージが表示される場合は、そのホストからの TCP/IP 接続に問題がないことを最初に確認してください。ネットワークに問題がある場合は、max_connect_errors 変数の値を増やしても意味はありません。

B.5.2.7 接続が多すぎます

mysqld サーバーに接続しようとしたときに「接続が多すぎます」というエラーが表示される場合は、使用可能なすべての接続がほかのクライアントによって使用されていることを意味します。

許可される接続数は、max_connections システム変数によって制御されます。デフォルト値は、MySQL が Apache Web サーバーとともに使用された場合にパフォーマンスを向上させるために 151 です。(以前のデフォルトは 100 でした。)より多くの接続をサポートする必要がある場合は、この変数により大きい値を設定してください。

mysqld は実際には max_connections+1 クライアントの接続を許可します。余分な接続は、SUPER 権限を持つアカウントが使用するために予約されています。SUPER 権限を管理者に付与して、通常のユーザー (その権限の必要のないユーザー) に付与しないことによって、権限のないクライアントが最大数接続されていても、管理者はサーバーに接続して SHOW PROCESSLIST を使用し、問題を診断することができます。セクション 13.7.5.30「SHOW PROCESSLIST 構文」を参照してください。

MySQL がサポートできる接続の最大数は、対象となるプラットフォームのスレッドライブラリの品質、使用可能な RAM の量、各接続で使用される RAM の量、各接続からのワークロード、および望ましい応答時間によって異なります。Linux または Solaris は、通常、500 から 1000 の同時接続をサポートでき、使用可能な RAM が数ギガバイトあり、各接続からのワークロードが少なく、応答時間の目標が厳しくない場合は、10,000 接続をサポートできます。Windows では、そのプラットフォームで使用されている POSIX 互換レイヤーのために、(オープンしているテーブル × 2 + オープンしている接続) < 2048 に制限されています。

open-files-limit を増やすことが必要となる可能性があります。MySQL が使用できるハンドル数に関するオペレーティングシステムの制限を緩和する方法については、セクション2.5「Linux に MySQL をインストールする」も参照してください。

B.5.2.8 メモリー不足

mysql クライアントプログラムを使用してクエリーを発行し、次のようなエラーを受け取った場合は、mysql にクエリーの結果全体を格納するための十分なメモリーがないことを意味しています。

```
mysql: Out of memory at line 42, 'malloc.c'
mysql: needed 8136 byte (8k), memory in use: 12481367 bytes (12189k)
ERROR 2008: MySQL client ran out of memory
```

この問題を解決するには、最初にクエリーが正しいかどうかを確認します。そのような多数の行が返されることが予想されるクエリーですか。そうではない場合は、クエリーを修正して再試行してください。予想される場合は、--quick オプションを指定して mysql を呼び出すことができます。これにより、mysql_use_result() C API 関数を使用して結果セットが取得されるようになり、クライアントへのロードが少なくなります (サーバーへのロードが増えます)。

B.5.2.9 MySQL サーバーが存在しなくなりました

このセクションでは、関連する「クエリー中にサーバーへの接続が失われました」というエラーについても説明します。

「MySQL サーバーが存在しなくなりました」というエラーのもっとも一般的な原因は、サーバーがタイムアウトして接続が閉じられた場合です。この場合は、通常、次のいずれかのエラーコードを受け取ります (受け取るエラーコードはオペレーティングシステムによって異なります)。

エラーコード	説明
CR_SERVER_GONE_ERROR	クライアントはサーバーに問い合わせを送信できませんでした。

エラーコード	説明
CR_SERVER_LOST	クライアントはサーバーに書き込むときにエラーを受け取りませんでしたが、問い合わせに対する完全な応答 (または何らかの応答) が得られませんでした。

デフォルトでは、何も発生しなかった場合、サーバーは 8 時間後に接続を閉じます。この時間制限を変更するには、mysqld を起動するときに wait_timeout 変数を設定します。セクション5.1.4「サーバーシステム変数」を参照してください。

スクリプトがある場合、クライアントが自動再接続を行うには、クエリーを再発行する必要があるだけです。これは、クライアントの自動再接続を有効にしている (mysql コマンド行クライアントのデフォルト) ことが前提です。

「MySQL サーバーが存在しなくなりました」というエラーのほかの一般的な原因を次に示します。

- ユーザー (またはデータベース管理者) が実行中のスレッドを KILL ステートメントまたは mysqladmin kill コマンドを使用して強制終了した。
- サーバーへの接続を閉じたあとにクエリーを実行しようとした。これはアプリケーションの論理エラーを示しており、修正する必要があります。
- 別のホスト上で実行されているクライアントアプリケーションに、そのホストから MySQL サーバーに接続するために必要な権限がない。
- クライアント側で TCP/IP 接続がタイムアウトした。これは、コマンド mysql_options(..., MYSQL_OPT_READ_TIMEOUT,...) または mysql_options(..., MYSQL_OPT_WRITE_TIMEOUT,...) を使用している場合に発生することがあります。この場合は、タイムアウト値を増やすと問題が解決されることがあります。
- サーバー側でタイムアウトが発生し、クライアントの自動再接続が無効にされている (MYSQL 構造体の reconnect フラグが 0 と等しい)。
- Windows クライアントを使用していて、コマンドが発行される前にサーバーによって接続がドロップされた (おそらく wait_timeout が期限切れになったため)。

  Windows での問題は、TCP/IP 接続をサーバーに書き込むときに MySQL が OS からエラーを受け取らず、接続から応答を読み取ろうとしたときにエラーを受け取ることです。

  この問題の解決策は、最後のクエリーから長い時間がたっている場合に接続に対して mysql_ping() を実行するか (これは Connector/ODBC が行なっていることです)、mysqld サーバーで wait_timeout に事実上タイムアウトすることがないような高い値を設定することです。
- 間違ったクエリーまたは長すぎるクエリーをサーバーに送信した場合にも、これらのエラーを受け取ることがあります。mysqld が長すぎるパケットまたは順序が間違っているパケットを受け取ると、クライアントで何らかの問題が発生していると見なして、接続を閉じます。大きなクエリーが必要な場合 (たとえば、大きな BLOB カラムを操作している場合)、クエリーの制限を緩和するには、サーバーの max_allowed_packet 変数 (デフォルト値は 1M バイト) を設定します。クライアント側の最大パケットサイズも増やす必要がある場合もあります。パケットサイズの設定については、セクションB.5.2.10「パケットが大きすぎます」を参照してください。

  多数の行を挿入する INSERT ステートメントまたは REPLACE ステートメントも、これらの種類のエラーの原因となることがあります。これらのステートメントのいずれかは、挿入される行数に関係なくサーバーに単一の要求を送信します。このため、1 つの INSERT または REPLACE で送信される行数を減らすことによって多くの場合エラーを防ぐことができます。
- また、16M バイト以上のパケットを送信する場合に、クライアントが 4.0.8 より古く、サーバーが 4.0.8 以降であると (またはその逆の場合)、接続を失うことがあります。
- ホスト名のルックアップが失敗した場合に、このエラーが発生することもあります (たとえば、サーバーまたはネットワークが依存する DNS サーバーが停止した場合)。これは、MySQL が名前解決についてホストシステムに依存しているが、それが動作しているかどうかは知ることができないためです。MySQL が認識している範囲では、この問題はほかのネットワークタイムアウトと区別が付きません。

  --skip-networking オプションを指定して MySQL が起動された場合に、「MySQL サーバーが存在しなくなりました」というエラーが表示されることもあります。

  MySQL のポート (デフォルトは 3306) がファイアウォールによってブロックされていて、MySQL サーバーへのすべての接続が遮断される場合、このエラーの原因となることがある別のネットワークの問題が発生します。

- アプリケーションで複数の子プロセスがフォークされ、すべての子プロセスが MySQL サーバーへの同じ接続を使用しようとした場合に、このエラーが発生することもあります。これは、各子プロセスが個別の接続を使用することによって防ぐことができます。
- クエリーの実行中にサーバーが停止するバグが発生した。

MySQL サーバーが停止して再起動されたかどうかを確認するには、mysqladmin version を実行してサーバーの稼働時間を検査します。mysqld がクラッシュして再起動されたためにクライアント接続が切断された場合は、そのクラッシュの原因を見つけることに集中してください。クエリーを再び発行するとサーバーが再び強制終了されるかどうかを確認することから始めます。セクションB.5.4.2「MySQL が繰り返しクラッシュする場合の対処方法」を参照してください。

接続の喪失に関する情報をさらに取得するには、--log-warnings=2 オプションを指定して mysqld を起動します。これにより、一部の切断に関するエラーが hostname.err ファイルに記録されます。セクション5.2.2「エラーログ」 を参照してください。

この問題に関するバグレポートを作成する場合は、次の情報を含めてください。

- MySQL サーバーが停止したかどうかを示します。これに関する情報はサーバーのエラーログにあります。セクションB.5.4.2「MySQL が繰り返しクラッシュする場合の対処方法」を参照してください。
- 特定のクエリーによって mysqld が強制終了し、クエリーを実行する前に関係するテーブルが CHECK TABLE でチェックされていた場合は、再現可能なテストケースを提供してください。セクション24.4「MySQL のデバッグおよび移植」 を参照してください。
- MySQL サーバーの wait_timeout システム変数の値 (mysqladmin variables を使用すると、この変数の値を取得できます。)
- 一般クエリーログを有効にして mysqld を実行し、問題のクエリーがログに出力されるかどうかを確認することを試みましたか。(セクション5.2.3「一般クエリーログ」を参照してください。)

セクションB.5.2.11「通信エラーおよび中止された接続」およびセクション1.6「質問またはバグをレポートする方法」も参照してください。

B.5.2.10 パケットが大きすぎます

パケットは、MySQL サーバーに送信される単一の SQL ステートメント、クライアントに送信される単一の行、またはマスターレプリケーションサーバーからスレーブに送信されるバイナリログイベントです。

MySQL 5.6 Server およびクライアント間で転送可能なパケットの最大サイズは 1G バイトです。

MySQL クライアントまたは mysqld サーバーが max_allowed_packet バイトより大きいパケットを受け取ると、ER_NET_PACKET_TOO_LARGE エラーが発行され、接続が失われます。一部のクライアントでは、パケットが大きすぎる場合、「クエリー中に MySQL サーバーへの接続が失われました」というエラーを受け取ることもあります。

クライアントとサーバーの両方にそれぞれ max_allowed_packet 変数があるため、大きなパケットを処理する場合は、クライアントとサーバーの両方のこの変数を増やす必要があります。

mysql クライアントプログラムを使用している場合、max_allowed_packet 変数のデフォルトは 16M バイトです。大きな値を設定するには、mysql を次のように起動します。

```
shell> mysql --max_allowed_packet=32M
```

これにより、パケットサイズが 32M バイトに設定されます。

サーバーのデフォルトの max_allowed_packet 値は 1M バイトです。サーバーが大きなクエリーを処理する必要がある場合 (たとえば、大きい BLOB カラムを操作している場合) は、この値を増やすことができます。たとえば、この変数に 16M バイトを設定するには、サーバーを次のように起動します。

```
shell> mysqld --max_allowed_packet=16M
```

オプションファイルを使用して max_allowed_packet を設定することもできます。たとえば、サーバー側のサイズを 16M バイトに設定するには、次の行をオプションファイルに追加します。

```
[mysqld]
max_allowed_packet=16M
```

追加のメモリーは必要なときにのみ割り当てられるため、この変数の値を増やしておくと安全です。たとえば、mysqld が追加のメモリーを割り当てるのは、長いクエリーが発行された場合、またはmysqld が大きな結果

行を返す必要がある場合のみです。この変数のデフォルト値が小さいのは、クライアントとサーバーの間の不正なパケットを捕捉するための予防措置であり、誤って大きなパケットが使用されてメモリー不足にならないようにするためでもあります。

大きい BLOB 値を使用しているが、そのクエリーを処理するための十分なメモリーへのアクセスを mysqld に与えていない場合にも、大きいパケットに関する予期しない問題が発生することがあります。これに当てはまると思われる場合は、mysqld_safe スクリプトの先頭に ulimit -d 256000 を追加して、mysqld を再起動してください。

B.5.2.11 通信エラーおよび中止された接続

通信エラー、中止された接続などの接続の問題が発生した場合は、次の情報ソースを使用して問題を診断してください。

- エラーログ。セクション5.2.2「エラーログ」 を参照してください。
- 一般クエリーログ。セクション5.2.3「一般クエリーログ」を参照してください。
- Aborted_xxx ステータス変数および Connection_errors_xxx ステータス変数。セクション5.1.6「サーバーステータス変数」を参照してください。
- host_cache パフォーマンススキーマテーブルを使用してアクセスできるホストキャッシュ。セクション8.11.5.2「DNS ルックアップの最適化とホストキャッシュ」およびセクション22.9.10.1「host_cache テーブル」を参照してください。

--log-warnings オプションを指定してサーバーを起動した場合は、次のようなメッセージがエラーログに出力されることがあります。

```
010301 14:38:23  Aborted connection 854 to db:
'users' user: 'josh'
```

クライアントが正常に接続したが、その後不適切に切断または終了した場合、サーバーは Aborted_clients ステータス変数をインクリメントし、「接続が中止されました」というメッセージをエラーログに記録します。この原因は次のいずれかです。

- クライアントプログラムが、終了する前に mysql_close() を呼び出さなかった。
- クライアントが、サーバーに要求を発行せずに wait_timeout 秒または interactive_timeout 秒を超えてスリープしている。セクション5.1.4「サーバーシステム変数」を参照してください。
- クライアントプログラムがデータ転送中に突然終了した。

クライアントも接続できない場合、サーバーは Aborted_connects ステータス変数をインクリメントします。接続の失敗は、次の原因で発生することがあります。

- クライアントにデータベースへの接続権限がない。
- クライアントが不正なパスワードを使用している。
- パケットに正しい情報が含まれていない。
- パケットの取得に connect_timeout 秒より長くかかる。セクション5.1.4「サーバーシステム変数」を参照してください。

これらのことが発生した場合は、何者かがサーバーに侵入しようとしていることを示している可能性があります。これらのタイプの問題に関するメッセージは、一般クエリーログに記録されます (有効にされている場合)。

中止されたクライアントまたは中止された接続に関する問題のその他の原因

- max_allowed_packet 変数の値が小さすぎるか、クエリーが mysqld 用に割り当てられているメモリーより大きいメモリーの領域を要求した。セクションB.5.2.10「パケットが大きすぎます」を参照してください。
- Linux で半二重および全二重の両方の Ethernet プロトコルが使用された。Linux の多くの Ethernet ドライバにはこのバグがあります。FTP を使用して、クライアントマシンとサーバーマシンの間で大きなファイルを転送することによって、このバグをテストしてください。転送がバースト-ポーズ-バースト-ポーズモードになる場合は、Linux の二重化シンドロームに陥っています。ネットワークカードとハブ/スイッチの二重化モードを全二重または半二重に切り替えて結果をテストし、最適な設定を判別します。
- 読み取りで中断が発生するスレッドライブラリの問題。
- 不適切に構成されている TCP/IP。

- 障害のある Ethernet、ハブ、スイッチ、ケーブルなど。これは、ハードウェアを交換することによってのみ適切に診断できます。

セクションB.5.2.9「MySQL サーバーが存在しなくなりました」も参照してください。

B.5.2.12 テーブルが満杯です

テーブルが満杯であるというエラーが発生した場合は、ディスクが満杯であるか、テーブルが最大サイズに達した可能性があります。MySQL データベースの事実上の最大テーブルサイズは、通常、MySQL の内部制限ではなくオペレーティングシステムのファイルサイズに関する制約によって判断します。セクションD.10.3「テーブルサイズの制限」を参照してください。

B.5.2.13 ファイルを作成/書き込みできない

一部のクエリーで次のタイプのエラーを受け取る場合は、MySQL が一時ディレクトリに結果セットの一時ファイルを作成できないことを意味します。

```
Can't create/write to file '\\sqla3fe_0.ism'.
```

上記のエラーは Windows での一般的なメッセージです。UNIX のメッセージも似ています。

解決策の 1 つは、--tmpdir オプションを指定して mysqld を起動するか、オプションファイルの [mysqld] セクションにこのオプションを追加することです。たとえば、C:\temp ディレクトリを指定するには、次の行を使用します。

```
[mysqld]
tmpdir=C:/temp
```

C:\temp ディレクトリが存在していて、MySQL サーバーが書き込むための十分な領域がある必要があります。セクション4.2.6「オプションファイルの使用」を参照してください。

このエラーの別の原因は権限の問題です。MySQL サーバーが tmpdir ディレクトリに書き込めることを確認してください。

perror で表示されるエラーコードも確認します。サーバーがテーブルに書き込むことができない原因の 1 つは、ファイルシステムが満杯であるためです。

```
shell> perror 28
OS error code  28:  No space left on device
```

起動時に次のタイプのエラーを受け取る場合は、データファイルの格納に使用されるファイルシステムまたはディレクトリが書き込み保護されていることを示しています。書き込みエラーがテストファイルに対するものであれば、このエラーは重大ではなく、無視しても安全です。

```
Can't create test file /usr/local/mysql/data/master.lower-test
```

B.5.2.14 コマンドは同期されていません

クライアントのコードで「コマンドは同期されていません。このコマンドは現在実行できません」というメッセージを受け取る場合は、クライアント関数を間違った順序で呼び出しています。

たとえば、これは、mysql_free_result() を呼び出す前に、mysql_use_result() を使用して、新しいクエリーを実行しようとした場合に発生することがあります。これは、データを返す 2 つのクエリーの間に mysql_use_result() または mysql_store_result() を呼び出さずに実行した場合にも発生することがあります。

B.5.2.15 ユーザーを無視します

次のエラーが表示される場合は、mysqld が起動されたときまたは付与テーブルをリロードしたときに、user テーブルで不正なパスワードを持つアカウントが見つかったことを意味します。

ユーザー 'some_user'@'some_host' のパスワードが不正です。ユーザーを無視します

その結果、このアカウントは権限システムによって無視されます。

次のリストは、この問題の考えられる原因および解決策を示しています。

- 新しいバージョンの mysqld を古い user テーブルとともに実行している可能性があります。これを確認するには、mysqlshow mysql user を実行して、Password カラムが 16 文字より短いかどうかをチェックします。短い場合、この状態を修正するには、scripts/add_long_password スクリプトを実行します。

- アカウントのパスワードが古いパスワード (8 文字) です。user テーブルのアカウントを更新して、新しいパスワードを設定します。
- PASSWORD() 関数を使用せずに user テーブルにパスワードが指定されました。mysql を使用し、user テーブルのアカウントを PASSWORD() 関数を使用して新しいパスワードで更新します。

```
mysql> UPDATE user SET Password=PASSWORD('newpwd')
    -> WHERE User='some_user' AND Host='some_host';
```

B.5.2.16 表 'tbl_name' は存在しません

次のエラーが表示される場合は、通常、指定された名前のデフォルトデータベースにテーブルが存在しないことを意味します。

```
Table 'tbl_name' doesn't exist
Can't find file: 'tbl_name' (errno: 2)
```

テーブルは存在しているが、誤った名前で参照している場合もあります。

- MySQL はディレクトリおよびファイルを使用してデータベースおよびテーブルを格納しているため、大文字/小文字が区別されるファイル名を持つファイルシステムにデータベースおよびテーブルがある場合は、データベース名およびテーブル名で大文字/小文字が区別されます。
- Windows などの大文字/小文字が区別されないファイルシステムの場合でも、クエリー内の特定のテーブルへのすべての参照で同じ表記を使用する必要があります。

デフォルトデータベースにあるテーブルを確認するには、SHOW TABLES を使用します。セクション 13.7.5「SHOW 構文」を参照してください。

B.5.2.17 文字セットを初期化できません

文字セットの問題がある場合は、次のようなエラーが表示されることがあります。

```
MySQL Connection Failed: Can't initialize character set charset_name
```

このエラーには次のいずれかの原因がある可能性があります。

- 文字セットがマルチバイト文字セットであり、クライアントでその文字セットがサポートされていない。この場合は、-DDEFAULT_CHARSET=charset_name オプションまたは -DWITH_EXTRA_CHARSETS=charset_name オプションを指定して CMake を実行することによって、クライアントを再コンパイルする必要があります。セクション2.9.4「MySQL ソース構成オプション」を参照してください。

  標準のすべての MySQL バイナリは、すべてのマルチバイト文字セットがサポートされる -DWITH_EXTRA_CHARSETS=complex を指定してコンパイルされています。セクション2.9.4「MySQL ソース構成オプション」を参照してください。

- 文字セットは mysqld にコンパイルされない単純な文字セットであり、文字セットの定義ファイルがクライアントが予期している場所にありません。

  この場合は、次のいずれかの方法を使用して問題を解決する必要があります。

  - その文字セットがサポートされるようにクライアントを再コンパイルします。セクション2.9.4「MySQL ソース構成オプション」を参照してください。
  - 文字セットの定義ファイルがあるディレクトリをクライアントに指定します。多くのクライアントでは、--character-sets-dir オプションを指定することによってこれを行うことができます。
  - 文字定義ファイルをクライアントが予期しているパスにコピーします。

B.5.2.18 'File' が見つかりません、および同様のエラー

MySQL から「エラー '...' が見つかりません (エラー番号: 23)」、「ファイル ... をオープンできません (エラー番号: 24)」、あるいは「エラー番号 23」または「エラー番号 24」のその他のエラーを受け取る場合は、MySQL サーバーに十分なファイルディスクリプタが割り当てられていないことを意味します。perror ユーティリティーを使用すると、エラー番号の意味の説明を取得できます。

```
shell> perror 23
OS error code  23:  File table overflow
```

```
shell> perror 24
OS error code  24:  Too many open files
shell> perror 11
OS error code  11:  Resource temporarily unavailable
```

ここでの問題は、mysqld が同時にオープンしたままにしようとしているファイルが多すぎることです。一度に多数のファイルをオープンしないように mysqld に通知するか、mysqld が使用できるファイルディスクリプタの数を増やします。

一度にオープンするファイル数を少なくするように mysqld に通知するには、table_open_cache システム変数の値 (デフォルト値は 64) を減らすことによってテーブルキャッシュを小さくします。セクション8.4.3.1「MySQL でのテーブルのオープンとクローズの方法」で説明されているように、状況によっては、サーバーがキャッシュサイズを一時的に拡張しようとする可能性があるため、これによってファイルディスクリプタの不足を完全に防ぐことはできません。max_connections の値を減らすことによっても、オープンファイルの数が減少します (デフォルト値は 100)。

mysqld が使用できるファイルディスクリプタの数を変更するには、mysqld_safe に --open-files-limit オプションを使用するか、open_files_limit システム変数を設定します。セクション5.1.4「サーバーシステム変数」を参照してください。これらの値を設定するもっとも簡単な方法は、オプションファイルにオプションを追加することです。セクション4.2.6「オプションファイルの使用」を参照してください。オープンファイルの制限の設定をサポートしていない古いバージョンの mysqld を使用している場合は、mysqld_safe スクリプトを編集できます。このスクリプトには、コメントアウトされた行 ulimit -n 256 があります。「#」 文字を削除してこの行をコメント解除し、数字 256 を変更して、mysqld が使用できるファイルディスクリプタの数を設定します。

--open-files-limit および ulimit を使用すると、ファイルディスクリプタの数を増やすことができますが、オペレーティングシステムが課している制限が上限となります。mysqld_safe または mysqld を root として起動した場合にのみオーバーライドできる「堅固な」制限もあります (この場合、起動後に root として実行され続けないように、--user オプションを指定してサーバーを起動する必要もあります)。各プロセスで使用できるファイルディスクリプタの数に関するオペレーティングシステムの制限を緩める必要がある場合は、システムのドキュメントを参照してください。

> **注記**
>
> tcsh シェルを実行している場合、ulimit は機能しません。また、tcsh では、現在の制限を問い合わせたときに不正な値が報告されます。この場合は、sh を使用して mysqld_safe を起動してください。

B.5.2.19 テーブルの破損の問題

--myisam-recover-options を指定して mysqld を起動した場合、MySQL は「not closed properly」または「crashed」としてマークされている MyISAM テーブルを自動的にチェックして修復しようとします。これが発生した場合、MySQL は hostname.err ファイルに「警告: テーブル ... をチェックしています」と書き込み、テーブルを修復する必要がある場合は、「警告: テーブルを修復しています」がそのあとに書き込まれます。これらのエラーを多数受け取り、その直前に予期しない mysqld の停止がなかった場合は、何らかの問題があるため、さらに調査する必要があります。

MySQL 5.6 では、サーバーが MyISAM テーブルの破損を検出すると、追加の情報 (ソースファイルの名前と行番号、テーブルにアクセスしていたスレッドのリストなど) をエラーログに書き込みます。たとえば、「thread_id=1 からエラーを受け取りました。mi_dynrec.c:368」です。これは、バグレポートに含めると役に立つ情報です。

セクション5.1.3「サーバーコマンドオプション」およびセクション24.4.1.7「テーブルが破損した場合のテストケースの作成」も参照してください。

B.5.3 インストール関連の問題

B.5.3.1 ファイル権限の問題

ファイル権限に問題がある場合は、mysqld が起動されたときに、UMASK 環境変数が不適切に設定された可能性があります。たとえば、MySQL はテーブルを作成するときに次のエラーメッセージを発行することがあります。

```
ERROR: Can't find file: 'path/with/filename.frm' (Errcode: 13)
```

デフォルトの UMASK 値は 0660 です。この動作を変更するには、mysqld_safe を次のように起動します。

```
shell> UMASK=384  # = 600 in octal
shell> export UMASK
shell> mysqld_safe &
```

デフォルトでは、MySQL はアクセス許可値 0700 でデータベースディレクトリを作成します。この動作を変更するには、UMASK_DIR 変数を設定します。この値を設定すると、新しいディレクトリは UMASK 値と UMASK_DIR 値を組み合わせたもので作成されます。たとえば、新しいすべてのディレクトリにグループアクセスを与える場合は、次のように実行します。

```
shell> UMASK_DIR=504  # = 770 in octal
shell> export UMASK_DIR
shell> mysqld_safe &
```

MySQL では、UMASK または UMASK_DIR の値がゼロで始まる場合、その値は 8 進数と見なされます。

セクション2.12「環境変数」を参照してください。

B.5.4 管理関連の問題

B.5.4.1 root のパスワードをリセットする方法

MySQL の root のパスワードを設定したことがない場合、root として接続するときにサーバーはパスワードを要求しません。ただし、これはセキュリティー保護されていません。パスワードを割り当てる手順については、セクション2.10.2「最初の MySQL アカウントのセキュリティー設定」を参照してください。

root のパスワードを知っているがそれを変更する場合は、セクション13.7.1.7「SET PASSWORD 構文」を参照してください。

root のパスワードを以前設定したが忘れた場合は、新しいパスワードを設定できます。次のセクションでは、Windows システムと UNIX システムでの手順、およびすべてのシステムに当てはまる一般的な手順について説明します。

root のパスワードのリセット: Windows システム

Windows では、次の手順を使用してすべての MySQL root アカウントのパスワードをリセットします。

1. Administrator としてシステムにログオンします。
2. MySQL サーバーが実行されている場合は停止します。Windows サービスとして実行されているサーバーの場合は、サービスマネージャーを開きます (「スタート」メニューから、「コントロール パネル」、「管理ツール」、「サービス」の順に選択します)。リスト内で MySQL サービスを見つけて、それを停止します。

   サーバーがサービスとして実行されていない場合は、タスクマネージャーを使用して強制的に停止する必要があることがあります。
3. 次のステートメントを含むテキストファイルを作成します。パスワードを使用するパスワードに置き換えます。

   ```
   UPDATE mysql.user SET Password=PASSWORD('MyNewPass') WHERE User='root';
   FLUSH PRIVILEGES;
   ```

   UPDATE ステートメントおよび FLUSH ステートメントは、それぞれ単一の行に記述します。UPDATE ステートメントはすべての root アカウントのパスワードをリセットし、FLUSH ステートメントは付与テーブルをメモリーにリロードするようにサーバーに通知して、パスワードの変更が認識されるようにします。
4. ファイルを保存します。この例では、ファイルに C:\mysql-init.txt という名前を付けます。
5. コンソールウィンドウを開いて、コマンドプロンプトを表示します (「スタート」メニューから「ファイル名を指定して実行」を選択し、実行するコマンドとして cmd を入力します)。
6. 特殊な --init-file オプション (オプション値のバックスラッシュは 2 つです) を指定して MySQL サーバーを起動します。

   ```
   C:\> C:\mysql\bin\mysqld --init-file=C:\\mysql-init.txt
   ```

   C:\mysql 以外の場所に MySQL をインストールした場合は、それに応じてコマンドを変更します。

   --init-file オプションで指定されたファイルの内容がサーバーの起動時に実行され、各 root アカウントのパスワードが変更されます。

   サーバーがログファイルではなくコンソールウィンドウに出力を表示するようにする場合は、--console オプションをコマンドに追加することもできます。

MySQL Installation Wizard を使用して MySQL をインストールした場合は、--defaults-file オプションを指定する必要があることがあります。

```
C:\> "C:\Program Files\MySQL\MySQL Server 5.6\bin\mysqld.exe"
      --defaults-file="C:\\Program Files\\MySQL\\MySQL Server 5.6\\my.ini"
      --init-file=C:\\mysql-init.txt
```

適切な --defaults-file 設定はサービスマネージャーを使用して見つけることができます (「スタート」メニューから「コントロール パネル」、「管理ツール」、「サービス」の順に選択します)。リスト内で MySQL サービスを見つけて、それを右クリックし、「プロパティ」オプションを選択します。「実行ファイルのパス」フィールドに --defaults-file 設定が含まれています。

7. サーバーが正常に起動されたら C:\mysql-init.txt を削除します。

新しいパスワードを使用して root として MySQL サーバーに接続できるようになりました。MySQL サーバーを停止して、通常モードで再起動します。サーバーをサービスとして実行している場合は、Windows の「サービス」ウィンドウから開始します。サーバーを手動で起動している場合は、通常使用するコマンドを使用してください。

root のパスワードのリセット: UNIX システム

UNIX では、次の手順を使用してすべての MySQL root アカウントのパスワードをリセットします。この手順は、サーバーの実行に通常使用する UNIX のログインアカウントを使用して実行されるようにサーバーが起動されることを想定しています。たとえば、mysql のログインアカウントを使用してサーバーを実行する場合は、この手順を使用する前に mysql としてログインしてください。または、root としてログインすることもできますが、この場合は --user=mysql オプションを指定して mysqld を起動する必要があります。--user=mysql を使用せずに root としてサーバーを起動した場合、サーバーは root が所有するファイル (ログファイルなど) をデータディレクトリに作成することがあり、以降のサーバーの起動で権限関連の問題の原因となることがあります。これが発生した場合は、それらのファイルの所有権を mysql に変更するか、それらを削除する必要があります。

1. mysqld サーバーが実行される UNIX ユーザー (たとえば、mysql) としてシステムにログオンします。

2. サーバーのプロセス ID が含まれている .pid ファイルを見つけます。このファイルの正確な場所と名前は、配布、ホスト名、および構成によって異なります。一般的な場所は、/var/lib/mysql/、/var/run/mysqld/、および /usr/local/mysql/data/ です。通常、ファイル名には .pid という拡張子が付けられており、mysqld またはシステムのホスト名で始まります。

   MySQL サーバーを停止するには、次のコマンドに .pid ファイルのパス名を使用して、通常の kill (kill -9 ではありません) を mysqld プロセスに送信します。

   ```
   shell> kill `cat /mysql-data-directory/host_name.pid`
   ```

   cat コマンドには逆引用符 (通常の引用符ではなく) を使用します。これにより、cat の出力が kill コマンドに指定されます。

3. 次のステートメントを含むテキストファイルを作成します。パスワードを使用するパスワードに置き換えます。

   ```
   UPDATE mysql.user SET Password=PASSWORD('MyNewPass') WHERE User='root';
   FLUSH PRIVILEGES;
   ```

   UPDATE ステートメントおよび FLUSH ステートメントは、それぞれ単一の行に記述します。UPDATE ステートメントはすべての root アカウントのパスワードをリセットし、FLUSH ステートメントは付与テーブルをメモリーにリロードするようにサーバーに通知して、パスワードの変更が認識されるようにします。

4. ファイルを保存します。この例では、ファイルに /home/me/mysql-init という名前を付けます。このファイルにはパスワードが含まれているため、ほかのユーザーが見ることができる場所に保存しないでください。mysql (サーバーが実行されるときのユーザー) としてログインしていない場合は、mysql による読み取りを許可する権限がファイルに設定されていることを確認してください。

5. 特殊な --init-file オプションを指定して MySQL サーバーを起動します。

   ```
   shell> mysqld_safe --init-file=/home/me/mysql-init &
   ```

   --init-file オプションで指定されたファイルの内容がサーバーの起動時に実行され、各 root アカウントのパスワードが変更されます。

6. サーバーが正常に起動されたら /home/me/mysql-init を削除します。

新しいパスワードを使用して root として MySQL サーバーに接続できるようになりました。サーバーを停止して、通常モードで再起動します。

root のパスワードのリセット: 一般的な手順

前のセクションでは、Windows システムおよび UNIX システムでのパスワードのリセットの手順を説明しました。ほかの方法としては、すべてのプラットフォームで mysql クライアントを使用して新しいパスワードを設定できます (ただし、この方法はセキュリティー保護の面で劣ります)。

1. mysqld を停止し、--skip-grant-tables オプションを指定して再起動します。これにより、だれでもパスワードなしで接続できるようになり、すべての権限が付与されます。これはセキュアではないため、リモートクライアントによる接続を回避するために、--skip-networking と組み合わせて --skip-grant-tables を使用する必要がある場合もあります。

2. 次のコマンドを使用して mysqld サーバーに接続します。

   ```
   shell> mysql
   ```

3. mysql クライアントで次のステートメントを発行します。パスワードを使用するパスワードに置き換えます。

   ```
   mysql> UPDATE mysql.user SET Password=PASSWORD('MyNewPass')
       ->               WHERE User='root';
   mysql> FLUSH PRIVILEGES;
   ```

   FLUSH ステートメントは、パスワードの変更を認識させるために、付与テーブルをメモリーにリロードするようにサーバーに通知します。

新しいパスワードを使用して root として MySQL サーバーに接続できるようになりました。サーバーを指定して、通常モード (--skip-grant-tables オプションおよび --skip-networking オプションを指定しない) で再起動します。

B.5.4.2 MySQL が繰り返しクラッシュする場合の対処方法

各 MySQL バージョンは、リリース前に多くのプラットフォームでテストされています。これは、MySQL にバグがないということではありませんが、バグがあってもごく少数であり、見つかることはまれです。問題が発生した場合は、システムがクラッシュした正確な原因を探すことが常に役に立ちます。問題の修正が迅速に得られる可能性が高まるためです。

まず、問題は mysqld サーバーが停止したことであるかどうか、またはクライアントに関連しているかどうかを判別してください。mysqld サーバーが稼働している時間を確認するには、mysqladmin version を実行します。mysqld が停止して再起動された場合は、サーバーのエラーログを確認すると原因が見つかる可能性があります。セクション5.2.2「エラーログ」 を参照してください。

一部のシステムでは、mysqld が停止したときのスタックトレースがエラーログに出力され、それを resolve_stack_dump プログラムを使用して変換できます。セクション24.4「MySQL のデバッグおよび移植」 を参照してください。エラーログに書き込まれる変数値は、常に 100% 正しいとは限りません。

多くのサーバーのクラッシュは、破損したデータファイルまたはインデックスファイルによって発生します。MySQL は、各 SQL ステートメントの実行後、クライアントに結果を通知する前に、ディスク上のファイルを write() システムコールを使用して更新します。(データファイルは書き込まれるがインデックスファイルは書き込まれない --delay-key-write を指定して実行している場合は異なります。)これは、mysqld がクラッシュしてもデータファイルのコンテンツは安全であることを意味します。フラッシュされていないデータがオペレーティングシステムによってディスクに書き込まれることが保証されているためです。各 SQL ステートメントのあとに MySQL がすべてのデータをディスクにフラッシュするようにするには、--flush オプションを指定して mysqld を起動します。

これは、通常、次のいずれかが発生していなければ、データが損なわれたテーブルができることはないことを意味します。

- MySQL サーバーまたはサーバーのホストが更新中に強制終了された。
- 更新の途中に mysqld が停止するバグが見つかった。
- 一部の外部プログラムは、テーブルを適切にロックせずに、mysqld と同時にデータファイルまたはインデックスファイルを操作しています。
- 適切なファイルシステムのロック (通常は lockd ロックマネージャーによって処理されます) をサポートしていないシステムで同じデータディレクトリを使用して多数の mysqld サーバーを実行しているか、外部ロックを無効にして複数のサーバーを実行している。

- 大きく破損したデータが含まれているクラッシュしたデータファイルまたはインデックスファイルがあり、mysqld が混乱した。
- データストレージのコードにバグが見つかった。可能性は低いですがあり得ることです。この場合は、修復されたテーブルのコピーに対して ALTER TABLE を使用することによって、別のエンジンへのストレージエンジンの変更を試みることができます。

何かがクラッシュした原因を判別するのは非常に難しいことであるため、まず、ほかの環境で動作する操作でも問題の環境ではクラッシュが発生するのかどうかを確認してください。次のことを試してください。

- mysqladmin shutdown を使用して mysqld サーバーを停止し、データディレクトリから myisamchk --silent --force */*.MYI を実行して、すべての MyISAM テーブルを確認し、mysqld を再起動します。これにより、クリーンな状態から実行していることが保証されます。第5章「MySQL サーバーの管理」 を参照してください。
- 一般クエリーログを有効にして (セクション5.2.3「一般クエリーログ」を参照してください) mysqld を起動します。ログに書き込まれた情報から、特定のクエリーによってサーバーが強制終了したかどうかを判別してください。すべてのバグの 95% は特定のクエリーに関連しています。通常、これは、サーバーが再起動される前のログファイルの最後のクエリーのいずれかです。セクション5.2.3「一般クエリーログ」を参照してください。クエリーを発行する前にすべてのテーブルを確認しても、特定のクエリーによって MySQL が繰り返し強制終了される場合は、バグとして判別されましたのでバグレポートを送信してください。セクション1.6「質問またはバグをレポートする方法」を参照してください。
- 問題を再現するために使用できるテストケースを作成してください。セクション24.4「MySQL のデバッグおよび移植」 を参照してください。
- mysql-test ディレクトリおよび MySQL ベンチマークでテストを実行してください。セクション 24.1.2「MySQL テストスイート」を参照してください。それらの環境では MySQL がよりよくテストされます。アプリケーションをシミュレートするコードをベンチマークに追加することもできます。ベンチマークは、ソース配布の場合は sql-bench ディレクトリ、バイナリ配布の場合は MySQL のインストールディレクトリの下の sql-bench ディレクトリにあります。
- fork_big.pl スクリプトを試してください。(ソース配布の tests ディレクトリにあります。)
- MySQL をデバッグ用に構成すると、何らかの問題がある場合に、考えられるエラーに関する情報を収集しやすくなります。-DWITH_DEBUG=1 オプションを指定して MySQL を再構成し、CMake を実行して再コンパイルします。セクション24.4「MySQL のデバッグおよび移植」 を参照してください。
- オペレーティングシステムに最新のパッチが適用されていることを確認してください。
- --skip-external-locking オプションを使用して mysqld を起動します。一部のシステムでは lockd ロックマネージャーが正常に動作しません。--skip-external-locking オプションは mysqld に外部ロックを使用しないように通知します。(これは、同じデータディレクトリで 2 つの mysqld サーバーは実行できず、myisamchk を使用する場合は注意する必要があることを意味します。それでも、テストとしてこのオプションを試すことには価値があります。)
- mysqld は実行されているが応答しない場合は、mysqladmin -u root processlist を試してください。mysqld が応答しないように見えても、ハングアップしていないことがあります。すべての接続が使用されているか、内部ロックの問題である可能性があります。mysqladmin -u root processlist は、通常、これらの場合でも接続を確立することができ、現在の接続数およびそのステータスに関する役に立つ情報が表示されます。
- ほかのクエリーを実行しているときに、別のウィンドウで mysqladmin -i 5 status コマンドまたは mysqladmin -i 5 -r status コマンドを実行して統計を生成します。
- 次の手順を試してください。

  1. gdb (または別のデバッガ) から mysqld を起動します。セクション24.4「MySQL のデバッグおよび移植」を参照してください。
  2. テストスクリプトを実行します。
  3. もっとも低い 3 つのレベルでバックトレースおよびローカル変数を出力します。gdb でこれを行うには、mysqld が gdb 内でクラッシュしたときに、次のコマンドを使用します。

```
backtrace
info local
up
info local
up
info local
```

gdb で、info threads を使用して存在するスレッドを検査し、thread N (ここで N はスレッド ID です) を使用して特定のスレッドに切り替えることもできます。

- Perl スクリプトを使用してアプリケーションをシミュレートし、MySQL をクラッシュまたは誤動作させてください。
- 通常のバグレポートを送信します。セクション1.6「質問またはバグをレポートする方法」を参照してください。通常より詳しく記述してください。MySQL は多くのユーザーの環境では動作しているため、クラッシュは問題となっているコンピュータにのみ存在する何かによって発生している可能性があります (たとえば、特定のシステムライブラリに関連するエラー)。
- 動的長の行を含むテーブルで問題があり、VARCHAR カラム (BLOB カラムまたは TEXT カラムではなく) のみを使用している場合は、ALTER TABLE を使用して、すべての VARCHAR を CHAR に変更してみてください。これにより、MySQL が固定長の行を使用するようになります。固定長の行では若干追加の領域が使用されますが、破損に対してより耐性があります。

  現在の動的行のコードは数年使用されており、問題はほとんど発生していませんが、動的長の行はその性質のため エラーが発生しやすい傾向があるので、この方法がうまくいくかどうかを試すのは良い考えである可能性があります。
- 問題を診断する場合は、ハードウェアの障害の可能性を考慮に入れてください。欠陥のあるハードウェアは、データ破損の原因となることがあります。ハードウェアをトラブルシューティングする場合は、メモリーおよびディスクのサブシステムに特に注意してください。

B.5.4.3 MySQL が満杯のディスクを処理する方法

このセクションでは、MySQL がディスク満杯エラー (「デバイスに領域が残っていない」など)、および割り当て超過エラー (「書き込みに失敗しました」、「ユーザーブロックの制限に達しました」など) に対処する方法について説明します。

このセクションは、MyISAM テーブルへの書き込みに関連しています。「行」および「レコード」への言及が「イベント」を意味すると理解する必要があることを除き、バイナリログファイルおよびバイナリログインデックスファイルへの書き込みにも当てはまります。

ディスク満杯状態が発生すると、MySQL は次のことを行います。

- 現在の行を書き込むための十分な領域があるかどうかを 1 分おきに確認します。十分な領域がある場合は、何事もなかったかのように稼働し続けます。
- ディスク満杯状態について警告するエントリをログファイルに 10 分おきに書き込みます。

問題を軽減するには次のアクションを行います。

- 続行する場合は、すべてのレコードを挿入するための十分なディスク領域を解放する必要があるだけです。
- または、スレッドを中止するには mysqladmin kill を使用します。スレッドは次回ディスクがチェックされるとき (1 分以内) に中止されます。
- ほかのスレッドが、ディスク満杯状態の原因となったテーブルを待機している可能性があります。複数の「ロックされた」スレッドがある場合は、ディスク満杯状態を待機していた 1 つのスレッドを強制終了すると、ほかのスレッドが続行できるようになります。

前述の動作の例外は、REPAIR TABLE または OPTIMIZE TABLE を使用する場合、あるいは LOAD DATA INFILE またはALTER TABLE ステートメントのあとにインデックスがバッチで作成される場合です。これらのすべてのステートメントでは大きい一時ファイルが作成されることがあり、それがそのまま残された場合、システムのほかの部分で大きな問題となる可能性があります。MySQL がこれらのいずれかの操作を実行していて、ディスクが満杯になった場合は、大きい一時ファイルが削除され、テーブルがクラッシュとしてマークされます。例外は ALTER TABLE の場合で、古いテーブルは変更されないままになります。

B.5.4.4 MySQL が一時ファイルを格納する場所

UNIX では、MySQL は一時ファイルを格納するディレクトリのパス名として、TMPDIR 環境変数の値を使用します。TMPDIR が設定されていない場合、MySQL はシステムのデフォルトを使用します。通常、これは /tmp、/var/tmp、または /usr/tmp です。

Windows では、MySQL は TMPDIR、TEMP、および TMP 環境変数の値を順番にチェックします。MySQL は最初に見つかった設定されている変数を使用し、残りの変数はチェックしません。TMPDIR、TEMP、および TMP

がいずれも設定されていない場合、MySQL は Windows システムのデフォルトを使用します。通常、これは C:\windows\temp\ です。

一時ファイルディレクトリが含まれているファイルシステムが小さすぎる場合は、mysqld に --tmpdir オプションを使用して、十分な領域があるファイルシステムのディレクトリを指定できます。レプリケーションのスレーブでは、LOAD DATA INFILE ステートメントをレプリケートするときに、--slave-load-tmpdir を使用して一時ファイルを保持するための別個のディレクトリを指定できます。

--tmpdir オプションには、ラウンドロビン方式で使用される複数のパスのリストを設定できます。パスは UNIX ではコロン文字 (「:」)、Windows ではセミコロン文字 (「;」) で区切るようにしてください。

> **注記**
>
> 負荷を効果的に分散するには、これらのパスに同じディスクの個別のパーティションではなく、個別の物理ディスクを指定してください。

MySQL サーバーがレプリケーションのスレーブとして動作する場合は、--slave-load-tmpdir にメモリーベースのファイルシステム上にあるディレクトリ、またはサーバーのホストが再起動されたときにクリアーされるディレクトリを設定しないようにしてください。レプリケーションスレーブは、一部の一時ファイルがマシンの再起動後も存続し、一時テーブルまたは LOAD DATA INFILE 操作を複製できるようにする必要があります。サーバーが再起動されたときに、スレーブの一時ファイルディレクトリ内のファイルが失われると、レプリケーションは失敗します。

MySQL は、mysqld が終了したら一時ファイルが削除されるようにしています。これがサポートされるプラットフォームでは (UNIX など)、ファイルをオープンしたあとにリンク解除することによってこれが行われます。この方法のデメリットは、名前がディレクトリのリストに表示されないことであり、一時ファイルディレクトリがあるファイルシステムを満杯にしている大きい一時ファイルが表示されません。(そのような場合は、mysqld に関連付けられている大きいファイルを識別するために、lsof +L1 が役に立つことがあります。)

通常、MySQL はソート (ORDER BY または GROUP BY) を行うときに、1 つまたは 2 つの一時ファイルを使用します。必要となる最大のディスク領域は次の式によって判別されます。

```
(length of what is sorted + sizeof(row pointer))
* number of matched rows
* 2
```

行ポインタのサイズは通常 4 バイトですが、大きいテーブルの場合は将来拡張される可能性があります。

一部の SELECT クエリーでは、MySQL は一時 SQL テーブルを作成します。これらは隠しテーブルではなく、SQL_* という形式の名前が付けられます。

ほとんどの場合、ALTER TABLE は元のテーブルの一時コピーを元のテーブルと同じディレクトリに作成します。ただし、ALTER TABLE でインプレース手法 (オンライン DDL) が使用された場合、InnoDB は一時ファイルを一時ファイルディレクトリに作成します。このディレクトリがそのようなファイルを保持するほどに十分に大きくない場合は、tmpdir システム変数に別のディレクトリを設定する必要があることがあります。オンライン DDL については、セクション14.11「InnoDB とオンライン DDL」を参照してください。

B.5.4.5 MySQL の UNIX ソケットファイルを保護または変更する方法

サーバーがローカルクライアントと通信するために使用する UNIX ソケットファイルのデフォルトの場所は、/tmp/mysql.sock です。(一部の配布形式ではディレクトリが異なる場合があり、たとえば RPM の場合は /var/lib/mysql です。)

UNIX の一部のバージョンでは、/tmp ディレクトリ内のファイル、または一時ファイルに使用されるほかの同様のディレクトリ内のファイルをだれでも削除できます。ソケットファイルがシステム上のそのようなディレクトリに配置されている場合は、問題となることがあります。

ほとんどのバージョンの UNIX では、/tmp ディレクトリを保護して、所有者またはスーパーユーザー (root) のみがファイルを削除できるようにすることができます。これを行うには、root として口グインして次のコマンドを使用することによって、/tmp ディレクトリにスティッキービットを設定します。

```
shell> chmod +t /tmp
```

スティッキービットが設定されたかどうかを確認するには、ls -ld /tmp を実行します。いちばん最後の権限文字が t である場合は、このビットが設定されています。

別の方法は、サーバーが UNIX ソケットファイルを作成する場所を変更することです。これを行う場合は、クライアントプログラムにもファイルの新しい場所を認識させてください。ファイルの場所を指定する方法はいくつかあります。

- グローバルまたはローカルのオプションファイルにパスを指定します。たとえば、次の行を /etc/my.cnf に指定します。

```
[mysqld]
socket=/path/to/socket

[client]
socket=/path/to/socket
```

  セクション4.2.6「オプションファイルの使用」を参照してください。

- mysqld_safe およびクライアントプログラムを実行するときに、--socket オプションをコマンド行に指定します。
- MYSQL_UNIX_PORT 環境変数に UNIX ソケットファイルのパスを設定します。
- 別のデフォルトの UNIX ソケットファイルの場所を使用するように、ソースから MySQL を再コンパイルします。CMake を実行するときに MYSQL_UNIX_ADDR オプションでファイルへのパスを定義します。セクション2.9.4「MySQL ソース構成オプション」を参照してください。

新しいソケットの場所が動作しているかどうかをテストするには、次のコマンドを使用してサーバーに接続します。

```
shell> mysqladmin --socket=/path/to/socket version
```

B.5.4.6 タイムゾーンの問題

SELECT NOW() でローカルの時間ではなく UTC で値が返される問題がある場合は、サーバーに現在のタイムゾーンを通知する必要があります。UNIX_TIMESTAMP() が間違った値を返す場合も同様です。これはサーバーが実行されている環境で行なってください (たとえば、mysqld_safe または mysql.server)。セクション2.12「環境変数」を参照してください。

サーバーのタイムゾーンを設定するには、mysqld_safe に --timezone=timezone_name オプションを指定します。mysqld を起動する前に、TZ 環境変数を設定することによって設定することもできます。

--timezone または TZ に許可される値は、システムによって異なります。許容可能な値を確認するには、オペレーティングシステムのドキュメントを参照してください。

B.5.5 クエリー関連の問題

B.5.5.1 文字列検索での大文字/小文字の区別

非バイナリ文字列の場合 (CHAR、VARCHAR、TEXT)、文字列検索では比較オペランドの照合順序が使用されます。バイナリ文字列の場合 (BINARY、VARBINARY、BLOB)、比較ではオペランドのバイトの数値が使用されます。これは、アルファベット文字の場合、比較で大文字/小文字が区別されることを意味します。

非バイナリ文字列とバイナリ文字列の比較は、バイナリ文字列の比較として扱われます。

単純な比較操作 (>=、>、=、<、<=、ソート、およびグループ化) は、各文字の「ソート値」に基づきます。同じソート値を持つ文字は同じ文字として扱われます。たとえば、「e」 と 「é」 が対象の照合順序で同じソート値を持つ場合は、等しいと判断されます。

デフォルトの文字セットおよび照合順序は latin1 および latin1_swedish_ci であるため、非バイナリ文字列の比較はデフォルトでは大文字/小文字が区別されません。これは、col_name LIKE 'a%' を使用して検索した場合、A または a で始まるすべてのカラム値が取得されることを意味します。この検索で大文字/小文字が区別されるようにするには、いずれかのオペランドで大文字/小文字を区別するか、バイナリ照合順序を持つようにします。たとえば、カラムおよび文字列を比較するときに両方が latin1 文字セットである場合は、COLLATE 演算子を使用して、どちらかのオペランドを latin1_general_cs または latin1_bin の照合順序にすることができます。

```
col_name COLLATE latin1_general_cs LIKE 'a%'
col_name LIKE 'a%' COLLATE latin1_general_cs
col_name COLLATE latin1_bin LIKE 'a%'
col_name LIKE 'a%' COLLATE latin1_bin
```

カラムで常に大文字/小文字が区別して扱われるようにするには、大文字/小文字が区別されるように宣言するか、バイナリ照合順序を指定して宣言します。セクション13.1.17「CREATE TABLE 構文」を参照してください。

非バイナリ文字列の大文字/小文字が区別される比較で大文字/小文字が区別されないようにするには、COLLATE を使用して、大文字/小文字が区別されない照合順序を指定します。通常、次の例の文字列では大文字/小文字が区別されますが、COLLATE によって比較で大文字/小文字が区別されないように変更されています。

```
mysql> SET @s1 = 'MySQL' COLLATE latin1_bin,
    ->     @s2 = 'mysql' COLLATE latin1_bin;
mysql> SELECT @s1 = @s2;
+-----------+
| @s1 = @s2 |
+-----------+
|         0 |
+-----------+
mysql> SELECT @s1 COLLATE latin1_swedish_ci = @s2;
+--------------------------------------+
| @s1 COLLATE latin1_swedish_ci = @s2 |
+--------------------------------------+
|                                    1 |
+--------------------------------------+
```

バイナリ文字列の比較では大文字/小文字が区別されます。この文字列を大文字/小文字が区別されないように比較するには、非バイナリ文字列に変換し、COLLATE を使用して大文字/小文字が区別されない照合順序を指定します。

```
mysql> SET @s = BINARY 'MySQL';
mysql> SELECT @s = 'mysql';
+-------------+
| @s = 'mysql' |
+-------------+
|           0 |
+-------------+
mysql> SELECT CONVERT(@s USING latin1) COLLATE latin1_swedish_ci = 'mysql';
+-------------------------------------------------------------------+
| CONVERT(@s USING latin1) COLLATE latin1_swedish_ci = 'mysql' |
+-------------------------------------------------------------------+
|                                                                 1 |
+-------------------------------------------------------------------+
```

値が非バイナリ文字列またはバイナリ文字列のどちらで比較されるかを判別するには、COLLATION() 関数を使用します。次の例は、VERSION() が大文字/小文字が区別されない照合順序の文字列を返すことを示しており、比較では大文字/小文字が区別されません。

```
mysql> SELECT COLLATION(VERSION());
+----------------------+
| COLLATION(VERSION()) |
+----------------------+
| utf8_general_ci      |
+----------------------+
```

バイナリ文字列の場合、照合順序値は binary であり、比較では大文字/小文字が区別されます。binary が表示される 1 つのコンテキストは、一般的なルールとしてバイナリ文字列を返す圧縮関数および暗号化関数の場合です。

```
mysql> SELECT COLLATION(ENCRYPT('x')), COLLATION(SHA1('x'));
+-------------------------+----------------------+
| COLLATION(ENCRYPT('x')) | COLLATION(SHA1('x')) |
+-------------------------+----------------------+
| binary                  | binary               |
+-------------------------+----------------------+
```

文字列のソート値を確認する場合は、WEIGHT_STRING() が役に立つことがあります。セクション12.5「文字列関数」を参照してください。

B.5.5.2 DATE カラムの使用に関する問題

DATE 値の形式は 'YYYY-MM-DD' です。標準 SQL に従うと、ほかの形式は許可されません。UPDATE の式および SELECT ステートメントの WHERE 句では、この形式を使用してください。例:

```
SELECT * FROM t1 WHERE date >= '2003-05-05';
```

利便性のため、日付が数値のコンテキストで使用されている場合、MySQL は日付を数値に自動的に変換します (その逆にも変換されます)。また、MySQL は、更新時、および日付を DATE、DATETIME、または TIMESTAMP カラムと比較する WHERE 句で、「緩やかな」文字列形式を許可します。「緩やかな」形式とは、各部分の区切り文字として句読点文字を使用できることを意味します。たとえば、'2004-08-15' と '2004#08#15' は同等です。MySQL は、日付として解釈できる場合、区切り文字が含まれていない文字列 ('20040815' など) も変換できます。

<、<=、=、>=、>、または BETWEEN 演算子を使用して、DATE、TIME、DATETIME、または TIMESTAMP を定数文字列と比較する場合、通常、MySQL はより速く比較するために (および「緩やかな」文字列チェックのため) 文字列を内部長整数に変換します。ただし、この変換には次の例外があります。

- 2 つのカラムを比較する場合
- DATE、TIME、DATETIME、または TIMESTAMP カラムと式を比較する場合
- 上記で一覧表示した比較方法以外の比較方法を使用する場合 (IN、STRCMP() など)。

これらの例外の場合、比較はオブジェクトを文字列に変換して文字列比較を実行することによって行われます。

安全に処理を行うには、時間値と文字列を比較する場合、文字列が文字列として比較されると想定し、適切な文字列関数を使用します。

特殊な「ゼロ」日付 '0000-00-00' は、'0000-00-00'. として格納および取得できます。'0000-00-00' 日付が Connector/ODBC を介して使用される場合、ODBC はそのような日付を処理できないため、NULL に自動的に変換されます。

MySQL が前述の変換を実行するため、次のステートメントは動作します (idate が DATE カラムであると想定しています)。

```
INSERT INTO t1 (idate) VALUES (19970505);
INSERT INTO t1 (idate) VALUES ('19970505');
INSERT INTO t1 (idate) VALUES ('97-05-05');
INSERT INTO t1 (idate) VALUES ('1997.05.05');
INSERT INTO t1 (idate) VALUES ('1997 05 05');
INSERT INTO t1 (idate) VALUES ('0000-00-00');

SELECT idate FROM t1 WHERE idate >= '1997-05-05';
SELECT idate FROM t1 WHERE idate >= 19970505;
SELECT MOD(idate,100) FROM t1 WHERE idate >= 19970505;
SELECT idate FROM t1 WHERE idate >= '19970505';
```

ただし、次のステートメントは動作しません。

```
SELECT idate FROM t1 WHERE STRCMP(idate,'20030505')=0;
```

STRCMP() は文字列関数であり、idate を 'YYYY-MM-DD' という形式の文字列に変換して、文字列比較を実行します。'20030505' は日付 '2003-05-05' に変換されずに、日付比較が実行されます。

ALLOW_INVALID_DATES SQL モードを有効にしている場合、MySQL は限定的なチェックのみが行われた日付を格納することを許可します。MySQL は、日が 1 から 31 までの範囲内にあり、月が 1 から 12 までの範囲内にあることのみを要求します。これにより、Web アプリケーションで年、月、および日を 3 つの別個のフィールドで取得して、ユーザーが入力したとおりに格納する (日付検証なしで) 場合に、MySQL が非常に便利になります。

MySQL は、日または月と日がゼロである日付の格納を許可します。これは、生年月日を DATE カラムに格納するが、その日付の一部のみがわかっている場合に便利です。日付でゼロの月または日の部分を無効にするには、NO_ZERO_IN_DATE SQL モードを有効にします。

MySQL では、「ダミーの日付」として '0000-00-00' の「ゼロ」の値を格納できます。これは、NULL 値を使用するよりも便利な場合があります。DATE カラムに格納される日付を妥当な値に変換できない場合、MySQL は '0000-00-00' を格納します。'0000-00-00' を無効にするには、NO_ZERO_DATE SQL モードを有効にします。

MySQL がすべての日付をチェックして、有効な日付のみを受け入れるようにするには (IGNORE によってオーバーライドされないかぎり)、sql_mode システム変数に "NO_ZERO_IN_DATE,NO_ZERO_DATE" を設定します。

B.5.5.3 NULL 値に関する問題

NULL 値の概念については、NULL が空の文字列 '' と同じであると考えがちな SQL の初心者が混乱することがよくあります。これらは同一ではありません。たとえば、次の 2 つのステートメントは完全に異なります。

```
mysql> INSERT INTO my_table (phone) VALUES (NULL);
mysql> INSERT INTO my_table (phone) VALUES ('');
```

両方のステートメントで phone カラムに値が挿入されていますが、最初のステートメントは NULL 値を挿入しており、2 番目のステートメントは空の文字列を挿入しています。最初のステートメントの意味は「電話番号がわからない」、2 番目のステートメントの意味は「この人は電話を持っていないため、電話番号がない」と見なすことができます。

NULL を処理する場合は、IS NULL 演算子と IS NOT NULL 演算子、および IFNULL() 関数を使用できます。

SQL では、NULL 値はほかの値 (NULL を含む) との比較で true になることはありません。NULL を含む式は、式に関連する演算子および関数のドキュメントに示されている場合を除き、常に NULL 値を生成します。次の例のすべてのカラムは NULL を返します。

```
mysql> SELECT NULL, 1+NULL, CONCAT('Invisible',NULL);
```

NULL であるカラム値を検索する場合、expr = NULL テストは使用できません。expr = NULL はどのような式の場合でも true にならないため、次のステートメントは行を返しません。

```
mysql> SELECT * FROM my_table WHERE phone = NULL;
```

NULL 値を検索するには、IS NULL テストを使用する必要があります。次のステートメントは、NULL の電話番号および空の電話番号を検索する方法を示しています。

```
mysql> SELECT * FROM my_table WHERE phone IS NULL;
mysql> SELECT * FROM my_table WHERE phone = '';
```

追加情報および例については、セクション3.3.4.6「NULL 値の操作」を参照してください。

MyISAM、InnoDB、または MEMORY ストレージエンジンを使用している場合は、NULL 値を持つことができるカラムにインデックスを追加できます。それ以外の場合は、インデックスが付けられるカラムを NOT NULL と宣言する必要があり、そのカラムには NULL を挿入できません。

LOAD DATA INFILE でデータが読み取られるときに、空のカラムまたは欠落しているカラムは '' で更新されます。NULL 値をカラムにロードするには、データファイルで \N を使用します。状況によっては、リテラル文字「NULL」も使用できます。セクション13.2.6「LOAD DATA INFILE 構文」を参照してください。

DISTINCT、GROUP BY、または ORDER BY が使用された場合、すべての NULL 値は等しいと見なされます。

ORDER BY を使用した場合、NULL 値は最初 (DESC を指定してソートを降順にした場合は最後) に表示されます。

集約 (サマリー) 関数 (COUNT()、MIN()、SUM() など) は NULL 値を無視します。例外は個別のカラム値ではなく行数をカウントする COUNT(*) です。たとえば、次のステートメントは 2 つのカウントを生成します。最初のカウントはテーブル内の行数のカウントであり、2 番目のカウントは age カラムの NULL 以外の値の数のカウントです。

```
mysql> SELECT COUNT(*), COUNT(age) FROM person;
```

一部のデータ型では、MySQL は NULL 値に対して特殊な処理を行います。NULL を TIMESTAMP カラムに挿入すると、現在の日付と時間が挿入されます。NULL を AUTO_INCREMENT 属性を持つ整数カラムまたは浮動小数点カラムに挿入すると、シーケンスの次の数値が挿入されます。

B.5.5.4 カラムエイリアスに関する問題

エイリアスをクエリーの選択リストに使用すると、カラムを別の名前にすることができます。GROUP BY、ORDER BY、または HAVING 句でエイリアスを使用して、カラムを参照できます。

```
SELECT SQRT(a*b) AS root FROM tbl_name
  GROUP BY root HAVING root > 0;
SELECT id, COUNT(*) AS cnt FROM tbl_name
  GROUP BY id HAVING cnt > 0;
SELECT id AS 'Customer identity' FROM tbl_name;
```

標準 SQL では、WHERE 句でのカラムエイリアスへの参照は許可されません。WHERE 句が評価されるときに、カラム値がまだ判別されていない場合があるため、この制限が課されています。たとえば、次のクエリーは不正です。

```
SELECT id, COUNT(*) AS cnt FROM tbl_name
  WHERE cnt > 0 GROUP BY id;
```

WHERE 句は GROUP BY 句に含められる行を判別しますが、行が選択されるまでわからないカラム値のエイリアスを参照して GROUP BY によってグループ化しています。

クエリーの選択リストで、引用したカラムエイリアスを指定するには、識別子または文字列引用文字を使用します。

```
SELECT 1 AS `one`, 2 AS 'two';
```

ステートメント内のどこに指定する場合でも、エイリアスへの引用した参照には、識別子引用符を使用する必要があります。そうしないと、参照は文字列リテラルとして扱われます。たとえば、次のステートメントはカラム id の値によってグループ化され、エイリアス `a` を使用して参照されます。

```
SELECT id AS 'a', COUNT(*) AS cnt FROM tbl_name
  GROUP BY `a`;
```

ただし、次のステートメントはリテラル文字列 'a' によってグループ化され、予期したとおりに動作しません。

```
SELECT id AS 'a', COUNT(*) AS cnt FROM tbl_name
  GROUP BY 'a';
```

B.5.5.5 非トランザクションテーブルのロールバックの失敗

ROLLBACK を実行しようとしたときに次のメッセージを受け取った場合は、トランザクションで使用された 1 つ以上のテーブルがトランザクションをサポートしていないことを意味します。

```
Warning: Some non-transactional changed tables couldn't be rolled back
```

これらの非トランザクションテーブルは、ROLLBACK ステートメントの影響を受けません。

トランザクション内でトランザクションテーブルと非トランザクションテーブルを意図的に混在させていない場合、このメッセージの原因は、トランザクションテーブルと考えていたテーブルが実際にはそうではなかったことである可能性があります。これは、mysqld サーバーによってサポートされていない (または起動オプションで無効にされた) トランザクションストレージエンジンを使用してテーブルを作成しようとした場合に発生することがあります。mysqld がストレージエンジンをサポートしない場合は、非トランザクションである MyISAM テーブルとしてテーブルが作成されます。

テーブルのストレージエンジンを確認するには、次のいずれかのステートメントを使用します。

```
SHOW TABLE STATUS LIKE 'tbl_name';
SHOW CREATE TABLE tbl_name;
```

セクション13.7.5.37「SHOW TABLE STATUS 構文」およびセクション13.7.5.12「SHOW CREATE TABLE 構文」を参照してください。

mysqld サーバーによってサポートされるストレージエンジンを確認するには、次のステートメントを使用します。

```
SHOW ENGINES;
```

詳細は、セクション13.7.5.17「SHOW ENGINES 構文」を参照してください。

B.5.5.6 関連するテーブルからの行の削除

related_table の DELETE ステートメントの合計長が 1M バイト (max_allowed_packet システム変数のデフォルト値) より大きい場合は、小さく分割して複数の DELETE ステートメントを実行してください。related_column にインデックスが付けられている場合は、ステートメントごとに 100 から 1,000 の related_column 値のみを指定することによって、DELETE が最速になる可能性があります。related_column にインデックスが付けられていない場合、速度は IN 句の引数の数の影響を受けません。

B.5.5.7 一致する行がない場合の問題の解決

多数のテーブルを使用する複雑なクエリーで行が返されない場合は、次の手順を使用して問題を判別してください。

1. EXPLAIN を指定してクエリーをテストし、明らかな間違いが見つかるかどうかを確認します。セクション13.8.2「EXPLAIN 構文」を参照してください。
2. WHERE 句で使用されているカラムのみを選択します。
3. 行が返されるまで、クエリーから一度に 1 つずつテーブルを削除します。テーブルが大きい場合、クエリーに LIMIT 10を使用するのは良い方法です。
4. クエリーから最後に削除したテーブルに対して一致する行を持つカラムに SELECT を発行します。
5. FLOAT カラムまたは DOUBLE カラムと 10 進数の数値を比較している場合、等式 (=) 比較は使用できません。すべての浮動小数点値が正確な精度で格納されるとはかぎらないため、この問題はほとんどのコンピュータ言語で一般的です。FLOAT を DOUBLE に変更すると解決することがあります。セクションB.5.5.8「浮動小数点値に関する問題」を参照してください。
6. 問題がまだ特定されない場合は、mysql test < query.sql を使用して実行可能な、問題が再現される最小限のテストを作成します。テストファイルを作成するには、mysqldump --quick db_name tbl_name_1 ... tbl_name_n > query.sql を使用してテーブルをダンプします。エディタでファイルを開いて、一部の挿入行を削除し (問題の再現に必要な分量以上にある場合)、ファイルの最後に SELECT ステートメントを追加します。

   次のコマンドを実行して、テストファイルで問題が再現されることを確認します。

```
shell> mysqladmin create test2
shell> mysql test2 < query.sql
```

テストファイルをバグレポートに添付します (セクション1.6「質問またはバグをレポートする方法」の手順を参照してください)。

B.5.5.8 浮動小数点値に関する問題

浮動小数点数は、近似値であり正確な値として格納されないため、混乱の原因となることがあります。SQL ステートメントで出力される浮動小数点値は、内部で表された値と同じではないことがあります。比較で浮動小数点値を正確な値として扱おうとすると、問題となることがあります。これらはまた、プラットフォームまたは実装の依存関係にも従います。FLOAT データ型および DOUBLE データ型では、これらの問題が発生することがあります。DECIMAL カラムの場合、MySQL は演算を 65 桁 (10 進数) の精度で実行するため、ほとんどの一般的な精度の問題が解決されます。

次の例では、DOUBLE を使用し、浮動小数点演算を使用して行われる計算がどのように浮動小数点エラーとなるかを示しています。

```
mysql> CREATE TABLE t1 (i INT, d1 DOUBLE, d2 DOUBLE);
mysql> INSERT INTO t1 VALUES (1, 101.40, 21.40), (1, -80.00, 0.00),
    -> (2, 0.00, 0.00), (2, -13.20, 0.00), (2, 59.60, 46.40),
    -> (2, 30.40, 30.40), (3, 37.00, 7.40), (3, -29.60, 0.00),
    -> (4, 60.00, 15.40), (4, -10.60, 0.00), (4, -34.00, 0.00),
    -> (5, 33.00, 0.00), (5, -25.80, 0.00), (5, 0.00, 7.20),
    -> (6, 0.00, 0.00), (6, -51.40, 0.00);

mysql> SELECT i, SUM(d1) AS a, SUM(d2) AS b
    -> FROM t1 GROUP BY i HAVING a <> b;

+------+-------+------+
| i    | a     | b    |
+------+-------+------+
|    1 |  21.4 | 21.4 |
|    2 |  76.8 | 76.8 |
|    3 |   7.4 |  7.4 |
|    4 |  15.4 | 15.4 |
|    5 |   7.2 |  7.2 |
|    6 | -51.4 |    0 |
+------+-------+------+
```

正しい結果です。最初の 5 レコードは比較を満たしていないように見えますが (a と b の値は異なるように見えません)、コンピュータのアーキテクチャー、コンパイラのバージョン、最適化レベルなどの要因によって、小数点以下 1 桁などの数字が異なるためにこのような結果となっている可能性があります。たとえば、CPU が異なると、浮動小数点数の評価が異なることがあります。

カラム d1 および d2 が DOUBLE ではなく DECIMAL として定義されていた場合、SELECT クエリーの結果は 1 行のみ (上記の最後の行) となります。

浮動小数点数の比較を正しく行うには、最初に数値の差異に関して受け入れられる許容度を決定し、許容値に対して比較を行います。たとえば、1 万分の 1 (0.0001) の精度内で同じであれば浮動小数点数が同じであると見なす場合は、許容値より大きい差異を見つけるように比較を記述してください。

```
mysql> SELECT i, SUM(d1) AS a, SUM(d2) AS b FROM t1
    -> GROUP BY i HAVING ABS(a - b) > 0.0001;
+------+-------+------+
| i    | a     | b    |
+------+-------+------+
|    6 | -51.4 |    0 |
+------+-------+------+
1 row in set (0.00 sec)
```

逆に、数値が同じである行を取得する場合は、テストで許容値内での差異を判断するようにします。

```
mysql> SELECT i, SUM(d1) AS a, SUM(d2) AS b FROM t1
    -> GROUP BY i HAVING ABS(a - b) <= 0.0001;
+------+------+------+
| i    | a    | b    |
+------+------+------+
|    1 | 21.4 | 21.4 |
|    2 | 76.8 | 76.8 |
|    3 |  7.4 |  7.4 |
|    4 | 15.4 | 15.4 |
|    5 |  7.2 |  7.2 |
+------+------+------+
```

```
5 rows in set (0.03 sec)
```

浮動小数点値はプラットフォームまたは実装の依存関係の影響を受けます。次のステートメントを実行するとします。

```
CREATE TABLE t1(c1 FLOAT(53,0), c2 FLOAT(53,0));
INSERT INTO t1 VALUES('1e+52','-1e+52');
SELECT * FROM t1;
```

一部のプラットフォームでは、SELECT ステートメントは inf および -inf を返します。ほかのプラットフォームでは、0 および -0 が返されます。

前述の問題は、マスターで mysqldump を使用してテーブルのコンテンツをダンプし、そのダンプファイルをスレーブにリロードすることによってレプリケーションのスレーブを作成しようとすると、浮動小数点カラムを含むテーブルが 2 つのホストで異なる内容になる可能性があることを示しています。

B.5.6 オプティマイザ関連の問題

MySQL はコストベースのオプティマイザを使用して、クエリーを実行する最適な方法を判別しています。多くの場合、MySQL は実行可能な最適なクエリー計画を計算できますが、データに関する情報を十分に取得できず、データについて「学習による」推測を行う必要がある場合があります。

MySQL で「適切」に処理されなかった場合に、MySQL に指示を送るために使用できるツールを次に示します。

- EXPLAIN ステートメントを使用して、MySQL がクエリーを処理する方法に関する情報を取得します。これを使用するには、キーワード EXPLAIN を SELECT ステートメントの前に追加します。

  ```
  mysql> EXPLAIN SELECT * FROM t1, t2 WHERE t1.i = t2.i;
  ```

  EXPLAIN については、セクション13.8.2「EXPLAIN 構文」で詳しく説明しています。

- ANALYZE TABLE tbl_name を使用して、スキャンされるテーブルのキー分布を更新します。セクション 13.7.2.1「ANALYZE TABLE 構文」を参照してください。

- スキャンされるテーブルに FORCE INDEX を使用して、テーブルスキャンは該当するインデックスを使用した場合と比較して著しく負荷が高いことを MySQL に通知します。

  ```
  SELECT * FROM t1, t2 FORCE INDEX (index_for_column)
  WHERE t1.col_name=t2.col_name;
  ```

  USE INDEX および IGNORE INDEX も役に立つことがあります。セクション13.2.9.3「インデックスヒントの構文」を参照してください。

- グローバルおよびテーブルレベルの STRAIGHT_JOIN。セクション13.2.9「SELECT 構文」を参照してください。

- グローバルなシステム変数またはスレッド固有のシステム変数をチューニングできます。たとえば、キースキャンでキー検索が 1,000 回を超えて行われることはないと想定するようにオプティマイザに通知するには、--max-seeks-for-key=1000 オプションを指定して mysqld を起動するか、SET max_seeks_for_key=1000 を使用します。セクション5.1.4「サーバーシステム変数」を参照してください。

B.5.7 テーブル定義関連の問題

B.5.7.1 ALTER TABLE での問題

ALTER TABLE を使用して文字セットまたは文字カラムの照合順序を変更するときに、重複キーエラーを受け取った場合、原因は新しいカラムの照合順序が同じ値に対して 2 つのキーをマップしたか、テーブルが破損していることです。後者の場合は、そのテーブルに対して REPAIR TABLE を実行してください。

ALTER TABLE が次のエラーで停止した場合、問題は ALTER TABLE の初期の処理中に MySQL がクラッシュしたことである可能性があり、A-xxx または B-xxx という名前の古いテーブルが残っていることがあります。

```
Error on rename of './database/name.frm'
to './database/B-xxx.frm' (Errcode: 17)
```

この場合は、MySQL のデータディレクトリに移動して、A- または B- で始まる名前を持つすべてのファイルを削除します。(それらのファイルを削除せずに、別の場所に移動することもできます。)

ALTER TABLE は次のように動作します。

- 要求された構造の変更を適用した A-xxx という名前の新しいテーブルを作成します。
- 元のテーブルから A-xxx にすべての行をコピーします。
- 元のテーブルの名前を B-xxx に変更します。
- A-xxx を元のテーブル名に変更します。
- B-xxx を削除します。

名前変更の操作に問題があった場合、MySQL は変更を取り消そうとします。重大な問題がある場合 (発生することはないはずですが)、MySQL は古いテーブルを B-xxx として残すことがあります。システムレベルでテーブルファイルを単純に名前変更することによって、データは元に戻ります。

トランザクションテーブルに対して ALTER TABLE を使用したとき、または Windows を使用しているときに、LOCK TABLE が発行されていた場合、ALTER TABLE はテーブルをロック解除します。これが行われるのは、InnoDB およびこれらのオペレーティングシステムは使用されているテーブルをドロップできないためです。

B.5.7.2 TEMPORARY テーブルに関する問題

次のリストは、TEMPORARY テーブルの使用に関する制限を示しています。

- TEMPORARY テーブルのタイプは、MEMORY、MyISAM、MERGE、または InnoDB のみです。

  MySQL Cluster では、一時テーブルはサポートされません。

- 同じクエリーで TEMPORARY テーブルを複数回参照することはできません。たとえば、次のステートメントは動作しません。

```
mysql> SELECT * FROM temp_table, temp_table AS t2;
ERROR 1137: Can't reopen table: 'temp_table'
```

  このエラーは、ストアドファンクション内で別のエイリアスで一時テーブルが複数回参照された場合にも発生します。これは、ファンクション内の別のステートメントで参照されていても発生します。

- SHOW TABLES ステートメントでは TEMPORARY テーブルは一覧表示されません。
- RENAME を使用して TEMPORARY テーブルの名前を変更することはできません。ただし、代わりに ALTER TABLE を使用できます。

```
mysql> ALTER TABLE orig_name RENAME new_name;
```

- レプリケーションで一時テーブルを使用する場合の既知の問題があります。詳細は、セクション17.4.1「レプリケーションの機能と問題」を参照してください。
- 一時テーブルがストアドファンクションの外部で作成されて、複数の呼び出し元関数および呼び出し先関数で参照される場合は、次のエラーが発生することがあります。

```
ERROR 1137: Can't reopen table: 'temp_table'
```

B.5.8 MySQL の既知の問題

このセクションでは、最新バージョンの MySQL の既知の問題を一覧表示します。

プラットフォーム固有の問題については、セクション2.1「一般的なインストールガイド」およびセクション24.4「MySQL のデバッグおよび移植」のインストールおよび移植の手順を参照してください。

次の問題は既知の問題です。

- IN のサブクエリーの最適化は、= ほど効果はありません。
- lower_case_table_names=2 (データベース名およびテーブル名に大文字/小文字のどちらが使用されたかを MySQL が認識するようになります) を使用していても、MySQL が関数 DATABASE() のデータベース名、またはさまざまな口グ内 (大文字/小文字が区別されないシステムの) で使用された表記を識別できません。
- スレーブで制約に別の名前がある可能性があるため、FOREIGN KEY 制約のドロップがレプリケーションで動作しません。
- REPLACE (および REPLACE オプションを指定した LOAD DATA) で ON DELETE CASCADE がトリガーされません。

- DISTINCT リストに指定されたすべてのカラムのみを使用しない場合、GROUP_CONCAT() 内で ORDER BY を指定した DISTINCT が動作しません。
- 大きい整数値 (2^{63} から 2^{64}−1 まで) を 10 進数カラムまたは文字列カラムに挿入する場合、数値は符号付き整数のコンテキストで評価されるため、負の値として挿入されます。
- ANALYZE TABLE、OPTIMIZE TABLE、および REPAIR TABLE が、INSERT DELAYED を使用している非トランザクションテーブルで問題となることがあります。
- ステートメントベースのバイナリロギングでは、マスターは実行されたクエリーをバイナリログに書き込みます。これは、ほとんどの場合に理想的に動作する非常に高速かつコンパクトで効率的なロギング方法です。ただし、データの変更が非決定的であるようにクエリーが設計されている (通常、レプリケーション以外でも推奨されるやり方ではありません) 場合、マスターおよびスレーブのデータで相違が発生する可能性があります。

  例:

  - ゼロ値または NULL 値を AUTO_INCREMENT カラムに挿入する CREATE TABLE ... SELECT ステートメントまたは INSERT ... SELECT ステートメント。
  - ON DELETE CASCADE プロパティーが指定された外部キーを持つテーブルから行を削除する場合の DELETE。
  - 挿入されるデータに重複キー値がある場合の REPLACE ... SELECT、INSERT IGNORE ... SELECT。

  これは、前述したクエリーに決定性順序を保証する ORDER BY 句がない場合にのみ発生することがあります。

  たとえば、ORDER BY が指定されていない INSERT ... SELECT の場合、マスターおよびスレーブでのオプティマイザの選択によって、SELECT が異なる順序で行を返すことがあります (それにより、行が異なるランクを持つことになり、AUTO_INCREMENT カラムで異なる数値が取得されます)。

  次の場合にのみ、マスターとスレーブでクエリーの最適化が異なる結果となります。

  - マスターとスレーブで、テーブルが異なるストレージエンジンを使用して格納される。(マスターとスレーブで異なるストレージエンジンを使用できます。たとえば、マスターでは InnoDB を使用するが、スレーブの使用可能なディスク領域が少ない場合は、スレーブで MyISAM を使用できます。)
  - MySQL のバッファーサイズ (key_buffer_size など) がマスターとスレーブで異なる。
  - マスターとスレーブで異なるバージョンの MySQL を実行していて、オプティマイザのコードがそれらのバージョンで異なる。

  この問題は、mysqlbinlog|mysql を使用したデータベースのリストアに影響することもあります。

  この問題を回避するもっとも簡単な方法は、行が常に同じ順序で格納または変更されるように、ORDER BY 句を前述の非決定性クエリーに追加することです。行ベースのロギング形式または混合したロギング形式を使用することでも、この問題が回避されます。
- スタートアップオプションにファイル名を指定しない場合、ログファイル名はサーバーのホスト名に基づいています。ホスト名を別の名前に変更した場合に同じログファイル名のままにするには、--log-bin=old_host_name-bin などのオプションを明示的に使用する必要があります。セクション5.1.3「サーバーコマンドオプション」を参照してください。または、ホスト名の変更が反映されるように、古いファイルを名前変更します。バイナリログの場合は、バイナリログのインデックスファイルを編集して、そこにあるバイナリログファイル名も修正する必要があります。(スレーブサーバーのリレーログも同様です。)
- LOAD DATA INFILE ステートメントのあとに残っている一時ファイルが、mysqlbinlog によって削除されません。セクション4.6.8「mysqlbinlog — バイナリログファイルを処理するためのユーティリティー」を参照してください。
- RENAME が TEMPORARY テーブル、または MERGE テーブルで使用されているテーブルで動作しません。
- SET CHARACTER SET を使用したときに、データベース、テーブル、およびカラムの名前に変換された文字を使用できません。
- LIKE ... ESCAPE の ESCAPE で、「_」 または 「%」 を使用できません。
- データ値を比較するときに、最初の max_sort_length バイトのみが使用されます。これは、最初の max_sort_length バイトで値が有意に識別されない場合、GROUP BY、ORDER BY、または DISTINCT で値を信頼して使用できないことを意味します。これを回避するには、変数値を増やします。max_sort_length のデフォルト値は 1024 であり、サーバーの起動時または実行時に変更できます。

- 数値計算は BIGINT または DOUBLE (通常、どちらも長さは 64 ビットです) で行われます。返される精度は関数によって異なります。一般的なルールとしては、ビット関数は BIGINT の精度、IF() と ELT() は BIGINT または DOUBLE の精度、および残りは DOUBLE の精度で実行されます。符号なしの long long 値がビットフィールド以外で 63 ビット (9223372036854775807) を超える値に解決される場合は、使用しないようにしてください。
- 1 つのテーブルには、最大 255 個の ENUM カラムおよび SET カラムを作成できます。
- 現在、MIN()、MAX()、およびその他の集約関数では、MySQL はセット内の文字列の相対位置ではなく文字列値で ENUM カラムおよび SET カラムを比較します。
- UPDATE ステートメントでは、カラムは左から右に更新されます。更新されたカラムを参照している場合は、元の値ではなく更新された値が取得されます。たとえば、次のステートメントでは KEY に 1 ではなく 2 がインクリメントされます。

```
mysql> UPDATE tbl_name SET KEY=KEY+1,KEY=KEY+1;
```

- 同じクエリーで複数の一時テーブルを参照することはできますが、特定の一時テーブルを複数回参照することはできません。たとえば、次のステートメントは動作しません。

```
mysql> SELECT * FROM temp_table, temp_table AS t2;
ERROR 1137: Can't reopen table: 'temp_table'
```

- 結合で「隠し」カラムを使用している場合は、オプティマイザでの DISTINCT の処理が異なることがあります。結合では、隠しカラムは結果の一部としてカウントされますが (表示されていなくても)、通常のクエリーでは、隠しカラムは DISTINCT 比較で考慮されません。

  次に例を示します。

```
SELECT DISTINCT mp3id FROM band_downloads
    WHERE userid = 9 ORDER BY id DESC;
```

  および

```
SELECT DISTINCT band_downloads.mp3id
    FROM band_downloads,band_mp3
    WHERE band_downloads.userid = 9
    AND band_mp3.id = band_downloads.mp3id
    ORDER BY band_downloads.id DESC;
```

  2 番目の例では、MySQL Server 3.23.x を使用すると、結果セットに 2 つの同じ行が取得されることがあります (隠しカラム id の値が異なる可能性があるためです)。

  これは、結果に ORDER BY のカラムがないクエリーでのみ発生します。
- 空のセットを返すクエリーに関する PROCEDURE を実行すると、PROCEDURE でカラムが変換されないことがあります。
- MERGE タイプのテーブルの作成で、基礎テーブルが互換性のあるタイプであるかどうかがチェックされません。
- ALTER TABLE を使用して、MERGE テーブルで使用されるテーブルに UNIQUE インデックスを追加し、次に MERGE テーブルに通常のインデックスを追加したときに、テーブルに古い UNIQUE ではないキーがあった場合、それらのテーブルのキー順序は異なります。これは、重複キーをできるだけ早く検出できるように、ALTER TABLE が通常のインデックスより UNIQUE インデックスを優先するためです。

付録 C MySQL リリースノート

MySQL Server および関連製品のリリースノートは、MySQL リファレンスマニュアルの一部としてではなく、スタンドアロン形式で発行されます。リリースノートについては、次のドキュメントを参照してください。

- MySQL 5.6 リリースノート
- MySQL Cluster NDB 7.3 リリースノート
- MySQL Cluster NDB 7.4 リリースノート
- MySQL Connector/ODBC リリースノート
- MySQL Connector/Net リリースノート (MySQL Visual Studio Plugin リリースノートを含む)
- MySQL Connector/J リリースノート
- MySQL Connector/C++ リリースノート
- MySQL Proxy リリースノート
- MySQL for Excel リリースノート
- MySQL Installer リリースノート

付録 D 制約と制限

目次

D.1 ストアドプログラムの制約 2973
D.2 条件処理の制約 2976
D.3 サーバー側のカーソルの制約 2976
D.4 サブクエリーの制約 2977
D.5 ビューの制約 2978
D.6 XA トランザクションの制約 2979
D.7 文字セットの制約 2980
D.8 パフォーマンススキーマの制約 2980
D.9 プラガブルな認証の制約 2981
D.10 MySQL での制限 2982
D.10.1 結合の制限 2982
D.10.2 データベースおよびテーブルの数に対する制限 2982
D.10.3 テーブルサイズの制限 2983
D.10.4 テーブルカラム数と行サイズの制限 2984
D.10.5 .frm ファイル構造により課せられる制限 2985
D.10.6 Windows プラットフォームの制限 2986

ここでは、サブクエリーやビューなどの MySQL 機能の使用に適用される制約について説明します。

D.1 ストアドプログラムの制約

これらの制約は、第20章「ストアドプログラムおよびビュー」で説明している機能に適用されます。

ここに記載されている制約の中には、すべてのストアドルーチン、つまりストアドプロシージャーとストアドファンクションの両方に適用されるものがあります。また、ストアドプロシージャーには適用されず、ストアドファンクションに固有の制約もいくつか存在します。

ストアドファンクションの制約は、トリガーにも適用されます。トリガーに固有の制約もいくつかあります。

ストアドプロシージャーの制約は、イベントスケジューラのイベント定義の DO 句にも適用されます。イベントに固有の制約もいくつかあります。

ストアドルーチンでは許可されていない SQL ステートメント

ストアドルーチンには自由に SQL ステートメントを含めることはできません。次のステートメントは許可されていません。

- LOCK TABLES および UNLOCK TABLES のロックステートメント。
- ALTER VIEW。
- LOAD DATA および LOAD TABLE。
- SQL 準備済みステートメント (PREPARE、EXECUTE、DEALLOCATE PREPARE) は、ストアドプロシージャーで使用できますが、ストアドファンクションやトリガーでは使用できません。そのため、ストアドファンクションとトリガーは動的 SQL (この場合はステートメントを文字列として構築してから実行します) を使用できません。
- 通常、SQL 準備済みステートメントで許可されていないステートメントは、ストアドプログラムでも許可されません。準備済みステートメントとしてサポートされているステートメントのリストについては、セクション13.5「準備済みステートメントのための SQL 構文」を参照してください。例外は SIGNAL、RESIGNAL、および GET DIAGNOSTICS であり、これらは準備済みステートメントとして許可されていませんが、ストアドプログラムで許可されます。
- ローカル変数はストアドプログラムの実行中にのみスコープ内にあるので、これらの参照は、ストアドプログラム内で作成された準備済みステートメントでは許可されていません。準備済みステートメントのスコープは現在のセッションであり、ストアドプログラムではないので、ステートメントはプログラムの終了後に実行でき、この時点で変数はスコープ内に存在しなくなります。たとえば、SELECT ... INTO local_var は準備済みステートメントとして使用できません。この制約は、ストアドプロシージャーおよびストアドファンクションのパラメータにも適用されます。セクション13.5.1「PREPARE 構文」を参照してください。

- 挿入は遅延されません。INSERT DELAYED 構文は受け入れられますが、ステートメントは通常の INSERT として扱われます。
- すべてのストアドプログラム (ストアドプロシージャーとストアドファンクション、トリガー、およびイベント) 内で、パーサーは、BEGIN [WORK] を BEGIN ... END ブロックの開始として扱います。このコンテキストでトランザクションを開始するには、代わりに START TRANSACTION を使用します。

ストアドファンクションの制約

次の追加ステートメントまたは操作は、ストアドファンクション内で許可されていません。これらはストアドプロシージャーで許可されていますが、ストアドファンクションまたはトリガー内から呼び出されるストアドプロシージャーを除きます。たとえば、ストアドプロシージャーで FLUSH を使用する場合、ストアドファンクションまたはトリガーからそのストアドプロシージャーを呼び出すことはできません。

- 明示的または暗黙的なコミットまたはロールバックを実行するステートメント。これらのステートメントのサポートは、SQL 標準では必要ありません。SQL 標準では、各 DBMS ベンダーがこれらのステートメントを許可するかどうかを決められると定めています。
- 結果セットを返すステートメント。これには、INTO var_list 句を含まない SELECT ステートメントや、SHOW、EXPLAIN、および CHECK TABLE などのほかのステートメントも含まれます。関数は、SELECT ... INTO var_list を使用するか、カーソルと FETCH ステートメントを使用すると、結果セットを処理できます。セクション13.2.9.1「SELECT ... INTO 構文」およびセクション13.6.6「カーソル」を参照してください。
- FLUSH ステートメント。
- ストアドファンクションは再帰的に使用できません。
- ストアドファンクションまたはトリガーは、そのストアドファンクションまたはトリガーを呼び出したステートメントによって (読み取りまたは書き込みに) すでに使用されているテーブルを変更できません。
- ストアドファンクションで、一時テーブルを異なるエイリアスで複数回参照する場合、ストアドファンクション内の別々のステートメントで参照を行う場合でも、「表を再オープンできません: 'tbl_name'」 というエラーが発生します。
- ストアドファンクションを呼び出す HANDLER ... READ ステートメントは、レプリケーションエラーを引き起こす可能性があり、許可されません。

トリガーの制約

トリガーの場合、さらに次の制約が適用されます。

- トリガーは外部キーアクションでアクティブ化されません。
- 行ベースのレプリケーションを使用する場合、スレーブのトリガーは、マスターから発行されたステートメントでアクティブ化されません。スレーブのトリガーは、ステートメントベースのレプリケーションを使用するときにアクティブ化されます。詳細は、セクション17.4.1.32「レプリケーションとトリガー」を参照してください。
- RETURN ステートメントはトリガーでは許可されていません。トリガーは値を返すことができません。すぐにトリガーを終了するには、LEAVE ステートメントを使用します。
- トリガーは、mysql データベース内のテーブルでは許可されていません。
- トリガーキャッシュは、ベースとなるオブジェクトのメタデータが変更された場合は検出しません。トリガーがテーブルを使用し、トリガーがキャッシュにロードされたあとにそのテーブルに変更があった場合、トリガーは古いメタデータを使用して動作します。

ストアドルーチン内の名前競合

ルーチンパラメータ、ローカル変数、およびテーブルカラムに同じ識別子が使用される場合があります。また、同じローカル変数名を、ネスト化されたブロックで使用することもできます。例:

```
CREATE PROCEDURE p (i INT)
BEGIN
  DECLARE i INT DEFAULT 0;
  SELECT i FROM t;
  BEGIN
```

```
  DECLARE i INT DEFAULT 1;
  SELECT i FROM t;
 END;
END;
```

このような場合、識別子はあいまいになり、次の優先順位ルールが適用されます。

- ローカル変数では、ルーチンパラメータやテーブルカラムが優先されます。
- ルーチンパラメータでは、テーブルカラムが優先されます。
- 内部ブロック内のローカル変数では、外部ブロック内のローカル変数が優先されます。

変数でテーブルカラムが優先される動作は、非標準です。

レプリケーションに関する考慮事項

ストアドルーチンを使用すると、レプリケーションの問題が生じることがあります。この問題については、セクション20.7「ストアドプログラムのバイナリロギング」で詳しく述べられています。

--replicate-wild-do-table=db_name.tbl_name オプションはテーブル、ビュー、およびトリガーに適用されます。ストアドプロシージャーと関数、またはイベントには適用されません。後者のオブジェクトで作用するステートメントをフィルタするには、1 つまたは複数の --replicate-*-db オプションを使用します。

デバッグに関する考慮事項

ストアドルーチンのデバッグ機能は存在しません。

SQL:2003 標準のサポート外の構文

MySQL ストアドルーチン構文は SQL:2003 標準に基づきます。この標準の次の項目は現在サポートされていません。

- UNDO ハンドラ
- FOR ループ

並列性に関する考慮事項

セッション間のやり取りの問題を防止するために、クライアントのステートメント発行時、サーバーではステートメントの実行に利用できるルーチンとトリガーのスナップショットが使用されます。つまり、サーバーは、ステートメントの実行中に使用される可能性のあるプロシージャー、関数、およびトリガーのリストを算出してロードし、ステートメントの実行に進みます。ステートメントの実行時は、ほかのセッションが実行するルーチンへの変更は認識されません。

並列性を最大にするために、ストアドファンクションでは、その副作用を最小限に抑える必要があります。特に、ストアドファンクション内のテーブルを更新することにより、そのテーブルでの並列操作が減少することがあります。ストアドファンクションは、実行前にテーブルロックを取得して、ステートメントが実行する順序とログに表示されるときの順序の不一致によるバイナリログの不整合を回避します。ステートメントベースのバイナリロギングが使用される場合、関数内で実行されるステートメントではなく、関数を呼び出すステートメントが記録されます。その結果、ベースとなる同じテーブルを更新するストアドファンクションは、並列で実行しません。対照的に、ストアドプロシージャーはテーブルレベルのロックを取得しません。ストアドプロシージャー内で実行されたすべてのステートメントは、ステートメントベースのバイナリロギングの場合でも、バイナリログに書き込まれます。セクション20.7「ストアドプログラムのバイナリロギング」を参照してください。

イベントスケジューラの制約

次の制限は、イベントスケジューラに固有のものです。

- イベント名は大文字と小文字を区別せずに処理されます。たとえば、anEvent と AnEvent という名前の 2 つのイベントを同じデータベース内に含めることはできません。
- イベントは、ストアドルーチン、トリガー、または別のイベントによって、作成、変更、削除できません。イベントは、ストアドルーチンやトリガーを作成、変更、削除することもできません。(Bug #16409、Bug #18896)
- LOCK TABLES ステートメントの有効時は、イベントでの DDL ステートメントは禁止されています。

- YEAR、QUARTER、MONTH、および YEAR_MONTH の間隔を使用したイベントのタイミングは、月で解決されます。ほかの間隔を使用したタイミングは秒で解決されます。同時に行われるようにスケジュール設定されたイベントは、指定の順序で実行できません。さらに、丸め、スレッドアプリケーションの特性、およびイベントを作成しその実行を信号で伝えるためにゼロ以外の時間長が必要になるため、イベントが 1、2 秒ほど遅れる場合があります。ただし、INFORMATION_SCHEMA.EVENTS テーブルの LAST_EXECUTED カラム、または mysql.event テーブルの last_executed カラムに示された時間は、常に実際のイベント実行時間の 1 秒以内の精度になります。(Bug #16522 も参照してください。)
- イベントの本体に含まれるステートメントはそれぞれ新しい接続で実行されるため、これらのステートメントは特定のユーザーセッションで、SHOW STATUS によって表示される Com_select や Com_insert などのサーバーのステートメント数に影響しません。ただし、このような数はグローバルスコープで更新されます。(Bug #16422)
- イベントは、Unix エポックの最後の時間以降をサポートしません。この時間は 2038 年の年頭あたりになります。このような日付はイベントスケジューラで特に許可されません。(Bug #16396)
- CREATE EVENT および ALTER EVENT ステートメントの ON SCHEDULE 句でのストアドファンクション、ユーザー定義関数、およびテーブルの参照はサポートされていません。このような種類の参照は許可されていません。(詳細は Bug #22830 を参照してください。)

D.2 条件処理の制約

SIGNAL、RESIGNAL、および GET DIAGNOSTICS は準備済みのステートメントとして許可されていません。たとえば、次のステートメントは無効です。

```
PREPARE stmt1 FROM 'SIGNAL SQLSTATE "02000"';
```

クラス '04' の SQLSTATE 値は特別扱いされません。ほかの例外と同じように扱われます。

標準 SQL には、ネスト化された実行のコンテキストごとの診断領域を含んだ、診断領域スタックがあります。標準 SQL 構文には、スタック領域を参照するための GET STACKED DIAGNOSTICS が含まれます。MySQL では、もっとも新しく書き込んだステートメントの情報を含む単一の診断領域があるため、STACKED キーワードをサポートしません。セクション13.6.7.7「MySQL の診断領域」も参照してください。

標準 SQL では、最初の条件は、以前の SQL ステートメントに対して返される SQLSTATE 値に関連します。MySQL ではこれは保証されていないので、メインエラーを取得するために、次のようにはできません。

```
GET DIAGNOSTICS CONDITION 1 @errno = MYSQL_ERRNO;
```

代わりに次のようにします。

```
GET DIAGNOSTICS @cno = NUMBER;
GET DIAGNOSTICS CONDITION @cno @errno = MYSQL_ERRNO;
```

D.3 サーバー側のカーソルの制約

サーバー側のカーソルは、mysql_stmt_attr_set() 関数を使用して C API に実装されます。ストアドルーチンのカーソルにも同じ実装が使用されます。サーバー側のカーソルによって、サーバー側で結果セットを生成できるようになりますが、クライアントが要求する行を除いてクライアントに転送することはできません。たとえば、クライアントがクエリーを実行するが、最初の行のみが必要な場合、残りの行は転送されません。

MySQL では、サーバー側のカーソルは内部一時テーブルに実体化されます。最初これは MEMORY テーブルですが、そのサイズが max_heap_table_size および tmp_table_size システム変数の最小値を超えると、MyISAM テーブルに変換されます。カーソルの結果セットを保持するために作成された内部一時テーブルには、内部一時テーブルをほかに使用する場合と同じ制約が適用されます。セクション8.4.4「MySQL が内部一時テーブルを使用する仕組み」を参照してください。この実装の制限の 1 つには、大きな結果セットの場合に、カーソルによる行の取得に時間がかかることがあるというものがあります。

カーソルは読み取り専用です。カーソルを使用して行を更新できません。

UPDATE WHERE CURRENT OF および DELETE WHERE CURRENT OF は、更新可能なカーソルがサポートされていないため実装されません。

カーソルは保持不可能です (コミット後、開いたままにはできません)。

カーソルは非センシティブです。

カーソルはスクロール不可です。

カーソルには名前が付けられません。ステートメントハンドラがカーソル ID として機能します。

準備済みステートメントごとに、カーソルを 1 つだけ開いておくことができます。複数のカーソルが必要な場合は、複数のステートメントを準備する必要があります。

結果セットを生成するステートメントで、準備モードでサポートされていないものにはカーソルを使用できません。このようなステートメントには、CHECK TABLE、HANDLER READ、SHOW BINLOG EVENTS などがあります。

D.4 サブクエリーの制約

- IN に対するサブクエリーの最適化は、= 演算子または IN(value_list) 演算子の場合のようには効果的ではありません。

  IN サブクエリーのパフォーマンスが不十分である一般的な例では、サブクエリーが少数の行を返すのに対し、外部クエリーがサブクエリーの結果と比較して多数の行を返す場合があります。

  この問題は、IN サブクエリーを使用するステートメントでは、オプティマイザが相関サブクエリーに書き換えることにあります。非相関サブクエリーを使用する次のステートメントについて説明します。

  ```
  SELECT ... FROM t1 WHERE t1.a IN (SELECT b FROM t2);
  ```

  オプティマイザがステートメントを相関サブクエリーに書き換えます。

  ```
  SELECT ... FROM t1 WHERE EXISTS (SELECT 1 FROM t2 WHERE t2.b = t1.a);
  ```

  内部クエリーと外部クエリーがそれぞれ M 行と N 行を返す場合、実行時間は、非相関サブクエリーの場合の O(M+N) ではなく、O(M×N) になります。

  つまり、IN サブクエリーは、このサブクエリーが返す値と同じ値をリストする IN(value_list) 演算子を使用して作成されたクエリーよりもかなり遅くなる場合があります。

- 一般に、テーブルを変更することも、サブクエリーの同じテーブルから選択することもできません。たとえば、この制限は次の形式のステートメントに適用されます。

  ```
  DELETE FROM t WHERE ... (SELECT ... FROM t ...);
  UPDATE t ... WHERE col = (SELECT ... FROM t ...);
  {INSERT|REPLACE} INTO t (SELECT ... FROM t ...);
  ```

  例外: FROM 句内の変更されたテーブルでサブクエリーを使用している場合、前述の禁止事項は適用されません。例:

  ```
  UPDATE t ... WHERE col = (SELECT * FROM (SELECT ... FROM t...) AS _t ...);
  ```

  ここでは、FROM 句内のサブクエリーの結果が一時テーブルとして格納されるので、t 内の対応する行は、t に対する更新が行われる前にすでに選択されています。

- 行比較演算は一部のみサポートされています。

  - expr [NOT] IN subquery の場合、expr は n タプル (行コンストラクタ構文を使用して指定します) にでき、サブクエリーは n タプルの行を返すことができます。したがって、許可されている構文は、具体的には row_constructor [NOT] IN table_subquery と表されます

  - expr op {ALL|ANY|SOME} subquery の場合、expr はスカラー値にする必要があり、サブクエリーはカラムサブクエリーにする必要があります。複合カラム行を返すことはできません。

  つまり、n タプルの行を返すサブクエリーの場合、次のものはサポートされています。

  ```
  (expr_1, ..., expr_n) [NOT] IN table_subquery
  ```

  ただし、次のものはサポートされていません。

  ```
  (expr_1, ..., expr_n) op {ALL|ANY|SOME} subquery
  ```

  IN の行比較がサポートされているのに、他はサポートされていない理由は、IN が、= 比較および AND 演算のシーケンスに、これを書き換えることによって実装されているためです。この方法は、ALL 、ANY、SOME には使用できません。

- FROM 句のサブクエリーは相関サブクエリーにはできません。これらは、クエリー実行中にすべて実体化 (結果セットを生成するように評価) されるので、外部クエリーの行ごとに評価できません。MySQL 5.6.3 より前で

は、実体化は外部クエリーの評価の前に行われます。5.6.3 以降では、オプティマイザは、結果が必要になるまで実体化を遅らせ、これにより実体化を回避できます。FROM 句内のサブクエリー (派生テーブル) の最適化を参照してください。

- MySQL は特定のサブクエリー演算子にサブクエリーの LIMIT をサポートしていません。

```
mysql> SELECT * FROM t1
    ->   WHERE s1 IN (SELECT s2 FROM t2 ORDER BY s1 LIMIT 1);
ERROR 1235 (42000): This version of MySQL doesn't yet support
 'LIMIT & IN/ALL/ANY/SOME subquery'
```

- オプティマイザは、サブクエリーに対してよりも結合に対するほうが完成度が高いので、多くの場合、サブクエリーを使用するステートメントは、結合として書き換えた場合のほうが効率的に実行できます。

  IN サブクエリーを SELECT DISTINCT 結合として書き換えることができる場合には、例外が生じます。例:

```
SELECT col FROM t1 WHERE id_col IN (SELECT id_col2 FROM t2 WHERE condition);
```

  このステートメントは次のように書き換えることができます。

```
SELECT DISTINCT col FROM t1, t2 WHERE t1.id_col = t2.id_col AND condition;
```

- MySQL では、サブクエリーで行をテーブルに挿入するなどのデータ変更の副作用があるストアドファンクションを参照できます。たとえば、f() が行を挿入する場合、次のクエリーはデータを変更できます。

```
SELECT ... WHERE x IN (SELECT f() ...);
```

  この動作は、SQL 標準に対する拡張です。MySQL では、オプティマイザでの処理方法の選択に応じて、特定のクエリーの実行ごとに f() が異なる回数実行される場合があるため、これにより不確定な結果が生成される可能性があります。

  ステートメントベースまたは混合形式のレプリケーションの場合、この不確定性には、このようなクエリーが、マスターとそのスレーブで異なる結果を生成することがあります。

- MySQL 5.6.3 より前では、FROM 句のサブクエリーは、結果を一時テーブルに実体化することにより評価され、このテーブルではインデックスが使用されません。5.6.3 以降では、クエリー実行がさらに高速になる場合は、オプティマイザは実体化されたテーブルにインデックスを作成します。FROM 句内のサブクエリー (派生テーブル) の最適化を参照してください。

D.5 ビューの制約

ビューの処理は最適化されていません。

- ビューにはインデックスを作成できません。
- マージアルゴリズムを使用して処理されたビューに、インデックスを使用することは可能です。ただし、TEMPTABLE アルゴリズムで処理されたビューは、そのベースとなるテーブルのインデックスを利用できません (ただし、一時テーブルの作成中にはインデックスを使用できます)。

ビューの FROM 句ではサブクエリーは使用できません。

一般的な原則では、テーブルを変更することも、サブクエリーの同じテーブルから選択することもできません。セクションD.4「サブクエリーの制約」を参照してください。

テーブルから選択するビューを選ぶ場合、ビューがサブクエリーのテーブルから選択される場合や、ビューがマージアルゴリズムを使用して評価される場合にも、同じ原則が適用されます。例:

```
CREATE VIEW v1 AS
SELECT * FROM t2 WHERE EXISTS (SELECT 1 FROM t1 WHERE t1.a = t2.a);

UPDATE t1, v2 SET t1.a = 1 WHERE t1.b = v2.b;
```

一時テーブルを使用してビューが評価される場合、ビューサブクエリーのテーブルから選択し、さらに外部クエリーでそのテーブルを変更することが可能です。この場合、ビューは一時テーブルに格納されるため、実際にはビューをサブクエリーのテーブルから選択して「同時に」変更しているのではありません。(これは、ビュー定義で ALGORITHM = TEMPTABLE を指定することによって、MySQL で TEMPTABLE アルゴリズムを強制的に使用させる 1 つの理由です。)

DROP TABLE または ALTER TABLE を使用すると、ビュー定義で使用されているテーブルを削除または変更できます。ビューを無効化する場合でも、DROP または ALTER 操作によって警告が発せられることはありま

せん。代わりに、あとからビューを使用するときにエラーが発生します。CHECK TABLE は、DROP または ALTER 操作で無効化されたビューのチェックに使用できます。

MySQL 5.6.5 より前では、ビュー定義は特定のステートメントによって「固定」されます。PREPARE で準備されたステートメントがビューを参照する場合、あとでステートメントが実行されるたびに確認されるビュー定義は、準備された時点のビュー定義になります。これは、ステートメントが準備されたあと、実行される前に、ビュー定義が変更されても同じことです。次の例で、EXECUTE ステートメントで返される結果は、現在の日時ではなく、ランダムな数値になります。

```
CREATE VIEW v AS SELECT RAND();
PREPARE s FROM 'SELECT * FROM v';
ALTER VIEW v AS SELECT NOW();
EXECUTE s;
```

ビューの更新可能性に関しては、どのビューでも理論的に更新可能であれば、実際に更新可能である必要があるというのがビューの全体的な目的です。これには、定義に UNION が記されているビューも含まれています。現時点では、理論的に更新可能なすべてのビューが、実際に更新可能というわけではありません。最初のビュー実装は、使用可能で更新可能なビューをできるだけ迅速に MySQL に提供するため、故意にこのように書かれていました。現在、理論的に更新可能なビューの多くは更新できますが、制限はまだ存在します。

- WHERE 句以外の場所にサブクエリーがある更新可能なビュー。SELECT リストにサブクエリーがあるビューの中には、更新可能な場合があります。
- UPDATE を使用して、結合として定義されたビューの複数のベースとなるテーブルは更新できません。
- DELETE を使用して、結合として定義されたビューは更新できません。

ビューの現在の実装には欠点があります。ビューの作成に必要な基本権限 (CREATE VIEW 権限と SELECT 権限) がユーザーに付与されていても、SHOW VIEW 権限も付与されていない場合、オブジェクトで SHOW CREATE VIEW を呼び出すことはできません。

権限の不足のために mysqldump が失敗する可能性があるという欠点によって、このコマンドを使用したデータベースのバックアップで問題が発生する場合があります。この問題については Bug #22062 で説明しています。

問題の回避策として、ビューが作成されたときに MySQL が暗黙的に SHOW VIEW 権限を与えないため、CREATE VIEW を認められているユーザーに、管理者が手動でこの権限を付与します。

ビューにはインデックスがないので、インデックスのヒントは適用されません。ビューからの選択時のインデックスヒントの使用は許可されていません。

SHOW CREATE VIEW は、カラムごとに AS alias_name 句を使用してビュー定義を表示します。式からカラムを作成する場合、デフォルトのエイリアスは式テキストになり、かなり長くなることがあります。CREATE VIEW ステートメント内のカラム名に対するエイリアスは、(256 文字の最大のエイリアス長ではなく) 64 文字の最大のカラム長に対してチェックされます。その結果、いずれかのカラムエイリアスが 64 文字を超えた場合、SHOW CREATE VIEW の出力から作成されたビューは失敗します。これによって、長すぎるエイリアスを持つビューに対し、次の環境で問題が起きる可能性があります。

- ビュー定義は、カラム長の制約を強制する新しいスレーブに複製できません。
- mysqldump で作成されたダンプファイルは、カラム長の制約を強制するサーバーにロードできません。

これらの問題を回避するには、短いカラム名を提供するエイリアスを使用するように、問題のある各ビュー定義を変更します。この場合、ビューは正しく複製され、エラーを生成しないでダンプおよびリロードできます。定義を変更するには、DROP VIEW および CREATE VIEW でビューを削除してから再度作成するか、CREATE OR REPLACE VIEW を使用して定義を置き換えます。

ダンプファイルのビュー定義リロード時に問題が発生する場合は、CREATE VIEW ステートメントを変更するようにダンプファイルを編集するとこの問題を回避できます。ただし、これは元のビュー定義を変更しないので、その後のダンプ操作の問題を引き起こす可能性があります。

D.6 XA トランザクションの制約

XA トランザクションのサポートは、InnoDB ストレージエンジンに限られています。

「外部 XA」 の場合、MySQL Server がリソースマネージャーとして機能し、クライアントプログラムがトランザクションマネージャーとして機能します。「内部 XA」 の場合、MySQL サーバー内のストレージエンジンがリソースマネージャーとして機能し、サーバー自体がトランザクションマネージャーとして機能します。内部 XA サポートは、個々のストレージエンジンの機能によって制限されています。内部 XA は、複数のストレージエンジンが関与する XA トランザクションを処理するために必要になります。内部 XA の実装では、ストレージエンジ

ンがテーブルハンドラレベルでの 2 フェーズコミットをサポートしている必要であり、現在これは InnoDB にのみ当てはまります。

XA START では、JOIN および RESUME 句はサポートされていません。

XA END では、SUSPEND [FOR MIGRATE] 句はサポートされていません。

xid 値の bqual 部分が、グローバルトランザクション内の XA トランザクションごとに異なる必要があるという要件が、現在の MySQL XA 実装の制限です。これは XA の仕様によるものではありません。

XA トランザクションが PREPARED 状態に達していれば、MySQL サーバーが (たとえば、Unix の kill -9 で) 強制終了されたり、異常にシャットダウンしたりした場合でも、サーバーが再起動したあとでトランザクションを継続できます。ただし、クライアントが再接続し、トランザクションをコミットした場合、そのトランザクションは、コミットされていてもバイナリログには記録されません。これは、データとバイナリログの同期性がなくなったことを意味します。XA はレプリケーションとともに安全に使用できません。

PREPARED 状態に達しているものも含めた保留中の XA トランザクションを、サーバーがロールバックする可能性があります。これは、クライアントの接続が終了して、サーバーが起動を続ける場合、またはクライアントが接続していて、サーバーが正常にシャットダウンする場合に起こります。(後者の場合、サーバーは、各接続に終了というマークを付けから、関連付けられた PREPARED XA トランザクションをロールバックします。)PREPARED XA トランザクションをコミットまたはロールバック可能である必要がありますが、これはバイナリロギングメカニズムに変更を加えずに行うことはできません。

D.7 文字セットの制約

- 識別子は、utf8 を使用して mysql データベーステーブル (user、db など) に格納されますが、識別子には Basic Multilingual Plane (BMP) の文字だけを含めることができます。識別子では補助文字は許可されません。
- ucs2、utf16、utf16le、および utf32 文字セットには次の制約があります。
  - これらはクライアント文字セットとして使用できません。つまり、SET NAMES または SET CHARACTER SET では機能しません。(セクション10.1.4「接続文字セットおよび照合順序」を参照してください。)
  - 現在、LOAD DATA INFILE を使用して、これらの文字セットを使用するデータファイルをロードできません。
  - FULLTEXT インデックスは、これらのいずれかの文字セットを使用するカラムでは作成できません。ただし、インデックスのないカラムでは IN BOOLEAN MODE 検索を実行できます。
  - ベースとなるシステム呼び出しではゼロバイトで終了する文字列が要求されるため、これらの文字セットを含む ENCRYPT() の使用はお勧めしません。
- REGEXP および RLIKE 演算子はバイト単位で機能するため、マルチバイトセーフではなく、マルチバイト文字セットを使用すると想定外の結果が生成される可能性があります。さらに、これらの演算子ではそのバイト値に基づいて文字が比較されるため、アクセント記号付き文字は、指定された照合順序では等しいとみなされた場合でも、等しいとして比較されない可能性があります。

D.8 パフォーマンススキーマの制約

パフォーマンススキーマでは、データの収集または生成に相互排他ロックを使用できないので、一貫性は保証されず、適切な結果にならないことがあります。performance_schema テーブルのイベント値は、非決定的であり、反復不可です。

別のテーブルにイベント情報を保存した場合、元のイベントはあとから利用できません。たとえば、performance_schema テーブルから一時テーブルにイベントを選択し、あとからそのテーブルと元のテーブルを結合させる場合、一致するものがない可能性があります。

mysqldump と BACKUP DATABASE は、performance_schema データベース内のテーブルを無視します。

performance_schema データベース内のテーブルは、LOCK TABLES でロックできませんが、setup_xxx テーブルは除きます。

performance_schema データベース内のテーブルにインデックスを設定できません。

performance_schema データベース内のテーブルを参照するクエリーの結果は、クエリーキャッシュに保存されません。

performance_schema データベース内のテーブルは複製されません。

パフォーマンススキーマは、libmysqld 組み込みサーバーでは利用できません。

タイマーの種類は、プラットフォームごとに異なります。performance_timers テーブルは使用可能なイベントタイマーを示します。特定のタイマー名に対するこのテーブルでの値が NULL の場合、そのタイマーはプラットフォームでサポートされていません。

ストレージエンジンに適用されるインストゥルメントは、すべてのストレージエンジンに実装されていないことがあります。各サードパーティーエンジンのインストゥルメンテーションはエンジン管理者の責任です。

D.9 プラガブルな認証の制約

このセクションの最初の部分では、セクション6.3.7「プラガブル認証」で説明しているプラガブルな認証フレームワークの適用基準に関する一般的な制約について説明します。2 番目の部分では、サードパーティーコネクタ開発者が、コネクタがプラガブルな認証機能を利用できる範囲と、対応性を高めるために行うステップについて判断する方法について説明します。

ここで使用する「ネイティブ認証」という語は、mysql.user テーブルの Password カラムに格納されたパスワードに対する認証を指します。これは、プラガブルな認証が実装される前に古い MySQL Server で提供されていたものと同じ認証方法です。これが引き続きデフォルトの方法ですが、現在はプラグインを使用して実装されます。「Windows ネイティブ認証」とは、Windows ネイティブ認証プラグイン (「Windows プラグイン」と略します) で実装された、すでに Windows にログインしているユーザーの資格証明を使用した認証を示します。

一般的なプラガブルな認証の制約

- Connector/C、Connector/C++: これらのコネクタを使用するクライアントは、ネイティブ認証を使用するアカウントを通じてのみ、サーバーに接続できます。

  例外: コネクタは、libmysqlclient に (静的ではなく) 動的にリンクするように構築された場合にプラガブルな認証をサポートし、最新バージョンの libmysqlclient がインストールされている場合、またはコネクタが最新の libmysqlclient に対してリンクするようにソースから再コンパイルされている場合にそのバージョンをロードします。

- Connector/J: このコネクタを使用するクライアントは、ネイティブ認証を使用するアカウントを通じてのみサーバーに接続できます。

- Connector/Net: Connector/Net 6.4.4 より前では、このコネクタを使用するクライアントは、ネイティブ認証を使用するアカウントを通じてのみサーバーに接続できます。6.4.4 以降では、クライアントは、Windows プラグインを使用するアカウントを通じてサーバーに接続することもできます。

- Connector/ODBC: Connector/ODBC 3.51.29 および 5.1.9 より前では、このコネクタを使用するクライアントは、ネイティブ認証を使用するアカウントを通じてのみサーバーに接続できます。3.51.29 および 5.1.9 以降では、Windows 用のこのコネクタのバイナリリリースを使用するクライアントは、PAM または Windows プラグインを使用するアカウントを通じてサーバーに接続することもできます。(これらの機能は、以前に使用されていた MySQL 5.1 libmysqlclient ではなく MySQL 5.5.16 libmysqlclient に対して Connector/ODBC バイナリをリンクした結果得られます。新しい libmysqlclient には、サーバー側の PAM および Windows 認証プラグインに必要なクライアント側のサポートが含まれます。)

- Connector/PHP: このコネクタを使用するクライアントは、PHP 用の MySQL ネイティブドライバ (mysqlnd) を使用してコンパイルされている場合、ネイティブ認証を使用するアカウントを通じてのみサーバーに接続できます。

- MySQL Proxy: MySQL Proxy 0.8.2 より前では、クライアントは、ネイティブ認証を使用するアカウントを通じてのみサーバーに接続できます。0.8.2 以降では、クライアントは、PAM プラグインを使用するアカウントを通じてサーバーに接続することもできます。0.8.3 以降では、クライアントは、Windows プラグインを使用するアカウントを通じてサーバーに接続することもできます。

- MySQL Enterprise Backup: バージョン 3.6.1 より前の MySQL Enterprise Backup は、ネイティブ認証を使用するアカウントを通じてのみサーバーへの接続をサポートします。3.6.1 以降では、MySQL Enterprise Backup は、ネイティブ以外の認証を使用するアカウントを通じてサーバーに接続できます。

- Windows ネイティブ認証: Windows プラグインを使用するアカウントを通じた接続は、Windows Domain セットアップを必要とします。これがない場合、NTLM 認証が使用され、ローカル接続だけが可能になります。つまり、クライアントとサーバーを同じコンピュータ上で実行する必要があります。

- プロキシユーザー: プロキシユーザーサポートは、プロキシユーザー機能を実装するプラグイン (つまり、接続しているユーザーの名前と異なるユーザー名を返す場合があるプラグイン) で認証されたアカウントを通じて、クライアントが接続できる範囲まで利用できます。たとえば、ネイティブ認証プラグインは、プロキシユーザーをサポートしませんが、PAM および Windows プラグインはサポートします。

- レプリケーション: MySQL 5.6.4 より前では、レプリケーションスレーブは、ネイティブ認証を使用するマスターアカウントを通じてのみマスターサーバーに接続できます。5.6.4 以降では、レプリケーションスレーブは、必要なクライアント側のプラグインが利用できる場合、ネイティブ以外の認証を使用するマスターアカウントを通じて接続することもできます。プラグインは、libmysqlclient に組み込まれている場合、デフォルトで利用できます。それ以外の場合、プラグインは、スレーブ plugin_dir システム変数によって指名された、スレーブ側のディレクトリにインストールする必要があります。
- FEDERATED テーブル: FEDERATED テーブルは、ネイティブ認証を使用するリモートサーバー上のアカウントを通じてのみリモートテーブルにアクセスできます。

プラガブルな認証とサードパーティーコネクタ

サードパーティーコネクタ開発者は、次のガイドラインを使用して、プラガブルな認証機能を利用するためのコネクタの準備と、対応性を高めるために行うステップについて判断できます。

- 変更が行われていない既存のコネクタは、ネイティブ認証を使用し、このコネクタを使用するクライアントは、ネイティブ認証を使用するアカウントを通じてのみサーバーに接続できます。ただし、最新バージョンのサーバーに対してコネクタをテストして、このような接続が引き続き問題なく機能することを検証する必要があります。

  例外: コネクタは、(静的ではなく) 動的に libmysqlclient にリンクしている場合に、変更せずにプラガブルな認証を処理でき、最新バージョンの libmysqlclient がインストールされている場合に、このバージョンをロードします。

- プラガブルな認証機能を利用するには、libmysqlclient ベースのコネクタを、最新バージョンの libmysqlclient に対して再リンクする必要があります。これにより、コネクタは、現在 libmysqlclient に組み込まれているクライアント側のプラグイン (PAM 認証に必要な平文プラグインや Windows ネイティブ認証に必要な Windows プラグインなど) を必要とするアカウントを通じた接続をサポートできるようになります。現在の libmysqlclient とのリンクによっても、コネクタは、デフォルトの MySQL プラグインディレクトリ (通常、ローカルサーバーの plugin_dir システム変数のデフォルト値で指名されたディレクトリ) にインストールされたクライアント側にアクセスできるようになります。

  コネクタが動的に libmysqlclient にリンクする場合、より新しいバージョンの libmysqlclient がクライアントホストにインストールされていることと、コネクタが実行時にそれをロードすることを確認する必要があります。

- コネクタが特定の認証方式をサポートするには、直接クライアント/サーバープロトコルにその方式を実装します。Connector/Net はこのアプローチを使用して、Windows ネイティブ認証のサポートを提供します。
- コネクタが、デフォルトのプラグインディレクトリとは異なるディレクトリから、クライアント側のプラグインをロードできる必要がある場合、クライアントユーザーがそのディレクトリを指定するための手段を実装する必要があります。この候補としては、コネクタがディレクトリ名を取得できるコマンド行オプションまたは環境変数などがあります。mysql や mysqladmin などの標準 MySQL クライアントプログラムは、--plugin-dir オプションを実装します。セクション23.7.14「C API クライアントプラグイン関数」も参照してください。
- コネクタでのプロキシユーザーのサポートは、このセクションで前述したように、コネクタがサポートする認証方式がプロキシユーザーを許可するかどうかによって異なります。

D.10 MySQL での制限

このセクションでは、MySQL 5.6 での現在の制限を示します。

D.10.1 結合の制限

1 つの結合で参照できるテーブルの最大数は 61 です。これはビューの定義で参照できるテーブルの数にも適用されます。

D.10.2 データベースおよびテーブルの数に対する制限

MySQL にはデータベース数の制限はありません。ベースとなるファイルシステムによっては、ディレクトリ数に制限がある場合があります。

MySQL にはテーブル数の制限はありません。ベースとなるファイルシステムによっては、テーブルを表すファイル数に制限がある場合があります。個々のストレージエンジンには、エンジン固有の制約が課される場合があります。InnoDB では、最大 40 億個のテーブルを使用できます。

D.10.3 テーブルサイズの制限

MySQL データベースの事実上の最大テーブルサイズは、通常、MySQL の内部制限ではなくオペレーティングシステムのファイルサイズに関する制約によって判断します。次の表に、オペレーティングシステムのファイルサイズの制限に関するいくつかの例を示します。これは、大まかなガイドに過ぎず、確実なものではありません。もっとも新しい情報については、使用しているオペレーティングシステムに固有のドキュメントを必ず確認してください。

オペレーティングシステム	ファイルサイズの制限
Win32 (FAT/FAT32)	2G バイト/4G バイト
Win32 (NTFS)	2T バイト (これを超える可能性あり)
Linux 2.2-Intel 32 ビット	2G バイト (LFS: 4G バイト)
Linux 2.4+	(ext3 ファイルシステムを使用) 4T バイト
Solaris 9/10	16T バイト
OS X (HFS+)	2T バイト

Windows ユーザーの場合、FAT および VFAT (FAT32) は、MySQL で本番使用に適しているとは見なされません。代わりに NTFS を使用してください。

Linux 2.2 では、ext2 ファイルシステム用の Large File Support (LFS) パッチを使用すると、サイズが 2G バイトより大きな MyISAM テーブルを取得できます。ほとんどの最近の Linux 配布は、カーネル 2.4 以上に基づいており、必要な LFS パッチはすべて含まれています。Linux 2.4 では、大きなファイル (最大 2T バイト) をサポートするための ReiserFS 用のパッチも存在しています。JFS および XFS では、Linux でペタバイト以上のファイルが可能です。

Linux における LFS に関する詳細な概要については、http://www.suse.de/~aj/linux_lfs.html にある Andreas Jaeger 氏の「Large File Support in Linux」のページを参照してください。

フルテーブルエラーが発生した場合、その理由はいくつかあります。

- ディスクがいっぱいになっている可能性がある。
- InnoDB ストレージエンジンが、複数のファイルから作成できるテーブルスペース内に InnoDB テーブルを保持している。このため、テーブルが個々のファイルの最大サイズを超えることが可能です。テーブルスペースには、生のディスクパーティションを含めることができ、非常に大きなテーブルが許可されます。最大のテーブルスペースサイズは 64T バイトです。

  InnoDB テーブルを使用している場合、InnoDB テーブルスペース内のスペースが不足します。この場合、解決策は InnoDB テーブルスペースを拡張することです。セクション14.5.7「InnoDB ログファイルの数またはサイズの変更、および InnoDB テーブルスペースのサイズの変更」を参照してください。
- サポートするファイルサイズが最大 2G バイトのオペレーティングシステムで MyISAM テーブルを使用しており、データファイルまたはインデックスファイルでこの制限に達している。
- MyISAM テーブルを使用していて、テーブルに必要な領域が内部ポインタサイズによって許可されているサイズを超えている。MyISAM では、データファイルとインデックスファイルのサイズはデフォルトで最大 256T バイトですが、この制限は、65,536T バイト ($256^7 - 1$ バイト) の最大許容サイズまで変更できます。

  デフォルトの制限より大きな MyISAM テーブルが必要であり、オペレーティングシステムが大きなファイルをサポートしている場合は、CREATE TABLE ステートメントは AVG_ROW_LENGTH オプションと MAX_ROWS オプションをサポートします。セクション13.1.17「CREATE TABLE 構文」を参照してください。サーバーはこれらのオプションを使用して、許可するテーブルのサイズを決定します。

  ポインタサイズが既存のテーブルには小さすぎる場合、テーブルの最大許容サイズを増やすように ALTER TABLE でオプションを変更できます。セクション13.1.7「ALTER TABLE 構文」を参照してください。

```
ALTER TABLE tbl_name MAX_ROWS=1000000000 AVG_ROW_LENGTH=nnn;
```

  BLOB または TEXT カラムを含むテーブルにのみ、AVG_ROW_LENGTH を指定する必要があります。この場合、MySQL は、行数だけに基づいて必要な領域を最適化できません。

  MyISAM テーブルのデフォルトのサイズ制限を変更するには、内部行ポインタに使用されるバイト数を設定する myisam_data_pointer_size を設定します。MAX_ROWS オプションを指定しない場合、新しいテーブルのポ

インタサイズを設定するためにこの値が使用されます。myisam_data_pointer_size の値は 2 から 7 で指定できます。4 の値は 4G バイトまでのテーブルを許可し、6 の値は 256T バイトまでのテーブルを許可します。

次のステートメントを使用すると、データファイルおよびインデックスファイルの最大サイズを確認できます。

```
SHOW TABLE STATUS FROM db_name LIKE 'tbl_name';
```

myisamchk -dv /path/to/table-index-file も使用できます。セクション13.7.5「SHOW 構文」またはセクション4.6.3「myisamchk — MyISAM テーブルメンテナンスユーティリティー」を参照してください。

MyISAM テーブルのファイルサイズ制限に対処するその他の方法は次のとおりです。

- 大きなテーブルが読み取り専用である場合、myisampack を使用してこのテーブルを圧縮できます。myisampack は通常、少なくとも 50% 圧縮するので、実質上、さらに大きなテーブルを保有できます。myisampack で複数のテーブルを単一のテーブルにマージすることもできます。セクション4.6.5「myisampack — 圧縮された読み取り専用の MyISAM テーブルの生成」を参照してください。
- MySQL には、単一の MERGE テーブルと同一の構造を持つ MyISAM テーブルの集まりを処理できるようにする MERGE ライブラリが含まれます。セクション15.7「MERGE ストレージエンジン」を参照してください。

- MEMORY (HEAP) ストレージエンジンを使用している場合は、max_heap_table_size システム変数の値を増やす必要があります。セクション5.1.4「サーバーシステム変数」を参照してください。

D.10.4 テーブルカラム数と行サイズの制限

テーブルあたり 4096 カラムというハード制限がありますが、特定のテーブルでは有効な最大数がこれより少なくなる可能性があります。正確な制限は、相互作用する複数の要因によって異なります。

- すべてのテーブル (ストレージエンジンには無関係) の最大行サイズは 65,535 バイトです。ストレージエンジンではこの制限に対してさらなる制約を加えられる場合があり、有効な最大行サイズは少なくなります。

  すべてのカラムの合計長は最大行サイズを超えられないので、このサイズはカラム数 (およびサイズの可能性もあり) を制約します。たとえば、utf8 文字では、文字あたり最大 3 バイトが必要になるので、CHAR(255) CHARACTER SET utf8 カラムの場合、サーバーは値ごとに 255 × 3 = 765 バイトを割り当てる必要があります。この結果、テーブルにはこのようなカラムを 65,535 / 765 = 85 以上は含められません。

  可変長カラムのストレージには長さバイトが含まれ、これには行サイズに対して評価されます。たとえば、VARCHAR(255) CHARACTER SET utf8 カラムは、値の長さを格納するために 2 バイトを使用するので、それぞれの値は最大 767 バイトを使用できます。

  BLOB カラムと TEXT カラムは、その内容が行の残りとは別に格納されるので、行サイズに対してそれぞれ 9 から 12 (1から4+8) バイトになります。

  カラムを NULL と宣言すると、使用できるカラムの最大数を減らすことができます。MyISAM テーブルの場合、NULL カラムは、値が NULL であるかどうかを記録するための追加領域を行内に必要とします。各 NULL カラムは 1 ビット余分に占め、もっとも近いバイトまで丸められます。バイトでの最大の行長は次のように計算できます。

```
row length = 1
           + (sum of column lengths)
           + (number of NULL columns + delete_flag + 7)/8
           + (number of variable-length columns)
```

  delete_flag は静的行フォーマットのテーブルに対しては 1 です。静的テーブルは、行が削除されているかどうかを示すフラグとして、行レコード内の 1 ビットを使用します。このフラグは動的行ヘッダーに格納されるので、動的テーブルの場合、delete_flag は 0 です。MyISAM テーブル形式の詳細は、セクション15.2.3「MyISAM テーブルのストレージフォーマット」を参照してください。

  InnoDB テーブルの場合、ストレージサイズは、NULL カラムおよび NOT NULL カラムの場合と同じなので、前述の計算は適用されません。

  カラムが必要とするサイズが 32,765 + 2 バイトと 32,766 + 2 バイトであり、65,535 バイトの最大行サイズ以内に収まるので、テーブル t1 を作成する次のステートメントは成功します。

```
mysql> CREATE TABLE t1
  -> (c1 VARCHAR(32765) NOT NULL, c2 VARCHAR(32766) NOT NULL)
  -> ENGINE = MyISAM CHARACTER SET latin1;
```

```
Query OK, 0 rows affected (0.02 sec)
```

カラムは NULL であり、MyISAM が必要とする追加領域によって行サイズが 65,535 バイトを超えるので、テーブル t2 を作成する次のステートメントは失敗します。

```
mysql> CREATE TABLE t2
    -> (c1 VARCHAR(32765) NULL, c2 VARCHAR(32766) NULL)
    -> ENGINE = MyISAM CHARACTER SET latin1;
ERROR 1118 (42000): Row size too large. The maximum row size for the
used table type, not counting BLOBs, is 65535. You have to change some
columns to TEXT or BLOBs
```

カラム長は 65,535 バイトの最大長内に収まっていますが、長さを記録するために 2 つの追加バイトが必要になり、このため行サイズが 65,535 バイトを超えるので、テーブル t3 を作成する次のステートメントは失敗します。

```
mysql> CREATE TABLE t3
    -> (c1 VARCHAR(65535) NOT NULL)
    -> ENGINE = MyISAM CHARACTER SET latin1;
ERROR 1118 (42000): Row size too large. The maximum row size for the
used table type, not counting BLOBs, is 65535. You have to change some
columns to TEXT or BLOBs
```

カラム長を 65,533 以下に減らすと、ステートメントは成功します。

- 個々のストレージエンジンで、テーブルカラム数を制限するその他の制約が適用される場合があります。例:
  - InnoDB が許可するカラム数は最大 1000 です。
  - InnoDB は、データベースページ (約 8,000 バイト) の半分以下に行サイズを制限し、VARBINARY、VARCHAR、BLOB、または TEXT カラムを含みません。
  - 異なる InnoDB ストレージフォーマット (COMPRESSED、REDUNDANT) では、使用するページヘッダーおよびトレーラデータの量が違うため、行に使用できるストレージ量に影響がおよびます。
- それぞれのテーブルにはテーブル定義を含む .frm ファイルがあります。この定義は、テーブルで許可されるカラム数に影響する方法でこのファイルの内容に影響を与えます。詳細は、セクションD.10.5「.frm ファイル構造により課せられる制限」を参照してください。

D.10.5 .frm ファイル構造により課せられる制限

それぞれのテーブルにはテーブル定義を含む .frm ファイルがあります。サーバーは次の式を使用して、64K バイトの上限に対して、ファイルに格納されたいくつかのテーブル情報をチェックします。

```
if (info_length+(ulong) create_fields.elements*FCOMP+288+
   n_length+int_length+com_length > 65535L || int_count > 255)
```

.frm ファイルに格納された情報のうち式に対してチェックされる部分は 64K バイトの制限を超えることはできません。したがって、テーブル定義がこのサイズに達した場合、これ以上カラムを追加できません。

式の関連因子は次のとおりです。

- info_length は、「画面」に必要な領域です。これは MySQL の Unireg から継承したものです。
- create_fields.elements はカラム数です。
- FCOMP は 17 です。
- n_length は、名前あたり 1 バイトの区切り文字を含む、すべてのカラム名の合計長です。
- int_length は、ENUM および SET カラムの値のリストに関連します。このコンテキストでは、「int」は「整数」の意味ではありません。これは「間隔」を意味し、ENUM および SET カラムの総称です。
- int_count は、一意の ENUM および SET 定義の数です。
- com_length はカラムコメントの合計長です。

前述の式には、許可されるテーブル定義に対し複数の意味があります。

- 長いカラム名を使用すると、ENUM または SET カラムを含めたり、カラムコメントを使用したりした場合と同様に、カラムの最大数が減ることがあります。

- テーブルに保持できる一意の ENUM および SET 定義は 255 以下です。同一要素リストを持つカラムは、この制限に対して同じと見なされます。たとえば、次の 2 つのカラムがテーブルに含まれる場合、これらは定義が同一なので、この制限に対して (2 つではなく) 1 つと見なされます。

```
e1 ENUM('a','b','c')
e2 ENUM('a','b','c')
```

- 一意の ENUM および SET 定義内の要素名の合計長は、64K バイトの制限に対して加えられるので、特定の ENUM カラムの要素の数に対する理論的な制限は 65,535 ですが、実際の限度は 3000 未満です。

D.10.6 Windows プラットフォームの制限

次の制限が、Windows プラットフォームで MySQL を使用する場合に適用されます。

- プロセスメモリー

  Windows 32 ビットプラットフォームでは、デフォルトで、MySQL などの単一プロセス内で 2G バイトを超える RAM を使用できません。これは、Windows 32 ビットでの物理アドレスの制限が 4G バイトであり、Windows 内のデフォルト設定では、カーネル (2G バイト) とユーザー/アプリケーション (2G バイト) とに仮想アドレス空間を分割するためです。

  Windows の一部のバージョンには、カーネルアプリケーションを減らすことによってより大きなアプリケーションに対応するブート時設定があります。または、2G バイト以上を使用するには、64 ビットバージョンの Windows を使用します。

- ファイルシステムエイリアス

  MyISAM テーブルの使用時には、Windows でエイリアスを使用して、別のボリューム上のデータファイルにリンクしててからメインの MySQL datadir の場所に戻るようにはリンクできません。

  この機能は、多くの場合、datadir オプションで構成されたデフォルトのデータディレクトリにメインの .frm ファイルを保持しながら、データファイルおよびインデックスファイルを RAID またはその他の高速ソリューションに移動させるために使用されます。

- ポート数の制限

  Windows システムにはクライアント接続のポートがおよそ 4,000 あり、1 つのポート接続が閉じるとそのポートを再度利用できるようになるまで 2 から 4 分かかります。クライアントがサーバーとの接続と切断を高い頻度で繰り返す環境では、閉じたポートが再度利用できるようになる前に、利用できるポートがすべて使用されてしまうことがあります。このようになると、MySQL Server は動作中であっても反応していないように見えます。ポートはマシンで実行されているほかのアプリケーションでも使用されている場合があり、このときには、MySQL に利用できるポート数は少なくなります。

  この問題の詳細は、http://support.microsoft.com/default.aspx?scid=kb;en-us;196271 を参照してください。

- DATA DIRECTORY および INDEX DIRECTORY

  CREATE TABLE の DATA DIRECTORY オプションは、セクション14.5.4「テーブルスペースの位置の指定」で説明するように、Windows では InnoDB テーブルに対してのみサポートされます。MyISAM およびその他のストレージエンジンの場合、CREATE TABLE の DATA DIRECTORY および INDEX DIRECTORY オプションは、Windows と、非機能的 realpath() 呼び出しを使用するほかのプラットフォームでは無視されます。

- DROP DATABASE

  別のセッションで使用されているデータベースは削除できません。

- 大文字と小文字を区別しない名前

  Windows ではファイル名の大文字と小文字は区別されないため、MySQL のデータベース名とテーブル名も Windows では大文字と小文字は区別されません。唯一の制約は、特定のステートメント全体で大文字と小文字を変更せずに、データベース名とテーブル名を指定する必要があるということだけです。セクション9.2.2「識別子の大文字と小文字の区別」を参照してください。

- ディレクトリ名とファイル名

  Windows では、MySQL Server は現行の ANSI コードページと互換性のあるディレクトリ名とファイル名のみをサポートします。たとえば、次の日本語のディレクトリ名は欧米の口ケール (コードページ 1252) では機能しません。

```
datadir="C:/私たちのプロジェクトのデータ"
```

同じ制限は、LOAD DATA INFILE のデータファイルパス名など、SQL ステートメントで参照されるディレクトリ名とファイル名にも適用されます。

- 「\」 (パス名の区切り文字)

  Windows でのパス名のコンポーネントは、「\」文字で区切られますが、これは MySQL のエスケープ文字でもあります。LOAD DATA INFILE または SELECT ... INTO OUTFILE を使用している場合は、Unix スタイルのファイル名と「/」文字を一緒に使用します。

```
mysql> LOAD DATA INFILE 'C:/tmp/skr.txt' INTO TABLE skr;
mysql> SELECT * INTO OUTFILE 'C:/tmp/skr.txt' FROM skr;
```

  または、「\」文字を 2 重にする必要があります。

```
mysql> LOAD DATA INFILE 'C:\\tmp\\skr.txt' INTO TABLE skr;
mysql> SELECT * INTO OUTFILE 'C:\\tmp\\skr.txt' FROM skr;
```

- パイプに関する問題

  パイプは Windows のコマンド行プロンプトからでは確実には機能しません。パイプに ^Z / CHAR(24) が含まれている場合、Windows はファイルの最後だと勘違いしてプログラムを中止します。

  これは主に、次のようにバイナリログを適用するときに問題になります。

```
C:\> mysqlbinlog binary_log_file | mysql --user=root
```

  ログを適用するときに問題が発生し、その原因が ^Z / CHAR(24) 文字によるものだと考えられる場合は、次の回避法を使用できます。

```
C:\> mysqlbinlog binary_log_file --result-file=/tmp/bin.sql
C:\> mysql --user=root --execute "source /tmp/bin.sql"
```

  後者のコマンドは、バイナリデータを含む SQL ファイルを確実に読み取るために使用することもできます。

MySQL 用語集

これらの用語は、MySQL データベースサーバーに関する情報で一般的に使用されます。この用語集は、InnoDB ストレージエンジンに関する用語のリファレンスとして作成され、大部分の定義は InnoDB 関連です。

.

.ARM ファイル
: ARCHIVE テーブルのメタデータ。.ARZ ファイルと対比してください。この拡張子を持つファイルは常に、MySQL Enterprise Backup 製品の mysqlbackup コマンドで生成されるバックアップに含められます。
.ARZ ファイル, MySQL Enterprise Backup, mysqlbackup コマンドも参照

.ARZ ファイル
: ARCHIVE テーブルのデータ。.ARM ファイルと対比してください。この拡張子を持つファイルは常に、MySQL Enterprise Backup 製品の mysqlbackup コマンドで生成されるバックアップに含められます。
.ARM ファイル, MySQL Enterprise Backup, mysqlbackup コマンドも参照

.cfg ファイル
: InnoDB トランスポータブルテーブルスペース機能で使用するメタデータファイル。これは、コマンド FLUSH TABLES ... FOR EXPORT で生成され、1 つまたは複数のテーブルを、別のサーバーにコピーできる一貫した状態にします。.cfg ファイルは、対応する .ibd ファイルとともにコピーされ、ALTER TABLE ... IMPORT TABLESPACE ステップ中に space ID などの .ibd ファイルの内部値を調整するために使用されます。
.ibd ファイル, スペース ID, トランスポータブルテーブルスペースも参照

.frm ファイル
: MySQL テーブルのメタデータ (テーブル定義など) を含むファイル。

  バックアップの場合、バックアップ後に変更または削除されたテーブルをリストアできるように、バックアップデータとともに .frm ファイルの完全セットを常に保持する必要があります。

  それぞれの InnoDB テーブルには .frm ファイルがありますが、InnoDB は独自のテーブルメタデータをシステムテーブルスペースに保持しています。InnoDB が InnoDB テーブルを処理する場合、.frm ファイルは不要です。

  これらのファイルは、MySQL Enterprise Backup 製品によってバックアップされます。バックアップが行われている間にこれらのファイルを ALTER TABLE 操作で変更してはいけないため、InnoDB でないテーブルを含むバックアップは .frm ファイルのバックアップ中に、FLUSH TABLES WITH READ LOCK 操作を実行してこのようなアクティビティーをフリーズします。バックアップをリストアするときに、バックアップ時点のデータベースの状態に一致するように、.frm ファイルが作成、変更、または削除される場合があります。
  MySQL Enterprise Backupも参照

.ibd ファイル
: file-per-table モードを使用して作成された各 InnoDB テーブルは、データベースディレクトリ内で、.ibd 拡張子の専用のテーブルスペースファイルに書き込まれます。このファイルにはテーブルデータと、テーブルのインデックスが含まれます。innodb_file_per_table オプションで制御される file-per-table モードは、InnoDB ストレージの使用法およびパフォーマンスの多くの側面に影響し、MySQL 5.6.7 以降でデフォルトで有効になっています。

  この拡張子は、ibdata ファイルから構成されるシステムテーブルスペースには適用されません。

  MySQL Enterprise Backup 製品によって .ibd ファイルが圧縮バックアップに含まれるとき、圧縮版は .ibz ファイルです。

  MySQL 5.6 以降で DATA DIRECTORY = 句を使用してテーブルが作成された場合、.ibd ファイルは、通常のデータベースディレクトリの外部に置かれ、.isl ファイルによってポイントされます。
  データベース, file-per-table, ibdata ファイル, .ibz ファイル, インデックス, innodb_file_per_table, .isl ファイル, MySQL Enterprise Backup, システムテーブルスペース, テーブル, テーブルスペースも参照

.ibz ファイル
: MySQL Enterprise Backup 製品が圧縮バックアップを実行するときに、これは、file-per-table 設定を使用して作成される各テーブルスペースファイルを、.ibd 拡張子から .ibz 拡張子に変換します。

  バックアップ中に適用される圧縮は、通常操作中にテーブルデータを圧縮されたままにする圧縮行フォーマットとは異なります。圧縮バックアップ操作では、すでに圧縮行フォーマットであるテーブルスペースについては圧縮ステップをスキップします。2 回目の圧縮では、バックアップ速度を低下させるだけで、ほとんどまたはまったく領域を節約しないためです。

圧縮バックアップ, 圧縮行フォーマット, file-per-table, .ibd ファイル, MySQL Enterprise Backup, テーブルスペースも参照

.isl ファイル
MySQL 5.6 以降の DATA DIRECTORY = 句で作成された InnoDB テーブルの、.ibd ファイルの場所を指定するファイル。これは、実際のシンボリックリンクメカニズムのプラットフォーム制限なしで、シンボリックリンクのように機能します。データベースディレクトリ外部 (たとえば、テーブルの使用状況に応じて特別に大きなまたは高速なストレージデバイス) に InnoDB テーブルスペースを格納できます。詳細は、セクション14.5.4「テーブルスペースの位置の指定」を参照してください。
データベース, .ibd ファイル, テーブル, テーブルスペースも参照

.MRG ファイル
MERGE ストレージエンジンで使用される、ほかのテーブルへの参照を含むファイル。この拡張子を持つファイルは常に、MySQL Enterprise Backup 製品の mysqlbackup コマンドで生成されるバックアップに含められます。
MySQL Enterprise Backup, mysqlbackup コマンドも参照

.MYD ファイル
MyISAM テーブルのデータを格納するために MySQL が使用するファイル。
.MYI ファイル, MySQL Enterprise Backup, mysqlbackup コマンドも参照

.MYI ファイル
MyISAM テーブルのインデックスを格納するために MySQL が使用するファイル。
.MYD ファイル, MySQL Enterprise Backup, mysqlbackup コマンドも参照

.OPT ファイル
データベース構成情報を含むファイル。この拡張子を持つファイルは常に、MySQL Enterprise Backup 製品の mysqlbackup コマンドで生成されるバックアップに含められます。
MySQL Enterprise Backup, mysqlbackup コマンドも参照

.PAR ファイル
パーティション定義を含むファイル。この拡張子を持つファイルは常に、MySQL Enterprise Backup 製品の mysqlbackup コマンドで生成されるバックアップに含められます。
MySQL Enterprise Backup, mysqlbackup コマンドも参照

.TRG ファイル
トリガーパラメータを含むファイル。この拡張子を持つファイルは常に、MySQL Enterprise Backup 製品の mysqlbackup コマンドで生成されるバックアップに含められます。
MySQL Enterprise Backup, mysqlbackup コマンド, .TRN ファイルも参照

.TRN ファイル
トリガー名前空間情報を含むファイル。この拡張子を持つファイルは常に、MySQL Enterprise Backup 製品の mysqlbackup コマンドで生成されるバックアップに含められます。
MySQL Enterprise Backup, mysqlbackup コマンド, .TRG ファイルも参照

2

2 フェーズコミット
XA 仕様に基づく分散型トランザクションの一部である操作。(2PC と略記されることがあります。)複数のデータベースがトランザクションに参加する場合、すべてのデータベースが変更をコミットするか、すべてのデータベースが変更をロールバックします。
コミット, ロールバック, トランザクション, XAも参照

A

ACID
原子性 (atomicity)、一貫性 (consistency)、分離性 (isolation)、持続性 (durability) を表す頭字語。これらの特性はすべてデータベースシステムで望ましく、すべてトランザクションの概念に密接に結び付けられています。InnoDB のトランザクション機能は、ACID の原則に準拠しています。

トランザクションは、コミットまたはロールバックできる原子的な作業単位です。トランザクションによってデータベースに複数の変更が行われた場合、トランザクションがコミットされるとすべての変更が完了し、トランザクションがロールバックされるとすべての変更が元に戻されます。

データベースは、それぞれのコミットまたはロールバックのあとでも、トランザクションの進行中でも、常に一貫した状態を保ちます。関連データが複数のテーブルにわたって更新されている場合、クエリーは、古い値と新しい値の混合ではなく、すべて古い値か、すべて新しい値のどちらかを見ます。

トランザクションは進行中、互いから保護 (分離) されます。それらは互いに干渉できず、互いのコミットされていないデータを見ることはできません。この分離性は、ロックメカニズムを通じて実現します。経験豊富なユーザーは、実際にトランザクションが互いに干渉しないと確信できれば、パフォーマンスと並列性の向上の代わりに保護の低下をトレードオフするように分離レベルを調整できます。

トランザクションの結果は持続的です。コミット操作が成功すると、そのトランザクションによって行われた変更は、データベース以外の多くのアプリケーションが脆弱である停電、システムのクラッシュ、競合状況、またはほかの潜在的な危険から保護されます。持続性には通常、ディスクストレージへの書き込みがかかわっており、書き込み操作中の停電またはソフトウェアクラッシュに対して保護するために一定の冗長性を備えています。(InnoDB では、二重書き込みバッファーが持続性をサポートします。)
原子的, コミット, 並列性, 二重書き込みバッファー, 分離レベル, ロック, ロールバック, トランザクションも参照

AHI
適応型ハッシュインデックス (Adaptive Hash Index) の頭字語。
適応型ハッシュインデックスも参照

AIO
非同期 I/O (Asynchronous I/O) の頭字語。この頭字語は InnoDB メッセージやキーワードで見られます。
非同期 I/Oも参照

Antelope
元の InnoDB ファイル形式のコード名。これは、冗長および簡易行フォーマットをサポートしますが、Barracuda ファイル形式で利用できるより新しい動的および圧縮行フォーマットはサポートしません。

アプリケーションが、InnoDB テーブル圧縮からメリットを受けられるか、動的行フォーマットからメリットを受けられる BLOB またはラージテキストカラムを使用する場合、一部のテーブルを Barracuda 形式に切り替えることができます。テーブルの作成前に innodb_file_format オプションを設定することによって、使用するファイル形式を選択します。
Barracuda, コンパクト行フォーマット, 圧縮行フォーマット, 動的行フォーマット, ファイル形式, innodb_file_format, 冗長行フォーマットも参照

B

B ツリー
データベースインデックスに一般的に使用されるツリーデータ構造。この構造は、常にソートされ続け、正確な一致 (等号演算子) および範囲 (大なり、小なり、BETWEEN 演算子など) の高速ルックアップを可能にします。このタイプのインデックスは、InnoDB や MyISAM などのほとんどのストレージエンジンで利用できます。

B ツリーノードには多くの子を含むことができるので、B ツリーは、ノードごとに 2 つの子に限られているバイナリツリーと同じではありません。

ハッシュインデックスと対比してください。こちらは MEMORY ストレージエンジンでのみ使用できます。MEMORY ストレージエンジンは、B ツリーインデックスも使用でき、一部のクエリーで範囲演算子を使用する場合は、MEMORY テーブルには B ツリーインデックスを選択してください。
ハッシュインデックスも参照

Barracuda
テーブルデータの圧縮をサポートする InnoDB ファイル形式のコード名。このファイル形式は、最初に InnoDB Plugin に導入されました。これは、InnoDB テーブル圧縮に対応した圧縮行フォーマットと、BLOB およびラージテキストカラムのストレージレイアウトを改善する動的行フォーマットをサポートします。これは innodb_file_format オプションで選択できます。

InnoDB システムテーブルスペースは元の Antelope ファイル形式で格納されるため、Barracuda ファイル形式を使用するには、システムテーブルスペースとは別の独自のテーブルスペースに新しく作成したテーブルを格納する file-per-table 設定も有効にする必要があります。

MySQL Enterprise Backup 製品バージョン 3.5 以降では、Barracuda ファイル形式を使用するテーブルスペースのバックアップをサポートします。
Antelope, コンパクト行フォーマット, 圧縮行フォーマット, 動的行フォーマット, ファイル形式, file-per-table, innodb_file_format, MySQL Enterprise Backup, 行フォーマット, システムテーブルスペースも参照

binlog
バイナリログファイルの非公式名。たとえば、電子メールメッセージやフォーラムディスカッションでこの略語を見ることがあります。
バイナリログも参照

C

CPU バウンド
: ワークロードの種類の 1つ。主なボトルネックがメモリー内の CPU 操作であるもの。通常、バッファープール内にすべての結果をキャッシュできる、読み取り中心の操作を含みます。
ボトルネック, バッファープール, CPU バウンド, ワークロードも参照

CRUD
: データベースアプリケーションの一般的な操作シーケンスである「作成 (create)、読み取り (read)、更新 (update)、削除 (delete)」の頭字語。多くの場合、どの言語でもすばやく実装でき、比較的単純にデータベースを使用するタイプのアプリケーション (基本的な DDL、DML、および SQL のクエリーステートメント) を示します。
DDL, DML, クエリー, SQLも参照

D

DCL
: データ制御言語 (Data Control Language)。権限を管理するための SQL ステートメントのセット。MySQL では、GRANT および REVOKE ステートメントから構成されます。DDL および DML と対比してください。
DDL, DML, SQLも参照

DDL
: データ定義言語 (Data Definition Language)。個々のテーブル行ではなくデータベース自体を操作するための SQL ステートメントのセット。CREATE、ALTER、および DROP ステートメントのすべての形式を含みます。TRUNCATE ステートメントも含まれます。DELETE FROM table_name ステートメントと動作が異なるためです (最終的な効果は似ていますが) 。

DDL ステートメントは自動的に現在のトランザクションをコミットします。それらをロールバックすることはできません。

InnoDB のオンライン DDL 機能は、CREATE INDEX、DROP INDEX、および多くのタイプの ALTER TABLE 操作のパフォーマンスを向上させます。詳細は、セクション14.11「InnoDB とオンライン DDL」を参照してください。InnoDB の file-per-table 設定も、DROP TABLE および TRUNCATE TABLE 操作の動作に影響を与える場合があります。

DML および DCL と対比してください。
コミット, DCL, DML, file-per-table, ロールバック, SQL, トランザクションも参照

DML
: データ操作言語 (Data Manipulation Language)。挿入、更新、および削除操作を実行するための SQL ステートメントのセット。SELECT ステートメントが DML ステートメントと見なされる場合があります。SELECT ... FOR UPDATE 形式が、INSERT、UPDATE、および DELETE と同じ、ロックに関する考慮事項に従うためです。

InnoDB テーブルの DML ステートメントはトランザクションのコンテキストで動作するため、その効果は単一の単位としてコミットまたはロールバックできます。

DDL および DCL と対比してください。
コミット, DCL, DDL, ロック, ロールバック, SQL, トランザクションも参照

F

file-per-table
: innodb_file_per_table オプションで制御される設定に対する一般名。これは、InnoDB ファイルストレージの多くの側面、機能の利用可能性、および I/O 特性に影響する、非常に重要な構成オプションです。MySQL 5.6.7 以降ではデフォルトで有効になっています。MySQL 5.6.7 より前では、デフォルトで無効になっています。

この設定が有効である間に作成されたテーブルごとに、データは、システムテーブルスペースの ibdata ファイルではなく、個別の .ibd ファイルに格納されます。テーブルデータが個々のファイルに格納されている場合、データ圧縮などの機能に必要な、デフォルト以外のファイル形式と行フォーマットを選択できる柔軟性が得られます。TRUNCATE TABLE 操作も高速化され、解放された領域は、InnoDB に予約されたままではなく、オペレーティングシステムで使用できます。

MySQL Enterprise Backup 製品は、テーブルがその独自のファイルに格納されるので柔軟性が高くなります。たとえば、テーブルをバックアップから排除できますが、これは個別のファイルに格納されている場合に限られます。したがって、この設定は、あまり頻繁にはバックアップされない、または異なるスケジュールでバックアップされるテーブルに適しています。

圧縮行フォーマット, 圧縮, ファイル形式, .ibd ファイル, ibdata ファイル, innodb_file_per_table, 行フォーマット, システムテーブルスペースも参照

FOREIGN KEY 制約
外部キー関係を通じてデータベース一貫性を維持するタイプの制約。ほかの種類の制約と同様に、データが一貫性を失った場合にデータが挿入または更新されるのを防止できます。ここでは、複数のテーブル内のデータ間の一貫性が失われることが防止されます。または、DML 操作が実行されるときに FOREIGN KEY 制約によって、外部キー作成時に指定された ON CASCADE オプションに基づいて、子行内のデータが削除されたり、別の値に変更されたり、NULL に設定されたりします。
子テーブル, 制約, DML, 外部キー, NULLも参照

FTS
ほとんどのコンテキストで、全文検索 (Full-Text Search) の頭字語。パフォーマンスディスカッションでは、フルテーブルスキャン (Full Table Scan) の頭字語の場合があります。
テーブルの完全スキャン, 全文検索も参照

FULLTEXT インデックス
MySQL 全文検索メカニズムでの検索インデックスを保持する、特殊なインデックス。ストップワードとして指定されたものを飛ばして、カラムの値からの単語を表します。もともとは、利用できるのは MyISAM テーブルだけでした。MySQL 5.6.4 以降では、InnoDB テーブルでも利用できます。
全文検索, インデックス, InnoDB, 検索インデックス, ストップワードも参照

G

GA
「一般提供 (Generally Available)」。ソフトウェア製品がベータを終え、販売、公式サポート、および本番使用に利用できるようになった段階。
ベータ, アーリーアダプタも参照

global_transaction
XA 操作に含まれるタイプのトランザクション。これは、それ自体はトランザクションであるけれども、すべてがグループとして正しく完了する必要があるか、グループとしてロールバックされる必要がある、いくつかのアクションから構成されます。基本的に、これは ACID 特性を 1 レベル上に拡張することにより、複数の ACID トランザクションを、同じく ACID 特性を持つグローバル操作のコンポーネントとして連携して実行できるようにします。この種の分散トランザクションの場合、SERIALIZABLE 分離レベルを使用して ACID 特性を実現する必要があります。
ACID, SERIALIZABLE, トランザクション, XAも参照

H

HDD
「ハードディスクドライブ (Hard Disk Drive)」の頭字語。SSD と対比されることが多く、スピニングプラッターを使用するストレージメディアを指します。そのパフォーマンス特性はディスクベースワークロードのスループットに影響を与えることがあります。
ディスクベース, SSDも参照

I

I/O バウンド
ディスクバウンドも参照

ib-file セット
MySQL データベース内で InnoDB によって管理されるファイルのセット (システムテーブルスペース、file-per-table テーブルスペース、(通常は 2 つの) Redo ログファイル)。InnoDB ファイル構造および形式の詳細なディスカッションで、さまざまな DBMS 製品間でのデータベースの意味と、MySQL データベースに含まれている可能性がある非 InnoDB ファイルとの間のあいまいさを避けるために使用されることがあります。
データベース, file-per-table, Redo ログ, システムテーブルスペースも参照

ibbackup_logfile
ホットバックアップ操作中に MySQL Enterprise Backup 製品により作成される補助的なバックアップファイル。ここには、バックアップの実行中に行われたデータ変更に関する情報が含まれます。ibbackup_logfile などの初期バックアップファイルは、バックアップ操作中に行われた変更がまだ組み込まれていないので、raw バックアップと呼ばれます。raw バックアップファイルへの適用ステップを実行したあと、結果として得られるファイルは、最終データ変更を含んでおり、準備されたバックアップと呼ばれます。この段階で、ibbackup_logfile ファイルは必要なくなります。
適用, ホットバックアップ, MySQL Enterprise Backup, 準備されたバックアップ, raw バックアップも参照

ibdata ファイル
InnoDB システムテーブルスペースを構成する、ibdata1 や ibdata2 などの名前を持つファイルのセット。これらのファイルには、InnoDB テーブルに関するメタデータ (データディクショナリ) と、Undo ログ、変更バッファー、二重書き込みバッファーのストレージ領域が含まれます。(各テーブルの作成時に file-per-table モードが有効であるかどうかによって) テーブルデータの一部またはすべてを含めることもできます。innodb_file_per_table オプションが有効であるときに、新しく作成されたテーブルのデータおよびインデックスは、システムテーブルスペースではなく個別の .ibd ファイルに格納されます。

ibdata ファイルが増大するときは、innodb_autoextend_increment 構成オプションの影響を受けます。
変更バッファー, データディクショナリ, 二重書き込みバッファー, file-per-table, .ibd ファイル, innodb_file_per_table, システムテーブルスペース, Undo ログも参照

ibtmp ファイル
非圧縮 InnoDB 一時テーブルと関連オブジェクト用の InnoDB 一時テーブルスペースデータファイル。構成ファイルオプション innodb_temp_data_file_path は、ユーザーが一時データファイルの相対パスを定義することを許可します。innodb_temp_data_file_path が指定されない場合のデフォルト動作は、データディレクトリ内に ibdata1 と一緒に、ibtmp1 という名前の単一自動拡張 12M バイトデータファイルを作成することです。
一時テーブルスペースも参照

ib_logfile
通常は ib_logfile0 および ib_logfile1 という名前が付けられ、Redo ログを形成するファイルのセット。ロググループと呼ばれることもあります。これらのファイルは、InnoDB テーブル内のデータを変更しようとするステートメントを記録します。これらのステートメントは、クラッシュ後の起動時に、未完了のトランザクションで書き込まれたデータを修正するために自動的に再現されます。

このデータは手動リカバリには使用できません。このタイプの操作には、バイナリログを使用してください。
バイナリログ, ロググループ, Redo ログも参照

ilist
InnoDB FULLTEXT インデックス内で、ドキュメント ID とトークン (つまり特定の語) の位置情報から構成されるデータ構造。
FULLTEXT インデックスも参照

INFORMATION_SCHEMA
MySQL データディクショナリへのクエリーインタフェースを提供するデータベースの名前。(この名前は ANSI SQL 標準で定義されています。)データベースに関する情報 (メタデータ) を調べるには、構造化されていない出力を返す SHOW コマンドを使用する代わりに、INFORMATION_SCHEMA.TABLES や INFORMATION_SCHEMA.COLUMNS などのテーブルを照会できます。

情報スキーマには、INNODB_LOCKS や INNODB_TRX など、InnoDB に固有のいくつかのテーブルが含まれます。これらのテーブルは、データベースがどのように構造化されているかを確認するためではなく、InnoDB テーブルの動作に関するリアルタイム情報を得るために使用してください (パフォーマンスモニタリング、チューニング、トラブルシューティングに役立ちます)。これらのテーブルは特に、圧縮、トランザクション、およびそれらに関連付けられたロックに関連する MySQL 機能についての情報を提供します。
圧縮, データディクショナリ, データベース, InnoDB, ロック, トランザクションも参照

InnoDB
高いパフォーマンスと、信頼性、堅牢性、および同時アクセスのためのトランザクション機能とを結合する MySQL コンポーネント。これは ACID 設計概念を具体化したものです。ストレージエンジンとして表現され、ENGINE=INNODB 句で作成または変更されたテーブルを処理します。アーキテクチャーの詳細および管理手順については第14章「InnoDB ストレージエンジン」、パフォーマンスのアドバイスについてはセクション8.5「InnoDB テーブルの最適化」を参照してください。

MySQL 5.5 以降では、InnoDB が新しいテーブルのデフォルトストレージエンジンなので、ENGINE=INNODB 句は必要ありません。MySQL 5.1 でのみ、高度な InnoDB 機能の多くで InnoDB Plugin と呼ばれるコンポーネントを有効にする必要があります。InnoDB テーブルがデフォルトである最近のリリースへの移行に関する考慮事項については、セクション14.1.1「デフォルトの MySQL ストレージエンジンとしての InnoDB」を参照してください。

InnoDB テーブルは理論的には、ホットバックアップに適しています。通常の処理を妨げることなく MySQL Server をバックアップできる MySQL Enterprise Backup 製品の詳細は、セクション25.2「MySQL Enterprise Backup」を参照してください。
ACID, ホットバックアップ, ストレージエンジン, トランザクションも参照

innodb_autoinc_lock_mode
innodb_autoinc_lock_mode オプションは、自動インクリメントロックに使用されるアルゴリズムを制御します。自動インクリメントする主キーがある場合は、innodb_autoinc_lock_mode=1 設定でのみステートメントベースレプリケーションを使用できます。この設定は、トランザクション内の複数行挿入が連続自動インクリメント値を受け取るため、連

続ロックモードと呼ばれます。innodb_autoinc_lock_mode=2 の場合は (挿入操作で並列性が向上)、ステートメントベースレプリケーションではなく行ベースレプリケーションを使用してください。この設定は、同時に実行される複数の複数行挿入ステートメントが自動インクリメント値を交互に受け取ることができるため、インターリーブロックモードと呼ばれます。innodb_autoinc_lock_mode=0 の設定は、以前 (従来) のデフォルト設定であり、互換性の目的以外では使用しないでください。
自動インクリメントロック, 混在モード挿入, 主キーも参照

innodb_file_format
innodb_file_format オプションは、このオプションの値を指定したあとに作成されるすべての InnoDB テーブルスペースのファイル形式を決定します。システムテーブルスペース以外のテーブルスペースを作成するには、file-per-table オプションも使用する必要があります。現在のところ、Antelope および Barracuda ファイル形式を指定できます。
Antelope, Barracuda, ファイル形式, file-per-table, innodb_file_per_table, システムテーブルスペース, テーブルスペースも参照

innodb_file_per_table
InnoDB ファイルストレージ、機能の利用可能性、および I/O 特性の多くの側面に影響する、非常に重要な構成オプション。MySQL 5.6.7 以降ではデフォルトで有効になっています。MySQL 5.6.7 より前では、デフォルトで無効になっています。innodb_file_per_table オプションは file-per-table モードをオンにし、新しく作成された InnoDB テーブルおよびそれに関連付けられたインデックスをシステムテーブルスペース外部のそれぞれ独自の .ibd ファイルに格納します。

このオプションは、DROP TABLE や TRUNCATE TABLE など、いくつかの SQL ステートメントのパフォーマンスおよびストレージに関する考慮事項に影響します。

このオプションは、テーブル圧縮などの多くのほかの InnoDB 機能や、MySQL Enterprise Backup での特定のテーブルのバックアップを十分に活用するために必要です。

このオプションは以前は静的でしたが、SET GLOBAL コマンドを使用して設定できるようになりました。

参照情報については、innodb_file_per_table を参照してください。使用法情報については、セクション14.5.2「InnoDB File-Per-Table モード」を参照してください。
圧縮, file-per-table, .ibd ファイル, MySQL Enterprise Backup, システムテーブルスペースも参照

innodb_lock_wait_timeout
innodb_lock_wait_timeout オプションは、共有リソースが利用できるようになるまで待機するか、または放棄してエラーを処理したり、再試行したり、アプリケーションで代替処理を行ったりするかのバランスを設定します。InnoDB トランザクションがロックを獲得するための指定待機時間を超えた場合は、ロールバックします。特に、異なるストレージエンジンで制御される複数のテーブルへの更新によってデッドロックが発生した場合に役立ちます。このようなデッドロックは自動的には検出されません。
デッドロック, デッドロック検出, ロック, 待機も参照

innodb_strict_mode
innodb_strict_mode オプションは、InnoDB が 厳密モード (通常は警告として扱われる条件でエラーを発生させる (そして基礎となるステートメントが失敗する)) で動作するかどうかを制御します。

このモードは MySQL 5.5.5 以降のデフォルト設定です。
厳密モードも参照

IOPS
1 秒あたりの I/O 操作 (I/O Operations Per Second) の頭字語。ビジーシステム、特に OLTP アプリケーションの一般的な測定基準。ストレージデバイスが処理できる最大値にこの値が近い場合、アプリケーションはディスクバウンドになり、スケーラビリティーを制限する場合があります。
ディスクバウンド, OLTP, スケーラビリティーも参照

K

KEY_BLOCK_SIZE
圧縮行フォーマットを使用する InnoDB テーブル内で、データページのサイズを指定するオプション。デフォルトは 8K バイトです。値を小さくすると、行サイズと圧縮比率の組み合わせによって内部制限に達する恐れがあります。
圧縮行フォーマットも参照

L

lock mode
共有ロックでは、トランザクションは行を読み取ることができます。複数のトランザクションが同時にその同じ行で S ロックを獲得できます。

排他 (X) ロックでは、トランザクションは行を更新または削除できます。ほかのトランザクションは、同時にその同じ行でどのようなロックも獲得できません。

インテンションロックは、テーブルレベルに適用され、トランザクションがテーブル内の行で獲得したいロックの種類を示すために使用されます。トランザクションごとに異なる種類のインテンションロックを同じテーブルで獲得できますが、最初のトランザクションがあるテーブルでインテンション排他 (IX) ロックを獲得すると、ほかのトランザクションはそのテーブルで S または X ロックを獲得できません。反対に、最初のトランザクションがあるテーブルでインテンション共有 (IS) ロックを獲得すると、ほかのトランザクションはそのテーブルで X ロックを獲得できません。2 フェーズプロセスは、ロックおよび互換性のある対応操作をブロックせずに、ロックリクエストを順番に解決できます。
インテンションロック, ロック, ロックも参照

loose_
MySQL 5.1 で、サーバーの起動後に InnoDB Plugin をインストールするときに、InnoDB 構成オプションに追加されるプリフィクス。したがって、現在のレベルの MySQL で認識されない新しい構成オプションは起動エラーを引き起こしません。MySQL は、このプリフィクスで始まる構成オプションを処理しますが、プリフィクスに続く部分が認識されるオプションでない場合、エラーではなく警告を返します。
プラグインも参照

LRU
「Least Recently Used」の頭字語。ストレージ領域を管理するための一般的な方法。より新しい項目をキャッシュするための領域が必要なときは、最近使用されていない項目は削除されます。InnoDB は、バッファープール内のページを管理するためにデフォルトで LRU メカニズムを使用しますが、ページが 1 回だけ読み取られる場合 (フルテーブルスキャン中など) を例外扱いします。LRU アルゴリズムのこのバリエーションはミッドポイント挿入戦略と呼ばれます。バッファープール管理が従来の LRU アルゴリズムと異なる点は、オプション innodb_old_blocks_pct、innodb_old_blocks_time と、新しい MySQL 5.6 オプション innodb_lru_scan_depth および innodb_flush_neighbors により微調整されることです。
バッファープール, エビクション, テーブルの完全スキャン, ミッドポイント挿入戦略, ページも参照

LSN
「ログシーケンス番号 (Log Sequence Number)」の頭字語。この任意の増加し続ける値は、Redo ログに記録される操作に対応する時点を表します。(この時点は、トランザクション境界を意識しません。1 つ以上のトランザクションの中間になることがあります。)クラッシュリカバリ中に InnoDB によって内部的に、バッファープールを管理するために使用されます。

MySQL 5.6.3 より前では、LSN は 4 バイト符号なし整数でした。LSN は、MySQL 5.6.3 で 8 バイト符号なし整数になりました。追加サイズ情報を格納するために追加バイトが必要だったので、Redo ログファイルサイズ限度が 4G バイトから 512G バイトに増加したためです。MySQL 5.6.3 以降でビルドされたアプリケーションのうち LSN 値を使用するものは、32 ビット変数ではなく 64 ビット変数を使用して LSN 値を格納および比較することをお勧めします。

MySQL Enterprise Backup 製品では、増分バックアップを取得する時点を表す LSN を指定できます。該当する LSN は、mysqlbackup コマンドの出力で表示されます。完全バックアップの時点に対応する LSN がわかれば、後続の増分バックアップを取得するためにその値を指定でき、その出力には次の増分バックアップのもう 1 つの LSN が含まれます。
クラッシュリカバリ, 増分バックアップ, MySQL Enterprise Backup, Redo ログ, トランザクションも参照

M

MDL
「メタデータロック (MetaData Lock)」の頭字語。
メタデータロックも参照

memcached
多くの MySQL および NoSQL ソフトウェアスタックで広く使用されているコンポーネント。単一値を高速で読み書きでき、結果全体をメモリーにキャッシュします。アプリケーションは従来、永続的ストレージに同じデータを MySQL データベースに書き込むため、またはまだメモリーにキャッシュされていない場合は MySQL データベースからデータを読み取るために、特別なロジックを必要としていました。現在は、アプリケーションは多くの言語のクライアントライブラリでサポートされる単純な memcached プロトコルを使用して、InnoDB または MySQL Cluster テーブルを使用する MySQL Server と直接通信できます。アプリケーションは、これらの NoSQL から MySQL テーブルへのインタフェースを利用することで、SQL コマンドを直接発行するよりも高い読み取り/書き込みパフォーマンスを実現でき、すでにインメモリーキャッシュ用に memcached が組み込まれているシステムのアプリケーションロジックと配備構成を簡略化できます。

InnoDB テーブルへの memcached インタフェースは、MySQL 5.6 以降で利用できます。詳細は、セクション 14.18「InnoDB と memcached の統合」を参照してください。MySQL Cluster テーブルへの memcached インタフェースは、MySQL Cluster 7.2 で使用できます。詳細は、http://dev.mysql.com/doc/ndbapi/en/ndbmemcache.htmlを参照してください。

InnoDB, NoSQLも参照

mtr
ミニトランザクションも参照

MVCC
「マルチバージョン並列性制御 (MultiVersion Concurrency Control)」の略語。この方法を使用すれば、特定の分離レベルを持つ InnoDB トランザクションが、一貫性読み取り操作を実行できます。つまり、ほかのトランザクションが更新している行を照会して、これらの更新が行われる前に値を確認できます。これは、ほかのトランザクションが保持しているロックのために待機することなく、クエリーが進行できるようにすることによって、並列性を高める強力な方法です。

この方法は、データベース世界で共通のものではありません。ほかのデータベース製品やほかの MySQL ストレージエンジンの中には、これをサポートしないものがあります。
ACID, 並列性, 一貫性読み取り, 分離レベル, ロック, トランザクションも参照

my.cnf
UNIX または Linux システムでの MySQL オプションファイルの名前。
my.ini, オプションファイルも参照

my.ini
Windows システムでの MySQL オプションファイルの名前。
my.cnf, オプションファイルも参照

mysql
mysql プログラムは MySQL データベース用のコマンド行インタープリターです。SQL ステートメントを処理し、mysqld デーモンにリクエストを渡すことによって SHOW TABLES などの MySQL 固有コマンドも処理します。
mysqld, SQLも参照

MySQL Enterprise Backup
MySQL データベースのホットバックアップを実行するライセンス製品。InnoDB テーブルをバックアップするときに効率性と柔軟性がもっとも高くなりますが、MyISAM とほかの種類のテーブルもバックアップできます。
ホットバックアップ, InnoDBも参照

mysqlbackup コマンド
MySQL Enterprise Backup 製品のコマンド行ツール。InnoDB テーブルにはホットバックアップ操作を、MyISAM とほかの種類のテーブルにはウォームバックアップ操作を実行します。このコマンドの詳細は、セクション25.2「MySQL Enterprise Backup」を参照してください。
ホットバックアップ, MySQL Enterprise Backup, ウォームバックアップも参照

mysqld
mysqld プログラムは、MySQL データベース用のデータベースエンジンです。UNIX デーモンまたは Windows サービスとして動作し、常にリクエストを待機し、バックグラウンドで保守作業を実行します。
mysqlも参照

mysqldump
データベース、テーブル、およびテーブルデータの組み合わせの論理バックアップを実行するコマンド。結果は、元のスキーマオブジェクトまたはデータ、あるいはその両方を再現する SQL ステートメントです。相当な量のデータの場合、MySQL Enterprise Backup などの物理バックアップソリューションが高速です (特にリストア操作で)。
論理バックアップ, MySQL Enterprise Backup, 物理バックアップ, リストアも参照

N

NoSQL
データ読み書きの主要なメカニズムとして SQL 言語を使用しない、一連のデータアクセステクノロジの一般的な用語。NoSQL テクノロジの中には、単一値の読み取りと買い込みだけを受け入れるキー値ストアとして機能するものがあります。ACID 原理の制限を緩和するものや、事前に計画されたスキーマが不要なものもあります。MySQL ユーザーは、memcached API を使用して何らかの MySQL テーブルに直接アクセスすることにより、速度と簡略化のための NoSQL スタイル処理と、柔軟性と利便性のための SQL 操作を組み合わせることができます。InnoDB テーブルへの memcached インタフェースは、MySQL 5.6 以降で利用できます。詳細は、セクション14.18「InnoDB と memcached の統合」を参照してください。MySQL Cluster テーブルへの memcached インタフェースは、MySQL Cluster 7.2 で使用できます。詳細は、http://dev.mysql.com/doc/ndbapi/en/ndbmemcache.htmlを参照してください。
ACID, InnoDB, memcached, スキーマ, SQLも参照

NOT NULL 制約
カラムが NULL 値を含むことができないと規定するタイプの制約。これは、データベースサーバーが誤って値が失われているデータを識別できるので、参照整合性の維持に役立ちます。また、オプティマイザはそのカラムのインデックス内のエントリ数を予測できるので、クエリー最適化に関係する演算でも役立ちます。

カラム, 制約, NULL, 主キー, 参照整合性も参照

NULL
データが存在しないことを示す、SQL での特殊な値。算術演算や等価性テストに NULL 値が含まれる場合は、それらは NULL 結果を返します。(したがってこれは、IEEE 浮動小数点の NaN (not a number) 概念に似ています。)AVG() などの集計計算は、除算の分母となる行数を決定するときに、NULL 値を含む行を無視します。NULL 値を扱う唯一のテストは、SQL イディオム IS NULL または IS NOT NULL を使用します。

NULL 値はインデックス操作で役割を果たします。パフォーマンスのため、データベースは失われたデータ値を追跡するオーバーヘッドを最小限に抑える必要があるためです。NULL 値は通常、インデックスに格納されません。標準比較演算子を使用してインデックスカラムをテストするクエリーは、そのカラムの行を NULL 値に照合することはできないためです。同じ理由で、一意のインデックスは NULL 値を防止しません。これらの値は単純にインデックスに表示されません。カラムで NOT NULL 制約を宣言することで、インデックスから外れる行がないことが再保証され、クエリー最適化が向上します (行数カウントおよびインデックスを使用するかどうかの評価の精度)。

主キーはテーブル内のすべての行を一意に識別できる必要があるので、単一カラム主キーには NULL 値を含めることはできず、複数カラム主キーにはすべてのカラムで NULL 値を持つ行を含めることはできません。

Oracle データベースでは NULL 値を文字列と連結できますが、InnoDB はこのような操作の結果を NULL として扱います。
インデックス, 主キー, SQLも参照

O

OLTP
「オンライントランザクション処理 (Online Transaction Processing)」の頭字語。データベースシステムまたはデータベースアプリケーションの 1 つ。多数のトランザクション、頻繁な書き込みと読み取りのワークロードを実行し、通常は一度に少量のデータに影響します。たとえば、航空便予約システムや銀行預金を処理するアプリケーションがあります。DML (挿入/更新/削除) 効率性とクエリー効率性とのバランスを取るために、データは正規化形式で編成される場合があります。データウェアハウスと対比してください。

InnoDB は、行レベルロックとトランザクション機能を備え、OLT アプリケーションで使用される MySQL テーブルに理想的なストレージエンジンです。
データウェアハウス, DML, InnoDB, クエリー, 行ロック, トランザクションも参照

P

page size
MySQL 5.5 以前のリリースでは、各 InnoDB ページのサイズは 16K バイトに固定されています。この値は、ほとんどの行のデータを保持できる大きさと、不要なデータをメモリーに転送するパフォーマンスオーバーヘッドを最小化できる小ささとを、調和させたものです。ほかの値は未テストで、サポートされません。

MySQL 5.6 以降、InnoDB インスタンスのページサイズは 4K バイト、8K バイト、または 16K バイトに設定でき、innodb_page_size 構成オプションで制御できます。MySQL 5.7.6 以降の InnoDB は、32K バイトおよび 64K バイトページサイズのサポートも提供します。どちらのページサイズでも、ROW_FORMAT=COMPRESSED はサポートされず、最大レコードサイズは 16K バイトです。

サイズは MySQL インスタンス作成時に設定し、その後は不変です。同じページサイズが、すべての InnoDB テーブルスペース (システムテーブルスペースと、file-per-table モードで別個に作成されるテーブルスペース) に適用されます。

ページサイズが小さいほど、小さなブロックサイズを使用するストレージデバイス (特に、OLTP アプリケーションなどのディスクバウンドワークロードでの SSD デバイス) のパフォーマンスに役立ちます。個々の行が更新されるときに、メモリーにコピーされたり、ディスクに書き込まれたり、再編成されたり、ロックされたりするときのデータ量が少なくなります。
ディスクバウンド, file-per-table, インスタンス, OLTP, ページ, SSD, システムテーブルスペース, テーブルスペースも参照

PITR
ポイントインタイムリカバリ (Point-In-Time Recovery) の頭字語。
ポイントインタイムリカバリも参照

Pthreads
POSIX スレッド標準。UNIX および Linux システムでのスレッドおよびロック操作の API を定義します。UNIX および Linux システムでは、InnoDB はこの相互排他ロック実装を使用します。
相互排他ロックも参照

R

RAID
「Redundant Array of Inexpensive Drives」の頭字語。複数のドライブに I/O 操作を分散することにより、ハードウェアレベルで並列性が向上し、低レベル書き込み操作の効率性が改善します (そうしない場合は順番に実行されます)。
並列性も参照

raw バックアップ
バイナリログと増分バックアップに反映される変更が適用される前の、MySQL Enterprise Backup 製品によって生成されるバックアップファイルの初期セット。この段階では、ファイルはリストアする準備ができていません。これらの変更が適用されたあと、ファイルは準備されたバックアップと呼ばれます。
バイナリログ, ホットバックアップ, ibbackup_logfile, 増分バックアップ, MySQL Enterprise Backup, 準備されたバックアップ, リストアも参照

READ COMMITTED
分離レベルの 1 つ。パフォーマンスを向上させるために、トランザクション間の保護を一部緩和するロック戦略を使用します。トランザクションは、ほかのトランザクションからのコミットされていないデータを見ることはできませんが、現在のトランザクションが開始したあとに別のトランザクションによってコミットされるデータを見ることはできます。したがって、トランザクションは不良データを見ることはありませんが、見るデータはある程度ほかのトランザクションのタイミングに依存する場合があります。

この分離レベルのトランザクションが UPDATE ... WHERE または DELETE ... WHERE 操作を実行するときに、ほかのトランザクションは待機が必要な場合があります。このトランザクションは、ほかのトランザクションを待機させずに SELECT ... FOR UPDATE および LOCK IN SHARE MODE 操作を実行できます。
ACID, 分離レベル, ロック, REPEATABLE READ, SERIALIZABLE, トランザクションも参照

READ UNCOMMITTED
トランザクション間にもっとも少ない量の保護を提供する分離レベル。クエリーは、通常は別のトランザクションを待機する状況で進行することを許可されるロック戦略を採用します。ただし、この追加パフォーマンスには、ほかのトランザクションによって変更されたけれどもまだコミットされていないデータ (ダーティー読み取りと呼ばれます) など、結果の信頼性の低いという犠牲が伴います。この分離レベルを使用するときは十分な注意が必要で、ほかのトランザクションが同時に何を実行しているかによって結果が一貫しなかったり再現性がなくなったりすることを認識してください。この分離レベルのトランザクションは通常、照会だけを行い、挿入、更新、削除操作は行いません。
ACID, ダーティー読み取り, 分離レベル, ロック, トランザクションも参照

Redo
DML ステートメントが InnoDB テーブルに変更を行うときに、Redo ログにレコード単位で記録されるデータ。クラッシュリカバリ中に、不完全なトランザクションによって書き込まれるデータを訂正するために使用されます。増加し続ける LSN 値は、Redo ログを通過した Redo データの累積量を表します。
クラッシュリカバリ, DML, LSN, Redo ログ, トランザクションも参照

Redo ログ
クラッシュリカバリ中に、不完全なトランザクションによって書き込まれるデータを訂正するために使用されるディスクベースデータ構造。通常の操作中に、InnoDB テーブルデータを変更するリクエスト (SQL ステートメント、または NoSQL インタフェースからの低レベル API 呼び出しから発生) をエンコードします。予期しないシャットダウン前にデータファイルの更新を終了していない変更は、自動的に再現されます。

Redo ログはファイルセットとして物理的に表され、通常は ib_logfile0 および ib_logfile1 という名前が付けられます。Redo ログ内のデータは、該当するレコードの単位でエンコードされます。このデータはまとめて Redo と呼ばれます。Redo ログをデータが通貨したことは、増加し続ける LSN 値で表されます。Redo ログの最大サイズは、最初は 4G バイトに制限されていましたが、MySQL 5.6.3 で 512G バイトに上げられています。

Redo ログのディスクレイアウトは、構成オプション innodb_log_file_size、innodb_log_group_home_dir、および (まれに) innodb_log_files_in_group に影響されます。Redo ログ操作のパフォーマンスは、ログバッファーにも影響されます。これは innodb_log_buffer_size 構成オプションによって制御されます。
クラッシュリカバリ, データファイル, ib_logfile, ログバッファー, LSN, Redo, シャットダウン, トランザクションも参照

REPEATABLE READ
InnoDB のデフォルト分離レベル。照会される行がほかのトランザクションによって変更されるのを防ぎます。したがって、反復不可能読み取りはブロックしますが、ファントム読み取りはしません。トランザクション内のすべてのクエリーが同じスナップショットからのデータを見る、つまりトランザクションが開始した時点のデータを見るように、適度に厳密なロック戦略を使用します。

この分離レベルのトランザクションが UPDATE ... WHERE、DELETE ... WHERE、SELECT ... FOR UPDATE、および LOCK IN SHARE MODE 操作を実行するときに、ほかのトランザクションは待機しなければいけない場合があります。

ACID, 一貫性読み取り, 分離レベル, ロック, ファントム, SERIALIZABLE, トランザクションも参照

rw ロック (読み書きロック)
特定のルールに基づく内部インメモリーデータ構造への共有アクセスロックを表現および適用するために、InnoDB が使用する低レベルオブジェクト。相互排他ロック (InnoDB が内部インメモリーデータ構造への排他アクセスを表現および適用するために使用) と対比してください。相互排他ロックと読み書きロックはまとめて、ラッチと呼ばれます。

rw ロックタイプには、s ロック (共有ロック)、x ロック (排他ロック)、および sx ロック (共有排他ロック) などがあります。

- s ロックは、共有リソースへの読み取りアクセスを提供します。
- x ロックは、共有リソースへの書き込みアクセスを提供し、ほかのスレッドによる不整合読み取りを許可しません。
- sx ロックは、共有リソースへの書き込みアクセスを提供し、ほかのスレッドによる不整合読み取りを許可します。sx ロックは、読み取り/書き込みワークロードの並列性を最適化し、スケーラビリティーを改善するために MySQL 5.7 で導入されました。

次の表に rw ロックタイプ互換性をまとめます。

	S	SX	X
S	互換	互換	競合
SX	互換	競合	競合
X	競合	競合	競合

ラッチ, ロック, 相互排他ロック, パフォーマンススキーマも参照

S

SERIALIZABLE
もっとも保守的なロック戦略を使用する分離レベル。このトランザクションが読み取ったデータを (終了するまで) ほかのトランザクションが挿入または変更することを防ぐため。この方法では、トランザクション内で同じクエリーを何度も実行でき、そのたびに同じ結果セットを取得することを保証できます。現在のトランザクションが開始してから別のトランザクションによってコミットされたデータを変更しようとすると、現在のトランザクションは待機します。

これは、SQL 標準で指定されるデフォルト分離レベルです。実際にはこの程度の厳密さはまれにしか必要でないので、InnoDB のデフォルト分離レベルは次にもっとも厳密な反復可能読み取りです。
ACID, 一貫性読み取り, 分離レベル, ロック, REPEATABLE READ, トランザクションも参照

SQL
データベース操作を実行するための標準である構造化クエリー言語。多くの場合、カテゴリ DDL、DML、およびクエリーに分けられます。MySQL には、レプリケーションなどのいくつかの追加ステートメントカテゴリが含まれます。SQL 構文の構成ブロックについては第9章「言語構造」、MySQL テーブルカラムに使用するデータ型については第11章「データ型」、SQL ステートメントとそれらに関連付けられるカテゴリの詳細は第13章「SQL ステートメントの構文」、クエリーで使用する標準および MySQL 固有の関数については第12章「関数と演算子」を参照してください。
DDL, DML, クエリー, レプリケーションも参照

SSD
「ソリッドステートドライブ (Solid-State Drive)」の頭字語。従来のハードディスクドライブ (HDD) とパフォーマンス特性が異なるタイプのストレージデバイス。ストレージ容量が小さく、ランダム読み取りが高速で、可動部分がなく、書き込みパフォーマンスに影響する考慮事項がいくつかあります。そのパフォーマンス特性は、ディスクバウンドワークロードのスループットに影響を与えることがあります。
ディスクバウンド, SSDも参照

T

TPS
「秒あたりのトランザクション数 (Transactions Per Second)」の頭字語。ベンチマークでときどき使用される測定単位。その値は、特定のベンチマークテストで表されるワークロードと、ハードウェア能力やデータベース構成などの制御要因との組み合わせに応じて異なります。
トランザクション, ワークロードも参照

U

Undo
: トランザクションの生存期間保持されるデータ。ロールバック操作の場合に元に戻せるようにすべての変更を記録します。これは、システムテーブルスペース内または別個の Undo テーブルスペース内の Undo ログに格納され、ロールバックセグメントとも呼ばれます。
ロールバック, ロールバックセグメント, システムテーブルスペース, トランザクション, Undo ログ, Undo テーブルスペースも参照

Undo テーブルスペース
: innodb_undo_tablespaces および innodb_undo_directory 構成オプションによって Undo ログがシステムテーブルスペースから分離されているときに、Undo ログを含むファイルセットの 1 つ。MySQL 5.6 以降にのみ適用されます。
システムテーブルスペース, Undo ログも参照

Undo バッファー
: Undo ログも参照

Undo ログ
: アクティブトランザクションによって変更されたデータのコピーを保持するストレージ領域。別のトランザクションが (一貫性読み取り操作の一部として) 元のデータを見る必要がある場合に、未変更データがこのストレージ領域から取得されます。

  この領域はデフォルトで物理的にシステムテーブルスペースの一部です。MySQL 5.6 以降では、innodb_undo_tablespaces および innodb_undo_directory 構成オプションを使用して、これを 1 つまたは複数の別個のテーブルスペースファイル (Undo テーブルスペース) に分割でき、オプションで SSD などの別のストレージデバイスに格納できます。

  Undo ログは、別個の部分、挿入 Undo バッファーと更新 Undo バッファーに分割されます。これらの部分はまとめて、Oracle DBA になじみ深い用語、ロールバックセグメントとも呼ばれます。
  一貫性読み取り, ロールバックセグメント, SSD, システムテーブルスペース, トランザクション, Undo テーブルスペースも参照

W

Windows
: 組み込み InnoDB ストレージエンジンと InnoDB Plugin が、MySQL Server と同じすべての Microsoft Windows バージョンでサポートされます。MySQL Enterprise Backup 製品は、InnoDB Hot Backup 製品の後継ですが、Windows システムをより包括的にサポートします。
InnoDB, MySQL Enterprise Backup, プラグインも参照

X

XA
: 分散型トランザクションを調整するための標準インタフェース。ACID コンプライアンスを維持しながら、複数のデータベースがトランザクションに参加できます。詳細は、セクション13.3.7「XA トランザクション」を参照してください。

  XA 分散トランザクションサポートは、デフォルトでオンになっています。この機能を使用していない場合でも、innodb_support_xa 構成オプションを無効にすることで、トランザクションごとに追加 fsync のパフォーマンスオーバーヘッドを回避できます。
  コミット, トランザクション, 2 フェーズコミットも参照

ア

アトミック命令
: 重要な低レベル操作を中断できないようにするために、CPU によって提供される特殊な命令。

アプリケーションプログラミングインタフェース (API)
: 関数またはプロシージャーのセット。API は、関数、プロシージャー、パラメータ、および戻り値の名前と型の安定的なセットです。

アーリーアダプタ
: ソフトウェア製品が一般的に重要度の低い設定でパフォーマンス、機能、および互換性について評価される、ベータに類似した段階。InnoDB では、ベータではなく、継続的なポイントリリースを経てしだいに GA リリースに至る、アーリーアダプタ名称を使用します。

ベータ, GAも参照

圧縮
使用するディスク領域の縮小、実行する I/O の減少、および使用するキャッシュ用のメモリーの軽減による幅広いメリットを伴う機能。データベース操作中、InnoDB テーブルおよびインデックスデータは圧縮形式で保持できます。

データはクエリーで必要になると圧縮解除され、DML 操作によって変更が行われると再圧縮されます。テーブルの圧縮を有効にしたあと、この処理はユーザーとアプリケーション開発者に対して透過的です。DBA は、information_schema テーブルを参照して、圧縮パラメータが MySQL インスタンスおよび特定の圧縮テーブルに対してどれだけ効率的に機能するかをモニターできます。

InnoDB テーブルデータが圧縮されると、圧縮はテーブル自体、関連付けられたインデックスデータ、およびバッファープールにロードされたページに適用されます。圧縮は、Undo バッファー内のページに適用されません。

テーブル圧縮機能では、MySQL 5.5 以降、または MySQL 5.1 以前の InnoDB Plugin を使用する必要があり、innodb_file_per_table 設定をオンにした状態で Barracuda ファイル形式と圧縮行フォーマットを使用するテーブルを作成する必要があります。テーブルごとの圧縮は、CREATE TABLE および ALTER TABLE ステートメントの KEY_BLOCK_SIZE 句によって影響されます。MySQL 5.6 以降の圧縮は、サーバー全体の構成オプション innodb_compression_failure_threshold_pct、innodb_compression_level、および innodb_compression_pad_pct_max にも影響されます。使用法の詳細は、セクション14.7「InnoDB 圧縮テーブル」を参照してください。

別の種類の圧縮は、MySQL Enterprise Backup 製品の圧縮バックアップ機能です。
Barracuda, バッファープール, 圧縮行フォーマット, DML, ホットバックアップ, インデックス, INFORMATION_SCHEMA, innodb_file_per_table, プラグイン, テーブル, Undo バッファーも参照

圧縮テーブル
データが圧縮形式で格納されたテーブル。InnoDB の場合、これは ROW_FORMAT=COMPRESSED で作成されたテーブルです。詳細は、セクション14.7「InnoDB 圧縮テーブル」を参照してください。
圧縮行フォーマット, 圧縮も参照

圧縮バックアップ
MySQL Enterprise Backup 製品の圧縮機能は、各テーブルスペースの圧縮コピーを作成し、.ibd から .ibz に拡張子を変更します。バックアップデータを圧縮すると、より多くのバックアップを手元に置いておくことができ、別のサーバーにバックアップを転送する時間が短縮します。データはリストア操作中に圧縮解除されます。圧縮バックアップ操作は、すでに圧縮されているテーブルを処理するときに、ふたたび圧縮してもほとんどまたはまったく領域の節約にならないので、そのテーブルの圧縮ステップをスキップします。

MySQL Enterprise Backup 製品が生成するファイルセット。ここでは、各テーブルスペースが圧縮されます。圧縮ファイルは、.ibz ファイル拡張子を使用して名前が変更されます。

バックアッププロセスの開始時に圧縮を適用すると、圧縮プロセス中のストレージオーバーヘッドを回避し、バックアップファイルを別のサーバーに転送するときのネットワークオーバーヘッドを回避するのに役立ちます。バイナリログを適用するプロセスはさらに時間がかかるようになり、バックアップファイルの圧縮解除が必要になります。
適用, バイナリログ, 圧縮, ホットバックアップ, MySQL Enterprise Backup, テーブルスペースも参照

圧縮失敗
実際にはエラーではなくむしろ、圧縮を DML 操作と組み合わせて使用するときに起きる可能性のある負荷のかかる操作です。これは、圧縮ページへの更新が変更記録に予約されたページ上の領域からオーバーフローするとき、すべての変更がテーブルデータに適用してページが再度圧縮されたとき、再圧縮されたデータが元のページ上で適合せず、MySQL がデータを 2 つの新しいページに分割しそれぞれを個別に圧縮するように要求するときに起こります。この状況の頻度を確認するには、テーブル INFORMATION_SCHEMA.INNODB_CMP を照会し、COMPRESS_OPS カラムの値が COMPRESS_OPS_OK カラムの値をどれだけ超えているかをチェックしてください。理論的には、圧縮失敗はめったに起こりません。起きた場合でも、構成オプション innodb_compression_level、innodb_compression_failure_threshold_pct、および innodb_compression_pad_pct_max を調整できます。
圧縮, DML, ページも参照

圧縮行フォーマット
InnoDB テーブルのデータおよびインデックス圧縮を有効にする行フォーマット。これは、InnoDB Plugin で導入され、Barracuda ファイル形式の一部として利用できます。ラージフィールドは、動的行フォーマットと同様に、残りの行データを保持するページとは別に格納されます。インデックスページとラージフィールドの両方が圧縮され、メモリーとディスクの節約をもたらします。データの構造に応じて、メモリーおよびディスク使用量の減少は、使用時にデータを圧縮解除するときのパフォーマンスオーバーヘッドを上回る場合も、上回らない場合もあります。使用法の詳細は、セクション14.7「InnoDB 圧縮テーブル」を参照してください。

InnoDB COMPRESSED 行フォーマットの追加情報については、セクション14.9.3「DYNAMIC および COMPRESSED 行フォーマット」を参照してください。

Barracuda, 圧縮, 動的行フォーマット, 行フォーマットも参照

新しい
InnoDB バッファープール内のページの特性。最近アクセスされたことがあるため、バッファープールデータ構想内で移動され、LRU アルゴリズによってすぐにフラッシュされないことを意味します。この用語は、バッファープールに関連するテーブルの一部の情報スキーマカラム名で使用されます。
バッファープール, フラッシュ, INFORMATION_SCHEMA, LRU, ページも参照

暗黙の行ロック
明確にリクエストしなくても、InnoDB が一貫性を保証するために獲得する行ロック。
行ロックも参照

イ

インスタンス
単一の mysqld デーモン。テーブルのセットで 1 つ以上のデータベースを表すデータディレクトリを管理します。同じサーバーマシン上に複数のインスタンスを持ち、それぞれが専用のデータディレクトリを管理し、独自のポートまたはソケットで待機することは、開発、テスト、および一部のレプリケーションシナリオにおいて一般的です。あるインスタンスがディスクバウンドワークロードを実行している状態でも、サーバーは追加インスタンスを実行するために余分な CPU およびメモリー容量を持つことができます。
データディレクトリ, データベース, ディスクバウンド, mysqld, レプリケーション, サーバーも参照

インストゥルメンテーション
ソースコードレベルでの、チューニングおよびデバッグのパフォーマンスデータを収集するための変更。MySQL では、インストゥルメンテーションによって収集されるデータは、INFORMATION_SCHEMA および PERFORMANCE_SCHEMA データベースを使用して SQL インタフェース経由で公開されます。
INFORMATION_SCHEMA, パフォーマンススキーマも参照

インテンションロック
テーブルレベルに適用するタイプのロック。トランザクションがテーブル内の行でどんなタイプのロックを獲得しようとしているかを示すために使用されます。トランザクションごとに異なる種類のインテンションロックを取得できますが、最初のトランザクションがあるテーブルで インテンション排他 (IX) ロックを獲得すると、ほかのトランザクションはそのテーブルで S または X ロックを獲得できません。反対に、最初のトランザクションがあるテーブルでインテンション共有 (IS) ロックを獲得すると、ほかのトランザクションはそのテーブルで X ロックを獲得できません。2 フェーズプロセスは、ロックおよび互換性のある対応操作をブロックせずに、ロックリクエストを順番に解決できます。このロックメカニズムの詳細は、セクション14.2.3「InnoDB のロックモード」を参照してください。
ロック, lock mode, ロックも参照

インテンション共有ロック
インテンションロックも参照

インテンション排他ロック
インテンションロックも参照

インデックス
テーブルの行を高速ルックアップする機能を提供するデータ構造。通常はこのために、特定のカラムまたはカラムセットのすべての値を表すツリー構造 (B ツリー) を形成します。

InnoDB テーブルは常に、主キーを表すクラスタ化されたインデックスを持ちます。1 つ以上のカラムに 1 つ以上のセカンダリインデックスを定義することもできます。セカンダリインデックスは、その構造に応じて、部分、カラム、または複合インデックスとして分類できます。

インデックスは、クエリーパフォーマンスの重要な側面です。データベースアーキテクトは、アプリケーションが必要とするデータを高速なルックアップできるように、テーブル、クエリー、およびインデックスを設計します。理想的なデータベース設計では、有用なときはカバリングインデックスを使用します。クエリー結果は、実際のテーブルデータを読み取らずに完全にインデックスから計算されます。親テーブルと子テーブルの両方に値が存在するかどうかを効率的に確認するために、各外部キー制約にもインデックスが必要です。

B ツリーインデックスがもっとも一般的ですが、MEMORY ストレージエンジンや InnoDB 適応型ハッシュインデックスと同様に、ハッシュインデックスには別の種類のデータ構造が使用されます。
適応型ハッシュインデックス, B ツリー, 子テーブル, クラスタ化されたインデックス, カラムインデックス, 複合インデックス, カバリングインデックス, 外部キー, ハッシュインデックス, 親テーブル, 部分インデックス, 主キー, クエリー, 行, セカンダリインデックス, テーブルも参照

インデックスキャッシュ
InnoDB 全文検索用のトークンデータを保持するメモリー領域。FULLTEXT インデックスの一部であるカラムでデータが挿入または更新されるときのディスク I/O を最小限に抑えるために、データをバッファリングします。インデックス

キャッシュがいっぱいになると、トークンデータがディスクに書き込まれます。InnoDB FULLTEXT インデックスごとに独自のインデックスキャッシュが割り当てられ、そのサイズは構成オプション innodb_ft_cache_size で制御されます。
全文検索, FULLTEXT インデックスも参照

インデックスヒント
オプティマイザが推奨するインデックスをオーバーライドするための拡張 SQL 構文。たとえば、FORCE INDEX、USE INDEX、および IGNORE INDEX 句。通常は、インデックス付きカラムに値が不均等に分散されていて、カーディナリティーを正確に見積もれないときに使用されます。
カーディナリティー, インデックスも参照

インデックスプリフィクス
複数のカラムに適用されるインデックス (複合インデックスと呼ばれます) で、インデックスの先頭または末尾カラム。複合インデックスの最初の 1、2、3 などのカラムを参照するクエリーはインデックスを使用できます (クエリーがインデックス内のすべてのカラムを参照するわけではない場合でも)。
複合インデックス, インデックスも参照

インデックス統計
統計も参照

インメモリーデータベース
ディスクブロックとメモリー領域間のディスク I/O および変換によるオーバーヘッドを回避するために、メモリー内にデータを保持するタイプのデータベースシステム。インメモリーデータベースの中には、持続性 (ACID 設計概念での「D」) を犠牲にし、ハードウェア、電源、およびほかのタイプの障害に脆弱であり、そのため読み取り専用操作に適しているものがあります。ほかのインメモリーデータベースでは、変更ログをディスクに記録したり不揮発性メモリーを使用したりなどの持続性メカニズムを使用します。

同じ種類のメモリー集約的処理に対応した MySQL 機能には、InnoDB のバッファープール、アダプティブハッシュインデックス、および読み取り専用トランザクション最適化、MEMORY ストレージエンジン、MyISAM キーキャッシュ、および MySQL クエリーキャッシュが含まれます。
ACID, 適応型ハッシュインデックス, バッファープール, ディスクベース, 読み取り専用トランザクションも参照

一時テーブル
データを真に永続的にする必要がないテーブル。たとえば、一時テーブルを複雑な計算または変換の中間結果のストレージ領域として使用できます。この中間データはクラッシュ後にリカバリする必要がありません。データベース製品は、データをディスクに書き込むことや再起動をまたがってデータを保護する手段について緻密さを軽減することで、一時テーブルに対する操作パフォーマンスを手軽に改善できます。

トランザクション終了時やセッション終了時などに、データ自体が所定時に自動的に削除されることもあります。一部のデータベース製品では、テーブル自体も自動的に削除されます。
テーブルも参照

一時テーブルスペース
非圧縮 InnoDB 一時テーブルと関連オブジェクト用のテーブルスペース。このテーブルスペースは MySQL 5.7.1 で導入されました。構成ファイルオプション innodb_temp_data_file_path は、ユーザーが一時データファイルの相対パスを定義することを許可します。innodb_temp_data_file_path が指定されない場合のデフォルト動作は、データディレクトリ内に ibdata1 と一緒に、ibtmp1 という名前の単一自動拡張 12M バイトデータファイルを作成することです。一時テーブルスペースは、サーバー起動ごとに再作成され、動的に生成されるスペース ID を受け取ります。これは、既存のスペース ID との競合回避に役立ちます。一時テーブルスペースを raw デバイスに置くことはできません。一時テーブルを作成できないときまたは作成時のエラーは致命的と扱われ、サーバー起動が拒否されます。

このテーブルスペースは、通常シャットダウンまたは初期中断 (たとえば、ユーザーが間違った起動オプションを指定したときに発生する可能性がある) 時は削除されます。一時テーブルスペースはクラッシュ発生時は削除されません。この場合、データベース管理者はこのテーブルスペースを手動で削除したり、同じ構成でサーバーを再起動 (一時テーブルスペースが削除されて再作成されます) したりできます。
ibtmp ファイルも参照

一般クエリーログ
MySQL Server によって処理された SQL ステートメントの診断およびトラブルシューティングに使用されるタイプのログ。ファイルまたはデータベーステーブルにも格納できます。この機能を使用するには、general_log 構成オプションを通じて有効にする必要があります。sql_log_off 構成オプションを通じて特定の接続に対してこれを無効にできます。

スロークエリーログよりも広い範囲のクエリーを記録します。レプリケーションに使用されるバイナリログとは異なり、一般クエリーログは SELECT ステートメントを含み、厳密な順序を維持しません。詳細は、セクション5.2.3「一般クエリーログ」を参照してください。
バイナリログ, 一般クエリーログ, ログも参照

一般ログ
一般クエリーログも参照

一貫性読み取り
同時に実行しているほかのトランザクションによって行われる変更にかかわらず、スナップショット情報を使用して特定の時点に基づくクエリー結果を提示する読み取り操作。照会されるデータが別のトランザクションによって変更されている場合、元のデータは Undo ログの内容に基づいて再構築されます。この方法は、ほかのトランザクションが終了するのを待機するようにトランザクションを強制することによって、並列性を減少させる可能性のあるいくつかのロック問題を回避します。

反復可能読み取り分離レベルでは、スナップショットは最初の読み取り操作が実行された時点に基づきます。コミットされた読み取り分離レベルでは、スナップショットはそれぞれの一貫性読み取り操作の時点にリセットされます。

一貫性読み取りは、InnoDB が READ COMMITTED および REPEATABLE READ 分離レベルで SELECT ステートメントを処理する際のデフォルトモードです。一貫性読み取りはアクセスするテーブルでロックを設定しないので、テーブルで一貫性読み取りが実行されている間、ほかのセッションはそれらのテーブルを自由に変更できます。

適用可能な分離レベルの技術的な詳細は、セクション14.2.4「一貫性非ロック読み取り」を参照してください。
ACID, 並列性, 分離レベル, ロック, MVCC, READ COMMITTED, READ UNCOMMITTED, REPEATABLE READ, SERIALIZABLE, トランザクション, Undo ログも参照

ウ

ウォームアップ
起動後の一定期間、標準的なワークロードでシステムを実行すること。通常の状況時のようにバッファープールとほかのメモリー領域がいっぱいになります。

このプロセスは、MySQL Server が再起動したり新しいワークロードを課せられたりしたときに、一定時間をかけて自然に発生します。MySQL 5.6 以降では、構成変数 innodb_buffer_pool_dump_at_shutdown=ON および innodb_buffer_pool_load_at_startup=ON を設定して再起動後にバッファープールの内容をメモリーに戻すことによって、ウォームアッププロセスを高速化できます。通常は、複数の実行間で一貫した結果を保証するために、一定期間ワークロードを実行してバッファープールをウォームアップしてから、パフォーマンステストを実行してください。そのようにしない場合、パフォーマンスが最初の実行中に不自然に低くなる場合があります。
バッファープール, ワークロードも参照

ウォームバックアップ
データベースの実行中に行われるが、バックアッププロセス中に一部のデータベース操作を制限するバックアップ。たとえば、テーブルが読み取り専用になることがあります。ビジーなアプリケーションおよび Web サイトの場合、ホットバックアップをお勧めします。
バックアップ, コールドバックアップ, ホットバックアップも参照

エ

エクステント
テーブルスペース内のページのグループ。16K バイトのデフォルトページサイズでは、エクステントに 64 ページが含まれます。MySQL 5.6 では、InnoDB インスタンスのページサイズには 4K バイト、8K バイト、16K バイトがあり、innodb_page_size 構成オプションで制御されます。4K バイト、8K バイト、および 16K バイトのページサイズでは、エクステントサイズは常に 1M バイト (つまり 1048576 バイト) です。

32K バイトおよび 64K バイトの InnoDB ページサイズのサポートが MySQL 5.7.6 で追加されました。32K バイトページサイズの場合、エクステントサイズは 2M バイトです。64K バイトページサイズの場合、エクステントサイズは 4M バイトです。

セグメント、先読みリクエスト、および二重書き込みバッファーなどの InnoDB 機能は、一度に 1 つのエクステントでデータの読み取り、書き込み、割り当て、または解放を行う I/O 操作を使用します。
二重書き込みバッファー, 隣接ページ, ページ, page size, 先読み, セグメント, テーブルスペースも参照

エビクション
キャッシュや、InnoDB バッファープールなどのほかの一時ストレージ領域から項目を削除する処理。常にではないけれども多くの場合、LRU アルゴリズムを使用して、削除する項目を決定します。ダーティーページが削除されるとき、その内容はディスクにフラッシュされ、隣接するダーティーページもフラッシュされる可能性があります。
バッファープール, ダーティーページ, フラッシュ, LRUも参照

エラーログ
MySQL 起動に関する情報と、重要な実行時エラーおよびクラッシュ情報を示すタイプのログ。詳細は、セクション5.2.2「エラーログ」を参照してください。
クラッシュ, ログも参照

永続的統計
MySQL 5.6 の機能の 1 つ。ディスク上の InnoDB テーブルのインデックス統計を格納し、クエリーの計画安定性が向上します。詳細は、セクション14.13.16.1「永続的オプティマイザ統計のパラメータの構成」を参照してください。
インデックス, オプティマイザ, 計画安定性, クエリー, テーブルも参照

オ

オフページカラム
長すぎるため B ツリーページに収まらない可変長データを含むカラム (BLOB や VARCHAR など)。データはオーバーフローページに格納されます。このようなストレージには、InnoDB Barracuda ファイル形式の DYNAMIC 行フォーマットが、古い COMPACT 行フォーマットよりも効率的です。
B ツリー, Barracuda, オーバーフローページも参照

オプション
MySQL の構成パラメータ。オプションファイルに格納されるか、コマンド行で渡されます。

InnoDB テーブルに適用されるオプションの場合、各オプション名はプリフィクス innodb_ で始まります。
InnoDB, オプションファイルも参照

オプションファイル
MySQL インスタンスの構成オプションを保持するファイル。従来より、Linux および UNIX ではこのファイルは my.cnf という名前で、Windows では my.ini という名前です。
構成ファイル, my.cnf, オプションも参照

オプティマイザ
該当するテーブルの特性とデータ分布に基づいて、クエリーに使用する最適なインデックスおよび結合順序を決定する MySQL コンポーネント。
インデックス, 結合, クエリー, テーブルも参照

オプティミスティック
リレーショナルデータベースシステムの低レベル実装決定を導く概念。リレーショナルデータベースでパフォーマンスと並列性が必要であることは、操作を迅速に開始またはディスパッチする必要があることを意味します。一貫性と参照整合性が必要であることは、どの操作も失敗する可能性があることを意味します。つまり、トランザクションがロールバックされ、DML 操作が制約を違反し、ロックのリクエストがデッドロックになり、ネットワークエラーがタイムアウトになる可能性があります。オプティミスティック戦略は、ほとんどのリクエストまたは試行が成功することを想定するもので、障害ケースに備えるための作業は相対的に少量です。この想定が true のときは、データベースは不要な作業をほとんど行いません。リクエストが失敗したときは、変更をクリーンアップして元に戻すために追加作業が必要になります。

InnoDB は、ロックやコミットなどの操作にオプティミスティック戦略を使用します。たとえば、トランザクションによって変更されたデータは、コミットが発生する前にデータファイルに書き込むことができるため、コミット自体は非常に速いですが、トランザクションがロールバックされた場合に変更を元に戻すためにより多くの作業が必要です。

オプティミスティック戦略の反対がペシミスティック戦略で、信頼性が低く頻繁に失敗する操作に対応するようにシステムが最適化されます。この概念はデータベースシステムではまれです。信頼性の高いハードウェア、ネットワーク、およびアルゴリズムを選択することに非常の多くの努力が注ぎ込まれるためです。
コミット, 並列性, DML, ロック, ペシミスティックも参照

オンライン
データベースのダウンタイム、ブロック、または制限された操作のないタイプの操作。通常は DDL に適用されます。高速インデックス作成など、制限された操作の期間を短かくする操作は、MySQL 5.6 で 幅広いオンライン DDL 操作セットに進化しました。

バックアップのコンテキストでは、ホットバックアップはオンライン操作、ウォームバックアップは部分的にオンライン操作です。
DDL, 高速インデックス作成, ホットバックアップ, オンライン DDL, ウォームバックアップも参照

オンライン DDL
DDL (主に ALTER TABLE) 操作中の InnoDB テーブルのパフォーマンス、並列性、および可用性を改善する機能。詳細は、セクション14.11「InnoDB とオンライン DDL」を参照してください。

詳細は、操作の種類に応じて異なります。場合によっては、ALTER TABLE の進行中にテーブルを同時に変更できます。この操作はテーブルコピーを行わずに、または特別に最適化されたタイプのテーブルコピーを使用せずに実行できる場合があります。領域の使用量は、innodb_online_alter_log_max_size 構成オプションで制御されます。

この機能は、MySQL 5.5、および MySQL 5.1 用の InnoDB Plugin の高速インデックス作成機能の拡張です。

DDL, 高速インデックス作成, オンラインも参照

オーバーフローページ
可変長カラム (BLOB や VARCHAR など) が長すぎて B ツリーページに収まらない場合に、それらを保持するために別個に割り当てられたディスクページ。関連付けられたカラムはオフページカラムと呼ばれます。
B ツリー, オフページカラム, ページも参照

親テーブル
子テーブルに (から) ポイントされた初期カラム値を保持する、外部キー関係のテーブル。親テーブル内の行を削除または更新したときの結果は、外部キー定義の ON UPDATE および ON DELETE 句によって異なります。子テーブル内の対応する値の行が、それに合わせて自動的に削除または更新されたり、これらのカラムが NULL に設定されたり、操作が妨げられたりします。
子テーブル, 外部キーも参照

カ

カウンタ
特定の種類の InnoDB 操作によって増分される値。サーバーがどれだけビジーかを測定する、パフォーマンス問題の原因をトラブルシューティングする、(たとえば、クエリーが使用する構成設定またはインデックスへの) 変更が目的の低レベル効果を持っているかどうかをテストする場合に役立ちます。performance_schema テーブルと information_schema テーブル、特に information_schema.innodb_metrics を通じて、さまざまなカウンタを利用できます。
INFORMATION_SCHEMA, メトリックカウンタ, パフォーマンススキーマも参照

カバリングインデックス
クエリーによって取得されたすべてのカラムを含むインデックス。完全なテーブル行を見つけるためのポインタとしてインデックス値を使用する代わりに、クエリーはインデックス構造から値を返し、ディスク I/O を節約します。InnoDB セカンダリインデックスには主キーカラムも含まれるので、InnoDB は MyISAM よりも多くのインデックスにこの最適化方法を適用できます。InnoDB は、トランザクションが終わるまで、そのトランザクションによって変更されたテーブルに対するクエリーにこの方法を適用できません。

正しいクエリーの場合、どのカラムインデックスまたは複合インデックスでも、カバリングインデックスとして機能できます。可能な場合は必ず、この最適化方法を活用するようにインデックスおよびクエリーを設計してください。
カラムインデックス, 複合インデックス, インデックス, セカンダリインデックスも参照

カラム
ストレージおよびセマンティクスがデータ型によって定義される、行内のデータ項目。それぞれのテーブルおよびインデックスは主に、そこに含まれるカラムセットによって定義されます。

それぞれのカラムにはカーディナリティー値があります。カラムは、そのテーブルの主キーであったり、主キーの一部であったりします。カラムは、一意制約または NOT NULL 制約、あるいはその両方により制限される場合があります。別のカラム内の値には、異なるテーブルにまたがっている場合でも、外部キー関係によってリンクできます。

MySQL 内部操作についてディスカッションするときに、シノニムとしてフィールドが使用されることがあります。
カーディナリティー, 外部キー, インデックス, 主キー, 行, SQL, テーブル, 一意制約も参照

カラムインデックス
単一カラムのインデックス。
複合インデックス, インデックスも参照

カラムプリフィクス
インデックスが CREATE INDEX idx ON t1 (c1(N)) などの長さ指定で作成された場合、カラム値の最初の N 文字がインデックスに格納されます。インデックスプリフィクスを小さいまま維持すると、インデックスがコンパクトになり、メモリーとディスク I/O の節約によりパフォーマンスが向上します。(ただし、インデックスプリフィクスを小さくし過ぎると、値の異なる行がクエリーオプティマイザには重複していると見えるようになり、クエリー最適化を妨げる可能性があります。)

バイナリ値や長いテキスト文字列を含むカラムの場合、ソートは主な考慮事項ではなく、値全体をインデックスに格納すると領域が浪費されるので、インデックスは自動的に、最初の N (通常 768) 文字を使用してルックアップおよびソートを行います。
インデックスも参照

カーソル
クエリー、または SQL WHERE 句を使用する検索を実行するほかの操作の結果セットを表すために使用される内部データ構造。これは、ほかの高級言語でのイテレータのように機能し、リクエストに応じて結果セットからそれぞれの値を生成します。

通常は SQL が自動的にカーソルの処理を扱いますが、パフォーマンスが重要なコードを処理するときは内部作業を詳しく調べることができます。
クエリーも参照

カーディナリティー
テーブルカラム内の異なる値の数。インデックスが関連付けられたカラムをクエリーが参照するときに、各カラムのカーディナリティーはどのアクセス方法がもっとも効率的かに影響します。たとえば、一意制約が適用されるカラムの場合、異なる値の数はテーブル内の行数に等しくなります。テーブル内の行数が 100 万なのに、特定のカラムの異なる値が 10 個だけの場合は、各値は (平均) 100,000 回発生します。したがって、SELECT c1 FROM t1 WHERE c1 = 50; などのクエリーが 1 行を返したり非常に多数の行を返したりする可能性があり、データベースサーバーはこのクエリーを c1 のカーディナリティーに応じてさまざまに処理できます。

カラム内の値が非常に不規則に分布している場合は、カーディナリティーが最善のクエリー計画を決定するうえで適切な方法でないことがあります。たとえば、SELECT c1 FROM t1 WHERE c1 = x; は x=50 のときに 1 行を返し、x=30 のときに 100 万行を返します。このような場合、どのルックアップ方法が特定のクエリーにより効率的かについてアドバイスを伝えるために、インデックスヒントを使用する必要があります。

カーディナリティーは、複合インデックスの場合と同様に、複数のカラムに存在する異なる値の数にも適用できます。

InnoDB の場合、インデックスのカーディナリティーを見積もるプロセスは、innodb_stats_sample_pages および innodb_stats_on_metadata 構成オプションの影響を受けます。永続的統計機能が有効のときは、見積もられる値はより安定します (MySQL 5.6 以降)。
カラム, 複合インデックス, インデックス, インデックスヒント, 永続的統計, ランダムダイブ, 選択性, 一意制約も参照

一意のインデックス
一意制約が適用されるカラムまたはカラムセットのインデックス。インデックスが重複値を含まないことがわかっているので、特定の種類のルックアップとカウント操作が通常の種類のインデックスよりも効率的になります。このタイプのインデックスに対するほとんどのルックアップは、単純に特定の値が存在するかどうかを判断することです。インデックス内の値の数は、テーブル内の行数と同じであるか、関連付けられたカラムが NULL 以外の値である行の数以上です。

挿入バッファリング最適化は一意のインデックスに適用されません。回避策として、InnoDB テーブルへの一括データロード中に一時的に unique_checks=0 を設定できます。
カーディナリティー, 挿入バッファリング, 一意制約, 一意のキーも参照

一意のキー
一意のインデックスを構成するカラムセット (1 つまたは複数)。正確に 1 行に一致する WHERE 条件を定義でき、クエリーが関連付けられた一意のインデックスを使用できるときは、ルックアップおよびエラー処理を非常に効率的に実行できます。
カーディナリティー, 一意制約, 一意のインデックスも参照

一意制約
重複値をカラムに含むことができないと表明するタイプの制約。リレーショナル代数の観点では、1 対 1 関係を指定するために使用されます。ある値を挿入できるかどうかをチェックするときの効率性のために (つまり、その値がカラムにまだ存在していない)、一意制約が基礎となる一意のインデックスでサポートされます。
制約, リレーショナル, 一意のインデックスも参照

可用性
ホストでの障害 (MySQL、オペレーティングシステム、ハードウェア、メンテナンス作業での障害を含む) に対処し、必要に応じてリカバリする能力 (ない場合は、ダウンタイムが発生する可能性があります)。多くの場合、大規模開発の重要な側面として、スケーラビリティーと組み合わされます。
スケーラビリティーも参照

外部キー
個別の InnoDB テーブル内の行の間のポインタ関係。外部キー関係は、親テーブルおよび子テーブルの 1 つのカラムに定義されます。

外部キーは、関連情報の高速ルックアップを有効にすることに加えて、データが挿入、更新、および削除されるときにこれらのポインタが無効になるのを防ぐことによって参照整合性を適用するのに役立ちます。この適用メカニズムは一種の制約です。別のテーブルをポイントする行は、関連付けられた外部キー値がその別のテーブルに存在しない場合、挿入できません。行が削除されているか、その外部キー値が変化していて、別のテーブル内の行がその外部キー値をポイントしている場合は、削除を防止したり、ほかのテーブル内の対応するカラム値が NULL になるようにしたり、ほかのテーブル内の対応する行を自動的に削除したりするように外部キーを設定できます。

正規化データベースを設計する際の段階の 1 つは、複製されるデータを特定し、そのデータを新しいテーブルに分離し、結合操作を使用して複数のテーブルを単一テーブルのように照会できるように外部キー関係を設定することです。
子テーブル, FOREIGN KEY 制約, 結合, 正規化, NULL, 親テーブル, 参照整合性, リレーショナルも参照

完全バックアップ
: 各 MySQL データベース内のすべてのテーブルと MySQL インスタンス内のすべてのデータベースを含むバックアップ。部分バックアップと対比してください。
バックアップ, データベース, インスタンス, 部分バックアップ, テーブルも参照

書き込み結合
: ダーティーページが InnoDB バッファープールからフラッシュされるときの書き込み操作を減らす最適化手法。ページ内の行が複数回更新されたり、同じページ上の複数の行が更新されたりする場合、変更ごとに一度の書き込みではなく、単一書き込み操作でこれらのすべての変更がデータファイルに格納されます。
バッファープール, ダーティーページ, フラッシュも参照

関連性
: 全文検索機能で、検索文字列と FULLTEXT インデックス内のデータとの間の類似性を示す数字。たとえば、単一語を検索するときに、その語は通常、一度だけ現れる行よりも、テキストに複数回発生した行に高い関連性があります。
全文検索, FULLTEXT インデックスも参照

キ

キャッシュ
: 頻繁または高速に取り出せるようにデータのコピーを格納するメモリー領域を表す一般的な用語。InnoDB では、主要な種類のキャッシュ構造がバッファープールです。
バッファー, バッファープールも参照

ギャップ
: InnoDB インデックスデータ構造内で、新しい値を挿入できる場所。SELECT ... FOR UPDATE などのステートメントを持つ行セットをロックすると、InnoDB はインデックス内の実際の値と同様に、ギャップに適用されるロックを作成できます。たとえば、更新に 10 より大きな値をすべて選択した場合、ギャップロックはほかのトランザクションが 10 を超える新しい値を挿入するのを妨げます。最小上限レコードと最大下限レコードは、現在のすべてのインデックス値よりも大きな値または小さな値をすべて含むギャップを表します。
並列性, ギャップロック, インデックス, 最大下限レコード, 分離レベル, 最小上限レコードも参照

ギャップロック
: インデックスレコード間のギャップのロック、または先頭インデックスレコードの前や末尾インデックスレコードのあとのギャップのロック。たとえば、SELECT c1 FOR UPDATE FROM t WHERE c1 BETWEEN 10 and 20; は、すでにカラムにこのような値があったかどうかにかかわらず、ほかのトランザクションがカラム t.c1 に 値 15 を挿入するのを妨ぎます。範囲内にあるすでに存在するすべての値の間のギャップがロックされるためです。レコードロックおよびネクストキーロックと対比してください。

  ギャップロックは、パフォーマンスと並列性とのトレードオフの一環であり、一部のトランザクション分離レベルで使用され、ほかでは使用されません。
ギャップ, 最大下限レコード, ロック, ネクストキーロック, レコードロック, 最小上限レコードも参照

共有テーブルスペース
: システムテーブルスペースを参照する別の方法。
システムテーブルスペースも参照

共有ロック
: ロックの一種。ほかのトランザクションがロックされたオブジェクトを読み取ることを許可し、それに対するほかの共有ロックを獲得することも許可するけれども、それに書き込むことは許可しません。排他ロックの反対。
排他ロック, ロック, トランザクションも参照

切り捨て
: テーブルおよび関連するインデックスを残しながら、テーブルの内容全体を削除する DDL 操作。ドロップと対比してください。概念的には WHERE 句のない DELETE ステートメントと同じ結果になりますが、内部での操作は異なります。InnoDB は新しい空のテーブルを作成し、古いテーブルを削除してから、新しいテーブルの名前を変更して古いテーブルと置き換えます。これは DDL 操作なので、ロールバックできません。

  切り捨てられるテーブルに別のテーブルを参照する外部キーが含まれる場合は、ON DELETE CASCADE 句の必要に応じて参照先テーブル内の対応する行を削除できるように、切り捨て操作はより遅い操作方法を使用して一度に 1 行を削除します。(MySQL 5.5 以降では、このより遅い形式の切り捨てを許可しない代わりに、外部キーが含まれている場合は エラーを返します。この場合は、代わりに DELETE ステートメントを使用してください。
DDL, ドロップ, 外部キー, ロールバックも参照

疑似レコード
: 現在存在しないキー値または範囲をロックするために使用される、インデックス内の人為レコード。

最大下限レコード, ロック, 最小上限レコードも参照

起動
MySQL Server を起動させるプロセス。通常、セクション4.3「MySQL サーバーとサーバー起動プログラム」に示すプログラムのいずれかによって行われます。シャットダウンの反対。
シャットダウンも参照

逆引用符
MySQL SQL ステートメント内の識別子は、特殊文字または予約語を含んでいる場合、逆引用符文字 (`) で囲む必要があります。たとえば、FOO#BAR というテーブルまたは SELECT というカラムを参照するには、`FOO#BAR` および `SELECT` として識別子を指定します。逆引用符は、安全性のレベルをさらに高めるので、識別子名が前もってわかっていない可能性のあるプログラム生成の SQL ステートメントで広く使用されています。

ほかのデータベースシステムの多くでは、このような特別な名前には二重引用符 (") を使用します。MySQL では移植性のために、ANSI_QUOTES モードを有効にして、逆引用符の代わりに二重引用符を使用して識別子名を修飾できます。
SQLも参照

ク

クエリー
SQL で、1 つ以上のテーブルから情報を読み取る操作。データの編成とクエリーのパラメータに応じて、ルックアップはインデックスを参照することで最適化される可能性があります。複数のテーブルが使用される場合、そのクエリーは結合と呼ばれます。

これまでの経緯が理由で、ステートメントの内部処理のディスカッションでは (DDL および DML ステートメントなどのほかのタイプの MySQL ステートメントを含めて)、「クエリー」をより広い意味で使用することがあります。
DDL, DML, インデックス, 結合, SQL, テーブルも参照

クエリーログ
一般クエリーログも参照

クエリー実行計画
使用するインデックスやテーブルを結合する順序など、クエリーをもっとも効率的に実行する方法についてオプティマイザによって行われる決定のセット。計画安定性には、特定のクエリーについて一貫して行われている同じ選択が含まれます。
インデックス, 結合, 計画安定性, クエリーも参照

クライアント
リクエストをサーバーに送信し、結果を解釈または処理するタイプのプログラム。クライアントソフトウェアは、一定の時間だけ実行する場合 (メールやチャットプログラムなど) や、インタラクティブに実行する場合 (mysql コマンドプロセッサなど) があります。
mysql, サーバーも参照

クラスタ化されたインデックス
主キーインデックスを表す InnoDB 用語。InnoDB テーブルストレージは、主キーカラムを使用するクエリーとソートの速度を上げるために、主キーカラムの値に基づいて編成されます。パフォーマンスが最適になるように、パフォーマンスがもっとも重要なクエリーに基づいて、主キーカラムを慎重に選択してください。クラスタ化されたインデックスのカラムを変更することは負荷のかかる操作なので、まれにしか、またはまったく更新されない主カラムを選択してください。

Oracle Database 製品では、この種のテーブルはインデックス編成テーブルと呼ばれています。
インデックス, 主キー, セカンダリインデックスも参照

クラッシュ
MySQL では、サーバーが通常のクリーンアップを行えない予期しないシャットダウン操作を一般的に示すために、用語「クラッシュ」を使用します。たとえば、クラッシュは、データベースサーバーマシンまたはストレージデバイスでのハードウェア障害、電源障害、MySQL Server の停止を招く潜在的なデータ不一致、DBA で開始された高速シャットダウン、またはその他の多くの理由によって発生することがあります。InnoDB テーブルの堅牢で自動的なクラッシュリカバリは、DBA が追加作業を行うことなく、サーバーが再起動するときにデータの一貫性を保証します。
クラッシュリカバリ, 高速シャットダウン, InnoDB, Redo ログ, シャットダウンも参照

クラッシュリカバリ
クラッシュ後に MySQL が再度起動したときに行われるクリーンアップアクティビティー。InnoDB テーブルの場合、不完全なトランザクションからの変更は、Redo ログからのデータを使用して再現されます。クラッシュ前にコミットされたけれども、まだデータファイルに書き込まれていない変更は、二重書き込みバッファーから再構築されます。通常どお

りデータベースがシャットダウンした場合、このタイプのアクティビティーは、パージ操作によってシャットダウン中に実行されます。

通常操作中、コミットされたデータは、データファイルに書き込まれる前に、一定期間変更バッファーに格納できます。データファイルを最新の状態に維持すること (通常の操作中にパフォーマンスオーバーヘッドをもたらす) と、データのバッファリング (シャットダウンおよびクラッシュリカバリの時間を長くする可能性がある) との間には、常にトレードオフがあります。
変更バッファー, コミット, クラッシュ, データファイル, 二重書き込みバッファー, InnoDB, パージ, Redo ログも参照

クリーンシャットダウン
エラーなしで完了し、終了前に InnoDB テーブルにすべての変更を適用するシャットダウン。クラッシュまたは高速シャットダウンとは異なります。低速シャットダウンのシノニム。
クラッシュ, 高速シャットダウン, シャットダウン, 低速シャットダウンも参照

クリーンページ
InnoDB バッファープール内のページ の 1 つ。ここでは、メモリー内で行われたすべての変更がデータファイルにも書き込まれて (フラッシュされて) います。ダーティーページの反対です。
バッファープール, データファイル, ダーティーページ, フラッシュ, ページも参照

グループコミット
コミットごとに別々にフラッシュおよび同期するのではなく、一連のコミット操作に対して一度、いくつかの低レベル I/O 操作 (ログ書き込み) を実行する InnoDB 最適化。

バイナリログが有効になっているときは通常、構成オプション sync_binlog=0 も設定してください。バイナリログのグループコミットは、これが 0 に設定されている場合にのみサポートされるためです。
コミット, プラグイン, XAも参照

組み込み
MySQL 内の組み込み InnoDB ストレージエンジンは、ストレージエンジンの元の配布形式です。InnoDB Plugin と対比してください。MySQL 5.5 以降では、InnoDB Plugin は、組み込み InnoDB ストレージエンジン (InnoDB 1.1 と呼ばれます) として、MySQL コードベースに元どおりマージされます。

この区別は、主に MySQL 5.1 で重要です。ここでは、機能またはバグ修正が InnoDB Plugin に適用され、組み込み InnoDB にされない場合や、その反対の場合があります。
InnoDB, プラグインも参照

行ベースレプリケーション
スレーブサーバーで個々の行を変更する方法を指定するイベントが、マスターサーバーから伝播される形式のレプリケーション。innodb_autoinc_lock_mode オプションのすべての設定に安全に使用できます。
自動インクリメントロック, innodb_autoinc_lock_mode, マスターサーバー, レプリケーション, スレーブサーバー, ステートメントベースレプリケーションも参照

行ロック
ロックの 1 つ。互換性のない方法で別のトランザクションが行にアクセスするのを防ぎます。同じテーブル内のほかの行には、ほかのトランザクションが自由に書き込むことができます。これは、InnoDB テーブルでの DML 操作によって行われるタイプのロックです。

MyISAM によって、またはオンライン DDL では行えない InnoDB テーブルでの DDL 操作中に使用される、テーブルロックと対比してください。これらのロックはテーブルへの同時アクセスをブロックします。
DDL, DML, InnoDB, ロック, ロック, オンライン DDL, テーブルロック, トランザクションも参照

ケ

原子的
SQL コンテキストでは、トランザクションとは、完全に完了する (コミット時)、またはまったく効果がない (ロールバック時) 作業の単位です。トランザクションの分割できない (「原子的」) という特性は、頭字語 ACID の「A」に当たります。
ACID, コミット, ロールバック, トランザクションも参照

厳密モード
innodb_strict_mode オプションで制御される設定の一般名。この設定をオンにすると、通常は警告として扱われる条件がエラーと見なされます。たとえば、ファイル形式と行フォーマットに関連するオプションの無効な組み合わせは、通常は警告を返してデフォルト値で継続しますが、CREATE TABLE 操作で失敗するようになります。

MySQL にも厳密モードと呼ばれるものがあります。
ファイル形式, innodb_strict_mode, 行フォーマットも参照

検索インデックス
MySQL の全文検索クエリーは、特殊なインデックス、FULLTEXT インデックスを使用します。MySQL 5.6.4 以降で、InnoDB および MyISAM テーブルの両方は FULLTEXT インデックスをサポートします。以前はこれらのインデックスは MyISAM テーブルでのみ利用可能でした。
全文検索, FULLTEXT インデックスも参照

結合
同じ値を保持するテーブル内のカラムを参照することによって、複数のテーブルからデータを取得するクエリー。理論的には、これらのカラムは InnoDB 外部キー関係の一部で、参照整合性と、結合カラムがインデックス付きであることを保証します。多くの場合、正規化データ設計で、繰り返される文字列を数値 ID に置き換えることによって領域を節約してクエリーパフォーマンスを改善するために、使用されます。
外部キー, インデックス, 正規化, クエリー, 参照整合性も参照

計画安定性
クエリー実行計画のプロパティーの 1 つ。オプティマイザが渡されたクエリーに毎回同じ選択を行うことで、パフォーマンスが一貫して予測可能になります。
クエリー, クエリー実行計画も参照

コ

コミット
トランザクションを終了させ、トランザクションによって行われた変更を永続的にする SQL ステートメント。これは、トランザクションで行われた変更を元どおりにするロールバックの反対です。

InnoDB はコミットにオプティミスティックメカニズムを使用するので、コミットが実際に行われる前に変更をデータファイルに書き込むことができます。この方法は、コミット自体をより高速にしますが、ロールバックの場合には必要な作業が増えるというトレードオフが生じます。

MySQL はデフォルトで、各 SQL ステートメントに続いてコミットを自動的に発行する自動コミット設定を使用します。
自動コミット, オプティミスティック, ロールバック, SQL, トランザクションも参照

コンパクト行フォーマット
MySQL 5.0.3 以降のデフォルト InnoDB 行フォーマット。Antelope ファイル形式を使用するテーブルで利用できます。この Null および可変長フィールドの表現は、以前のデフォルト (冗長行フォーマット) よりもコンパクトです。

InnoDB の B ツリーインデックスは行ルックアップを非常に高速にするため、すべての行を同じサイズに維持することにパフォーマンス上のメリットはほとんどありません。

InnoDB COMPACT 行フォーマットの詳細は、セクション14.9.4「COMPACT および REDUNDANT 行フォーマット」を参照してください。
Antelope, ファイル形式, 冗長行フォーマット, 行フォーマットも参照

コールドバックアップ
データベースがシャットダウンしている間に行われるバックアップ。ビジーなアプリケーションおよび Web サイトの場合は、これは実用的でない可能性があり、ウォームバックアップまたはホットバックアップをお勧めします。
バックアップ, ホットバックアップ, ウォームバックアップも参照

合成キー
値が任意に割り当てられるインデックスカラム (通常は主キー)。多くの場合、自動インクリメントカラムを使用して行われます。完全に任意として値を扱うことによって、過度に制限されたルールと欠陥のあるアプリケーション想定を回避できます。たとえば、ある従業員が雇用を承認されたけれども、実際には入社していない場合、従業員数を表す数値シーケンスにギャップがある可能性があります。または、従業員番号 100 と従業員番号 500 が会社を退職してあとで再入社した場合、前者が後者よりもあとの雇用日になることがあります。数値も、予測可能な長さより短い値になります。たとえば、「道路」、「大通り」、「高速道路」などを意味する数値コードを格納することで、何度もこれらの文字列を繰り返すよりも領域効率が向上します。

サロゲートキーとも呼ばれます。ナチュラルキーと対比してください。
自動インクリメント, ナチュラルキー, 主キー, サロゲートキーも参照

固定行フォーマット
この行フォーマットは、InnoDB ではなく MyISAM ストレージエンジンで使用されます。オプション row_format=fixed を使用して InnoDB テーブルを作成する場合、InnoDB はこのオプションの代わりにコンパクト行フォーマットを使用するように変換します。ただし、SHOW TABLE STATUS レポートなどの出力には fixed 値が表示される場合があります。
コンパクト行フォーマット, 行フォーマットも参照

子テーブル
外部キー関係で子テーブルとは、その行が別のテーブル内の行を参照 (またはポイント) し、特定のカラムについて同じ値を持つもののことです。これは、FOREIGN KEY ... REFERENCES 句、およびオプションで ON UPDATE および ON DELETE 句を含むテーブルです。行を子テーブル内に作成するには、その前に親テーブル内に対応する行が存在している必要があります。子テーブル内の値は、外部キーの作成時に使用される ON CASCADE に基づいて、親テーブルでの削除または更新操作を禁止したり、子テーブル内で自動的に削除または更新したりする場合があります。
外部キー, 親テーブルも参照

構成ファイル
起動時に MySQL が使用するオプション値を保持するファイル。従来より、Linux および UNIX ではこのファイルは my.cnf という名前で、Windows では my.ini という名前です。ファイルの [mysqld] セクションで、InnoDB に関連した多数のオプションを設定できます。

このファイルは通常、場所 /etc/my.cnf、/etc/mysql/my.cnf、/usr/local/mysql/etc/my.cnf、および ~/.my.cnf で検索されます。このファイルの検索パスの詳細は、セクション4.2.6「オプションファイルの使用」を参照してください。

MySQL Enterprise Backup 製品を使用するときには通常、2 つの構成ファイルを使用します。データが得られる場所と構造化される方法を指定するもの (実際のサーバー用の元の構成ファイルである場合もあります) と、バックアップデータの保存先とそれが構造化される方法を指定する小さなオプションセットを含む必要最低限のものです。MySQL Enterprise Backup 製品で使用される構成ファイルは通常の構成ファイルには通常は含まれていないオプションを含んでいる必要があり、その場合には既存の構成ファイルに MySQL Enterprise Backup で使用するいくつかのオプションを追加する必要がある場合があります。
my.cnf, オプションファイルも参照

混在モード挿入
INSERT ステートメントの 1 つ。自動インクリメント値が新しい行のすべてではなく一部に指定されます。たとえば、複数値 INSERT では、一部のケースで自動インクリメントカラムの値を、ほかのケースで NULL を指定できます。InnoDB は、カラム値が NULL として指定された行に自動インクリメント値を生成します。別の例が、INSERT ... ON DUPLICATE KEY UPDATE ステートメントです。ここでは、INSERT ではなく UPDATE ステートメントとして処理される重複行がある場合、それらに自動インクリメント値が生成されますが、使用されません。

レプリケーション構成のマスターおよびスレーブサーバー間で一貫性の問題を引き起こす可能性があります。innodb_autoinc_lock_mode 構成オプションの値の調整が必要な場合があります。
自動インクリメント, innodb_autoinc_lock_mode, マスターサーバー, レプリケーション, スレーブサーバーも参照

降順インデックス
一部のデータベースシステムで利用できるタイプのインデックス。ここでは ORDER BY column DESC 句を処理するようにインデックスストレージが最適化されます。現在のところ、MySQL は、CREATE TABLE ステートメントで DESC キーワードを許可しますが、結果のインデックスに特殊なストレージレイアウトを使用しません。
インデックスも参照

高位境界値
上限を表す値。実行時に超えるべきでないハード制限、または実際に到達した最大値の記録。低位境界値と対比してください。
低位境界値も参照

高速インデックス作成
関連付けられたテーブルを完全に書き直す必要性をなくすことによって InnoDB セカンダリインデックスの作成を高速化する、InnoDB Plugin に最初に導入されて現在 5.5 以降の MySQL Server に組み込まれている機能。高速化はセカンダリインデックスの削除にも適用されます。

インデックスを保守することで多数のデータ転送操作にパフォーマンスオーバーヘッドを増やす場合があるので、セカンダリインデックスを用意せずに ALTER TABLE ... ENGINE=INNODB や INSERT INTO ... SELECT * FROM ... などの操作を行い、あとでインデックスを作成することを検討してみてください。

MySQL 5.6 では、この機能はより一般的になっています。インデックスが作成されている間にテーブルに対する読み取りと書き込むを行うことができ、テーブルをコピーせずに、またはDML 操作をブロックせずに、あるいはその両方を行わずに、多種多様な ALTER TABLE 操作を実行できます。したがって、MySQL 5.6 以降では通常、この機能セットを高速インデックス作成ではなくオンライン DDL と呼んでいます。

関連情報については、セクション14.11「InnoDB とオンライン DDL」を参照してください。
DML, インデックス, オンライン DDL, セカンダリインデックスも参照

高速シャットダウン
構成設定 innodb_fast_shutdown=1 に基づいた、InnoDB のデフォルトシャットダウン手順。時間を節約するために、特定のフラッシュ操作がスキップされます。フラッシュ操作は次の起動中にクラッシュリカバリの場合と同じメカニズム

を使用して実行されるので、通常の使用中にはこのタイプのシャットダウンが安全です。アップグレードまたはダウングレードのためにデータベースをシャットダウンしている場合は、代わりに低速シャットダウンを行なって、すべての該当する変更がシャットダウン中にデータファイルに適用されることを保証してください。
クラッシュリカバリ, データファイル, フラッシュ, シャットダウン, 低速シャットダウンも参照

サ

サブリスト
バッファープールを表すリスト構造内では、比較的古いページと比較的新しいページがリストの異なる部分で表されます。パラメータセットは、これらの部分のサイズと、新しいページと古いページ間の分割ポイントを制御します。
バッファープール, エビクション, リスト, LRUも参照

サロゲートキー
合成キーのシノニム名。
合成キーも参照

サーバー
継続的に実行するタイプのプログラム。別のプログラム (クライアント) からリクエストを受信しそれに基づいて行動するのを待機します。多くの場合、コンピュータ全体が 1 つ以上のサーバープログラムを実行することを目的とするため (データベースサーバー、Web サーバー、アプリケーションサーバー、これらの組み合わせなど)、用語サーバーはサーバーソフトウェアを実行するコンピュータを指す場合もあります。
クライアント, mysqldも参照

先読み
これらのページがすぐに必要になると予想して、非同期的にページ (エクステント全体) のグループをバッファープールにプリフェッチするタイプの I/O リクエスト。線形先読み方法は、 1 エクステントのすべてのページを、先行するエクステント内のページのアクセスパターンに基づいてプリフェッチするもので、InnoDB Plugin for MySQL 5.1 で始まるすべての MySQL バージョンに組み込まれています。ランダム先読み方法は、エクステントから特定数のページがバッファープールに入ると、同じエクステントのすべてのページをプリフェッチするものです。ランダム先読みは、MySQL 5.5 には組み込まれていませんが、innodb_random_read_ahead 構成オプションの制御下で、MySQL 5.6 に再導入されています。
バッファープール, エクステント, ページも参照

削除
InnoDB が DELETE ステートメントを処理すると、行が即座に削除とマークされ、これ以降クエリーから返されなくなります。ストレージは以降の任意の時点で、別のスレッドによって実行される、パージ操作と呼ばれる定期的ガベージコレクション中に解放されます。大量のデータを削除する場合、独自のパフォーマンス特性を持つ関連操作は、切り捨ておよびドロップです。
ドロップ, パージ, 切り捨ても参照

削除バッファリング
DELETE 操作によるインデックス変更を即座に書き込むのではなく、挿入バッファーに格納して、物理的な書き込みを実行してランダム I/O を最小限に抑える方法。(削除操作は 2 ステッププロセスなので、この操作は、通常はインデックスレコードに削除とマークする書き込みをバッファーに入れます。)これは変更バッファリングの一種です。ほかのタイプには挿入バッファリングとパージバッファリングがあります。
変更バッファー, 変更バッファリング, 挿入バッファー, 挿入バッファリング, パージバッファリングも参照

参照整合性
常に一貫する形式でデータを保守する方法。ACID 概念の一部です。特に、異なるテーブル内のデータは外部キー制約の使用を通じて一貫性が保持され、これによって変更が発生するのを防止したり、それらの変更をすべての関連テーブルに自動的に伝播したりできます。関連メカニズムには、重複値が間違って挿入されるのを防ぐ一意制約と、ブランク値が間違って挿入されるのを防ぐ NOT NULL 制約が含まれます。
ACID, FOREIGN KEY 制約, NOT NULL 制約, 一意制約も参照

最大下限レコード
インデックス内の疑似レコードで、そのインデックスの最小値を下回るギャップを表します。トランザクションに SELECT ... FOR UPDATE ... WHERE col < 10; などのステートメントがあり、カラム内の最小値が 5 である場合、これは、ほかのトランザクションが 0 や -10 などのさらに小さな値を挿入するのを防ぐ最大下限レコードでのロックです。
ギャップ, インデックス, 疑似レコード, 最小上限レコードも参照

最小上限レコード
インデックス内の疑似レコードで、そのインデックスの最大値を上回るギャップを表します。トランザクションに SELECT ... FOR UPDATE ... WHERE col > 10; などのステートメントが含まれ、カラム内の最大値が 20 の場合、これは、ほかのトランザクションが 50 や 100 などのさらに大きな値を挿入することを防ぐ最小上限レコードに対するロックです。

ギャップ, 最大下限レコード, 疑似レコードも参照

シ

システムテーブルスペース
InnoDB テーブル関連オブジェクト (データディクショナリ) と Undo ログ、変更バッファー、二重書き込みバッファーのストレージ領域のメタデータを含むデータファイル (ibdata ファイル) の小さなセット。innodb_file_per_table の設定に応じて、テーブルの作成時に一部またはすべての InnoDB テーブルのテーブルとインデックスデータが含まれる場合もあります。システムテーブルスペース内のデータとメタデータは、MySQL インスタンス内のすべてのデータベースに適用されます。

MySQL 5.6.5 で導入されたリグレッションにより、innodb_file_per_table が有効のときに、FULLTEXT インデックステーブルは独自の個々のテーブルスペースではなく InnoDB システムテーブルスペース (スペース 0) に作成されます。このバグは、MySQL 5.6.20 および MySQL 5.7.5 で修正されました (Bug#18635485)。

MySQL 5.6.7 より前のデフォルトは、すべての InnoDB テーブルとインデックスをシステムテーブルスペース内部に保持することで、多くの場合、このファイルが非常に大きくなっていました。システムテーブルスペースは決して縮小しないので、大容量の一時データがロードされてから削除された場合、ストレージの問題が発生する可能性があります。MySQL 5.6.7 以降では、デフォルトは file-per-table モードで、各テーブルとそれに関連付けられたインデックスが別個の .ibd ファイルに格納されます。この新しいデフォルトによって、テーブル圧縮や DYNAMIC 行フォーマットなど、Barracuda ファイル形式に依存する InnoDB 機能をより簡単に使用できるようになります。

MySQL 5.6 以降では、innodb_undo_tablespaces オプションの値を設定すると、Undo ログが 1 つ以上の別個のテーブルスペースファイルに分割されます。これらのファイルは引き続きシステムテーブルスペースの一部と見なされます。

すべてのテーブルデータをシステムテーブルスペースまたは別個の .ibd ファイルのどちらに保持するかは、ストレージ管理全般に影響します。MySQL Enterprise Backup 製品は、大きなファイルの小さなセットをバックアップすることも、多数のより小さなファイルをバックアップすることもできます。数千のテーブルを持つシステムでは、数千の .ibd ファイルを処理するファイルシステム操作がボトルネックになることがあります。
Barracuda, 変更バッファー, 圧縮, データディクショナリ, データベース, 二重書き込みバッファー, 動的行フォーマット, file-per-table, .ibd ファイル, ibdata ファイル, innodb_file_per_table, インスタンス, MySQL Enterprise Backup, テーブルスペース, Undo ログも参照

シャットダウン
MySQL Server を停止させるプロセス。このプロセスはデフォルトで InnoDB テーブルのクリーンアップ操作を行うので、シャットダウンは遅くなりますが、その後の起動は速くなります。クリーンアップ操作を省略した場合、シャットダウンは速くなりますが、次の起動中にクリーンアップを行う必要があります。

シャットダウンモードは、innodb_fast_shutdown オプションで制御されます。
高速シャットダウン, InnoDB, 低速シャットダウン, 起動も参照

シャープチェックポイント
すべてのダーティーバッファープールページ (その Redo エントリが Redo ログのどこかに含まれている) をディスクにフラッシュするプロセス。InnoDB がログファイルの一部を再利用する前に発生します。ログファイルは循環的に使用されます。通常は、書き込みの多いワークロードで発生します。
ダーティーページ, フラッシュ, Redo ログ, ワークロードも参照

主キー
テーブル内のすべての行を一意に識別できるカラムセット (および暗黙的に、このカラムセットに基づくインデックス)。したがって、NULL 値を一切含まない一意インデックスである必要があります。

InnoDB は、すべてのテーブルにこのようなインデックス (クラスタ化されたインデックスまたはクラスタインデックスとも呼ばれます) があることを要求し、主キーのカラム値に基づいてテーブルストレージを編成します。

主キー値を選択するときは、ほかの何らかのソースから派生した値 (ナチュラルキー) に依存するのではなく、任意の値 (合成キー) を使用することを検討してください。
クラスタ化されたインデックス, インデックス, ナチュラルキー, 合成キーも参照

冗長行フォーマット
もっとも古い InnoDB 行フォーマット。Antelope ファイル形式を使用するテーブルに利用できます。MySQL 5.0.3 より前は、これが InnoDB で利用できる唯一の行フォーマットでした。My SQL 5.0.3 以降では、デフォルトはコンパクト行フォーマットです。古い InnoDB テーブルとの互換性のために、冗長行フォーマットを引き続き指定できます。

InnoDB REDUNDANT 行フォーマットの詳細は、セクション14.9.4「COMPACT および REDUNDANT 行フォーマット」を参照してください。
Antelope, コンパクト行フォーマット, ファイル形式, 行フォーマットも参照

準備されたバックアップ
バックアップファイルセットの 1 つ。バイナリログおよび増分バックアップを適用するすべての段階が終了したあとに、MySQL Enterprise Backup 製品によって生成されます。結果ファイルは、リストアできる状態です。適用ステップより前のファイルは raw バックアップと呼ばれます。
バイナリログ, ホットバックアップ, 増分バックアップ, MySQL Enterprise Backup, raw バックアップ, リストアも参照

自動インクリメント
カラム内に昇順で値を自動的に追加する、テーブルカラムのプロパティー (AUTO_INCREMENT キーワードで指定します)。InnoDB は主キーカラムについてのみ自動インクリメントをサポートします。

これにより、開発者の作業が節約され、新しい行を挿入するときに新しい一意値を生成する必要はありません。カラムには NULL でなく一意値が含まれているとわかっているので、クエリーオプティマイザに有用な情報をもたらします。このようなカラムからの値は、さまざまなコンテキストでルックアップキーとして使用でき、自動生成されるので絶えず変更する理由はありません。このため、多くの場合、主キーカラムは自動インクリメントとして指定されます。

自動インクリメントカラムは、ステートメントベースレプリケーションで問題になることがあります。スレーブでステートメントを再現しても、タイミングの問題のためにマスターと同じカラム値セットが生成されない場合があるためです。自動インクリメントする主キーがある場合は、innodb_autoinc_lock_mode=1 の設定だけでステートメントベースレプリケーションを使用できます。挿入操作でより高い並列性を許可する innodb_autoinc_lock_mode=2 を使用する場合は、ステートメントベースレプリケーションではなく行ベースレプリケーションを使用してください。innodb_autoinc_lock_mode=0 の設定は、以前 (従来) のデフォルト設定であり、互換性の目的以外では使用しないでください。
自動インクリメントロック, innodb_autoinc_lock_mode, 主キー, 行ベースレプリケーション, ステートメントベースレプリケーションも参照

自動インクリメントロック
自動インクリメント主キーの利便性には、並列性とのトレードオフが若干伴います。もっとも単純なケースでは、あるトランザクションがテーブルに値を挿入している場合に、ほかのトランザクションはそのテーブルへのそれぞれの挿入を待機する必要があるので、最初のトランザクションによって挿入された行が、連続する主キー値を受け取ります。InnoDB は最適化と innodb_autoinc_lock_mode オプションを含んでいるので、挿入操作について、自動インクリメント値の予測可能なシーケンスと最大の並列性とをトレードオフする方法を選択できます。
自動インクリメント, 並列性, innodb_autoinc_lock_modeも参照

自動コミット
各 SQL ステートメントのあとにコミット操作を実行する設定。このモードは、複数のステートメントにわたるトランザクションで InnoDB テーブルを操作する場合には推奨されません。これは、InnoDB テーブルでの読み取り専用トランザクションのパフォーマンスに役立つことがあります。この場合、特に MySQL 5.6.4 以降で、Undo データのロックおよび生成のオーバーヘッドを最小限に抑えます。これは、MyISAM テーブル (トランザクションが適用されない) を操作する場合にも適切です。
コミット, ロック, 読み取り専用トランザクション, SQL, トランザクション, Undoも参照

ス

スキーマ
概念的には、スキーマは、テーブル、テーブルカラム、カラムのデータ型、インデックス、外部キーなど、相互に関連するデータベースオブジェクトのセットです。カラムがテーブルを構成する、外部キーがテーブルやカラムを参照するなどの理由で、これらのオブジェクトは SQL 構文を通じて接続されます。理論的には、これらは論理的にも接続し、統合されたアプリケーションまたは柔軟なフレームワークの一部として連携します。たとえば、information_schema および performance_schema データベースは、テーブルとそこに含まれるカラム間の密接な関係を強調するために、その名前で「schema」を使用しています。

MySQL では、スキーマは物理的にデータベースと同義です。MySQL SQL 構文で、DATABASE の代わりにキーワード SCHEMA を代用できます。たとえば、CREATE DATABASE の代わりに CREATE SCHEMA を使用できます。

ほかのデータベース製品では区別しているものがあります。たとえば、Oracle Database 製品では、スキーマはデータベースの一部、つまり単一ユーザーが所有するテーブルおよびほかのオブジェクトだけを表します。
データベース, ib-file セット, INFORMATION_SCHEMA, パフォーマンススキーマも参照

スケーラビリティー
システム能力の限界を超えてもパフォーマンスが突然落ちることがなく、システムにより多くの作業を追加したりより多くの同時リクエストを発行したりできること。ソフトウェアアーキテクチャー、ハードウェア構成、アプリケーションコーディング、およびワークロードのタイプはすべて、スケーラビリティーで役割を担います。システムがその最大能力に達したときにスケーラビリティーを増やす一般的な方法には、スケールアップ (既存のハードウェアまたはソフトウェアの能力を増大させる) とスケールアウト (新しいサーバーやより多くの MySQL インスタンスを追加する) があります。多くの場合、大規模開発の重要な側面として、可用性と組み合わされます。
可用性, スケールアウト, スケールアップも参照

スケールアウト
新しいサーバーやより多くの MySQL インスタンスを追加することによってスケーラビリティーを増大させる方法。たとえば、レプリケーション、MySQL Cluster、接続プール、またはサーバーのグループに作業を分散させるほかの機能を設定すること。スケールアップと対比してください。
スケーラビリティー, スケールアップも参照

スケールアップ
既存のハードウェアまたはソフトウェアの能力を高めることによりスケーラビリティーを増大させる方法。たとえば、サーバー上でメモリーを増やすこと、innodb_buffer_pool_size や innodb_buffer_pool_instances などのメモリー関連パラメータを調整すること。スケールアウトと対比してください。
スケーラビリティー, スケールアウトも参照

ステミング
単数形と複数形、過去、現在、および未来時制など、共通語幹に基づいて単語のさまざまなバリエーションを検索する機能。この機能は現在、MyISAM 全文検索機能でサポートされますが、InnoDB テーブルの FULLTEXT インデックスではされません。
全文検索, FULLTEXT インデックスも参照

ステートメントベースレプリケーション
SQL ステートメントがマスターサーバーから送信され、スレーブサーバー上で再現される形式のレプリケーション。auto-increment locking の潜在的なタイミング問題を回避するために、innodb_autoinc_lock_mode オプションの設定には注意が必要です。
自動インクリメントロック, innodb_autoinc_lock_mode, マスターサーバー, レプリケーション, 行ベースレプリケーション, スレーブサーバーも参照

ストップワード
FULLTEXT インデックスで、十分に一般的または些細なので検索インデックスから除外し、検索クエリーで無視できると考えられている単語。InnoDB テーブルと MyISAM テーブルとでは、異なる構成設定がストップワード処理を制御します。詳細は、セクション12.9.4「全文ストップワード」を参照してください。
FULLTEXT インデックス, 検索インデックスも参照

ストレージエンジン
MySQL データベースのコンポーネントの 1 つ。データの格納、更新、および照会という低レベル作業を実行します。MySQL 5.5 以降では、InnoDB が新しいテーブルのデフォルトストレージエンジンであり、MyISAM の代替となるものです。メモリー使用とディスク使用、読み取り速度と書き込み速度、速度と堅牢性など、異なる要因間トレードオフのために異なるストレージエンジンが設計されています。各ストレージエンジンが特定のテーブルを管理するので、InnoDB テーブル、MyISAM テーブルなどと呼びます。

MySQL Enterprise Backup 製品は、InnoDB テーブルのバックアップに最適化されています。MyISAM およびほかのストレージエンジンで扱われるテーブルもバックアップできます。
InnoDB, MySQL Enterprise Backup, テーブルタイプも参照

スナップショット
特定時点のデータの表現。ほかのトランザクションによって変更がコミットされても変化しません。一貫性読み取りを許可する分離レベルで使用されます。
コミット, 一貫性読み取り, 分離レベル, トランザクションも参照

スピン
リソースが利用できるようになるかどうかを継続的にテストするタイプの待機操作。この方法は、通常短期間だけ保持されるリソースに使用されます。この場合、スレッドをスリープにしてコンテキストスイッチを実行するよりも、「ビジーループ」で待機するほうが効率的です。リソースが短時間で利用できなくなると、スピンループは中止し、別の待機方法が使用されます。
ラッチ, ロック, 相互排他ロック, 待機も参照

スペース ID
MySQL インスタンス内で InnoDB テーブルスペースを一意に識別するために使用される識別子。システムテーブルスペースのスペース ID は常にゼロです。この同じ ID がシステムテーブルスペース内のすべてのテーブルに適用されます。file-per-table モードで作成される各テーブルスペースファイルには、独自のスペース ID も割り当てられます。

MySQL 5.6 より前では、このハードコードされた値によって、MySQL インスタンス間で InnoDB テーブルスペースファイルを移動することが困難でした。MySQL 5.6 以降では、ステートメント FLUSH TABLES ... FOR EXPORT、ALTER TABLE ... DISCARD TABLESPACE、および ALTER TABLE ... IMPORT TABLESPACE を含むトランスポータブルテーブルスペース機能を使用することによって、インスタンス間でテーブルスペースファイルをコピーできます。スペース ID を調整するために必要な情報は、テーブルスペースとともにコピーする .cfg ファイルで伝えられます。詳細は、セクション14.5.5「テーブルスペースの別のサーバーへのコピー (トランスポータブルテーブルスペース)」を参照してください。

.cfg ファイル, file-per-table, .ibd ファイル, システムテーブルスペース, テーブルスペース, トランスポータブルテーブルスペースも参照

スレッド
通常はプロセスより軽量で、高度な並列性に対応できる処理単位。
並列性, マススレッド, プロセス, Pthreadsも参照

スレーブサーバー
多くの場合、「スレーブ」と短縮されます。レプリケーションシナリオのデータベースサーバーマシンの 1 つ。別のサーバー (マスター) から変更を受け取り、これらの同じ変更を適用します。したがって、ある程度遅れる可能性がありますが、マスターと同じ内容を保持します。

MySQL のスレーブサーバーは一般に、ディザスタリカバリで使用され、障害の発生したマスターサーバーに代わります。これらはまた一般に、データベース構成変更がパフォーマンスや信頼性の問題を起こさないことを保証するために、ソフトウェアアップグレードと新しい設定をテストする場合にも使用されます。

スレーブサーバーは通常、ユーザークエリーと同様に、マスターからリレーされるすべての DML (書き込み) 操作を処理するので、ワークロードが高くなります。スレーブサーバーがマスターからの変更を十分高速に適用できることを保証するために、それらは多くの場合、高速 I/O デバイスおよび十分な CPU とメモリーを備えて複数のデータベースインスタンスを同じスレーブサーバー上で実行します。たとえば、マスターサーバーがハードドライブストレージを使用し、スレーブサーバーが SSD を使用する場合があります。
DML, レプリケーション, サーバー, SSDも参照

スロークエリーログ
MySQL Server によって処理される SQL ステートメントのパフォーマンスチューニングに使用されるタイプのログ。ログ情報はファイルに格納されます。使用するにはこの機能を有効にする必要があります。どのカテゴリの「低速」 SQL ステートメントがログに記録されるかを制御してください。詳細は、セクション5.2.5「スロークエリーログ」を参照してください。
一般クエリーログ, ログも参照

セ

セカンダリインデックス
テーブルカラムのサブセットを表すタイプの InnoDB インデックス。InnoDB テーブルは、ゼロ個、1 個、または複数個のセカンダリインデックスを持つことができます。(クラスタ化されたインデックスと対比してください。こちらは各 InnoDB テーブルに必要で、すべてのテーブルカラムのデータを格納します。)

セカンダリインデックスは、インデックスカラムからの値だけを必要とするクエリーを満たすために使用できます。より複雑なクエリーの場合、テーブル内の該当行を識別するために使用でき、それらはクラスタ化されたインデックスを使用するルックアップで取得されます。

セカンダリインデックスの作成および削除には従来、InnoDB テーブル内のすべてのデータをコピーするためにかなりのオーバーヘッドが必要でした。InnoDB Plugin の高速インデックス作成機能によって、InnoDB セカンダリインデックスの CREATE INDEX および DROP INDEX ステートメントの速度が向上します。
クラスタ化されたインデックス, 高速インデックス作成, インデックスも参照

セグメント
InnoDB テーブルスペース内の境界。テーブルスペースをディレクトリに例えると、セグメントはそのディレクトリ内のファイルに似ています。セグメントは増えることができます。新しいセグメントを作成できます。

たとえば、file-per-table テーブルスペース内で、テーブルデータは 1 セグメント内にあり、それぞれに関連付けられたインデックスがそれが属するセグメント内にあります。システムテーブルスペースは、多くのテーブルとそれらに関連付けられたインデックスを保持できるため、多くの異なるセグメントを含みます。システムテーブルスペースには、Undo ログを構成する最大 128 のロールバックセグメントも含まれます。

セグメントは、データの挿入と削除に応じて拡大、縮小します。セグメントがより多くの領域を必要とする場合、一度に 1 エクステント (1M バイト) ずつ拡張されます。同様に、セグメントは、あるエクステント内のすべてのデータが不要になると、そのエクステント分の領域を解放します。
エクステント, file-per-table, ロールバックセグメント, システムテーブルスペース, テーブルスペース, Undo ログも参照

セーブポイント
セーブポイントは、ネストされたトランザクションの実装に役立ちます。それらは、Iより大きなトランザクションの一部であるテーブル上での操作にスコープを提供するために使用できます。たとえば、予約システムで旅行をスケジュールするときに、いくつかの異なる航空便を予約する場合があります。希望の航空便を利用できない場合、予約に成功したそれより早い航空便をロールバックすることなく、その 1 行程の予約に関与する変更をロールバックできます。

ロールバック, トランザクションも参照

全文検索
SQL LIKE 演算子を使用したり、アプリケーションレベル検索アルゴリズムを作成したりするよりも、より高速、より便利で、かつより柔軟な方法で単語、語句、単語のブール結合などをテーブルデータ内で検索するための MySQL 機能。これは、SQL 関数 MATCH() および FULLTEXT インデックスを使用します。
FULLTEXT インデックスも参照

制約
データが一貫性を失うのを防ぐために、データベース変更をブロックできる自動テスト。(コンピュータサイエンス用語では、不変条件に関連する一種のアサーション。)制約は、データ一貫性を維持するための、ACID 概念の重要な構成要素です。MySQL によってサポートされる制約には、FOREIGN KEY 制約と一意制約があります。
ACID, 外部キー, リレーショナル, 一意制約も参照

正規化
データベース設計戦略の 1 つ。データは複数のテーブルに分割されていて、圧縮された値を ID で表された単一行に複製することで、冗長または長い値を格納、照会、および更新することを回避します。通常は OLTP アプリケーションで使用されます。

たとえば、住所に一意 ID を与えることで、国勢調査データベースはその ID を家族の各メンバーに関連付けることによって、この住所の住居人という関係を表現できます (米国のある街のメインストリート 123 などの複雑な値のコピーを複数格納するのではなく)。

別の例としては、単純な住所録アプリケーションでは、ある人の名前および住所と同じテーブルにそれぞれの電話番号を格納しますが、電話会社データベースでは、各電話番号に特別な ID を与えてその番号と ID を別のテーブルに格納します。この正規化表現によって、市外局番が分かれるときの大幅な更新を簡略化できます。

正規化が常に推奨されるわけではありません。主に照会されるだけで、更新されるのは全体を削除してリロードする場合だけのデータは、多くの場合、重複値の冗長コピーを持つ少数の大きなテーブルで保持されます。このデータ表現は、非正規化と呼ばれ、データウェアハウスアプリケーションでよく見られます。
非正規化, 外部キー, OLTP, リレーショナルも参照

選択性
データ分布のプロパティーの 1 つ。カラム内の個別値の数 (その カーディナリティー) をテーブル内のレコード数で割ったもの。高い選択性は、カラム値が比較的一意であり、インデックスを通じて効率的に取得できることを意味します。WHERE 句内のテストがテーブル内の少ない数 (または割合) の行にのみ一致すると予測できる場合 (またはクエリーオプティマイザがそう予測する場合)、クエリーがインデックスを使用してそのテストを最初に評価すると、全体的に効率的である傾向があります。
カーディナリティー, クエリーも参照

静止
データベースアクティビティーの量を減らすこと。多くの場合、ALTER TABLE、バックアップ、シャットダウンなどの操作に備えるためです。最大限のフラッシュの実行を伴う場合があるため (そうでない場合もありますが)、InnoDB はバックグラウンド I/O の実行を継続しません。

MySQL 5.6 以降では、構文 FLUSH TABLES ... FOR EXPORT によって InnoDB テーブルの一部のデータがディスクに書き込まれるので、そのデータファイルをコピーすることでこれらのテーブルをより簡単にバックアップできます。
バックアップ, フラッシュ, InnoDB, シャットダウンも参照

ソ

ソートバッファー
InnoDB インデックスの作成中にデータをソートするために使用されるバッファー。ソートバッファーサイズは、innodb_sort_buffer_size 構成オプションを使用して構成されます。

増分バックアップ
ある時点以降に変更されたデータだけを保存する、MySQL Enterprise Backup 製品で実行されるタイプのホットバックアップ。完全バックアップと一連の増分バックアップがあれば、いくつかの完全バックアップを手元においておくストレージオーバーヘッドなしで、長期間にわたるバックアップデータを再構築できます。完全バックアップをリストアしてから各増分バックアップを連続して適用したり、各増分バックアップを完全バックアップに適用することでこれを最新の状態に保った状態で単一リストア操作を実行したりできます。

変更データの粒度はページレベルです。実際は 1 つのページが複数の行をカバーすることがあります。変更された各ページがバックアップに含まれます。
ホットバックアップ, MySQL Enterprise Backup, ページも参照

挿入
: SQL での主要な DML 操作の 1 つ。挿入のパフォーマンスは、データウェアハウスシステム (数百万行をテーブルにロードする) と OLTP システム (多数の並列接続が行を同じテーブルに任意の順序で挿入する可能性がある) で重要な要因です。挿入パフォーマンスが重要な場合は、変更バッファリングで使用される挿入バッファーや自動インクリメントカラムなどの InnoDB 機能を学習することをお勧めします。
自動インクリメント, 変更バッファリング, データウェアハウス, DML, InnoDB, 挿入バッファー, OLTP, SQLも参照

挿入バッファリング
: INSERT 操作によるセカンダリインデックス変更を即座に書き込むのではなく、挿入バッファーに格納することで、ランダム I/O を最小限に抑えるように物理書き込みを実行できる手法。これは変更バッファリングの 1 つのタイプです。ほかに削除バッファリングとパージバッファリングがあります。

セカンダリインデックスが一意の場合には挿入バッファリングは使用されません。新しいエントリが書き出される前に新しい値の一意性を検証できないためです。ほかの種類の変更バッファリングは一意インデックスに有効です。
変更バッファー, 変更バッファリング, 削除バッファリング, 挿入バッファー, パージバッファリング, 一意のインデックスも参照

挿入バッファー
: 変更バッファーの旧名。変更バッファリングには、挿入だけでなく削除および更新操作が含まれるので、「変更バッファー」が望ましい用語です。
変更バッファー, 変更バッファリングも参照

相互排他ロック
: 「相互排他ロック変数」の非公式な略語。(相互排他自体は「相互的な排他」の短縮です。)内部インメモリーデータ構造への排他的アクセスロックを表し、適用するために、InnoDB が使用する低レベルオブジェクト。ロックが獲得されると、ほかのプロセスやスレッドなどは同じロックを獲得できなくなります。読み書きロックと対比してください。これは、内部インメモリーデータ構造への共有アクセスロックを表し、適用するために、InnoDB が使用するものです。相互排他ロックと読み書きロックはまとめて、ラッチと呼ばれます。
ラッチ, ロック, パフォーマンススキーマ, Pthreads, rw ロック (読み書きロック)も参照

タ

タプル
: 順序付けられた要素セットを指定する技術用語。これは抽象概念で、データベース理論の公式ディスカッションで使用されます。データベースフィールドでのタプルは通常、テーブル行のカラムによって表されます。これらはクエリー (テーブルの一部のカラムだけ、または結合されたテーブルからのカラムを取得したクエリー、など) の結果セットで表される場合もあります。
カーソルも参照

ダーティーページ
: メモリー内で更新された、InnoDB バッファープール内のページ。ここでは、変更はまだデータファイルに書き込まれて (フラッシュされて) いません。クリーンページの反対です。
バッファープール, クリーンページ, データファイル, フラッシュ, ページも参照

ダーティー読み取り
: 信頼できないデータ、つまり別のトランザクションによって更新されたけれども、まだコミットされていないデータを取得する操作。これは、コミットされた読み取りと呼ばれる分離レベルでのみ可能です。

この種の操作は、データベース設計の ACID 原則には準拠しません。これは非常にリスクが高いと見なされます。データをロールバックできたり、コミットされる前にさらに更新できたりするためです。この場合、ダーティー読み取りを行うトランザクションは、正確であると確定されていないデータを使用することになります。

この正反対が一貫性読み取りです。InnoDB はあらゆる手を尽くして、トランザクションが別のトランザクションによって更新された情報を読み取らないことを保証します (別のトランザクションがその間にコミットした場合でも)。
ACID, コミット, 一貫性読み取り, 分離レベル, READ COMMITTED, READ UNCOMMITTED, ロールバックも参照

待機
: ロック、相互排他ロック、またはラッチを獲得するなどの操作を即座に完了できないとき、InnoDB は一時停止して再試行します。この一時停止のメカニズムはかなり入念に設計されているため、この操作には独自の名前、待機が付けられています。個々のスレッドは、内部 InnoDB スケジューリング、オペレーティングシステム wait() 呼び出し、および短期間スピンループの組み合わせを使用して一時停止されます。

重い負荷と多数のトランザクションが発生するシステムでは、SHOW INNODB STATUS コマンドからの出力を使用して、スレッドに非常に多くの時間を待機に消費しているかどうか、該当する場合は並列性をどのように改善できるかを判断できます。

並列性, ラッチ, ロック, 相互排他ロック, スピンも参照

チ

チェックサム
テーブルスペース内のページがディスクから InnoDB バッファープールに読み込まれるときに破損を検出する、InnoDB での検証メカニズム。innodb_checksums 構成オプションで、この機能のオンとオフが切り替わります。MySQL 5.6 では、構成オプション innodb_checksum_algorithm=crc32 も指定することによって、高速なチェックサムアルゴリズムを有効にできます。

innochecksum コマンドは、MySQL Server がシャットダウンする間に、指定のテーブルスペースファイルのチェックサム値をテストすることによって、破損の問題の診断に役立ちます。

MySQL はレプリケーションのためにチェックサムも使用します。詳細は、構成オプション binlog_checksum、master_verify_checksum、および slave_sql_verify_checksum を参照してください。
バッファープール, ページ, テーブルスペースも参照

チェックポイント
バッファープールにキャッシュされているデータページに変更が行われると、これらの変更は少しあとからデータファイルに書き込まれます。これはフラッシュと呼ばれるプロセスです。チェックポイントは、データファイルに正しく書き込まれている (LSN 値で表される) 最新の変更のレコードです。
バッファープール, データファイル, フラッシュ, ファジーチェックポイント, LSNも参照

テ

テキストコレクション
FULLTEXT インデックスに含まれるカラムセット。
FULLTEXT インデックスも参照

テーブル
各 MySQL テーブルは、特定のストレージエンジンに関連付けられます。InnoDB テーブルは、パフォーマンス、スケーラビリティー、バックアップ、管理、およびアプリケーション開発に影響する、特定の物理および論理特性を持っています。

ファイルストレージの観点では、各 InnoDB テーブルは、単一の大きな InnoDB システムテーブルスペースの一部か、またはテーブルが file-per-table モードで作成される場合は別個の .ibd ファイル内です。.ibd ファイルは、テーブルとそのすべてのインデックスのデータを保持し、テーブルスペースと呼ばれます。

file-per-table テーブルで作成される InnoDB テーブルは、Barracuda ファイル形式を使用できます。Barracuda テーブルは、DYNAMIC 行フォーマットまたは COMPRESSED 行フォーマットを使用できます。これらの比較的新しい設定は、圧縮、高速インデックス作成、オフページカラムなどのいくつかの InnoDB 機能を有効にします

MySQL 5.1 以前との後方互換性のため、システムテーブルスペース内部の InnoDB テーブルは、コンパクト行フォーマットと冗長行フォーマットをサポートする Antelope ファイル形式を使用する必要があります。

InnoDB テーブルの行は、テーブルの主キーカラムに基づいてエントリがソートされた、クラスタ化されたインデックスと呼ばれるインデックス構造に編成されます。データアクセスは主キーカラムでフィルタおよびソートするクエリーに最適化され、各インデックスには各エントリに関連付けられた主キーカラムのコピーが含まれます。主キーカラムの値を変更することは負荷の高い操作です。したがって、InnoDB テーブル設計の重要な側面は、もっとも重要なクエリーで使用されるカラムで主キーを選択し、主キーは短くし、値はめったに変更しないことです。
Antelope, バックアップ, Barracuda, クラスタ化されたインデックス, コンパクト行フォーマット, 圧縮行フォーマット, 圧縮, 動的行フォーマット, 高速インデックス作成, file-per-table, .ibd ファイル, インデックス, オフページカラム, 主キー, 冗長行フォーマット, 行, システムテーブルスペース, テーブルスペースも参照

テーブルの完全スキャン
インデックスを使用して選択した部分だけでなく、テーブルの内容全体を読み取る必要がある操作。通常は、小さなルックアップテーブル、またはすべての利用可能なデータが集約および分析される大きなテーブルを持つデータウェアハウジング状況で実行されます。これらの操作が行われる頻度と、利用可能なメモリーに関連するテーブルのサイズは、クエリー最適化とバッファープール管理で使用されるアルゴリズムに密接な関係があります。

インデックスの目的は、大きなテーブル内で特定の値または値の範囲をルックアップできるようにし、したがって有用なときはフルテーブルスキャンを回避することです。
バッファープール, インデックス, LRUも参照

テーブルスキャン
テーブルの完全スキャンも参照

テーブルスペース
１つ以上の InnoDB テーブルおよび関連付けられたインデックスのデータを保持できるデータファイル。システムテーブルスペースは、データディクショナリを構成するテーブルを含み、MySQL 5.6 以前ではデフォルトでほかの InnoDB テーブルをすべて保持します。innodb_file_per_table オプションをオンにすると (MySQL 5.6 以降のデフォルト)、新しく作成されるテーブルに独自のテーブルスペース、テーブルごとに別個のデータファイルが割り当てられます。

innodb_file_per_table オプションをオンにして複数のテーブルスペースを使用することは、テーブル圧縮やトランスポータブルテーブルスペースなどの多くの MySQL 機能を使用し、ディスク使用状況を管理することにとって非常に重要です。詳細は、セクション14.5.2「InnoDB File-Per-Table モード」を参照してください。

組み込み InnoDB ストレージエンジンで作成されるテーブルスペースは、InnoDB Plugin と上方互換です。InnoDB Plugin で作成されるテーブルスペースは、Antelope ファイル形式を使用する場合は、組み込み InnoDB ストレージエンジンと下方互換です。

MySQL Cluster はそのテーブルもテーブルスペースにグループ化します。詳細は、セクション18.5.12.1「MySQL Cluster ディスクデータオブジェクト」を参照してください。
Antelope, Barracuda, 圧縮行フォーマット, データディクショナリ, データファイル, file-per-table, インデックス, innodb_file_per_table, システムテーブルスペース, テーブルも参照

テーブルスペースディクショナリ
InnoDB テーブルスペース内で、テーブルのデータディクショナリメタデータの表現。テーブルが開かれるときにこのメタデータを .frm ファイルに対して一貫性をチェックすることで、古い .frm ファイルが原因のエラーを診断できます。この情報は、システムテーブルスペースの一部である InnoDB テーブルと、file-per-table オプションのために独自の .ibd ファイルを持つテーブルに提示されます。
データディクショナリ, file-per-table, .frm ファイル, .ibd ファイル, システムテーブルスペース, テーブルスペースも参照

テーブルタイプ
ストレージエンジンの古いシノニムです。InnoDB テーブル、MyISAM テーブルなどと呼びます。
InnoDB, ストレージエンジンも参照

テーブルロック
ほかのトランザクションがテーブルにアクセスすることを防ぐロック。InnoDB では、DML ステートメントとクエリーの処理にオンライン DDL、行ロック、一貫性読み取りなどの方法を使用することで、このようなロックを不要にするためにかなりの努力を割いています。SQL から LOCK TABLE ステートメントを使用してこのようなロックを作成できます。ほかのデータベースシステムまたは MySQL ストレージエンジンから移行するステップの 1 つは、可能なときはこのようなステートメントを削除することです。
一貫性読み取り, DML, ロック, ロック, オンライン DDL, クエリー, 行ロック, テーブル, トランザクションも参照

テーブル統計
統計も参照

ディスクバウンド
主なボトルネックがディスク I/O であるタイプのワークロード。(I/O バウンドとも呼ばれます。)通常は、ディスクへの頻繁な書き込みや、バッファープールに収められるよりも多くのデータのランダム読み取りがかかわります。
ボトルネック, バッファープール, CPU バウンド, ワークロードも参照

ディスクベース
主にディスクストレージ (ハードドライブまたはその同等物) 上のデータを編成するタイプのデータベース。データは、ディスクとメモリー間でやり取りされて操作されます。インメモリーデータベースの反対です。InnoDB はディスクベースですが、バッファープールなどの機能、複数のバッファープールインスタンス、および特定の種類のワークロードが主にメモリーから作業できるようにする適応型ハッシュインデックスも含まれます。
適応型ハッシュインデックス, バッファープール, インメモリーデータベースも参照

デッドロック
さまざまなトランザクションが進行できない状況 (それぞれが、他方が必要とするロックを保持しているため)。リソースが利用可能になるまで両方のトランザクションが待機しているため、どちらもそれが保持しているロックを解放しません。

(UPDATE や SELECT ... FOR UPDATE などのステートメントを通じて) トランザクションが複数のテーブル内の行を反対の順にロックすると、デッドロックが発生することがあります。デッドロックは、このようなステートメントがインデックスレコードとギャップの範囲をロックし、各トランザクションが一部のロックを取得するけれども、タイミングの問題によりほかを取得しない場合にも発生することがあります。

デッドロックの可能性を減らすには、LOCK TABLE ステートメントではなくトランザクションを使用したり、データを挿入または更新するトランザクションを長期間開かないでいいように小さくしたり、さまざまなトランザクションが複数のテーブルまたは大きな範囲の行を更新するときに、各トランザクションで同じ操作順序を使用したり (SELECT ... FOR

UPDATE など)、SELECT ... FOR UPDATE および UPDATE ... WHERE ステートメントで使用されるカラムにインデックスを作成したりしてください。デッドロックの可能性は、分離レベルに影響を受けません。分離レベルは読み取り操作の動作を変更し、デッドロックは書き込み操作のために発生するからです。

デッドロックが発生した場合、InnoDB は状況を検出し、いずれかのトランザクション (デッドロック対象) をロールバックします。したがって、アプリケーションロジックが完全に正しい場合でも、トランザクションを再試行する必要があるケースを扱う必要があります。InnoDB ユーザートランザクションでの最後のデッドロックを確認するには、コマンド SHOW ENGINE INNODB STATUS を使用してください。デッドロックが頻繁に発生して、トランザクション構造やアプリケーションエラー処理に問題があるらしいと思われる場合は、mysqld エラーログにすべてのデッドロックに関する情報を出力するために、innodb_print_all_deadlocks 設定を有効にした状態で実行してください。

自動的にデッドロックを検出して処理する方法に関する背景情報については、セクション14.2.10「デッドロックの検出とロールバック」を参照してください。デッドロック状況を回避しリカバリするためのヒントについては、セクション14.2.11「デッドロックの対処方法」を参照してください。
並列性, ギャップ, 分離レベル, ロック, ロック, ロールバック, トランザクション, デッドロック対象も参照

デッドロック対象
: デッドロックが検出されたときにロールバック対象として自動的に選択されるトランザクション。InnoDB は、更新した行数がもっとも少ないトランザクションをロールバックします。
デッドロック, デッドロック検出, innodb_lock_wait_timeoutも参照

デッドロック検出
: デッドロックが起きていることを自動的に検出し、関係するトランザクションのいずれか (デッドロック対象) を自動的にロールバックするメカニズム。
デッドロック, ロールバック, トランザクション, デッドロック対象も参照

データウェアハウス
: 主に大きなクエリーを実行するデータベースシステムまたはアプリケーション。読み取り専用または読み取りが大半のデータは、クエリーの効率を高めるために非正規化された形式で編成できます。MySQL 5.6 以降では、読み取り専用トランザクションの最適化からメリットを得ることができます。OLTP と対比してください。
非正規化, OLTP, クエリー, 読み取り専用トランザクションも参照

データディクショナリ
: テーブル、インデックス、テーブルカラムなどの InnoDB 関連オブジェクトを追跡するメタデータ。このメタデータは、InnoDB システムテーブルスペースに物理的に置かれています。これまでの経緯が理由で、これは .frm ファイルに格納された情報とある程度重複します。

MySQL Enterprise Backup 製品は常にシステムテーブルスペースをバックアップするため、すべてのバックアップにデータディクショナリの内容が含まれます。
カラム, .frm ファイル, ホットバックアップ, インデックス, MySQL Enterprise Backup, システムテーブルスペース, テーブルも参照

データディレクトリ
: それぞれの MySQL インスタンスが InnoDB 用のデータファイルを保持しているディレクトリと、個々のデータベースを表すディレクトリ。datadir 構成オプションによって制御されます。
データファイル, インスタンスも参照

データファイル
: 物理的に InnoDB テーブルおよびインデックスデータを含むファイル。システムテーブルスペース (データディクショナリと同様に複数の InnoDB テーブルを保持できる) の場合のように、データファイルとテーブルとの間に 1 対多関係が存在することがあります。file-per-table 設定が有効で、新しく作成する各テーブルが個別のテーブルスペースに格納されるときのように、データファイルとテーブルとの間に 1 対 1 関係が存在することもあります。
データディクショナリ, file-per-table, インデックス, システムテーブルスペース, テーブル, テーブルスペースも参照

データベース
: MySQL データディレクトリ内では、各データベースは個別のディレクトリで表されます。InnoDB システムテーブルスペース (MySQL インスタンス内で複数のデータベースからテーブルデータを保持できる) は、個々のデータベースディレクトリの外部に存在するそのデータファイルに保持されます。file-per-table モードが有効のときは、個々の InnoDB テーブルを表す .ibd ファイルがデータベースディレクトリ内部に格納されます。

長年 MySQL を使用している人にとって、データベースはなじみ深い概念です。Oracle Database のバックグラウンドを持つユーザーは、MySQL でのデータベースの意味は Oracle Database でスキーマと呼ばれているものに似ています。
データファイル, file-per-table, .ibd ファイル, インスタンス, スキーマ, システムテーブルスペースも参照

データ定義言語
: DDLも参照

データ操作言語
DMLも参照

低位境界値
下限を表す値。通常は、何らかの訂正アクションが始まったり、より積極的になったりするしきい値です。高位境界値と対比してください。
高位境界値も参照

低速シャットダウン
完了前に追加 InnoDB フラッシュ操作を行うタイプのシャットダウン。クリーンシャットダウンとも呼ばれます。構成パラメータ innodb_fast_shutdown=0 またはコマンド SET GLOBAL innodb_fast_shutdown=0; で指定されます。シャットダウン自体は時間がかかる可能性がありますが、その時間は後続の起動で節約されます。
クリーンシャットダウン, 高速シャットダウン, シャットダウンも参照

転置インデックス
ドキュメント検索システムに最適化され、InnoDB 全文検索の実装で使用されるデータ構造。転置インデックスとして実装される InnoDB FULLTEXT インデックスは、テーブル行の場所ではなく、ドキュメント内での各語の位置を記録します。単一カラム値 (テキスト文字列として格納されたドキュメント) は多くのエントリで転置インデックスで表現されます。
全文検索, FULLTEXT インデックス, ilistも参照

適応型ハッシュインデックス
メモリー内にハッシュインデックスを構築することにより、= および IN 演算子を使用したルックアップを高速化できる、InnoDB テーブルの最適化。MySQL は、InnoDB テーブルのインデックス検索をモニターし、ハッシュインデックスによりクエリーにメリットがある場合は、頻繁にアクセスされるインデックスページに対してこれを自動的に構築します。ある意味では、適応型ハッシュインデックスは、十分なメインメモリーを利用するように MySQL を実行時に構成するので、メインメモリーデータベースのアーキテクチャーに近づいています。この機能は、innodb_adaptive_hash_index 構成オプションで制御されます。この機能は、一部のワークロードにはメリットがあってもほかのものにはメリットがなく、ハッシュインデックスに使用されるメモリーはバッファープールで予約されているので、通常はこの機能を有効にした状態と無効にした状態でベンチマークを行うことをお勧めします。

ハッシュインデックスは常に、B ツリー構造として構成されている、既存の InnoDB セカンダリインデックスに基づいて構築されます。MySQL は、インデックスに対する検索パターンに応じて、B ツリーに定義された任意の長さのキーのプリフィクスに、ハッシュインデックスを構築できます。ハッシュインデックスは部分的であってもかまいません。B ツリーインデックス全体をバッファープールにキャッシュする必要はありません。

MySQL 5.6 以降では、InnoDB テーブルで高速な単一値ルックアップを利用するには、このほかに、InnoDB との memcached インタフェースを使用するという方法があります。詳細は、セクション14.18「InnoDB と memcached の統合」を参照してください。
B ツリー, バッファープール, ハッシュインデックス, memcached, ページ, セカンダリインデックスも参照

適応型フラッシュ
チェックポイントによって生じる I/O オーバーヘッドを軽減する InnoDB テーブル用のアルゴリズム。MySQL は、変更されたすべてのページをバッファープールからデータファイルに一度にフラッシュするのではなく、変更されたページの小さなセットを定期的にフラッシュします。適応型フラッシュアルゴリズムは、フラッシュの頻度と Redo 情報の生成速度に基づいてこれらの定期フラッシュの最適な実行頻度を見積もることによって、このプロセスを拡張します。最初は MySQL 5.1 で InnoDB Plugin に導入されました。
バッファープール, チェックポイント, データファイル, フラッシュ, InnoDB, ページ, Redo ログも参照

適用
MySQL Enterprise Backup 製品で生成されたバックアップに、バックアップ進行中に行われた最新の変更が含まれない場合、これらの変更を含むようにバックアップファイルを更新するプロセスは適用ステップと呼ばれます。これは mysqlbackup コマンドの apply-log オプションで指定されます。

変更が適用されるまでは、このファイルは raw バックアップと呼ばれます。変更が適用されたあとは、このファイルは準備されたバックアップと呼ばれます。変更は、ibbackup_logfile ファイルに記録されます。適用ステップが終了すると、このファイルは不要になります。
ホットバックアップ, ibbackup_logfile, MySQL Enterprise Backup, 準備されたバックアップ, raw バックアップも参照

ト

トラブルシューティング
InnoDB の信頼性とパフォーマンスの問題をトラブルシューティングするためのリソースには、情報スキーマテーブルが含まれます。

トランザクション
トランザクションは、作業の原子単位で、この単位でコミットまたはロールバックできます。トランザクションによってデータベースに複数の変更が行われた場合、トランザクションがコミットされるとすべての変更が完了し、トランザクションがロールバックされるとすべての変更が元に戻されます。

InnoDB が実装するデータベーストランザクションには、原始性、一貫性、分離性、持続性を表す頭字語 ACID で総称される特性があります。
ACID, コミット, 分離レベル, ロック, ロールバックも参照

トランザクション ID
各行に関連付けられた内部フィールド。このフィールドは、どのトランザクションがその行をロックしたかを記録するために、INSERT、UPDATE、および DELETE 操作によって物理的に変更されます。
暗黙の行ロックも参照

トランスポータブルテーブルスペース
テーブルスペースがあるインスタンスから別のインスタンスに移動されることを許可する機能。従来の InnoDB テーブルスペースではこれができませんでした。すべてのテーブルデータがシステムテーブルスペースの一部であったためです。MySQL 5.6 以降では、FLUSH TABLES ... FOR EXPORT 構文で別のサーバーにコピーできるように InnoDB テーブルを準備してから、ALTER TABLE ... DISCARD TABLESPACE および ALTER TABLE ... IMPORT TABLESPACE をほかのサーバー上で実行すると、コピーされたデータファイルがほかのインスタンスに移動します。テーブルスペースがインポートされるときに、別個の .cfg ファイルが、.ibd ファイルとともにコピーされ、テーブルメタデータ (たとえばスペース ID) の更新に使用されます。使用に関する情報は セクション14.5.5「テーブルスペースの別のサーバーへのコピー (トランスポータブルテーブルスペース)」を参照してください。
.ibd ファイル, スペース ID, システムテーブルスペース, テーブルスペースも参照

ドキュメント ID
InnoDB 全文検索機能における、各 ilist 値に関連付けられたドキュメントを一意に識別するための、FULLTEXT インデックスを含むテーブル内の特殊カラム。その名前は FTS_DOC_ID (大文字必須) です。カラム自体は、BIGINT UNSIGNED NOT NULL 型で、FTS_DOC_ID_INDEX という名前の一意インデックス付きである必要があります。テーブルの作成時にこのカラムを定義することが推奨されます。InnoDB が FULLTEXT インデックスの作成中にカラムをテーブルに追加する必要がある場合、インデックス作成操作の負荷が大幅に増大します。
全文検索, FULLTEXT インデックス, ilistも参照

ドロップ
DDL 操作の一種。DROP TABLE や DROP INDEX などのステートメントを通じてスキーマオブジェクトを削除します。これは内部的に ALTER TABLE ステートメントにマッピングします。InnoDB の観点からは、このような操作のパフォーマンス考慮事項としては、相互関連オブジェクトがすべて更新されるようにデータディクショナリをロックする時点と、バッファープールなどのメモリー構造を更新する時点があります。テーブルの場合、ドロップ操作には、切り捨て操作 (TRUNCATE TABLE ステートメント) と多少異なる特性があります。
バッファープール, データディクショナリ, DDL, テーブル, 切り捨てても参照

動的行フォーマット
InnoDB Plugin で導入された行フォーマットの 1 つ。Barracuda ファイル形式の一部として利用できます。行データを保持しているページの残りの外部に TEXT および BLOB フィールドが格納されるので、これはラージオブジェクトを含む行にとって非常に効率的です。ラージフィールドは通常、クエリー条件を評価するためにアクセスされることはないので、頻繁にはバッファープールに読み込まれません。その結果、I/O 操作は少なくなり、キャッシュメモリーの利用率が改善します。

InnoDB DYNAMIC 行フォーマットの追加情報については、セクション14.9.3「DYNAMIC および COMPRESSED 行フォーマット」を参照してください。
Barracuda, バッファープール, ファイル形式, 行フォーマットも参照

統計
各 InnoDB テーブルおよびインデックスに関連する評価値。効率的なクエリー実行計画の構築に使用されます。メイン値は、カーディナリティー (個別値の数) と、テーブル行またはインデックスエントリの合計数です。テーブルの統計は、その主キーインデックス内のデータを表します。セカンダリインデックスの統計は、このインデックスで扱われる行を表します。

値は正確なカウントではなく、見積もりです。あらゆる瞬間にさまざまなトランザクションが同じテーブルからの行を挿入したり削除したりしている可能性があるためです。値が頻繁に再計算されないように、永続的統計を有効にできます。この場合、値は InnoDB システムテーブルに格納され、ANALYZE TABLE ステートメントを発行するときにのみ更新されます。

innodb_stats_method 構成オプションで統計を計算するときに、NULL 値がどのように扱われるかを制御できます。

INFORMATION_SCHEMA および PERFORMANCE_SCHEMA テーブルを通じて、ほかのタイプの統計をデータベースオブジェクトおよびデータベースアクティビティーに利用できます。

カーディナリティー, インデックス, INFORMATION_SCHEMA, NULL, パフォーマンススキーマ, 永続的統計, 主キー, クエリー実行計画, セカンダリインデックス, テーブル, トランザクションも参照

ナ

ナチュラルキー
値が現実世界の何らかの意味を持つインデックスカラム (通常は主キー)。通常は、次の理由のため推奨されていません。

- 値が万が一変化した場合、クラスタ化されたインデックスを再ソートし、それぞれのセカンダリインデックスで繰り返される主キー値のコピーを更新するために、多数インデックス保守が必要になる可能性があります。
- 一見したところ安定した値でも、データベースで正しく表すことが難しい予測不可能な変化をすることがあります。たとえば、1 つの国が 2 つ以上に分かれ、元の国コードが古くなることがあります。または、一意値に関するルールに例外が発生する場合があります。たとえば、納税者 ID が単一の人物に一意であるように意図されている場合でも、データベースでは、ID 窃盗などでそのルールに違反するレコードを処理する必要があることがあります。また、納税者 ID やその他の機密 ID 番号からは、低品質の主キーが作成されます。それらはセキュリティー保護し、暗号化し、またはほかのカラムと異なる方法で扱う必要がある場合があるためです。

したがって通常は、自動インクリメントカラムを使用するなど、任意の数値を使用して合成キーを作成することをお勧めします。
自動インクリメント, 主キー, セカンダリインデックス, 合成キーも参照

ニ

二重書き込みバッファー
InnoDB は、二重書き込みと呼ばれる新しいファイルフラッシュ方法を使用します。ページをデータファイルに書き込む前に、InnoDB は最初に二重書き込みバッファーと呼ばれる連続領域に書き込みます。二重書き込みバッファーへの書き込みとフラッシュが完了したあとにはじめて、InnoDB はそれらのページをデータファイル内の適切な位置に書き込みます。ページ書き込みの最中にオペレーティングシステム、ストレージサブシステム、または mysqld プロセスのクラッシュが発生した場合、InnoDB は、あとでクラッシュリカバリ中にそのページの正常なコピーを二重書き込みバッファーから見つけることができます。

データは常に 2 度書き込まれますが、二重書き込みバッファーには、2 倍の I/O オーバーヘッドも 2 倍の I/O 操作も不要です。データは、オペレーティングシステムへの単一 fsync() 呼び出しで、大きなシーケンシャルチャンクとしてバッファー自体に書き込まれます。

二重書き込みバッファーをオフにするには、オプション innodb_doublewrite=0 を指定します。
クラッシュリカバリ, データファイル, ページ, パージも参照

ネ

ネクストキーロック
インデックスレコードでのレコードロックと、インデックスレコードの前のギャップでのギャップロックの組み合わせ。
ギャップロック, ロック, レコードロックも参照

ハ

ハッシュインデックス
大なりや BETWEEN などの範囲演算子ではなく、等値演算子を使用するクエリーを対象とするタイプのインデックス。MEMORY テーブルに利用できます。これまでの経緯が理由でハッシュインデックスは MEMORY テーブルのデフォルトですが、そのストレージエンジンは B ツリーインデックスもサポートします。こちらのほうが汎用目的のクエリーにとって適切な選択肢であることが多いです。

MySQL には、このインデックス型のバリアントである適応型ハッシュインデックスが含まれます。これは必要に応じて実行時の条件に基づいて InnoDB テーブル用に自動的に構築されます。
適応型ハッシュインデックス, B ツリー, インデックス, InnoDBも参照

ハートビート
システムが適切に機能していることを示すために送信される定期的メッセージ。レプリケーションコンテキストでは、マスターがこのようなメッセージの送信を停止した場合、スレーブの 1 つがそれに代わることができます。クラスタ環境内のすべてのサーバーが正しく動作していることを確認するために、それらの間で類似の方法を使用できます。
レプリケーションも参照

バイナリログ
テーブルデータを変更しようとするすべてのステートメントのレコードを含むファイル。レプリケーションシナリオでスレーブサーバーを最新にしたり、バックアップからテーブルデータをリストアしたあとでデータベースを最新にしたりするために、これらのステートメントを再現できます。バイナリロギング機能は、オンとオフを切り替えられますが、レプリケーションを使用したりバックアップを実行したりする場合は、常に有効にしておくことをお勧めします。

mysqlbinlog コマンドを使用することによって、バイナリログの内容を調べたり、レプリケーションまたはリカバリ中にこれらのステートメントを再現したりできます。バイナリログの詳細は、セクション5.2.4「バイナリログ」を参照してください。バイナリログに関連した MySQL 構成オプションについては、セクション17.1.4.4「バイナリログのオプションと変数」を参照してください。

MySQL Enterprise Backup 製品の場合、バイナリログのファイル名とファイル内での現在の位置が重要な詳細です。レプリケーションコンテキストでバックアップを取得するときにマスターサーバーに関するこの情報を記録するために、--slave-info オプションを指定できます。

MySQL 5.0 より前では、更新ログと呼ばれる同様の機能を利用できました。MySQL 5.0 以上では、更新ログがバイナリログに置き換わりました。
binlog, MySQL Enterprise Backup, レプリケーションも参照

バウンス
直後に再起動が行われるシャットダウン操作。パフォーマンスとスループットが迅速に高いレベルに戻るように、ウォームアップ期間が比較的短ければ理想的です。
シャットダウンも参照

バックアップ
保護するために、MySQL インスタンスから一部またはすべてのテーブルデータおよびメタデータをコピーするプロセス。コピーされたファイルのセットを指す場合もあります。これは DBA のきわめて重要なタスクです。このプロセスの反対がリストア操作です。

MySQL では、物理バックアップは MySQL Enterprise Backup 製品で実行され、論理バックアップは mysqldump コマンドで実行されます。これらの方法は、バックアップデータのサイズおよび表現と速度 (特にリストア操作の速度) の点で特性が異なります。

バックアップはさらに、通常のデータベース操作に干渉する程度に応じて、ホット、ウォーム、コールドに分類されます。(干渉の程度は、ホットバックアップがもっとも少なく、コールドバックアップがもっとも多くなります。)
コールドバックアップ, ホットバックアップ, 論理バックアップ, MySQL Enterprise Backup, mysqldump, 物理バックアップ, ウォームバックアップも参照

バッファー
一時ストレージに使用されるメモリーまたはディスク領域。多数の小さな I/O 操作ではなく少数の大きなものでディスクに効率的に書き込めるように、データはメモリーにバッファリングされます。データは、信頼性を高めるためにディスクにバッファリングされるので、考えられる最悪の場合にクラッシュやほかの障害が発生してもリカバリできます。InnoDB により使用されるバッファーの主なタイプは、バッファープール、二重書き込みバッファー、および挿入バッファーです。
バッファープール, クラッシュ, 二重書き込みバッファー, 挿入バッファーも参照

バッファープール
テーブルとインデックスの両方のキャッシュされた InnoDB データを保持するメモリー領域。大容量読み取り操作の効率を高めるため、バッファープールは複数行を保持できるページに分割されます。キャッシュ管理の効率のために、バッファープールはページのリンクリストとして実装されます。まれにしか使用されないデータは、LRU アルゴリズムのバリエーションを使用してキャッシュからエージアウトされます。大容量メモリーを備えたシステムでは、バッファープールを複数のバッファープールインスタンスに分割することにより、並列性を改善できます。

複数の InnoDB ステータス変数、information_schema テーブル、および performance_schema テーブルは、バッファープールの内部動作のモニターに役立ちます。MySQL 5.6 以降では、innodb_buffer_pool_dump_at_shutdown や innodb_buffer_pool_load_at_startup などの InnoDB 構成変数セットを通じて、シャットダウンおよび再起動中に自動的に、または随時手動で、バッファープールの内容をダンプおよびリストアすることもできます。
バッファープールインスタンス, LRU, ページ, ウォームアップも参照

バッファープールインスタンス
バッファープールは複数の領域に分割できますが、そのいずれか。innodb_buffer_pool_instances 構成オプションで制御されます。innodb_buffer_pool_size で指定された総メモリーサイズは、すべてのインスタンスに配分されます。複数のバッファープールインスタンスは通常、数 G バイトを InnoDB バッファープールに、1G バイト以上を各インスタンスに割り当てるシステムに適しています。多くの並列セッションからの大容量のデータをバッファープールにロードしそこで検索するシステム上では、複数のインスタンスがあれば、バッファープールを管理するデータ構造への排他的アクセスに対する競合が軽減します。
バッファープールも参照

バディーアロケータ
InnoDB バッファープールでさまざまなサイズのページを管理するためのメカニズム。
バッファープール, ページ, page sizeも参照

パフォーマンススキーマ
performance_schema スキーマは (MySQL 5.5 以降)、MySQL Server の多くの内部パーツのパフォーマンス特性に関する詳細情報を取得するために照会できる、テーブルセットを提供します。
ラッチ, 相互排他ロック, rw ロック (読み書きロック)も参照

パージ
別個のスレッドによって実行されるタイプのガベージコレクション。定期的スケジュールで実行されます。パージにはこれらのアクションが含まれます: インデックスから古い値を削除する、以前の DELETE ステートメントで削除とマークされた行を物理的に削除する。
クラッシュリカバリ, 削除, 二重書き込みバッファーも参照

パージスレッド
InnoDB プロセス内のスレッドの 1 つで、定期パージ操作を実行するためのもの。MySQL 5.6 以降では、複数のパージスレッドが innodb_purge_threads 構成オプションによって有効になっています。
パージ, スレッドも参照

パージバッファリング
DELETE 操作によるインデックス変更を即座に書き込むのではなく、挿入バッファーに格納して、物理的な書き込みを実行してランダム I/O を最小限に抑える方法。(削除操作は 2 ステッププロセスなので、この操作は、通常は以前に削除とマークされたインデックスレコードをパージする書き込みをバッファリングします。)これは変更バッファリングの一種です。ほかには挿入バッファリングと削除バッファリングがあります。
変更バッファー, 変更バッファリング, 削除バッファリング, 挿入バッファー, 挿入バッファリングも参照

パージ遅延
InnoDB 履歴リストの別の名前。innodb_max_purge_lag 構成オプションに関連しています。
履歴リスト, パージも参照

半一貫性読み取り
UPDATE ステートメントで使用されるタイプの読み取り操作。コミットされた読み取りと一貫性読み取りの組み合わせ。UPDATE ステートメントがすでにロックされている行を調べるとき、行が UPDATE の WHERE 条件に一致しているかどうかを MySQL が判断できるように、InnoDB は最後にコミットされたバージョンを MySQL に返します。行が一致する場合 (更新する必要がある場合)、MySQL は行を再度読み取り、InnoDB は今度はそれをロックするか、その口ックを待機します。このタイプの読み取り操作は、トランザクションの分離レベルがコミットされた読み取りで場合、または innodb_locks_unsafe_for_binlog オプションが有効である場合にのみ発生できます。
一貫性読み取り, 分離レベル, READ COMMITTEDも参照

反復不可能読み取り
あるクエリーがデータを取得し、同じトランザクション内のその後のクエリーが同じデータであるはずのものを取得するけれども、それらのクエリーが異なる結果を返す状況 (その間にコミットしている別のトランザクションによって変更された)。

この種の操作は、データベース設計の ACID 原則に反します。トランザクション内のデータは、予測可能で安定した関係を持ち、一貫しているべきです。

さまざまな分離レベルの中で、反復不可能読み取りは、シリアライズ可能読み取りと反復可能読み取りレベルによって防止され、一貫性読み取りとコミットされていない読み取りレベルで許可されます。
ACID, 一貫性読み取り, 分離レベル, READ UNCOMMITTED, REPEATABLE READ, SERIALIZABLE, トランザクションも参照

排他ロック
ほかのトランザクションが同じ行をロックするのを回避するタイプのロック。この種のロックは、トランザクション分離レベルに応じて、ほかのトランザクションが同じ行に書き込むのをブロックしたり、ほかのトランザクションが同じ行を読み取るのをブロックしたりできます。デフォルト InnoDB 分離レベル、REPEATABLE READ は、排他ロックを持つ行をトランザクションが読み取ることを許可する (一貫性読み取りと呼ばれる方法) ことによって、より高い並列性を実現します。
並列性, 一貫性読み取り, 分離レベル, ロック, REPEATABLE READ, 共有ロック, トランザクションも参照

破損ページ
I/O デバイス構成とハードウェア障害の組み合わせが原因で発生する可能性のあるエラー状況。データが InnoDB ページサイズ (デフォルトで 16K バイト) より小さなチャンクで書き出される場合、書き込み中のハードウェア障害によってページの一部だけがディスクに格納されることがあります。InnoDB 二重書き込みバッファーを使用することで、この可能性から保護されます。

二重書き込みバッファーも参照

ヒ

ビジネスルール
: 営利企業を運営するために使用される、ビジネスソフトウェアの基盤を形作るアクションの関係およびシーケンス。これらのルールは、法律によって規定されたり、企業ポリシーで規定されたります。慎重に計画することで、データベースでエンコードされ適用される関係と、アプリケーションロジックを通じて実行されるアクションが、企業の実際のポリシーを正確に反映し、現実の状況を扱うことができます。

  たとえば、従業員が会社を退職すると、人事部からアクションシーケンスがトリガーされます。人事データベースには、雇用されたけれどもまだ就業していない人物に関するデータを表すために、柔軟性も必要になることがあります。オンラインサービスで口座を閉鎖すると、データがデータベースから削除されたり、口座が再度開設されたりした場合にリカバリできるようにデータが移動またはフラグ付けされたります。企業は、給与が負数でないなどの基本的なサニティーチェックに加えて、給与の最大、最小、および調整に関するポリシーを確立できます。小売データベースでは、同じシリアル番号の購入を複数回返すことを禁止したり、一定値を超えるクレジットカード購入を禁止したりしますが、詐欺の検出に使用されるデータベースでは、このようなことを許可する場合があります。
  リレーショナルも参照

非ブロック I/O
: 非同期 I/Oと同じ意味を持つ業界用語。
  非同期 I/Oも参照

非ロック読み取り
: SELECT ... FOR UPDATE または SELECT ... LOCK IN SHARE MODE 句を使用しないクエリー。読み取り専用トランザクションでグローバルテーブルに許可される唯一の種類のクエリー。ロック読み取りの反対。
  ロック読み取り, クエリー, 読み取り専用トランザクションも参照

非同期 I/O
: I/O が完了するまでほかの処理の進行を許可する I/O 操作のタイプ。非ブロック I/O とも呼ばれ、AIO と略記されます。InnoDB は、実際にはリクエストされていないがすぐに必要になる可能性のあるページをバッファープールに読み取るなど、データベースの信頼性に影響を与えずに並列で実行できる操作にこのタイプの I/O を使用します。

  InnoDB は従来、Windows システムでのみ非同期 I/O を使用してきました。InnoDB Plugin 1.1 および MySQL 5.5 以降から、InnoDB は Linux システムで非同期 I/O を使用します。この変更により、libaio への依存関係がもたらされます。Linux システムでの非同期 I/O は、innodb_use_native_aio オプションを使用して構成されます。これはデフォルトで有効になっています。ほかの Unix 系システムでは、InnoDB は同期 I/O だけを使用します。
  バッファープール, 非ブロック I/Oも参照

非正規化
: 外部キーと結合クエリーでテーブルをリンクするのではなく、複数のテーブルにわたってデータを複製するデータストレージ戦略。通常はデータウェアハウスアプリケーションで使用されます。この場合、データはロード後に更新されません。このようなアプリケーションでは、更新中にデータの一貫性を維持することを簡略化するよりも、クエリーパフォーマンスが重要になります。正規化と対比してください。
  データウェアハウス, 正規化も参照

フ

ファイル形式
: InnoDB で各テーブルに使用される形式。通常は、各テーブルが個別の .ibd ファイルに格納されるように、file-per-table 設定が有効にされます。現在、InnoDB で利用できるファイル形式は Antelope および Barracuda と呼ばれています。各ファイル形式は、1 つ以上の行フォーマットをサポートします。Barracuda テーブルで利用できる行フォーマット、COMPRESSED および DYNAMIC は、InnoDB テーブルの新しい重要なストレージ機能を有効にします。
  Antelope, Barracuda, file-per-table, .ibd ファイル, ibdata ファイル, 行フォーマットも参照

ファジーチェックポイント
: データベース処理を妨害するダーティーページを一度にすべてフラッシュするのではなく、ダーティーページの小さなバッチをバッファープールからフラッシュする方法。
  バッファープール, ダーティーページ, フラッシュも参照

ファントム
: あるクエリーの結果セットに出現するけれども、以前のクエリーの結果セットにない行。たとえば、あるクエリーがトランザクション内で 2 度実行されて、その間に別のトランザクションがそのクエリーの WHERE 句に一致する新しい行を挿入または行を更新したあとにコミットされた場合です。

この現象がファントム読み取りと呼ばれます。このことから保護することは、反復不可能読み取りよりも困難です。最初のクエリー結果セットからのすべての行をロックしても、ファントムが出現する変更は防止されないためです。

さまざまな分離レベルの中で、ファントム読み取りは、シリアライズ可能読み取りで防止され、反復可能読み取り、一貫性読み取り、およびコミットされていない読み取りレベルで許可されます。
一貫性読み取り, 分離レベル, 反復不可能読み取り, READ UNCOMMITTED, REPEATABLE READ, SERIALIZABLE, トランザクションも参照

フィルファクタ
InnoDB インデックスにおいて、ページが分割される前にインデックスデータで占められるページの割合。ページ間でインデックスデータが最初に分割されるときの未使用領域によって、負荷のかかるインデックス保守操作を必要とすることなく、より長い文字列値で行を更新できます。フィルファクタが低すぎた場合、インデックスは必要以上の領域を消費し、インデックスを読み取るときに余分な I/O オーバーヘッドが生じます。フィルファクタが高すぎると、カラム値の長さが増える更新で、インデックス保守の追加 I/O オーバーヘッドが生じる可能性があります。詳細は、セクション 14.2.13.4「InnoDB インデックスの物理構造」を参照してください。
インデックス, ページも参照

フラッシュ
メモリー領域または一時ディスクストレージ領域にバッファリングされていた変更をデータベースファイルに書き込むこと。定期的にフラッシュされる InnoDB ストレージ構造には、Redo ログ、Undo ログ、およびバッファープールが含まれます。

フラッシュは、メモリー領域がいっぱいになってシステムが一部の領域を解放する必要があるため、コミット操作が、トランザクションからの変更を完結できることを意味するため、または低速シャットダウン操作が、すべての未処理作業を完結するべきであることを意味するため、行われます。バッファリングされているデータすべてを一度にフラッシュすることが重要でないときは、InnoDB は、ファジーチェックポイントという方法を使用して、ページの小さなバッチをフラッシュし、I/O オーバーヘッドを分散させることができます。
バッファープール, コミット, ファジーチェックポイント, 隣接ページ, Redo ログ, 低速シャットダウン, Undo ログも参照

フラッシュリスト
バッファープール内のダーティーページ (変更されて、ディスクに書き戻す必要があるページ) を追跡する内部 InnoDB データ構造。このデータ構造は、InnoDB の内部ミニトランザクションによって頻繁に更新されるため、バッファープールへの同時アクセスを許可するように独自の相互排他ロックで保護されています。
バッファープール, ダーティーページ, LRU, ミニトランザクション, 相互排他ロック, ページ, ページクリーナーも参照

ブラインドクエリー拡張
WITH QUERY EXPANSION 句で有効になる全文検索の特別なモード。これは検索を 2 度実行し、2 度目の検索には、最初の検索で検出されたドキュメントからのもっとも関連性の強い少数の単語をつなぎ合わせた、独自の検索語句を使用します。この方法は、主に短い検索語句、おそらく単一単語のみに適用されます。これは、正確な検索語句がドキュメント内で現れない場合に、関連した一致を見つけることができます。
全文検索も参照

プラグイン
MySQL 5.1 以前で、個別にインストールできる形式の InnoDB ストレージエンジン。これらのリリースに内蔵されている InnoDB に含まれない機能およびパフォーマンス拡張機能を含んでいます。

MySQL 5.5 以降の場合、MySQL 配布に InnoDB 1.1 と呼ばれる最新の InnoDB 機能およびパフォーマンス拡張機能が含まれ、個別の InnoDB Plugin は存在しません。

この区別は、主に MySQL 5.1 で重要です。ここでは、機能またはバグ修正が InnoDB Plugin に適用され、組み込み InnoDB にされない場合や、その反対の場合があります。
組み込み, InnoDBも参照

プリフィクス
インデックスプリフィクスも参照

プロセス
実行中プログラムのインスタンス。オペレーティングシステムは、複数の実行中プロセスを切り替えることで、一定程度の並列性を実現します。ほとんどのオペレーティングシステムのプロセスには、リソースを共有する複数の実行スレッドを含めることができます。スレッド間のコンテキスト切り替えは、プロセス間の同等切り替えより高速です。
並列性, スレッドも参照

分離レベル
データベース処理の基礎の 1 つ。分離は、頭字語 ACID の I です。分離レベルは、複数のトランザクションが同時に変更を行ったりクエリーを実行したりしているときに、パフォーマンスと信頼性のバランス、一貫性、結果の再現性を微調整する設定です。

もっとも高い一貫性および保護からもっとも低いものまで、InnoDB でサポートされる分離レベルは、SERIALIZABLE、REPEATABLE READ、READ COMMITTED、および READ UNCOMMITTED です。

InnoDB テーブルの多くのユーザーは、すべての操作にデフォルト分離レベル (REPEATABLE READ) を保持できます。上級ユーザーは、OLTP 処理でスケーラビリティーの境界をプッシュするとき、または小さな不整合が大量のデータの集計結果に影響しないデータウェアハウス操作中に、コミットされた読み取りレベルを選択できます。両端のレベル (SERIALIZABLE および READ UNCOMMITTED) は、処理動作をまれにしか使用されない程度に変更します。
ACID, READ COMMITTED, READ UNCOMMITTED, REPEATABLE READ, SERIALIZABLE, トランザクションも参照

物理
ディスクブロック、メモリーページ、ファイル、ビット、ディスク読み取りなど、ハードウェア関連側面がかかわるタイプの操作。物理側面は通常、上級者レベルのパフォーマンスチューニングおよび問題診断中に重要になります。論理と対比してください。
論理, 物理バックアップも参照

物理バックアップ
実際のデータファイルをコピーするバックアップ。たとえば、MySQL Enterprise Backup 製品の mysqlbackup コマンドは物理バックアップを返します。その出力に mysqld サーバーが直接使用できるデータファイルが含まれているためです (リストア操作の速度が向上します)。論理バックアップと対比してください。
バックアップ, 論理バックアップ, MySQL Enterprise Backup, リストアも参照

複合インデックス
複数のカラムを含むインデックス。
インデックス, インデックスプリフィクスも参照

部分インデックス
カラム値の一部 (通常は、長い VARCHAR 値の最初の N 文字 (プリフィクス) だけを表す) インデックス。
インデックス, インデックスプリフィクスも参照

部分バックアップ
MySQL データベースのテーブルの一部、または MySQL インスタンスのデータベースの一部を含むバックアップ。完全バックアップと対比してください。
バックアップ, 完全バックアップ, テーブルも参照

へ

ベータ
ソフトウェア製品の存続期間の初期段階。評価にのみ利用できる期間で、通常は確定したリリース番号や 1 未満の数がありません。InnoDB ではベータの名称を使用せず、複数のポイントリリースに進展し GA リリースに至るアーリーアダプタフェーズを使用します。
アーリーアダプタ, GAも参照

ペシミスティック
パフォーマンスまたは並列性を犠牲にして、安全性を優先する概念。リクエストまたは試行が高い割合で失敗する可能性がある場合、または失敗したリクエストの結果が深刻である場合に適しています。InnoDB は、ペシミスティックロック戦略と呼ばれるものを使用して、デッドロックの可能性を最小限に抑えます。アプリケーションレベルでは、トランザクションが必要とするすべてのロックを最初に獲得するペシミスティック戦略を使用することによって、デッドロックを回避できる可能性があります。

多くのデータベースに組み込まれているメカニズムでは、反対のオプティミスティック概念が使用されます。
デッドロック, ロック, オプティミスティックも参照

ページ
InnoDB がディスク (データファイル) とメモリー (バッファープール) 間で一度に転送するデータ量を表す単位。ページには 1 つ以上の行を含めることができます (各行のデータ量に依存)。行全体が単一ページに収まらない場合、InnoDB はその行に関する情報を 1 ページに格納できるように、追加ポインタスタイルデータ構造を設定します。

各ページにより多くのデータを収める方法の 1 つは、圧縮行フォーマットを使用ことです。BLOB またはラージテキストフィールドを使用するテーブルの場合、コンパクト行フォーマットを使用してこれらのラージカラムを残りの行とは別個に格納することで、これらのカラムを参照しないクエリーの I/O オーバーヘッドとメモリー使用量を減らすことができます。

InnoDB は、I/O スループットを増やすためにページセットをバッチとして読み書きするときは、一度に 1 エクステントを読み書きします。

MySQL インスタンス内のすべての InnoDB ディスクデータ構造は、同じページサイズを共有します。
バッファープール, コンパクト行フォーマット, 圧縮行フォーマット, データファイル, エクステント, page size, 行も参照

ページクリーナー
InnoDB バックグラウンドスレッドの 1 つ。ダーティーページをバッファープールからフラッシュします。MySQL 5.6 より前では、このアクティビティーはマスタースレッドによって実行されていました
バッファープール, ダーティーページ, フラッシュ, マスタースレッド, スレッドも参照

並列性
複数の操作 (データベース用語では、トランザクション) が互いに干渉することなく同時に実行できること。並列性はパフォーマンスにも関係しています。理論的には、ロックの効率的なメカニズムを使用して、複数の同時トランザクションの保護が最小のパフォーマンスオーバーヘッドで機能するためです。
ACID, ロック, トランザクションも参照

変更バッファリング
変更バッファーを使用する機能の汎用用語で、挿入バッファリング、削除バッファリング、およびパージバッファリングから構成されます。SQL ステートメントから生じるインデックス変更は、通常はランダム I/O 操作を使用し、バックグラウンドスレッドによって保持されて定期的に実行されます。この操作シーケンスは、値が即座にディスクに書き込まれる場合よりも効率的に、一連のインデックス値のディスクブロックを書き込むことができます。innodb_change_buffering および innodb_change_buffer_max_size 構成オプションによって制御されます。
変更バッファー, 削除バッファリング, 挿入バッファリング, パージバッファリングも参照

変更バッファー
セカンダリインデックス内のページへの変更を記録する特殊なデータ構造。これらの値は、SQL INSERT、UPDATE、または DELETE ステートメント (DML) の結果として発生する可能性があります。変更バッファーを使用する一連の機能は、まとめて変更バッファリングと呼ばれ、挿入バッファリング、削除バッファリング、およびパージバッファリングから構成されています。

セカンダリインデックスからの関連ページがバッファープールに存在しない場合、変更は変更バッファーにのみ記録されます。関連付けられた変更が変更バッファー内にまだあるときに該当するインデックスページがバッファープールに読み取られた場合、そのページに関する変更は、変更バッファーからのデータを使用してバッファープールに適用 (マージ) されます。システムがほとんどアイドル状態になっているとき、または低速シャットダウン中に実行するパージ操作は、定期的に新しいインデックスページをディスクに書き込みます。パージ操作は、それぞれの値を即座にディスクに書き込む場合よりも効率的に、一連のインデックス値のディスクブロックを書き込むことができます。

物理的に変更バッファーは、システムテーブルスペースの一部なので、インデックス変更はデータベースの再起動をまたがってバッファーに残ります。変更は、ほかの読み取り操作によってページがバッファープールに読み取られたときにのみ、適用 (マージ) されます。

変更バッファーに格納されたデータの種類と容量は、innodb_change_buffering および innodb_change_buffer_max_size 構成オプションで制御されます。変更バッファー内の現在のデータに関する情報を確認するには、SHOW ENGINE INNODB STATUS コマンドを発行してください。

以前には挿入バッファーと呼ばれていました。
バッファープール, 変更バッファリング, 削除バッファリング, DML, 挿入バッファー, 挿入バッファリング, マージ, ページ, パージ, パージバッファリング, セカンダリインデックス, システムテーブルスペースも参照

ホ

ホット
行、テーブル、または内部データ構造が非常に頻繁にアクセスされ、何らかの形式のロックまたは相互排他が必要になり、結果としてパフォーマンスまたはスケーラビリティーの問題が発生する状況。

「ホット」は一般に望ましくない状態を示しますが、ホットバックアップは優先されるタイプのバックアップです。
ホットバックアップも参照

ホットバックアップ
データベースが実行中で、アプリケーションがそれに対して読み取りおよび書き込みを行なっている間に取得されるバックアップ。バックアップはデータファイルを単純にコピーするだけではありません。バックアップが進行していた間に挿入または更新されたデータを含める必要があり、バックアップが進行していた間に削除されたデータを排除する必要があり、コミットされなかった変更を無視する必要があります。

InnoDB テーブルだけでなく、特に MyISAM およびほかのストレージエンジンからのテーブルのホットバックアップも実行する Oracle 製品は、MySQL Enterprise Backup として知られています。

ホットバックアッププロセスは 2 つのステージから構成されます。データファイルの初期コピーは raw バックアップを生成します。適用ステップにより、バックアップの実行中に行われたデータベースへの変更が組み込まれます。変更の適用により、準備されたバックアップが生成されます。これらのファイルは、必要な場合はいつでもリストアできる状態です。

適用, MySQL Enterprise Backup, 準備されたバックアップ, raw バックアップも参照

ボトルネック
システム内で、全体的なスループットを制限する影響を持つ、サイズまたは容量に制約がある部分。たとえば、メモリー領域が必要な容量に満たないことがありますが、この場合、必要な単一リソースにアクセスするだけで、複数の CPU コアが同時に実行できなくなったり、ディスク I/O が完了するまで待機することで、CPU がフル稼働できなくなったりする可能性があります。ボトルネックを取り除くと、並列性が改善する傾向があります。たとえば、複数の InnoDB バッファープールインスタンスを持つことができれば、複数のセッションが同時にこのバッファープールに対して読み取り/書き込みするときの競合が軽減します。
バッファープール, 並列性も参照

ポイントインタイムリカバリ
特定日時のデータベースの状態を再作成するためにバックアップをリストアするプロセス。一般に PITR と略記されます。指定した時間がバックアップの時点に正確に対応する可能性は低いので、この方法は通常、物理バックアップおよび論理バックアップと組み合わせる必要があります。たとえば、MySQL Enterprise Backup 製品では、指定した特定の時点より前に取得した最後のバックアップをリストアしてから、そのバックアップ時点から PITR 時点までのバイナリログから変更を再現します。
バックアップ, 論理バックアップ, MySQL Enterprise Backup, 物理バックアップ, PITRも参照

マ

マスターサーバー
よく「マスター」と短縮されます。レプリケーションシナリオでのデータベースサーバーマシンで、データの初期挿入、更新、および削除リクエストを処理します。これらの変更は、スレーブサーバーと呼ばれるほかのサーバーに伝播されて、再現されます。
レプリケーション, スレーブサーバーも参照

マススレッド
バックグラウンドでさまざまなタスクを実行する InnoDB スレッド。これらのタスクのほとんどは、挿入バッファーからの変更を適切なセカンダリインデックスに書き込むなど、I/O 関連です。

並列性を改善するために、アクションがマススレッドから個別のバックグラウンドスレッドに移される場合があります。たとえば、MySQL 5.6 以降では、ダーティーページは、マススレッドではなくページクリーナースレッドで、バッファープールからフラッシュされます。
バッファープール, ダーティーページ, フラッシュ, 挿入バッファー, ページクリーナー, スレッドも参照

マルチコア
MySQL Server などのマルチスレッドプログラムを活用できるタイプのプロセッサ。

マルチバージョン並列性制御
MVCCも参照

マージ
ページがバッファープールに読み込まれたときや、変更バッファーに記録された適用可能な変更がバッファープール内のページに組み込まれたときなどに、メモリーにキャッシュされたデータに変更を適用すること。更新されたデータは最終的に、フラッシュメカニズムによってテーブルスペースに書き込まれます。
バッファープール, 変更バッファー, フラッシュ, テーブルスペースも参照

ミ

ミッドポイント挿入戦略
InnoDB バッファープールリストの「最新の」末尾ではなく、中間のどこかにページを最初に読み込ませる方法。このポイントの正確な位置は、innodb_old_blocks_pct オプションの設定に基づいて変わります。その意図は、フルテーブルスキャン中などに一度だけ読み取られるブロックが、厳密な LRU アルゴリズムを使用した場合よりも早くバッファープールからエージアウトできるということです。
バッファープール, テーブルの完全スキャン, LRU, ページも参照

ミニトランザクション
DML 操作中に内部データ構造に物理レベルで変更を行うときの、InnoDB 処理の内部フェーズ。ミニトランザクション (mtr) にはロールバックの概念はありません。単一トランザクション内で複数のミニトランザクションを発生できます。ミニトランザクションは、クラッシュリカバリ中に使用される Redo ログに情報を書き込みます。ミニトランザクションは、たとえばバックグラウンドスレッドによるパージ処理中など、通常のトランザクションのコンテキスト外でも発生できます。
コミット, クラッシュリカバリ, DML, 物理, パージ, Redo ログ, ロールバック, トランザクションも参照

メ

メタデータロック
別のトランザクションによって同時に使用されているテーブルでの DDL 操作を防止するタイプのロック。詳細は、セクション8.10.4「メタデータのロック」を参照してください。

オンライン操作への拡張機能は (特に MySQL 5.6 以降)、メタデータロックの量を減らすことに注力しています。その目標は、ほかのトランザクションによってテーブルに照会や更新などが行われている間に、テーブル構造を変更しない DDL 操作 (InnoDB テーブルに対する CREATE INDEX や DROP INDEX など) を進行できるようにすることです。
DDL, ロック, オンライン, トランザクションも参照

メトリックカウンタ
MySQL 5.6 以降で、information_schema の innodb_metrics テーブルによって実装された機能。低レベル InnoDB 操作に関するカウントおよび合計を照会し、その結果を performance_schema からのデータと組み合わせてパフォーマンスチューニングに使用できます。
カウンタ, INFORMATION_SCHEMA, パフォーマンススキーマも参照

ユ

行
カラムセットによって定義される論理データ構造。行セットがテーブルを構成します。InnoDB データファイル内で、各ページには 1 つまたは複数の行を含められます。

InnoDB は MySQL 構文との一貫性のために用語行フォーマットを使用しますが、行フォーマットは各テーブルのプロパティーであり、そのテーブル内のすべての行に適用されます。
カラム, データファイル, ページ, 行フォーマット, テーブルも参照

行フォーマット
InnoDB テーブルからの行のディスクストレージフォーマット。InnoDB に圧縮などの新しい機能が用意されるのに伴い、ストレージの効率性とパフォーマンスの向上をサポートするために新しい行フォーマットが導入されます。

テーブルごとに独自の行フォーマットがあり、ROW_FORMAT オプションを通じて指定されます。各 InnoDB テーブルの行フォーマットを確認するには、コマンド SHOW TABLE STATUS を発行します。システムテーブルスペース内のすべてのテーブルが同じ行フォーマットを共有するので、ほかの行フォーマットを活用するには通常、各テーブルは個別のテーブルスペースに格納されるように innodb_file_per_table オプションを設定する必要があります。
コンパクト行フォーマット, 圧縮行フォーマット, 動的行フォーマット, 固定行フォーマット, 冗長行フォーマット, 行, テーブルも参照

行レベルロック
InnoDB テーブルに使用されるロックメカニズム。テーブルロックではなく行ロックに依存します。複数のトランザクションが同時に同じテーブルを変更できます。2 つのトランザクションが同じ行を変更しようとした場合にのみ、トランザクションの一方は他方が終わる (およびその行ロックを解放する) まで待機します。
InnoDB, ロック, 行ロック, テーブルロック, トランザクションも参照

ヨ

読み取りビュー
InnoDB の MVCC メカニズムで使用される内部スナップショット。ある種のトランザクションは、その分離レベル に応じて、トランザクション (または、場合によってはステートメント) が開始した時点のデータ値を見ます。読み取りビューを使用する分離レベルは、REPEATABLE READ、READ COMMITTED、および READ UNCOMMITTED です。
分離レベル, MVCC, READ COMMITTED, READ UNCOMMITTED, REPEATABLE READ, トランザクションも参照

読み取り専用トランザクション
トランザクションごとの読み取りビューを作成することに関係する管理の一部を省略することによって、InnoDB テーブルに最適化できるタイプのトランザクション。非ロック読み取りクエリーだけを実行できます。構文 START TRANSACTION READ ONLY で明示的に開始したり、特定の条件下で自動的に開始したりできます。詳細は、セクション14.13.14「InnoDB の読み取り専用トランザクションの最適化」を参照してください。
非ロック読み取り, 読み取りビュー, トランザクションも参照

ラ

ラッチ
独自の内部メモリー構造にロックを実装するために InnoDB で使用される軽量構造で、通常はミリ秒またはマイクロ秒単位で計測される短い期間保持されます。相互排他ロック (排他アクセスの場合) と読み書きロック (共有アクセスの場

合) の両方を含む一般用語。ある種のラッチは、データディクショナリ相互排他ロックなど、InnoDB パフォーマンスチューニングにとって重要です。ラッチの使用と競合に関する統計は、パフォーマンススキーマインタフェースから入手できます。
データディクショナリ, ロック, ロック, 相互排他ロック, パフォーマンススキーマ, rw ロック (読み書きロック)も参照

ランダムダイブ
カラム内のさまざまな値の数 (カラムのカーディナリティー) をすばやく見積もる方法。InnoDB は、インデックスからランダムにページをサンプリングし、そのデータを使用してさまざまな値の数を見積もります。この操作は、各テーブルが最初に開かれるときに発生します。

従来は、サンプリングされるページ数は 8 に固定されていました。現在は、innodb_stats_sample_pages パラメータの設定によって決定されます。

ランダムページが選択される方法は、innodb_use_legacy_cardinality_algorithm パラメータの設定に応じて変わります。デフォルト設定 (OFF) のランダム性は古いリリースのものより向上しています。
カーディナリティーも参照

リ

リスト
InnoDB バッファープールは、メモリーページのリストとして表されます。リストは、新しいページがアクセスされてバッファープールに読み込まれたとき、バッファープール内のページが再度アクセスされてより新しいと見なされたとき、長時間アクセスされていないページがバッファープールから削除されたときに、並べ替えられます。バッファープールは、実際にはサブリストに分割され、その置換ポリシーはよく知られた LRU 手法のバリエーションです。
バッファープール, エビクション, LRU, サブリストも参照

リストア
MySQL Enterprise Backup 製品からのバックアップファイルセットを MySQL で使用できるように準備するプロセス。この操作は、破損したデータベースを修復して、以前のどこか特定の時点に戻したり、(レプリケーションコンテキストで) 新しいスレーブデータベースを設定したりするために実行できます。MySQL Enterprise Backup 製品では、この操作は mysqlbackup コマンドの copy-back オプションで実行されます。
ホットバックアップ, MySQL Enterprise Backup, mysqlbackup コマンド, 準備されたバックアップ, レプリケーションも参照

リレーショナル
現代のデータベースシステムの重要な側面。データベースサーバーは、1 対 1、1 対多、多対 1、一意性などの関係を工ンコードし適用します。たとえば、住所データベースでは、ある人にゼロ個、1 個、または複数個の電話番号が関連付けられていたり、単一電話番号が家族の複数のメンバーに関連付けられていたりします。金融データベースでは、1 人の人に正確に 1 つの納税者 ID が割り当てられる必要があり、納税者 ID は 1 人の人にのみ関連付けることができます。

データベースサーバーは、これらの関係を使用して、不良データが挿入されるのを防ぎ、情報をルックアップする効率的な方法を見つけることができます。たとえば、値が一意であると宣言されている場合、サーバーは、最初の一致が見つかるとすぐに検索を停止し、同じ値の 2 番目のコピーを挿入しようとする試みを拒否できます。

データベースレベルではこれらの関係は、テーブル内のカラム、一意および NOT NULL 制約、外部キー、さまざまな種類の結合操作などの SQL 機能を通じて表されます。複雑な関係には通常、複数のテーブル間で分割されたデータが使用されます。多くの場合、データは正規化されるので、1 対多の関係での重複値は一度だけ格納されます。

数学的なコンテキストでは、データベース内の関係は集合論から派生されます。たとえば、WHERE 句の OR および AND 演算子は、論理和と論理積の概念を表します。
ACID, 制約, 外部キー, 正規化も参照

履歴リスト
削除マークが付けられたレコードが InnoDB パージ操作で処理されるようにスケジュールされているトランザクションのリスト。Undo ログに記録されます。履歴リストの長さはコマンド SHOW ENGINE INNODB STATUS で報告されます。履歴リストが innodb_max_purge_lag 構成オプションの値よりも長くなった場合、パージ操作が削除済みレコードのフラッシュを完了できるように各 DML 操作が少し遅くなります。

パージラグとも呼ばれます。
フラッシュ, パージ, パージ遅延, ロールバックセグメント, トランザクション, Undo ログも参照

隣接ページ
特定のページと同じエクステント内のページ。ページがフラッシュの対象として選択されると、ダーティーである隣接ページがある場合は、通常はそれらも従来ハードディスクの I/O 最適化としてフラッシュされます。MySQL 5.6 以降では、この動作は構成変数 innodb_flush_neighbors で制御できます。小さなデータバッチをランダムな場所に書き込むため同じオーバーヘッドは発生しない SSD ドライブでは、この設定をオフにできます。

ダーティーページ, エクステント, フラッシュ, ページも参照

レ

レコードロック
インデックスレコードでのロック。たとえば、SELECT c1 FOR UPDATE FROM t WHERE c1 = 10; は、ほかのトランザクションの、t.c1 の値が 10 である行が挿入、更新、削除されるのを回避します。ギャップロックおよびネクストキーロックと対比してください。
ギャップロック, ロック, ネクストキーロックも参照

レプリケーション
すべてのデータベースが同じデータを持つように、マスターデータベースから 1 つまたは複数のスレーブデータベースに変更を送信する作業。この方法は、スケーラビリティー向上のためのロードバランシング、ディザスタリカバリ、ソフトウェアアップグレードおよび構成変更のテストなど、幅広く使用されます。変更は、行ベースレプリケーションおよびステートメントベースレプリケーションと呼ばれる方法によって、データベース間で送信できます。
行ベースレプリケーション, ステートメントベースレプリケーションも参照

連結されたインデックス
複合インデックスも参照

ロ

ログ
InnoDB コンテキストでは、「ログ」または「ログファイル」は通常、ib_logfile* ファイルによって表される Redo ログを示します。もう 1 つのログ領域 (物理的にはシステムテーブルスペースの一部) は Undo ログです。

MySQL で重要なほかの種類のログには、エラーログ (起動および実行時の問題を診断するため), バイナリログ (レプリケーションを操作し、特定時点のリストアを実行するため)、一般クエリーログ (アプリケーションの問題を診断するため)、およびスロークエリーログ (パフォーマンスの問題を診断するため) があります。
バイナリログ, エラーログ, 一般クエリーログ, ib_logfile, Redo ログ, スロークエリーログ, システムテーブルスペース, Undo ログも参照

ロググループ
Redo ログを構成するファイルのセット。通常、ib_logfile0 および ib_logfile1 の名前が付けられています。(この理由のため、ib_logfile と総称される場合があります。)
ib_logfile, Redo ログも参照

ログバッファー
Redo ログを構成するログファイルに書き込まれるデータを保持するメモリー領域。これは、innodb_log_buffer_size 構成オプションで制御されます。
ログファイル, Redo ログも参照

ログファイル
Redo ログを構成する ib_logfileN ファイルの 1 つ。データは、ログバッファーメモリー領域からこれらのファイルに書き込まれます。
ib_logfile, ログバッファー, Redo ログも参照

ロック
ロック戦略の一環として、テーブル、行、内部データ構造などのリソースへのアクセスを制御するオブジェクトの高レベル概念。パフォーマンスを集中的にチューニングする場合は、相互排他ロックやラッチなど、ロックを実装する実際の構造を徹底的に調べることができます。
ラッチ, lock mode, ロック, 相互排他ロックも参照

ロック
ほかのトランザクションによって照会または変更されているデータをトランザクションが見たり変更したりすることを防止するシステム。ロック戦略は、データベース操作の信頼性および一貫性 (ACID 概念の原則) と、良質な並列性に必要なパフォーマンスとのバランスを取る必要があります。ロック戦略を調整するときは、多くの場合、分離レベルを選択し、すべてのデータベース操作がその分離レベルについて安全で信頼性を持つことを保証する必要があります。
ACID, 並列性, 分離レベル, ラッチ, ロック, 相互排他ロック, トランザクションも参照

ロックエスカレーション
一部のデータベースシステムで使用される、多くの行ロックを単一テーブルロックに変換する操作。メモリー領域を節約しながら、テーブルへの並列アクセスを減らします。InnoDB は、行ロックに領域効率のよい表現を使用するので、ロックエスカレーションは必要ありません。
ロック, 行ロック, テーブルロックも参照

ロック読み取り
: InnoDB テーブルでのロック操作も実行する SELECT ステートメント。SELECT ... FOR UPDATE または SELECT ... LOCK IN SHARE MODE のどちらか。トランザクションの分離レベルに応じて、デッドロックを発生させる可能性があります。非ロック読み取りの反対。読み取り専用トランザクション内のグローバルテーブルには許可されません。
デッドロック, 分離レベル, ロック, 非ロック読み取り, 読み取り専用トランザクションも参照

ロールバック
: トランザクションが行なった変更を元に戻してトランザクションを終了する SQL ステートメント。これは、トランザクションで行われた変更を永続的にするコミットの反対です。

MySQL はデフォルトで、各 SQL ステートメントに続いてコミットを自動的に発行する自動コミット設定を使用します。ロールバック方法を使用する前に、この設定を変更する必要があります。
ACID, コミット, トランザクションも参照

ロールバックセグメント
: Undo ログを含むストレージ領域。システムテーブルスペースの一部です。
システムテーブルスペース, Undo ログも参照

論理
: テーブル、クエリー、インデックス、その他の SQL 概念など、高レベル抽象側面を含むタイプの操作。論理側面は通常、データベース管理およびアプリケーション開発を便利で使用可能なものにするために重要です。物理と対比してください。
論理バックアップ, 物理も参照

論理バックアップ
: 実際のデータファイルをコピーせずにテーブル構造とデータを再生成するバックアップ。たとえば、mysqldump コマンドは論理バックアップを生成します。その出力に、データを再作成できる CREATE TABLE や INSERT などのステートメントが含まれるためです。物理バックアップと対比してください。論理バックアップは柔軟性 (たとえば、リストア前に、テーブル定義を編集したりステートメントを挿入したりできます) を提供しますが、物理バックアップよりもリストアにかなり長い時間がかかる可能性があります。
バックアップ, mysqldump, 物理バックアップ, リストアも参照

ワ

ワークロード
: 標準またはピーク使用時にデータベースアプリケーションによって実行される、SQL およびほかのデータベース操作の組み合わせおよび量。ボトルネックを識別するパフォーマンステスト中や容量計画中に、データベースに特定のワークロードを課すことができます。
ボトルネック, CPU バウンド, ディスクバウンド, SQLも参照

付録 E サードパーティーコンポーネントライセンス

目次

E.1 ANTLR 3 ライセンス 3040
E.2 Artistic ライセンス (Perl) 1.0 3040
E.3 Boost ライブラリライセンス 3042
E.4 dtoa.c ライセンス 3042
E.5 Editline ライセンス (libedit) ライセンス 3042
E.6 Editline ライセンス (libedit) ライセンス 3044
E.7 Expect.pm ライセンス 3047
E.8 Facebook Fast Checksum Patch ライセンス 3053
E.9 Facebook Patches ライセンス 3053
E.10 FindGTest.cmake ライセンス 3054
E.11 Fred Fish's Dbug Library ライセンス 3054
E.12 getarg ライセンス 3055
E.13 Gmock ライセンス 3056
E.14 GNU General Public License バージョン 2.0、1991 年 6 月 3056
E.15 GNU General Public License バージョン 3.0、2007 年 6 月 29 日および GCC Runtime Library Exception バージョン 3.1、2009 年 3 月 31 日 3061
E.16 GNU Lesser General Public License バージョン 2.1、1999 年 2 月 3070
E.17 GNU Libtool ライセンス 3077
E.18 GNU Readline ライセンス 3078
E.19 GNU Standard C++ ライブラリ (libstdc++) ライセンス 3078
E.20 Google Controlling Master Thread I/O Rate Patch ライセンス 3079
E.21 Google Perftools (TCMalloc ユーティリティー) ライセンス 3080
E.22 Google SMP Patch ライセンス 3080
E.23 Janson ライセンス 3081
E.24 lib_sql.cc ライセンス 3081
E.25 libevent ライセンス 3081
E.26 Linux-PAM ライセンス 3083
E.27 md5 (メッセージダイジェストアルゴリズム 5) ライセンス 3084
E.28 memcached ライセンス 3084
E.29 Memcached.pm ライセンス 3084
E.30 mkpasswd.pl ライセンス 3085
E.31 Node.js ライセンス 3088
E.32 nt_servc (Windows NT Service クラスライブラリ) ライセンス 3093
E.33 OpenPAM ライセンス 3093
E.34 OpenSSL v1.0 ライセンス 3093
E.35 Paramiko ライセンス 3095
E.36 Percona Multiple I/O スレッドパッチライセンス 3095
E.37 Pion ライセンス 3095
E.38 Python ライセンス 3096
E.39 Red Hat RPM Spec ファイルライセンス 3105
E.40 RegEX-Spencer ライブラリライセンス 3105
E.41 Richard A.O'Keefe 文字列ライブラリライセンス 3105
E.42 sajson ライセンス 3106
E.43 SHA-1 in C ライセンス 3106
E.44 Unicode データファイル 3106
E.45 V8 ライセンス 3107
E.46 zlib ライセンス 3110

MySQL のテストに使用した、MySQL Server のソースおよびコンポーネントに含めたライブラリのリストを、次に示します。これらを作成したすべての人々に感謝します。一部のコンポーネントでは、それらを含む製品のドキュメントにそれらのライセンス条項を含めることを要求される場合があります。これらのライセンス条項への相互参照はリスト内の該当する項目で示されます。

- GroupLens リサーチプロジェクト

  MySQL 品質保証チームは、MovieLens データセット (71,567 人のユーザーによる 10,681 本の映画に対する 1,000 万の評価と 100,000 のタグ) を MySQL 製品のテストに使用させていただいたことに感謝の念を示し、データセットを利用させていただいたことについてミネソタ大学での GroupLens リサーチプロジェクトにお礼を申し上げたいと思っています。

E.1 ANTLR 3 ライセンス

The following software may be included in this product:

ANTLR 3

ANTLR 3 License
[The BSD License]
Copyright (c) 2003-2007, Terence Parr
All rights reserved.

Redistribution and use in source and binary forms, with or without
modification, are permitted provided that the following conditions
are met:

* Redistributions of source code must retain the above copyright
  notice, this list of conditions and the following disclaimer.
* Redistributions in binary form must reproduce the above copyright
  notice, this list of conditions and the following disclaimer in
  the documentation and/or other materials provided with the
  distribution.
* Neither the name of the author nor the names of its contributors
  may be used to endorse or promote products derived from this
  software without specific prior written permission.

THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS
"AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT
LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS
FOR A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE
COPYRIGHT OWNER OR CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT,
INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING,
BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES;
LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER
CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT
LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN
ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE
POSSIBILITY OF SUCH DAMAGE.

E.2 Artistic ライセンス (Perl) 1.0

The "Artistic License"

Preamble

The intent of this document is to state the conditions under which a
Package may be copied, such that the Copyright Holder maintains some
semblance of artistic control over the development of the package,
while giving the users of the package the right to use and distribute
the Package in a more-or-less customary fashion, plus the right to make
reasonable modifications.

Definitions:

> "Package" refers to the collection of files distributed by the
> Copyright Holder, and derivatives of that collection of files
> created through textual modification.
>
> "Standard Version" refers to such a Package if it has not been
> modified, or has been modified in accordance with the wishes
> of the Copyright Holder as specified below.
>
> "Copyright Holder" is whoever is named in the copyright or
> copyrights for the package.
>
> "You" is you, if you're thinking about copying or distributing
> this Package.
>
> "Reasonable copying fee" is whatever you can justify on the
> basis of media cost, duplication charges, time of people involved,
> and so on. (You will not be required to justify it to the
> Copyright Holder, but only to the computing community at large
> as a market that must bear the fee.)
>
> "Freely Available" means that no fee is charged for the item
> itself, though there may be fees involved in handling the item.
> It also means that recipients of the item may redistribute it
> under the same conditions they received it.

1. You may make and give away verbatim copies of the source form of the Standard Version of this Package without restriction, provided that you duplicate all of the original copyright notices and associated disclaimers.

2. You may apply bug fixes, portability fixes and other modifications derived from the Public Domain or from the Copyright Holder. A Package modified in such a way shall still be considered the Standard Version.

3. You may otherwise modify your copy of this Package in any way, provided that you insert a prominent notice in each changed file stating how and when you changed that file, and provided that you do at least ONE of the following:

a) place your modifications in the Public Domain or otherwise make them Freely Available, such as by posting said modifications to Usenet or an equivalent medium, or placing the modifications on a major archive site such as uunet.uu.net, or by allowing the Copyright Holder to include your modifications in the Standard Version of the Package.

b) use the modified Package only within your corporation or organization.

c) rename any non-standard executables so the names do not conflict with standard executables, which must also be provided, and provide a separate manual page for each non-standard executable that clearly documents how it differs from the Standard Version.

d) make other distribution arrangements with the Copyright Holder.

4. You may distribute the programs of this Package in object code or executable form, provided that you do at least ONE of the following:

a) distribute a Standard Version of the executables and library files, together with instructions (in the manual page or equivalent) on where to get the Standard Version.

b) accompany the distribution with the machine-readable source of the Package with your modifications.

c) give non-standard executables non-standard names, and clearly document the differences in manual pages (or equivalent), together with instructions on where to get the Standard Version.

d) make other distribution arrangements with the Copyright Holder.

5. You may charge a reasonable copying fee for any distribution of this Package. You may charge any fee you choose for support of this Package. You may not charge a fee for this Package itself. However, you may distribute this Package in aggregate with other (possibly commercial) programs as part of a larger (possibly commercial) software distribution provided that you do not advertise this Package as a product of your own. You may embed this Package's interpreter within an executable of yours (by linking); this shall be construed as a mere form of aggregation, provided that the complete Standard Version of the interpreter is so embedded.

6. The scripts and library files supplied as input to or produced as output from the programs of this Package do not automatically fall under the copyright of this Package, but belong to whoever generated them, and may be sold commercially, and may be aggregated with this Package. If such scripts or library files are aggregated with this Package via the so-called "undump" or "unexec" methods of producing a binary executable image, then distribution of such an image shall neither be construed as a distribution of this Package nor shall it fall under the restrictions of Paragraphs 3 and 4, provided that you do not represent such an executable image as a Standard Version of this Package.

7. C subroutines (or comparably compiled subroutines in other languages) supplied by you and linked into this Package in order to emulate subroutines and variables of the language defined by this Package shall not be considered part of this Package, but are the equivalent of input as in Paragraph 6, provided these subroutines do not change the language in any way that would cause it to fail the regression tests for the language.

8. Aggregation of this Package with a commercial distribution is always permitted provided that the use of this Package is embedded; that is, when no overt attempt is made to make this Package's interfaces visible to the end user of the commercial distribution. Such use shall not be construed as a distribution of this Package.

```
9. The name of the Copyright Holder may not be used to endorse or promote
products derived from this software without specific prior written
permission.

10. THIS PACKAGE IS PROVIDED "AS IS" AND WITHOUT ANY EXPRESS OR
IMPLIED WARRANTIES, INCLUDING, WITHOUT LIMITATION, THE IMPLIED
WARRANTIES OF MERCHANTIBILITY AND FITNESS FOR A PARTICULAR PURPOSE.

                        The End
```

E.3 Boost ライブラリライセンス

この製品には、次のソフトウェアが含まれている場合があります。

Boost C++ ライブラリ

このソフトウェアの使用は、次のライセンスの条項に従います。

```
Boost Software License - Version 1.0 - August 17th, 2003

Permission is hereby granted, free of charge, to any person or
organization obtaining a copy of the software and accompanying
documentation covered by this license (the "Software") to use,
reproduce, display, distribute, execute, and transmit the Software,
and to prepare derivative works of the Software, and to permit
third-parties to whom the Software is furnished to do so, all
subject to the following:

The copyright notices in the Software and this entire statement,
including the above license grant, this restriction and the
following disclaimer, must be included in all copies of the
Software, in whole or in part, and all derivative works of the
Software, unless such copies or derivative works are solely in the
form of machine-executable object code generated by a source
language processor.

THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND,
EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF
MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE, TITLE AND
NON-INFRINGEMENT. IN NO EVENT SHALL THE COPYRIGHT HOLDERS OR ANYONE
DISTRIBUTING THE SOFTWARE BE LIABLE FOR ANY DAMAGES OR OTHER
LIABILITY, WHETHER IN CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT
OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS
IN THE SOFTWARE.
```

E.4 dtoa.c ライセンス

この製品には、次のソフトウェアが含まれている場合があります。

dtoa.c

```
The author of this software is David M. Gay.

Copyright (c) 1991, 2000, 2001 by Lucent Technologies.

Permission to use, copy, modify, and distribute this software for
any purpose without fee is hereby granted, provided that this entire
notice is included in all copies of any software which is or includes
a copy or modification of this software and in all copies of the
supporting documentation for such software.

THIS SOFTWARE IS BEING PROVIDED "AS IS", WITHOUT ANY EXPRESS OR
IMPLIED WARRANTY. IN PARTICULAR, NEITHER THE AUTHOR NOR LUCENT
MAKES ANY REPRESENTATION OR WARRANTY OF ANY KIND CONCERNING THE
MERCHANTABILITY OF THIS SOFTWARE OR ITS FITNESS FOR ANY PARTICULAR
PURPOSE.
```

E.5 Editline ライセンス (libedit) ライセンス

この製品には、次のソフトウェアが含まれている場合があります。

Editline ライブラリ (libedit)

一部のファイルは次のとおりです。

```
Copyright (c) 1992, 1993
```

一部のファイルは次のとおりです。

一部のファイルは次のとおりです。

```
Redistribution and use in source and binary forms,
with or without modification, are permitted provided
that the following conditions are met:

1. Redistributions of source code must retain the
   above copyright notice, this list of conditions
   and the following disclaimer.
2. Redistributions in binary form must reproduce
   the above copyright notice, this list of conditions
   and the following disclaimer in the documentation
   and/or other materials provided with the distribution.

THIS SOFTWARE IS PROVIDED BY THE NETBSD FOUNDATION, INC.
AND CONTRIBUTORS "AS IS" AND ANY EXPRESS OR IMPLIED
WARRANTIES, INCLUDING, BUT NOT LIMITED  TO, THE IMPLIED
WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A
PARTICULAR PURPOSE ARE DISCLAIMED.  IN NO EVENT SHALL
THE FOUNDATION OR CONTRIBUTORS BE LIABLE FOR ANY
DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR
CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO,
PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF
USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER
CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN
CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE
OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS
SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH
DAMAGE.
```

一部のファイルは次のとおりです。

```
Copyright (c) 1998 Todd C. Miller <Todd.Miller@courtesan.com>

Permission to use, copy, modify, and distribute this
software for any purpose with or without fee is hereby
granted, provided that the above copyright notice and
this permission notice appear in all copies.

THE SOFTWARE IS PROVIDED "AS IS" AND TODD C. MILLER
DISCLAIMS ALL WARRANTIES WITH REGARD TO THIS SOFTWARE
INCLUDING ALL IMPLIED WARRANTIES OF MERCHANTABILITY
AND FITNESS. IN NO EVENT SHALL TODD C. MILLER BE LIABLE
FOR ANY SPECIAL, DIRECT, INDIRECT, OR CONSEQUENTIAL
DAMAGES OR ANY DAMAGES WHATSOEVER RESULTING FROM
LOSS OF USE, DATA OR PROFITS, WHETHER IN AN ACTION
OF CONTRACT, NEGLIGENCE OR OTHER TORTIOUS ACTION,
ARISING OUT OF OR IN CONNECTION WITH THE USE OR
PERFORMANCE OF THIS SOFTWARE.
```

E.6 Editline ライセンス (libedit) ライセンス

この製品には、次のソフトウェアが含まれている場合があります。

Editline ライブラリ (libedit)

一部のファイルは次のとおりです。

```
Copyright (c) 1992, 1993
The Regents of the University of California. All rights reserved.

This code is derived from software contributed to
Berkeley by Christos Zoulas of Cornell University.

Redistribution and use in source and binary forms,
with or without modification, are permitted provided
that the following conditions are met:

1. Redistributions of source code must retain the
  above copyright notice, this list of conditions
  and the following disclaimer.
2. Redistributions in binary form must reproduce the
  above copyright notice, this list of conditions and
  the following disclaimer in the documentation and/or
  other materials provided with the distribution.
3. Neither the name of the University nor the names of
  its contributors may be used to endorse or promote
  products derived from this software without specific
  prior written permission.
```

```
THIS SOFTWARE IS PROVIDED BY THE REGENTS AND CONTRIBUTORS
"AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING,
BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY
AND FITNESS FOR A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO
EVENT SHALL THE REGENTS OR CONTRIBUTORS BE LIABLE FOR ANY
DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR
CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO,
PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE,
DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED
AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT
LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING
IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED
OF THE POSSIBILITY OF SUCH DAMAGE.
```

一部のファイルは次のとおりです。

```
Copyright (c) 2001 The NetBSD Foundation, Inc.
All rights reserved.

This code is derived from software contributed to The NetBSD Foundation

by Anthony Mallet.

Redistribution and use in source and binary forms,
with or without modification, are permitted provided
that the following conditions are met:

1. Redistributions of source code must retain the
   above copyright notice, this list of conditions
   and the following disclaimer.
2. Redistributions in binary form must reproduce the
   above copyright notice, this list of conditions and the
   following disclaimer in the documentation and/or
   other materials provided with the distribution.

THIS SOFTWARE IS PROVIDED BY THE NETBSD FOUNDATION, INC.
AND CONTRIBUTORS "AS IS" AND ANY EXPRESS OR IMPLIED
WARRANTIES, INCLUDING, BUT NOT LIMITED  TO, THE IMPLIED
WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A
PARTICULAR PURPOSE ARE DISCLAIMED.  IN NO EVENT SHALL
THE FOUNDATION OR CONTRIBUTORS BE LIABLE FOR ANY
DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR
CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO,
PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF
USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER
CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN
CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE
OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS
SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH
DAMAGE.
```

一部のファイルは次のとおりです。

```
Copyright (c) 1997 The NetBSD Foundation, Inc.
All rights reserved.

This code is derived from software contributed to The NetBSD Foundation

by Jaromir Dolecek.

Redistribution and use in source and binary forms,
with or without modification, are permitted provided
that the following conditions are met:

1. Redistributions of source code must retain the
   above copyright notice, this list of conditions
   and the following disclaimer.
2. Redistributions in binary form must reproduce
   the above copyright notice, this list of conditions
   and the following disclaimer in the documentation
   and/or other materials provided with the distribution.

THIS SOFTWARE IS PROVIDED BY THE NETBSD FOUNDATION, INC.
AND CONTRIBUTORS "AS IS" AND ANY EXPRESS OR IMPLIED
WARRANTIES, INCLUDING, BUT NOT LIMITED  TO, THE IMPLIED
WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A
PARTICULAR PURPOSE ARE DISCLAIMED.  IN NO EVENT SHALL
THE FOUNDATION OR CONTRIBUTORS BE LIABLE FOR ANY
DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR
CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO,
```

```
PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF
USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER
CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN
CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE
OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS
SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH
DAMAGE.
```

一部のファイルは次のとおりです。

```
Copyright (c) 1998 Todd C. Miller <Todd.Miller@courtesan.com>

Permission to use, copy, modify, and distribute this
software for any purpose with or without fee is hereby
granted, provided that the above copyright notice and
this permission notice appear in all copies.

THE SOFTWARE IS PROVIDED "AS IS" AND TODD C. MILLER
DISCLAIMS ALL WARRANTIES WITH REGARD TO THIS SOFTWARE
INCLUDING ALL IMPLIED WARRANTIES OF MERCHANTABILITY
AND FITNESS. IN NO EVENT SHALL TODD C. MILLER BE LIABLE
FOR ANY SPECIAL, DIRECT, INDIRECT, OR CONSEQUENTIAL
DAMAGES OR ANY DAMAGES WHATSOEVER RESULTING FROM
LOSS OF USE, DATA OR PROFITS, WHETHER IN AN ACTION
OF CONTRACT, NEGLIGENCE OR OTHER TORTIOUS ACTION,
ARISING OUT OF OR IN CONNECTION WITH THE USE OR
PERFORMANCE OF THIS SOFTWARE.
```

一部のファイルは次のとおりです。

```
Copyright (c) 1998 The NetBSD Foundation, Inc.
All rights reserved.

This code is derived from software contributed to The NetBSD
Foundation by Christos Zoulas.

Redistribution and use in source and binary forms, with or
without modification, are permitted provided that the following
conditions
are met:
1. Redistributions of source code must retain the above copyright
   notice, this list of conditions and the following disclaimer.
2. Redistributions in binary form must reproduce the above
   copyright notice, this list of conditions and the following
   disclaimer in the documentation and/or other materials provided
   with the distribution.

THIS SOFTWARE IS PROVIDED BY THE NETBSD FOUNDATION, INC. AND
CONTRIBUTORS "AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES,
INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF
MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE
DISCLAIMED. IN NO EVENT SHALL THE FOUNDATION OR CONTRIBUTORS
BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY,
OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT
OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS;
OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF
LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING
NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS
SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.
```

一部のファイルは次のとおりです。

```
Copyright (c) 2009 The NetBSD Foundation, Inc.
All rights reserved.

Redistribution and use in source and binary forms, with or without
modification, are permitted provided that the following conditions
are met:
1. Redistributions of source code must retain the above copyright
   notice, this list of conditions and the following disclaimer.
2. Redistributions in binary form must reproduce the above copyright
   notice, this list of conditions and the following disclaimer in the
   documentation and/or other materials provided with the
   distribution.
3. All advertising materials mentioning features or use of this
   software must display the following acknowledgement:
       This product includes software developed by the NetBSD
       Foundation, Inc. and its contributors.
4. Neither the name of The NetBSD Foundation nor the names of its
   contributors may be used to endorse or promote products derived
```

```
  from this software without specific prior written permission.

THIS SOFTWARE IS PROVIDED BY THE NETBSD FOUNDATION, INC. AND
CONTRIBUTORS "AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES,
INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF
MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE
DISCLAIMED. IN NO EVENT SHALL THE FOUNDATION OR CONTRIBUTORS
BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY,
OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT
OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS;
OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF
LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING
NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF
THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.
```

E.7 Expect.pm ライセンス

この製品には、次のソフトウェアが含まれている場合があります。

Expect.pm Perl モジュール

```
Expect.pm is licensed under the Perl license, which is essentially a dual
license.

Oracle may use, redistribute and/or modify this code under the terms of
either:

 a) the GNU General Public License as published by the Free Software
Foundation; either version 1, or (at your option) any later version, or

 b) the "Artistic License" which comes with the Expect/pr code.

Oracle elects to use the GPLv2 for version of MySQL that are licensed under
the GPL.

Oracle elects to use the Artistic license for all other (commercial) versions
of MySQL.

A copy of the GPLv2 and the Artistic License (Perl) 1.0 must be included with
any distribution:

The GNU General Public License (GPL-2.0)
Version 2, June 1991

Copyright (C) 1989, 1991 Free Software Foundation, Inc.
59 Temple Place, Suite 330, Boston, MA 02111-1307 USA

Everyone is permitted to copy and distribute verbatim copies
of this license document, but changing it is not allowed.

Preamble

The licenses for most software are designed to take away your freedom to
share and change it. By contrast, the GNU General Public License is intended
to guarantee your freedom to share and change free software--to make sure the
software is free for all its users. This General Public License applies to
most of the Free Software Foundation's software and to any other program
whose authors commit to using it. (Some other Free Software Foundation
software is covered by the GNU Library General Public License instead.) You
can apply it to your programs, too.

When we speak of free software, we are referring to freedom, not price. Our
General Public Licenses are designed to make sure that you have the freedom
to distribute copies of free software (and charge for this service if you
wish), that you receive source code or can get it if you want it, that you
can change the software or use pieces of it in new free programs; and that
you know you can do these things.

To protect your rights, we need to make restrictions that forbid anyone to
deny you these rights or to ask you to surrender the rights. These
restrictions translate to certain responsibilities for you if you distribute
copies of the software, or if you modify it.

For example, if you distribute copies of such a program, whether gratis or
for a fee, you must give the recipients all the rights that you have. You
must make sure that they, too, receive or can get the source code. And you
must show them these terms so they know their rights.

We protect your rights with two steps: (1) copyright the software, and (2)
```

offer you this license which gives you legal permission to copy, distribute
and/or modify the software.

Also, for each author's protection and ours, we want to make certain that
everyone understands that there is no warranty for this free software. If the
software is modified by someone else and passed on, we want its recipients to
know that what they have is not the original, so that any problems introduced
by others will not reflect on the original authors' reputations.

Finally, any free program is threatened constantly by software patents. We
wish to avoid the danger that redistributors of a free program will
individually obtain patent licenses, in effect making the program
proprietary. To prevent this, we have made it clear that any patent must be
licensed for everyone's free use or not licensed at all.

The precise terms and conditions for copying, distribution and modification
follow.

TERMS AND CONDITIONS FOR COPYING, DISTRIBUTION AND MODIFICATION

0. This License applies to any program or other work which contains a notice
placed by the copyright holder saying it may be distributed under the terms
of this General Public License. The "Program", below, refers to any such
program or work, and a "work based on the Program" means either the Program
or any derivative work under copyright law: that is to say, a work containing
the Program or a portion of it, either verbatim or with modifications and/or
translated into another language. (Hereinafter, translation is included
without limitation in the term "modification".) Each licensee is addressed as
"you".

Activities other than copying, distribution and modification are not covered
by this License; they are outside its scope. The act of running the Program
is not restricted, and the output from the Program is covered only if its
contents constitute a work based on the Program (independent of having been
made by running the Program). Whether that is true depends on what the
Program does.

1. You may copy and distribute verbatim copies of the Program's source code
as you receive it, in any medium, provided that you conspicuously and
appropriately publish on each copy an appropriate copyright notice and
disclaimer of warranty; keep intact all the notices that refer to this
License and to the absence of any warranty; and give any other recipients of
the Program a copy of this License along with the Program.

You may charge a fee for the physical act of transferring a copy, and you may
at your option offer warranty protection in exchange for a fee.

2. You may modify your copy or copies of the Program or any portion of it,
thus forming a work based on the Program, and copy and distribute such
modifications or work under the terms of Section 1 above, provided that you
also meet all of these conditions:

    a) You must cause the modified files to carry prominent notices stating
that you changed the files and the date of any change.

    b) You must cause any work that you distribute or publish, that in whole
or in part contains or is derived from the Program or any part thereof, to be
licensed as a whole at no charge to all third parties under the terms of this
License.

    c) If the modified program normally reads commands interactively when
run, you must cause it, when started running for such interactive use in the
most ordinary way, to print or display an announcement including an
appropriate copyright notice and a notice that there is no warranty (or else,
saying that you provide a warranty) and that users may redistribute the
program under these conditions, and telling the user how to view a copy of
this License. (Exception: if the Program itself is interactive but does not
normally print such an announcement, your work based on the Program is not
required to print an announcement.)

These requirements apply to the modified work as a whole. If identifiable
sections of that work are not derived from the Program, and can be reasonably
considered independent and separate works in themselves, then this License,
and its terms, do not apply to those sections when you distribute them as
separate works. But when you distribute the same sections as part of a whole
which is a work based on the Program, the distribution of the whole must be
on the terms of this License, whose permissions for other licensees extend to
the entire whole, and thus to each and every part regardless of who wrote it.

Thus, it is not the intent of this section to claim rights or contest your

rights to work written entirely by you; rather, the intent is to exercise the
right to control the distribution of derivative or collective works based on
the Program.

In addition, mere aggregation of another work not based on the Program with
the Program (or with a work based on the Program) on a volume of a storage or
distribution medium does not bring the other work under the scope of this
License.

3. You may copy and distribute the Program (or a work based on it, under
Section 2) in object code or executable form under the terms of Sections 1
and 2 above provided that you also do one of the following:

    a) Accompany it with the complete corresponding machine-readable source
code, which must be distributed under the terms of Sections 1 and 2 above on
a medium customarily used for software interchange; or,

    b) Accompany it with a written offer, valid for at least three years, to
give any third party, for a charge no more than your cost of physically
performing source distribution, a complete machine-readable copy of the
corresponding source code, to be distributed under the terms of Sections 1
and 2 above on a medium customarily used for software interchange; or,

    c) Accompany it with the information you received as to the offer to
distribute corresponding source code. (This alternative is allowed only for
noncommercial distribution and only if you received the program in object
code or executable form with such an offer, in accord with Subsection b
above.)

The source code for a work means the preferred form of the work for making
modifications to it. For an executable work, complete source code means all
the source code for all modules it contains, plus any associated interface
definition files, plus the scripts used to control compilation and
installation of the executable. However, as a special exception, the source
code distributed need not include anything that is normally distributed (in
either source or binary form) with the major components (compiler, kernel,
and so on) of the operating system on which the executable runs, unless that
component itself accompanies the executable.

If distribution of executable or object code is made by offering access to
copy from a designated place, then offering equivalent access to copy the
source code from the same place counts as distribution of the source code,
even though third parties are not compelled to copy the source along with the
object code.

4. You may not copy, modify, sublicense, or distribute the Program except as
expressly provided under this License. Any attempt otherwise to copy, modify,
sublicense or distribute the Program is void, and will automatically
terminate your rights under this License. However, parties who have received
copies, or rights, from you under this License will not have their licenses
terminated so long as such parties remain in full compliance.

5. You are not required to accept this License, since you have not signed it.
However, nothing else grants you permission to modify or distribute the
Program or its derivative works. These actions are prohibited by law if you
do not accept this License. Therefore, by modifying or distributing the
Program (or any work based on the Program), you indicate your acceptance of
this License to do so, and all its terms and conditions for copying,
distributing or modifying the Program or works based on it.

6. Each time you redistribute the Program (or any work based on the Program),
the recipient automatically receives a license from the original licensor to
copy, distribute or modify the Program subject to these terms and conditions.
You may not impose any further restrictions on the recipients' exercise of
the rights granted herein. You are not responsible for enforcing compliance
by third parties to this License.

7. If, as a consequence of a court judgment or allegation of patent
infringement or for any other reason (not limited to patent issues),
conditions are imposed on you (whether by court order, agreement or
otherwise) that contradict the conditions of this License, they do not excuse
you from the conditions of this License. If you cannot distribute so as to
satisfy simultaneously your obligations under this License and any other
pertinent obligations, then as a consequence you may not distribute the
Program at all. For example, if a patent license would not permit
royalty-free redistribution of the Program by all those who receive copies
directly or indirectly through you, then the only way you could satisfy both
it and this License would be to refrain entirely from distribution of the
Program.

If any portion of this section is held invalid or unenforceable under any
particular circumstance, the balance of the section is intended to apply and
the section as a whole is intended to apply in other circumstances.

It is not the purpose of this section to induce you to infringe any patents
or other property right claims or to contest validity of any such claims;
this section has the sole purpose of protecting the integrity of the free
software distribution system, which is implemented by public license
practices. Many people have made generous contributions to the wide range of
software distributed through that system in reliance on consistent
application of that system; it is up to the author/donor to decide if he or
she is willing to distribute software through any other system and a licensee
cannot impose that choice.

This section is intended to make thoroughly clear what is believed to be a
consequence of the rest of this License.

8. If the distribution and/or use of the Program is restricted in certain
countries either by patents or by copyrighted interfaces, the original
copyright holder who places the Program under this License may add an
explicit geographical distribution limitation excluding those countries, so
that distribution is permitted only in or among countries not thus excluded.
In such case, this License incorporates the limitation as if written in the
body of this License.

9. The Free Software Foundation may publish revised and/or new versions of
the General Public License from time to time. Such new versions will be
similar in spirit to the present version, but may differ in detail to address
new problems or concerns.

Each version is given a distinguishing version number. If the Program
specifies a version number of this License which applies to it and "any later
version", you have the option of following the terms and conditions either of
that version or of any later version published by the Free Software
Foundation. If the Program does not specify a version number of this License,
you may choose any version ever published by the Free Software Foundation.

10. If you wish to incorporate parts of the Program into other free programs
whose distribution conditions are different, write to the author to ask for
permission. For software which is copyrighted by the Free Software
Foundation, write to the Free Software Foundation; we sometimes make
exceptions for this. Our decision will be guided by the two goals of
preserving the free status of all derivatives of our free software and of
promoting the sharing and reuse of software generally.

NO WARRANTY

11. BECAUSE THE PROGRAM IS LICENSED FREE OF CHARGE, THERE IS NO WARRANTY FOR
THE PROGRAM, TO THE EXTENT PERMITTED BY APPLICABLE LAW. EXCEPT WHEN OTHERWISE
STATED IN WRITING THE COPYRIGHT HOLDERS AND/OR OTHER PARTIES PROVIDE THE
PROGRAM "AS IS" WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESSED OR IMPLIED,
INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND
FITNESS FOR A PARTICULAR PURPOSE. THE ENTIRE RISK AS TO THE QUALITY AND
PERFORMANCE OF THE PROGRAM IS WITH YOU. SHOULD THE PROGRAM PROVE DEFECTIVE,
YOU ASSUME THE COST OF ALL NECESSARY SERVICING, REPAIR OR CORRECTION.

12. IN NO EVENT UNLESS REQUIRED BY APPLICABLE LAW OR AGREED TO IN WRITING
WILL ANY COPYRIGHT HOLDER, OR ANY OTHER PARTY WHO MAY MODIFY AND/OR
REDISTRIBUTE THE PROGRAM AS PERMITTED ABOVE, BE LIABLE TO YOU FOR DAMAGES,
INCLUDING ANY GENERAL, SPECIAL, INCIDENTAL OR CONSEQUENTIAL DAMAGES ARISING
OUT OF THE USE OR INABILITY TO USE THE PROGRAM (INCLUDING BUT NOT LIMITED TO
LOSS OF DATA OR DATA BEING RENDERED INACCURATE OR LOSSES SUSTAINED BY YOU OR
THIRD PARTIES OR A FAILURE OF THE PROGRAM TO OPERATE WITH ANY OTHER
PROGRAMS), EVEN IF SUCH HOLDER OR OTHER PARTY HAS BEEN ADVISED OF THE
POSSIBILITY OF SUCH DAMAGES.

END OF TERMS AND CONDITIONS

How to Apply These Terms to Your New Programs

If you develop a new program, and you want it to be of the greatest possible
use to the public, the best way to achieve this is to make it free software
which everyone can redistribute and change under these terms.

To do so, attach the following notices to the program. It is safest to attach
them to the start of each source file to most effectively convey the
exclusion of warranty; and each file should have at least the "copyright"
line and a pointer to where the full notice is found.

    One line to give the program's name and a brief idea of what it does.

Copyright (C) <year> <name of author>

This program is free software; you can redistribute it and/or modify it
under the terms of the GNU General Public License as published by the Free
Software Foundation; either version 2 of the License, or (at your option) any
later version.

This program is distributed in the hope that it will be useful, but
WITHOUT ANY WARRANTY; without even the implied warranty of MERCHANTABILITY or
FITNESS FOR A PARTICULAR PURPOSE. See the GNU General Public License for more
details.

You should have received a copy of the GNU General Public License along
with this program; if not, write to the Free Software Foundation, Inc., 59
Temple Place, Suite 330, Boston, MA 02111-1307 USA

Also add information on how to contact you by electronic and paper mail.

If the program is interactive, make it output a short notice like this when
it starts in an interactive mode:

Gnomovision version 69, Copyright (C) year name of author Gnomovision
comes with ABSOLUTELY NO WARRANTY; for details type `show w'. This is free
software, and you are welcome to redistribute it under certain conditions;
type `show c' for details.

The hypothetical commands `show w' and `show c' should show the appropriate
parts of the General Public License. Of course, the commands you use may be
called something other than `show w' and `show c'; they could even be
mouse-clicks or menu items--whatever suits your program.

You should also get your employer (if you work as a programmer) or your
school, if any, to sign a "copyright disclaimer" for the program, if
necessary. Here is a sample; alter the names:

Yoyodyne, Inc., hereby disclaims all copyright interest in the program
`Gnomovision' (which makes passes at compilers) written by James Hacker.

signature of Ty Coon, 1 April 1989
Ty Coon, President of Vice

This General Public License does not permit incorporating your program into
proprietary programs. If your program is a subroutine library, you may
consider it more useful to permit linking proprietary applications with the
library. If this is what you want to do, use the GNU Library General Public
License instead of this License.

The "Artistic License"

Preamble

The intent of this document is to state the conditions under which a
Package may be copied, such that the Copyright Holder maintains some
semblance of artistic control over the development of the package,
while giving the users of the package the right to use and distribute
the Package in a more-or-less customary fashion, plus the right to make
reasonable modifications.

Definitions:

"Package" refers to the collection of files distributed by the
Copyright Holder, and derivatives of that collection of files
created through textual modification.

"Standard Version" refers to such a Package if it has not been
modified, or has been modified in accordance with the wishes
of the Copyright Holder as specified below.

"Copyright Holder" is whoever is named in the copyright or
copyrights for the package.

"You" is you, if you're thinking about copying or distributing
this Package.

"Reasonable copying fee" is whatever you can justify on the
basis of media cost, duplication charges, time of people involved,
and so on. (You will not be required to justify it to the
Copyright Holder, but only to the computing community at large

as a market that must bear the fee.)

"Freely Available" means that no fee is charged for the item itself, though there may be fees involved in handling the item. It also means that recipients of the item may redistribute it under the same conditions they received it.

1. You may make and give away verbatim copies of the source form of the Standard Version of this Package without restriction, provided that you duplicate all of the original copyright notices and associated disclaimers.

2. You may apply bug fixes, portability fixes and other modifications derived from the Public Domain or from the Copyright Holder. A Package modified in such a way shall still be considered the Standard Version.

3. You may otherwise modify your copy of this Package in any way, provided that you insert a prominent notice in each changed file stating how and when you changed that file, and provided that you do at least ONE of the following:

a) place your modifications in the Public Domain or otherwise make them Freely Available, such as by posting said modifications to Usenet or an equivalent medium, or placing the modifications on a major archive site such as uunet.uu.net, or by allowing the Copyright Holder to include your modifications in the Standard Version of the Package.

b) use the modified Package only within your corporation or organization.

c) rename any non-standard executables so the names do not conflict with standard executables, which must also be provided, and provide a separate manual page for each non-standard executable that clearly documents how it differs from the Standard Version.

d) make other distribution arrangements with the Copyright Holder.

4. You may distribute the programs of this Package in object code or executable form, provided that you do at least ONE of the following:

a) distribute a Standard Version of the executables and library files, together with instructions (in the manual page or equivalent) on where to get the Standard Version.

b) accompany the distribution with the machine-readable source of the Package with your modifications.

c) give non-standard executables non-standard names, and clearly document the differences in manual pages (or equivalent), together with instructions on where to get the Standard Version.

d) make other distribution arrangements with the Copyright Holder.

5. You may charge a reasonable copying fee for any distribution of this Package. You may charge any fee you choose for support of this Package. You may not charge a fee for this Package itself. However, you may distribute this Package in aggregate with other (possibly commercial) programs as part of a larger (possibly commercial) software distribution provided that you do not advertise this Package as a product of your own. You may embed this Package's interpreter within an executable of yours (by linking); this shall be construed as a mere form of aggregation, provided that the complete Standard Version of the interpreter is so embedded.

6. The scripts and library files supplied as input to or produced as output from the programs of this Package do not automatically fall under the copyright of this Package, but belong to whoever generated them, and may be sold commercially, and may be aggregated with this Package. If such scripts or library files are aggregated with this Package via the so-called "undump" or "unexec" methods of producing a binary executable image, then distribution of such an image shall neither be construed as a distribution of this Package nor shall it fall under the restrictions of Paragraphs 3 and 4, provided that you do not represent such an executable image as a Standard Version of this Package.

7. C subroutines (or comparably compiled subroutines in other languages) supplied by you and linked into this Package in order to emulate subroutines and variables of the language defined by this Package shall not be considered part of this Package, but are the equivalent of input as in Paragraph 6, provided these subroutines do not change the language in any way that would cause it to fail the

regression tests for the language.

8. Aggregation of this Package with a commercial distribution is always
permitted provided that the use of this Package is embedded; that is,
when no overt attempt is made to make this Package's interfaces visible
to the end user of the commercial distribution. Such use shall not be
construed as a distribution of this Package.

9. The name of the Copyright Holder may not be used to endorse or promote
products derived from this software without specific prior written
permission.

10. THIS PACKAGE IS PROVIDED "AS IS" AND WITHOUT ANY EXPRESS OR
IMPLIED WARRANTIES, INCLUDING, WITHOUT LIMITATION, THE IMPLIED
WARRANTIES OF MERCHANTIBILITY AND FITNESS FOR A PARTICULAR PURPOSE.

The End

E.8 Facebook Fast Checksum Patch ライセンス

この製品には、次のソフトウェアが含まれている場合があります。

Facebook Fast Checksum Patch

Copyright (C) 2009-2010 Facebook, Inc. All Rights Reserved.

Redistribution and use in source and binary forms, with or without
modification, are permitted provided that the following conditions are met:

1. Redistributions of source code must retain the above copyright notice,
   this list of conditions and the following disclaimer.
2. Redistributions in binary form must reproduce the above copyright notice,
   this list of conditions and the following disclaimer in the documentation
   and/or other materials provided with the distribution.

THIS SOFTWARE IS PROVIDED BY FACEBOOK, INC. “AS IS” AND ANY EXPRESS OR
IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF
MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO
EVENT SHALL FACEBOOK, INC. BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL,
SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO,
PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS;
OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY,
WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR
OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF
ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.

Also included:

crc32.c -- compute the CRC-32 of a buf stream
Copyright (C) 1995-2005 Mark Adler

This software is provided 'as-is', without any express or implied
warranty. In no event will the authors be held liable for any damages
arising from the use of this software.

Permission is granted to anyone to use this software for any purpose,
including commercial applications, and to alter it and redistribute it
freely, subject to the following restrictions:

1. The origin of this software must not be misrepresented; you must not
   claim that you wrote the original software. If you use this software
   in a product, an acknowledgment in the product documentation would be
   appreciated but is not required.
2. Altered source versions must be plainly marked as such, and must not be
   misrepresented as being the original software.
3. This notice may not be removed or altered from any source distribution.

Jean-loup Gailly jloup@gzip.org
Mark Adler madler@alumni.caltech.edu

E.9 Facebook Patches ライセンス

この製品には、次のソフトウェアが含まれている場合があります。

Copyright (c) 2012, Facebook, Inc.
All rights reserved.
Redistribution and use in source and binary forms, with or without
modification, are permitted provided that the following conditions are met:

E.10 FindGTest.cmake ライセンス

この製品には、次のソフトウェアが含まれている場合があります。

E.11 Fred Fish's Dbug Library ライセンス

この製品には、次のソフトウェアが含まれている場合があります。

Fred Fish's Dbug Library

```
                    N O T I C E

                Copyright Abandoned, 1987, Fred Fish


    This previously copyrighted work has been placed into the  public
    domain  by  the  author  and  may be freely used for any purpose,
    private or commercial.


    Because of the number of inquiries I was receiving about the  use
    of this product in commercially developed works I have decided to
    simply make it public domain to further its unrestricted use.   I
    specifically  would  be  most happy to see this material become a
    part of the standard Unix distributions by AT&T and the  Berkeley
    Computer  Science  Research Group, and a standard part of the GNU
    system from the Free Software Foundation.


    I would appreciate it, as a courtesy, if this notice is  left  in
    all copies and derivative works.  Thank you.


    The author makes no warranty of any kind  with  respect  to  this
    product  and  explicitly disclaims any implied warranties of mer-
    chantability or fitness for any particular purpose.
```

The dbug_analyze.c file is subject to the following notice:

```
    Copyright June 1987, Binayak Banerjee
    All rights reserved.

    This program may be freely distributed under the same terms and
    conditions as Fred Fish's Dbug package.
```

E.12 getarg ライセンス

この製品には、次のソフトウェアが含まれている場合があります。

getarg 関数 (getarg.h、getarg.c ファイル)

```
Copyright (c) 1997 – 2000 Kungliga Tekniska Högskolan
(Royal Institute of Technology, Stockholm, Sweden).
All rights reserved.

Redistribution and use in source and binary forms, with
or without modification, are permitted provided that the
following conditions are met:

1. Redistributions of source code must retain the above
   copyright notice, this list of conditions and the
   following disclaimer.
2. Redistributions in binary form must reproduce the above
   copyright notice, this list of conditions and the following
   disclaimer in the documentation and/or other materials
   provided with the distribution.
3. Neither the name of the Institute nor the names of its
   contributors may be used to endorse or promote products
   derived from this software without specific prior written
   permission.

THIS SOFTWARE IS PROVIDED BY THE INSTITUTE AND CONTRIBUTORS
```

```
"AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING,
BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY
AND FITNESS FOR A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO
EVENT SHALL THE INSTITUTE OR CONTRIBUTORS BE LIABLE FOR ANY
DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL
DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE
GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS
INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY,
WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING
NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF
THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH
DAMAGE.
```

E.13 Gmock ライセンス

この Oracle 製品は Gmock (gtest 付属) を含んでいるか、参照します。これは次の条件下で Oracle にライセンスが付与されています。

```
Copyright 2008, Google Inc.
All rights reserved.

Redistribution and use in source and binary forms, with or without
modification, are permitted provided that the following conditions are met:

* Redistributions of source code must retain the above copyright notice, this
list of conditions and the following disclaimer.
* Redistributions in binary form must reproduce the above copyright notice,
this list of conditions and the following disclaimerin the documentation
and/or other materials provided with the distribution.
* Neither the name of Google Inc. nor the names of its contributors may be
used to endorse or promote products derived from this software without
specific prior written permission.

THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS "AS IS"
AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE
IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE
ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT OWNER OR CONTRIBUTORS BE
LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR
CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF
SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS
INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN
CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE)
ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE
POSSIBILITY OF SUCH DAMAGE.
```

E.14 GNU General Public License バージョン 2.0、1991 年 6 月

```
The following applies to all products licensed under the GNU General
Public License, Version 2.0: You may not use the identified files
except in compliance with the GNU General Public License, Version
2.0 (the "License.") You may obtain a copy of the License at
http://www.gnu.org/licenses/gpl-2.0.txt. A copy of the license is
also reproduced below. Unless required by applicable law or agreed
to in writing, software distributed under the License is distributed
on an "AS IS" BASIS, WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND,
either express or implied. See the License for the specific language
governing permissions and limitations under the License.

GNU GENERAL PUBLIC LICENSE
Version 2, June 1991

Copyright (C) 1989, 1991 Free Software Foundation, Inc.,
51 Franklin Street, Fifth Floor, Boston, MA 02110-1301 USA
Everyone is permitted to copy and distribute verbatim
copies of this license document, but changing it is not
allowed.

                    Preamble

  The licenses for most software are designed to take away your
freedom to share and change it.  By contrast, the GNU General Public
License is intended to guarantee your freedom to share and change free
software--to make sure the software is free for all its users.  This
General Public License applies to most of the Free Software
Foundation's software and to any other program whose authors commit to
using it.  (Some other Free Software Foundation software is covered by
the GNU Lesser General Public License instead.)  You can apply it to
your programs, too.
```

When we speak of free software, we are referring to freedom, not price. Our General Public Licenses are designed to make sure that you have the freedom to distribute copies of free software (and charge for this service if you wish), that you receive source code or can get it if you want it, that you can change the software or use pieces of it in new free programs; and that you know you can do these things.

To protect your rights, we need to make restrictions that forbid anyone to deny you these rights or to ask you to surrender the rights. These restrictions translate to certain responsibilities for you if you distribute copies of the software, or if you modify it.

For example, if you distribute copies of such a program, whether gratis or for a fee, you must give the recipients all the rights that you have. You must make sure that they, too, receive or can get the source code. And you must show them these terms so they know their rights.

We protect your rights with two steps: (1) copyright the software, and (2) offer you this license which gives you legal permission to copy, distribute and/or modify the software.

Also, for each author's protection and ours, we want to make certain that everyone understands that there is no warranty for this free software. If the software is modified by someone else and passed on, we want its recipients to know that what they have is not the original, so that any problems introduced by others will not reflect on the original authors' reputations.

Finally, any free program is threatened constantly by software patents. We wish to avoid the danger that redistributors of a free program will individually obtain patent licenses, in effect making the program proprietary. To prevent this, we have made it clear that any patent must be licensed for everyone's free use or not licensed at all.

The precise terms and conditions for copying, distribution and modification follow.

GNU GENERAL PUBLIC LICENSE
TERMS AND CONDITIONS FOR COPYING, DISTRIBUTION AND MODIFICATION

0. This License applies to any program or other work which contains a notice placed by the copyright holder saying it may be distributed under the terms of this General Public License. The "Program", below, refers to any such program or work, and a "work based on the Program" means either the Program or any derivative work under copyright law: that is to say, a work containing the Program or a portion of it, either verbatim or with modifications and/or translated into another language. (Hereinafter, translation is included without limitation in the term "modification".) Each licensee is addressed as "you".

Activities other than copying, distribution and modification are not covered by this License; they are outside its scope. The act of running the Program is not restricted, and the output from the Program is covered only if its contents constitute a work based on the Program (independent of having been made by running the Program). Whether that is true depends on what the Program does.

1. You may copy and distribute verbatim copies of the Program's source code as you receive it, in any medium, provided that you conspicuously and appropriately publish on each copy an appropriate copyright notice and disclaimer of warranty; keep intact all the notices that refer to this License and to the absence of any warranty; and give any other recipients of the Program a copy of this License along with the Program.

You may charge a fee for the physical act of transferring a copy, and you may at your option offer warranty protection in exchange for a fee.

2. You may modify your copy or copies of the Program or any portion of it, thus forming a work based on the Program, and copy and distribute such modifications or work under the terms of Section 1 above, provided that you also meet all of these conditions:

a) You must cause the modified files to carry prominent notices stating that you changed the files and the date of any change.

b) You must cause any work that you distribute or publish, that in whole or in part contains or is derived from the Program or any

part thereof, to be licensed as a whole at no charge to all third parties under the terms of this License.

c) If the modified program normally reads commands interactively when run, you must cause it, when started running for such interactive use in the most ordinary way, to print or display an announcement including an appropriate copyright notice and a notice that there is no warranty (or else, saying that you provide a warranty) and that users may redistribute the program under these conditions, and telling the user how to view a copy of this License. (Exception: if the Program itself is interactive but does not normally print such an announcement, your work based on the Program is not required to print an announcement.)

These requirements apply to the modified work as a whole. If identifiable sections of that work are not derived from the Program, and can be reasonably considered independent and separate works in themselves, then this License, and its terms, do not apply to those sections when you distribute them as separate works. But when you distribute the same sections as part of a whole which is a work based on the Program, the distribution of the whole must be on the terms of this License, whose permissions for other licensees extend to the entire whole, and thus to each and every part regardless of who wrote it.

Thus, it is not the intent of this section to claim rights or contest your rights to work written entirely by you; rather, the intent is to exercise the right to control the distribution of derivative or collective works based on the Program.

In addition, mere aggregation of another work not based on the Program with the Program (or with a work based on the Program) on a volume of a storage or distribution medium does not bring the other work under the scope of this License.

3. You may copy and distribute the Program (or a work based on it, under Section 2) in object code or executable form under the terms of Sections 1 and 2 above provided that you also do one of the following:

a) Accompany it with the complete corresponding machine-readable source code, which must be distributed under the terms of Sections 1 and 2 above on a medium customarily used for software interchange; or,

b) Accompany it with a written offer, valid for at least three years, to give any third party, for a charge no more than your cost of physically performing source distribution, a complete machine-readable copy of the corresponding source code, to be distributed under the terms of Sections 1 and 2 above on a medium customarily used for software interchange; or,

c) Accompany it with the information you received as to the offer to distribute corresponding source code. (This alternative is allowed only for noncommercial distribution and only if you received the program in object code or executable form with such an offer, in accord with Subsection b above.)

The source code for a work means the preferred form of the work for making modifications to it. For an executable work, complete source code means all the source code for all modules it contains, plus any associated interface definition files, plus the scripts used to control compilation and installation of the executable. However, as a special exception, the source code distributed need not include anything that is normally distributed (in either source or binary form) with the major components (compiler, kernel, and so on) of the operating system on which the executable runs, unless that component itself accompanies the executable.

If distribution of executable or object code is made by offering access to copy from a designated place, then offering equivalent access to copy the source code from the same place counts as distribution of the source code, even though third parties are not compelled to copy the source along with the object code.

4. You may not copy, modify, sublicense, or distribute the Program except as expressly provided under this License. Any attempt otherwise to copy, modify, sublicense or distribute the Program is void, and will automatically terminate your rights under this License. However, parties who have received copies, or rights, from you under this License will not have their licenses terminated so long as such parties remain in full compliance.

5. You are not required to accept this License, since you have not signed it. However, nothing else grants you permission to modify or distribute the Program or its derivative works. These actions are prohibited by law if you do not accept this License. Therefore, by modifying or distributing the Program (or any work based on the Program), you indicate your acceptance of this License to do so, and all its terms and conditions for copying, distributing or modifying the Program or works based on it.

6. Each time you redistribute the Program (or any work based on the Program), the recipient automatically receives a license from the original licensor to copy, distribute or modify the Program subject to these terms and conditions. You may not impose any further restrictions on the recipients' exercise of the rights granted herein. You are not responsible for enforcing compliance by third parties to this License.

7. If, as a consequence of a court judgment or allegation of patent infringement or for any other reason (not limited to patent issues), conditions are imposed on you (whether by court order, agreement or otherwise) that contradict the conditions of this License, they do not excuse you from the conditions of this License. If you cannot distribute so as to satisfy simultaneously your obligations under this License and any other pertinent obligations, then as a consequence you may not distribute the Program at all. For example, if a patent license would not permit royalty-free redistribution of the Program by all those who receive copies directly or indirectly through you, then the only way you could satisfy both it and this License would be to refrain entirely from distribution of the Program.

If any portion of this section is held invalid or unenforceable under any particular circumstance, the balance of the section is intended to apply and the section as a whole is intended to apply in other circumstances.

It is not the purpose of this section to induce you to infringe any patents or other property right claims or to contest validity of any such claims; this section has the sole purpose of protecting the integrity of the free software distribution system, which is implemented by public license practices. Many people have made generous contributions to the wide range of software distributed through that system in reliance on consistent application of that system; it is up to the author/donor to decide if he or she is willing to distribute software through any other system and a licensee cannot impose that choice.

This section is intended to make thoroughly clear what is believed to be a consequence of the rest of this License.

8. If the distribution and/or use of the Program is restricted in certain countries either by patents or by copyrighted interfaces, the original copyright holder who places the Program under this License may add an explicit geographical distribution limitation excluding those countries, so that distribution is permitted only in or among countries not thus excluded. In such case, this License incorporates the limitation as if written in the body of this License.

9. The Free Software Foundation may publish revised and/or new versions of the General Public License from time to time. Such new versions will be similar in spirit to the present version, but may differ in detail to address new problems or concerns.

Each version is given a distinguishing version number. If the Program specifies a version number of this License which applies to it and "any later version", you have the option of following the terms and conditions either of that version or of any later version published by the Free Software Foundation. If the Program does not specify a version number of this License, you may choose any version ever published by the Free Software Foundation.

10. If you wish to incorporate parts of the Program into other free programs whose distribution conditions are different, write to the author to ask for permission. For software which is copyrighted by the Free Software Foundation, write to the Free Software Foundation; we sometimes make exceptions for this. Our decision will be guided by the two goals of preserving the free status of all derivatives of our free software and of promoting the sharing and reuse of software generally.

NO WARRANTY

11. BECAUSE THE PROGRAM IS LICENSED FREE OF CHARGE, THERE IS NO WARRANTY FOR THE PROGRAM, TO THE EXTENT PERMITTED BY APPLICABLE LAW. EXCEPT WHEN OTHERWISE STATED IN WRITING THE COPYRIGHT HOLDERS AND/OR OTHER PARTIES PROVIDE THE PROGRAM "AS IS" WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESSED OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE. THE ENTIRE RISK AS TO THE QUALITY AND PERFORMANCE OF THE PROGRAM IS WITH YOU. SHOULD THE PROGRAM PROVE DEFECTIVE, YOU ASSUME THE COST OF ALL NECESSARY SERVICING, REPAIR OR CORRECTION.

12. IN NO EVENT UNLESS REQUIRED BY APPLICABLE LAW OR AGREED TO IN WRITING WILL ANY COPYRIGHT HOLDER, OR ANY OTHER PARTY WHO MAY MODIFY AND/OR REDISTRIBUTE THE PROGRAM AS PERMITTED ABOVE, BE LIABLE TO YOU FOR DAMAGES, INCLUDING ANY GENERAL, SPECIAL, INCIDENTAL OR CONSEQUENTIAL DAMAGES ARISING OUT OF THE USE OR INABILITY TO USE THE PROGRAM (INCLUDING BUT NOT LIMITED TO LOSS OF DATA OR DATA BEING RENDERED INACCURATE OR LOSSES SUSTAINED BY YOU OR THIRD PARTIES OR A FAILURE OF THE PROGRAM TO OPERATE WITH ANY OTHER PROGRAMS), EVEN IF SUCH HOLDER OR OTHER PARTY HAS BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGES.

END OF TERMS AND CONDITIONS

How to Apply These Terms to Your New Programs

If you develop a new program, and you want it to be of the greatest possible use to the public, the best way to achieve this is to make it free software which everyone can redistribute and change under these terms.

To do so, attach the following notices to the program. It is safest to attach them to the start of each source file to most effectively convey the exclusion of warranty; and each file should have at least the "copyright" line and a pointer to where the full notice is found.

```
<one line to give the program's name and a brief idea of what it
does.>
Copyright (C) <year>  <name of author>

This program is free software; you can redistribute it and/or
modify it under the terms of the GNU General Public License as
published by     the Free Software Foundation; either version
2 of the License, or (at your option) any later version.

This program is distributed in the hope that it will be useful,
but WITHOUT ANY WARRANTY; without even the implied warranty of
MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
GNU General Public License for more details.

You should have received a copy of the GNU General Public License
along with this program; if not, write to the Free Software
Foundation, Inc., 51 Franklin Street, Fifth Floor, Boston, MA
02110-1301 USA.
```

Also add information on how to contact you by electronic and paper mail.

If the program is interactive, make it output a short notice like this when it starts in an interactive mode:

```
Gnomovision version 69, Copyright (C) year name of author
Gnomovision comes with ABSOLUTELY NO WARRANTY; for details
type 'show w'. This is free software, and you are welcome
to redistribute it under certain conditions; type 'show c'
for details.
```

The hypothetical commands 'show w' and 'show c' should show the appropriate parts of the General Public License. Of course, the commands you use may be called something other than 'show w' and 'show c'; they could even be mouse-clicks or menu items--whatever suits your program.

You should also get your employer (if you work as a programmer) or your school, if any, to sign a "copyright disclaimer" for the program, if necessary. Here is a sample; alter the names:

```
Yoyodyne, Inc., hereby disclaims all copyright interest in the
program 'Gnomovision' (which makes passes at compilers) written
by James Hacker.

<signature of Ty Coon>, 1 April 1989
```

Ty Coon, President of Vice

This General Public License does not permit incorporating your program
into proprietary programs. If your program is a subroutine library,
you may consider it more useful to permit linking proprietary
applications with the library. If this is what you want to do, use
the GNU Lesser General Public License instead of this License.

E.15 GNU General Public License バージョン 3.0、2007 年 6 月 29 日および GCC Runtime Library Exception バージョン 3.1、2009 年 3 月 31 日

GNU GENERAL PUBLIC LICENSE
Version 3, 29 June 2007

Copyright (C) 2007 Free Software Foundation, Inc. <http://fsf.org/>
Everyone is permitted to copy and distribute verbatim copies of
this license document, but changing it is not allowed.

Preamble

The GNU General Public License is a free, copyleft license for
software and other kinds of works.

The licenses for most software and other practical works are
designed to take away your freedom to share and change the works.
By contrast, the GNU General Public License is intended to guarantee
your freedom to share and change all versions of a program--to make
sure it remains free software for all its users. We, the Free
Software Foundation, use the GNU General Public License for most
of our software; it applies also to any other work released this
way by its authors. You can apply it to your programs, too.

When we speak of free software, we are referring to freedom, not
price. Our General Public Licenses are designed to make sure that
you have the freedom to distribute copies of free software (and
charge for them if you wish), that you receive source code or can
get it if you want it, that you can change the software or use
pieces of it in new free programs, and that you know you can do
these things.

To protect your rights, we need to prevent others from denying
you these rights or asking you to surrender the rights. Therefore,
you have certain responsibilities if you distribute copies of the
software, or if you modify it: responsibilities to respect the
freedom of others.

For example, if you distribute copies of such a program, whether
gratis or for a fee, you must pass on to the recipients the same
freedoms that you received. You must make sure that they, too,
receive or can get the source code. And you must show them these
terms so they know their rights.

Developers that use the GNU GPL protect your rights with two
steps: (1) assert copyright on the software, and (2) offer you this
License giving you legal permission to copy, distribute and/or
modify it.

For the developers' and authors' protection, the GPL clearly
explains that there is no warranty for this free software. For
both users' and authors' sake, the GPL requires that modified
versions be marked as changed, so that their problems will not be
attributed erroneously to authors of previous versions.

Some devices are designed to deny users access to install or run
modified versions of the software inside them, although the
manufacturer can do so. This is fundamentally incompatible with
the aim of protecting users' freedom to change the software. The
systematic pattern of such abuse occurs in the area of products for
individuals to use, which is precisely where it is most unacceptable.
Therefore, we have designed this version of the GPL to prohibit the
practice for those products. If such problems arise substantially
in other domains, we stand ready to extend this provision to those
domains in future versions of the GPL, as needed to protect the
freedom of users.

Finally, every program is threatened constantly by software

patents. States should not allow patents to restrict development
and use of software on general-purpose computers, but in those that
do, we wish to avoid the special danger that patents applied to a
free program could make it effectively proprietary. To prevent
this, the GPL assures that patents cannot be used to render the
program non-free.

The precise terms and conditions for copying, distribution and
modification follow.

TERMS AND CONDITIONS

0. Definitions.

"This License" refers to version 3 of the GNU General Public License.

"Copyright" also means copyright-like laws that apply to other kinds
of works, such as semiconductor masks.

"The Program" refers to any copyrightable work licensed under this
License. Each licensee is addressed as "you". "Licensees" and
"recipients" may be individuals or organizations.

To "modify" a work means to copy from or adapt all or part of the
work in a fashion requiring copyright permission, other than the
making of an exact copy. The resulting work is called a "modified
version" of the earlier work or a work "based on" the earlier work.

A "covered work" means either the unmodified Program or a work based
on the Program.

To "propagate" a work means to do anything with it that, without
permission, would make you directly or secondarily liable for
infringement under applicable copyright law, except executing it
on a computer or modifying a private copy. Propagation includes
copying, distribution (with or without modification), making available
to the public, and in some countries other activities as well.

To "convey" a work means any kind of propagation that enables other
parties to make or receive copies. Mere interaction with a user
through a computer network, with no transfer of a copy, is not
conveying.

An interactive user interface displays "Appropriate Legal Notices"
to the extent that it includes a convenient and prominently visible
feature that (1) displays an appropriate copyright notice, and (2)
tells the user that there is no warranty for the work (except to
the extent that warranties are provided), that licensees may convey
the work under this License, and how to view a copy of this License.
If the interface presents a list of user commands or options, such
as a menu, a prominent item in the list meets this criterion.

1. Source Code.

The "source code" for a work means the preferred form of the work
for making modifications to it. "Object code" means any non-source
form of a work.

A "Standard Interface" means an interface that either is an official
standard defined by a recognized standards body, or, in the case
of interfaces specified for a particular programming language, one
that is widely used among developers working in that language.

The "System Libraries" of an executable work include anything, other
than the work as a whole, that (a) is included in the normal form
of packaging a Major Component, but which is not part of that Major
Component, and (b) serves only to enable use of the work with that
Major Component, or to implement a Standard Interface for which an
implementation is available to the public in source code form. A
"Major Component", in this context, means a major essential component
(kernel, window system, and so on) of the specific operating system
(if any) on which the executable work runs, or a compiler used to
produce the work, or an object code interpreter used to run it.

The "Corresponding Source" for a work in object code form means all
the source code needed to generate, install, and (for an executable
work) run the object code and to modify the work, including scripts
to control those activities. However, it does not include the
work's System Libraries, or general-purpose tools or generally
available free programs which are used unmodified in performing

those activities but which are not part of the work. For example,
Corresponding Source includes interface definition files associated
with source files for the work, and the source code for shared
libraries and dynamically linked subprograms that the work is
specifically designed to require, such as by intimate data communication
or control flow between those subprograms and other parts of the
work.

The Corresponding Source need not include anything that users can
regenerate automatically from other parts of the Corresponding
Source.

The Corresponding Source for a work in source code form is that
same work.

2. Basic Permissions.

All rights granted under this License are granted for the term of
copyright on the Program, and are irrevocable provided the stated
conditions are met. This License explicitly affirms your unlimited
permission to run the unmodified Program. The output from running
a covered work is covered by this License only if the output, given
its content, constitutes a covered work. This License acknowledges
your rights of fair use or other equivalent, as provided by copyright
law.

You may make, run and propagate covered works that you do not convey,
without conditions so long as your license otherwise remains in
force. You may convey covered works to others for the sole purpose
of having them make modifications exclusively for you, or provide
you with facilities for running those works, provided that you
comply with the terms of this License in conveying all material for
which you do not control copyright. Those thus making or running
the covered works for you must do so exclusively on your behalf,
under your direction and control, on terms that prohibit them from
making any copies of your copyrighted material outside their
relationship with you.

Conveying under any other circumstances is permitted solely under
the conditions stated below. Sublicensing is not allowed; section
10 makes it unnecessary.

3. Protecting Users' Legal Rights From Anti-Circumvention Law.

No covered work shall be deemed part of an effective technological
measure under any applicable law fulfilling obligations under article
11 of the WIPO copyright treaty adopted on 20 December 1996, or
similar laws prohibiting or restricting circumvention of such
measures.

When you convey a covered work, you waive any legal power to forbid
circumvention of technological measures to the extent such circumvention
is effected by exercising rights under this License with respect
to the covered work, and you disclaim any intention to limit operation
or modification of the work as a means of enforcing, against the
work's users, your or third parties' legal rights to forbid
circumvention of technological measures.

4. Conveying Verbatim Copies.

You may convey verbatim copies of the Program's source code as you
receive it, in any medium, provided that you conspicuously and
appropriately publish on each copy an appropriate copyright notice;
keep intact all notices stating that this License and any non-permissive
terms added in accord with section 7 apply to the code; keep intact
all notices of the absence of any warranty; and give all recipients
a copy of this License along with the Program.

You may charge any price or no price for each copy that you convey,
and you may offer support or warranty protection for a fee.

5. Conveying Modified Source Versions.

You may convey a work based on the Program, or the modifications
to produce it from the Program, in the form of source code under
the terms of section 4, provided that you also meet all of these
conditions:

a) The work must carry prominent notices stating that you
modified it, and giving a relevant date.

b) The work must carry prominent notices stating that it is
released under this License and any conditions added under
section 7. This requirement modifies the requirement in
section 4 to "keep intact all notices".

c) You must license the entire work, as a whole, under this
License to anyone who comes into possession of a copy. This
License will therefore apply, along with any applicable section 7
additional terms, to the whole of the work, and all its parts,
regardless of how they are packaged. This License gives no
permission to license the work in any other way, but it does not
invalidate such permission if you have separately received it.

d) If the work has interactive user interfaces, each must display
Appropriate Legal Notices; however, if the Program has
interactive interfaces that do not display Appropriate Legal
Notices, your work need not make them do so.

A compilation of a covered work with other separate and independent
works, which are not by their nature extensions of the covered work,
and which are not combined with it such as to form a larger program,
in or on a volume of a storage or distribution medium, is called
an "aggregate" if the compilation and its resulting copyright are
not used to limit the access or legal rights of the compilation's
users beyond what the individual works permit. Inclusion of a
covered work in an aggregate does not cause this License to apply
to the other parts of the aggregate.

6. Conveying Non-Source Forms.

You may convey a covered work in object code form under the terms
of sections 4 and 5, provided that you also convey the machine-readable
Corresponding Source under the terms of this License, in one of
these ways:

a) Convey the object code in, or embodied in, a physical product
(including a physical distribution medium), accompanied by the
Corresponding Source fixed on a durable physical medium
customarily used for software interchange.

b) Convey the object code in, or embodied in, a physical product
(including a physical distribution medium), accompanied by a
written offer, valid for at least three years and valid for as
long as you offer spare parts or customer support for that
product model, to give anyone who possesses the object code
either (1) a copy of the Corresponding Source for all the
software in the product that is covered by this License, on a
durable physical medium customarily used for software
interchange, for a price no more than your reasonable cost
of physically performing this conveying of source, or (2)
access to copy the Corresponding Source from a network server
at no charge.

c) Convey individual copies of the object code with a copy of the
written offer to provide the Corresponding Source. This
alternative is allowed only occasionally and noncommercially, and
only if you received the object code with such an offer, in
accord with subsection 6b.

d) Convey the object code by offering access from a designated
place (gratis or for a charge), and offer equivalent access to
the Corresponding Source in the same way through the same place
at no further charge. You need not require recipients to copy
the Corresponding Source along with the object code. If the
place to copy the object code is a network server, the
Corresponding Source may be on a different server (operated
by you or a third party) that supports equivalent copying
facilities, provided you maintain clear directions next to the
object code saying where to find the Corresponding Source.
Regardless of what server hosts the Corresponding Source, you
remain obligated to ensure that it is available for as long
as needed to satisfy these requirements.

e) Convey the object code using peer-to-peer transmission,
provided you inform other peers where the object code and
Corresponding Source of the work are being offered to the
general public at no charge under subsection 6d.

A separable portion of the object code, whose source code is excluded

from the Corresponding Source as a System Library, need not be
included in conveying the object code work.

A "User Product" is either (1) a "consumer product", which means
any tangible personal property which is normally used for personal,
family, or household purposes, or (2) anything designed or sold for
incorporation into a dwelling. In determining whether a product
is a consumer product, doubtful cases shall be resolved in favor
of coverage. For a particular product received by a particular
user, "normally used" refers to a typical or common use of that
class of product, regardless of the status of the particular user
or of the way in which the particular user actually uses, or expects
or is expected to use, the product. A product is a consumer product
regardless of whether the product has substantial commercial,
industrial or non-consumer uses, unless such uses represent the
only significant mode of use of the product.

"Installation Information" for a User Product means any methods,
procedures, authorization keys, or other information required to
install and execute modified versions of a covered work in that
User Product from a modified version of its Corresponding Source.
The information must suffice to ensure that the continued functioning
of the modified object code is in no case prevented or interfered
with solely because modification has been made.

If you convey an object code work under this section in, or with,
or specifically for use in, a User Product, and the conveying occurs
as part of a transaction in which the right of possession and use
of the User Product is transferred to the recipient in perpetuity
or for a fixed term (regardless of how the transaction is characterized),
the Corresponding Source conveyed under this section must be
accompanied by the Installation Information. But this requirement
does not apply if neither you nor any third party retains the ability
to install modified object code on the User Product (for example,
the work has been installed in ROM).

The requirement to provide Installation Information does not include
a requirement to continue to provide support service, warranty, or
updates for a work that has been modified or installed by the
recipient, or for the User Product in which it has been modified
or installed. Access to a network may be denied when the modification
itself materially and adversely affects the operation of the network
or violates the rules and protocols for communication across the
network.

Corresponding Source conveyed, and Installation Information provided,
in accord with this section must be in a format that is publicly
documented (and with an implementation available to the public in
source code form), and must require no special password or key for
unpacking, reading or copying.

7. Additional Terms.

"Additional permissions" are terms that supplement the terms of
this License by making exceptions from one or more of its conditions.
Additional permissions that are applicable to the entire Program
shall be treated as though they were included in this License, to
the extent that they are valid under applicable law. If additional
permissions apply only to part of the Program, that part may be
used separately under those permissions, but the entire Program
remains governed by this License without regard to the additional
permissions.

When you convey a copy of a covered work, you may at your option
remove any additional permissions from that copy, or from any part
of it. (Additional permissions may be written to require their own
removal in certain cases when you modify the work.) You may place
additional permissions on material, added by you to a covered work,
for which you have or can give appropriate copyright permission.

Notwithstanding any other provision of this License, for material
you add to a covered work, you may (if authorized by the copyright
holders of that material) supplement the terms of this License with
terms:

a) Disclaiming warranty or limiting liability differently from
the terms of sections 15 and 16 of this License; or

b) Requiring preservation of specified reasonable legal notices
or author attributions in that material or in the Appropriate

Legal Notices displayed by works containing it; or

c) Prohibiting misrepresentation of the origin of that material,
or requiring that modified versions of such material be marked
in reasonable ways as different from the original version; or

d) Limiting the use for publicity purposes of names of licensors
or authors of the material; or

e) Declining to grant rights under trademark law for use of some
trade names, trademarks, or service marks; or

f) Requiring indemnification of licensors and authors of that
material by anyone who conveys the material (or modified versions
of it) with contractual assumptions of liability to the
recipient, for any liability that these contractual assumptions
directly impose on those licensors and authors.

All other non-permissive additional terms are considered "further
restrictions" within the meaning of section 10. If the Program as
you received it, or any part of it, contains a notice stating that
it is governed by this License along with a term that is a further
restriction, you may remove that term. If a license document
contains a further restriction but permits relicensing or conveying
under this License, you may add to a covered work material governed
by the terms of that license document, provided that the further
restriction does not survive such relicensing or conveying.

If you add terms to a covered work in accord with this section, you
must place, in the relevant source files, a statement of the
additional terms that apply to those files, or a notice indicating
where to find the applicable terms.

Additional terms, permissive or non-permissive, may be stated in
the form of a separately written license, or stated as exceptions;
the above requirements apply either way.

8. Termination.

You may not propagate or modify a covered work except as expressly
provided under this License. Any attempt otherwise to propagate
or modify it is void, and will automatically terminate your rights
under this License (including any patent licenses granted under the
third paragraph of section 11).

However, if you cease all violation of this License, then your
license from a particular copyright holder is reinstated (a)
provisionally, unless and until the copyright holder explicitly and
finally terminates your license, and (b) permanently, if the copyright
holder fails to notify you of the violation by some reasonable means
prior to 60 days after the cessation.

Moreover, your license from a particular copyright holder is
reinstated permanently if the copyright holder notifies you of the
violation by some reasonable means, this is the first time you have
received notice of violation of this License (for any work) from
that copyright holder, and you cure the violation prior to 30 days
after your receipt of the notice.

Termination of your rights under this section does not terminate
the licenses of parties who have received copies or rights from you
under this License. If your rights have been terminated and not
permanently reinstated, you do not qualify to receive new licenses
for the same material under section 10.

9. Acceptance Not Required for Having Copies.

You are not required to accept this License in order to receive or
run a copy of the Program. Ancillary propagation of a covered work
occurring solely as a consequence of using peer-to-peer transmission
to receive a copy likewise does not require acceptance. However,
nothing other than this License grants you permission to propagate or
modify any covered work. These actions infringe copyright if you do
not accept this License. Therefore, by modifying or propagating a
covered work, you indicate your acceptance of this License to do so.

10. Automatic Licensing of Downstream Recipients.

Each time you convey a covered work, the recipient automatically
receives a license from the original licensors, to run, modify and

propagate that work, subject to this License. You are not responsible for enforcing compliance by third parties with this License.

An "entity transaction" is a transaction transferring control of an organization, or substantially all assets of one, or subdividing an organization, or merging organizations. If propagation of a covered work results from an entity transaction, each party to that transaction who receives a copy of the work also receives whatever licenses to the work the party's predecessor in interest had or could give under the previous paragraph, plus a right to possession of the Corresponding Source of the work from the predecessor in interest, if the predecessor has it or can get it with reasonable efforts.

You may not impose any further restrictions on the exercise of the rights granted or affirmed under this License. For example, you may not impose a license fee, royalty, or other charge for exercise of rights granted under this License, and you may not initiate litigation (including a cross-claim or counterclaim in a lawsuit) alleging that any patent claim is infringed by making, using, selling, offering for sale, or importing the Program or any portion of it.

11. Patents.

A "contributor" is a copyright holder who authorizes use under this License of the Program or a work on which the Program is based. The work thus licensed is called the contributor's "contributor version".

A contributor's "essential patent claims" are all patent claims owned or controlled by the contributor, whether already acquired or hereafter acquired, that would be infringed by some manner, permitted by this License, of making, using, or selling its contributor version, but do not include claims that would be infringed only as a consequence of further modification of the contributor version. For purposes of this definition, "control" includes the right to grant patent sublicenses in a manner consistent with the requirements of this License.

Each contributor grants you a non-exclusive, worldwide, royalty-free patent license under the contributor's essential patent claims, to make, use, sell, offer for sale, import and otherwise run, modify and propagate the contents of its contributor version.

In the following three paragraphs, a "patent license" is any express agreement or commitment, however denominated, not to enforce a patent (such as an express permission to practice a patent or covenant not to sue for patent infringement). To "grant" such a patent license to a party means to make such an agreement or commitment not to enforce a patent against the party.

If you convey a covered work, knowingly relying on a patent license, and the Corresponding Source of the work is not available for anyone to copy, free of charge and under the terms of this License, through a publicly available network server or other readily accessible means, then you must either (1) cause the Corresponding Source to be so available, or (2) arrange to deprive yourself of the benefit of the patent license for this particular work, or (3) arrange, in a manner consistent with the requirements of this License, to extend the patent license to downstream recipients. "Knowingly relying" means you have actual knowledge that, but for the patent license, your conveying the covered work in a country, or your recipient's use of the covered work in a country, would infringe one or more identifiable patents in that country that you have reason to believe are valid.

If, pursuant to or in connection with a single transaction or arrangement, you convey, or propagate by procuring conveyance of, a covered work, and grant a patent license to some of the parties receiving the covered work authorizing them to use, propagate, modify or convey a specific copy of the covered work, then the patent license you grant is automatically extended to all recipients of the covered work and works based on it.

A patent license is "discriminatory" if it does not include within the scope of its coverage, prohibits the exercise of, or is conditioned on the non-exercise of one or more of the rights that are specifically granted under this License. You may not convey a covered work if you are a party to an arrangement with a third party that is in the business of distributing software, under which you make payment to the third party based on the extent of your activity of conveying

the work, and under which the third party grants, to any of the
parties who would receive the covered work from you, a discriminatory
patent license (a) in connection with copies of the covered work
conveyed by you (or copies made from those copies), or (b) primarily
for and in connection with specific products or compilations that
contain the covered work, unless you entered into that arrangement,
or that patent license was granted, prior to 28 March 2007.

Nothing in this License shall be construed as excluding or limiting
any implied license or other defenses to infringement that may
otherwise be available to you under applicable patent law.

12. No Surrender of Others' Freedom.

If conditions are imposed on you (whether by court order, agreement
or otherwise) that contradict the conditions of this License, they
do not excuse you from the conditions of this License. If you
cannot convey a covered work so as to satisfy simultaneously your
obligations under this License and any other pertinent obligations,
then as a consequence you may not convey it at all. For example,
if you agree to terms that obligate you to collect a royalty for
further conveying from those to whom you convey the Program, the
only way you could satisfy both those terms and this License would
be to refrain entirely from conveying the Program.

13. Use with the GNU Affero General Public License.

Notwithstanding any other provision of this License, you have
permission to link or combine any covered work with a work licensed
under version 3 of the GNU Affero General Public License into a
single combined work, and to convey the resulting work. The terms
of this License will continue to apply to the part which is the
covered work, but the special requirements of the GNU Affero General
Public License, section 13, concerning interaction through a network
will apply to the combination as such.

14. Revised Versions of this License.

The Free Software Foundation may publish revised and/or new versions
of the GNU General Public License from time to time. Such new
versions will be similar in spirit to the present version, but may
differ in detail to address new problems or concerns.

Each version is given a distinguishing version number. If the
Program specifies that a certain numbered version of the GNU General
Public License "or any later version" applies to it, you have the
option of following the terms and conditions either of that numbered
version or of any later version published by the Free Software
Foundation. If the Program does not specify a version number of
the GNU General Public License, you may choose any version ever
published by the Free Software Foundation.

If the Program specifies that a proxy can decide which future
versions of the GNU General Public License can be used, that proxy's
public statement of acceptance of a version permanently authorizes
you to choose that version for the Program.

Later license versions may give you additional or different
permissions. However, no additional obligations are imposed on any
author or copyright holder as a result of your choosing to follow
a later version.

15. Disclaimer of Warranty.

THERE IS NO WARRANTY FOR THE PROGRAM, TO THE EXTENT PERMITTED BY
APPLICABLE LAW. EXCEPT WHEN OTHERWISE STATED IN WRITING THE COPYRIGHT
HOLDERS AND/OR OTHER PARTIES PROVIDE THE PROGRAM "AS IS" WITHOUT
WARRANTY OF ANY KIND, EITHER EXPRESSED OR IMPLIED, INCLUDING, BUT
NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS
FOR A PARTICULAR PURPOSE. THE ENTIRE RISK AS TO THE QUALITY AND
PERFORMANCE OF THE PROGRAM IS WITH YOU. SHOULD THE PROGRAM PROVE
DEFECTIVE, YOU ASSUME THE COST OF ALL NECESSARY SERVICING, REPAIR
OR CORRECTION.

16. Limitation of Liability.

IN NO EVENT UNLESS REQUIRED BY APPLICABLE LAW OR AGREED TO IN WRITING
WILL ANY COPYRIGHT HOLDER, OR ANY OTHER PARTY WHO MODIFIES AND/OR
CONVEYS THE PROGRAM AS PERMITTED ABOVE, BE LIABLE TO YOU FOR DAMAGES,
INCLUDING ANY GENERAL, SPECIAL, INCIDENTAL OR CONSEQUENTIAL DAMAGES

ARISING OUT OF THE USE OR INABILITY TO USE THE PROGRAM (INCLUDING
BUT NOT LIMITED TO LOSS OF DATA OR DATA BEING RENDERED INACCURATE
OR LOSSES SUSTAINED BY YOU OR THIRD PARTIES OR A FAILURE OF THE
PROGRAM TO OPERATE WITH ANY OTHER PROGRAMS), EVEN IF SUCH HOLDER
OR OTHER PARTY HAS BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGES.

17. Interpretation of Sections 15 and 16.

If the disclaimer of warranty and limitation of liability provided
above cannot be given local legal effect according to their terms,
reviewing courts shall apply local law that most closely approximates
an absolute waiver of all civil liability in connection with the
Program, unless a warranty or assumption of liability accompanies a
copy of the Program in return for a fee.

END OF TERMS AND CONDITIONS

How to Apply These Terms to Your New Programs

If you develop a new program, and you want it to be of the greatest
possible use to the public, the best way to achieve this is to make
it free software which everyone can redistribute and change under
these terms.

To do so, attach the following notices to the program. It is safest
to attach them to the start of each source file to most effectively
state the exclusion of warranty; and each file should have at least
the "copyright" line and a pointer to where the full notice is
found.

```
<one line to give the program's name and a brief idea of what it does.>
Copyright (C) <year>  <name of author>

This program is free software: you can redistribute it and/or
modify it under the terms of the GNU General Public License
as published by the Free Software Foundation, either version 3
of the License, or (at your option) any later version.

This program is distributed in the hope that it will be useful,
but WITHOUT ANY WARRANTY; without even the implied warranty of
MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
GNU General Public License for more details.

You should have received a copy of the GNU General Public License
along with this program.  If not, see
<http://www.gnu.org/licenses/>.
```

Also add information on how to contact you by electronic and paper mail.

If the program does terminal interaction, make it output a short
notice like this when it starts in an interactive mode:

```
<program>  Copyright (C) <year>  <name of author>
This program comes with ABSOLUTELY NO WARRANTY; for details type 'show w'.
This is free software, and you are welcome to redistribute it
under certain conditions; type 'show c' for details.
```

The hypothetical commands 'show w' and 'show c' should show the
appropriate parts of the General Public License. Of course, your
program's commands might be different; for a GUI interface, you
would use an "about box".

You should also get your employer (if you work as a programmer) or
school, if any, to sign a "copyright disclaimer" for the program,
if necessary. For more information on this, and how to apply and
follow the GNU GPL, see <http://www.gnu.org/licenses/>.

The GNU General Public License does not permit incorporating your
program into proprietary programs. If your program is a subroutine
library, you may consider it more useful to permit linking proprietary
applications with the library. If this is what you want to do, use
the GNU Lesser General Public License instead of this License. But
first, please read <http://www.gnu.org/philosophy/why-not-lgpl.html>.
==

==
GCC RUNTIME LIBRARY EXCEPTION

Version 3.1, 31 March 2009

Copyright © 2009 Free Software Foundation, Inc. <http://fsf.org/>

Everyone is permitted to copy and distribute verbatim copies of
this license document, but changing it is not allowed.

This GCC Runtime Library Exception ("Exception") is an additional
permission under section 7 of the GNU General Public License, version
3 ("GPLv3"). It applies to a given file (the "Runtime Library")
that bears a notice placed by the copyright holder of the file
stating that the file is governed by GPLv3 along with this Exception.

When you use GCC to compile a program, GCC may combine portions of
certain GCC header files and runtime libraries with the compiled
program. The purpose of this Exception is to allow compilation of
non-GPL (including proprietary) programs to use, in this way, the
header files and runtime libraries covered by this Exception.

0. Definitions.

A file is an "Independent Module" if it either requires the Runtime
Library for execution after a Compilation Process, or makes use of
an interface provided by the Runtime Library, but is not otherwise
based on the Runtime Library.

"GCC" means a version of the GNU Compiler Collection, with or without
modifications, governed by version 3 (or a specified later version)
of the GNU General Public License (GPL) with the option of using
any subsequent versions published by the FSF.

"GPL-compatible Software" is software whose conditions of propagation,
modification and use would permit combination with GCC in accord
with the license of GCC.

"Target Code" refers to output from any compiler for a real or
virtual target processor architecture, in executable form or suitable
for input to an assembler, loader, linker and/or execution phase.
Notwithstanding that, Target Code does not include data in any
format that is used as a compiler intermediate representation, or
used for producing a compiler intermediate representation.

The "Compilation Process" transforms code entirely represented in
non-intermediate languages designed for human-written code, and/or
in Java Virtual Machine byte code, into Target Code. Thus, for
example, use of source code generators and preprocessors need not
be considered part of the Compilation Process, since the Compilation
Process can be understood as starting with the output of the
generators or preprocessors.

A Compilation Process is "Eligible" if it is done using GCC, alone
or with other GPL-compatible software, or if it is done without
using any work based on GCC. For example, using non-GPL-compatible
Software to optimize any GCC intermediate representations would not
qualify as an Eligible Compilation Process.

1. Grant of Additional Permission.

You have permission to propagate a work of Target Code formed by
combining the Runtime Library with Independent Modules, even if
such propagation would otherwise violate the terms of GPLv3, provided
that all Target Code was generated by Eligible Compilation Processes.
You may then convey such a combination under terms of your choice,
consistent with the licensing of the Independent Modules.

2. No Weakening of GCC Copyleft.

The availability of this Exception does not imply any general
presumption that third-party software is unaffected by the copyleft
requirements of the license of GCC.
==

E.16 GNU Lesser General Public License バージョン 2.1、1999 年 2 月

The following applies to all products licensed under the
GNU Lesser General Public License, Version 2.1: You may
not use the identified files except in compliance with
the GNU Lesser General Public License, Version 2.1 (the
"License"). You may obtain a copy of the License at

http://www.gnu.org/licenses/lgpl-2.1.html. A copy of the
license is also reproduced below. Unless required by
applicable law or agreed to in writing, software distributed
under the License is distributed on an "AS IS" BASIS,
WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express
or implied. See the License for the specific language governing
permissions and limitations under the License.

GNU LESSER GENERAL PUBLIC LICENSE
Version 2.1, February 1999

[This is the first released version of the Lesser GPL. It also counts
as the successor of the GNU Library Public License, version 2, hence
the version number 2.1.]

Preamble

The licenses for most software are designed to take away your
freedom to share and change it. By contrast, the GNU General Public
Licenses are intended to guarantee your freedom to share and change
free software--to make sure the software is free for all its users.

This license, the Lesser General Public License, applies to some
specially designated software packages--typically libraries--of the
Free Software Foundation and other authors who decide to use it. You
can use it too, but we suggest you first think carefully about whether
this license or the ordinary General Public License is the better
strategy to use in any particular case, based on the explanations below.

When we speak of free software, we are referring to freedom of use,
not price. Our General Public Licenses are designed to make sure that
you have the freedom to distribute copies of free software (and charge
for this service if you wish); that you receive source code or can get
it if you want it; that you can change the software and use pieces of
it in new free programs; and that you are informed that you can do
these things.

To protect your rights, we need to make restrictions that forbid
distributors to deny you these rights or to ask you to surrender these
rights. These restrictions translate to certain responsibilities for
you if you distribute copies of the library or if you modify it.

For example, if you distribute copies of the library, whether gratis
or for a fee, you must give the recipients all the rights that we gave
you. You must make sure that they, too, receive or can get the source
code. If you link other code with the library, you must provide
complete object files to the recipients, so that they can relink them
with the library after making changes to the library and recompiling
it. And you must show them these terms so they know their rights.

We protect your rights with a two-step method: (1) we copyright the
library, and (2) we offer you this license, which gives you legal
permission to copy, distribute and/or modify the library.

To protect each distributor, we want to make it very clear that
there is no warranty for the free library. Also, if the library is
modified by someone else and passed on, the recipients should know
that what they have is not the original version, so that the original
author's reputation will not be affected by problems that might be
introduced by others.

Finally, software patents pose a constant threat to the existence of
any free program. We wish to make sure that a company cannot
effectively restrict the users of a free program by obtaining a
restrictive license from a patent holder. Therefore, we insist that
any patent license obtained for a version of the library must be
consistent with the full freedom of use specified in this license.

Most GNU software, including some libraries, is covered by the
ordinary GNU General Public License. This license, the GNU Lesser
General Public License, applies to certain designated libraries, and
is quite different from the ordinary General Public License. We use
this license for certain libraries in order to permit linking those
libraries into non-free programs.

When a program is linked with a library, whether statically or using a shared library, the combination of the two is legally speaking a combined work, a derivative of the original library. The ordinary General Public License therefore permits such linking only if the entire combination fits its criteria of freedom. The Lesser General Public License permits more lax criteria for linking other code with the library.

We call this license the "Lesser" General Public License because it does Less to protect the user's freedom than the ordinary General Public License. It also provides other free software developers Less of an advantage over competing non-free programs. These disadvantages are the reason we use the ordinary General Public License for many libraries. However, the Lesser license provides advantages in certain special circumstances.

For example, on rare occasions, there may be a special need to encourage the widest possible use of a certain library, so that it becomes a de-facto standard. To achieve this, non-free programs must be allowed to use the library. A more frequent case is that a free library does the same job as widely used non-free libraries. In this case, there is little to gain by limiting the free library to free software only, so we use the Lesser General Public License.

In other cases, permission to use a particular library in non-free programs enables a greater number of people to use a large body of free software. For example, permission to use the GNU C Library in non-free programs enables many more people to use the whole GNU operating system, as well as its variant, the GNU/Linux operating system.

Although the Lesser General Public License is Less protective of the users' freedom, it does ensure that the user of a program that is linked with the Library has the freedom and the wherewithal to run that program using a modified version of the Library.

The precise terms and conditions for copying, distribution and modification follow. Pay close attention to the difference between a "work based on the library" and a "work that uses the library". The former contains code derived from the library, whereas the latter must be combined with the library in order to run.

GNU LESSER GENERAL PUBLIC LICENSE
TERMS AND CONDITIONS FOR COPYING, DISTRIBUTION AND MODIFICATION

0. This License Agreement applies to any software library or other program which contains a notice placed by the copyright holder or other authorized party saying it may be distributed under the terms of this Lesser General Public License (also called "this License"). Each licensee is addressed as "you".

A "library" means a collection of software functions and/or data prepared so as to be conveniently linked with application programs (which use some of those functions and data) to form executables.

The "Library", below, refers to any such software library or work which has been distributed under these terms. A "work based on the Library" means either the Library or any derivative work under copyright law: that is to say, a work containing the Library or a portion of it, either verbatim or with modifications and/or translated straightforwardly into another language. (Hereinafter, translation is included without limitation in the term "modification".)

"Source code" for a work means the preferred form of the work for making modifications to it. For a library, complete source code means all the source code for all modules it contains, plus any associated interface definition files, plus the scripts used to control compilation and installation of the library.

Activities other than copying, distribution and modification are not covered by this License; they are outside its scope. The act of running a program using the Library is not restricted, and output from such a program is covered only if its contents constitute a work based on the Library (independent of the use of the Library in a tool for writing it). Whether that is true depends on what the Library does and what the program that uses the Library does.

1. You may copy and distribute verbatim copies of the Library's complete source code as you receive it, in any medium, provided that you conspicuously and appropriately publish on each copy an

appropriate copyright notice and disclaimer of warranty; keep intact all the notices that refer to this License and to the absence of any warranty; and distribute a copy of this License along with the Library.

You may charge a fee for the physical act of transferring a copy, and you may at your option offer warranty protection in exchange for a fee.

2. You may modify your copy or copies of the Library or any portion of it, thus forming a work based on the Library, and copy and distribute such modifications or work under the terms of Section 1 above, provided that you also meet all of these conditions:

a) The modified work must itself be a software library.

b) You must cause the files modified to carry prominent notices stating that you changed the files and the date of any change.

c) You must cause the whole of the work to be licensed at no charge to all third parties under the terms of this License.

d) If a facility in the modified Library refers to a function or a table of data to be supplied by an application program that uses the facility, other than as an argument passed when the facility is invoked, then you must make a good faith effort to ensure that, in the event an application does not supply such function or table, the facility still operates, and performs whatever part of its purpose remains meaningful.

(For example, a function in a library to compute square roots has a purpose that is entirely well-defined independent of the application. Therefore, Subsection 2d requires that any application-supplied function or table used by this function must be optional: if the application does not supply it, the square root function must still compute square roots.)

These requirements apply to the modified work as a whole. If identifiable sections of that work are not derived from the Library, and can be reasonably considered independent and separate works in themselves, then this License, and its terms, do not apply to those sections when you distribute them as separate works. But when you distribute the same sections as part of a whole which is a work based on the Library, the distribution of the whole must be on the terms of this License, whose permissions for other licensees extend to the entire whole, and thus to each and every part regardless of who wrote it.

Thus, it is not the intent of this section to claim rights or contest your rights to work written entirely by you; rather, the intent is to exercise the right to control the distribution of derivative or collective works based on the Library.

In addition, mere aggregation of another work not based on the Library with the Library (or with a work based on the Library) on a volume of a storage or distribution medium does not bring the other work under the scope of this License.

3. You may opt to apply the terms of the ordinary GNU General Public License instead of this License to a given copy of the Library. To do this, you must alter all the notices that refer to this License, so that they refer to the ordinary GNU General Public License, version 2, instead of to this License. (If a newer version than version 2 of the ordinary GNU General Public License has appeared, then you can specify that version instead if you wish.) Do not make any other change in these notices.

Once this change is made in a given copy, it is irreversible for that copy, so the ordinary GNU General Public License applies to all subsequent copies and derivative works made from that copy.

This option is useful when you wish to copy part of the code of the Library into a program that is not a library.

4. You may copy and distribute the Library (or a portion or derivative of it, under Section 2) in object code or executable form under the terms of Sections 1 and 2 above provided that you accompany it with the complete corresponding machine-readable source code, which must be distributed under the terms of Sections 1 and 2 above on a medium customarily used for software interchange.

If distribution of object code is made by offering access to copy from a designated place, then offering equivalent access to copy the source code from the same place satisfies the requirement to distribute the source code, even though third parties are not compelled to copy the source along with the object code.

5. A program that contains no derivative of any portion of the Library, but is designed to work with the Library by being compiled or linked with it, is called a "work that uses the Library". Such a work, in isolation, is not a derivative work of the Library, and therefore falls outside the scope of this License.

However, linking a "work that uses the Library" with the Library creates an executable that is a derivative of the Library (because it contains portions of the Library), rather than a "work that uses the library". The executable is therefore covered by this License. Section 6 states terms for distribution of such executables.

When a "work that uses the Library" uses material from a header file that is part of the Library, the object code for the work may be a derivative work of the Library even though the source code is not. Whether this is true is especially significant if the work can be linked without the Library, or if the work is itself a library. The threshold for this to be true is not precisely defined by law.

If such an object file uses only numerical parameters, data structure layouts and accessors, and small macros and small inline functions (ten lines or less in length), then the use of the object file is unrestricted, regardless of whether it is legally a derivative work. (Executables containing this object code plus portions of the Library will still fall under Section 6.)

Otherwise, if the work is a derivative of the Library, you may distribute the object code for the work under the terms of Section 6. Any executables containing that work also fall under Section 6, whether or not they are linked directly with the Library itself.

6. As an exception to the Sections above, you may also combine or link a "work that uses the Library" with the Library to produce a work containing portions of the Library, and distribute that work under terms of your choice, provided that the terms permit modification of the work for the customer's own use and reverse engineering for debugging such modifications.

You must give prominent notice with each copy of the work that the Library is used in it and that the Library and its use are covered by this License. You must supply a copy of this License. If the work during execution displays copyright notices, you must include the copyright notice for the Library among them, as well as a reference directing the user to the copy of this License. Also, you must do one of these things:

a) Accompany the work with the complete corresponding machine-readable source code for the Library including whatever changes were used in the work (which must be distributed under Sections 1 and 2 above); and, if the work is an executable linked with the Library, with the complete machine-readable "work that uses the Library", as object code and/or source code, so that the user can modify the Library and then relink to produce a modified executable containing the modified Library. (It is understood that the user who changes the contents of definitions files in the Library will not necessarily be able to recompile the application to use the modified definitions.)

b) Use a suitable shared library mechanism for linking with the Library. A suitable mechanism is one that (1) uses at run time a copy of the library already present on the user's computer system, rather than copying library functions into the executable, and (2) will operate properly with a modified version of the library, if the user installs one, as long as the modified version is interface-compatible with the version that the work was made with.

c) Accompany the work with a written offer, valid for at least three years, to give the same user the materials specified in Subsection 6a, above, for a charge no more than the cost of performing this distribution.

d) If distribution of the work is made by offering access to copy from a designated place, offer equivalent access to copy the above

specified materials from the same place.

e) Verify that the user has already received a copy of these materials or that you have already sent this user a copy.

For an executable, the required form of the "work that uses the Library" must include any data and utility programs needed for reproducing the executable from it. However, as a special exception, the materials to be distributed need not include anything that is normally distributed (in either source or binary form) with the major components (compiler, kernel, and so on) of the operating system on which the executable runs, unless that component itself accompanies the executable.

It may happen that this requirement contradicts the license restrictions of other proprietary libraries that do not normally accompany the operating system. Such a contradiction means you cannot use both them and the Library together in an executable that you distribute.

7. You may place library facilities that are a work based on the Library side-by-side in a single library together with other library facilities not covered by this License, and distribute such a combined library, provided that the separate distribution of the work based on the Library and of the other library facilities is otherwise permitted, and provided that you do these two things:

a) Accompany the combined library with a copy of the same work based on the Library, uncombined with any other library facilities. This must be distributed under the terms of the Sections above.

b) Give prominent notice with the combined library of the fact that part of it is a work based on the Library, and explaining where to find the accompanying uncombined form of the same work.

8. You may not copy, modify, sublicense, link with, or distribute the Library except as expressly provided under this License. Any attempt otherwise to copy, modify, sublicense, link with, or distribute the Library is void, and will automatically terminate your rights under this License. However, parties who have received copies, or rights, from you under this License will not have their licenses terminated so long as such parties remain in full compliance.

9. You are not required to accept this License, since you have not signed it. However, nothing else grants you permission to modify or distribute the Library or its derivative works. These actions are prohibited by law if you do not accept this License. Therefore, by modifying or distributing the Library (or any work based on the Library), you indicate your acceptance of this License to do so, and all its terms and conditions for copying, distributing or modifying the Library or works based on it.

10. Each time you redistribute the Library (or any work based on the Library), the recipient automatically receives a license from the original licensor to copy, distribute, link with or modify the Library subject to these terms and conditions. You may not impose any further restrictions on the recipients' exercise of the rights granted herein. You are not responsible for enforcing compliance by third parties with this License.

11. If, as a consequence of a court judgment or allegation of patent infringement or for any other reason (not limited to patent issues), conditions are imposed on you (whether by court order, agreement or otherwise) that contradict the conditions of this License, they do not excuse you from the conditions of this License. If you cannot distribute so as to satisfy simultaneously your obligations under this License and any other pertinent obligations, then as a consequence you may not distribute the Library at all. For example, if a patent license would not permit royalty-free redistribution of the Library by all those who receive copies directly or indirectly through you, then the only way you could satisfy both it and this License would be to refrain entirely from distribution of the Library.

If any portion of this section is held invalid or unenforceable under any particular circumstance, the balance of the section is intended to apply, and the section as a whole is intended to apply in other circumstances.

It is not the purpose of this section to induce you to infringe any

patents or other property right claims or to contest validity of any
such claims; this section has the sole purpose of protecting the
integrity of the free software distribution system which is
implemented by public license practices. Many people have made
generous contributions to the wide range of software distributed
through that system in reliance on consistent application of that
system; it is up to the author/donor to decide if he or she is willing
to distribute software through any other system and a licensee cannot
impose that choice.

This section is intended to make thoroughly clear what is believed to
be a consequence of the rest of this License.

  12. If the distribution and/or use of the Library is restricted in
certain countries either by patents or by copyrighted interfaces, the
original copyright holder who places the Library under this License
may add an explicit geographical distribution limitation excluding
those countries, so that distribution is permitted only in or among
countries not thus excluded. In such case, this License incorporates
the limitation as if written in the body of this License.

  13. The Free Software Foundation may publish revised and/or new
versions of the Lesser General Public License from time to time.
Such new versions will be similar in spirit to the present version,
but may differ in detail to address new problems or concerns.

Each version is given a distinguishing version number. If the Library
specifies a version number of this License which applies to it and
"any later version", you have the option of following the terms and
conditions either of that version or of any later version published by
the Free Software Foundation. If the Library does not specify a
license version number, you may choose any version ever published by
the Free Software Foundation.

  14. If you wish to incorporate parts of the Library into other free
programs whose distribution conditions are incompatible with these,
write to the author to ask for permission. For software which is
copyrighted by the Free Software Foundation, write to the Free
Software Foundation; we sometimes make exceptions for this. Our
decision will be guided by the two goals of preserving the free status
of all derivatives of our free software and of promoting the sharing
and reuse of software generally.

                NO WARRANTY

  15. BECAUSE THE LIBRARY IS LICENSED FREE OF CHARGE, THERE IS NO
WARRANTY FOR THE LIBRARY, TO THE EXTENT PERMITTED BY APPLICABLE LAW.
EXCEPT WHEN OTHERWISE STATED IN WRITING THE COPYRIGHT HOLDERS AND/OR
OTHER PARTIES PROVIDE THE LIBRARY "AS IS" WITHOUT WARRANTY OF ANY
KIND, EITHER EXPRESSED OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE
IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR
PURPOSE. THE ENTIRE RISK AS TO THE QUALITY AND PERFORMANCE OF THE
LIBRARY IS WITH YOU. SHOULD THE LIBRARY PROVE DEFECTIVE, YOU ASSUME
THE COST OF ALL NECESSARY SERVICING, REPAIR OR CORRECTION.

  16. IN NO EVENT UNLESS REQUIRED BY APPLICABLE LAW OR AGREED TO IN
WRITING WILL ANY COPYRIGHT HOLDER, OR ANY OTHER PARTY WHO MAY MODIFY
AND/OR REDISTRIBUTE THE LIBRARY AS PERMITTED ABOVE, BE LIABLE TO YOU
FOR DAMAGES, INCLUDING ANY GENERAL, SPECIAL, INCIDENTAL OR
CONSEQUENTIAL DAMAGES ARISING OUT OF THE USE OR INABILITY TO USE THE
LIBRARY (INCLUDING BUT NOT LIMITED TO LOSS OF DATA OR DATA BEING
RENDERED INACCURATE OR LOSSES SUSTAINED BY YOU OR THIRD PARTIES OR A
FAILURE OF THE LIBRARY TO OPERATE WITH ANY OTHER SOFTWARE), EVEN IF
SUCH HOLDER OR OTHER PARTY HAS BEEN ADVISED OF THE POSSIBILITY OF SUCH
DAMAGES.

            END OF TERMS AND CONDITIONS

        How to Apply These Terms to Your New Libraries

  If you develop a new library, and you want it to be of the greatest
possible use to the public, we recommend making it free software that
everyone can redistribute and change. You can do so by permitting
redistribution under these terms (or, alternatively, under the terms
of the ordinary General Public License).

  To apply these terms, attach the following notices to the library.
It is safest to attach them to the start of each source file to most
effectively convey the exclusion of warranty; and each file should
have at least the "copyright" line and a pointer to where the full

```
notice is found.

    <one line to give the library's name and a brief idea of what it does.>
    Copyright (C) <year>  <name of author>

    This library is free software; you can redistribute it and/or
    modify it under the terms of the GNU Lesser General Public
    License as published by the Free Software Foundation; either
    version 2.1 of the License, or (at your option) any later version.

    This library is distributed in the hope that it will be useful,
    but WITHOUT ANY WARRANTY; without even the implied warranty of
    MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the GNU
    Lesser General Public License for more details.

    You should have received a copy of the GNU Lesser General Public
    License along with this library; if not, write to the Free Software
    Foundation, Inc., 51 Franklin Street, Fifth Floor, Boston, MA
    02110-1301  USA

Also add information on how to contact you by electronic and paper mail.

You should also get your employer (if you work as a programmer) or your
school, if any, to sign a "copyright disclaimer" for the library, if
necessary.  Here is a sample; alter the names:

  Yoyodyne, Inc., hereby disclaims all copyright interest in the
  library `Frob' (a library for tweaking knobs) written by James
  Random Hacker.

  <signature of Ty Coon>, 1 April 1990
  Ty Coon, President of Vice

That's all there is to it!
```

E.17 GNU Libtool ライセンス

この製品には、次のソフトウェアが含まれている場合があります。

GNU Libtool (GNU ポータブルライブラリツール)

```
If you are receiving a copy of the Oracle software in
source code, you are also receiving a copy of two files
(ltmain.sh and ltdl.h) generated by the GNU Libtool in
source code. If you received the Oracle software under
a license other than a commercial (non-GPL) license,
then the terms of the Oracle license do NOT apply to
these files from GNU Libtool; they are licensed under
the following licenses, separately from the Oracle
programs you receive.

Oracle elects to use GNU General Public License version
2 (GPL) for any software where a choice of GPL or GNU
Lesser/Library General Public License (LGPL) license
versions are made available with the language indicating
that GPL/LGPL or any later version may be used, or where
a choice of which version of the GPL/LGPL is applied is
unspecified.

From GNU Libtool:

ltmain.sh - Provide generalized library-building support
services.
NOTE: Changing this file will not affect anything until
you rerun configure.
Copyright (C) 1996, 1997, 1998, 1999, 2000, 2001, 2003, 2004,
2005, 2006, 2007 Free Software Foundation, Inc.
Originally by Gordon Matzigkeit, 1996

This program is free software; you can redistribute it
and/or modify it under the terms of the GNU General
Public License as published by the Free Software Foundation;
either version 2 of the License, or (at your option) any
later version.

This program is distributed in the hope that it will be
useful, but WITHOUT ANY WARRANTY; without even the implied
warranty of MERCHANTABILITY or FITNESS FOR A PARTICULAR
PURPOSE. See the GNU General Public License for more
```

```
details. You should have received a copy of the GNU General
Public License along with this program; if not, write to
the Free Software Foundation, Inc., 51 Franklin Street,
Fifth Floor, Boston, MA 02110-1301, USA.

As a special exception to the GNU General Public License,
if you distribute this file as part of a program that
contains a configuration script generated by Autoconf,
you may include it under the same distribution terms that
you use for the rest of that program.
```

このコンポーネントはセクションE.14「GNU General Public License バージョン 2.0、1991 年 6 月」に基づいてライセンスが付与されています

E.18 GNU Readline ライセンス

この製品には、次のソフトウェアが含まれている場合があります。

GNU Readline ライブラリ

```
GNU Readline Library
With respect to MySQL Server/Cluster software licensed
under GNU General Public License, you are receiving a
copy of the GNU Readline Library in source code. The
terms of any Oracle license that might accompany the
Oracle programs do NOT apply to the GNU Readline Library;
it is licensed under the following license, separately
from the Oracle programs you receive. Oracle elects to
use GNU General Public License version 2 (GPL) for any
software where a choice of GPL license versions are
made available with the language indicating that GPLv2
or any later version may be used, or where a choice of
which version of the GPL is applied is unspecified.
```

このコンポーネントはセクションE.14「GNU General Public License バージョン 2.0、1991 年 6 月」に基づいてライセンスが付与されています

E.19 GNU Standard C++ ライブラリ (libstdc++) ライセンス

この製品には次のソフトウェアが含まれている場合があります: GNU Standard C++ ライブラリ (libstdc++)

このコンポーネントはセクションE.15「GNU General Public License バージョン 3.0、2007 年 6 月 29 日および GCC Runtime Library Exception バージョン 3.1、2009 年 3 月 31 日」に基づいてライセンスが付与されています。

追加通知:

```
==
 Copyright (c) 1994
 Hewlett-Packard Company

 Permission to use, copy, modify, distribute and sell this software
 and its documentation for any purpose is hereby granted without fee,
 provided that the above copyright notice appear in all copies and
 that both that copyright notice and this permission notice appear
 in supporting documentation.  Hewlett-Packard Company makes no
 representations about the suitability of this software for any
 purpose.  It is provided "as is" without express or implied
 warranty.
==

==
 Copyright (c) 1996,1997
 Silicon Graphics Computer Systems, Inc.

 Permission to use, copy, modify, distribute and sell this software
 and its documentation for any purpose is hereby granted without fee,
 provided that the above copyright notice appear in all copies and
 that both that copyright notice and this permission notice appear
 in supporting documentation.  Silicon Graphics makes no
 representations about the suitability of this software for any
 purpose.  It is provided "as is" without express or implied
 warranty.
==

==
 shared_count.hpp
```

E.20 Google Controlling Master Thread I/O Rate Patch ライセンス

この製品には、次のソフトウェアが含まれている場合があります。

Google Controlling Master Thread I/O Rate Patch

```
COPYRIGHT OWNER OR CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT,
INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING,
BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES;
LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER
CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT
LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN
ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE
POSSIBILITY OF SUCH DAMAGE.
```

E.21 Google Perftools (TCMalloc ユーティリティー) ライセンス

この製品には、次のソフトウェアが含まれている場合があります。

```
Google Perftools (TCMalloc utility)

Copyright (c) 1998-2006, Google Inc.
All rights reserved.

Redistribution and use in source and binary forms, with or
without modification, are permitted provided that the following
conditions are met:

    * Redistributions of source code must retain the above
      copyright notice, this list of conditions and the following
      disclaimer.
    * Redistributions in binary form must reproduce the above
      copyright notice, this list of conditions and the following
      disclaimer in the documentation and/or other materials provided
      with the distribution.
    * Neither the name of Google Inc. nor the names of its
      contributors may be used to endorse or promote products
      derived from this software without specific prior written
      permission.

THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS
"AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT
LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR
A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT
OWNER OR CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL,
SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT
LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE,
DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY
THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT
(INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE
OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.
```

E.22 Google SMP Patch ライセンス

この製品には、次のソフトウェアが含まれている場合があります。

Google SMP Patch

```
Google SMP patch

Copyright (c) 2008, Google Inc.
All rights reserved.

Redistribution and use in source and binary forms, with or without
modification, are permitted provided that the following conditions
are met:

 * Redistributions of source code must retain the above copyright
   notice, this list of conditions and the following disclaimer.
 * Redistributions in binary form must reproduce the above copyright
   notice, this list of conditions and the following disclaimer in the
   documentation and/or other materials provided with the distribution.
 * Neither the name of the Google Inc. nor the names of its contributors
   may be used to endorse or promote products derived from this software
   without specific prior written permission.

THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS
"AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT
LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS
FOR A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE
COPYRIGHT OWNER OR CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT,
INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING,
BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES;
LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER
```

```
CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT
LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN
ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE
POSSIBILITY OF SUCH DAMAGE.
```

E.23 Janson ライセンス

この製品には、次のソフトウェアが含まれている場合があります。

Janson 2.6

```
Include the following verbatim in the documentation:

Licence Text:

Copyright (c) (c) 2009-2013 Petri Lehtinen <petri@digip.org>

Permission is hereby granted, free of charge, to any person obtaining a copy
of this software and associated documentation files (the "Software"), to deal
in the Software without restriction, including without limitation the rights
to use, copy, modify, merge, publish, distribute, sublicense, and/or sell
copies of the Software, and to permit persons to whom the Software is
furnished to do so, subject to the following conditions:

The above copyright notice and this permission notice shall be included in
all copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR
IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY,
FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE
AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER
LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM,
OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN
THE SOFTWARE.
```

E.24 lib_sql.cc ライセンス

この製品には、次のソフトウェアが含まれている場合があります。

lib_sql.cc

```
Copyright (c) 2000
SWsoft company

This material is provided "as is", with absolutely no warranty
expressed or implied. Any use is at your own risk.

Permission to use or copy this software for any purpose is hereby
granted without fee, provided the above notices are retained on
all copies. Permission to modify the code and to distribute modified
code is granted, provided the above notices are retained, and a
notice that the code was modified is included with the above copyright
notice.

This code was modified by the MySQL team.
```

E.25 libevent ライセンス

この製品には、次のソフトウェアが含まれている場合があります。

```
libevent

Copyright (c) 2000-2007 Niels Provos <provos@citi.umich.edu>
All rights reserved.

Redistribution and use in source and binary forms, with or without
modification, are permitted provided that the following conditions
are met:
1. Redistributions of source code must retain the above copyright
   notice, this list of conditions and the following disclaimer.
2. Redistributions in binary form must reproduce the above copyright
   notice, this list of conditions and the following disclaimer in the
   documentation and/or other materials provided with the distribution.
3. The name of the author may not be used to endorse or promote products
   derived from this software without specific prior written permission.

THIS SOFTWARE IS PROVIDED BY THE AUTHOR ``AS IS'' AND ANY EXPRESS OR
```

IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES
OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE DISCLAIMED.
IN NO EVENT SHALL THE AUTHOR BE LIABLE FOR ANY DIRECT, INDIRECT,
INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT
NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE,
DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY
THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT
(INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF
THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE

==
Parts developed by Adam Langley
==

==
log.c
Based on err.c, which was adapted from OpenBSD libc *err*warncode.

Copyright (c) 2005 Nick Mathewson
Copyright (c) 2000 Dug Song
Copyright (c) 1993 The Regents of the University of California.
All rights reserved.

Redistribution and use in source and binary forms, with or without
modification, are permitted provided that the following conditions
are met:
1. Redistributions of source code must retain the above copyright
   notice, this list of conditions and the following disclaimer.
2. Redistributions in binary form must reproduce the above copyright
   notice, this list of conditions and the following disclaimer in
   the documentation and/or other materials provided with the
   distribution.
3. Neither the name of the University nor the names of its
   contributors may be used to endorse or promote products derived
   from this software without specific prior written permission.

THIS SOFTWARE IS PROVIDED BY THE REGENTS AND CONTRIBUTORS "AS IS"
AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED
TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A
PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE REGENTS
OR CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL,
SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT
LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF
USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED
AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT
LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING
IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF
THE POSSIBILITY OF SUCH DAMAGE.
==

==
min_heap.h

Copyright (c) 2006 Maxim Yegorushkin
All rights reserved.

Redistribution and use in source and binary forms, with or without
modification, are permitted provided that the following conditions
are met:
1. Redistributions of source code must retain the above copyright
   notice, this list of conditions and the following disclaimer.
2. Redistributions in binary form must reproduce the above copyright
   notice, this list of conditions and the following disclaimer in
   the documentation and/or other materials provided with the
   distribution.
3. The name of the author may not be used to endorse or promote
   products derived from this software without specific prior
   written permission.

THIS SOFTWARE IS PROVIDED BY THE AUTHOR "AS IS" AND ANY EXPRESS
OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED
WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE
ARE DISCLAIMED. IN NO EVENT SHALL THE AUTHOR BE LIABLE FOR ANY
DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL
DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE
GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS
INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER
IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR
OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN
IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.

```
==

==
win32.c

Copyright 2000-2002 Niels Provos
Copyright 2003 Michael A. Davis
All rights reserved.

Redistribution and use in source and binary forms, with or without
modification, are permitted provided that the following conditions
are met:
1. Redistributions of source code must retain the above copyright
   notice, this list of conditions and the following disclaimer.
2. Redistributions in binary form must reproduce the above copyright
   notice, this list of conditions and the following disclaimer in
   the documentation and/or other materials provided with the
   distribution.
3.  The name of the author may not be used to endorse or promote
    products derived from this software without specific prior
    written permission.

THIS SOFTWARE IS PROVIDED BY THE AUTHOR "AS IS" AND ANY EXPRESS
OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED
WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE
ARE DISCLAIMED. IN NO EVENT SHALL THE AUTHOR BE LIABLE FOR ANY
DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL
DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE
GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS
INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER
IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR
OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN
IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.
==
```

E.26 Linux-PAM ライセンス

この製品には、次のソフトウェアが含まれている場合があります。

```
Linux-PAM (pam-devel, Pluggable authentication modules for Linux)

Copyright Theodore Ts'o, 1996. All rights reserved.

(For the avoidance of doubt, Oracle uses and distributes this
component under the terms below and elects not to do so under
the GPL even though the GPL is referenced as an option below.)

Redistribution and use in source and binary forms, with or
without modification, are permitted provided that the following
conditions are met:

1. Redistributions of source code must retain the above copyright
   notice, and the entire permission notice in its entirety,
   including the disclaimer of warranties.
2. Redistributions in binary form must reproduce the above copyright
   notice, this list of conditions and the following disclaimer in
   the documentation and/or other materials provided with the
   distribution.
3. The name of the author may not be used to endorse or promote
   products derived from this software without specific prior
   written permission.

ALTERNATIVELY, this product may be distributed under the terms
of the GNU Public License, in which case the provisions of the
GPL are required INSTEAD OF the above restrictions. (This clause
is necessary due to a potential bad interaction between the GPL
and the restrictions contained in a BSD-style copyright.)

THIS SOFTWARE IS PROVIDED ``AS IS'' AND ANY EXPRESS OR IMPLIED
WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES
OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE
DISCLAIMED. IN NO EVENT SHALL THE AUTHOR BE LIABLE FOR ANY DIRECT,
INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES
(INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR
SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT,
STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE)
ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED
OF THE POSSIBILITY OF SUCH DAMAGE.
```

E.27 md5 (メッセージダイジェストアルゴリズム 5) ライセンス

この製品には、次のソフトウェアが含まれている場合があります。

```
md5 (Message-Digest Algorithm 5)

 This code implements the MD5 message-digest algorithm.
 The algorithm is due to Ron Rivest.  This code was
 written by Colin Plumb in 1993, no copyright is claimed.
 This code is in the public domain; do with it what you wish.

 Equivalent code is available from RSA Data Security, Inc.
 This code has been tested against that, and is equivalent,
 except that you don't need to include two pages of legalese
 with every copy.

 The code has been modified by Mikael Ronstroem to handle
 calculating a hash value of a key that is always a multiple
 of 4 bytes long. Word 0 of the calculated 4-word hash value
 is returned as the hash value.
```

E.28 memcached ライセンス

この製品には、次のソフトウェアが含まれている場合があります。

memcached

```
Copyright (c) 2003, Danga Interactive, Inc.
All rights reserved.

Redistribution and use in source and binary forms, with or without
modification, are permitted provided that the following conditions
are met:

    * Redistributions of source code must retain the above copyright
notice, this list of conditions and the following disclaimer.

    * Redistributions in binary form must reproduce the above
copyright notice, this list of conditions and the following disclaimer
in the documentation and/or other materials provided with the
distribution.

    * Neither the name of the Danga Interactive nor the names of its
contributors may be used to endorse or promote products derived from
this software without specific prior written permission.

THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS
"AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT
LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR
A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT
OWNER OR CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL,
SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT
LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE,
DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY
THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT
(INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE
OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.
```

E.29 Memcached.pm ライセンス

この製品には、次のソフトウェアが含まれている場合があります。

Memcached.pm

```
Memcached.pm is licensed under the Perl license.

Oracle may use, redistribute and/or modify this code under the terms of
either:

 a) the GNU General Public License as published by the Free Software
Foundation; either version 1, or (at your option) any later version, or

 b) the "Artistic License" which comes with the Expect/pr code.

Oracle elects to use the GPLv2 for version of MySQL that are licensed under
the GPL.
```

Oracle elects to use the Artistic license for all other (commercial) versions
of MySQL.

A copy of the GPLv2 and the Artistic License (Perl) 1.0 must be included with
any distribution.

このコンポーネントはセクションE.14「GNU General Public License バージョン 2.0、1991 年 6 月」に基づいてライセンスが付与されています

このコンポーネントはセクションE.2「Artistic ライセンス (Perl) 1.0」に基づいてライセンスが付与されています

E.30 mkpasswd.pl ライセンス

この製品には、次のソフトウェアが含まれている場合があります。

mkpasswd.pl Perl モジュール

Copyright (C) 2003-2004 by Chris Grau

This library is free software; you can redistribute it and/or modify it under
the same terms as Perl itself, either Perl version 5.8.1 or, at your option,
any later version of Perl 5 you may have available.

The Perl 5.8.1 license (from http://www.cpan.org/src/5.0/perl-5.8.1.tar.gz - main readme file):

Perl Kit, Version 5

Copyright (C) 1993, 1994, 1995, 1996, 1997, 1998
1999, 2000, 2001, by Larry Wall and others

All rights reserved.

This program is free software; you can redistribute it and/or modify
it under the terms of either:

a) the GNU General Public License as published by the Free
Software Foundation; either version 1, or (at your option) any
later version, or

b) the "Artistic License" which comes with this Kit.

This program is distributed in the hope that it will be useful,
but WITHOUT ANY WARRANTY; without even the implied warranty of
MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See either
the GNU General Public License or the Artistic License for more details.

You should have received a copy of the Artistic License with this
Kit, in the file named "Artistic". If not, I'll be glad to provide one.

You should also have received a copy of the GNU General Public License
along with this program in the file named "Copying". If not, write to the

Free Software Foundation, Inc., 59 Temple Place, Suite 330, Boston, MA
02111-1307, USA or visit their web page on the internet at
http://www.gnu.org/copyleft/gpl.html.

For those of you that choose to use the GNU General Public License,
my interpretation of the GNU General Public License is that no Perl
script falls under the terms of the GPL unless you explicitly put
said script under the terms of the GPL yourself. Furthermore, any
object code linked with perl does not automatically fall under the
terms of the GPL, provided such object code only adds definitions
of subroutines and variables, and does not otherwise impair the
resulting interpreter from executing any standard Perl script. I
consider linking in C subroutines in this manner to be the moral
equivalent of defining subroutines in the Perl language itself. You
may sell such an object file as proprietary provided that you provide
or offer to provide the Perl source, as specified by the GNU General
Public License. (This is merely an alternate way of specifying input
to the program.) You may also sell a binary produced by the dumping of
a running Perl script that belongs to you, provided that you provide or
offer to provide the Perl source as specified by the GPL. (The
fact that a Perl interpreter and your code are in the same binary file
is, in this case, a form of mere aggregation.) This is my interpretation
of the GPL. If you still have concerns or difficulties understanding
my intent, feel free to contact me. Of course, the Artistic License
spells all this out for your protection, so you may prefer to use that.

Perl is a language that combines some of the features of C, sed, awk
and shell. See the manual page for more hype. There are also many Perl
books available, covering a wide variety of topics, from various publishers.
See pod/perlbook.pod for more information.

Please read all the directions below before you proceed any further, and
then follow them carefully.

After you have unpacked your kit, you should have all the files listed
in MANIFEST.

Installation

1) Detailed instructions are in the file "INSTALL", which you should
read if you are either installing on a system resembling Unix
or porting perl to another platform. For non-Unix platforms, see the
corresponding README.

2) Read the manual entries before running perl.

3) IMPORTANT! Help save the world! Communicate any problems and suggested
patches to perlbug@perl.org so we can keep the world in sync.
If you have a problem, there's someone else out there who either has had
or will have the same problem. It's usually helpful if you send the
output of the "myconfig" script in the main perl directory.

If you've succeeded in compiling perl, the perlbug script in the "utils"
subdirectory can be used to help mail in a bug report.

If possible, send in patches such that the patch program will apply them.
Context diffs are the best, then normal diffs. Don't send ed scripts--
I've probably changed my copy since the version you have.

The latest versions of perl are always available on the various CPAN
(Comprehensive Perl Archive Network) sites around the world.
See <URL:http://www.cpan.org/src/>.

Just a personal note: I want you to know that I create nice things like this
because it pleases the Author of my story. If this bothers you, then your
notion of Authorship needs some revision. But you can use perl anyway. :-)

The author.
==========

The "Artistic License"

Preamble

The intent of this document is to state the conditions under which a
Package may be copied, such that the Copyright Holder maintains some
semblance of artistic control over the development of the package,
while giving the users of the package the right to use and distribute
the Package in a more-or-less customary fashion, plus the right to make
reasonable modifications.

Definitions:

"Package" refers to the collection of files distributed by the
Copyright Holder, and derivatives of that collection of files
created through textual modification.

"Standard Version" refers to such a Package if it has not been
modified, or has been modified in accordance with the wishes
of the Copyright Holder as specified below.

"Copyright Holder" is whoever is named in the copyright or
copyrights for the package.

"You" is you, if you're thinking about copying or distributing
this Package.

"Reasonable copying fee" is whatever you can justify on the
basis of media cost, duplication charges, time of people involved,
and so on. (You will not be required to justify it to the
Copyright Holder, but only to the computing community at large
as a market that must bear the fee.)

"Freely Available" means that no fee is charged for the item
itself, though there may be fees involved in handling the item.
It also means that recipients of the item may redistribute it
under the same conditions they received it.

1. You may make and give away verbatim copies of the source form of the
Standard Version of this Package without restriction, provided that you
duplicate all of the original copyright notices and associated disclaimers.

2. You may apply bug fixes, portability fixes and other modifications
derived from the Public Domain or from the Copyright Holder. A Package
modified in such a way shall still be considered the Standard Version.

3. You may otherwise modify your copy of this Package in any way, provided
that you insert a prominent notice in each changed file stating how and
when you changed that file, and provided that you do at least ONE of the
following:

    a) place your modifications in the Public Domain or otherwise make them
    Freely Available, such as by posting said modifications to Usenet or
    an equivalent medium, or placing the modifications on a major archive
    site such as uunet.uu.net, or by allowing the Copyright Holder to include
    your modifications in the Standard Version of the Package.

    b) use the modified Package only within your corporation or organization.

    c) rename any non-standard executables so the names do not conflict
    with standard executables, which must also be provided, and provide
    a separate manual page for each non-standard executable that clearly
    documents how it differs from the Standard Version.

    d) make other distribution arrangements with the Copyright Holder.

4. You may distribute the programs of this Package in object code or
executable form, provided that you do at least ONE of the following:

    a) distribute a Standard Version of the executables and library files,
    together with instructions (in the manual page or equivalent) on where
    to get the Standard Version.

    b) accompany the distribution with the machine-readable source of
    the Package with your modifications.

    c) give non-standard executables non-standard names, and clearly
    document the differences in manual pages (or equivalent), together
    with instructions on where to get the Standard Version.

    d) make other distribution arrangements with the Copyright Holder.

5. You may charge a reasonable copying fee for any distribution of this
Package. You may charge any fee you choose for support of this
Package. You may not charge a fee for this Package itself. However,
you may distribute this Package in aggregate with other (possibly
commercial) programs as part of a larger (possibly commercial) software
distribution provided that you do not advertise this Package as a
product of your own. You may embed this Package's interpreter within
an executable of yours (by linking); this shall be construed as a mere
form of aggregation, provided that the complete Standard Version of the
interpreter is so embedded.

6. The scripts and library files supplied as input to or produced as
output from the programs of this Package do not automatically fall
under the copyright of this Package, but belong to whoever generated
them, and may be sold commercially, and may be aggregated with this
Package. If such scripts or library files are aggregated with this
Package via the so-called "undump" or "unexec" methods of producing a
binary executable image, then distribution of such an image shall
neither be construed as a distribution of this Package nor shall it
fall under the restrictions of Paragraphs 3 and 4, provided that you do
not represent such an executable image as a Standard Version of this
Package.

7. C subroutines (or comparably compiled subroutines in other
languages) supplied by you and linked into this Package in order to
emulate subroutines and variables of the language defined by this
Package shall not be considered part of this Package, but are the
equivalent of input as in Paragraph 6, provided these subroutines do
not change the language in any way that would cause it to fail the
regression tests for the language.

8. Aggregation of this Package with a commercial distribution is always
permitted provided that the use of this Package is embedded; that is,
when no overt attempt is made to make this Package's interfaces visible
to the end user of the commercial distribution. Such use shall not be
construed as a distribution of this Package.

9. The name of the Copyright Holder may not be used to endorse or promote
products derived from this software without specific prior written
permission.

10. THIS PACKAGE IS PROVIDED "AS IS" AND WITHOUT ANY EXPRESS OR
IMPLIED WARRANTIES, INCLUDING, WITHOUT LIMITATION, THE IMPLIED
WARRANTIES OF MERCHANTIBILITY AND FITNESS FOR A PARTICULAR PURPOSE.

The End

E.31 Node.js ライセンス

この製品には、次のソフトウェアが含まれている場合があります。

Copyright Joyent, Inc. and other Node contributors. All rights reserved.
Permission is hereby granted, free of charge, to any person obtaining a copy
of this software and associated documentation files (the "Software"), to
deal in the Software without restriction, including without limitation the
rights to use, copy, modify, merge, publish, distribute, sublicense, and/or
sell copies of the Software, and to permit persons to whom the Software is
furnished to do so, subject to the following conditions:

The above copyright notice and this permission notice shall be included in
all copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR
IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY,
FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE
AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER
LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING
FROM, OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS
IN THE SOFTWARE.

====

This license applies to all parts of Node that are not externally
maintained libraries. The externally maintained libraries used by Node are:

- V8, located at deps/v8. V8's license follows:
  """
    This license applies to all parts of V8 that are not externally
    maintained libraries. The externally maintained libraries used by V8
    are:

      - PCRE test suite, located in
        test/mjsunit/third_party/regexp-pcre.js. This is based on the
        test suite from PCRE-7.3, which is copyrighted by the University
        of Cambridge and Google, Inc. The copyright notice and license
        are embedded in regexp-pcre.js.

      - Layout tests, located in test/mjsunit/third_party. These are
        based on layout tests from webkit.org which are copyrighted by
        Apple Computer, Inc. and released under a 3-clause BSD license.

      - Strongtalk assembler, the basis of the files assembler-arm-inl.h,
        assembler-arm.cc, assembler-arm.h, assembler-ia32-inl.h,
        assembler-ia32.cc, assembler-ia32.h, assembler-x64-inl.h,
        assembler-x64.cc, assembler-x64.h, assembler-mips-inl.h,
        assembler-mips.cc, assembler-mips.h, assembler.cc and assembler.h.
        This code is copyrighted by Sun Microsystems Inc. and released
        under a 3-clause BSD license.

      - Valgrind client API header, located at third_party/valgrind/valgrind.h
        This is release under the BSD license.

    These libraries have their own licenses; we recommend you read them,
    as their terms may differ from the terms below.

    Copyright 2006-2012, the V8 project authors. All rights reserved.
    Redistribution and use in source and binary forms, with or without
    modification, are permitted provided that the following conditions are
    met:

```
      * Redistributions of source code must retain the above copyright
        notice, this list of conditions and the following disclaimer.
      * Redistributions in binary form must reproduce the above
        copyright notice, this list of conditions and the following
        disclaimer in the documentation and/or other materials provided
        with the distribution.
      * Neither the name of Google Inc. nor the names of its
        contributors may be used to endorse or promote products derived
        from this software without specific prior written permission.

    THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS
    "AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT
    LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR
    A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT
    OWNER OR CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL,
    SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT
    LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE,
    DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY
    THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT
    (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE
    OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.
  """

- libev, located at deps/uv/src/unix/ev. libev's license follows:
  """
    All files in libev are Copyright (C)2007,2008,2009 Marc Alexander Lehmann.

    Redistribution and use in source and binary forms, with or without
    modification, are permitted provided that the following conditions are
    met:

        * Redistributions of source code must retain the above copyright
          notice, this list of conditions and the following disclaimer.

        * Redistributions in binary form must reproduce the above
          copyright notice, this list of conditions and the following
          disclaimer in the documentation and/or other materials provided
          with the distribution.

    THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS
    "AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT
    LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR
    A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT
    OWNER OR CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL,
    SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT
    LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE,
    DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY
    THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT
    (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE
    OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.

    Alternatively, the contents of this package may be used under the terms
    of the GNU General Public License ("GPL") version 2 or any later version,
    in which case the provisions of the GPL are applicable instead of the
    above. If you wish to allow the use of your version of this package only
    under the terms of the GPL and not to allow others to use your version of
    this file under the BSD license, indicate your decision by deleting the
    provisions above and replace them with the notice and other provisions
    required by the GPL in this and the other files of this package. If you do
    not delete the provisions above, a recipient may use your version of this
    file under either the BSD or the GPL.
  """

- libeio, located at deps/uv/src/unix/eio. libeio's license follows:
  """
    All files in libeio are Copyright (C)2007,2008 Marc Alexander Lehmann.

    Redistribution and use in source and binary forms, with or without
    modification, are permitted provided that the following conditions are
    met:

        * Redistributions of source code must retain the above copyright
          notice, this list of conditions and the following disclaimer.

        * Redistributions in binary form must reproduce the above
          copyright notice, this list of conditions and the following
          disclaimer in the documentation and/or other materials provided
          with the distribution.

    THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS
```

```
  "AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT
  LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR
  A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT
  OWNER OR CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL,
  SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT
  LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE,
  DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY
  THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT
  (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE
  OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.

  Alternatively, the contents of this package may be used under the terms
  of the GNU General Public License ("GPL") version 2 or any later version,
  in which case the provisions of the GPL are applicable instead of the
  above. If you wish to allow the use of your version of this package only
  under the terms of the GPL and not to allow others to use your version of
  this file under the BSD license, indicate your decision by deleting the
  provisions above and replace them with the notice and other provisions
  required by the GPL in this and the other files of this package. If you do
  not delete the provisions above, a recipient may use your version of this
  file under either the BSD or the GPL.
 """

- WAF build system, located at tools/waf*. WAF's license follows:
 """
  Copyright Thomas Nagy, 2005-2011

  Redistribution and use in source and binary forms, with or without
  modification, are permitted provided that the following conditions
  are met:

  1. Redistributions of source code must retain the above copyright
     notice, this list of conditions and the following disclaimer.

  2. Redistributions in binary form must reproduce the above copyright
     notice, this list of conditions and the following disclaimer in the
     documentation and/or other materials provided with the distribution.

  3. The name of the author may not be used to endorse or promote products
     derived from this software without specific prior written permission.

  THIS SOFTWARE IS PROVIDED BY THE AUTHOR "AS IS" AND ANY EXPRESS OR
  IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED
  WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE
  DISCLAIMED. IN NO EVENT SHALL THE AUTHOR BE LIABLE FOR ANY DIRECT,
  INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES
  (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR
  SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
  HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT,
  STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING
  IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE
  POSSIBILITY OF SUCH DAMAGE.
 """

- C-Ares, an asynchronous DNS client, located at deps/uv/src/ares. C-Ares license
  follows
 """
   /* Copyright 1998 by the Massachusetts Institute of Technology.
    *
    * Permission to use, copy, modify, and distribute this
    * software and its documentation for any purpose and without
    * fee is hereby granted, provided that the above copyright
    * notice appear in all copies and that both that copyright
    * notice and this permission notice appear in supporting
    * documentation, and that the name of M.I.T. not be used in
    * advertising or publicity pertaining to distribution of the
    * software without specific, written prior permission.
    * M.I.T. makes no representations about the suitability of
    * this software for any purpose.  It is provided "as is"
    * without express or implied warranty.
 """


- HTTP Parser, located at deps/http_parser. HTTP Parser's license follows:
 """
  http_parser.c is based on src/http/ngx_http_parse.c from NGINX copyright
  Igor Sysoev.

  Additional changes are licensed under the same terms as NGINX and
  copyright Joyent, Inc. and other Node contributors. All rights reserved.
```

```
  Permission is hereby granted, free of charge, to any person obtaining a copy
  of this software and associated documentation files (the "Software"), to
  deal in the Software without restriction, including without limitation the
  rights to use, copy, modify, merge, publish, distribute, sublicense, and/or
  sell copies of the Software, and to permit persons to whom the Software is
  furnished to do so, subject to the following conditions:

  The above copyright notice and this permission notice shall be included in
  all copies or substantial portions of the Software.

  THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR
  IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY,
  FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE
  AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER
  LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING
  FROM, OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS
  IN THE SOFTWARE.
 """

- tools/cpplint.py is a C++ linter. Its license follows:
 """
  # Copyright (c) 2009 Google Inc. All rights reserved.
  #
  # Redistribution and use in source and binary forms, with or without
  # modification, are permitted provided that the following conditions are
  # met:
  #
  #    * Redistributions of source code must retain the above copyright
  # notice, this list of conditions and the following disclaimer.
  #    * Redistributions in binary form must reproduce the above
  # copyright notice, this list of conditions and the following disclaimer
  # in the documentation and/or other materials provided with the
  # distribution.
  #    * Neither the name of Google Inc. nor the names of its
  # contributors may be used to endorse or promote products derived from
  # this software without specific prior written permission.
  #
  # THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS
  # "AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT
  # LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR
  # A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT
  # OWNER OR CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL,
  # SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT
  # LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE,
  # DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY
  # THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT
  # (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE
  # OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.
 """

- lib/buffer_ieee754.js. Its license follows:
 """
  // Copyright (c) 2008, Fair Oaks Labs, Inc.
  // All rights reserved.
  //
  // Redistribution and use in source and binary forms, with or without
  // modification, are permitted provided that the following conditions are met:
  //
  //  * Redistributions of source code must retain the above copyright notice,
  //    this list of conditions and the following disclaimer.
  //
  //  * Redistributions in binary form must reproduce the above copyright notice,
  //    this list of conditions and the following disclaimer in the documentation
  //    and/or other materials provided with the distribution.
  //
  //  * Neither the name of Fair Oaks Labs, Inc. nor the names of its contributors
  //    may be used to endorse or promote products derived from this software
  //    without specific prior written permission.
  //
  // THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS "AS IS"
  // AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE
  // IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE
  // ARE DISCLAIMED.  IN NO EVENT SHALL THE COPYRIGHT OWNER OR CONTRIBUTORS BE
  // LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR
  // CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF
  // SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS
  // INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN
  // CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE)
  // ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE
```

```
  // POSSIBILITY OF SUCH DAMAGE.
 """

- lib/punycode.js is copyright 2011 Mathias Bynens <http://mathiasbynens.be/>
  and released under the MIT license.
 """
   * Punycode.js <http://mths.be/punycode>
   * Copyright 2011 Mathias Bynens <http://mathiasbynens.be/>
   * Available under MIT license <http://mths.be/mit>
 """

- tools/gyp GYP is a meta-build system. GYP's license follows:
 """
   Copyright (c) 2009 Google Inc. All rights reserved.

   Redistribution and use in source and binary forms, with or without
   modification, are permitted provided that the following conditions are
   met:

      * Redistributions of source code must retain the above copyright
   notice, this list of conditions and the following disclaimer.
      * Redistributions in binary form must reproduce the above
   copyright notice, this list of conditions and the following disclaimer
   in the documentation and/or other materials provided with the
   distribution.
      * Neither the name of Google Inc. nor the names of its
   contributors may be used to endorse or promote products derived from
   this software without specific prior written permission.

   THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS
   "AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT
   LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR
   A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT
   OWNER OR CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL,
   SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT
   LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE,
   DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY
   THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT
   (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE
   OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.
 """

- Zlib at deps/zlib. zlib's license follows
 """
   /* zlib.h -- interface of the 'zlib' general purpose compression library
     version 1.2.4, March 14th, 2010

     Copyright (C) 1995-2010 Jean-loup Gailly and Mark Adler

     This software is provided 'as-is', without any express or implied
     warranty.  In no event will the authors be held liable for any damages
     arising from the use of this software.

     Permission is granted to anyone to use this software for any purpose,
     including commercial applications, and to alter it and redistribute it
     freely, subject to the following restrictions:

     1. The origin of this software must not be misrepresented; you must not
        claim that you wrote the original software. If you use this software
        in a product, an acknowledgment in the product documentation would be
        appreciated but is not required.
     2. Altered source versions must be plainly marked as such, and must not be
        misrepresented as being the original software.
     3. This notice may not be removed or altered from any source distribution.

     Jean-loup Gailly
     Mark Adler

   */
 """

- tools/doc/node_modules/marked Marked is a Markdown parser. Marked's
  license follows
 """
@    Copyright (c) 2011-2012, Christopher Jeffrey (https://github.com/chjj/)

   Permission is hereby granted, free of charge, to any person obtaining a copy
   of this software and associated documentation files (the "Software"), to deal
   in the Software without restriction, including without limitation the rights
   to use, copy, modify, merge, publish, distribute, sublicense, and/or sell
```

```
  copies of the Software, and to permit persons to whom the Software is
  furnished to do so, subject to the following conditions:

  The above copyright notice and this permission notice shall be included in
  all copies or substantial portions of the Software.

  THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR
  IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY,
  FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE
  AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER
  LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM,
  OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN
  THE SOFTWARE.
"""
```

E.32 nt_servc (Windows NT Service クラスライブラリ) ライセンス

この製品には、次のソフトウェアが含まれている場合があります。

nt_servc (Windows NT Service クラスライブラリ)

```
Windows NT Service class library
Copyright Abandoned 1998 Irena Pancirov - Irnet Snc
This file is public domain and comes with NO WARRANTY of any kind
```

E.33 OpenPAM ライセンス

この製品には、次のソフトウェアが含まれている場合があります。

OpenPAM

```
Copyright (c) 2002-2003 Networks Associates Technology, Inc.
Copyright (c) 2004-2007 Dag-Erling Smørgrav
All rights reserved.

This software was developed for the FreeBSD Project by
ThinkSec AS and Network Associates Laboratories, the
Security Research Division of Network Associates, Inc.
under DARPA/SPAWAR contract N66001-01-C-8035 ("CBOSS"),
as part of the DARPA CHATS research program.

Redistribution and use in source and binary forms,
with or without modification, are permitted provided
that the following conditions are met:

1. Redistributions of source code must retain the above
   copyright notice, this list of conditions and the
   following disclaimer.
2. Redistributions in binary form must reproduce the
   above copyright notice, this list of conditions and
   the following disclaimer in the documentation and/or
   other materials provided with the distribution.
3. The name of the author may not be used to endorse or
   promote products derived from this software without
   specific prior written permission.

THIS SOFTWARE IS PROVIDED BY THE AUTHOR AND CONTRIBUTORS
"AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING,
BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY
AND FITNESS FOR A PARTICULAR PURPOSE ARE DISCLAIMED.  IN
NO EVENT SHALL THE AUTHOR OR CONTRIBUTORS BE LIABLE FOR
ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR
CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO,
PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE,
DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED
AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT
LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE)
ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN
IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.
```

E.34 OpenSSL v1.0 ライセンス

この製品には、次のソフトウェアが含まれている場合があります。

OpenSSL v1.0

```
NOTE: Does not apply to GPL licensed server (OpenSSL is not shipped with it)
```

LICENSE ISSUES
==============
The OpenSSL toolkit stays under a dual license, i.e. both the conditions of
the OpenSSL License and the original SSLeay license apply to the toolkit. See
below for the actual license texts. Actually both licenses are BSD-style Open
Source licenses. In case of any license issues related to OpenSSL please
contact openssl-core@openssl.org.

OpenSSL License

/ ==
Copyright (c) 1998-2008 The OpenSSL Project.
All rights reserved.
Redistribution and use in source and binary forms, with or without
modification, are permitted provided that the following conditions are met:

1. Redistributions of source code must retain the above copyright notice,
this list of conditions and the following disclaimer.
2. Redistributions in binary form must reproduce the above copyright notice,
this list of conditions and the following disclaimer in the documentation
and/or other materials provided with the distribution.
3. All advertising materials mentioning features or use of this software must
display the following acknowledgment: "This product includes software
developed by the OpenSSL Project for use in the OpenSSL Toolkit. (Link1 /)"

4. The names "OpenSSL Toolkit" and "OpenSSL Project" must not be used to
endorse or promote products derived from this software without prior written
permission. For written permission, please contact openssl-core@openssl.org.

5. Products derived from this software may not be called "OpenSSL" nor may
"OpenSSL" appear in their names without prior written permission of the
OpenSSL Project.
6. Redistributions of any form whatsoever must retain the following
acknowledgment: "This product includes software developed by the OpenSSL
Project for use in the OpenSSL Toolkit (Link2 /)"

THIS SOFTWARE IS PROVIDED BY THE OpenSSL PROJECT ``AS IS'' AND ANY EXPRESSED
OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES
OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE DISCLAIMED. IN
NO EVENT SHALL THE OpenSSL PROJECT OR ITS CONTRIBUTORS BE LIABLE FOR ANY
DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES
(INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES;
 LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND
ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT
(INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF
THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.
==
This product includes cryptographic software written by Eric Young
(eay@cryptsoft.com). This product includes software written by Tim Hudson
(tjh@cryptsoft.com).

Original SSLeay License

/ Copyright (C) 1995-1998 Eric Young (eay@cryptsoft.com)
All rights reserved.
This package is an SSL implementation written by Eric Young
(eay@cryptsoft.com). The implementation was written so as to conform with
Netscapes SSL.   This library is free for commercial and non-commercial use
as long as the following conditions are aheared to. The following conditions
apply to all code found in this distribution, be it the RC4, RSA, lhash,
DES, etc., code; not just the SSL code. The SSL documentation included with
this distribution is covered by the same copyright terms except that the
holder is Tim Hudson (tjh@cryptsoft.com).   Copyright remains Eric Young's,
and as such any Copyright notices in the code are not to be removed. If this
package is used in a product, Eric Young should be given attribution as the
author of the parts of the library used. This can be in the form of a
textual message at program startup or in documentation (online or textual)
provided with the package.   Redistribution and use in source and binary
forms, with or without modification, are permitted provided that the
following conditions are met: 1. Redistributions of source code must retain
the copyright notice, this list of conditions and the following disclaimer.
2. Redistributions in binary form must reproduce the above copyright notice,
this list of conditions and the following disclaimer in the documentation
and/or other materials provided with the distribution. 3. All advertising
materials mentioning features or use of this software must display the
following acknowledgement: "This product includes cryptographic software
written by Eric Young (eay@cryptsoft.com)" The word 'cryptographic' can be
left out if the routines from the library being used are not cryptographic

```
related :-).  4. If you include any Windows specific code (or a derivative
thereof) from the apps directory (application code) you must include an
acknowledgement:  "This product includes software written by Tim Hudson
(tjh@cryptsoft.com)"   THIS SOFTWARE IS PROVIDED BY ERIC YOUNG ``AS IS'' AND
ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED
WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE  ARE
DISCLAIMED. IN NO EVENT SHALL THE AUTHOR OR CONTRIBUTORS BE LIABLE  FOR ANY
DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL  DAMAGES
(INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS  OR SERVICES;
LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)  HOWEVER CAUSED AND
ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT  LIABILITY, OR TORT
(INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY  OUT OF THE USE OF
THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF  SUCH DAMAGE.   The
license and distribution terms for any publically available version or
derivative of this code cannot be changed. i.e. this code cannot simply be
copied and put under another distribution license  [including the GNU Public
License.]
```

E.35 Paramiko ライセンス

この製品には、次のソフトウェアが含まれている場合があります。

```
Paramiko

You are receiving a copy of Paramiko in both source and object code. The
terms of the Oracle license do NOT apply to the Paramiko program; it is
licensed under the following license, separately from the Oracle
programs you receive. If you do not wish to install this program, you
may delete the Paramiko folder and all its contents.
```

このコンポーネントはセクションE.16「GNU Lesser General Public License バージョン 2.1、1999 年 2 月」に基づいてライセンスが付与されています。

E.36 Percona Multiple I/O スレッドパッチライセンス

この製品には、次のソフトウェアが含まれている場合があります。

Percona Multiple I/O スレッドパッチ

```
Copyright (c) 2008, 2009 Percona Inc
All rights reserved.

Redistribution and use of this software in source and binary forms,
with or without modification, are permitted provided that the
following conditions are met:

 * Redistributions of source code must retain the above copyright
   notice, this list of conditions and the following disclaimer.
 * Redistributions in binary form must reproduce the above copyright
   notice, this list of conditions and the following disclaimer in the
   documentation and/or other materials provided with the distribution.
 * Neither the name of Percona Inc. nor the names of its contributors
   may be used to endorse or promote products derived from this software
   without specific prior written permission of Percona Inc.

THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS
"AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT
LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS
FOR A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE
COPYRIGHT OWNER OR CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT,
INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING,
BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES;
LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER
CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT
LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN
ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE
POSSIBILITY OF SUCH DAMAGE.
```

E.37 Pion ライセンス

この Oracle 製品は Pion を含んでいるか、参照します。これは次の条件下で Oracle にライセンスが付与されています。

```
Boost Software License - Version 1.0 - August 17th, 2003

Permission is hereby granted, free of charge, to any person or organization
```

obtaining a copy of the software and accompanying documentation covered by
this license (the "Software") to use, reproduce, display, distribute,
execute, and transmit the Software, and to prepare derivative works of the
Software, and to permit third-parties to whom the Software is furnished to
do so, all subject to the following:

The copyright notices in the Software and this entire statement, including
the above license grant, this restriction and the following disclaimer,
must be included in all copies of the Software, in whole or in part, and
all derivative works of the Software, unless such copies or derivative
works are solely in the form of machine-executable object code generated by
a source language processor.

THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR
IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY,
FITNESS FOR A PARTICULAR PURPOSE, TITLE AND NON-INFRINGEMENT. IN NO EVENT
SHALL THE COPYRIGHT HOLDERS OR ANYONE DISTRIBUTING THE SOFTWARE BE LIABLE
FOR ANY DAMAGES OR OTHER LIABILITY, WHETHER IN CONTRACT, TORT OR OTHERWISE,
ARISING FROM, OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER
DEALINGS IN THE SOFTWARE.

E.38 Python ライセンス

この製品には、次のソフトウェアが含まれている場合があります。

Python Programming Language

This is the official license for the Python 2.7 release:

A. HISTORY OF THE SOFTWARE
Python was created in the early 1990s by Guido van Rossum at Stichting
Mathematisch Centrum (CWI, see http://www.cwi.nl) in the Netherlands
as a successor of a language called ABC. Guido remains Python's
principal author, although it includes many contributions from others.

In 1995, Guido continued his work on Python at the Corporation for
National Research Initiatives (CNRI, see http://www.cnri.reston.va.us)
in Reston, Virginia where he released several versions of the
software.

In May 2000, Guido and the Python core development team moved to
BeOpen.com to form the BeOpen PythonLabs team. In October of the same
year, the PythonLabs team moved to Digital Creations (now Zope
Corporation, see http://www.zope.com). In 2001, the Python Software
Foundation (PSF, see http://www.python.org/psf/) was formed, a
non-profit organization created specifically to own Python-related
Intellectual Property. Zope Corporation is a sponsoring member of
the PSF.

All Python releases are Open Source (see http://www.opensource.org for
the Open Source Definition). Historically, most, but not all, Python
releases have also been GPL-compatible; the table below summarizes
the various releases.

Release	Derived from	Year	Owner	GPL-compatible? (1)
0.9.0 thru 1.2		1991-1995	CWI	yes
1.3 thru 1.5.2	1.2	1995-1999	CNRI	yes
1.6	1.5.2	2000	CNRI	no
2.0	1.6	2000	BeOpen.com	no
1.6.1	1.6	2001	CNRI	yes (2)
2.1	2.0+1.6.1	2001	PSF	no
2.0.1	2.0+1.6.1	2001	PSF	yes
2.1.1	2.1+2.0.1	2001	PSF	yes
2.2	2.1.1	2001	PSF	yes
2.1.2	2.1.1	2002	PSF	yes
2.1.3	2.1.2	2002	PSF	yes
2.2.1	2.2	2002	PSF	yes
2.2.2	2.2.1	2002	PSF	yes
2.2.3	2.2.2	2003	PSF	yes
2.3	2.2.2	2002-2003	PSF	yes
2.3.1	2.3	2002-2003	PSF	yes
2.3.2	2.3.1	2002-2003	PSF	yes
2.3.3	2.3.2	2002-2003	PSF	yes
2.3.4	2.3.3	2004	PSF	yes
2.3.5	2.3.4	2005	PSF	yes
2.4	2.3	2004	PSF	yes
2.4.1	2.4	2005	PSF	yes

```
   2.4.2         2.4.1      2005       PSF        yes
   2.4.3         2.4.2      2006       PSF        yes
   2.5           2.4        2006       PSF        yes
   2.5.1         2.5        2007       PSF        yes
   2.5.2         2.5.1      2008       PSF        yes
   2.5.3         2.5.2      2008       PSF        yes
   2.6           2.5        2008       PSF        yes
   2.6.1         2.6        2008       PSF        yes
   2.6.2         2.6.1      2009       PSF        yes
   2.6.3         2.6.2      2009       PSF        yes
   2.6.4         2.6.3      2010       PSF        yes
   2.7           2.6        2010       PSF        yes


Footnotes:

(1) GPL-compatible doesn't mean that we're distributing Python under
    the GPL.  All Python licenses, unlike the GPL, let you distribute
    a modified version without making your changes open source.  The
    GPL-compatible licenses make it possible to combine Python with
    other software that is released under the GPL; the others don't.

(2) According to Richard Stallman, 1.6.1 is not GPL-compatible,
    because its license has a choice of law clause.  According to
    CNRI, however, Stallman's lawyer has told CNRI's lawyer that 1.6.1
    is "not incompatible" with the GPL.

Thanks to the many outside volunteers who have worked under Guido's
direction to make these releases possible.

B. TERMS AND CONDITIONS FOR ACCESSING OR OTHERWISE USING PYTHON

PYTHON SOFTWARE FOUNDATION LICENSE VERSION 2
--------------------------------------------

1. This LICENSE AGREEMENT is between the Python Software Foundation
("PSF"), and the Individual or Organization ("Licensee") accessing and
otherwise using this software ("Python") in source or binary form and
its associated documentation.

2. Subject to the terms and conditions of this License Agreement, PSF
hereby grants Licensee a nonexclusive, royalty-free, world-wide
license to reproduce, analyze, test, perform and/or display publicly,
prepare derivative works, distribute, and otherwise use Python
alone or in any derivative version, provided, however, that PSF's
License Agreement and PSF's notice of copyright, i.e., "Copyright (c)
2001, 2002, 2003, 2004, 2005, 2006 Python Software Foundation; All Rights
Reserved" are retained in Python alone or in any derivative version
prepared by Licensee.

3. In the event Licensee prepares a derivative work that is based on
or incorporates Python or any part thereof, and wants to make
the derivative work available to others as provided herein, then
Licensee hereby agrees to include in any such work a brief summary of
the changes made to Python.

4. PSF is making Python available to Licensee on an "AS IS"
basis.  PSF MAKES NO REPRESENTATIONS OR WARRANTIES, EXPRESS OR
IMPLIED.  BY WAY OF EXAMPLE, BUT NOT LIMITATION, PSF MAKES NO AND
DISCLAIMS ANY REPRESENTATION OR WARRANTY OF MERCHANTABILITY OR FITNESS
FOR ANY PARTICULAR PURPOSE OR THAT THE USE OF PYTHON WILL NOT
INFRINGE ANY THIRD PARTY RIGHTS.

5. PSF SHALL NOT BE LIABLE TO LICENSEE OR ANY OTHER USERS OF PYTHON
FOR ANY INCIDENTAL, SPECIAL, OR CONSEQUENTIAL DAMAGES OR LOSS AS
A RESULT OF MODIFYING, DISTRIBUTING, OR OTHERWISE USING PYTHON,
OR ANY DERIVATIVE THEREOF, EVEN IF ADVISED OF THE POSSIBILITY THEREOF.

6. This License Agreement will automatically terminate upon a material
breach of its terms and conditions.

7. Nothing in this License Agreement shall be deemed to create any
relationship of agency, partnership, or joint venture between PSF and
Licensee.  This License Agreement does not grant permission to use PSF
trademarks or trade name in a trademark sense to endorse or promote
products or services of Licensee, or any third party.

8. By copying, installing or otherwise using Python, Licensee
agrees to be bound by the terms and conditions of this License
Agreement.
```

BEOPEN.COM LICENSE AGREEMENT FOR PYTHON 2.0

BEOPEN PYTHON OPEN SOURCE LICENSE AGREEMENT VERSION 1

1. This LICENSE AGREEMENT is between BeOpen.com ("BeOpen"), having an
office at 160 Saratoga Avenue, Santa Clara, CA 95051, and the
Individual or Organization ("Licensee") accessing and otherwise using
this software in source or binary form and its associated
documentation ("the Software").

2. Subject to the terms and conditions of this BeOpen Python License
Agreement, BeOpen hereby grants Licensee a non-exclusive,
royalty-free, world-wide license to reproduce, analyze, test, perform
and/or display publicly, prepare derivative works, distribute, and
otherwise use the Software alone or in any derivative version,
provided, however, that the BeOpen Python License is retained in the
Software, alone or in any derivative version prepared by Licensee.

3. BeOpen is making the Software available to Licensee on an "AS IS"
basis. BEOPEN MAKES NO REPRESENTATIONS OR WARRANTIES, EXPRESS OR
IMPLIED. BY WAY OF EXAMPLE, BUT NOT LIMITATION, BEOPEN MAKES NO AND
DISCLAIMS ANY REPRESENTATION OR WARRANTY OF MERCHANTABILITY OR FITNESS
FOR ANY PARTICULAR PURPOSE OR THAT THE USE OF THE SOFTWARE WILL NOT
INFRINGE ANY THIRD PARTY RIGHTS.

4. BEOPEN SHALL NOT BE LIABLE TO LICENSEE OR ANY OTHER USERS OF THE
SOFTWARE FOR ANY INCIDENTAL, SPECIAL, OR CONSEQUENTIAL DAMAGES OR LOSS
AS A RESULT OF USING, MODIFYING OR DISTRIBUTING THE SOFTWARE, OR ANY
DERIVATIVE THEREOF, EVEN IF ADVISED OF THE POSSIBILITY THEREOF.

5. This License Agreement will automatically terminate upon a material
breach of its terms and conditions.

6. This License Agreement shall be governed by and interpreted in all
respects by the law of the State of California, excluding conflict of
law provisions. Nothing in this License Agreement shall be deemed to
create any relationship of agency, partnership, or joint venture
between BeOpen and Licensee. This License Agreement does not grant
permission to use BeOpen trademarks or trade names in a trademark
sense to endorse or promote products or services of Licensee, or any
third party. As an exception, the "BeOpen Python" logos available at
http://www.pythonlabs.com/logos.html may be used according to the
permissions granted on that web page.

7. By copying, installing or otherwise using the software, Licensee
agrees to be bound by the terms and conditions of this License
Agreement.

CNRI LICENSE AGREEMENT FOR PYTHON 1.6.1
----

1. This LICENSE AGREEMENT is between the Corporation for National
Research Initiatives, having an office at 1895 Preston White Drive,
Reston, VA 20191 ("CNRI"), and the Individual or Organization
("Licensee") accessing and otherwise using Python 1.6.1 software in
source or binary form and its associated documentation.

2. Subject to the terms and conditions of this License Agreement, CNRI
hereby grants Licensee a nonexclusive, royalty-free, world-wide
license to reproduce, analyze, test, perform and/or display publicly,
prepare derivative works, distribute, and otherwise use Python 1.6.1
alone or in any derivative version, provided, however, that CNRI's
License Agreement and CNRI's notice of copyright, i.e., "Copyright (c)
1995-2001 Corporation for National Research Initiatives; All Rights
Reserved" are retained in Python 1.6.1 alone or in any derivative
version prepared by Licensee. Alternately, in lieu of CNRI's License
Agreement, Licensee may substitute the following text (omitting the
quotes): "Python 1.6.1 is made available subject to the terms and
conditions in CNRI's License Agreement. This Agreement together with
Python 1.6.1 may be located on the Internet using the following
unique, persistent identifier (known as a handle): 1895.22/1013. This
Agreement may also be obtained from a proxy server on the Internet
using the following URL: http://hdl.handle.net/1895.22/1013".

3. In the event Licensee prepares a derivative work that is based on
or incorporates Python 1.6.1 or any part thereof, and wants to make

the derivative work available to others as provided herein, then
Licensee hereby agrees to include in any such work a brief summary of
the changes made to Python 1.6.1.

4. CNRI is making Python 1.6.1 available to Licensee on an "AS IS"
basis. CNRI MAKES NO REPRESENTATIONS OR WARRANTIES, EXPRESS OR
IMPLIED. BY WAY OF EXAMPLE, BUT NOT LIMITATION, CNRI MAKES NO AND
DISCLAIMS ANY REPRESENTATION OR WARRANTY OF MERCHANTABILITY OR FITNESS
FOR ANY PARTICULAR PURPOSE OR THAT THE USE OF PYTHON 1.6.1 WILL NOT
INFRINGE ANY THIRD PARTY RIGHTS.

5. CNRI SHALL NOT BE LIABLE TO LICENSEE OR ANY OTHER USERS OF PYTHON
1.6.1 FOR ANY INCIDENTAL, SPECIAL, OR CONSEQUENTIAL DAMAGES OR LOSS AS
A RESULT OF MODIFYING, DISTRIBUTING, OR OTHERWISE USING PYTHON 1.6.1,
OR ANY DERIVATIVE THEREOF, EVEN IF ADVISED OF THE POSSIBILITY THEREOF.

6. This License Agreement will automatically terminate upon a material
breach of its terms and conditions.

7. This License Agreement shall be governed by the federal
intellectual property law of the United States, including without
limitation the federal copyright law, and, to the extent such
U.S. federal law does not apply, by the law of the Commonwealth of
Virginia, excluding Virginia's conflict of law provisions.
Notwithstanding the foregoing, with regard to derivative works based
on Python 1.6.1 that incorporate non-separable material that was
previously distributed under the GNU General Public License (GPL), the
law of the Commonwealth of Virginia shall govern this License
Agreement only as to issues arising under or with respect to
Paragraphs 4, 5, and 7 of this License Agreement. Nothing in this
License Agreement shall be deemed to create any relationship of
agency, partnership, or joint venture between CNRI and Licensee. This
License Agreement does not grant permission to use CNRI trademarks or
trade name in a trademark sense to endorse or promote products or
services of Licensee, or any third party.

8. By clicking on the "ACCEPT" button where indicated, or by copying,
installing or otherwise using Python 1.6.1, Licensee agrees to be
bound by the terms and conditions of this License Agreement.

ACCEPT

CWI LICENSE AGREEMENT FOR PYTHON 0.9.0 THROUGH 1.2

Copyright (c) 1991 - 1995, Stichting Mathematisch Centrum Amsterdam,
The Netherlands. All rights reserved.

Permission to use, copy, modify, and distribute this software and its
documentation for any purpose and without fee is hereby granted,
provided that the above copyright notice appear in all copies and that
both that copyright notice and this permission notice appear in
supporting documentation, and that the name of Stichting Mathematisch
Centrum or CWI not be used in advertising or publicity pertaining to
distribution of the software without specific, written prior
permission.

STICHTING MATHEMATISCH CENTRUM DISCLAIMS ALL WARRANTIES WITH REGARD TO
THIS SOFTWARE, INCLUDING ALL IMPLIED WARRANTIES OF MERCHANTABILITY AND
FITNESS, IN NO EVENT SHALL STICHTING MATHEMATISCH CENTRUM BE LIABLE
FOR ANY SPECIAL, INDIRECT OR CONSEQUENTIAL DAMAGES OR ANY DAMAGES
WHATSOEVER RESULTING FROM LOSS OF USE, DATA OR PROFITS, WHETHER IN AN
ACTION OF CONTRACT, NEGLIGENCE OR OTHER TORTIOUS ACTION, ARISING OUT
OF OR IN CONNECTION WITH THE USE OR PERFORMANCE OF THIS SOFTWARE.

Licenses and Acknowledgements for Incorporated Software
==========

This section is an incomplete, but growing list of licenses and acknowledgements
for third-party software incorporated in the Python distribution.

Mersenne Twister
==========
The _random module includes code based on a download from
  http://www.math.keio.ac.jp/ matumoto/MT2002/emt19937ar.html.
The following are the verbatim comments from the original code:

A C-program for MT19937, with initialization improved 2002/1/26.
Coded by Takuji Nishimura and Makoto Matsumoto.

```
Before using, initialize the state by using init_genrand(seed)
or init_by_array(init_key, key_length).

Copyright (C) 1997 - 2002, Makoto Matsumoto and Takuji Nishimura,
All rights reserved.

Redistribution and use in source and binary forms, with or without
modification, are permitted provided that the following conditions
are met:

 1. Redistributions of source code must retain the above copyright
    notice, this list of conditions and the following disclaimer.

 2. Redistributions in binary form must reproduce the above copyright
    notice, this list of conditions and the following disclaimer in the
    documentation and/or other materials provided with the
    distribution.

 3. The names of its contributors may not be used to endorse or promote
    products derived from this software without specific prior written
    permission.

THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS
"AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT
LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR
A PARTICULAR PURPOSE ARE DISCLAIMED.  IN NO EVENT SHALL THE COPYRIGHT OWNER OR
CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL,
EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO,
PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR
PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF
LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING
NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS
SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.

Any feedback is very welcome.
http://www.math.keio.ac.jp/matumoto/emt.html
email: matumoto@math.keio.ac.jp

Sockets
=======
The socket module uses the functions, getaddrinfo(), and getnameinfo(), which
are coded in separate source files from the WIDE Project, http://www.wide.ad.jp/.

Copyright (C) 1995, 1996, 1997, and 1998 WIDE Project.
All rights reserved.

Redistribution and use in source and binary forms, with or without
modification, are permitted provided that the following conditions
are met:
1. Redistributions of source code must retain the above copyright
   notice, this list of conditions and the following disclaimer.
2. Redistributions in binary form must reproduce the above copyright
   notice, this list of conditions and the following disclaimer in the
   documentation and/or other materials provided with the distribution.
3. Neither the name of the project nor the names of its contributors
   may be used to endorse or promote products derived from this
   software without specific prior written permission.

THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS
"AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT
LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR
A PARTICULAR PURPOSE ARE DISCLAIMED.  IN NO EVENT SHALL THE COPYRIGHT OWNER OR
CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL,
EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO,
PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR
PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF
LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING
NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS
SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.

Floating point exception control
================================
The source for the fpectl module includes the following notice:

  ---------------------------------------------------------------------
 /                    Copyright (c) 1996.                            \
|         The Regents of the University of California.                |
|                    All rights reserved.                           |
|                                                                   |
```

```
|  Permission to use, copy, modify, and distribute this software for  |
|  any purpose without fee is hereby granted, provided that this en-  |
|  tire notice is included in all copies of any software which is or  |
|  includes  a  copy  or  modification  of  this software and in all  |
|  copies of the supporting documentation for such software.          |
|                                                                     |
|  This  work was produced at the University of California, Lawrence  |
|  Livermore National Laboratory under  contract  no.  W-7405-ENG-48  |
|  between  the  U.S.  Department  of  Energy and The Regents of the  |
|  University of California for the operation of UC LLNL.             |
|                                                                     |
|                              DISCLAIMER                             |
|                                                                     |
|  This  software was prepared as an account of work sponsored by an  |
|  agency of the United States Government. Neither the United States  |
|  Government  nor the University of California nor any of their em-  |
|  ployees, makes any warranty, express or implied, or  assumes  any  |
|  liability  or  responsibility  for the accuracy, completeness, or  |
|  usefulness of any information,  apparatus,  product,  or  process  |
|  disclosed,  or  represents  that  its  use  would  not  infringe   |
|  privately-owned rights. Reference herein to any specific  commer-  |
|  cial  products,  process,  or  service  by trade name, trademark,  |
|  manufacturer, or otherwise, does not  necessarily  constitute  or  |
|  imply  its endorsement, recommendation, or favoring by the United  |
|  States Government or the University of California. The views  and  |
|  opinions  of authors expressed herein do not necessarily state or  |
|  reflect those of the United States Government or  the  University  |
|  of  California,  and shall not be used for advertising or product  |
\  endorsement purposes.                                             /
 ---------------------------------------------------------------------
```

MD5 message digest algorithm
============================
The source code for the md5 module contains the following notice:

Copyright (C) 1999, 2002 Aladdin Enterprises. All rights reserved.

This software is provided 'as-is', without any express or implied
warranty. In no event will the authors be held liable for any damages
arising from the use of this software.

Permission is granted to anyone to use this software for any purpose,
including commercial applications, and to alter it and redistribute it
freely, subject to the following restrictions:

1. The origin of this software must not be misrepresented; you must not
   claim that you wrote the original software. If you use this software
   in a product, an acknowledgment in the product documentation would
   be appreciated but is not required.
2. Altered source versions must be plainly marked as such, and must not
   be misrepresented as being the original software.
3. This notice may not be removed or altered from any source
   distribution.

L. Peter Deutsch
ghost@aladdin.com

Independent implementation of MD5 (RFC 1321).

This code implements the MD5 Algorithm defined in RFC 1321, whose
text is available at
      http://www.ietf.org/rfc/rfc1321.txt
The code is derived from the text of the RFC, including the test suite
(section A.5) but excluding the rest of Appendix A. It does not include
any code or documentation that is identified in the RFC as being
copyrighted.

The original and principal author of md5.h is L. Peter Deutsch
<ghost@aladdin.com>. Other authors are noted in the change history
that follows (in reverse chronological order):

2002-04-13 lpd Removed support for non-ANSI compilers; removed
      references to Ghostscript; clarified derivation from RFC 1321;
      now handles byte order either statically or dynamically.
1999-11-04 lpd Edited comments slightly for automatic TOC extraction.
1999-10-18 lpd Fixed typo in header comment (ansi2knr rather than md5);
      added conditionalization for C++ compilation from Martin
      Purschke <purschke@bnl.gov>.
1999-05-03 lpd Original version.

Asynchronous socket services
==========
The asynchat and asyncore modules contain the following notice:

Copyright 1996 by Sam Rushing

All Rights Reserved

Permission to use, copy, modify, and distribute this software and
its documentation for any purpose and without fee is hereby
granted, provided that the above copyright notice appear in all
copies and that both that copyright notice and this permission
notice appear in supporting documentation, and that the name of Sam
Rushing not be used in advertising or publicity pertaining to
distribution of the software without specific, written prior
permission.

SAM RUSHING DISCLAIMS ALL WARRANTIES WITH REGARD TO THIS SOFTWARE,
INCLUDING ALL IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS, IN
NO EVENT SHALL SAM RUSHING BE LIABLE FOR ANY SPECIAL, INDIRECT OR
CONSEQUENTIAL DAMAGES OR ANY DAMAGES WHATSOEVER RESULTING FROM LOSS
OF USE, DATA OR PROFITS, WHETHER IN AN ACTION OF CONTRACT,
NEGLIGENCE OR OTHER TORTIOUS ACTION, ARISING OUT OF OR IN
CONNECTION WITH THE USE OR PERFORMANCE OF THIS SOFTWARE.

Cookie management
==========
The Cookie module contains the following notice:

Copyright 2000 by Timothy O'Malley <timo@alum.mit.edu>

All Rights Reserved

Permission to use, copy, modify, and distribute this software
and its documentation for any purpose and without fee is hereby
granted, provided that the above copyright notice appear in all
copies and that both that copyright notice and this permission
notice appear in supporting documentation, and that the name of
Timothy O'Malley not be used in advertising or publicity
pertaining to distribution of the software without specific, written
prior permission.

Timothy O'Malley DISCLAIMS ALL WARRANTIES WITH REGARD TO THIS
SOFTWARE, INCLUDING ALL IMPLIED WARRANTIES OF MERCHANTABILITY
AND FITNESS, IN NO EVENT SHALL Timothy O'Malley BE LIABLE FOR
ANY SPECIAL, INDIRECT OR CONSEQUENTIAL DAMAGES OR ANY DAMAGES
WHATSOEVER RESULTING FROM LOSS OF USE, DATA OR PROFITS,
WHETHER IN AN ACTION OF CONTRACT, NEGLIGENCE OR OTHER TORTIOUS
ACTION, ARISING OUT OF OR IN CONNECTION WITH THE USE OR
PERFORMANCE OF THIS SOFTWARE.

Profiling
==========
The profile and pstats modules contain the following notice:

Copyright 1994, by InfoSeek Corporation, all rights reserved.
Written by James Roskind

Permission to use, copy, modify, and distribute this Python software
and its associated documentation for any purpose (subject to the
restriction in the following sentence) without fee is hereby granted,
provided that the above copyright notice appears in all copies, and
that both that copyright notice and this permission notice appear in
supporting documentation, and that the name of InfoSeek not be used in
advertising or publicity pertaining to distribution of the software
without specific, written prior permission. This permission is
explicitly restricted to the copying and modification of the software
to remain in Python, compiled Python, or other languages (such as C)
wherein the modified or derived code is exclusively imported into a
Python module.

INFOSEEK CORPORATION DISCLAIMS ALL WARRANTIES WITH REGARD TO THIS
SOFTWARE, INCLUDING ALL IMPLIED WARRANTIES OF MERCHANTABILITY AND
FITNESS. IN NO EVENT SHALL INFOSEEK CORPORATION BE LIABLE FOR ANY
SPECIAL, INDIRECT OR CONSEQUENTIAL DAMAGES OR ANY DAMAGES WHATSOEVER
RESULTING FROM LOSS OF USE, DATA OR PROFITS, WHETHER IN AN ACTION OF
CONTRACT, NEGLIGENCE OR OTHER TORTIOUS ACTION, ARISING OUT OF OR IN
CONNECTION WITH THE USE OR PERFORMANCE OF THIS SOFTWARE.

Execution tracing

```
=================
The trace module contains the following notice:

portions copyright 2001, Autonomous Zones Industries, Inc., all rights...
err...  reserved and offered to the public under the terms of the
Python 2.2 license.
Author: Zooko O'Whielacronx
http://zooko.com/
mailto:zooko@zooko.com

Copyright 2000, Mojam Media, Inc., all rights reserved.
Author: Skip Montanaro

Copyright 1999, Bioreason, Inc., all rights reserved.
Author: Andrew Dalke

Copyright 1995-1997, Automatrix, Inc., all rights reserved.
Author: Skip Montanaro

Copyright 1991-1995, Stichting Mathematisch Centrum, all rights reserved.

Permission to use, copy, modify, and distribute this Python software and
its associated documentation for any purpose without fee is hereby
granted, provided that the above copyright notice appears in all copies,
and that both that copyright notice and this permission notice appear in
supporting documentation, and that the name of neither Automatrix,
Bioreason or Mojam Media be used in advertising or publicity pertaining to
distribution of the software without specific, written prior permission.

UUencode and UUdecode functions
===============================

The uu module contains the following notice:

Copyright 1994 by Lance Ellinghouse
Cathedral City, California Republic, United States of America.
                    All Rights Reserved
Permission to use, copy, modify, and distribute this software and its
documentation for any purpose and without fee is hereby granted,
provided that the above copyright notice appear in all copies and that
both that copyright notice and this permission notice appear in
supporting documentation, and that the name of Lance Ellinghouse
not be used in advertising or publicity pertaining to distribution
of the software without specific, written prior permission.
LANCE ELLINGHOUSE DISCLAIMS ALL WARRANTIES WITH REGARD TO
THIS SOFTWARE, INCLUDING ALL IMPLIED WARRANTIES OF MERCHANTABILITY AND
FITNESS, IN NO EVENT SHALL LANCE ELLINGHOUSE CENTRUM BE LIABLE
FOR ANY SPECIAL, INDIRECT OR CONSEQUENTIAL DAMAGES OR ANY DAMAGES
WHATSOEVER RESULTING FROM LOSS OF USE, DATA OR PROFITS, WHETHER IN AN
ACTION OF CONTRACT, NEGLIGENCE OR OTHER TORTIOUS ACTION, ARISING OUT
OF OR IN CONNECTION WITH THE USE OR PERFORMANCE OF THIS SOFTWARE.

Modified by Jack Jansen, CWI, July 1995:
- Use binascii module to do the actual line-by-line conversion
  between ascii and binary. This results in a 1000-fold speedup. The C
  version is still 5 times faster, though.
- Arguments more compliant with Python standard

XML Remote Procedure Calls¶

The xmlrpclib module contains the following notice:

    The XML-RPC client interface is

Copyright (c) 1999-2002 by Secret Labs AB
Copyright (c) 1999-2002 by Fredrik Lundh

By obtaining, using, and/or copying this software and/or its
associated documentation, you agree that you have read, understood,
and will comply with the following terms and conditions:

Permission to use, copy, modify, and distribute this software and
its associated documentation for any purpose and without fee is
hereby granted, provided that the above copyright notice appears in
all copies, and that both that copyright notice and this permission
notice appear in supporting documentation, and that the name of
Secret Labs AB or the author not be used in advertising or publicity
pertaining to distribution of the software without specific, written
prior permission.
```

SECRET LABS AB AND THE AUTHOR DISCLAIMS ALL WARRANTIES WITH REGARD
TO THIS SOFTWARE, INCLUDING ALL IMPLIED WARRANTIES OF MERCHANT-
ABILITY AND FITNESS. IN NO EVENT SHALL SECRET LABS AB OR THE AUTHOR
BE LIABLE FOR ANY SPECIAL, INDIRECT OR CONSEQUENTIAL DAMAGES OR ANY
DAMAGES WHATSOEVER RESULTING FROM LOSS OF USE, DATA OR PROFITS,
WHETHER IN AN ACTION OF CONTRACT, NEGLIGENCE OR OTHER TORTIOUS
ACTION, ARISING OUT OF OR IN CONNECTION WITH THE USE OR PERFORMANCE
OF THIS SOFTWARE.

test_epoll
==========
The test_epoll contains the following notice:

Copyright (c) 2001-2006 Twisted Matrix Laboratories.

Permission is hereby granted, free of charge, to any person obtaining
a copy of this software and associated documentation files (the
"Software"), to deal in the Software without restriction, including
without limitation the rights to use, copy, modify, merge, publish,
distribute, sublicense, and/or sell copies of the Software, and to
permit persons to whom the Software is furnished to do so, subject to
the following conditions:

The above copyright notice and this permission notice shall be
included in all copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND,
EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF
MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND
NONINFRINGEMENT. IN NO EVENT SHALL THE AUTHORS OR COPYRIGHT HOLDERS BE
LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN ACTION
OF CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION
WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.

Select kqueue
=============
The select and contains the following notice for the kqueue interface:

Copyright (c) 2000 Doug White, 2006 James Knight, 2007 Christian Heimes
All rights reserved.

Redistribution and use in source and binary forms, with or without
modification, are permitted provided that the following conditions
are met:
1. Redistributions of source code must retain the above copyright
   notice, this list of conditions and the following disclaimer.
2. Redistributions in binary form must reproduce the above copyright
   notice, this list of conditions and the following disclaimer in the
   documentation and/or other materials provided with the distribution.

THIS SOFTWARE IS PROVIDED BY THE AUTHOR AND CONTRIBUTORS ``AS IS'' AND
ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE
IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE
ARE DISCLAIMED. IN NO EVENT SHALL THE AUTHOR OR CONTRIBUTORS BE LIABLE
FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL
DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS
OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT
LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY
OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF
SUCH DAMAGE.

strtod and dtoa
===============
The file Python/dtoa.c, which supplies C functions dtoa and strtod for conversion
of C doubles to and from strings, is derived from the file of the same name by
David M. Gay, currently available from http://www.netlib.org/fp/. The original
file, as retrieved on March 16, 2009, contains the following copyright and
licensing notice:

/**
 *
 * The author of this software is David M. Gay.
 *
 * Copyright (c) 1991, 2000, 2001 by Lucent Technologies.
 *
 * Permission to use, copy, modify, and distribute this software for
 * any purpose without fee is hereby granted, provided that this entire
 * notice is included in all copies of any software which is or
 * includes a copy or modification of this software and in all copies

```
 * of the supporting documentation for such software.
 *
 * THIS SOFTWARE IS BEING PROVIDED "AS IS", WITHOUT ANY EXPRESS OR
 * IMPLIED WARRANTY.  IN PARTICULAR, NEITHER THE AUTHOR NOR LUCENT
 * MAKES ANY REPRESENTATION OR WARRANTY OF ANY KIND CONCERNING THE
 * MERCHANTABILITY OF THIS SOFTWARE OR ITS FITNESS FOR ANY PARTICULAR
 * PURPOSE.
 *
 ************************************************************/
```

E.39 Red Hat RPM Spec ファイルライセンス

この製品には、次のソフトウェアが含まれている場合があります。

Red Hat RPM Spec ファイル

```
You are receiving a copy of the Red Hat spec file. The terms of the Oracle
license do NOT apply to the Red Hat spec file; it is licensed under the
following license, separately from the Oracle programs you receive.

                GNU GENERAL PUBLIC LICENSE
                  Version 2, June 1991

Copyright (C) 1989, 1991 Free Software Foundation, Inc.,
51 Franklin Street, Fifth Floor, Boston, MA 02110-1301 USA
Everyone is permitted to copy and distribute verbatim copies
of this license document, but changing it is not allowed.
[for rest of text, see following link]
```

このコンポーネントはセクションE.14「GNU General Public License バージョン 2.0、1991 年 6 月」に基づいてライセンスが付与されています

E.40 RegEX-Spencer ライブラリライセンス

この製品には次のソフトウェアが含まれている場合があります: Henry Spencer の正規表現ライブラリ (RegEX-Spencer)

```
Copyright 1992, 1993, 1994 Henry Spencer.  All rights reserved.
This software is not subject to any license of the American Telephone
and Telegraph Company or of the Regents of the University of California.

Permission is granted to anyone to use this software for any purpose on
any computer system, and to alter it and redistribute it, subject
to the following restrictions:

1. The author is not responsible for the consequences of use of this
   software, no matter how awful, even if they arise from flaws in it.

2. The origin of this software must not be misrepresented, either by
   explicit claim or by omission.  Since few users ever read sources,
   credits must appear in the documentation.

3. Altered versions must be plainly marked as such, and must not be
   misrepresented as being the original software.  Since few users
   ever read sources, credits must appear in the documentation.

4. This notice may not be removed or altered.
```

E.41 Richard A.O'Keefe 文字列ライブラリライセンス

この製品には、次のソフトウェアが含まれている場合があります。

Richard A.O'Keefe 文字列ライブラリ

```
The Richard O'Keefe String Library is subject to the following notice:

These files are in the public domain.  This includes getopt.c, which
is the work of Henry Spencer, University of Toronto Zoology, who
says of it "None of this software is derived from Bell software. I
had no access to the source for Bell's versions at the time I wrote
it.  This software is hereby explicitly placed in the public domain.
It may be used for any purpose on any machine by anyone." I would
greatly prefer it if *my* material received no military use.

The t_ctype.h file is subject to the following notice:
```

```
Copyright (C) 1998, 1999 by Pruet Boonma, all rights reserved.
Copyright (C) 1998 by Theppitak Karoonboonyanan, all rights reserved.

  Permission to use, copy, modify, distribute and sell this software and its
documentation for any purpose is hereby granted without fee, provided that the above
copyright notice appear in all copies.

  Smaphan Raruenrom and Pruet Boonma makes no representations about
the suitability of this software for any purpose. It is provided
"as is" without express or implied warranty.
```

E.42 sajson ライセンス

この製品には、次のソフトウェアが含まれている場合があります。

sajson

```
Copyright (c) 2012 Chad Austin

Permission is hereby granted, free of charge, to any person obtaining
a copy of this software and associated documentation files (the
"Software"), to deal in the Software without restriction, including
without limitation the rights to use, copy, modify, merge, publish,
distribute, sublicense, and/or sell copies of the Software, and to
permit persons to whom the Software is furnished to do so, subject
to the following conditions:

The above copyright notice and this permission notice shall be
included in all copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND,
EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF
MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT.
IN NO EVENT SHALL THE AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR
ANY CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN ACTION OF
CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION
WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.
```

E.43 SHA-1 in C ライセンス

この製品には、次のソフトウェアが含まれている場合があります。

SHA-1 in C

```
SHA-1 in C
By Steve Reid <steve@edmweb.com>
100% Public Domain
```

E.44 Unicode データファイル

この製品には、次のソフトウェアが含まれている場合があります。

Unicode データファイル

```
COPYRIGHT AND PERMISSION NOTICE

Copyright © 1991-2014 Unicode, Inc. All rights reserved. Distributed under
the Terms of Use in http://www.unicode.org/copyright.html.

Permission is hereby granted, free of charge, to any person obtaining a copy
of the Unicode data files and any associated documentation (the "Data Files")
or Unicode software and any associated documentation (the "Software") to deal
in the Data Files or Software without restriction, including without
limitation the rights to use, copy, modify, merge, publish, distribute,
and/or sell copies of the Data Files or Software, and to permit persons to
whom the Data Files or Software are furnished to do so, provided that (a) the
above copyright notice(s) and this permission notice appear with all copies
of the Data Files or Software, (b) both the above copyright notice(s) and
this permission notice appear in associated documentation, and (c) there is
clear notice in each modified Data File or in the Software as well as in the
documentation associated with the Data File(s) or Software that the data or
software has been modified.

THE DATA FILES AND SOFTWARE ARE PROVIDED "AS IS", WITHOUT WARRANTY OF ANY
KIND, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF
MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT OF
THIRD PARTY RIGHTS. IN NO EVENT SHALL THE COPYRIGHT HOLDER OR HOLDERS
```

```
INCLUDED IN THIS NOTICE BE LIABLE FOR ANY CLAIM, OR ANY SPECIAL INDIRECT OR
CONSEQUENTIAL DAMAGES, OR ANY DAMAGES WHATSOEVER RESULTING FROM LOSS OF USE,
DATA OR
PROFITS, WHETHER IN AN ACTION OF CONTRACT, NEGLIGENCE OR OTHER TORTIOUS
ACTION, ARISING OUT OF OR IN CONNECTION WITH THE USE OR PERFORMANCE OF THE
DATA FILES OR SOFTWARE.

Except as contained in this notice, the name of a copyright holder shall not
be used in advertising or otherwise to promote the sale, use or other
dealings in these Data Files or Software without prior written authorization
of the copyright holder.
```

E.45 V8 ライセンス

この製品には、次のソフトウェアが含まれている場合があります。

```
This license applies to all parts of V8 that are not externally
maintained libraries.  The externally maintained libraries used by V8
are:
.
 - PCRE test suite, located in
   test/mjsunit/third_party/regexp-pcre.js.  This is based on the
   test suite from PCRE-7.3, which is copyrighted by the University
   of Cambridge and Google, Inc.  The copyright notice and license
   are embedded in regexp-pcre.js.
.
 - Layout tests, located in test/mjsunit/third_party.  These are
   based on layout tests from webkit.org which are copyrighted by
   Apple Computer, Inc. and released under a 3-clause BSD license.
.
 - Strongtalk assembler, the basis of the files assembler-arm-inl.h,
   assembler-arm.cc, assembler-arm.h, assembler-ia32-inl.h,
   assembler-ia32.cc, assembler-ia32.h, assembler-x64-inl.h,
   assembler-x64.cc, assembler-x64.h, assembler-mips-inl.h,
   assembler-mips.cc, assembler-mips.h, assembler.cc and assembler.h.
   This code is copyrighted by Sun Microsystems Inc. and released
   under a 3-clause BSD license.
.
 - Valgrind client API header, located at third_party/valgrind/valgrind.h
   This is release under the BSD license.
.
These libraries have their own licenses; we recommend you read them,
as their terms may differ from the terms below.
.
Copyright 2006-2012, the V8 project authors. All rights reserved.
Redistribution and use in source and binary forms, with or without
modification, are permitted provided that the following conditions are
met:
.
    * Redistributions of source code must retain the above copyright
      notice, this list of conditions and the following disclaimer.
    * Redistributions in binary form must reproduce the above
      copyright notice, this list of conditions and the following
      disclaimer in the documentation and/or other materials provided
      with the distribution.
    * Neither the name of Google Inc. nor the names of its
      contributors may be used to endorse or promote products derived
      from this software without specific prior written permission.
.
THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS
"AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT
LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR
A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT
OWNER OR CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL,
SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT
LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE,
DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY
THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT
(INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE
OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.
.
========================================================================
.
License for  PCRE test suite (regexp-pcre.js):
// PCRE LICENCE
// ------------
//
// PCRE is a library of functions to support regular expressions whose syntax
// and semantics are as close as possible to those of the Perl 5 language.
```

```
//
// Release 7 of PCRE is distributed under the terms of the "BSD" licence, as
// specified below. The documentation for PCRE, supplied in the "doc"
// directory, is distributed under the same terms as the software itself.
//
// The basic library functions are written in C and are freestanding. Also
// included in the distribution is a set of C++ wrapper functions.
//
// THE BASIC LIBRARY FUNCTIONS
// ---------------------------
//
// Written by:       Philip Hazel
// Email local part: ph10
// Email domain:     cam.ac.uk
//
// University of Cambridge Computing Service,
// Cambridge, England.
//
// Copyright (c) 1997-2007 University of Cambridge
// All rights reserved.
//
//
// THE C++ WRAPPER FUNCTIONS
// -------------------------
//
// Contributed by:   Google Inc.
//
// Copyright (c) 2007, Google Inc.
// All rights reserved.
//
//
// THE "BSD" LICENCE
// -----------------
//
// Redistribution and use in source and binary forms, with or without
// modification, are permitted provided that the following conditions are met:
//
//     * Redistributions of source code must retain the above copyright notice,
//       this list of conditions and the following disclaimer.
//
//     * Redistributions in binary form must reproduce the above copyright
//       notice, this list of conditions and the following disclaimer in the
//       documentation and/or other materials provided with the distribution.
//
//     * Neither the name of the University of Cambridge nor the name of Google
//       Inc. nor the names of their contributors may be used to endorse or
//       promote products derived from this software without specific prior
//       written permission.
//
// THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS "AS IS"
// AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE
// IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE
// ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT OWNER OR CONTRIBUTORS BE
// LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR
// CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF
// SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS
// INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN
// CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE)
// ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE
// POSSIBILITY OF SUCH DAMAGE.
//
// End
.
========================================================================
 License for layout test files (array-isarray.js, array-splice-webkit.js,
object-keys.js and string-trim.js):
.
// Copyright (c) 2006-2009 Apple Computer, Inc. All rights reserved.
//
// Redistribution and use in source and binary forms, with or without
// modification, are permitted provided that the following conditions
// are met:
//
// 1. Redistributions of source code must retain the above copyright
// notice, this list of conditions and the following disclaimer.
//
// 2. Redistributions in binary form must reproduce the above
// copyright notice, this list of conditions and the following
// disclaimer in the documentation and/or other materials provided
// with the distribution.
```

```
//
// 3. Neither the name of the copyright holder(s) nor the names of any
// contributors may be used to endorse or promote products derived
// from this software without specific prior written permission.
//
// THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS
// "AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT
// LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS
// FOR A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE
// COPYRIGHT OWNER OR CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT,
// INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES
// (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR
// SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
// HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT,
// STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE)
// ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED
// OF THE POSSIBILITY OF SUCH DAMAGE.
.
========================================================================

License for Strongtalk assembler files (LICENSE.strongtalk):
Copyright (c) 1994-2006 Sun Microsystems Inc.
All Rights Reserved.
.
Redistribution and use in source and binary forms, with or without
modification, are permitted provided that the following conditions are
met:
.
- Redistributions of source code must retain the above copyright notice,
this list of conditions and the following disclaimer.
.
- Redistribution in binary form must reproduce the above copyright
notice, this list of conditions and the following disclaimer in the
documentation and/or other materials provided with the distribution.
.
- Neither the name of Sun Microsystems or the names of contributors may
be used to endorse or promote products derived from this software without
specific prior written permission.
.
THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS "AS
IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO,
THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR
PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT OWNER OR
CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL,
EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO,
PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR
PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF
LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING
NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS
SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.
.
========================================================================

License for the valgrind header file  (LICENSE.valgrind):
.
Notice that the following BSD-style license applies to this one
file (valgrind.h) only.  The rest of Valgrind is licensed under the
terms of the GNU General Public License, version 2, unless
otherwise indicated.  See the COPYING file in the source
distribution for details.
.
----------------------------------------------------------------
.
This file is part of Valgrind, a dynamic binary instrumentation
framework.
.
Copyright (C) 2000-2007 Julian Seward.  All rights reserved.
.
Redistribution and use in source and binary forms, with or without
modification, are permitted provided that the following conditions
are met:
.
1. Redistributions of source code must retain the above copyright
   notice, this list of conditions and the following disclaimer.
.
2. The origin of this software must not be misrepresented; you must
   not claim that you wrote the original software.  If you use this
   software in a product, an acknowledgment in the product
   documentation would be appreciated but is not required.
.
```

```
3. Altered source versions must be plainly marked as such, and must
   not be misrepresented as being the original software.
.
4. The name of the author may not be used to endorse or promote
   products derived from this software without specific prior written
   permission.
.
THIS SOFTWARE IS PROVIDED BY THE AUTHOR ``AS IS'' AND ANY EXPRESS
OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED
WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE
ARE DISCLAIMED.  IN NO EVENT SHALL THE AUTHOR BE LIABLE FOR ANY
DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL
DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE
GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS
INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY,
WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING
NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS
SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.
```

E.46 zlib ライセンス

この製品には、次のソフトウェアが含まれている場合があります。

zlib

Oracle は、この製品で使用されている zlib 汎用圧縮ライブラリの作成における Jean-loup Gailly と Mark Adler の貢献に大変感謝しています。

```
zlib.h -- interface of the 'zlib' general purpose compression library
Copyright (C) 1995-2004 Jean-loup Gailly and Mark Adler

zlib.h -- interface of the 'zlib' general purpose compression library
version 1.2.3, July 18th, 2005
Copyright (C) 1995-2005 Jean-loup Gailly and Mark Adler

zlib.h -- interface of the 'zlib' general purpose compression library
version 1.2.5, April 19th, 2010
Copyright (C) 1995-2010 Jean-loup Gailly and Mark Adler

This software is provided 'as-is', without any express or implied warranty.
In no event will the authors be held liable for any damages arising from the
use of this software. Permission is granted to anyone to use this software
for any purpose,including commercial applications, and to alter it and
redistribute it freely, subject to the following restrictions:

1. The origin of this software must not be misrepresented; you must not
   claim that you wrote the original software. If you use this software
   in a product, an acknowledgment in the product documentation would
   be appreciated but is not required.
2. Altered source versions must be plainly marked as such, and must not
   be misrepresented as being the original software.
3. This notice may not be removed or altered from any source distribution.

Jean-loup Gailly jloup@gzip.org
Mark Adler madler@alumni.caltech.edu
```

全般的な索引

シンボル

!= (等しくない), 1076
! (論理 NOT), 1079
", 923
#mysql50 識別子プリフィクス, 924, 928
%, 1106
% (モジュロ), 1110
% (ワイルドカード文字), 918
* (乗算), 1105
+ (加算), 1105
-? オプション (MySQL Cluster プログラム), 2285
-c オプション (MySQL Cluster プログラム), 2285
-c オプション (ndb_mgmd) (廃止), 2233
-d オプション (ndb_index_stat), 2256
-d オプション (ndb_mgmd), 2233
-e オプション (ndb_mgm), 2238
-f オプション (ndb_mgmd), 2233
-I オプション (ndbinfo_select_all), 2229
-n option (ndbmtd), 2226
-n オプション (ndbd), 2226
-p オプション, 645
-P オプション (ndb_mgmd), 2234
-V オプション (MySQL Cluster プログラム), 2287
--disable オプションプリフィクス, 235
--enable オプションプリフィクス, 235
--loose オプションプリフィクス, 235
--master-info-repository オプション, 1953
--maximum オプションプリフィクス, 235
--password オプション, 645
--relay-log-info-repository オプション, 1953
--skip オプションプリフィクス, 235
- (減算), 1105
- (単項マイナス), 1105
.ARM ファイル, 2989
.ARZ ファイル, 2989
.cfg ファイル, 2989
.frm ファイル, 2989
.ibd ファイル, 2989
.ibz ファイル, 2989
.isl ファイル, 2990
.MRG ファイル, 2990
.my.cnf file, 233
.my.cnf ファイル, 236, 237, 626, 646, 677
.MYD ファイル, 2990
.MYI ファイル, 2990
.mylogin.cnf ファイル, 236, 355
.mysql_history ファイル, 281, 646
.mysql_secret ファイル, 131, 257
.OPT ファイル, 2990
.PAR ファイル, 2990
.pid (プロセス ID) ファイル, 771
.TRG ファイル, 2990
.TRN ファイル, 2990
/etc/passwd, 657, 1334
/ (除算), 1105
10 進演算, 1214
16 進数リテラル, 921
2 フェーズコミット, 574, 574, 2990
3306 port, 444
3306 ポート, 158
:= (割り当て), 939
:= (割り当て演算子), 1080
= (等しい), 1075
= (割り当て), 939
= (割り当て演算子), 1081
\0 (ASCII NUL), 918, 1318
\b (バックスペース), 918, 1318
\N (NULL), 1318
\n (改行), 918, 1318
\n (ラインフィード), 918, 1318
\r (復帰改行), 918, 1318
\t (タブ), 918, 1318
\Z (Ctrl+Z) ASCII 26, 918, 1318
\. (mysql クライアントコマンド), 217, 283
\\ (エスケープ), 918
\' (単一引用符), 918
\" (二重引用符), 918
^ (ビット単位の XOR), 1162
_rowid, 1265
_ (ワイルドカード文字), 918
`, 923
|| (論理 OR), 1080
| (ビット単位の OR), 1162
~ (ビット反転), 1162
アーリーアダプタ, 3001
アカウント
  root, 173
  権限の追加, 682
  デフォルト, 173
  匿名ユーザー, 173
アカウント名, 670
悪意のある SQL ステートメント
  および MySQL Cluster, 2345
アクセス権限, 661
アクセス制御, 672
アクセスは拒否されましたエラー, 2942
新しい, 3003
圧縮, 1555, 3002
  BLOB、VARCHAR、および TEXT, 1562
  KEY_BLOCK_SIZE, 1559
  圧縮済みページサイズ, 1559
  アプリケーションとスキーマ設計, 1558
  アルゴリズム, 1560
  オーバーフローページ, 1562
  概要, 1555
  構成の特性, 1559
  実装, 1560
  情報スキーマ, 1685
  調整, 1556
  データとインデックス, 1561
  データの特性, 1557
  テーブルでの有効化, 1555
  バッファープールの考慮事項, 1562
  変更ログ, 1561
  モニタリング, 1560
  ログファイル形式, 1563
  ワークロードの特性, 1559
圧縮行フォーマット, 1572, 1572, 3002
圧縮失敗, 3002
圧縮テーブル, 350, 1769, 3002
圧縮バックアップ, 3002
アップグレード, 176, 177
  &current-series;への, 179
  MySQL APT リポジトリを使用して, 179
  MySQL Cluster, 2065, 2297

MySQL Yum リポジトリを使用して, 178
別のアーキテクチャー, 190
アップグレードとダウングレード (MySQL Cluster)
バージョン間の互換性, 2065
アトミック命令, 3001
アプリケーションプログラミングインタフェース (API), 3001
暗号化, 656, 715
暗号化関数, 1163
安全でないステートメント (レプリケーション), 1872
定義済み, 1871
安全なステートメント (レプリケーション)
defined, 1871
暗黙のカラム変更, 1285
暗黙の行ロック, 3003
暗黙のデフォルト値, 1054
アンロード
テーブル, 204
移植
ほかのシステムへの, 2822
移植性, 775
型, 1058
以前の起動後の CMake の実行, 146, 162
一時テーブル, 3004
およびレプリケーション, 1993
内部, 835
問題, 2967
一時テーブルスペース, 3004
一時ファイル, 2958
書き込みアクセス, 169
一括ロード
InnoDB テーブルの, 838
MyISAM テーブルの, 843
一貫性読み取り, 1504, 3005
一致する行がない, 2964
一般クエリーログ, 606, 3004
一般情報, 1
イベント, 2473, 2481
削除, 1293
作成, 1246
ステータス変数, 2487
制約, 2973
変更, 1223
メタデータ, 2484
イベントグループ, 1378
イベントスケジューラ, 2473, 2481
SQL ステートメント, 2484
イベントの削除, 1293
イベントの作成, 1246
イベントの変更, 1223
イベントメタデータ, 2484
および mysqladmin デバッグ, 2485
および MySQL 権限, 2485
および SHOW PROCESSLIST, 2482
およびレプリケーション, 1988, 1988
開始および停止, 2482
概念, 2481
時間表現, 2484
ステータス情報の取得, 2485
スレッドの状態, 915
有効および無効, 2482
イベントタイプ (MySQL Cluster), 2300, 2302
イベントの重大度レベル (MySQL Cluster), 2301
イベントログ (MySQL Cluster), 2299, 2300, 2301
イベントログの形式 (MySQL Cluster), 2302
インスタンス, 3003
インストゥルメンテーション, 3003
インストール, 86
Linux RPM パッケージ, 129
OS X DMG パッケージ, 110
Perl, 193
Perl を Windows に, 194
Solaris PKG パッケージ, 139
概要, 43
ソース配布, 143
バイナリ配布, 56
ユーザー定義関数, 2819
インストール後
セットアップとテスト, 163
複数サーバー, 620
インストールの概要, 143
インストールのレイアウト, 56, 56
インターネットリレーチャット, 20
インデックス, 1250, 3003
および BLOB カラム, 826, 1266
および IS NULL, 830
および LIKE, 829
および ndb_restore, 2271
および TEXT カラム, 826, 1266
カラム, 826
キーキャッシュへの割り当て, 1480
再作成, 189
削除, 1231, 1294
作成および削除, 1602
と NULL 値, 1265
名前, 923
の使用, 824
左端のプリフィクス, 825, 827
プライマリ (クラスタ化) とセカンダリ, 1602
ブロックサイズ, 490
マルチカラム, 827
マルチパート, 1250
インデックスキャッシュ, 3003
インデックスダイブ (統計の推定用), 1681
インデックス統計
NDB, 2126
インデックスの左端のプリフィクス, 825, 827
インデックスヒント, 1329, 1340, 3004
インデックスプリフィクス, 3004
インデックスレコードロック
InnoDB, 1502, 1507, 1509, 1640
インテンションロック, 3003
イントロデューサ
文字列リテラル, 917, 951
インポート
データ, 283, 316
インメモリーデータベース, 3004
引用
カラムエイリアス, 2963
引用符
文字列内, 918
引用符を使用, 919
カラムエイリアス, 923
ウォームアップ, 3005
ウォームバックアップ, 3005
永続的統計, 3006
エイリアス
GROUP BY 句, 1214
式, 1214

式に関する, 1329
テーブル, 1330
名前, 923
エイリアス名
大文字と小文字の区別, 925
エクステント, 3005
エスケープ (\\), 918
エスケープシーケンス
オプションファイル, 238
文字列, 917
エビクション, 3005
エラー
UDF の処理, 2818
アクセスは拒否されました, 2942
一般的, 2941
およびレプリケーション, 1995
既知, 2967
サブクエリー内, 1350
情報のソース, 2877
接続が失われました, 2945
ディレクトリチェックサム, 139
テーブルのチェック, 767
リスト, 2942
リンク, 2665
レポート, 21, 21
エラーコード, 390
エラー番号, 390
エラーメッセージ
言語, 988, 988
表示, 390
ファイルが見つかりません, 2953
エラーログ, 3005
エラーログ (MySQL Cluster), 2227
円記号 (日本語), 2861
演算
算術, 1105
演算子, 1061
キャスト, 1104, 1149
優先順位, 1074
論理, 1079
割り当て, 939, 1080
エンタープライズコンポーネント
MySQL Enterprise Audit, 726, 2835
MySQL Enterprise Backup, 2834
MySQL Enterprise Encryption, 2835
MySQL Enterprise Monitor, 2833
MySQL Enterprise Security, 700, 706, 2834
MySQL Thread Pool, 2835
MySQL スレッドプール, 896
大きなテーブル
NDB での作成, 1270
と MySQL Cluster, 1270
多くのサーバーの起動, 620
オーバーフローの処理, 1021
オーバーフローページ, 3007
オープンソース
定義, 4
オープンテーブル, 834
オープンの数, 288
遅いクエリー, 288
同じ値が優先 (競合解決), 2402
オプション, 3006
CMake, 150
libmysqld, 2658
myisamchk, 337
MySQL で提供されている, 197
組み込みサーバー, 2658
コマンド行
mysql, 267
mysqladmin, 289
ブール値, 235
レプリケーション, 1979
オプションファイル, 236, 677, 3006
エスケープシーケンス, 238
オプションプリフィクス
--disable, 235
--enable, 235
--loose, 235
--maximum, 235
--skip, 235
オプティマイザ, 3006
およびレプリケーション, 1994
切り替え可能な最適化, 859
クエリー計画評価, 859
制御, 859
オプティマイザ統計, 1681
InnoDB テーブル, 1676
オプティミスティック, 3006
オフページカラム, 3006
オペレーティングシステム
サポートされている, 45
ファイルサイズの制限, 2983
親テーブル, 3007
オンライン, 3006
オンライン DDL, 1575
クラッシュリカバリ, 1604
制限, 1605
並列性, 1581
例, 1584
オンライン DDL, 3006
オンラインアップグレードとダウングレード (MySQL Cluster), 2297
オンラインのアップグレードとダウングレード (MySQL Cluster)
ノード更新の順序, 2299
カーソル, 1393, 3007
カーディナリティー, 815, 3008
改行 (\n), 918, 1318
開始
コメント, 31
開発ソースツリー, 148
外部キー, 30, 219, 1232, 3008
削除, 1233, 1283
制約, 32, 32
外部キー制約, 1280
InnoDB, 1551
制約, 1551
とオンライン DDL, 1605
外部ロック, 432, 529, 767, 879, 909
概要, 1
カウンタ, 3007
カウント
テーブルの行, 212
書き込みアクセス
tmp, 169
書き込み結合, 3009
角括弧, 1011
拡張機能

標準 SQL への, 24
加算 (+), 1105
一意キー
およびパーティショニングキー, 2465
制約, 32
一意制約, 3008
一意の ID, 2755
一意のインデックス, 3008
一意のキー, 3008
型
移植性, 1058
カラム, 1011, 1058
時間, 1056
数値, 1056
データ, 1011
テーブルの, 1759
日付, 1056
日付と時間, 1022
文字列, 1033, 1057
型変換, 1071, 1075
() (括弧), 1074
括弧
角, 1011
括弧 (と), 1074
稼働時間, 288
カバリングインデックス, 3007
可用性, 3008
カラム, 3007
カラム
インデックス, 826
型, 1011
ストレージ要件, 1055
選択, 206
その他の型, 1058
名前, 923
表示, 321
変更, 1231
カラムインデックス, 3007
カラムエイリアス
引用, 2963
引用符を使用, 923
問題, 2963
カラムコメント, 1264
カラムストレージ, 1264
カラムフォーマット, 1264
カラムプリフィクス, 3007
カラム名
大文字と小文字の区別, 925
カレンダー, 1131
環境変数, 229, 244, 677
AUTHENTICATION_PAM_LOG, 705
CC, 162, 192
CXX, 162, 192
DBI_TRACE, 192, 2825
DBI_USER, 192
HOME, 192, 281
LD_LIBRARY_PATH, 194
LD_RUN_PATH, 192, 194
LIBMYSQL_ENABLE_CLEARTEXT_PLUGIN, 192
LIBMYSQL_PLUGIN_DIR, 192, 2753
LIBMYSQL_PLUGINS, 192, 2753
MYSQL_DEBUG, 192, 229, 2828
MYSQL_GROUP_SUFFIX, 192
MYSQL_HISTFILE, 192, 281
MYSQL_HISTIGNORE, 192, 281
MYSQL_HOME, 192
MYSQL_HOST, 192, 233
MYSQL_PS1, 192
MYSQL_PWD, 192, 229, 233
MYSQL_TCP_PORT, 192, 229, 625, 626
MYSQL_TEST_LOGIN_FILE, 192
MYSQL_UNIX_PORT, 169, 192, 229, 625, 626
PATH, 164, 192, 230
TMPDIR, 169, 192, 229, 2958
TZ, 192, 2960
UMASK, 192, 2953
UMASK_DIR, 192, 2954
USER, 192, 233
のリスト, 191
韓国語, 2861
監査プラグイン, 2773
監査ログプラグイン, 726
関数, 1061
C API, 2671
C プリペアドステートメント API, 2726, 2727
GROUP BY, 1207
GTID, 1193
SELECT および WHERE 句の, 1061
暗号化, 1163
およびレプリケーション, 1990
キャスト, 1149
グループ化, 1074
削除, 1438
作成, 1438
算術, 1161
情報, 1170
新規, 2812
数学, 1107
ストアド, 2475
制御フロー, 1081
その他, 1200
ネイティブ
追加, 2821
日付/時間, 1113
ビット, 1161
文字列, 1083
文字列比較, 1096
ユーザー定義, 1438, 1438, 2812
追加, 2813
関数名
あいまいさの解決, 928
構文解析, 928
完全バックアップ, 3009
カンマ区切り値データ、読み取り, 1316, 1334
管理
サーバー, 286
管理クライアント (MySQL Cluster), 2237
(参照 mgm)
管理ノード (MySQL Cluster), 2230
(参照 mgmd)
定義, 2011
管理プログラム, 228
関連性, 3009
キー, 826
2 つを使用した検索, 221
外部, 30, 219
マルチカラム, 827
キーキャッシュ

MyISAM, 864
インデックスの割り当て, 1480
キー値ストア, 830
キー容量
MyISAM, 1767
キーワード, 931
幾何図形, 1042
疑似レコード, 3009
規則
構文, 2
表記, 2
既知のエラー, 2967
起動, 3010
mysqld, 658
サーバーの, 164
サーバーを自動的に, 169
起動オプション
デフォルト, 236
起動パラメータ
mysql, 267
mysqladmin, 289
逆引用符, 3010
キャスト, 1071, 1075, 1149
キャスト演算子, 1149
キャスト関数, 1149
キャッシュ, 3009
クリア, 1481
ギャップ, 3009
ギャップイベント, 2377
ギャップロック, 3009
InnoDB, 1502, 1507, 1509, 1640
競合解決
MySQL Cluster レプリケーションで, 2399
mysqld 起動オプション, 2400
および ndb_replication システムテーブル, 2401
有効化, 2401
共有テーブルスペース, 3009
共有メモリートランスポータ (参照 MySQL Cluster)
共有ロック, 3009
切り捨て, 3009
近似値リテラル, 919, 1214
近接検索, 1135
空間関数, 1179
クエリー, 3010
速度, 775
入力, 198
パフォーマンスの推定, 858
例, 217
クエリー拡張, 1140
クエリーキャッシュ, 868
および ndbinfo データベーステーブル, 2321
およびパーティション化されたテーブル, 2463
スレッドの状態, 911
クエリー実行計画, 3010
区切り文字がない文字列, 921
行ベースレプリケーション, 3011
デメリット, 1869
メリット, 1869
行ロック, 3011
組み込み, 3011
組み込み MySQL サーバーライブラリ, 2657
クライアント, 3010
クライアント
スレッド, 2666
デバッグ, 2828
クライアント接続スレッド, 894
クライアントツール, 2653
クライアントのコンパイル
Unix 上, 2664
Windows 上, 2665
クライアントプログラム, 227
構築, 2664
クラスタ化されたインデックス, 3010
InnoDB, 1514
クラスタデータベース (OBSOLETE) (参照 MySQL Cluster レプリケーション)
クラスタリング (参照 MySQL Cluster)
クラスタログ, 2299, 2300
クラッシュ, 2823, 3010
繰り返される, 2956
リカバリ, 767
レプリケーション, 1992
クラッシュセーフバイナリログ, 13
クラッシュセーフレプリケーション, 13, 1903, 1954
クラッシュリカバリ, 3010
クリア
キャッシュ, 1481
クリーンシャットダウン, 601, 622, 1992, 3011
クリーンページ, 3011
グループ BY
標準 SQL への拡張, 1330
グループ化
式, 1074
グループコミット, 1672, 3011
グローバリゼーション, 945
グローバル権限, 1418, 1427
計画安定性, 3012
計算, 1214
カーディナリティー, 1458
日付, 207
行セットの集約値, 1207
桁, 1011
結合, 3012
Nested Loop アルゴリズム, 792
結合アルゴリズム
Block Nested Loop, 789
Nested Loop, 789
結合型
ALL, 851
const, 849
eq_ref, 850
fulltext, 850
index, 851
index_merge, 850
index_subquery, 851
range, 851
ref, 850
ref_or_null, 850
system, 849
unique_subquery, 850
権限
アクセス, 661
およびレプリケーション, 1994
削除, 685, 1418, 1418
追加, 682
デフォルト, 173
取り消し, 1427
破棄, 685

表示, 1457
付与, 1418
変更, 676
権限システム, 661
権限情報
ロケーション, 666
権限チェック
速度への影響, 817, 817
権限の変更, 676
言語サポート
エラーメッセージ, 988
検索
2 つのキー, 221
MySQL Web ページ, 21
全文, 1132
大文字/小文字の区別, 2960
検索インデックス, 3012
減算 (-), 1105
原子的, 3011
厳密値リテラル, 1214
厳密モード, 3011
高位境界値, 3013
攻撃者
セキュリティー, 656
貢献した会社
のリスト, 41
貢献者
のリスト, 35
降順インデックス, 3013
更新可能なビュー, 2489
構成
MySQL Cluster, 2141
合成キー, 3012
高精度計算, 1214
構成ファイル, 677, 3013
高速インデックス作成, 1575, 3013
高速シャットダウン, 3013
構築
クライアントプログラム, 2664
構文
正規表現, 1099
構文規則, 2
コールドバックアップ, 3012
互換性
mSQL との, 1099
MySQL バージョン間の, 179
ODBC との, 925, 1014, 1071, 1076, 1263, 1337
ODBC による, 532
Oracle との, 27, 1210, 1231, 1488
PostgreSQL との, 28
標準 SQL との, 24
国際化, 945
固定行フォーマット, 3012
固定小数点数演算, 1214
子テーブル, 3013
コマンド
バイナリ配布, 57
コマンド行ツール, 86, 266
コマンドオプション
mysql, 267
mysqladmin, 289
mysqld, 424
コマンドオプション (MySQL Cluster)
mysqld, 2178
ndb_mgm, 2237
ndb_mgmd, 2230
ndbd, 2222
ndbinfo_select_all, 2229
コマンド行オプション (MySQL Cluster), 2284
コマンド行履歴
mysql, 281
コマンド構文, 3
コマンドは同期されていません, 2951
コミット, 3012
コメント
開始, 31
追加, 943
コメントの構文, 943
混合ステートメント (レプリケーション), 1998
混在モード挿入, 3013
コンパイル
最適化, 880
ユーザー定義関数, 2819
コンパクト行フォーマット, 3012
サーバー, 3014
起動, 164
起動と停止, 169
起動の問題, 171
再起動, 166
シグナル処理, 600
シャットダウン, 166
接続, 197, 230
切断, 197
デバッグ, 2823
複数, 620
ログ, 602
サーバー側のカーソル
制約, 2976
サーバーの管理, 286
サーバープラグイン, 593
サーバー変数, 1477 (参照 システム変数)
サービス
プラグインのための, 2810
再起動
サーバー, 166
再構成, 162
最後の行
一意の ID, 2755
再作成
付与テーブル, 169
最小外接矩形, 1191, 1192
最小上限レコード, 3014
サイズ
表示, 1011
再接続
自動, 2634, 2756
最大下限レコード, 3014
最大数
結合あたりの最大テーブル数, 2982
データベースの最大数, 2982
テーブルあたりの最大カラム数, 2984
テーブルの最大数, 2982
最大値
最大行サイズ, 2984
テーブルサイズ, 2983
最適化, 774, 776, 781
Batched Key Access, 798, 799
BLOB 型, 833

Block Nested Loop, 798, 799
DELETE ステートメント, 817
DISTINCT, 806
DML ステートメント, 816
filesort, 802
GROUP BY, 804
InnoDB テーブル, 836
INSERT ステートメント, 816
LEFT JOIN, 788
LIMIT 句, 813
MEMORY テーブル, 845
Multi-Range Read, 797
MyISAM テーブル, 842
PERFORMANCE_SCHEMA, 903
SQL ステートメント, 775
UPDATE ステートメント, 817
インデックス, 824
外部キー, 825
サブクエリー, 806, 810
サブクエリー実体化, 808
主キー, 825
数値型, 832
多数のテーブル, 833
ディスク I/O, 887
データサイズ, 830
テーブル, 770
ネットワークの使用, 894
ヒント, 822
フルテーブルスキャン, 815
ベンチマーク, 901
メモリーの使用, 891
文字および文字列型, 832
先読み, 3014
線形, 1663
ランダム, 1663
削減
データサイズ, 830
削除, 3014
mysql.sock, 2959
インデックス, 1231, 1294
外部キー, 1233, 1283
関数, 1438
主キー, 1231
スキーマ, 1293
データベース, 1293
テーブル, 1295
ユーザー, 685, 685, 1418, 1418, 1418
行, 2964
削除バッファリング, 3014
作成
関数, 1438
スキーマ, 1245
データベース, 200, 1245
テーブル, 202
デフォルトの起動オプション, 236
バグレポート, 21
サブクエリー, 1344 (参照 サブクエリー)
ALL を使用した, 1346
ANY、IN、SOME を使用した, 1346
EXISTS を使用した, 1348
NOT EXISTS を使用した, 1348
ROW を使用した, 1347
エラー, 1350
結合としての書き換え, 1353
最適化, 810
制約, 2977
相関, 1348
サブクエリー実体化, 808, 808
サブクエリーの最適化, 806
サブパーティショニング, 2432, 2432
サブパーティション
既知の問題, 2464
サブリスト, 3014
サポート
オペレーティングシステム, 45
サロゲートキー, 3014
算術関数, 1161
算術式, 1105
参照, 1232
参照整合性, 1495, 1496, 3014
サンドボックスモード, 688
シーケンス, 221
シーケンスエミュレーション, 1177
シェル構文, 3
時間型, 1056
時間表現
イベントスケジューラ, 2484
時間リテラル, 920
式
拡張, 210
式のエイリアス, 1214, 1329
式の構文, 942
識別子, 923
引用符付与, 923
大文字と小文字の区別, 925
識別子の引用符付与, 923
シグナル
サーバー応答, 600
事後フィルタリング
パフォーマンススキーマ, 2577
システム
権限, 661
セキュリティー, 643
システムテーブルスペース, 3015
システムの最適化, 880
システム変数, 452, 548, 1477
audit_log_buffer_size, 741
audit_log_connection_policy, 742
audit_log_current_session, 742
audit_log_exclude_accounts, 742
audit_log_file, 743
audit_log_flush, 743
audit_log_format, 743
audit_log_include_accounts, 744
audit_log_policy, 744
audit_log_rotate_on_size, 745
audit_log_statement_policy, 745
audit_log_strategy, 746
authentication_windows_log_level, 467
authentication_windows_use_principal_name, 467
auto_increment_increment, 1896
auto_increment_offset, 1898
autocommit, 468
automatic_sp_privileges, 468
back_log, 468
basedir, 469
big_tables, 469
bind_address, 469

binlog_cache_size, 1933
binlog_checksum, 1934
binlog_direct_non_transactional_updates, 1934
binlog_error_action, 1935
binlog_format, 1935
binlog_gtid_recovery_simplified, 1936
binlog_max_flush_queue_time, 1937
binlog_order_commits, 1937
binlog_row_image, 1938
binlog_rows_query_log_events, 1939
binlog_stmt_cache_size, 1939
binlogging_impossible_mode, 1937
block_encryption_mode, 470
bulk_insert_buffer_size, 470
character_set_client, 470
character_set_connection, 471
character_set_database, 471
character_set_filesystem, 471
character_set_results, 472
character_set_server, 472
character_set_system, 472
character_sets_dir, 472
collation_connection, 472
collation_database, 472
collation_server, 473
completion_type, 473
concurrent_insert, 473
connect_timeout, 474
core_file, 474
datadir, 475
date_format, 475
datetime_format, 475
debug, 475
debug_sync, 476
default_storage_engine, 476
default_tmp_storage_engine, 476
default_week_format, 477
delay_key_write, 477
delayed_insert_limit, 478
delayed_insert_timeout, 478
delayed_queue_size, 478
disable_gtid_unsafe_statements, 1945
disconnect_on_expired_password, 479
div_precision_increment, 479
end_markers_in_json, 480
enforce_gtid_consistency, 1946
engine_condition_pushdown, 480
error_count, 481
event_scheduler, 481
expire_logs_days, 481
explicit_defaults_for_timestamp, 481
external_user, 482
flush, 482
flush_time, 483
foreign_key_checks, 483
ft_boolean_syntax, 483
ft_max_word_len, 484
ft_min_word_len, 484
ft_query_expansion_limit, 484
ft_stopword_file, 485
general_log, 485
general_log_file, 485
group_concat_max_len, 485
gtid_done, 1946
gtid_executed, 1946
gtid_lost, 1947
gtid_mode, 1947
gtid_next, 1947
gtid_owned, 1948
gtid_purged, 1948
have_compress, 486
have_crypt, 486
have_csv, 486
have_dynamic_loading, 486
have_geometry, 486
have_innodb, 486
have_openssl, 486
have_partitioning, 486
have_profiling, 486
have_query_cache, 486
have_rtree_keys, 486
have_ssl, 487
have_symlink, 487
hostname, 487
identity, 487
ignore_builtin_innodb, 1613
ignore_db_dirs, 487
init_connect, 488
init_file, 488
init_slave, 1916
innodb_adaptive_flushing, 1613
innodb_additional_mem_pool_size, 1614
innodb_autoextend_increment, 1616
innodb_autoinc_lock_mode, 1617
innodb_buffer_pool_instances, 1618
innodb_buffer_pool_size, 1620
innodb_change_buffering, 1621
innodb_checksums, 1623
innodb_commit_concurrency, 1623
innodb_concurrency_tickets, 1625
innodb_data_file_path, 1625
innodb_data_home_dir, 1626
innodb_disable_sort_file_cache, 1626
innodb_doublewrite, 1626
innodb_fast_shutdown, 1626
innodb_file_format, 1627
innodb_file_format_check, 1627
innodb_file_format_max, 1627
innodb_file_per_table, 1628
innodb_flush_log_at_timeout, 1628
innodb_flush_log_at_trx_commit, 1629
innodb_flush_method, 1630
innodb_force_recovery, 1632
innodb_io_capacity, 1637
innodb_lock_wait_timeout, 1639
innodb_locks_unsafe_for_binlog, 1639
innodb_log_buffer_size, 1641
innodb_log_file_size, 1642
innodb_log_files_in_group, 1643
innodb_log_group_home_dir, 1643
innodb_max_dirty_pages_pct, 1644
innodb_max_purge_lag, 1644
innodb_max_purge_lag_delay, 1645
innodb_mirrored_log_groups, 1645
innodb_old_blocks_pct, 1646
innodb_old_blocks_time, 1647
innodb_open_files, 1647
innodb_purge_batch_size, 1649

innodb_purge_threads, 1649
innodb_read_ahead_threshold, 1650
innodb_read_io_threads, 1651
innodb_replication_delay, 1652
innodb_rollback_on_timeout, 1652
innodb_spin_wait_delay, 1653
innodb_stats_method, 1654
innodb_stats_on_metadata, 1654
innodb_stats_sample_pages, 1656
innodb_status_output, 1656
innodb_status_output_locks, 1657
innodb_strict_mode, 1657
innodb_support_xa, 1657
innodb_sync_spin_loops, 1658
innodb_table_locks, 1658
innodb_thread_concurrency, 1659
innodb_thread_sleep_delay, 1660
innodb_use_native_aio, 1662
innodb_use_sys_malloc, 1662
innodb_version, 1662
innodb_write_io_threads, 1662
insert_id, 489
interactive_timeout, 489
join_buffer_size, 489
keep_files_on_create, 490
key_buffer_size, 490
key_cache_age_threshold, 491
key_cache_block_size, 491
key_cache_division_limit, 492
large_files_support, 492
large_page_size, 492
large_pages, 492
last_insert_id, 493
lc_messages, 493
lc_messages_dir, 493
lc_time_names, 493
license, 493
local_infile, 493
lock_wait_timeout, 494
locked_in_memory, 494
log, 494
log_bin, 1940
log_bin_basename, 1940
log_bin_index, 1940
log_bin_trust_function_creators, 494
log_bin_use_v1_row_events, 1940
log_error, 495
log_output, 495
log_queries_not_using_indexes, 495
log_slave_updates, 1941
log_slow_queries, 496
log_throttle_queries_not_using_indexes, 496
log_warnings, 496
long_query_time, 497
low_priority_updates, 497
lower_case_file_system, 497
lower_case_table_names, 498
master_info_repository, 1917
master_verify_checksum, 1941
max_allowed_packet, 498
max_binlog_cache_size, 1942
max_binlog_size, 1942
max_binlog_stmt_cache_size, 1942
max_connect_errors, 499
max_connections, 499
max_delayed_threads, 500
max_error_count, 500
max_heap_table_size, 500
max_insert_delayed_threads, 501
max_join_size, 501
max_length_for_sort_data, 501
max_prepared_stmt_count, 502
max_relay_log_size, 502
max_seeks_for_key, 502
max_sort_length, 503
max_sp_recursion_depth, 503
max_tmp_tables, 503
max_user_connections, 503
max_write_lock_count, 504
metadata_locks_cache_size, 504
metadata_locks_hash_instances, 505
min_examined_row_limit, 505
myisam_data_pointer_size, 506
myisam_max_sort_file_size, 506
myisam_mmap_size, 506
myisam_recover_options, 507
myisam_repair_threads, 507
myisam_sort_buffer_size, 507
myisam_stats_method, 507
myisam_use_mmap, 508
named_pipe, 508
ndb_log_empty_epochs, 2198
ndb_log_exclusive_reads, 2198
ndb_log_orig, 2198
ndb_log_transaction_id, 2199
net_buffer_length, 508
net_read_timeout, 509
net_retry_count, 509
net_write_timeout, 509
new, 510
old, 510
old_alter_table, 510
old_passwords, 510
open_files_limit, 511
optimizer_join_cache_level, 512
optimizer_prune_level, 512
optimizer_search_depth, 513
optimizer_switch, 513
optimizer_trace, 515
optimizer_trace_features, 515
optimizer_trace_limit, 516
optimizer_trace_max_mem_size, 516
optimizer_trace_offset, 516
performance_schema, 2640
performance_schema_accounts_size, 2640
performance_schema_digests_size, 2641
performance_schema_events_stages_history_long_size, 2641
performance_schema_events_stages_history_size, 2641
performance_schema_events_statements_history_long_size, 2642
performance_schema_events_statements_history_size, 2642
performance_schema_events_waits_history_long_size, 2642
performance_schema_events_waits_history_size, 2642
performance_schema_hosts_size, 2643
performance_schema_max_cond_classes, 2643

performance_schema_max_cond_instances, 2643
performance_schema_max_file_classes, 2643
performance_schema_max_file_handles, 2643
performance_schema_max_file_instances, 2644
performance_schema_max_mutex_classes, 2644
performance_schema_max_mutex_instances, 2644
performance_schema_max_rwlock_classes, 2644
performance_schema_max_rwlock_instances, 2645
performance_schema_max_socket_classes, 2645
performance_schema_max_socket_instances, 2645
performance_schema_max_stage_classes, 2645
performance_schema_max_statement_classes, 2646
performance_schema_max_table_handles, 2646
performance_schema_max_table_instances, 2646
performance_schema_max_thread_classes, 2646
performance_schema_max_thread_instances, 2647
performance_schema_session_connect_attrs_size, 2647
performance_schema_setup_actors_size, 2647
performance_schema_setup_objects_size, 2647
performance_schema_users_size, 2648
pid_file, 516
plugin_dir, 516
port, 517
preload_buffer_size, 517
profiling, 517
profiling_history_size, 517
protocol_version, 518
proxy_user, 518
pseudo_slave_mode, 518
pseudo_thread_id, 518
query_alloc_block_size, 518
query_cache_limit, 519
query_cache_min_res_unit, 519
query_cache_size, 519
query_cache_type, 520
query_cache_wlock_invalidate, 520
query_prealloc_size, 521
rand_seed1, 521
rand_seed2, 521
range_alloc_block_size, 521
read_buffer_size, 522
read_only, 522
read_rnd_buffer_size, 523
relay_log, 1917
relay_log_basename, 1917
relay_log_index, 1917
relay_log_info_file, 1917
relay_log_info_repository, 1918
relay_log_purge, 524
relay_log_recovery, 1918
relay_log_space_limit, 524
report_host, 524
report_password, 524
report_port, 525
report_user, 525
rpl_semi_sync_master_enabled, 525
rpl_semi_sync_master_timeout, 525
rpl_semi_sync_master_trace_level, 525
rpl_semi_sync_master_wait_no_slave, 526
rpl_semi_sync_slave_enabled, 526
rpl_semi_sync_slave_trace_level, 526
rpl_stop_slave_timeout, 1919
secure_auth, 527
secure_file_priv, 527
server_id, 527
server_id_bits, 2206
sha256_password_private_key_path, 528
sha256_password_public_key_path, 528
shared_memory, 528
shared_memory_base_name, 529
simplified_binlog_gtid_recovery, 1949
skip_external_locking, 529
skip_name_resolve, 529
skip_networking, 529
skip_show_database, 529
slave_checkpoint_group, 1919
slave_checkpoint_period, 1919
slave_compressed_protocol, 1920
slave_exec_mode, 1920
slave_load_tmpdir, 1920
slave_max_allowed_packet, 1921
slave_net_timeout, 1921
slave_parallel_workers, 1921
slave_pending_jobs_size_max, 1922
slave_rows_search_algorithms, 1923
slave_skip_errors, 1924
slave_sql_verify_checksum, 1924
slave_transaction_retries, 1924
slave_type_conversions, 1925
slow_launch_time, 530
slow_query_log, 530
slow_query_log_file, 530
socket, 530
sort_buffer_size, 531
sql_auto_is_null, 532
sql_big_selects, 532
sql_buffer_result, 532
sql_log_bin, 532
sql_log_off, 533
sql_mode, 533
sql_notes, 534
sql_quote_show_create, 534
sql_safe_updates, 534
sql_select_limit, 534
sql_slave_skip_counter, 1925
sql_warnings, 534
ssl_ca, 534
ssl_capath, 535
ssl_cert, 535
ssl_cipher, 535
ssl_crl, 535
ssl_crlpath, 535
ssl_key, 536
storage_engine, 536
sync_binlog, 1943
sync_frm, 537
sync_master_info, 1926
sync_relay_log, 1926
sync_relay_log_info, 1927
system_time_zone, 537
sysvar_stored_program_cache, 536
table_definition_cache, 537
table_open_cache, 538
table_open_cache_instances, 538
thread_cache_size, 538
thread_concurrency, 539
thread_handling, 539
thread_pool_algorithm, 540

thread_pool_high_priority_connection, 540
thread_pool_max_unused_threads, 540
thread_pool_prio_kickup_timer, 541
thread_pool_size, 541
thread_pool_stall_limit, 542
thread_stack, 542
time_format, 542
time_zone, 543
timed_mutexes, 543
timestamp, 543
tmp_table_size, 543
tmpdir, 544
transaction_alloc_block_size, 544
transaction_allow_batching, 2206
transaction_prealloc_size, 544
tx_isolation, 545
tx_read_only, 545
unique_checks, 546
updatable_views_with_limit, 546
validate_password_dictionary_file, 654
validate_password_length, 655
validate_password_mixed_case_count, 655
validate_password_number_count, 655
validate_password_policy, 656
validate_password_special_char_count, 656
validate_user_plugins, 546
version, 547
version_comment, 547
version_compile_machine, 547
version_compile_os, 547
wait_timeout, 547
warning_count, 548
およびレプリケーション, 1999
事前フィルタリング
パフォーマンススキーマ, 2577
実行
ANSI モード, 25
クエリー, 198
バッチモード, 216
複数サーバー, 620
実行スレッド (MySQL Cluster), 2115
質問
回答, 20
質問数, 288
質問への回答
エチケット, 20
自動インクリメント, 3016
自動インクリメントロック, 3016
自動コミット, 3016
シャープチェックポイント, 3015
シャットダウン, 601, 3015
サーバーの, 166
修復
テーブル, 293, 768
修復オプション
myisamchk, 340
集約関数のための呼び出しシーケンス
UDF, 2816
主キー, 3015
およびパーティショニングキー, 2465
削除, 1231
制約, 32
循環レプリケーション
MySQL Cluster で, 2378, 2395, 2399
準結合, 806
準同期レプリケーション, 1975
インストール, 1976
管理インタフェース, 1976
構成, 1977
モニタリング, 1978
準同期レプリケーションプラグイン, 2772
準備されたバックアップ, 3016
準備済みステートメント, 1382, 1386, 1386, 1386
条件, 1395, 1455, 1478
照合
文字列, 991
照合順序
INFORMATION_SCHEMA, 964
追加, 992
名前, 958
変更, 992
乗算 (*), 1105
小数点, 1011
小数秒
およびレプリケーション, 1989
小数秒の精度, 1011, 1014
冗長行フォーマット, 1572, 3015
情報関数, 1170
情報スキーマ
InnoDB テーブル, 1684
除算 (/), 1105
処理
エラー, 2818
引数, 2817
シングルユーザーモード (MySQL Cluster), 2291, 2317
および ndb_restore, 2261
信号
制約, 2976
シンボリックリンク, 888, 890
推定
クエリーパフォーマンス, 858
数学関数, 1107
数値, 919
数値型, 1056
数値スケール, 1011
数値精度, 1011
スーパーユーザー, 173
スキーマ, 3016
削除, 1293
作成, 1245
変更, 1222
スクリプト, 246, 250
mysql_install_db, 168
SQL, 266
スクリプトファイル, 216
スケーラビリティー, 3016
スケール
演算, 1214
数値, 1011
スケールアウト, 3017
スケールアップ, 3017
スタートアップパラメータ, 880
チューニング, 880
ステータス
テーブル, 1474
ステータス変数, 560, 1473
Audit_log_current_size, 746
Audit_log_event_max_drop_size, 747

Audit_log_events, 747
Audit_log_events_filtered, 747
Audit_log_events_lost, 747
Audit_log_events_written, 747
Audit_log_total_size, 747
Audit_log_write_waits, 747
MySQL Cluster, 2209
Rpl_semi_sync_master_clients, 580
Rpl_semi_sync_master_net_avg_wait_time, 580
Rpl_semi_sync_master_net_wait_time, 580
Rpl_semi_sync_master_net_waits, 580
Rpl_semi_sync_master_no_times, 580
Rpl_semi_sync_master_no_tx, 580
Rpl_semi_sync_master_status, 581
Rpl_semi_sync_master_timefunc_failures, 581
Rpl_semi_sync_master_tx_avg_wait_time, 581
Rpl_semi_sync_master_tx_wait_time, 581
Rpl_semi_sync_master_tx_waits, 581
Rpl_semi_sync_master_wait_pos_backtraverse, 581
Rpl_semi_sync_master_wait_sessions, 581
Rpl_semi_sync_master_yes_tx, 581
Rpl_semi_sync_slave_status, 581
ステータスログ (レプリケーション), 1953
ステートメント
GRANT, 682
INSERT, 683
複合, 1387
レプリケーションスレーブ, 1373
レプリケーションマスター, 1371
ステートメントベースレプリケーション, 3017
安全でないステートメント, 1868
デメリット, 1868
メリット, 1868
ステミング, 3017
ストアドファンクション, 2475
と INSERT DELAYED, 1308
ストアドプログラム, 1387, 2473
再解析, 873
ストアドプロシージャー, 2474
ストアドルーチン
LAST_INSERT_ID(), 2476
およびレプリケーション, 1988
制約, 2973
メタデータ, 2476
ストップワード, 1140, 3017
ストップワードリスト
ユーザー定義の, 1143
ストライピング
定義, 888
ストレージエンジン, 3017
ARCHIVE, 1776
InnoDB, 1495
NDB と InnoDB の違い, 2020
PERFORMANCE_SCHEMA, 2565
およびアプリケーション機能要件, 2022
可用性, 2020
サポートされるアプリケーション, 2022
使用シナリオ, 2022
選択, 1759
デフォルトとしての InnoDB, 1496
ストレージエンジンプラグイン, 2771
ストレージノード - データノード、ndbd、ndbmtd を参照 (参照 データノード、ndbd、ndbmtd)
ストレージノード - データノード、ndbd を参照 (参照 データノード、ndbd)
ストレージ要件
データ型, 1055
ストレージ領域
最適化, 830
スナップショット, 3017
スピン, 3017
スペース ID, 3017
スレーブサーバー, 3018
スレッド, 1461, 2767, 3018
表示, 1461
モニタリング, 903, 1461, 2513, 2634
スレッドキャッシュ, 894
スレッドクライアント, 2666
スレッド数, 288
スレッドテーブル
performance_schema, 2634
スレッドのコマンド, 904
Binlog Dump, 904
Change user, 904
Close stmt, 904
Connect, 904
Connect Out, 904
Create DB, 904
Daemon, 904
Debug, 904
Delayed insert, 904
Drop DB, 904
Error, 904
Execute, 904
Fetch, 904
Field List, 904
Init DB, 904
Kill, 904
Long Data, 904
Ping, 904
Prepare, 905
Processlist, 905
Query, 905
Quit, 905
Refresh, 905
Register Slave, 905
Reset stmt, 905
Set option, 905
Shutdown, 905
Sleep, 905
Statistics, 905
Table Dump, 905
Time, 905
スレッドのサポート, 45
スレッドの状態, 903
After create, 905
allocating local table, 911
altering table, 905
Analyzing, 906
Changing master, 914
Checking master version, 912
checking permissions, 906
checking privileges on cached query, 912
checking query cache for query, 912
Checking table, 906
cleaning up, 906
Clearing, 915

closing tables, 906
committing alter table to storage engine, 906
Committing events to binlog, 914
Connecting to master, 912
converting HEAP to MyISAM, 906
copy to tmp table, 906
Copying to group table, 906
Copying to tmp table, 906
Copying to tmp table on disk, 906
Creating delayed handler, 911
Creating index, 906
Creating sort index, 906
creating table, 906
Creating tmp table, 906
deleting from main table, 906
deleting from reference tables, 907
discard_or_import_tablespace, 907
end, 907
executing, 907
Execution of init_command, 907
Finished reading one binlog; switching to next binlog, 912
Flushing tables, 907
freeing items, 907
FULLTEXT initialization, 907
got handler lock, 911
got old table, 911
init, 907
Initialized, 915
insert, 911
invalidating query cache entries, 912
Killed, 907
Killing slave, 914
logging slow query, 907
login, 907
Making temporary file (append) before replaying LOAD DATA INFILE, 913
Making temporary file (create) before replaying LOAD DATA INFILE, 914
manage keys, 908
Master has sent all binlog to slave; waiting for binlog to be updated, 912
MySQL Cluster, 914
NULL, 907
Opening master dump table, 914
Opening mysql.ndb_apply_status, 914
Opening table, 908
Opening tables, 908
optimizing, 908
preparing for alter table, 908
preparing, 908
Processing events from schema table, 914
Processing events, 914
Purging old relay logs, 908
query end, 908
Queueing master event to the relay log, 913
Reading event from the relay log, 913
Reading from net, 908
Reading master dump table data, 914
Rebuilding the index on master dump table, 914
Reconnecting after a failed binlog dump request, 913
Reconnecting after a failed master event read, 913
Registering slave on master, 913
Removing duplicates, 908
removing tmp table, 908
rename, 908
rename result table, 908
Reopen tables, 908
Repair by sorting, 908
Repair done, 908
Repair with keycache, 909
Requesting binlog dump, 913
reschedule, 911
Rolling back, 909
Saving state, 909
Searching rows for update, 909
Sending binlog event to slave, 912
sending cached result to client, 912
setup, 909
Shutting down, 914
Slave has read all relay log; waiting for more updates, 914
Sorting for group, 909
Sorting for order, 909
Sorting index, 909
Sorting result, 909
statistics, 909
storing result in query cache, 912
storing row into queue, 911
Syncing ndb table schema operation and binlog, 914
System lock, 909
update, 909
updating main table, 909
updating reference tables, 910
Updating, 909
upgrading lock, 911
User lock, 910
User sleep, 910
Waiting for allowed to take ndbcluster global schema lock, 915
Waiting for commit lock, 910
waiting for delay_list, 911
Waiting for event from ndbcluster, 915
Waiting for first event from ndbcluster, 915
Waiting for global read lock, 910, 910
waiting for handler insert, 911
waiting for handler lock, 911
waiting for handler open, 911
Waiting for INSERT, 911
Waiting for master to send event, 913
Waiting for master update, 912
Waiting for ndbcluster binlog update to reach current position, 915
Waiting for ndbcluster global schema lock, 915
Waiting for ndbcluster to start, 915
Waiting for next activation, 915
Waiting for query cache lock, 912
Waiting for scheduler to stop, 915
Waiting for schema epoch, 915
Waiting for schema metadata lock, 910
Waiting for slave mutex on exit, 913, 914
Waiting for stored function metadata lock, 910
Waiting for stored procedure metadata lock, 910
Waiting for table level lock, 910
Waiting for table metadata lock, 910
Waiting for table, 910
Waiting for tables, 910
Waiting for the next event in relay log, 913
Waiting for the slave SQL thread to free enough relay log space, 913

Waiting for trigger metadata lock, 910
Waiting on cond, 910
Waiting on empty queue, 915
Waiting to finalize termination, 912
Waiting to reconnect after a failed binlog dump request, 913
Waiting to reconnect after a failed master event read, 913
Waiting until MASTER_DELAY seconds after master executed event, 914
Writing to net, 910
一般的な, 905
イベントスケジューラ, 915
クエリーキャッシュ, 911
遅延挿入, 910
レプリケーションスレーブ, 912, 913, 914
レプリケーションマスター, 912
スロークエリーログ, 617, 3018
正確値リテラル, 919
正規化, 3019
正規表現の構文, 1099
制御フロー関数, 1081
制限
InnoDB, 1552
MySQL の制限, 2982
MySQL の制限、MySQL での制限, 2982
ファイルサイズ, 2983
レプリケーション, 1979
静止, 3019
整数, 919
整数演算, 1214
精度
演算, 1214
小数秒, 1011, 1014
数値, 1011
正当な名, 923
制約, 32, 3019
InnoDB, 1552
performance_schema データベース, 2980
XA トランザクション, 2979
イベント, 2973
外部キー, 1280
サーバー側のカーソル, 2976
サブクエリー, 2977
信号, 2976
ストアドルーチン, 2973
トリガー, 2973
ビュー, 2978
プラガブルな認証, 2981
文字セット, 2980
セーブポイント, 3018
セカンダリインデックス, 3018
InnoDB, 1514
セキュリティー
InnoDB memcached インタフェース用, 1731
および NDB ユーティリティー, 2347
および悪意のある SQL ステートメント, 2345
攻撃者に対する, 656
セキュリティーシステム, 661
セグメント, 3018
設計
問題, 2967
セッション変数
およびレプリケーション, 1999
接続
SSH を使用したリモート, 725
検証, 672
サーバーに, 197, 230
中止, 2950
接続が失われましたエラー, 2945
接続文字列 (参照 MySQL Cluster)
切断
サーバーから, 197
設定
パスワード, 687
セットアップ
インストール後, 163
選択
MySQL のバージョン, 45
データ型, 1058
データベース, 201
選択性, 3019
全文
ストップワードリスト, 1143
全文検索, 1132, 3019
全文パーサープラグイン, 2771
相関サブクエリー, 1348
相互排他ロック, 3020
挿入, 3020
同時, 875, 877
の速度, 816
挿入可能なビュー
挿入可能, 2490
挿入バッファー, 1519, 3020
挿入バッファリング, 3020
無効化, 1670
増分バックアップ, 3019
増分リカバリ, 764
MySQL Cluster レプリケーションを使用, 2394
ソース配布
インストール, 143
ソート
データ, 206
テーブルの行, 206
付与テーブル, 674, 675
ソートバッファー, 3019
属性降格
レプリケーション, 1984
属性昇格
ndb_restore, 2264
レプリケーション, 1984
速度
クエリーの, 775, 775
挿入, 816
レプリケーションで増加, 1857
その他の関数, 1200
ダーティーページ, 1613, 1664, 3020
ダーティー読み取り, 3020
待機, 3020
タイプコード
C プリペアドステートメント API, 2725
タイムアウト, 1201, 1311
connect_timeout 変数, 293
shutdown_timeout 変数, 293
タイムアウト (レプリケーション), 1997
タイムゾーン
アップグレード, 1005
うるう秒, 1006
およびレプリケーション, 1997

サポート, 1003
タイムゾーンテーブル, 260
タイムゾーンの問題, 2960
大文字/小文字の区別
検索での, 2960
文字列比較, 1096
大文字小文字の区別
アクセスチェック, 670
データベース名の, 26
テーブル名の, 26
大文字と小文字の区別
識別子での, 925
名前での, 925
レプリケーションフィルタリングオプション, 1959
ダウングレード, 176, 186
MySQL Cluster, 2065, 2297
ダウンロード, 46
タブ (\t), 918, 1318
タプル, 3020
ダブル書き込みバッファー, 577
単一引用符 (\'), 918
単項マイナス (-), 1105
単純な関数のための呼び出しシーケンス
UDF, 2814
ダンプ
データベースとテーブル, 300, 381
端末モニター
定義, 197
チェック
テーブルのエラー, 767
チェックサム, 3021
チェックサムエラー, 139
チェックポイント, 3021
遅延挿入
スレッドの状態, 910
遅延レプリケーション, 1978
中国語、日本語、韓国語の文字セット
よくある質問, 2861
中止されたクライアント, 2950
中止された接続, 2950
抽出
日付, 207
チュートリアル, 197
チューニング, 774
調整
InnoDB 圧縮テーブル, 1557
地理空間特性, 1042
地理的特性, 1042
追加
新しいアカウント権限, 682
新しいユーザー権限, 682
新規関数, 2812
ネイティブ関数, 2821
文字セット, 989
ユーザー定義関数, 2813
ツール
mysqld_multi, 250
mysqld_safe, 246
コマンド行, 86, 266
のリスト, 40
低位境界値, 3024
停止
サーバー, 169
定数テーブル, 777
ディスク
データの分割, 890
ディスク上のデータ (MySQL Cluster)
および INFORMATION_SCHEMA.FILES テーブル, 2552
ディスクデータテーブル (MySQL Cluster) (参照 MySQL Cluster ディスクデータ)
ディスクバウンド, 3022
ディスクパフォーマンス, 887
ディスクベース, 3022
低速シャットダウン, 3024
ディレクトリ構造
デフォルト, 56
データ
インポート, 283, 316
サイズ, 830
テーブルへのロード, 203
取り出し, 204
データウェアハウス, 3023
データ型, 1011
BIGINT, 1013
BINARY, 1018, 1035
BIT, 1012
BLOB, 1018, 1036
BOOL, 1012, 1058
BOOLEAN, 1012, 1058
C API, 2661
CHAR, 1017, 1033
CHAR VARYING, 1017
CHARACTER, 1017
CHARACTER VARYING, 1017
DATE, 1015, 1023
DATETIME, 1015, 1023
DEC, 1014
DECIMAL, 1013, 1214
DOUBLE, 1014
DOUBLE PRECISION, 1014
ENUM, 1019, 1037
FIXED, 1014
FLOAT, 1014, 1014, 1014
GEOMETRY, 1043
GEOMETRYCOLLECTION, 1043
INT, 1013
INTEGER, 1013
LINESTRING, 1043
LONG, 1036
LONGBLOB, 1018
LONGTEXT, 1018
MEDIUMBLOB, 1018
MEDIUMINT, 1013
MEDIUMTEXT, 1018
MULTILINESTRING, 1043
MULTIPOINT, 1043
MULTIPOLYGON, 1043
NATIONAL CHAR, 1017
NATIONAL VARCHAR, 1017
NCHAR, 1017
NUMERIC, 1014
NVARCHAR, 1017
POINT, 1043
POLYGON, 1043
REAL, 1014
SET, 1019, 1040
SMALLINT, 1013
TEXT, 1018, 1036

TIME, 1016, 1025
TIMESTAMP, 1015, 1023
TINYBLOB, 1018
TINYINT, 1012
TINYTEXT, 1018
VARBINARY, 1018, 1035
VARCHAR, 1017, 1033
VARCHARACTER, 1017
YEAR, 1016, 1025
概要, 1011
データディクショナリ, 1528, 3023
データディレクトリ, 3023
mysql_upgrade_info ファイル, 262
データノード (MySQL Cluster), 2222, 2229
定義, 2011
データファイル, 3023
データベース, 3023
コピー, 190
削除, 1293
作成, 200, 1245, 1245
使用, 200
情報, 215
シンボリックリンク, 888
選択, 201, 1491
ダンプ, 300, 381
定義, 4
名前, 923
バックアップ, 751
表示, 321
複製, 1857
変更, 1222
データベース情報
取得, 1443
データベースのコピー, 190
データベース名
大文字小文字の区別, 26
大文字と小文字の区別, 925
データベースメタデータ, 2500
テーブル, 3021
BLACKHOLE, 1778
const, 849
CSV, 1774
EXAMPLE, 1789
FEDERATED, 1784
HEAP, 1771
InnoDB, 1495
MEMORY, 1771
MERGE, 1780
MyISAM, 1763
system, 849
圧縮フォーマット, 1769
エラーチェック, 767
多すぎる, 835
オープン, 834
カラムの選択, 206
行のカウント, 212
行のサイズ, 1055
行の削除, 2964
行の選択, 205
行のソート, 206
コピー, 1277, 1278
最後の行の一意の ID, 2755
再作成, 189
最大サイズ, 2983
最適化, 770
削除, 1295
作成, 202
修復, 189, 293, 768
情報, 215, 343
シンボリックリンク, 889
ステータスの表示, 1474
ダンプ, 300, 381
断片化, 1434
チェック, 340
定数, 777
データの取り出し, 204
データのロード, 203
デフラグ, 771, 1434, 1768
動的, 1768
閉じる, 834
名前, 923
パーティション化, 1780
パフォーマンスの向上, 830
表示, 321
開く, 834
複数, 213
変更, 1226, 1232, 2966
保守, 293
保守スケジュール, 771
マージ, 1780
テーブルあたりのカラム数
最大数, 2984
テーブルがいっぱい, 469
テーブル型
選択, 1759
テーブルが満杯です, 2951
テーブルキャッシュ, 834
テーブル情報
myisamchk, 343
テーブル数
フラッシュ, 288
テーブルスキャン, 1664
テーブルスペース, 3022
テーブルスペースディクショナリ, 3022
テーブルスペースモニター
InnoDB, 1574, 1709, 1722
テーブルタイプ, 3022
テーブルのエイリアス, 1330
テーブルの完全スキャン, 3021
テーブルのコピー, 1277, 1278
テーブルのサイズ, 2983
テーブル名
大文字小文字の区別, 26
大文字と小文字の区別, 925
テーブルモニター
InnoDB, 1709, 1754
テーブルレベルロック, 874
テーブルロック, 3022
デーモンプラグイン, 2772
適応型ハッシュインデックス, 1520, 3024
適応型フラッシュ, 3024
テキストコレクション, 3021
テキストファイル
インポート, 283, 316, 1313
テキストファイルから SQL ステートメントを実行する, 283
テキストファイルからの SQL ステートメントの実行, 216
適用, 3024
テスト

インストール, 164
インストール後, 163
サーバーへの接続, 672
デッドロック, 874, 1363, 1502, 1509, 1512, 1649, 1997, 2650, 3022
デッドロック検出, 3023
デッドロック対象, 3023
デバッグ
クライアント, 2828
サーバー, 2823
デバッグサポート, 150
デフォルト
組み込み, 2658
権限, 173
デフォルトアカウント, 173
デフォルトオプション, 236
デフォルト値, 1054, 1264, 1306
BLOB および TEXT カラム, 1036
暗黙の, 1054
明示的な, 1054
抑制, 33
デフォルトのインストール場所, 56
デフォルトのホスト名, 230
電子メールリスト, 18
転置インデックス, 3024
統計, 3025
同時挿入, 875, 877
動的行フォーマット, 1572, 3025
動的テーブルの特徴, 1768
ドキュメント
韓国語の, 2861
中国語の, 2861
日本語の, 2861
ドキュメント ID, 3025
ドキュメント作成者
のリスト, 39
匿名ユーザー, 173, 174, 672, 675
閉じる
テーブル, 834
トラブルシューティング, 2877, 3024
ALTER TABLE での問題, 2966
C API, 2754
InnoDB テーブルのデフラグ, 1575
InnoDB デッドロック, 1512
InnoDB のエラー, 1755
InnoDB のリカバリに関する問題, 1752
MySQL Enterprise Monitor を使用した, 2833
MySQL Server のコンパイル, 162
MySQL パフォーマンススキーマによる, 2650
OS のエラーコード, 1757
起動の問題, 171
接続の問題, 676
レプリケーション, 2003
トランザクション, 1502, 3025
およびレプリケーション, 1997, 1997
サポート, 29, 1495
分離レベル, 1502
メタデータのロック, 878
トランザクション ID, 3025
トランザクションアクセスモード, 1365
トランザクションセーフテーブル, 29, 1495
トランザクション分離レベル, 1365
READ COMMITTED, 1367
READ UNCOMMITTED, 1367
REPEATABLE READ, 1366
SERIALIZABLE, 1367
トランスポータブルテーブルスペース, 1532, 3025
トリガー, 1287, 1296, 1476, 2473, 2477
LAST_INSERT_ID(), 2476
およびレプリケーション, 1988, 1998
制約, 2973
と INSERT DELAYED, 1308
メタデータ, 2480
トリガーの表示, 1476
取り消し
権限, 1427
取り出し
テーブルのデータ, 204
トレースファイル
ndbmtd, 2230
トレースファイル (MySQL Cluster), 2227
ドロップ, 3025
内部仕様, 2767
内部のメモリーアロケータ
無効化, 1669
内部ロック, 874
ナチュラルキー, 3026
名前, 923
大文字と小文字の区別, 925
変数, 939
名前付きパイプ, 101, 105
名前のないビュー, 1349
二重引用符 (\"), 918
二重書き込みバッファー, 1573, 1626, 3026
日本語、韓国語、中国語の文字セット
よくある質問, 2861
日本語の文字セット
変換, 2861
入力
クエリー, 198
認証
InnoDB memcached インタフェース用, 1731
認証プラグイン, 2773
auth_socket, 711
auth_test_plugin, 711
authentication_pam, 700
authentication_windows, 706
authentication_windows_client, 706
mysql_clear_password, 710
mysql_native_password, 693
mysql_old_password, 694
sha256_password, 698
test_plugin_server, 711
ネイティブ関数
追加, 2821
ネイティブスレッドのサポート, 45
ネイティブのバックアップとリストア
バックアップ識別子, 2294
ネクストキーキーロック
InnoDB, 1640
ネクストキーロック, 3026
InnoDB, 1502, 1507, 1509
ネストされたクエリー, 1344
ネチケット, 20
ネットマスク表記
アカウント名における, 671
ネットワークポート
および MySQL Cluster, 2344

年齢
計算, 207
ノードグループ (MySQL Cluster), 2014
ノードログ (MySQL Cluster), 2299
パージ, 3028
パージスケジューリング, 1675
パージスレッド, 3028
パージ遅延, 3028
パージバッファリング, 3028
バージョン
最新, 46
選択, 45
パーティショニング, 2411
COLUMNS, 2422
および FULLTEXT インデックス, 2463
および SQL モード, 1997, 2461
および一時テーブル, 2463, 2465
および外部キー, 2463
およびキーキャッシュ, 2463
およびクエリーキャッシュ, 2463
およびサブクエリー, 2464
および日付, 2415
およびレプリケーション, 1997
概念, 2413
キーによる, 2431
最適化, 2451, 2453
サブパーティショニング, 2464
サポート, 2411
ストレージエンジン (制限), 2468
制限, 2460
線形ハッシュによる, 2430
タイプ, 2415
パーティショニングキーのデータ型, 2463
パーティショニング式で許可されない演算子, 2460
パーティショニング式で許可される関数, 2469
パーティショニング式でサポートされる演算子, 2460
パーティションの最大数, 2462
ハッシュによる, 2428
範囲による, 2416
有効化, 2411
リストによる, 2420
リソース, 2412
利点, 2414
リニアキーによる, 2432
パーティショニングキーおよび一意キー, 2465
パーティショニングキーおよび主キー, 2465
パーティショニング情報ステートメント, 2451
パーティション
管理, 2439
切り捨て, 2439
最適化, 2450
修復, 2450
チェック, 2450
追加および削除, 2439
分割およびマージ, 2439
分析, 2450
変更, 2439
パーティション (MySQL Cluster), 2014
パーティション化
MySQL Cluster でのサポート, 2024
パーティション化されたテーブルで使用される EXPLAIN, 2451
パーティション管理, 2439
パーティションに関する情報を取得する, 2451
パーティションプルーニング, 2453
ハートビート, 3026
排他ロック, 3028
バイナリデータで引用符を使用, 919
バイナリ配布
インストール, 56
バイナリロギング
および MySQL Cluster, 2029
バイナリログ, 607, 3027
イベントグループ, 1378
配布された権限 (MySQL Cluster), 2362
および NDB API アプリケーション, 2365
バウンス, 3027
破棄
ユーザー, 685
バグ
MySQL Cluster
レポート, 2253
既知, 2967
レポート, 2, 21
バグデータベース, 21
パス名区切り文字
Windows, 238
パスワード
InnoDB memcached インタフェース用, 1731
root ユーザー, 173
管理者ガイドライン, 646
失くした, 2954
セキュリティー, 645, 661
設定, 687, 1424, 1428
ハッシュ, 647
有効期限, 688
ユーザーガイドライン, 645
ユーザーの, 681
リセット, 2954
ロギング, 647
忘れた, 2954
「パスワードが現在のポリシー要件を満たしていません」
パスワードエラー, 652
パスワード検証, 652
パスワードの暗号化
可逆性, 1168
パスワードの期限切れ, 688
パスワードの設定, 1428
パスワードポリシー, 653
派生テーブル, 1349
最適化, 809
破損, 1752
破損ページ, 1573, 3028
パターンマッチング, 210, 1099
発音
MySQL, 5
バックアップ, 751, 2834, 3027
InnoDB, 1720
MySQL Cluster での, 2261, 2292, 2292, 2293, 2295
MySQL Cluster レプリケーションで, 2390
mysqldump による, 759
データベースとテーブル, 300, 381
バックアップからのリストア
MySQL Cluster レプリケーションで, 2390
バックアップ識別子
ネイティブのバックアップとリストア, 2294
バックアップ、トラブルシューティング
MySQL Cluster での, 2296

バックアップの構成
MySQL Cluster での, 2295
バックアップのリストア
MySQL Cluster での, 2261
バックグラウンドスレッド
書き込み, 1672
マスター, 1664, 1672
読み取り, 1672
バックスペース (\b), 918, 1318
バックスラッシュ
エスケープ文字, 917
パッケージ
のリスト, 40
ハッシュインデックス, 829, 3026
バッチ SQL ファイル, 266
バッチ更新 (MySQL Cluster レプリケーション), 2387
バッチモード, 216
バッファー, 3027
バッファーサイズ, 861
mysqld サーバー, 880
クライアント, 2653
バッファープール, 861, 1664, 3027
と圧縮テーブル, 1562
バッファープールインスタンス, 3027
バディーアロケータ, 1685, 3028
幅
表示, 1011
パフォーマンス, 774
推定, 858
ディスク I/O, 887
ベンチマーク, 902
パフォーマンススキーマ, 1674, 2565, 3028
イベントフィルタリング, 2577
メモリーの使用, 2574
パラメータ
サーバー, 880
範囲外の処理, 1021
半一貫性読み取り, 3028
InnoDB, 1640
ハンドラ, 1395
反復不可能読み取り, 3028
比較演算子, 1074
引数の処理, 2817
ビジネスルール, 3029
非正規化, 3029
日付
パーティショニングで使用される, 2415
パーティショニングで使用される (例), 2418, 2429, 2433, 2453
日付/時間関数, 1113
日付型, 1056
日付値
問題点, 1024
日付と時間型, 1022
日付の計算, 207
日付リテラル, 920
ビット関数, 1161
例, 221
非同期 I/O, 3029
非同期レプリケーション (参照 MySQL Cluster レプリケーション)
等しい (=), 1075
等しくない (!=), 1076
等しくない (<>), 1076
非トランザクションテーブル, 2964
非ブロック I/O, 3029
ビュー, 1289, 2473, 2488
アルゴリズム, 2488
およびレプリケーション, 2001
権限, 2979
更新可能, 1289, 2490
制限, 2979
制約, 2978
メタデータ, 2491
問題, 2979
表記規則, 2
表示
情報
SHOW, 1443, 1445, 1457, 1459, 1476
カーディナリティー, 1458
照合順序, 1458
データベース情報, 321, 321
テーブルステータス, 1474
表示サイズ, 1011
表示幅, 1011
標準 SQL
との違い, 28, 1427
に対する拡張機能, 26
への拡張, 24
標準互換性, 24
標準モニター
InnoDB, 1709
開いているテーブル数, 288
開く
テーブル, 834
非ロック読み取り, 3029
ヒント, 26
インデックス, 1330, 1340
最適化, 822
ファイアウォール (ソフトウェア)
および MySQL Cluster, 2342, 2344
ファイル
DDL ログ, 618
my.cnf, 1979
Not Found メッセージ, 2953
tmp, 169
一般クエリーログ, 606
エラーメッセージ, 988
権限, 2953
サイズの制限, 2983
修復, 340
スクリプト, 216
スロークエリーログ, 617
テキスト, 283, 316
バイナリログ, 607
メタデータログ, 618
ログ, 619
ファイル形式, 1566, 3029
Antelope, 1562
Barracuda, 1555
識別, 1570
ダウングレード, 1571
ファイルを作成/書き込みできない, 2951
ファジーチェックポイント, 3029
ファントム, 3029
ファントム行, 1508
フィールド
変更, 1231

フィルファクタ, 1519, 3030
ブール検索, 1135
ブール値オプション, 235
フェイルオーバー
  Java クライアント, 2013
  MySQL Cluster レプリケーションで, 2389
フォーラム, 20
負荷エミュレーション, 325
不完全な更新
  およびレプリケーション, 1995
複合インデックス, 3031
複合ステートメント, 1387
複合パーティショニング, 2432
複数サーバー, 620
複数の mysqld, 250
複数のディスクを使用したデータの起動, 890
複数のバッファープール, 1666
複数のロールバックセグメント, 1674
副選択, 1344
復帰 (\r), 918, 1318
復帰改行 (\r), 918, 1318
物理, 3031
物理バックアップ, 3031
浮動, 919
浮動小数点数, 1014
浮動小数点値
  およびレプリケーション, 1989
部分インデックス, 3031
部分バックアップ, 3031
付与
  権限, 1418
付与テーブル
  構造, 666
  再作成, 169
  ソート, 674, 675
付与テーブルの配布 (MySQL Cluster), 2362
ブラインドクエリー拡張, 1140, 3030
プラガブルな認証
  制約, 2981
プラガブル認証
  PAM, 700
  Windows, 706
プラグイン, 3030
  INFORMATION_SCHEMA, 2772
  アクティブ化, 593
  アンインストール, 593, 1440
  インストール, 593, 1439
  監査, 2773
  監査ログ, 726
  サーバー, 593
  準同期レプリケーション, 2772
  ストレージエンジン, 2771
  全文パーサー, 2771
  追加, 2768
  デーモン, 2772
  認証, 2773
プラグイン API, 593, 2768
プラグインサービス, 2810
プラグインのアクティブ化, 593
プラグインのアンインストール, 593, 1440
プラグインのインストール, 593, 1439
フラッシュ, 3030
フラッシュテーブル数, 288
フラッシュリスト, 3030
プリペアドステートメント, 2722
  再準備, 873
フルテーブルスキャン
  回避, 815
プログラム
  管理, 228
  クライアント, 227, 2664
  ストアド, 1387, 2473
  ユーティリティー, 228
プログラム開発ユーティリティー, 229
プログラムのオプション (MySQL Cluster), 2284
プログラム変数
  設定, 241
プログラム変数の設定, 241
プログラム例
  C API, 2664
プロシージャー
  ストアド, 2475
プロセス, 3030
  表示, 1461
プロセス管理 (MySQL Cluster), 2222
プロセスのサポート, 45
プロンプト
  意味, 199
分離レベル, 1502, 3030
並列性, 1495, 3032
  コミットの, 1623
  スレッドの, 1659
  チケット, 1625
ページ, 3031
ページクリーナー, 3032
ページサイズ
  InnoDB, 1519, 1553
ベータ, 3031
ペシミスティック, 3031
ベトナム語, 2861
変更
  カラム, 1231
  スキーマ, 1222
  ソケットの場所, 171, 2959
  データベース, 1222
  テーブル, 1226, 1232, 2966
  フィールド, 1231
  リリースノート, 2971
変更バッファー, 3032
変更バッファリング, 3032
  無効化, 1670
変数
  mysqld, 881
  およびレプリケーション, 1999
  環境, 229
  サーバー, 1477
  システム, 452, 548, 1477
  ステータス, 560, 1473
  ユーザー, 939
ベンチマーク, 901, 902
ポイントインタイムリカバリ, 764, 3033
  MySQL Cluster レプリケーションを使用, 2394
方法
  ロック, 874
ポート, 158, 172, 191, 230, 624, 644, 676, 725, 2604, 2942
保守
  テーブル, 293, 770
  ログファイル, 619

ホスト名, 230
アカウント名における, 670
デフォルト, 230
デフォルトアカウントの, 173
ホスト名キャッシュ, 895
ホスト名の解決, 895
ホット, 3032
ホットバックアップ, 3032
ボトルネック, 3033
翻訳者
のリスト, 39
マージ, 3033
マイナス
単項 (-), 1105
負の値, 919
マスターサーバー, 3033
マスタースレッド, 3033
マスターとスレーブの同期
MySQL Cluster レプリケーションで, 2392
マッチング
パターン, 210
マニュアル
オンラインでの場所, 2
構文規則, 2
入手可能な形式, 2
表記規則, 2
マニュアルのオンラインでの場所, 2
マルチカラムインデックス, 827
マルチコア, 3033
マルチパートインデックス, 1250
マルチバイト文字, 991
マルチバイト文字セット, 2952
マルチマスターレプリケーション
MySQL Cluster で, 2395, 2399
丸め, 1214
丸め誤差, 1013
満杯のディスク, 2958, 2958
ミッドポイント挿入, 1664
ミッドポイント挿入戦略, 3033
ミニトランザクション, 3033
ミラーサイト, 46
無効データ
制約, 33
明示的なデフォルト値, 1054
命名
MySQL のリリース, 45
メーリングリスト, 18
アーカイブの場所, 18
ガイドライン, 20
メタデータ
InnoDB, 2523, 2523
ストアドルーチン, 2476
データベース, 2500
トリガー, 2480
ビュー, 2491
メタデータのロック
トランザクション, 878
メタデータログ, 618
メタデータロック, 3034
メトリックカウンタ, 3034
メモリーアロケータ
innodb_use_sys_malloc, 1669
メモリー使用
MySQL Cluster, 2025
メモリー使用量
myisamchk, 348
メモリーの使用, 891
パフォーマンススキーマ, 2574
モード
バッチ, 216
文字
マルチバイト, 991
文字セット, 945
およびレプリケーション, 1981
制約, 2980
追加, 989
レパートリー, 965
文字セットレパートリー, 972
モジュロ (%), 1110
モジュロ (MOD), 1110
文字列
エスケープシーケンス, 917
区切り文字がない, 921
定義, 917
文字列型, 1033, 1057
文字列関数, 1083
文字列照合, 991
文字列の置換
replace ユーティリティー, 391
文字列の連結, 917, 1086
文字列比較
大文字/小文字の区別, 1096
文字列比較関数, 1096
文字列リテラルイントロデューサ, 917, 951
もっとも大きいタイムスタンプが優先 (競合解決), 2402
もっとも大きいタイムスタンプ、削除が優先 (競合解決), 2403
戻り値
UDF, 2818
モニター
InnoDB, 1574, 1709, 1722, 1751, 1754
端末, 197
モニタリング, 2833
スレッド, 903
問題
DATE カラム, 2961
MySQL Server のコンパイル, 162
Perl のインストール, 194
Solaris へのインストール, 139
アクセスは拒否されましたエラー, 2942
一般的なエラー, 2941
サーバーの起動, 171
接続が失われましたエラー, 2945
タイムゾーン, 2960
テーブルロック, 876
リンク, 2665
レポート, 2, 21
問題点
日付値, 1024
有効数値
例, 919
ユーザー
root, 173
削除, 685, 1418
ユーザーアカウント
作成, 1415
名前の変更, 1427
変更, 1414

リソース制限, 503, 685, 1425
ユーザーアカウントの作成, 1415
ユーザーアカウントの名前の変更, 1427
ユーザーアカウントの変更, 1414
ユーザー権限
削除, 685, 1418, 1418
追加, 682
破棄, 685
ユーザー定義関数, 1438, 1438
追加, 2812, 2813
ユーザー変数, 939
およびレプリケーション, 1999
ユーザー名
アカウント名における, 670
デフォルトアカウントの, 173
とパスワード, 681
優先順位
演算子, 1074
ユーティリティー
プログラム開発, 229
ユーティリティープログラム, 228
行, 3034
一致の問題, 2964
カウント, 212
削除, 2964
選択, 205
ソート, 206
ロック, 30
行サイズ
最大値, 2984
行サブクエリー, 1347
行フォーマット, 3034
行レベルロック, 874, 3034
抑制
デフォルト値, 33
読み取り競合の検出と解決
MySQL Cluster レプリケーションで, 2408
読み取り専用トランザクション, 3034
読み取りビュー, 3034
予約語, 931
およびレプリケーション, 1994
より多い (>), 1076
より多いか等しい (>=), 1076
より少ない (<), 1076
より少ないか等しい (<=), 1076
ラージページのサポート, 892
ライブラリ
libmysqlclient, 2653
libmysqld, 2653
ラインフィード (\n), 918, 1318
ラッチ, 3034
ラッパー
Eiffel, 2765
ラベル
ストアドプログラムブロック, 1387
ランダムダイブ, 3035
リカバリ
クラッシュから, 767
増分, 764
ポイントインタイム, 764
リスト, 3035
リストア, 3035
リソース制限
ユーザーアカウント, 503, 685, 1425
リソース不足エラー
およびパーティション化されたテーブル, 2462
リテラル, 917
リモート管理 (MySQL Cluster)
およびセキュリティー上の問題, 2344
リリース
命名スキーム, 45
リリースノート, 2971
リリース番号, 45
リレーショナル, 3035
リレーショナルデータベース
定義, 4
リレーログ (レプリケーション), 1953
履歴リスト, 3035
リンク, 2664
エラー, 2665
シンボリック, 888
問題, 2665
隣接ページ, 3035
例
myisamchk の出力, 343
圧縮テーブル, 351
クエリー, 217
レコードレベルロック
InnoDB, 1502, 1507, 1509, 1640
レコードロック, 3036
レパートリー
文字セット, 965, 972
レプリカ (MySQL Cluster), 2014
レプリケーション, 1857, 3036
BLACKHOLE, 1980
MySQL Cluster で, 2375
(参照 MySQL Cluster レプリケーション)
STATEMENT 形式と互換性のないステートメント, 1868
ZFS, 1804
安全および安全でないステートメント, 1871
および AUTO_INCREMENT, 1980
および CREATE ... IF NOT EXISTS, 1981
および CREATE TABLE ... SELECT, 1981
および DATA DIRECTORY, 1987
および DROP ... IF EXISTS, 1983
および FLUSH, 1989
および INDEX DIRECTORY, 1987
および LAST_INSERT_ID(), 1980
および LIMIT, 1991
および LOAD DATA, 1992
および max_allowed_packet, 1993
および MEMORY テーブル, 1993
および mysql (システム) データベース, 1994
および REPAIR TABLE ステートメント, 1437, 1992
および SQL モード, 1997
および TIMESTAMP, 1980
およびTRUNCATE TABLE, 1999
および一時テーブル, 1993
および関数, 1990
およびクエリーオプティマイザ, 1994
および権限, 1994
および小数秒, 1989
およびスケジュールされたイベント, 1988, 1988
およびストアドルーチン, 1988
およびスレーブ上エラー, 1995
およびタイムゾーン, 1997
およびトランザクション, 1997, 1997
およびトリガー, 1988, 1998

およびパーティショニング, 1997
およびビュー, 2000
および不完全な更新, 1995
および浮動小数点値, 1989
および変数, 1999
および文字セット, 1981
および呼び出される機能, 1988
および予約語, 1994
クラッシュ, 1992
シャットダウンおよび再起動, 1993
シャットダウンと再起動, 1992
循環, 2378
準同期, 1975
ステータスログ, 1953
属性降格, 1984
属性昇格, 1984
タイムアウト, 1997
遅延, 1978
テーブルが異なるマスターとスレーブ, 1983
行ベース対ステートメントベース, 1867
リレーログ, 1953
レプリケーションオプション, 1979
レプリケーション形式
比較, 1867
レプリケーションスレーブ
ステートメント, 1373
スレッドの状態, 912, 913, 914
レプリケーション制限, 1979
レプリケーションで増加
速度, 1857
レプリケーションの実装, 1951
レプリケーション、非同期 (参照 MySQL Cluster レプリケーション)
レプリケーションフィルタリングオプション
および大文字と小文字の区別, 1959
レプリケーションマスター
ステートメント, 1371
スレッドの状態, 912
レプリケーションログテーブル, 1953
レポート
エラー, 21
バグ, 2, 21
問題, 2
連結
文字列, 917, 1086
ローカライズ, 945
ロード
テーブル, 203
ローリング再起動 (MySQL Cluster), 2297
ロールバック, 3037
ロールバックセグメント, 1535, 3037
ロギング
パスワード, 647
ロギングコマンド (MySQL Cluster), 2300
ログ, 3036
サーバー, 602
フラッシュ, 602
ロググループ, 3036
ログバッファー, 3036
ログファイル, 3036
保守, 619
ログファイル (MySQL Cluster), 2227
ndbmtd, 2230
ロック, 880, 3036, 3036
外部, 432, 529, 767, 879, 909
情報スキーマ, 1686
テーブルレベル, 874
内部, 874
行レベル, 30, 874
ロックエスカレーション, 3036
ロック方法, 874
ロックモニター
InnoDB, 1709
ロック読み取り, 3037
論理, 3037
論理演算子, 1079
論理バックアップ, 3037
ワークロード, 3037
ワイルドカード
mysql.columns_priv テーブル内, 675
mysql.db テーブル内, 675
mysql.procs_priv テーブル内, 675
mysql.tables_priv テーブル内, 675
アカウント名における, 671
および LIKE, 829
ワイルドカード文字 (%), 918
ワイルドカード文字 (_), 918
割り当て演算子, 1080
:=, 1080
=, 1081

A

abort-slave-event-count オプション
mysqld, 1899
ABS(), 1107
accounts テーブル
performance_schema, 2618
ACID, 2990
ACID, 29, 1495, 1500
ACL, 661
ACOS(), 1107
ActiveState Perl, 194
add-drop-database オプション
mysqldump, 306
add-drop-table オプション
mysqldump, 306
add-drop-trigger オプション
mysqldump, 306
add-locks オプション
mysqldump, 313
ADDDATE(), 1116
ADDTIME(), 1116
addtodest オプション
mysqlhotcopy, 382
AES_DECRYPT(), 1164
AES_ENCRYPT(), 1164
After create
スレッドの状態, 905
AHI, 2991
AIO, 2991
all-databases オプション
mysqlcheck, 296
mysqldump, 311
all-in-1 オプション
mysqlcheck, 296
all-tablespaces オプション
mysqldump, 306
ALL, 1332, 1346

allocating local table
スレッドの状態, 911
allow-keywords オプション
mysqldump, 307
allow-suspicious-udfs オプション
mysqld, 425
ALLOW_INVALID_DATES SQL モード, 587
allowold オプション
mysqlhotcopy, 382
all オプション
mysql_config_editor, 359
ALL 結合型
オプティマイザ, 851
ALTER COLUMN, 1231
ALTER DATABASE, 1222
ALTER EVENT, 1223
およびレプリケーション, 1988
ALTER FUNCTION, 1225
ALTER LOGFILE GROUP, 1224
(参照 MySQL Cluster ディスクデータ)
ALTER PROCEDURE, 1226
ALTER SCHEMA, 1222
ALTER SERVER, 1226
ALTER TABLE, 1226, 1232, 2966
ROW_FORMAT, 1571
およびレプリケーションログテーブル, 1953
ALTER TABLESPACE, 1244
(参照 MySQL Cluster ディスクデータ)
ALTER USER, 1414
ALTER VIEW, 1245
altering table
スレッドの状態, 905
&& (論理 AND), 1080
& (ビット単位の AND), 1162
ANALYSE()
PROCEDURE, 833
ANALYZE TABLE, 1430
およびパーティショニング, 2450
analyze オプション
myisamchk, 342
mysqlcheck, 296
Analyzing
スレッドの状態, 906
AND
ビット単位, 1162
論理, 1080
ANSI SQL モード, 586, 592
ANSI_QUOTES SQL モード, 587
ansi オプション
mysqld, 425
ANSI モード
実行, 25
Antelope, 2991
Antelope ファイル形式, 1566, 1572, 1627
ANY, 1346
Apache, 223
API, 2653
Perl, 2763
のリスト, 40
[api] (MySQL Cluster), 2155
API ノード (参照 SQL ノード)
API ノード (MySQL Cluster)
定義, 2011
append オプション (ndb_restore), 2267
apply-slave-statements オプション
mysqldump, 308
apply_status テーブル (OBSOLETE), 2383
(参照 MySQL Cluster レプリケーション)
Arbitration, 2107
ArbitrationDelay, 2079, 2129
ArbitrationRank, 2079, 2129
ArbitrationTimeout, 2107
arbitrator_validity_detail
ndbinfo テーブル, 2323
arbitrator_validity_summary
ndbinfo テーブル, 2323
ARCHIVE ストレージエンジン, 1759, 1776
Area(), 1187, 1188
AS, 1329, 1335
AsBinary(), 1183
ASCII(), 1085
ASIN(), 1107
AsText(), 1183
ASYMMETRIC_DECRYPT(), 1198
ASYMMETRIC_DERIVE(), 1198
ASYMMETRIC_ENCRYPT(), 1198
ASYMMETRIC_SIGN(), 1199
ASYMMETRIC_VERIFY(), 1199
ATAN2(), 1107
ATAN(), 1107
audit-log オプション
mysqld, 740
audit_log_buffer_size システム変数, 741
audit_log_connection_policy システム変数, 742
audit_log_current_session システム変数, 742
Audit_log_current_size ステータス変数, 746
Audit_log_event_max_drop_size ステータス変数, 747
Audit_log_events_filtered ステータス変数, 747
Audit_log_events_lost ステータス変数, 747
Audit_log_events_written ステータス変数, 747
Audit_log_events ステータス変数, 747
audit_log_exclude_accounts システム変数, 742
audit_log_file システム変数, 743
audit_log_flush システム変数, 743
audit_log_format システム変数, 743
audit_log_include_accounts システム変数, 744
audit_log_policy システム変数, 744
audit_log_rotate_on_size システム変数, 745
audit_log_statement_policy システム変数, 745
audit_log_strategy システム変数, 746
Audit_log_total_size ステータス変数, 747
Audit_log_write_waits ステータス変数, 747
audit_log プラグイン
起動の失敗, 736
auth_socket 認証プラグイン, 711
auth_test_plugin 認証プラグイン, 711
AUTHENTICATION_PAM_LOG 環境変数, 705
authentication_pam 認証プラグイン, 700
authentication_windows_client 認証プラグイン, 706
authentication_windows_log_level システム変数, 467
authentication_windows_use_principal_name システム変数, 467
authentication_windows 認証プラグイン, 706
auto-generate-sql-add-autoincrement オプション
mysqlslap, 328
auto-generate-sql-execute-number オプション
mysqlslap, 328
auto-generate-sql-guid-primary オプション

mysqlslap, 328
auto-generate-sql-load-type オプション
mysqlslap, 328
auto-generate-sql-secondary-indexes オプション
mysqlslap, 328
auto-generate-sql-unique-query-number オプション
mysqlslap, 328
auto-generate-sql-unique-write-number オプション
mysqlslap, 329
auto-generate-sql-write-number オプション
mysqlslap, 329
auto-generate-sql オプション
mysqlslap, 328
auto-rehash オプション
mysql, 269
auto-repair オプション
mysqlcheck, 296
auto-vertical-output オプション
mysql, 269
auto.cnf ファイル, 1880
と SHOW SLAVE HOSTS, 1467
AUTO_INCREMENT, 221, 1019
および NULL 値, 2963
およびレプリケーション, 1980
auto_increment_increment システム変数, 1896
auto_increment_offset システム変数, 1898
autocommit システム変数, 468
automatic_sp_privileges システム変数, 468
AutoReconnect
API および SQL ノード, 2130
AVG(), 1208
AVG(DISTINCT), 1208

B

back_log システム変数, 468
backup_path オプション (ndb_restore), 2266
BackupDataBufferSize, 2111, 2295
BackupDataDir, 2083
backupid オプション (ndb_restore), 2265
BackupLogBufferSize, 2112, 2295
BackupMaxWriteSize, 2113, 2295
BackupMemory, 2112, 2295
BackupReportFrequency, 2112
BackupWriteSize, 2113, 2295
BACKUP イベント (MySQL Cluster), 2306
backup オプション
myisamchk, 340
myisampack, 350
Barracuda, 2991
Barracuda ファイル形式, 1555, 1566, 1572, 1627
base64-output オプション
mysqlbinlog, 364
basedir オプション
mysql.server, 250
mysql_install_db, 256
mysql_plugin, 259
mysql_upgrade, 263
mysqld, 425
mysqld_safe, 247
basedir システム変数, 469
BatchByteSize, 2129
Batched Key Access
最適化, 798, 799
BatchSize, 2129
BatchSizePerLocalScan, 2090
batch オプション
mysql, 269
Bazaar ツリー, 148
BEGIN, 1355, 1387
XA トランザクション, 1369
ラベル, 1387
BENCHMARK(), 1171
BETWEEN ... AND, 1077
big-tables オプション
mysqld, 425
big5, 2861
big_tables システム変数, 469
BIGINT データ型, 1013
BIN(), 1085
binary-mode オプション
mysql, 269
BINARY, 1149
BINARY データ型, 1018, 1035
bind-address オプション
mysql, 269
mysqladmin, 290
mysqlbinlog, 365
mysqlcheck, 296
mysqld, 426
mysqldump, 304
mysqlimport, 318
mysqlshow, 323
bind-address オプション (ndb_mgmd), 2231
bind_address システム変数, 469
binlog, 2991
binlog-checksum オプション
mysqld, 1932
binlog-do-db オプション
mysqld, 1930
binlog-format オプション
mysqld, 426
binlog-ignore-db オプション
mysqld, 1931
binlog-row-event-max-size オプション
mysqlbinlog, 365
mysqld, 1928
binlog-rows-query-log-events オプション
mysqld, 1933
BINLOG, 1480
Binlog Dump
スレッドのコマンド, 904
binlog_cache_size システム変数, 1933
binlog_checksum システム変数, 1934
binlog_direct_non_transactional_updates システム変数, 1934
binlog_error_action システム変数, 1935
binlog_format
BLACKHOLE, 1980
binlog_format システム変数, 1935
binlog_gtid_recovery_simplified, 1936
binlog_index テーブル (OBSOLETE) (参照 MySQL Cluster レプリケーション)
binlog_max_flush_queue_time システム変数, 1937
binlog_order_commits システム変数, 1937
binlog_row_image システム変数, 1938
binlog_rows_query_log_events システム変数, 1939
binlog_stmt_cache_size システム変数, 1939
binlogging_impossible_mode システム変数, 1937
BINLOG ステートメント

mysqlbinlog 出力, 374
BIT_AND(), 1208
BIT_COUNT, 221
BIT_COUNT(), 1162
BIT_LENGTH(), 1085
BIT_OR, 221
BIT_OR(), 1208
BIT_XOR(), 1208
BIT データ型, 1012
BLACKHOLE
binlog_format, 1980
レプリケーション, 1980
BLACKHOLE ストレージエンジン, 1759, 1778
blob-info オプション
ndb_desc, 2251
BLOB カラム
インデックス設定, 826, 1266
サイズ, 1057
デフォルト値, 1036
バイナリデータの挿入, 919
BLOB データ型, 1018, 1036
block-search オプション
myisamchk, 342
Block Nested Loop
最適化, 798, 799
Block Nested Loop 結合アルゴリズム, 789
block_encryption_mode システム変数, 470
blocks
ndbinfo テーブル, 2323
BOOLEAN データ型, 1012
BOOL データ型, 1012
bootstrap オプション
mysqld, 427
brief オプション
mysqlaccess, 360
browser-start-page オプション
ndb_setup.py, 2277
Buffer(), 1189
bugs.mysql.com, 21
BUILD_CONFIG オプション
CMake, 154
builddir オプション
mysql_install_db, 256
BuildIndexThreads, 2114
bulk_insert_buffer_size システム変数, 470
B ツリー, 2991
B ツリーインデックス, 829, 1519

C

C++, 2657
C:\my.cnf ファイル, 626
C API, 2653
関数, 2671
データ型, 2661
プログラム例, 2664
リンクの問題, 2665
ca-certs-file オプション
ndb_setup.py, 2277
CACHE INDEX, 1480
およびパーティショニング, 2463
cache_policies テーブル, 1745
CALL, 1298
CASE, 1082, 1390
CAST, 1150
CC 環境変数, 162, 192
CEIL(), 1108
CEILING(), 1108
Centroid(), 1188
cert-file オプション
ndb_setup.py, 2277
cflags オプション
mysql_config, 388
CHANGE MASTER TO, 1373
MySQL Cluster で, 2385
Change user
スレッドのコマンド, 904
ChangeLog, 2971
Changing master
スレッドの状態, 914
CHAR(), 1085
CHAR VARYING データ型, 1017
CHAR_LENGTH(), 1085
character-set-client-handshake オプション
mysqld, 427
character-set-filesystem オプション
mysqld, 427
character-set-server オプション
mysqld, 428
character-sets-dir オプション
myisamchk, 340
myisampack, 350
mysql, 270
mysql_upgrade, 263
mysqladmin, 290
mysqlbinlog, 365
mysqlcheck, 296
mysqld, 427
mysqldump, 308
mysqlimport, 318
mysqlshow, 323
character-sets-dir オプション (MySQL Cluster プログラム), 2285, 2285
CHARACTER VARYING データ型, 1017
CHARACTER_LENGTH(), 1086
character_set_client システム変数, 470
character_set_connection システム変数, 471
character_set_database システム変数, 471
character_set_filesystem システム変数, 471
character_set_results システム変数, 472
character_set_server システム変数, 472
character_set_system システム変数, 472
CHARACTER_SETS
INFORMATION_SCHEMA テーブル, 2502
character_sets_dir システム変数, 472
CHARACTER データ型, 1017
CHARSET(), 1171
charset オプション
comp_err, 254
charset コマンド
mysql, 277
CHAR データ型, 1017, 1033
check-only-changed オプション
myisamchk, 340
mysqlcheck, 296
check-orphans オプション
ndb_blob_tool, 2239
check-upgrade オプション
mysqlcheck, 296

CHECK TABLE, 1430
  およびパーティショニング, 2450
Checking master version
  スレッドの状態, 912
checking permissions
  スレッドの状態, 906
checking privileges on cached query
  スレッドの状態, 912
checking query cache for query
  スレッドの状態, 912
Checking table
  スレッドの状態, 906
CHECKPOINT イベント (MySQL Cluster), 2302
checkpoint オプション
  mysqlhotcopy, 382
Checksum, 2134, 2140
Checksum (MySQL Cluster), 2138
CHECKSUM TABLE, 1433
check オプション
  myisamchk, 340, 340
  mysqlcheck, 296
chroot オプション
  mysqld, 428
  mysqlhotcopy, 382
CJK (中国語、日本語、韓国語)
  Access、PHP など, 2861
  Access、PHP などの問題., 2861
  big5, 2861
  Big5 文字セットの問題 (中国語), 2861
  CJKV, 2861
  euckr 文字セットの問題 (韓国語), 2861
  FAQ, 2861
  gb2312、gbk, 2861
  GB 文字セットの問題 (中国語), 2861
  LIKE および FULLTEXT, 2861
  LIKE および FULLTEXT の問題, 2861
  MySQL 4.0 の動作, 2861
  ORDER BY の扱い, 2861, 2861
  Unicode の照合順序, 2861
  円記号, 2861
  円記号の問題 (日本語), 2861
  韓国語のドキュメント, 2861
  韓国語の文字セット, 2861
  疑問符として表示される文字, 2861
  使用可能な文字セット, 2861
  照合順序, 2861, 2861
  ソート順の問題, 2861, 2861
  ソートの問題, 2861, 2861
  中国語のドキュメント, 2861
  データの切り捨て, 2861
  データの切り捨ての問題, 2861
  データベース名とテーブル名, 2861
  特殊文字を使用できるか, 2861
  日本語のドキュメント, 2861
  日本語の文字セット, 2861
  日本語の文字セットの変換の問題, 2861
  ベトナム語, 2861
  文字が拒否される, 2861
  文字を使用できるかどうかのテスト, 2861
CJK 文字でのデータの切り捨て, 2861
cleaning up
  スレッドの状態, 906
clear command
  mysql, 277
Clearing
  スレッドの状態, 915
CLOSE, 1394
Close stmt
  スレッドのコマンド, 904
closing tables
  スレッドの状態, 906
cluster.binlog_index テーブル (OBSOLETE) (参照 MySQL Cluster レプリケーション)
cluster_operations
  ndbinfo テーブル, 2324
cluster_replication データベース (OBSOLETE) (参照 MySQL Cluster レプリケーション)
cluster_transactions
  ndbinfo テーブル, 2325
CLUSTERLOG STATISTICS コマンド (MySQL Cluster), 2306
CLUSTERLOG コマンド (MySQL Cluster), 2300
CMake
  BUILD_CONFIG オプション, 154
  CMAKE_BUILD_TYPE オプション, 154
  CMAKE_C_FLAGS オプション, 160
  CMAKE_CXX_FLAGS オプション, 160
  CMAKE_INSTALL_PREFIX オプション, 154
  COMPILATION_COMMENT オプション, 156
  CPACK_MONOLITHIC_INSTALL オプション, 154
  DEFAULT_CHARSET オプション, 156
  DEFAULT_COLLATION オプション, 156
  ENABLE_DEBUG_SYNC オプション, 157
  ENABLE_DOWNLOADS オプション, 157
  ENABLE_DTRACE オプション, 157
  ENABLE_GCOV オプション, 157
  ENABLE_GPROF オプション, 157
  ENABLED_LOCAL_INFILE オプション, 157
  ENABLED_PROFILING オプション, 157
  IGNORE_AIO_CHECK オプション, 157
  INNODB_PAGE_ATOMIC_REF_COUNT オプション, 157
  INSTALL_BINDIR オプション, 154
  INSTALL_DOCDIR オプション, 154
  INSTALL_DOCREADMEDIR オプション, 154
  INSTALL_INCLUDEDIR オプション, 154
  INSTALL_INFODIR オプション, 154
  INSTALL_LAYOUT オプション, 154
  INSTALL_LIBDIR オプション, 155
  INSTALL_MANDIR オプション, 155
  INSTALL_MYSQLSHAREDIR オプション, 155
  INSTALL_MYSQLTESTDIR オプション, 155
  INSTALL_PLUGINDIR オプション, 155
  INSTALL_SBINDIR オプション, 155
  INSTALL_SCRIPTDIR オプション, 155
  INSTALL_SHAREDIR オプション, 155
  INSTALL_SQLBENCHDIR オプション, 155
  INSTALL_SUPPORTFILESDIR オプション, 155
  MEMCACHED_HOME オプション, 161
  MYSQL_DATADIR オプション, 155
  MYSQL_MAINTAINER_MODE オプション, 158
  MYSQL_PROJECT_NAME オプション, 158
  MYSQL_TCP_PORT オプション, 158
  MYSQL_UNIX_ADDR オプション, 158
  ODBC_INCLUDES オプション, 155
  ODBC_LIB_DIR オプション, 155
  OPTIMIZER_TRACE オプション, 158
  SUNPRO_CXX_LIBRARY オプション, 160
  SYSCONFDIR オプション, 155

TMPDIR オプション, 156
VERSION ファイル, 163
WITH_ASAN オプション, 158
WITH_BUNDLED_LIBEVENT オプション, 161
WITH_BUNDLED_MEMCACHED オプション, 161
WITH_CLASSPATH オプション, 161
WITH_DEBUG オプション, 158
WITH_DEFAULT_COMPILER_OPTIONS オプション, 160
WITH_DEFAULT_FEATURE_SET オプション, 158
WITH_EDITLINE オプション, 158
WITH_EMBEDDED_SERVER オプション, 158
WITH_EMBEDDED_SHARED_LIBRARY オプション, 158
WITH_ERROR_INSERT オプション, 161
WITH_EXTRA_CHARSETS オプション, 159
WITH_INNODB_MEMCACHED オプション, 159
WITH_LIBEDIT オプション, 159
WITH_LIBEVENT オプション, 159
WITH_LIBWRAP オプション, 159
WITH_NDB_BINLOG オプション, 161
WITH_NDB_DEBUG オプション, 161
WITH_NDB_JAVA オプション, 162
WITH_NDB_PORT オプション, 162
WITH_NDB_TEST オプション, 162
WITH_NDBCLUSTER_STORAGE_ENGINE オプション, 161
WITH_NDBCLUSTER オプション, 161
WITH_NDBMTD オプション, 161
WITH_READLINE オプション, 159
WITH_SSL オプション, 159
WITH_UNIXODBC オプション, 159
WITH_VALGRIND オプション, 159
WITH_ZLIB オプション, 160
WITHOUT_SERVER オプション, 160
以前の起動後の実行, 146, 162
オプション, 150
CMAKE_BUILD_TYPE オプション
CMake, 154
CMAKE_C_FLAGS オプション
CMake, 160
CMAKE_CXX_FLAGS オプション
CMake, 160
CMAKE_INSTALL_PREFIX オプション
CMake, 154
CMakeCache.txt ファイル, 162
COALESCE(), 1077
COERCIBILITY(), 1171
collation-server オプション
mysqld, 428
COLLATION(), 1172
COLLATION_CHARACTER_SET_APPLICABILITY
INFORMATION_SCHEMA テーブル, 2503
collation_connection システム変数, 472
collation_database システム変数, 472
collation_server システム変数, 473
COLLATIONS
INFORMATION_SCHEMA テーブル, 2502
column-names オプション
mysql, 270
column-type-info オプション
mysql, 270
COLUMN_PRIVILEGES
INFORMATION_SCHEMA テーブル, 2504
COLUMNS
INFORMATION_SCHEMA テーブル, 2503
columns オプション
mysqlimport, 318
COLUMNS パーティショニング, 2422
comments オプション
mysql, 270
mysqldump, 307
COMMIT, 28, 1355
XA トランザクション, 1369
committing alter table to storage engine
スレッドの状態, 906
Committing events to binlog
スレッドの状態, 914
commit オプション
mysqlaccess, 360
mysqlslap, 329
comp_err, 226, 254
charset オプション, 254
debug-info オプション, 254
debug オプション, 254
header_file オプション, 254
help オプション, 254
in_file オプション, 254
name_file オプション, 254
out_dir オプション, 254
out_file オプション, 255
statefile オプション, 255
version オプション, 255
compact オプション
mysqldump, 309
compatible オプション
mysqldump, 309
COMPILATION_COMMENT オプション
CMake, 156
complete-insert オプション
mysqldump, 310
completion_type システム変数, 473
COMPRESS(), 1166
CompressedBackup, 2099
CompressedLCP, 2099
compress オプション
mysql, 270
mysql_upgrade, 264
mysqladmin, 290
mysqlcheck, 296
mysqldump, 304
mysqlimport, 318
mysqlshow, 323
mysqlslap, 329
[computer] (MySQL Cluster), 2157
CONCAT(), 1086
CONCAT_WS(), 1086
concurrency オプション
mysqlslap, 329
concurrent_insert システム変数, 473
cond_instances テーブル
performance_schema, 2602
config-cache オプション (ndb_mgmd), 2232
config-file オプション
my_print_defaults, 389
ndb_config, 2242
config-file オプション (ndb_mgmd), 2233
config.ini (MySQL Cluster), 2059, 2069, 2070, 2236
config_from_node オプション
ndb_config, 2241

config_options テーブル, 1745
config_params
  ndbinfo テーブル, 2325
configdir オプション (ndb_mgmd), 2232
configinfo オプション
  ndb_config, 2245
configure オプション
  MySQLInstallerConsole, 86
connect-delay オプション (ndbd), 2227
connect-delay オプション (ndbmtd), 2227
connect-expired-password オプション
  mysql, 270
connect-retries オプション (ndbd), 2226
connect-retries オプション (ndbmtd), 2226
connect-string オプション (MySQL Cluster プログラム), 2285
Connect
  スレッドのコマンド, 904
Connect Out
  スレッドのコマンド, 904
connect_timeout システム変数, 474
connect_timeout 変数, 276, 293
ConnectBackoffMaxTime, 2132
ConnectCheckIntervalDelay, 2102
Connecting to master
  スレッドの状態, 912
connection-server-id オプション
  mysqlbinlog, 365
connection-timeout option (ndb_error_reporter), 2253
CONNECTION_ID(), 1172
ConnectionMap, 2128
connections オプション
  ndb_config, 2243
CONNECTION イベント (MySQL Cluster), 2302
Connector/C++, 2653, 2657
Connector/C, 2653
Connector/J, 2653, 2656
Connector/Net, 2653, 2656
Connector/ODBC, 2653, 2656
Connector/Python, 2653, 2657
Connector, 2653
connect コマンド
  mysql, 277
console オプション
  mysqld, 428
CONSTRAINTS
  INFORMATION_SCHEMA テーブル, 2520
const テーブル
  オプティマイザ, 849, 1332
containers テーブル, 1745
Contains(), 1191
CONV(), 1108
CONVERT, 1150
CONVERT TO, 1233
CONVERT_TZ(), 1116
converting HEAP to MyISAM
  スレッドの状態, 906
copy to tmp table
  スレッドの状態, 906
Copying to group table
  スレッドの状態, 906
Copying to tmp table
  スレッドの状態, 906
Copying to tmp table on disk
  スレッドの状態, 906
copy オプション
  mysqlaccess, 360
core-file-size オプション
  mysqld_safe, 247
core-file オプション
  mysqld, 428
core-file オプション (MySQL Cluster プログラム), 2286
core_file システム変数, 474
correct-checksum オプション
  myisamchk, 341
COS(), 1108
COT(), 1108
COUNT(), 1208
COUNT(DISTINCT), 1208
counters
  ndbinfo テーブル, 2325
count オプション
  myisam_ftdump, 334
  mysqladmin, 290
  mysqlshow, 323
CPACK_MONOLITHIC_INSTALL オプション
  CMake, 154
CPU バウンド, 2992
CR_SERVER_GONE_ERROR, 2947
CR_SERVER_LOST_ERROR, 2947
crash-me, 902
crash-me プログラム, 901
crash-safe レプリケーション, 1915
CrashOnCorruptedTuple, 2098
CRC32(), 1108
create-options オプション
  mysqldump, 310
create-schema オプション
  mysqlslap, 329
CREATE ... IF NOT EXISTS
  およびレプリケーション, 1981
CREATE DATABASE, 1245
Create DB
  スレッドのコマンド, 904
CREATE EVENT, 1246
  およびレプリケーション, 1988
CREATE FUNCTION, 1254, 1438
CREATE INDEX, 1250, 1576
CREATE LOGFILE GROUP, 1253
  (参照 MySQL Cluster ディスクデータ)
CREATE NODEGROUP コマンド (MySQL Cluster), 2291
CREATE PROCEDURE, 1254
CREATE SCHEMA, 1245
CREATE SERVER, 1259
CREATE TABLE ... SELECT
  およびレプリケーション, 1981
CREATE TABLE, 1260
  DIRECTORY オプション
    およびレプリケーション, 1987
  KEY_BLOCK_SIZE, 1559
  ROW_FORMAT, 1571
  テーブル圧縮のオプション, 1555
CREATE TABLESPACE, 1285
  (参照 MySQL Cluster ディスクデータ)
CREATE TRIGGER, 1287
CREATE USER, 1415
CREATE VIEW, 1289
CREATE_ASYMMETRIC_PRIV_KEY(), 1199
CREATE_ASYMMETRIC_PUB_KEY(), 1200

CREATE_DH_PARAMETERS(), 1200
CREATE_DIGEST(), 1200
create オプション
  mysqlslap, 329
Creating delayed handler
  スレッドの状態, 911
Creating index
  スレッドの状態, 906
Creating sort index
  スレッドの状態, 906
creating table
  スレッドの状態, 906
Creating tmp table
  スレッドの状態, 906
cross-bootstrap オプション
  mysql_install_db, 256
CROSS JOIN, 1335
Crosses(), 1191
CRUD, 2992
csv オプション
  mysqlslap, 329
CSV ストレージエンジン, 1759, 1774
CSV データ、読み取り, 1316, 1334
(Ctrl+Z) \Z, 918, 1318
CURDATE(), 1116
CURRENT_DATE, 1116
CURRENT_TIME, 1116
CURRENT_TIMESTAMP, 1116
CURRENT_USER(), 1172
CURTIME(), 1117
cxxflags オプション
  mysql_config, 388
CXX 環境変数, 162, 192
C プリペアドステートメント API
  関数, 2726, 2727
  タイプコード, 2725

D

Daemon
  スレッドのコマンド, 904
daemon_memcached_enable_binlog システム変数, 1611
daemon_memcached_engine_lib_name システム変数, 1611
daemon_memcached_engine_lib_path システム変数, 1612
daemon_memcached_option システム変数, 1612
daemon_memcached_r_batch_size システム変数, 1612
daemon_memcached_w_batch_size システム変数, 1613
daemon オプション (ndb_mgmd), 2233
data-file-length オプション
  myisamchk, 341
DATA DIRECTORY
  およびレプリケーション, 1987
DATABASE(), 1173
database option
  ndb_desc, 2251
databases オプション
  mysqlcheck, 296
  mysqldump, 312
database オプション
  mysql, 270
  mysqlbinlog, 366
  ndb_blob_tool, 2239
  ndb_show_tables, 2279
database オプション (ndb_index_stat), 2256
DataDir, 2079, 2083
datadir オプション
  mysql.server, 250
  mysql_install_db, 256
  mysql_plugin, 259
  mysql_upgrade, 264
  mysqld, 429
  mysqld_safe, 247
datadir システム変数, 475
DataMemory, 2084
DATE, 2961
DATE(), 1117
DATE_ADD(), 1117
DATE_FORMAT(), 1119
date_format システム変数, 475
DATE_SUB(), 1117, 1120
DATEDIFF(), 1117
datetime_format システム変数, 475
DATETIME データ型, 1015, 1023
DATE カラム
  問題, 2961
DATE データ型, 1015, 1023
DAY(), 1120, 1170
DAYNAME(), 1120
DAYOFMONTH(), 1120
DAYOFWEEK(), 1121
DAYOFYEAR(), 1121
DB2 SQL モード, 592
DBI->quote, 919
DBI->trace, 2825
DBI/DBD インタフェース, 2763
DBI_TRACE 環境変数, 192, 2825
DBI_USER 環境変数, 192
DBI インタフェース, 2763
DBUG パッケージ, 2829
db オプション
  mysqlaccess, 361
db テーブル
  ソート, 675
DCL, 2992
DCL, 1418, 1427
DDL, 2992
DDL, 1222
DDL ログ, 618
DEALLOCATE PREPARE, 1383, 1386
debug-check オプション
  mysql, 270
  mysql_upgrade, 264
  mysqladmin, 290
  mysqlbinlog, 366
  mysqlcheck, 296
  mysqldump, 307
  mysqlimport, 318
  mysqlshow, 323
  mysqlslap, 329
debug-info オプション
  comp_err, 254
  mysql, 270
  mysql_upgrade, 264
  mysqladmin, 290
  mysqlbinlog, 367
  mysqlcheck, 297
  mysqldump, 307
  mysqlimport, 318
  mysqlshow, 323

mysqlslap, 329
debug-level オプション
ndb_setup.py, 2277
debug-sync-timeout オプション
mysqld, 429
Debug
スレッドのコマンド, 904
debug_sync システム変数, 476
debug オプション
comp_err, 254
my_print_defaults, 389
myisamchk, 337
myisampack, 350
mysql, 270
mysql_config_editor, 359
mysql_upgrade, 264
mysqlaccess, 361
mysqladmin, 290
mysqlbinlog, 366
mysqlcheck, 296
mysqld, 429
mysqldump, 307
mysqldumpslow, 380
mysqlhotcopy, 382
mysqlimport, 318
mysqlshow, 323
mysqlslap, 329
debug オプション (MySQL Cluster プログラム), 2286
debug システム変数, 475
deb ファイル
MySQL APT リポジトリ, 129
MySQL SLES リポジトリ, 129
DECIMAL データ型, 1013, 1214
DECLARE, 1388
DECODE(), 1166
decode_bits myisamchk 変数, 338
DEC データ型, 1014
default-authentication-plugin オプション
mysqld, 429
default-auth オプション
mysql, 270
mysql_upgrade, 264
mysqladmin, 290
mysqlbinlog, 367
mysqlcheck, 297
mysqldump, 304
mysqlimport, 318
mysqlshow, 323
mysqlslap, 329
default-character-set オプション
mysql, 270
mysql_upgrade, 264
mysqladmin, 290
mysqlcheck, 297
mysqldump, 308
mysqlimport, 318
mysqlshow, 323
default-storage-engine オプション
mysqld, 430
default-time-zone オプション
mysqld, 430
DEFAULT
制約, 33
DEFAULT(), 1201
DEFAULT value 句, 1054
DEFAULT_CHARSET オプション
CMake, 156
DEFAULT_COLLATION オプション
CMake, 156
default_storage_engine システム変数, 476
default_tmp_storage_engine システム変数, 476
default_week_format システム変数, 477
DefaultHashMapSize, 2091, 2131
DefaultOperationRedoProblemAction
API および SQL ノード, 2131
defaults-extra-file オプション, 240, 256
my_print_defaults, 389
myisamchk, 338
mysql, 270
mysql_upgrade, 264
mysqladmin, 290
mysqlbinlog, 367
mysqlcheck, 297
mysqld, 431
mysqld_multi, 251
mysqld_safe, 247
mysqldump, 305
mysqlimport, 318
mysqlshow, 323
mysqlslap, 329
defaults-file オプション, 240, 257
my_print_defaults, 389
myisamchk, 338
mysql, 271
mysql_upgrade, 264
mysqladmin, 291
mysqlbinlog, 367
mysqlcheck, 297
mysqld, 431
mysqld_multi, 251
mysqld_safe, 247
mysqldump, 305
mysqlimport, 318
mysqlshow, 324
mysqlslap, 329
defaults-group-suffix オプション, 240
my_print_defaults, 389
myisamchk, 338
mysql, 271
mysql_upgrade, 264
mysqladmin, 291
mysqlbinlog, 367
mysqlcheck, 297
mysqld, 431
mysqldump, 306
mysqlimport, 318
mysqlshow, 324
mysqlslap, 330
DEFAULT 値句, 1264
DEGREES(), 1108
delay-key-write オプション
mysqld, 431, 1766
delay_key_write システム変数, 477
delayed-insert オプション
mysqldump, 313
DELAYED, 1310
無視される場合, 1308
Delayed insert

スレッドのコマンド, 904
delayed_insert_limit, 1311
delayed_insert_limit システム変数, 478
delayed_insert_timeout システム変数, 478
delayed_queue_size システム変数, 478
delay オプション (ndbinfo_select_all), 2229
delete-master-logs オプション
mysqldump, 308
delete-orphans オプション
ndb_blob_tool, 2239
DELETE, 1300
および MySQL Cluster, 2025
delete オプション
mysqlimport, 319
delete オプション (ndb_index_stat), 2256
deleting from main table
スレッドの状態, 906
deleting from reference tables
スレッドの状態, 907
delimiter オプション
mysql, 271
mysqlslap, 330
ndb_select_all, 2274
delimiter コマンド
mysql, 277
demo_test テーブル, 1729
des-key-file オプション
mysqld, 431
DES_DECRYPT(), 1166
DES_ENCRYPT(), 1166
DESC, 1488
descending オプション
ndb_select_all, 2274
DESCRIBE, 215, 1488
description オプション
myisamchk, 342
detach オプション
mysqlslap, 330
dict_obj_types
ndbinfo テーブル, 2326
Dimension(), 1184
disable-gtid-unsafe-statements オプション, 1943
disable-indexes オプション (ndb_restore), 2271
disable-keys オプション
mysqldump, 313
disable-log-bin オプション
mysqlbinlog, 367
disable named commands オプション
mysql, 271
disable_gtid_unsafe_statements システム変数, 1945
DISCARD TABLESPACE, 1233, 1541
discard_or_import_tablespace
スレッドの状態, 907
disconnect-slave-event-count オプション
mysqld, 1899
disconnect_on_expired_password システム変数, 479
Disjoint(), 1192
disk_write_speed_aggregate
ndbinfo テーブル, 2327
disk_write_speed_aggregate_node
ndbinfo テーブル, 2329
disk_write_speed_base
ndbinfo テーブル, 2327
DiskCheckpointSpeed, 2105
DiskCheckpointSpeedInRestart, 2105
DiskIOThreadPool, 2122, 2124
Diskless, 2098
diskpagebuffer
ndbinfo テーブル, 2330
DiskPageBufferMemory, 2121, 2124
DiskSyncSize, 2105
disk オプション
ndb_select_all, 2274
DISTINCT, 206, 806, 1332
AVG(), 1208
COUNT(), 1208
MAX(), 1209
MIN(), 1209
SUM(), 1210
DISTINCTROW, 1332
DIV, 1106
div_precision_increment システム変数, 479
DML, 2992
DML, 1298
DELETE ステートメント, 1300
INSERT ステートメント, 1305
UPDATE ステートメント, 1353
DNS, 895
DO, 1303
DocBook XML
ドキュメントソース形式, 2
dont_ignore_systab_0 オプション (ndb_restore), 2268
DOUBLE PRECISION データ型, 1014
DOUBLE データ型, 1014
DROP ... IF EXISTS
およびレプリケーション, 1983
DROP DATABASE, 1292
Drop DB
スレッドのコマンド, 904
DROP EVENT, 1293
DROP FOREIGN KEY, 1233, 1283
DROP FUNCTION, 1295, 1438
DROP INDEX, 1231, 1294, 1576
DROP LOGFILE GROUP, 1294
(参照 MySQL Cluster ディスクデータ)
DROP NODEGROUP コマンド (MySQL Cluster), 2291
DROP PREPARE, 1386
DROP PRIMARY KEY, 1231
DROP PROCEDURE, 1295
DROP SCHEMA, 1293
DROP SERVER, 1295
DROP TABLE, 1295
および MySQL Cluster, 2025
DROP TABLESPACE, 1296
(参照 MySQL Cluster ディスクデータ)
DROP TRIGGER, 1296
DROP USER, 1418
DROP VIEW, 1296
dry-scp option (ndb_error_reporter), 2254
dryrun オプション
mysqlhotcopy, 382
DTrace, 626
および memcached, 1816
DUAL, 1329
dump-date オプション
mysqldump, 307
dump-file オプション
ndb_blob_tool, 2239

dump-slave オプション
  mysqldump, 308
DUMPFILE, 1334
dump オプション
  myisam_ftdump, 334
dump オプション (ndb_index_stat), 2256

E

edit コマンド
  mysql, 278
ego コマンド
  mysql, 278
Eiffel ラッパー, 2765
ELT(), 1086
embedded オプション
  mysql_config, 388
enable-cleartext-plugin オプション
  mysql, 271
  mysqladmin, 291
  mysqlslap, 330
enable-named-pipe オプション
  mysqld, 431
ENABLE_DEBUG_SYNC オプション
  CMake, 157
ENABLE_DOWNLOADS オプション
  CMake, 157
ENABLE_DTRACE オプション
  CMake, 157
ENABLE_GCOV オプション
  CMake, 157
ENABLE_GPROF オプション
  CMake, 157
ENABLED_LOCAL_INFILE オプション
  CMake, 157
ENABLED_PROFILING オプション
  CMake, 157
ENCODE(), 1167
ENCRYPT(), 1167
end
  スレッドの状態, 907
END, 1387
end_markers_in_json システム変数, 480
EndPoint(), 1185
enforce-gtid-consistency オプション, 1944
enforce_gtid_consistency システム変数, 1946
engine_condition_pushdown システム変数, 480
ENGINES
  INFORMATION_SCHEMA テーブル, 2504
engine オプション
  mysqlslap, 330
ENTER SINGLE USER MODE コマンド (MySQL Cluster), 2291
ENUM
  サイズ, 1058
ENUM データ型, 1019, 1037
Envelope(), 1184
eq_ref 結合型
  オプティマイザ, 850
Equals(), 1192
Error
  スレッドのコマンド, 904
error_count システム変数, 481
ERROR_FOR_DIVISION_BY_ZERO SQL モード, 587
ERROR イベント (MySQL Cluster), 2305
event-scheduler オプション
  mysqld, 432
event_scheduler システム変数, 481
EventLogBufferSize, 2109
EVENTS
  INFORMATION_SCHEMA テーブル, 2486, 2504
events_stages_current テーブル
  performance_schema, 2610
events_stages_history_long テーブル
  performance_schema, 2611
events_stages_history テーブル
  performance_schema, 2611
events_stages_summary_by_account_by_event_name table
  performance_schema, 2629
events_stages_summary_by_host_by_event_name table
  performance_schema, 2629
events_stages_summary_by_thread_by_event_name テーブル
  performance_schema, 2623
events_stages_summary_by_user_by_event_name テーブル
  performance_schema, 2629
events_stages_summary_global_by_event_name テーブル
  performance_schema, 2623
events_statements_current テーブル
  performance_schema, 2614
events_statements_history_long テーブル
  performance_schema, 2617
events_statements_history テーブル
  performance_schema, 2617
events_statements_summary_by_account_by_event_name テーブル
  performance_schema, 2629
events_statements_summary_by_digest テーブル
  performance_schema, 2623
events_statements_summary_by_host_by_event_name テーブル
  performance_schema, 2629
events_statements_summary_by_thread_by_event_name テーブル
  performance_schema, 2623
events_statements_summary_by_user_by_event_name テーブル
  performance_schema, 2629
events_statements_summary_global_by_event_name テーブル
  performance_schema, 2623
events_waits_current テーブル
  performance_schema, 2607
events_waits_history_long テーブル
  performance_schema, 2609
events_waits_history テーブル
  performance_schema, 2609
events_waits_summary_by_account_by_event_name テーブル
  performance_schema, 2629
events_waits_summary_by_host_by_event_name テーブル
  performance_schema, 2629
events_waits_summary_by_instance テーブル
  performance_schema, 2622
events_waits_summary_by_thread_by_event_name テーブル
  performance_schema, 2622
events_waits_summary_by_user_by_event_name テーブル
  performance_schema, 2629
events_waits_summary_global_by_event_name テーブル

performance_schema, 2622
events オプション
mysqldump, 312
example オプション
mysqld_multi, 251
EXAMPLE ストレージエンジン, 1759, 1789
exclude-databases オプション (ndb_restore), 2269
exclude-gtids オプション
mysqlbinlog, 367
exclude-intermediate-sql-tables オプション (ndb_restore), 2272
exclude-missing-columns オプション (ndb_restore), 2270
exclude-missing-tables オプション (ndb_restore), 2271
exclude-tables オプション (ndb_restore), 2269
Execute
スレッドのコマンド, 904
EXECUTE, 1383, 1386
ExecuteOnComputer, 2077, 2081, 2128
execute オプション
mysql, 271
execute オプション (ndb_mgm), 2238
executing
スレッドの状態, 907
Execution of init_command
スレッドの状態, 907
EXISTS
サブクエリーを使用した, 1347
exit-info オプション
mysqld, 432
EXIT SINGLE USER MODE コマンド (MySQL Cluster), 2291
exit コマンド
mysql, 278
EXIT コマンド (MySQL Cluster), 2291
EXP(), 1108
expire_logs_days システム変数, 481
EXPLAIN, 846, 1488
EXPLAIN PARTITIONS, 2451, 2451
explicit_defaults_for_timestamp システム変数, 481
EXPORT_SET(), 1086
extend-check オプション
myisamchk, 340, 341
extended-insert オプション
mysqldump, 313
extended オプション
mysqlcheck, 297
ExteriorRing(), 1187
external-locking オプション
mysqld, 432
external_user システム変数, 482
extra-file オプション
my_print_defaults, 389
extra-node-info option
ndb_desc, 2251
extra-partition-info option
ndb_desc, 2251
EXTRACT(), 1121
ExtractValue(), 1154
ExtraSendBufferMemory
API ノード, 2130
データノード, 2125

F

FALSE, 919, 922
テスト, 1076, 1076
FAQ
C API, 2754
MySQL Cluster, 2850
コネクタおよび API, 2871
レプリケーション, 2871
fast オプション
myisamchk, 340
mysqlcheck, 297
FEDERATED ストレージエンジン, 1759, 1784
Fetch
スレッドのコマンド, 904
FETCH, 1394
FIELD(), 1087
Field List
スレッドのコマンド, 904
fields-enclosed-by オプション
mysqldump, 310, 319
fields-enclosed-by オプション (ndb_restore), 2267
fields-escaped-by オプション
mysqldump, 310, 319
fields-optionally-enclosed-by オプション
mysqldump, 310, 319
fields-optionally-enclosed-by オプション (ndb_restore), 2267
fields-terminated-by オプション
mysqldump, 310, 319
fields-terminated-by オプション (ndb_restore), 2267, 2267
fields オプション
ndb_config, 2244
file-per-table, 2992
FILE, 1088
file_instances テーブル
performance_schema, 2602
file_summary_by_event_name テーブル
performance_schema, 2625
file_summary_by_instance テーブル
performance_schema, 2625
FILES
INFORMATION_SCHEMA テーブル, 2552
filesort の最適化, 802
FileSystemPath, 2083
FileSystemPathDataFiles, 2122
FileSystemPathDD, 2122
FileSystemPathUndoFiles, 2123
FIND_IN_SET(), 1087
Finished reading one binlog; switching to next binlog
スレッドの状態, 912
fix-db-names オプション
mysqlcheck, 297
fix-table-names オプション
mysqlcheck, 297
FIXED データ型, 1014
FLOAT データ型, 1014, 1014, 1014
FLOOR(), 1109
flush-logs オプション
mysqldump, 313
flush-privileges オプション
mysqldump, 314
FLUSH, 1481
およびレプリケーション, 1989
flush_time システム変数, 483
Flushing tables
スレッドの状態, 907
flushlog オプション
mysqlhotcopy, 382

flush オプション
  mysqld, 432
flush システム変数, 482
FOR UPDATE, 1332
force-if-open オプション
  mysqlbinlog, 367
force-read オプション
  mysqlbinlog, 367
FORCE INDEX, 1340, 2966
FORCE KEY, 1340
force オプション
  myisamchk, 340, 341
  myisampack, 350
  mysql, 271
  mysql_convert_table_format, 384
  mysql_install_db, 257
  mysql_upgrade, 264
  mysqladmin, 291
  mysqlcheck, 297
  mysqldump, 307
  mysqlimport, 319
FOREIGN KEY 制約, 2993
foreign_key_checks システム変数, 483
FORMAT(), 1087
FOUND_ROWS(), 1173
FragmentLogFileSize, 2092
FreeBSD トラブルシューティング, 163
freeing items
  スレッドの状態, 907
FROM, 1329
FROM_BASE64()(), 1087
FROM_DAYS(), 1121
FROM_UNIXTIME(), 1121
fs option (ndb_error_reporter), 2254
ft_boolean_syntax システム変数, 483
ft_max_word_len myisamchk 変数, 338
ft_max_word_len システム変数, 484
ft_min_word_len myisamchk 変数, 338
ft_min_word_len システム変数, 484
ft_query_expansion_limit システム変数, 484
ft_stopword_file myisamchk 変数, 338
ft_stopword_file システム変数, 485
FTS, 2993
FULLTEXT, 1132
FULLTEXT initialization
  スレッドの状態, 907
FULLTEXT インデックス
  InnoDB, 1515
FULLTEXT インデックス, 2993
fulltext 結合型
  オプティマイザ, 850

G

GA, 2993
gb2312、gbk, 2861
gci64 オプション
  ndb_select_all, 2274
gci オプション
  ndb_select_all, 2274
GCP 停止エラー (MySQL Cluster), 2124
gdb
  使用, 2824
gdb オプション
  mysqld, 433
general-log オプション
  mysqld, 433
General Public License, 4
general_log_file システム変数, 485
general_log システム変数, 485
GeomCollFromText(), 1182
GeomCollFromWKB(), 1182
GeometryCollection(), 1183
GeometryCollectionFromText(), 1182
GeometryCollectionFromWKB(), 1182
GEOMETRYCOLLECTION データ型, 1043
GeometryFromText(), 1182
GeometryFromWKB(), 1182
GeometryN(), 1188
GeometryType(), 1184
GEOMETRY データ型, 1043
GeomFromText(), 1182, 1184
GeomFromWKB(), 1182, 1184
GET DIAGNOSTICS, 1398
GET_FORMAT(), 1121
GET_LOCK(), 1201
GIS, 1041
GLength(), 1185, 1187
GLOBAL_STATUS
  INFORMATION_SCHEMA テーブル, 2507
global_transaction, 2993
GLOBAL_VARIABLES
  INFORMATION_SCHEMA テーブル, 2508
Google Test, 157
got handler lock
  スレッドの状態, 911
got old table
  スレッドの状態, 911
go コマンド
  mysql, 278
GRANT, 1418
GRANTS, 1457
GRANT ステートメント, 682
GREATEST(), 1077
GROUP BY, 804
  エイリアス, 1214
  標準 SQL の拡張, 1213
GROUP BY 関数, 1207
GROUP_CONCAT(), 1209
group_concat_max_len システム変数, 485
>= (より多いか等しい), 1076
>> (右シフト), 1162
gtid-mode オプション (mysqld), 1945
gtid_done システム変数, 1946
gtid_executed システム変数, 1946
gtid_lost システム変数, 1947
gtid_mode システム変数, 1947
gtid_next システム変数, 1947
gtid_owned システム変数, 1948
gtid_purged システム変数, 1948
GTID_SUBSET(), 1193
GTID_SUBTRACT(), 1194
GTID 関数, 1193
> (より多い), 1076

H

HANDLER, 1304
HASH パーティショニング, 2428
HASH パーティション

管理, 2444
分割およびマージ, 2444
have_compress システム変数, 486
have_crypt システム変数, 486
have_csv システム変数, 486
have_dynamic_loading システム変数, 486
have_geometry システム変数, 486
have_innodb システム変数, 486
have_openssl システム変数, 486
have_partitioning システム変数, 486
have_profiling システム変数, 486
have_query_cache システム変数, 486
have_rtree_keys システム変数, 486
have_ssl システム変数, 487
have_symlink システム変数, 487
HAVING, 1330
HDD, 2993
header_file オプション
comp_err, 254
header オプション
ndb_select_all, 2274
HEAP ストレージエンジン, 1759, 1771
HeartbeatIntervalDbApi, 2101
HeartbeatIntervalDbDb, 2100
HeartbeatIntervalMgmdMgmd
管理ノード, 2080
HeartbeatOrder, 2101
HeartbeatThreadPriority, 2079, 2130
help オプション
comp_err, 254
my_print_defaults, 389
myisam_ftdump, 334
myisamchk, 337
myisampack, 350
mysql, 269
mysql_config_editor, 359
mysql_convert_table_format, 384
mysql_find_rows, 385
mysql_install_db, 256
mysql_plugin, 259
mysql_setpermission, 386
mysql_upgrade, 263
mysql_waitpid, 386
mysqlaccess, 360
mysqladmin, 290
mysqlbinlog, 364
mysqlcheck, 295
mysqld, 425
mysqld_multi, 251
mysqld_safe, 247
mysqldump, 307
mysqldumpslow, 380
mysqlhotcopy, 382
mysqlimport, 318
MySQLInstallerConsole, 87
mysqlshow, 323
mysqlslap, 328
ndb_setup.py, 2277
perror, 391
resolve_stack_dump, 390
resolveip, 392
HELP オプション
myisamchk, 337
help オプション (MySQL Cluster プログラム), 2285
help コマンド
mysql, 277
HELP コマンド (MySQL Cluster), 2289
HELP ステートメント, 1489
hex-blob オプション
mysqldump, 310
HEX(), 1087, 1109
hexdump オプション
mysqlbinlog, 367
hex オプション (ndb_restore), 2267
HIGH_NOT_PRECEDENCE SQL モード, 587
HIGH_PRIORITY, 1332
histignore オプション
mysql, 271
HOME 環境変数, 192, 281
Host*ScId* パラメータ, 2138
host.frm
検出の問題, 166
host_cache テーブル
performance_schema, 2631
HostName1, 2133, 2136, 2139
HostName2, 2133, 2137, 2139
HostName, 2077, 2081, 2129
HostName (MySQL Cluster), 2341
hostname システム変数, 487
hosts テーブル
performance_schema, 2619
host オプション, 232
mysql, 271
mysql_config_editor, 359
mysql_convert_table_format, 384
mysql_setpermission, 386
mysql_upgrade, 264
mysqlaccess, 361
mysqladmin, 291
mysqlbinlog, 367
mysqlcheck, 297
mysqldump, 304
mysqlhotcopy, 382
mysqlimport, 319
mysqlshow, 324
mysqlslap, 330
ndb_config, 2243
host テーブル
ソート, 675
HOUR(), 1122
howto オプション
mysqlaccess, 361
html オプション
mysql, 271

I

i-am-a-dummy オプション
mysql, 274
ib-file セット, 1566, 2993
ib_logfile, 2994
ibbackup_logfile, 2993
ibdata ファイル, 2994
ibtmp ファイル, 2994
icc
MySQL ビルド, 56
および MySQL Cluster のサポート, 2822
Id, 2076, 2081, 2128
ID

一意, 2755
identity システム変数, 487
id オプション
ndb_config, 2243
IF, 1390
IF(), 1082
IFNULL(), 1082
ignore-builtin-innodb オプション
mysqld, 1610
ignore-db-dir オプション
mysqld, 433
ignore-lines オプション
mysqlimport, 319
ignore-spaces オプション
mysql, 271
ignore-table オプション
mysqldump, 312
IGNORE
パーティション化されたテーブル, 1308
IGNORE INDEX, 1340
IGNORE KEY, 1340
IGNORE_AIO_CHECK オプション
CMake, 157
ignore_builtin_innodb システム変数, 1613
ignore_db_dirs システム変数, 487
IGNORE_SPACE SQL モード, 588
ignore オプション
mysqlimport, 319
ilist, 2994
IMPORT TABLESPACE, 1233, 1541
IN, 1078, 1346
in_file オプション
comp_err, 254
include-databases オプション (ndb_restore), 2269
include-gtids オプション
mysqlbinlog, 367
include-master-host-port オプション
mysqldump, 309
include-tables オプション (ndb_restore), 2269
include オプション
mysql_config, 388
INDEX DIRECTORY
およびレプリケーション, 1987
index_merge 結合型
オプティマイザ, 850
index_subquery 結合型
オプティマイザ, 851
IndexMemory, 2085
IndexStatAutoCreate
データノード, 2126
IndexStatAutoUpdate
データノード, 2126
IndexStatSaveScale
データノード, 2127
IndexStatSaveSize
データノード, 2126
IndexStatTriggerPct
データノード, 2127
IndexStatTriggerScale
データノード, 2127
IndexStatUpdateDelay
データノード, 2127
index 結合型
オプティマイザ, 851
INET6_ATON(), 1202
INET6_NTOA(), 1203
INET_ATON(), 1202
INET_NTOA(), 1202
INFORMATION_SCHEMA.ENGINES テーブル
および MySQL Cluster, 2318
INFORMATION_SCHEMA.GLOBAL_STATUS テーブル
および MySQL Cluster, 2319
INFORMATION_SCHEMA.GLOBAL_VARIABLES テーブル
と MySQL Cluster, 2318
INFORMATION_SCHEMA, 2500
INNODB_CMP_RESET テーブル, 1685
INNODB_CMPMEM_RESET テーブル, 1685
INNODB_CMPMEM テーブル, 1685
INNODB_CMP テーブル, 1685
INNODB_LOCK_WAITS テーブル, 1686
INNODB_LOCKS テーブル, 1686
INNODB_METRICS テーブル, 2545
INNODB_TRX テーブル, 1686
およびセキュリティー上の問題, 2346
照合順序と検索, 964
INFORMATION_SCHEMA, 2994
INFORMATION_SCHEMA プラグイン, 2772
information オプション
myisamchk, 340
INFO イベント (MySQL Cluster), 2305
init-command オプション
mysql, 271
init-file オプション
mysqld, 433
Init DB
スレッドのコマンド, 904
init
スレッドの状態, 907
init_connect システム変数, 488
init_file システム変数, 488
init_slave システム変数, 1916
InitFragmentLogFiles, 2092
initial-start オプション (ndbd), 2225
initial-start オプション (ndbmtd), 2225
Initialized
スレッドの状態, 915
InitialLogFileGroup, 2123
InitialNoOfOpenFiles, 2093
InitialTablespace, 2124
initial オプション (ndb_mgmd), 2234
initial オプション (ndbd), 2224
initial オプション (ndbmtd), 2224
INNER JOIN, 1335
innochecksum, 228, 332
InnoDB, 2994
innodb-status-file オプション
mysqld, 1611
InnoDB, 1495
file-per-table 設定, 1528
FULLTEXT インデックス, 1515
MySQL Cluster との比較, 2020, 2020, 2022, 2022
NFS, 1523, 1552
RAW パーティション, 1538
Solaris 10 x86_64 の問題, 139
一貫性読み取り, 1504
インデックス, 1514
インデックスレコードロック, 1502, 1507, 1509, 1640
およびアプリケーション機能要件, 2022

オンライン DDL, 1575
外部キー制約, 1551
可用性, 2020
ギャップロック, 1502, 1507, 1509, 1640
クラスタ化されたインデックス, 1514
クラッシュリカバリ, 1722
構成データファイルとメモリー割り当て, 1522
構成パラメータ, 1605
サポートされるアプリケーション, 2022
システムテーブルスペースの設定, 1527
システム変数, 1605
自動インクリメントカラム, 1546
自動コミットモード, 1512, 1541
ストレージ要件, 1055
制限と制約, 1552
セカンダリインデックス, 1514
挿入バッファー, 1519
チェックポイント, 1574
ディスク I/O, 1573
データファイル, 1536
テーブル, 1514, 1539
その他のストレージエンジンからの変換, 1542
テーブルの移行, 1540
適応型ハッシュインデックス, 1520
デッドロックの検出, 1512
デフォルトのストレージエンジンとしての考慮事項, 1496
トラブルシューティング, 1751
I/O に関する問題, 1751
InnoDB のエラーコード, 1755
OS のエラーコード, 1756
SQL のエラー, 1755
オンライン DDL, 1605
データディクショナリに関する問題, 1753
テーブルのデフラグ, 1575
デッドロック, 1512
パフォーマンスの問題, 836
リカバリに関する問題, 1752
トランザクションモデル, 1502
ネクストキーロック, 1502, 1507, 1509, 1640
バックアップ, 1720
半一貫性読み取り, 1640
ファイル領域管理, 1573
ページサイズ, 1519, 1553
マルチバージョン, 1513
モニター, 1574, 1709, 1722, 1751, 1754
行構造, 1520
レコードレベルロック, 1502, 1507, 1509, 1640
レプリケーション, 1723
ログファイル, 1536
ロック, 1502
ロックモード, 1502
ロック読み取り, 1506
innodb_adaptive_flushing, 1664
innodb_adaptive_flushing_lwm システム変数, 1613
innodb_adaptive_flushing システム変数, 1613
innodb_adaptive_hash_index
と innodb_thread_concurrency, 1671
innodb_adaptive_hash_index システム変数, 1614
innodb_adaptive_max_sleep_delay システム変数, 1614
innodb_additional_mem_pool_size システム変数, 1614
と innodb_use_sys_malloc, 1669
innodb_api_bk_commit_interval システム変数, 1615
innodb_api_disable_rowlock システム変数, 1615
innodb_api_enable_binlog システム変数, 1616
innodb_api_enable_mdl システム変数, 1616
innodb_api_trx_level システム変数, 1616
innodb_autoextend_increment システム変数, 1616
innodb_autoinc_lock_mode, 2994
innodb_autoinc_lock_mode システム変数, 1617
INNODB_BUFFER_PAGE_LRU テーブル, 2541
INNODB_BUFFER_PAGE テーブル, 2540
innodb_buffer_pool_dump_at_shutdown システム変数, 1617
innodb_buffer_pool_dump_now システム変数, 1617
innodb_buffer_pool_filename システム変数, 1618
innodb_buffer_pool_instances システム変数, 1618
innodb_buffer_pool_load_abort システム変数, 1619
innodb_buffer_pool_load_at_startup システム変数, 1619
innodb_buffer_pool_load_now システム変数, 1619
innodb_buffer_pool_size システム変数, 1620
INNODB_BUFFER_POOL_STATS テーブル, 2543
innodb_change_buffer_max_size システム変数, 1620
innodb_change_buffering, 1670
innodb_change_buffering システム変数, 1621
innodb_checksum_algorithm システム変数, 1621
innodb_checksums システム変数, 1623
innodb_cmp_per_index_enabled システム変数, 1623
INNODB_CMP_PER_INDEX_RESET テーブル, 2525
INNODB_CMP_PER_INDEX テーブル, 2525
INNODB_CMP_RESET テーブル, 2524
INNODB_CMPMEM_RESET テーブル, 2526
INNODB_CMPMEM テーブル, 2526
INNODB_CMP テーブル, 2524
innodb_commit_concurrency システム変数, 1623
innodb_compression_failure_threshold_pct システム変数, 1624
innodb_compression_level システム変数, 1624
innodb_compression_pad_pct_max システム変数, 1624
innodb_concurrency_tickets, 1671
innodb_concurrency_tickets システム変数, 1625
innodb_data_file_path システム変数, 1625
innodb_data_home_dir システム変数, 1626
innodb_disable_sort_file_cache システム変数, 1626
innodb_doublewrite システム変数, 1626
innodb_fast_shutdown システム変数, 1626
innodb_file_format, 1566, 2995
Antelope, 1562
Barracuda, 1555
識別, 1570
innodb_file_format_check, 1568
innodb_file_format_check システム変数, 1627
innodb_file_format_max システム変数, 1627
innodb_file_format システム変数, 1627
innodb_file_per_table, 1555, 2995
innodb_file_per_table システム変数, 1628
innodb_flush_log_at_timeout システム変数, 1628
innodb_flush_log_at_trx_commit システム変数, 1629
innodb_flush_method システム変数, 1630
innodb_flush_neighbors システム変数, 1631
innodb_flushing_avg_loops システム変数, 1631
innodb_force_load_corrupted システム変数, 1632
innodb_force_recovery システム変数, 1632
innodb_ft_aux_table システム変数, 1632
INNODB_FT_BEING_DELETED テーブル, 2551
innodb_ft_cache_size システム変数, 1633
INNODB_FT_CONFIG テーブル, 2546
INNODB_FT_DEFAULT_STOPWORD テーブル, 2547
INNODB_FT_DELETED テーブル, 2550
innodb_ft_enable_diag_print システム変数, 1633

innodb_ft_enable_stopword システム変数, 1634
INNODB_FT_INDEX_CACHE テーブル, 2549
INNODB_FT_INDEX_TABLE テーブル, 2548
innodb_ft_max_token_size システム変数, 1634
innodb_ft_min_token_size システム変数, 1634
innodb_ft_num_word_optimize システム変数, 1635
innodb_ft_result_cache_limit システム変数, 1635
innodb_ft_server_stopword_table システム変数, 1635
innodb_ft_sort_pll_degree システム変数, 1636
innodb_ft_total_cache_size システム変数, 1636
innodb_ft_user_stopword_table システム変数, 1636
innodb_index_stats テーブル, 1676
innodb_io_capacity, 1672
innodb_io_capacity_max システム変数, 1638
innodb_io_capacity システム変数, 1637
innodb_large_prefix システム変数, 1638
innodb_lock_wait_timeout, 2995
innodb_lock_wait_timeout システム変数, 1639
INNODB_LOCK_WAITS テーブル, 2530
innodb_locks_unsafe_for_binlog システム変数, 1639
INNODB_LOCKS テーブル, 2529
innodb_log_buffer_size システム変数, 1641
innodb_log_compressed_pages システム変数, 1642
innodb_log_file_size システム変数, 1642
innodb_log_files_in_group システム変数, 1643
innodb_log_group_home_dir システム変数, 1643
innodb_lru_scan_depth システム変数, 1643
innodb_max_dirty_pages_pct, 1664
innodb_max_dirty_pages_pct_lwm システム変数, 1644
innodb_max_dirty_pages_pct システム変数, 1644
innodb_max_purge_lag_delay システム変数, 1645
innodb_max_purge_lag システム変数, 1644
innodb_memcached_config.sql スクリプト, 1729
innodb_memcache データベース, 1729, 1745
INNODB_METRICS テーブル, 2545
innodb_mirrored_log_groups システム変数, 1645
innodb_monitor_disable システム変数, 1645
innodb_monitor_enable システム変数, 1646
innodb_monitor_reset_all システム変数, 1646
innodb_monitor_reset システム変数, 1646
innodb_old_blocks_pct, 1664
innodb_old_blocks_pct システム変数, 1646
innodb_old_blocks_time, 1664
innodb_old_blocks_time システム変数, 1647
innodb_online_alter_log_max_size システム変数, 1647
innodb_open_files システム変数, 1647
innodb_optimize_fulltext_only システム変数, 1648
INNODB_PAGE_ATOMIC_REF_COUNT オプション
  CMake, 157
innodb_page_size システム変数, 1648
innodb_print_all_deadlocks システム変数, 1649
  innodb_print_all_deadlocks, 1649
innodb_purge_batch_size システム変数, 1649
innodb_purge_threads システム変数, 1649
innodb_random_read_ahead システム変数, 1650
innodb_read_ahead_threshold, 1663
innodb_read_ahead_threshold システム変数, 1650
innodb_read_io_threads, 1672
innodb_read_io_threads システム変数, 1651
innodb_read_only システム変数, 1651
innodb_replication_delay システム変数, 1652
innodb_rollback_on_timeout システム変数, 1652
innodb_rollback_segments システム変数, 1652
innodb_sort_buffer_size システム変数, 1652
innodb_spin_wait_delay, 1673
innodb_spin_wait_delay システム変数, 1653
innodb_stat_persistent システム変数, 1655
innodb_stats_auto_recalc システム変数, 1653
innodb_stats_method システム変数, 1654
innodb_stats_on_metadata システム変数, 1654
innodb_stats_persistent_sample_pages システム変数, 1655
innodb_stats_persistent システム変数
  innodb_stats_persistent, 1655
innodb_stats_sample_pages システム変数, 1656
innodb_stats_transient_sample_pages, 1681
innodb_stats_transient_sample_pages システム変数, 1656
innodb_status_output_locks システム変数, 1657
innodb_status_output システム変数, 1656
innodb_strict_mode, 2995
innodb_strict_mode システム変数, 1657
innodb_support_xa システム変数, 1657
innodb_sync_array_size システム変数, 1658
innodb_sync_spin_loops システム変数, 1658
INNODB_SYS_COLUMNS テーブル, 2533
INNODB_SYS_DATAFILES テーブル, 2537
INNODB_SYS_FIELDS テーブル, 2534
INNODB_SYS_FOREIGN_COLS テーブル, 2535
INNODB_SYS_FOREIGN テーブル, 2535
INNODB_SYS_INDEXES テーブル, 2532
INNODB_SYS_TABLESPACES テーブル, 2537
INNODB_SYS_TABLESTATS テーブル, 2536
INNODB_SYS_TABLES テーブル, 2531
innodb_table_locks システム変数, 1658
innodb_table_stats テーブル, 1676
innodb_thread_concurrency, 1671
innodb_thread_concurrency システム変数, 1659
innodb_thread_sleep_delay, 1671
innodb_thread_sleep_delay システム変数, 1660
INNODB_TRX テーブル, 2527
innodb_undo_directory システム変数, 1660
innodb_undo_logs システム変数, 1661
innodb_undo_tablespaces システム変数, 1661
innodb_use_native_aio システム変数, 1662
innodb_use_sys_malloc
  と innodb_thread_concurrency, 1671
innodb_use_sys_malloc システム変数, 1662, 1669
innodb_version システム変数, 1662
innodb_write_io_threads, 1672
innodb_write_io_threads システム変数, 1662
innodb オプション
  mysqld, 1610
InnoDB ストレージエンジン, 1495, 1759
InnoDB テーブル, 29
InnoDB バッファープール, 861
InnoDB パラメータ、新しいデフォルト
  innodb_max_dirty_pages_pct, 1664
InnoDB パラメータ、新規
  innodb_adaptive_flushing, 1664
  innodb_change_buffering, 1670
  innodb_file_format_check, 1568
  innodb_io_capacity, 1672
  innodb_large_prefix, 1638
  innodb_read_ahead_threshold, 1663
  innodb_read_io_threads, 1672
  innodb_spin_wait_delay, 1673
  innodb_stats_transient_sample_pages, 1681
  innodb_use_sys_malloc, 1669
  innodb_write_io_threads, 1672

insert-ignore オプション
  mysqldump, 313
INSERT ... SELECT, 1309
INSERT DELAYED, 1310, 1310
INSERT, 816, 1305
insert
  スレッドの状態, 911
INSERT(), 1088
insert_id システム変数, 489
INSERT ステートメント
  権限付与, 683
install-manual オプション
  mysqld, 434
INSTALL PLUGIN, 1439
INSTALL_BINDIR オプション
  CMake, 154
INSTALL_DOCDIR オプション
  CMake, 154
INSTALL_DOCREADMEDIR オプション
  CMake, 154
INSTALL_INCLUDEDIR オプション
  CMake, 154
INSTALL_INFODIR オプション
  CMake, 154
INSTALL_LAYOUT オプション
  CMake, 154
INSTALL_LIBDIR オプション
  CMake, 155
INSTALL_MANDIR オプション
  CMake, 155
INSTALL_MYSQLSHAREDIR オプション
  CMake, 155
INSTALL_MYSQLTESTDIR オプション
  CMake, 155
INSTALL_PLUGINDIR オプション
  CMake, 155
INSTALL_SBINDIR オプション
  CMake, 155
INSTALL_SCRIPTDIR オプション
  CMake, 155
INSTALL_SHAREDIR オプション
  CMake, 155
INSTALL_SQLBENCHDIR オプション
  CMake, 155
INSTALL_SUPPORTFILESDIR オプション
  CMake, 155
install オプション
  mysqld, 434
  MySQLInstallerConsole, 87
install オプション (ndb_mgmd), 2237
install オプション (ndbd), 2226
install オプション (ndbmtd), 2226
INSTR(), 1088
INTEGER データ型, 1013
interactive_timeout システム変数, 489
interactive オプション (ndb_mgmd), 2233
InteriorRingN(), 1187
Intersects(), 1192
INTERVAL(), 1078
INTO
  SELECT, 1333
INT データ型, 1013
invalidating query cache entries
  スレッドの状態, 912
IOPS, 2995
IPv6 アドレス
  アカウント名における, 670
  デフォルトアカウントの, 173
IPv6 接続, 174
IP アドレス
  アカウント名における, 670
  デフォルトアカウントの, 173
IRC, 20
IS boolean_value, 1076
IS NOT boolean_value, 1076
IS NOT NULL, 1077
IS NULL, 787, 1076
  およびインデックス, 830
IS_FREE_LOCK(), 1203
IS_IPV4(), 1203
IS_IPV4_COMPAT(), 1204
IS_IPV4_MAPPED(), 1204
IS_IPV6(), 1204
IS_USED_LOCK(), 1205
IsClosed(), 1186, 1187
IsEmpty(), 1184
ISNULL(), 1078
ISOLATION LEVEL, 1365
IsSimple(), 1184
ITERATE, 1391
iterations オプション
  mysqlslap, 330

J

Java, 2656
jdbc:mysql:loadbalance://, 2013
JDBC, 2653
JOIN, 1335
join_buffer_size システム変数, 489
join オプション
  myisampack, 350

K

keep-my-cnf オプション
  mysql_install_db, 257
keep_files_on_create システム変数, 490
keepold オプション
  mysqlhotcopy, 382
key-file オプション
  ndb_setup.py, 2277
KEY_BLOCK_SIZE, 1555, 1559
KEY_BLOCK_SIZE, 2995
key_buffer_size myisamchk 変数, 338
key_buffer_size システム変数, 490
key_cache_age_threshold システム変数, 491
key_cache_block_size システム変数, 491
key_cache_division_limit システム変数, 492
KEY_COLUMN_USAGE
  INFORMATION_SCHEMA テーブル, 2508
keys-used オプション
  myisamchk, 341
keys オプション
  mysqlshow, 324
KEY パーティショニング, 2431
KEY パーティション
  管理, 2444
  分割およびマージ, 2444
Kill

スレッドのコマンド, 904
KILL, 1485
Killed
スレッドの状態, 907
Killing slave
スレッドの状態, 914

L

language オプション
mysqld, 434
large-pages オプション
mysqld, 434
large_files_support システム変数, 492
large_page_size システム変数, 492
large_pages システム変数, 492
LAST_DAY(), 1122
LAST_INSERT_ID(), 1174, 1308
およびストアドルーチン, 2476
およびトリガー, 2476
およびレプリケーション, 1980
ロックに関する考慮事項, 30
last_insert_id システム変数, 493
LateAlloc, 2097
lc-messages-dir オプション
mysqld, 435
lc-messages オプション
mysqld, 435
lc_messages_dir システム変数, 493
lc_messages システム変数, 493
lc_time_names システム変数, 493
LCASE(), 1088
LcpScanProgressTimeout, 2094
LD_LIBRARY_PATH 環境変数, 194
LD_RUN_PATH 環境変数, 192, 194
ldata オプション
mysql_install_db, 257
LDML 構文, 999
LEAST(), 1078
LEAVE, 1391
ledir オプション
mysqld_safe, 247
LEFT JOIN, 788, 1335
LEFT(), 1088
LEFT OUTER JOIN, 1335
LENGTH(), 1088
length オプション
myisam_ftdump, 334
libaio, 57, 134, 157
LIBMYSQL_ENABLE_CLEARTEXT_PLUGIN 環境変数, 192
LIBMYSQL_PLUGIN_DIR 環境変数, 192, 2753
LIBMYSQL_PLUGINS 環境変数, 192, 2753
libmysqlclient ライブラリ, 2653
libmysqld-libs オプション
mysql_config, 388
libmysqld, 2657
オプション, 2658
libmysqld ライブラリ, 2653
libs_r オプション
mysql_config, 388
libs オプション
mysql_config, 388
license システム変数, 493
LIKE, 1096
およびインデックス, 829
およびワイルドカード, 829
LIMIT, 813, 1173, 1331
およびレプリケーション, 1992
line-numbers オプション
mysql, 271
LINEAR HASH パーティショニング, 2430
LINEAR KEY パーティショニング, 2432
LineFromText(), 1182
LineFromWKB(), 1182
lines-terminated-by オプション
mysqldump, 310, 319
LineString(), 1183
LineStringFromText(), 1182
LineStringFromWKB(), 1182
LINESTRING データ型, 1043
list オプション
MySQLInstallerConsole, 87
LIST パーティショニング, 2420, 2422
LIST パーティション
管理, 2439
追加および削除, 2439
LN(), 1109
LOAD DATA INFILE, 1313, 2963
LOAD DATA
およびレプリケーション, 1992
LOAD INDEX INTO CACHE
およびパーティショニング, 2463
LOAD XML, 1321
LOAD_FILE(), 1088
local-infile オプション
mysql, 272
local-load オプション
mysqlbinlog, 367
local-service オプション
mysqld, 435
local_infile システム変数, 493
localhost, 231
LOCALTIME, 1122
LOCALTIMESTAMP, 1122
local オプション
mysqlimport, 319
LOCATE(), 1089
lock-all-tables オプション
mysqldump, 314
lock-tables オプション
mysqldump, 314
mysqlimport, 319
LOCK IN SHARE MODE, 1332
lock mode, 2995
LOCK TABLES, 1360
lock_wait_timeout システム変数, 494
locked_in_memory システム変数, 494
LockExecuteThreadToCPU, 2113
LockMaintThreadsToCPU, 2113
LockPagesInMainMemory, 2097
lock オプション
ndb_select_all, 2273
log-bin-index オプション
mysqld, 1929
log-bin-trust-function-creators オプション
mysqld, 1929
log-bin-use-v1-row-events オプション
mysqld, 1929
log-bin オプション

mysqld, 1928
log-error オプション
mysqld, 436
mysqld_safe, 247
mysqldump, 307
log-isam オプション
mysqld, 436
log-name オプション (ndb_mgmd), 2234
log-output オプション
mysqld, 436
log-queries-not-using-indexes オプション
mysqld, 436
log-raw オプション
mysqld, 437
log-short-format オプション
mysqld, 437, 1930
log-slave-updates オプション
mysqld, 1899
log-slow-admin-statements オプション
mysqld, 437
log-slow-queries オプション
mysqld, 437
log-slow-slave-statements オプション
mysqld, 1900
log-tc-size オプション
mysqld, 438
log-tc オプション
mysqld, 438
log-warnings オプション
mysqld, 438, 1900
LOG10(), 1110
LOG2(), 1109
LOG(), 1109
log_bin_basename システム変数, 1940
log_bin_index システム変数, 1940
log_bin_trust_function_creators システム変数, 494
log_bin_use_v1_row_events システム変数, 1940
log_bin システム変数, 1940
log_error システム変数, 495
log_output システム変数, 495
log_queries_not_using_indexes システム変数, 495
log_slave_updates システム変数, 1941
log_slow_admin_statements システム変数
mysqld, 496
log_slow_queries システム変数, 496
log_slow_slave_statements システム変数
mysqld, 1916
log_throttle_queries_not_using_indexes システム変数, 496
log_warnings システム変数, 496
logbuffers
ndbinfo テーブル, 2331
LogDestination, 2078
logging slow query
スレッドの状態, 907
login-path option, 389
login-path オプション, 240
mysql, 272
mysql_config_editor, 359
mysql_upgrade, 264
mysqladmin, 291
mysqlbinlog, 368
mysqlcheck, 297
mysqldump, 304
mysqlimport, 319
mysqlshow, 324
mysqlslap, 330
login
スレッドの状態, 907
LogLevelCheckpoint, 2110
LogLevelCongestion, 2110
LogLevelConnection, 2110
LogLevelError, 2110
LogLevelInfo, 2110
LogLevelNodeRestart, 2110
LogLevelShutdown, 2109
LogLevelStartup, 2109
LogLevelStatistic, 2109
logspaces
ndbinfo テーブル, 2331
log オプション
mysqld, 435
mysqld_multi, 251
log システム変数, 494
Long Data
スレッドのコマンド, 904
long_query_time システム変数, 497
LONGBLOB データ型, 1018
LongMessageBuffer, 2090
LONGTEXT データ型, 1018
LONG データ型, 1036
LOOP, 1392
ラベル, 1387
loops オプション
ndb_show_tables, 2279
loops オプション (ndb_index_stat), 2258
loops オプション (ndbinfo_select_all), 2229
loose_, 2996
lossy-conversions オプション (ndb_restore), 2265
lost+found ディレクトリ, 433
low-priority-updates オプション
mysqld, 439
low-priority オプション
mysqlimport, 319
low_priority_updates システム変数, 497
LOWER(), 1089
lower_case_file_system システム変数, 497
lower_case_table_names システム変数, 498
LPAD(), 1089
LRU, 2996
LRU ページの置き換え, 1664
LSN, 2996
<=> (と等しい), 1075
<= (より少ないか等しい), 1076
<> (等しくない), 1076
<< (左シフト), 221, 1162
LTRIM(), 1089
< (より少ない), 1076

M

MAKE_SET(), 1089
MAKEDATE(), 1122
MAKETIME(), 1123
Making temporary file (append) before replaying LOAD DATA INFILE
スレッドの状態, 913
Making temporary file (create) before replaying LOAD DATA INFILE
スレッドの状態, 914

manage keys
スレッドの状態, 908
master-data オプション
mysqldump, 309
master-info-file オプション
mysqld, 1901
master-info-repository オプション
mysqld, 1915
master-retry-count オプション
mysqld, 1901
master-verify-checksum オプション
mysqld, 1932
Master has sent all binlog to slave; waiting for binlog to be updated
スレッドの状態, 912
master_info_repository システム変数, 1917
MASTER_POS_WAIT(), 1205, 1378
master_verify_checksum システム変数, 1941
MATCH ... AGAINST(), 1132
max-binlog-dump-events オプション
mysqld, 1933
max-record-length オプション
myisamchk, 341
max-relay-log-size オプション
mysqld, 1901
MAX(DISTINCT), 1209
MAX(), 1209
max_allowed_packet
およびレプリケーション, 1993
max_allowed_packet システム変数, 498
max_allowed_packet 変数, 276
max_binlog_cache_size システム変数, 1942
max_binlog_size システム変数, 1942
max_binlog_stmt_cache_size システム変数, 1942
max_connect_errors システム変数, 499
MAX_CONNECTIONS_PER_HOUR, 685
max_connections システム変数, 499
max_delayed_threads システム変数, 500
max_error_count システム変数, 500
max_heap_table_size システム変数, 500
max_insert_delayed_threads システム変数, 501
max_join_size システム変数, 501
max_join_size 変数, 276
max_length_for_sort_data システム変数, 501
max_prepared_stmt_count システム変数, 502
MAX_QUERIES_PER_HOUR, 685
max_relay_log_size システム変数, 502
MAX_ROWS
および DataMemory (MySQL Cluster), 2084
および MySQL Cluster, 2414
と MySQL Cluster, 1270
と NDB, 1270
max_seeks_for_key システム変数, 502
max_sort_length システム変数, 503
max_sp_recursion_depth システム変数, 503
max_tmp_tables システム変数, 503
MAX_UPDATES_PER_HOUR, 685
MAX_USER_CONNECTIONS, 685
max_user_connections システム変数, 503
max_write_lock_count システム変数, 504
MaxAllocate, 2091
MaxBufferedEpochBytes, 2104
MaxBufferedEpochs, 2103
MAXDB SQL モード, 592
MaxDiskWriteSpeed, 2106
MaxDiskWriteSpeedOtherNodeRestart, 2106
MaxDiskWriteSpeedOwnRestart, 2106
MaxDMLOperationsPerTransaction, 2089
MaxLCPStartDelay, 2093
MaxNoOfAttributes, 2094
MaxNoOfConcurrentIndexOperations, 2089
MaxNoOfConcurrentOperations, 2088
MaxNoOfConcurrentScans, 2090
MaxNoOfConcurrentSubOperations, 2096
MaxNoOfConcurrentTransactions, 2087
MaxNoOfExecutionThreads
ndbmtd, 2115
MaxNoOfFiredTriggers, 2089
MaxNoOfIndexes, 2096
MaxNoOfLocalOperations, 2088
MaxNoOfLocalScans, 2090
MaxNoOfOpenFiles, 2093
MaxNoOfOrderedIndexes, 2095
MaxNoOfSavedMessages, 2093
MaxNoOfSubscribers, 2096
MaxNoOfSubscriptions, 2096
MaxNoOfTables, 2094
MaxNoOfTriggers, 2095
MaxNoOfUniqueHashIndexes, 2095
MaxParallelScansPerFragment, 2091
MaxScanBatchSize, 2130
MaxStartFailRetries, 2126
MBR, 1191, 1192
MBRContains(), 1192
MBRDisjoint(), 1192
MBREqual(), 1192
MBRIntersects(), 1192
MBROverlaps(), 1193
MBRTouches(), 1193
MBRWithin(), 1193
MD5(), 1167
MDL, 2996
medium-check オプション
myisamchk, 340
mysqlcheck, 298
MEDIUMBLOB データ型, 1018
MEDIUMINT データ型, 1013
MEDIUMTEXT データ型, 1018
membership
ndbinfo テーブル, 2331
memcached, 2996
memcached, 1724
MEMCACHED_HOME オプション
CMake, 161
MEMCACHED_SASL_PWDB 環境変数, 1731
memcapable コマンド, 1726
memlock オプション
mysqld, 439
memory_per_fragment
ndbinfo テーブル, 2334
memoryusage
ndbinfo テーブル, 2333
MEMORY ストレージエンジン, 1759, 1771
およびレプリケーション, 1993
MemReportFrequency, 2111
MERGE ストレージエンジン, 1759, 1780
MERGE テーブル
定義済み, 1780

metadata_locks_cache_size システム変数, 504
metadata_locks_hash_instances システム変数, 505
method オプション
  mysqlhotcopy, 383
[mgm] (MySQL Cluster), 2153
mgmd (MySQL Cluster)
  定義, 2011
    (参照 管理ノード (MySQL Cluster))
MICROSECOND(), 1123
MID(), 1090
min-examined-row-limit オプション
  mysqld, 439
MIN(DISTINCT), 1209
MIN(), 1209
min_examined_row_limit システム変数, 505
MinDiskWriteSpeed, 2106
MinFreePct, 2084, 2087
MINUTE(), 1123
MLineFromText(), 1182
MLineFromWKB(), 1182
MOD(), 1110
modify オプション
  MySQLInstallerConsole, 88
MOD (モジュロ), 1110
MONTH(), 1123
MONTHNAME(), 1123
MPointFromText(), 1182
MPointFromWKB(), 1182
MPolyFromText(), 1182
MPolyFromWKB(), 1182
msql2mysql, 387
mSQL との互換性, 1099
MSSQL SQL モード, 592
Multi-Range Read
  最適化, 797
MultiLineString(), 1183
MultiLineStringFromText(), 1182
MultiLineStringFromWKB(), 1182
MULTILINESTRING データ型, 1043
MultiPoint(), 1183
MultiPointFromText(), 1182
MultiPointFromWKB(), 1182
MULTIPOINT データ型, 1043
MultiPolygon(), 1183
MultiPolygonFromText(), 1182
MultiPolygonFromWKB(), 1182
MULTIPOLYGON データ型, 1043
mutex_instances テーブル
  performance_schema, 2603
MVCC, 2997
MVCC (マルチバージョン並列処理制御), 1513
my-print-defaults オプション
  mysql_plugin, 259
my.cnf, 2997
  MySQL Cluster での, 2296
  および MySQL Cluster, 2059, 2069, 2070
my.cnf ファイル, 1979
my.ini, 2997
My
  由来, 7
my_bool C 型, 2668
my_bool 値
  出力, 2668
my_init(), 2750
my_print_defaults, 229, 389
  config-file オプション, 389
  debug オプション, 389
  defaults-extra-file オプション, 389
  defaults-file オプション, 389
  defaults-group-suffix オプション, 389
  extra-file オプション, 389
  help オプション, 389
  no-defaults オプション, 389
  verbose オプション, 390
  version オプション, 390
my_ulonglong C 型, 2668
my_ulonglong 値
  出力, 2668
mycnf オプション
  ndb_config, 2242
  ndb_mgmd, 2233
myisam-block-size オプション
  mysqld, 440
myisam-recover-options オプション
  mysqld, 440, 1766
MyISAM
  InnoDB へのテーブルの変換, 1542
  圧縮テーブル, 350, 1769
myisam_block_size myisamchk 変数, 338
myisam_data_pointer_size システム変数, 506
myisam_ftdump, 228, 333
  count オプション, 334
  dump オプション, 334
  help オプション, 334
  length オプション, 334
  stats オプション, 334
  verbose オプション, 334
myisam_max_sort_file_size システム変数, 506
myisam_mmap_size システム変数, 506
myisam_recover_options システム変数, 507
myisam_repair_threads システム変数, 507
myisam_sort_buffer_size myisamchk 変数, 338
myisam_sort_buffer_size システム変数, 507
myisam_stats_method システム変数, 507
myisam_use_mmap システム変数, 508
myisamchk, 228, 334
  analyze オプション, 342
  backup オプション, 340
  block-search オプション, 342
  character-sets-dir オプション, 340
  check-only-changed オプション, 340
  check オプション, 340
  correct-checksum オプション, 341
  data-file-length オプション, 341
  debug オプション, 337
  defaults-extra-file オプション, 338
  defaults-file オプション, 338
  defaults-group-suffix オプション, 338
  description オプション, 342
  extend-check オプション, 340, 341
  fast オプション, 340
  force オプション, 340, 341
  help オプション, 337
  HELP オプション, 337
  information オプション, 340
  keys-used オプション, 341
  max-record-length オプション, 341
  medium-check オプション, 340

no-defaults オプション, 338
no-symlinks オプション, 341
parallel-recover オプション, 341
print-defaults オプション, 338
quick オプション, 341
read-only オプション, 340
recover オプション, 341
safe-recover オプション, 341
set-auto-increment[オプション, 342
set-character-set オプション, 341
set-collation オプション, 342
silent オプション, 338
sort-index オプション, 342
sort-records オプション, 342
sort-recover オプション, 342
tmpdir オプション, 342
unpack オプション, 342
update-state オプション, 340
verbose オプション, 338
version オプション, 338
wait オプション, 338
オプション, 337
出力例, 343
myisamlog, 228, 349
myisampack, 228, 350, 1285, 1769
backup オプション, 350
character-sets-dir オプション, 350
debug オプション, 350
force オプション, 350
help オプション, 350
join オプション, 350
silent オプション, 351
test オプション, 351
tmpdir オプション, 351
verbose オプション, 351
version オプション, 351
wait オプション, 351
MyISAM キーキャッシュ, 864
MyISAM ストレージエンジン, 1759, 1763
mysql, 2997
mysql.event テーブル, 2487
mysql.ndb_binlog_index テーブル, 2382
(参照 MySQL Cluster レプリケーション)
mysql.server, 226, 250
basedir オプション, 250
datadir オプション, 250
pid-file オプション, 250
service-startup-timeout オプション, 250
use-mysqld_safe オプション, 250
user オプション, 250
mysql.slave_master_info テーブル, 1953
mysql.slave_relay_log_info テーブル, 1953
mysql.sock
保護, 2959
MYSQL323 SQL モード, 592
MYSQL40 SQL モード, 592
MySQL 5.6 で削除された機能, 8
MySQL 5.6 の新機能, 8
MySQL 5.6 の非推奨の機能, 8
MySQL APT リポジトリ, 129, 179
MySQL Cluster Auto-Installer, 2033
「Define Attributes」画面, 2043
「Define Cluster」画面, 2039
「Define Hosts」画面, 2040
「Define Processes」画面, 2041
「Deploy Cluster」画面, 2045
アーキテクチャー, 2034
および ndb_setup.py, 2038
および Python, 2034
および setup.bat (Windows), 2038
起動, 2038
サポートされる Web ブラウザ, 2033
サポートされるプラットフォーム, 2033
使用, 2038
セキュリティーの問題, 2034
セットアッププログラム, 2276
セットアッププログラム (Windows), 2276
ソフトウェア要件, 2033
プロセスの削除, 2043
プロセスの追加, 2042
ホストの追加と削除, 2041
要件, 2033
「ようこそ」画面, 2039
リモートホストでの認証, 2034
リモートホストとローカルホスト, 2034, 2034
レイアウト, 2035
MySQL Cluster Manager
と ndb_mgm, 2289
MySQL Cluster, 2008
API ノード, 2011, 2127
Auto-Installer による配備, 2033
BACKUP イベント, 2306
CHECKPOINT イベント, 2302
CLUSTERLOG STATISTICS コマンド, 2306
CLUSTERLOG コマンド, 2300
CONNECTION イベント, 2302
CREATE NODEGROUP コマンド, 2291
DROP NODEGROUP コマンド, 2291
ENTER SINGLE USER MODE コマンド, 2291
ERROR イベント, 2305
EXIT SINGLE USER MODE コマンド, 2291
EXIT コマンド, 2291
FAQ, 2850
GCP 停止エラー, 2124
HELP コマンド, 2289
HostName パラメータ
およびセキュリティー, 2341
icc を使用したコンパイル, 2822
INFO イベント, 2305
InnoDB との比較, 2020, 2020, 2022, 2022
IP アドレス設定, 2032
Java クライアント, 2013
MAX_ROWS, 1270
mgm, 2284
mgmd, 2284
mgmd プロセス, 2230
mgm 管理クライアント, 2306
mgm クライアント, 2289
mgm プロセス, 2237
mysqld オプションおよび変数, 2162
mysqld プロセス, 2178, 2296
ndb_mgm, 2061, 2237
ndb_mgmd プロセス, 2230
ndbd, 2222, 2284
ndbd プロセス, 2222, 2307
ndbinfo_select_all, 2228
ndbmtd, 2229
NODERESTART イベント, 2303

QUIT コマンド, 2291
REPORT コマンド, 2290
RESTART コマンド, 2290
RPM のインストール (Linux), 2049
SCHEMA イベント, 2305
SCI (スケーラブルコヒーラントインタフェース), 2138, 2221
SHOW コマンド, 2289
SHUTDOWN コマンド, 2291
SINGLEUSER イベント, 2305
SQL ノード, 2011, 2127, 2296
START BACKUP コマンド, 2390
STARTUP イベント, 2302
START コマンド, 2290
STATISTICS イベント, 2304
STATUS コマンド, 2290
STOP コマンド, 2290
アップグレードとダウングレード, 2065, 2297
イベントタイプ, 2300, 2302
イベントの重大度レベル, 2301
イベントロギングのしきい値, 2301
イベントログ, 2299, 2300
イベントログの形式, 2302
インストール, 2031
インストール (Linux), 2047
インストール (Windows), 2052
インターコネクト, 2220
エラーログ, 2227
大きなテーブルの作成, 1270
および DNS, 2032
および MySQL root ユーザー, 2347
および MySQL 権限, 2344
および MySQL ルートユーザー, 2345
およびアプリケーション機能要件, 2022
およびトランザクション, 2084
およびネットワーク, 2016
概念, 2011
概要, 2010
可用性, 2020
"簡易" 構成, 2067
管理, 2178, 2222, 2230, 2237, 2237, 2284, 2287, 2289, 2306
管理クライアント (ndb_mgm), 2237
管理コマンド, 2306
管理ノード, 2011, 2076, 2230
起動, 2067
起動フェーズ (サマリー), 2288
起動または再起動, 2288
共有メモリートランスポータ, 2136
クエリーの実行, 2062
クラスタログ, 2299, 2300
構成, 2031, 2066, 2067, 2075, 2076, 2080, 2127, 2236, 2295, 2296
構成の変更, 2297
構成パラメータ, 2141, 2142, 2153, 2155, 2157
構成ファイル, 2059, 2069
構成 (例), 2070
コマンド, 2178, 2222, 2230, 2237, 2289
再起動, 2064
サポートされるアプリケーション, 2022
実行時統計, 2306
実行スレッド, 2115
シャットダウン, 2064
情報源, 2010
シングルユーザーモード, 2291, 2317
スタンドアロン MySQL Server との比較, 2020, 2020, 2022, 2022
ステータス変数, 2209
ストレージ要件, 1055
スレッドの状態, 914
セキュリティー, 2341
  および HostName パラメータ, 2341
  およびネットワーク構成, 2342
  およびネットワークポート, 2344
  およびファイアウォール, 2342, 2344
  およびリモート管理, 2344
  ネットワーク, 2341
セキュリティー手順, 2346
接続文字列, 2074
ソースからのインストール (Linux), 2051
ソースからのインストール (Windows), 2055
通信プロトコルの危険性, 2341
ディスクデータテーブル (参照 MySQL Cluster ディスクデータ)
データノード, 2011, 2080, 2222, 2229
テーブルとデータの使用, 2062
と INFORMATION_SCHEMA, 2346
トランザクション処理, 2027
トランスポータ
  TCP/IP, 2135
  共有メモリー (SHM), 2136
  スケーラブルコヒーラントインタフェース (SCI), 2138
トレースファイル, 2227
ネットワーク, 2135, 2136, 2138
ネットワーク構成
  およびセキュリティー, 2342
ネットワークトランスポータ, 2220, 2221
ノード間の直接接続, 2135
ノード識別子, 2136, 2136, 2138, 2138
ノードとタイプ, 2011
ノードとノードグループ, 2014
ノードの起動, 2055, 2061
ノードの障害 (シングルユーザーモード), 2317
ノードホストの定義, 2075
ノードログ, 2299
パーティション, 2014
パーティション化のサポート, 2024
バイナリのインストール (Windows), 2052
バイナリリリースのインストール (Linux), 2047
バックアップ, 2261, 2292, 2292, 2293, 2295, 2296
バックアップのトラブルシューティング, 2296
バックアップのリストア, 2261
パフォーマンス, 2221
複数の管理サーバー, 2299
プロセス管理, 2222
ベンチマーク, 2221
メモリー使用状況とリカバリ, 2298
メモリーの使用量とリカバリ, 2025
モニタリング, 2365
モニタリング用の SQL ステートメント, 2317
要件, 2016
リセット, 2297
利用可能なプラットフォーム, 2008
レプリカ, 2014
レプリケーション, 2375
  (参照 MySQL Cluster レプリケーション)
レプリケーション用に準備, 2385
ローリング再起動 (複数の管理サーバー), 2299

ロギングコマンド, 2300
ログファイル, 2227, 2230
MySQL Cluster NDB 7.3, 2018
MySQL Cluster NDB 7.4, 2018
MySQL Cluster ディスクデータ, 2347
ストレージの要件, 2352
ディスクデータオブジェクトの削除, 2350
テーブルスペースの作成, 2349
テーブルの作成, 2348, 2350
ログファイルグループの作成, 2348
MySQL Cluster に関連する SQL ステートメント, 2317
MySQL Cluster のインストール, 2031
Linux, 2047
Linux RPM, 2049
Linux ソースリリース, 2051
Linux バイナリリリース, 2047
Windows, 2052
Windows ソース, 2055
Windows バイナリリリース, 2052
MySQL Cluster の開発, 2018
MySQL Cluster の管理, 2237, 2287
MySQL Cluster の構成, 2031, 2066, 2236, 2296
MySQL Cluster の構成 (概念), 2011
MySQL Cluster の新機能, 2018
MySQL Cluster の制限, 2023, 2023
エラーの処理と報告, 2028
および標準の MySQL の制限との違い, 2025
現在のバージョンで解決された前のバージョンの, 2030
構成によって課される, 2025
構文, 2023
ジオメトリデータ型, 2024
実装, 2029
ディスクデータストレージ, 2029
データベースオブジェクト, 2028
トランザクション, 2026
パーティション化, 2024
バイナリロギング, 2029
パフォーマンス, 2028
複数の MySQL サーバー, 2030
複数の管理サーバー, 2030
未サポート機能, 2028
メモリー使用量とトランザクション処理, 2027
レプリケーション, 2025
MySQL Cluster のセキュリティー保護, 2346
MySQL Cluster の操作手順, 2031
MySQL Cluster プログラム, 2222
MySQL Cluster プログラムの使用, 2222
MySQL Cluster プロセス, 2222
MySQL Cluster プロセスの管理, 2222
MySQL Cluster レプリケーション, 2375
reset-slave.pl バックアップ自動化スクリプト, 2392
および --initial オプション, 2380
および NDB API _slave ステータス変数, 2376
および一意キー, 2379
および主キー, 2379
および循環レプリケーション, 2378
および単一障害点, 2388
既知の問題, 2377
起動, 2386
ギャップイベント, 2377
競合解決, 2399
コンセプト, 2376, 2376
循環レプリケーション, 2395
準備, 2385
使用されたシステムテーブル, 2382
スレーブ側の NDB 以外のストレージエンジン, 2380
接続の喪失, 2377
バックアップ, 2390
バックアップからのリストア, 2390
フェイルオーバー, 2388, 2389
ポイントインタイムリカバリ, 2394
マスターとスレーブの同期, 2392
マルチマスターレプリケーション, 2395
要件, 2376
読み取り競合の検出と解決, 2408
MYSQL C 型, 2667
MySQL Dolphin の名前, 7
MySQL Enterprise Audit, 726, 2835
MySQL Enterprise Backup, 2997
MySQL Enterprise Backup, 2834
MySQL Enterprise Encryption, 2835
MySQL Enterprise Monitor, 2833
MySQL Enterprise Security, 700, 706, 2834
MySQL Enterprise Thread Pool, 2835
MySQL Installer, 62, 62
MySQL
アップグレード, 261
定義, 4
はじめに, 4
発音, 5
mysql, 227, 266
auto-rehash オプション, 269
auto-vertical-output オプション, 269
batch オプション, 269
binary-mode オプション, 269
bind-address オプション, 269
character-sets-dir オプション, 270
charset コマンド, 277
clear コマンド, 277
column-names オプション, 270
column-type-info オプション, 270
comments オプション, 270
compress オプション, 270
connect-expired-password オプション, 270
connect コマンド, 277
database オプション, 270
debug-check オプション, 270
debug-info オプション, 270
debug オプション, 270
default-auth オプション, 270
default-character-set オプション, 270
defaults-extra-file オプション, 270
defaults-file オプション, 271
defaults-group-suffix オプション, 271
delimiter オプション, 271
delimiter コマンド, 277
disable named commands オプション, 271
edit コマンド, 278
ego コマンド, 278
enable-cleartext-plugin オプション, 271
execute オプション, 271
exit コマンド, 278
force オプション, 271
go コマンド, 278
help オプション, 269
help コマンド, 277
histignore オプション, 271
host オプション, 271

html オプション, 271
i-am-a-dummy オプション, 274
ignore-spaces オプション, 271
init-command オプション, 271
line-numbers オプション, 271
local-infile オプション, 272
login-path オプション, 272
named-commands オプション, 272
no-auto-rehash オプション, 272
no-beep オプション, 272
no-defaults オプション, 272
nopager コマンド, 278
notee コマンド, 278
nowarning コマンド, 278
one-database オプション, 272
pager オプション, 273
pager コマンド, 278
password オプション, 273
pipe オプション, 273
plugin-dir オプション, 273
port オプション, 273
print-defaults オプション, 273
print コマンド, 279
prompt オプション, 273
prompt コマンド, 279
protocol オプション, 273
quick オプション, 273
quit コマンド, 279
raw オプション, 273
reconnect オプション, 274
rehash コマンド, 279
safe-updates オプション, 274
secure-auth オプション, 274
server-public-key-path オプション, 274
shared-memory-base-name オプション, 275
show-warnings オプション, 275
sigint-ignore オプション, 275
silent オプション, 275
skip-column-names オプション, 275
skip-line-numbers オプション, 275
socket オプション, 275
source コマンド, 279
SSL オプション, 275
status コマンド, 279
system コマンド, 279
table オプション, 275
tee オプション, 275
tee コマンド, 279
unbuffered オプション, 275
user オプション, 275
use コマンド, 279
verbose オプション, 275
version オプション, 275
vertical オプション, 275
wait オプション, 276
warnings コマンド, 279
xml オプション, 276
MySQL Notifier, 88
mysql prompt コマンド, 280
MySQL Server
mysqld, 245, 393
MySQL Server のコンパイル
問題, 162
MySQL SLES リポジトリ, 129
mysql source (テキストファイルから読み取るためのコマンド), 217, 283
MySQL Yum リポジトリ, 123, 127, 178
mysql \. (テキストファイルから読み取るためのコマンド), 217, 283
mysql_affected_rows(), 2676
mysql_autocommit(), 2676
MYSQL_BIND C 型, 2723
mysql_change_user(), 2677
mysql_character_set_name(), 2678
mysql_clear_password 認証プラグイン, 710
mysql_client_find_plugin(), 2752
mysql_client_register_plugin(), 2753
mysql_close(), 2678
mysql_cluster_backup_privileges, 2363
mysql_cluster_move_grant_tables, 2363
mysql_cluster_move_privileges, 2363
mysql_cluster_privileges_are_distributed, 2363
mysql_cluster_restore_local_privileges, 2364
mysql_cluster_restore_privileges, 2364
mysql_cluster_restore_privileges_from_local, 2364
mysql_commit(), 2678
mysql_config, 388
cflags オプション, 388
cxxflags オプション, 388
embedded オプション, 388
include オプション, 388
libmysqld-libs オプション, 388
libs_r オプション, 388
libs オプション, 388
plugindir オプション, 388
port オプション, 388
socket オプション, 388
version オプション, 388
mysql_config_editor, 228, 355
all オプション, 359
debug オプション, 359
help オプション, 359
host オプション, 359
login-path オプション, 359
password オプション, 359
port オプション, 359
socket オプション, 359
user オプション, 359
verbose オプション, 359
version オプション, 359
warn オプション, 360
mysql_connect(), 2678
mysql_convert_table_format, 228, 384
force オプション, 384
help オプション, 384
host オプション, 384
password オプション, 384
port オプション, 384
socket オプション, 384
type オプション, 384
user オプション, 384
verbose オプション, 384
version オプション, 384
mysql_create_db(), 2679
mysql_data_seek(), 2679
MYSQL_DATADIR オプション
CMake, 155
mysql_debug(), 2680

MYSQL_DEBUG 環境変数, 192, 229, 2828
mysql_drop_db(), 2680
mysql_dump_debug_info(), 2681
mysql_eof(), 2681
mysql_errno(), 2682
mysql_error(), 2683
mysql_escape_string(), 2683
mysql_fetch_field(), 2683
mysql_fetch_field_direct(), 2684
mysql_fetch_fields(), 2684
mysql_fetch_lengths(), 2685
mysql_fetch_row(), 2685
MYSQL_FIELD C 型, 2667
mysql_field_count(), 2686, 2699
MYSQL_FIELD_OFFSET C 型, 2668
mysql_field_seek(), 2687
mysql_field_tell(), 2687
mysql_find_rows, 228, 384
  help オプション, 385
  regexp オプション, 385
  rows オプション, 385
  skip-use-db オプション, 385
  start_row オプション, 385
mysql_fix_extensions, 228, 385
mysql_free_result(), 2687
mysql_get_character_set_info(), 2688
mysql_get_client_info(), 2688
mysql_get_client_version(), 2688
mysql_get_host_info(), 2689
mysql_get_proto_info(), 2689
mysql_get_server_info(), 2689
mysql_get_server_version(), 2690
mysql_get_ssl_cipher(), 2690
MYSQL_GROUP_SUFFIX 環境変数, 192
mysql_hex_string(), 2690
MYSQL_HISTFILE 環境変数, 192, 281
MYSQL_HISTIGNORE 環境変数, 192, 281
MYSQL_HOME 環境変数, 192
MYSQL_HOST 環境変数, 192, 233
mysql_info(), 1234, 1307, 1321, 1355, 2691
mysql_init(), 2691
mysql_insert_id(), 30, 1308, 2692
mysql_install_db, 168, 226, 255
  basedir オプション, 256
  builddir オプション, 256
  cross-bootstrap オプション, 256
  datadir オプション, 256
  force オプション, 257
  help オプション, 256
  keep-my-cnf オプション, 257
  ldata オプション, 257
  random-passwords オプション, 257
  rpm オプション, 258
  skip-name-resolve オプション, 258
  srcdir オプション, 258
  user オプション, 258
  verbose オプション, 258
  windows オプション, 258
mysql_kill(), 2693
mysql_library_end(), 2694
mysql_library_init(), 2694
mysql_list_dbs(), 2695
mysql_list_fields(), 2696
mysql_list_processes(), 2697
mysql_list_tables(), 2697
mysql_load_plugin(), 2753
mysql_load_plugin_v(), 2754
MYSQL_MAINTAINER_MODE オプション
  CMake, 158
mysql_more_results(), 2698
mysql_native_password 認証プラグイン, 693
mysql_next_result(), 2698
mysql_num_fields(), 2699
mysql_num_rows(), 2700
mysql_old_password 認証プラグイン, 694
mysql_options4(), 2705
mysql_options(), 2700
mysql_ping(), 2706
mysql_plugin, 227, 258
  basedir オプション, 259
  datadir オプション, 259
  help オプション, 259
  my-print-defaults オプション, 259
  mysqld オプション, 259
  no-defaults オプション, 259
  plugin-dir オプション, 260
  plugin-ini オプション, 260
  print-defaults オプション, 260
  verbose オプション, 260
  version オプション, 260
mysql_plugin_options(), 2754
MYSQL_PROJECT_NAME オプション
  CMake, 158
MYSQL_PS1 環境変数, 192
MYSQL_PWD 環境変数, 192, 229, 233
mysql_query(), 2707, 2754
mysql_real_connect(), 2707
mysql_real_escape_string(), 919, 1090, 2710
mysql_real_query(), 2711
mysql_refresh(), 2712
mysql_reload(), 2713
MYSQL_RES C 型, 2667
mysql_rollback(), 2713
MYSQL_ROW C 型, 2667
mysql_row_seek(), 2714
mysql_row_tell(), 2714
mysql_secure_installation, 227, 260
mysql_select_db(), 2714
MYSQL_SERVER_AUTH_INFO プラグイン構造体, 2803
mysql_server_end(), 2751
mysql_server_init(), 2751
mysql_set_character_set(), 2715
mysql_set_local_infile_default(), 2715, 2715
mysql_set_server_option(), 2717
mysql_setpermission, 229, 385
  help オプション, 386
  host オプション, 386
  password オプション, 386
  port オプション, 386
  socket オプション, 386
  user オプション, 386
mysql_shutdown(), 2717
mysql_sqlstate(), 2718
mysql_ssl_set(), 2718
mysql_stat(), 2719
MYSQL_STMT C 型, 2723
mysql_stmt_affected_rows(), 2730
mysql_stmt_attr_get(), 2730

mysql_stmt_attr_set(), 2730
mysql_stmt_bind_param(), 2731
mysql_stmt_bind_result(), 2732
mysql_stmt_close(), 2733
mysql_stmt_data_seek(), 2733
mysql_stmt_errno(), 2733
mysql_stmt_error(), 2734
mysql_stmt_execute(), 2734
mysql_stmt_fetch(), 2737
mysql_stmt_fetch_column(), 2741
mysql_stmt_field_count(), 2742
mysql_stmt_free_result(), 2742
mysql_stmt_init(), 2742
mysql_stmt_insert_id(), 2742
mysql_stmt_next_result(), 2743
mysql_stmt_num_rows(), 2744
mysql_stmt_param_count(), 2744
mysql_stmt_param_metadata(), 2744
mysql_stmt_prepare(), 2744
mysql_stmt_reset(), 2745
mysql_stmt_result_metadata, 2746
mysql_stmt_row_seek(), 2747
mysql_stmt_row_tell(), 2747
mysql_stmt_send_long_data(), 2747
mysql_stmt_sqlstate(), 2749
mysql_stmt_store_result(), 2749
mysql_store_result(), 2719, 2754
MYSQL_TCP_PORT オプション
  CMake, 158
MYSQL_TCP_PORT 環境変数, 192, 229, 625, 626
MYSQL_TEST_LOGIN_FILE 環境変数, 192
mysql_thread_end(), 2750
mysql_thread_id(), 2720
mysql_thread_init(), 2751
mysql_thread_safe(), 2751
MYSQL_TIME C 型, 2725
mysql_tzinfo_to_sql, 227, 260
MYSQL_UNIX_ADDR オプション
  CMake, 158
MYSQL_UNIX_PORT 環境変数, 169, 192, 229, 625, 626
mysql_upgrade, 227, 261, 677
  basedir オプション, 263
  character-sets-dir オプション, 263
  compress オプション, 264
  datadir オプション, 264
  debug-check オプション, 264
  debug-info オプション, 264
  debug オプション, 264
  default-auth オプション, 264
  default-character-set オプション, 264
  defaults-extra-file オプション, 264
  defaults-file オプション, 264
  defaults-group-suffix オプション, 264
  force オプション, 264
  help オプション, 263
  host オプション, 264
  login-path オプション, 264
  mysql_upgrade_info ファイル, 262
  no-defaults オプション, 265
  password オプション, 265
  pipe オプション, 265
  plugin-dir オプション, 265
  port オプション, 265
  print-defaults オプション, 265
  protocol オプション, 265
  shared-memory-base-name オプション, 265
  socket オプション, 265
  SSL オプション, 265
  tmpdir オプション, 266
  upgrade-system-tables オプション, 266
  user オプション, 266
  verbose オプション, 266
  version-check オプション, 266
  write-binlog オプション, 266
mysql_upgrade_info ファイル
  mysql_upgrade, 262
mysql_use_result(), 2721
mysql_waitpid, 229, 386
  help オプション, 386
  verbose オプション, 386
  version オプション, 386
mysql_warning_count(), 2722
mysql_zap, 229, 387
mysqlaccess, 228, 360
  brief オプション, 360
  commit オプション, 360
  copy オプション, 360
  db オプション, 361
  debug オプション, 361
  help オプション, 360
  host オプション, 361
  howto オプション, 361
  old_server オプション, 361
  password オプション, 361
  plan オプション, 361
  preview オプション, 361
  relnotes オプション, 361
  rhost オプション, 361
  rollback オプション, 361
  spassword オプション, 361
  superuser オプション, 361
  table オプション, 361
  user オプション, 361
  version オプション, 362
mysqladmin, 227, 286, 1246, 1293, 1473, 1477, 1481, 1485
  bind-address オプション, 290
  character-sets-dir オプション, 290
  compress オプション, 290
  count オプション, 290
  debug-check オプション, 290
  debug-info オプション, 290
  debug オプション, 290
  default-auth オプション, 290
  default-character-set オプション, 290
  defaults-extra-file オプション, 290
  defaults-file オプション, 291
  defaults-group-suffix オプション, 291
  enable-cleartext-plugin オプション, 291
  force オプション, 291
  help オプション, 290
  host オプション, 291
  login-path オプション, 291
  no-beep オプション, 291
  no-defaults オプション, 291
  password オプション, 291
  pipe オプション, 291
  plugin-dir オプション, 292
  port オプション, 292

print-defaults オプション, 292
protocol オプション, 292
relative オプション, 292
secure-auth オプション, 292
shared-memory-base-name オプション, 292
silent オプション, 292
sleep オプション, 292
socket オプション, 292
SSL オプション, 292
user オプション, 292
verbose オプション, 293
version オプション, 293
vertical オプション, 293
wait オプション, 293
mysqladmin オプション
mysqld_multi, 252
mysqladmin コマンドオプション, 289
mysqlbackup コマンド, 2997
mysqlbinlog, 228, 362
base64-output オプション, 364
bind-address オプション, 365
binlog-row-event-max-size オプション, 365
character-sets-dir オプション, 365
connection-server-id オプション, 365
database オプション, 366
debug-check オプション, 366
debug-info オプション, 367
debug オプション, 366
default-auth オプション, 367
defaults-extra-file オプション, 367
defaults-file オプション, 367
defaults-group-suffix オプション, 367
disable-log-bin オプション, 367
exclude-gtids オプション, 367
force-if-open オプション, 367
force-read オプション, 367
help オプション, 364
hexdump オプション, 367
host オプション, 367
include-gtids オプション, 367
local-load オプション, 367
login-path オプション, 368
no-defaults オプション, 368
offset オプション, 368
password オプション, 368
plugin-dir オプション, 368
port オプション, 368
print-defaults オプション, 368
protocol オプション, 368
raw オプション, 368
read-from-remote-master オプション, 369
read-from-remote-server オプション, 369
result-file オプション, 369
secure-auth オプション, 369
server-id-bits オプション, 369
server-id オプション, 369
set-charset オプション, 369
shared-memory-base-name オプション, 369
short-form オプション, 370
skip-gtids オプション, 370
socket オプション, 370
start-datetime オプション, 370
start-position オプション, 370
stop-datetime オプション, 370
stop-never-slave-server-id オプション, 370
stop-never オプション, 370
stop-position オプション, 371
to-last-log オプション, 371
user オプション, 371
verbose オプション, 371
verify-binlog-checksum オプション, 371
version オプション, 371
mysqlbug, 255
mysqlcheck, 227, 293
all-databases オプション, 296
all-in-1 オプション, 296
analyze オプション, 296
auto-repair オプション, 296
bind-address オプション, 296
character-sets-dir オプション, 296
check-only-changed オプション, 296
check-upgrade オプション, 296
check オプション, 296
compress オプション, 296
databases オプション, 296
debug-check オプション, 296
debug-info オプション, 297
debug オプション, 296
default-auth オプション, 297
default-character-set オプション, 297
defaults-extra-file オプション, 297
defaults-file オプション, 297
defaults-group-suffix オプション, 297
extended オプション, 297
fast オプション, 297
fix-db-names オプション, 297
fix-table-names オプション, 297
force オプション, 297
help オプション, 295
host オプション, 297
login-path オプション, 297
medium-check オプション, 298
no-defaults オプション, 298
optimize オプション, 298
password オプション, 298
pipe オプション, 298
plugin-dir オプション, 298
port オプション, 298
print-defaults オプション, 298
protocol オプション, 298
quick オプション, 298
repair オプション, 298
secure-auth オプション, 299
shared-memory-base-name オプション, 299
silent オプション, 299
skip-database オプション, 299
socket オプション, 299
SSL オプション, 299
tables オプション, 299
use-frm オプション, 299
user オプション, 299
verbose オプション, 299
version オプション, 299
write-binlog オプション, 299
mysqld, 2997
mysqld-version オプション
mysqld_safe, 248
mysqld (MySQL Cluster), 2222

[mysqld] (MySQL Cluster), 2155
mysqld, 226
abort-slave-event-count オプション, 1899
allow-suspicious-udfs オプション, 425
ansi オプション, 425
audit-log オプション, 740
basedir オプション, 425
big-tables オプション, 425
bind-address オプション, 426
binlog-checksum オプション, 1932
binlog-do-db オプション, 1930
binlog-format オプション, 426
binlog-ignore-db オプション, 1931
binlog-row-event-max-size オプション, 1928
binlog-rows-query-log-events オプション, 1933
bootstrap オプション, 427
character-set-client-handshake オプション, 427
character-set-filesystem オプション, 427
character-set-server オプション, 428
character-sets-dir オプション, 427
chroot オプション, 428
collation-server オプション, 428
console オプション, 428
core-file オプション, 428
datadir オプション, 429
debug-sync-timeout オプション, 429
debug オプション, 429
default-authentication-plugin オプション, 429
default-storage-engine オプション, 430
default-time-zone オプション, 430
defaults-extra-file オプション, 431
defaults-file オプション, 431
defaults-group-suffix オプション, 431
delay-key-write オプション, 431, 1766
des-key-file オプション, 431
disconnect-slave-event-count オプション, 1899
enable-named-pipe オプション, 431
event-scheduler オプション, 432
exit-info オプション, 432
external-locking オプション, 432
flush オプション, 432
gdb オプション, 433
general-log オプション, 433
help オプション, 425
ignore-builtin-innodb オプション, 1610
ignore-db-dir オプション, 433
init-file オプション, 433
innodb-status-file オプション, 1611
innodb オプション, 1610
install-manual オプション, 434
install オプション, 434
language オプション, 434
large-pages オプション, 434
lc-messages-dir オプション, 435
lc-messages オプション, 435
local-service オプション, 435
log-bin-index オプション, 1929
log-bin-trust-function-creators オプション, 1929
log-bin-use-v1-row-events オプション, 1929
log-bin オプション, 1928
log-error オプション, 436
log-isam オプション, 436
log-output オプション, 436
log-queries-not-using-indexes オプション, 436
log-raw オプション, 437
log-short-format オプション, 437, 1930
log-slave-updates オプション, 1899
log-slow-admin-statements オプション, 437
log-slow-queries オプション, 437
log-slow-slave-statements オプション, 1900
log-tc-size オプション, 438
log-tc オプション, 438
log-warnings オプション, 438, 1900
log_slow_admin_statements システム変数, 496
log_slow_slave_statements システム変数, 1916
log オプション, 435
low-priority-updates オプション, 439
master-info-file オプション, 1901
master-info-repository オプション, 1915
master-retry-count オプション, 1901
master-verify-checksum オプション, 1932
max-binlog-dump-events オプション, 1933
max-relay-log-size オプション, 1901
memlock オプション, 439
min-examined-row-limit オプション, 439
myisam-block-size オプション, 440
myisam-recover-options オプション, 440, 1766
MySQL Cluster での役割 (参照 SQL ノード (MySQL Cluster))
MySQL Cluster プロセスとしての, 2178, 2296
MySQL Server, 245, 393
ndb-batch-size オプション, 2178
ndb-blob-read-batch-bytes オプション, 2179
ndb-blob-write-batch-bytes オプション, 2179
ndb-cluster-connection-pool オプション, 2178
ndb-connectstring オプション, 2180
ndb-log-apply-status, 2182
ndb-log-empty-epochs, 2183
ndb-log-exclusive-reads, 2183
ndb-log-orig, 2184
ndb-log-transaction-id, 2184
ndb-nodeid, 2184
ndbcluster オプション, 2182
no-defaults オプション, 441
old-alter-table オプション, 441
old-style-user-limits オプション, 441
one-thread オプション, 441
open-files-limit オプション, 441
partition オプション, 442
performance-schema-consumer-events-stages-current オプション, 2639
performance-schema-consumer-events-stages-history-long オプション, 2639
performance-schema-consumer-events-stages-history オプション, 2639
performance-schema-consumer-events-statements-current オプション, 2639
performance-schema-consumer-events-statements-history-long オプション, 2639
performance-schema-consumer-events-statements-history オプション, 2639
performance-schema-consumer-events-waits-current オプション, 2639
performance-schema-consumer-events-waits-history-long オプション, 2639
performance-schema-consumer-events-waits-history オプション, 2639

performance-schema-consumer-global-instrumentation オプション, 2639
performance-schema-consumer-statements-digest オプション, 2639
performance-schema-consumer-thread-instrumentation オプション, 2639
performance-schema-consumer-xxx オプション, 2639
performance-schema-instrument オプション, 2639
pid-file オプション, 442
plugin-load-add オプション, 443
plugin-load オプション, 443
plugin オプションプリフィクス, 442
port-open-timeout オプション, 444
port オプション, 444
print-defaults オプション, 444
read-only オプション, 1901
relay-log-index オプション, 1902
relay-log-info-file オプション, 1903
relay-log-info-repository オプション, 1915
relay-log-purge オプション, 1903
relay-log-recovery オプション, 1903
relay-log-space-limit オプション, 1904
relay-log オプション, 1902
remove オプション, 444
replicate-do-db オプション, 1904
replicate-do-table オプション, 1906
replicate-ignore-db オプション, 1905
replicate-ignore-table オプション, 1906
replicate-rewrite-db オプション, 1907
replicate-same-server-id オプション, 1907
replicate-wild-do-table オプション, 1907
replicate-wild-ignore-table オプション, 1908
report-host オプション, 1908
report-password オプション, 1909
report-port オプション, 1909
report-user オプション, 1909
safe-mode オプション, 445
safe-user-create オプション, 445
secure-auth オプション, 445
secure-file-priv オプション, 445
server-id-bits オプション, 2188
server-id オプション, 1879
server_uuid 変数, 1880
shared-memory-base-name オプション, 446
shared-memory オプション, 446
show-slave-auth-info オプション, 1909
skip-concurrent-insert オプション, 446
skip-event-scheduler オプション, 446
skip-grant-tables オプション, 446
skip-host-cache オプション, 447
skip-innodb オプション, 447, 1611
skip-name-resolve オプション, 447
skip-ndbcluster オプション, 2188
skip-networking オプション, 447
skip-partition オプション, 447
skip-show-database オプション, 448
skip-slave-start オプション, 1912
skip-stack-trace オプション, 448
skip-symbolic-links オプション, 448
skip-thread-priority オプション, 448
slave-checkpoint-group オプション, 1910
slave-checkpoint-period オプション, 1910
slave-load-tmpdir オプション, 1912
slave-max-allowed-packet, 1912
slave-net-timeout オプション, 1913
slave-parallel-workers オプション, 1910
slave-pending-jobs-size-max オプション, 1911
slave-rows-search-algorithms, 1913
slave-skip-errors オプション, 1914
slave-sql-verify-checksum オプション, 1914
slave_compressed_protocol オプション, 1912
slow-query-log オプション, 448
slow-start-timeout オプション, 449
socket オプション, 449
sporadic-binlog-dump-fail オプション, 1933
sql-mode オプション, 449
SSL オプション, 447
standalone オプション, 447
super-large-pages オプション, 448
symbolic-links オプション, 448
sysdate-is-now オプション, 450
tc-heuristic-recover オプション, 450
temp-pool オプション, 451
tmpdir オプション, 451
transaction-isolation オプション, 451
transaction-read-only オプション, 451
user オプション, 452
validate-password オプション, 654
verbose オプション, 452
version オプション, 452
起動, 658
コマンドオプション, 424
mysqld_multi, 226, 250
defaults-extra-file オプション, 251
defaults-file オプション, 251
example オプション, 251
help オプション, 251
log オプション, 251
mysqladmin オプション, 252
mysqld オプション, 252
no-defaults オプション, 251
no-log オプション, 252
password オプション, 252
silent オプション, 252
tcp-ip オプション, 252
user オプション, 252
verbose オプション, 252
version オプション, 252
mysqld_safe, 226, 246
basedir オプション, 247
core-file-size オプション, 247
datadir オプション, 247
defaults-extra-file オプション, 247
defaults-file オプション, 247
help オプション, 247
ledir オプション, 247
log-error オプション, 247
malloc-lib オプション, 247
mysqld-version オプション, 248
mysqld オプション, 248
nice オプション, 248
no-defaults オプション, 248
open-files-limit オプション, 248
pid-file オプション, 248
plugin-dir オプション, 248
port オプション, 248
skip-kill-mysqld オプション, 248
skip-syslog オプション, 248

socket オプション, 248
syslog-tag オプション, 249
syslog オプション, 248
timezone オプション, 249
user オプション, 249
mysqldump, 2997
mysqldump, 191, 227, 300
add-drop-database オプション, 306
add-drop-table オプション, 306
add-drop-trigger オプション, 306
add-locks オプション, 313
all-databases オプション, 311
all-tablespaces オプション, 306
allow-keywords オプション, 307
apply-slave-statements オプション, 308
bind-address オプション, 304
character-sets-dir オプション, 308
comments オプション, 307
compact オプション, 309
compatible オプション, 309
complete-insert オプション, 310
compress オプション, 304
create-options オプション, 310
databases オプション, 312
debug-check オプション, 307
debug-info オプション, 307
debug オプション, 307
default-auth オプション, 304
default-character-set オプション, 308
defaults-extra-file オプション, 305
defaults-file オプション, 305
defaults-group-suffix オプション, 306
delayed-insert オプション, 313
delete-master-logs オプション, 308
disable-keys オプション, 313
dump-date オプション, 307
dump-slave オプション, 308
events オプション, 312
extended-insert オプション, 313
fields-enclosed-by オプション, 310, 319
fields-escaped-by オプション, 310, 319
fields-optionally-enclosed-by オプション, 310, 319
fields-terminated-by オプション, 310, 319
flush-logs オプション, 313
flush-privileges オプション, 314
force オプション, 307
help オプション, 307
hex-blob オプション, 310
host オプション, 304
ignore-table オプション, 312
include-master-host-port オプション, 309
insert-ignore オプション, 313
lines-terminated-by オプション, 310, 319
lock-all-tables オプション, 314
lock-tables オプション, 314
log-error オプション, 307
login-path オプション, 304
master-data オプション, 309
no-autocommit オプション, 314
no-create-db オプション, 306
no-create-info オプション, 306
no-data オプション, 312
no-defaults オプション, 306
no-set-names オプション, 308
no-tablespaces オプション, 306
opt オプション, 313
order-by-primary オプション, 314
password オプション, 304
pipe オプション, 304
plugin-dir オプション, 304
port オプション, 305
print-defaults オプション, 306
protocol オプション, 305
quick オプション, 313
quote-names オプション, 310
replace オプション, 306
result-file オプション, 310
routines オプション, 312
secure-auth オプション, 305
set-charset オプション, 308
set-gtid-purged オプション, 309
shared-memory-base-name オプション, 314
single-transaction オプション, 314
skip-comments オプション, 307
skip-opt オプション, 313
socket オプション, 305
SSL オプション, 305
tables オプション, 312
tab オプション, 310
triggers オプション, 312
tz-utc オプション, 310
user オプション, 305
verbose オプション, 307
version オプション, 308
where オプション, 312
xml オプション, 311
回避策, 2979
回避方法, 316
バックアップに使用, 759
ビュー, 316, 2979
問題, 316, 2979
mysqldumpslow, 228, 379
debug オプション, 380
help オプション, 380
verbose オプション, 381
mysqld オプション, 881
disable-gtid-unsafe-statements, 1943
enforce-gtid-consistency, 1944
gtid-mode, 1945
malloc-lib, 247
mysql_plugin, 259
mysqld_multi, 252
mysqld_safe, 248
mysqld オプションおよび変数
MySQL Cluster, 2162
mysqld サーバー
バッファーサイズ, 880
mysqld のテスト
mysqltest, 2768
mysqlhotcopy, 228, 381
addtodest オプション, 382
allowold オプション, 382
checkpoint オプション, 382
chroot オプション, 382
debug オプション, 382
dryrun オプション, 382
flushlog オプション, 382
help オプション, 382

host オプション, 382
keepold オプション, 382
method オプション, 383
noindices オプション, 383
old_server オプション, 383
password オプション, 383
port オプション, 383
quiet オプション, 383
record_log_pos オプション, 383
regexp オプション, 383
resetmaster オプション, 383
resetslave オプション, 383
socket オプション, 383
suffix オプション, 383
tmpdir オプション, 383
user オプション, 383
mysqlimport, 191, 227, 316, 1313
bind-address オプション, 318
character-sets-dir オプション, 318
columns オプション, 318
compress オプション, 318
debug-check オプション, 318
debug-info オプション, 318
debug オプション, 318
default-auth オプション, 318
default-character-set オプション, 318
defaults-extra-file オプション, 318
defaults-file オプション, 318
defaults-group-suffix オプション, 318
delete オプション, 319
force オプション, 319
help オプション, 318
host オプション, 319
ignore-lines オプション, 319
ignore オプション, 319
local オプション, 319
lock-tables オプション, 319
login-path オプション, 319
low-priority オプション, 319
no-defaults オプション, 319
password オプション, 320
pipe オプション, 320
plugin-dir オプション, 320
port オプション, 320
print-defaults オプション, 320
protocol オプション, 320
replace オプション, 320
secure-auth オプション, 320
shared-memory-base-name オプション, 320
silent オプション, 321
socket オプション, 321
SSL オプション, 321
use-threads オプション, 321
user オプション, 321
verbose オプション, 321
version オプション, 321
MySQLInstallerConsole, 86
configure オプション, 86
help オプション, 87
install オプション, 87
list オプション, 87
modify オプション, 88
remove オプション, 88
status オプション, 88
update オプション, 88
upgrade オプション, 88
mysqlshow, 227, 321
bind-address オプション, 323
character-sets-dir オプション, 323
compress オプション, 323
count オプション, 323
debug-check オプション, 323
debug-info オプション, 323
debug オプション, 323
default-auth オプション, 323
default-character-set オプション, 323
defaults-extra-file オプション, 323
defaults-file オプション, 324
defaults-group-suffix オプション, 324
help オプション, 323
host オプション, 324
keys オプション, 324
login-path オプション, 324
no-defaults オプション, 324
password オプション, 324
pipe オプション, 324
plugin-dir オプション, 324
port オプション, 324
print-defaults オプション, 325
protocol オプション, 325
secure-auth オプション, 325
shared-memory-base-name オプション, 325
show-table-type オプション, 325
socket オプション, 325
SSL オプション, 325
status オプション, 325
user オプション, 325
verbose オプション, 325
version オプション, 325
mysqlslap, 227, 325
auto-generate-sql-add-autoincrement オプション, 328
auto-generate-sql-execute-number オプション, 328
auto-generate-sql-guid-primary オプション, 328
auto-generate-sql-load-type オプション, 328
auto-generate-sql-secondary-indexes オプション, 328
auto-generate-sql-unique-query-number オプション, 328
auto-generate-sql-unique-write-number オプション, 329
auto-generate-sql-write-number オプション, 329
auto-generate-sql オプション, 328
commit オプション, 329
compress オプション, 329
concurrency オプション, 329
create-schema オプション, 329
create オプション, 329
csv オプション, 329
debug-check オプション, 329
debug-info オプション, 329
debug オプション, 329
default-auth オプション, 329
defaults-extra-file オプション, 329
defaults-file オプション, 329
defaults-group-suffix オプション, 330
delimiter オプション, 330
detach オプション, 330
enable-cleartext-plugin オプション, 330
engine オプション, 330
help オプション, 328
host オプション, 330

iterations オプション, 330
login-path オプション, 330
no-defaults オプション, 330
no-drop オプション, 330
number-char-cols オプション, 330
number-int-cols オプション, 330
number-of-queries オプション, 331
only-print オプション, 331
password オプション, 331
pipe オプション, 331
plugin-dir オプション, 331
port オプション, 331
post-query オプション, 331
post-system オプション, 331
pre-query オプション, 331
pre-system オプション, 331
print-defaults オプション, 331
protocol オプション, 331
query オプション, 332
secure-auth オプション, 332
shared-memory-base-name オプション, 332
silent オプション, 332
socket オプション, 332
SSL オプション, 332
user オプション, 332
verbose オプション, 332
version オプション, 332
mysqltest
MySQL テストスイート, 2768
MySQL 権限
および MySQL Cluster, 2345
mysql コマンド
のリスト, 276
mysql コマンドオプション, 267
MySQL システムテーブル
および MySQL Cluster, 2345
およびレプリケーション, 1994
MySQL ストレージエンジン, 1759
MySQL スレッドプール, 896
MySQL ソース配布, 45
MySQL のアップグレード, 261
MySQL の主な機能, 5
MySQL の機能, 5
MySQL の空間拡張, 1041
MySQL の取得, 46
MySQL の名前, 7
MySQL のバージョン, 46
MySQL の歴史, 7, 7
MySQL バイナリ配布, 45
MySQL メーリングリスト, 18
mysql 履歴ファイル, 281
MySQL をダウンロードするための URL, 46

N

NAME_CONST(), 1205, 2496
name_file オプション
comp_err, 254
named-commands オプション
mysql, 272
named_pipe システム変数, 508
NATIONAL CHAR データ型, 1017
NATIONAL VARCHAR データ型, 1017
NATURAL LEFT JOIN, 1335
NATURAL LEFT OUTER JOIN, 1335
NATURAL RIGHT JOIN, 1335
NATURAL RIGHT OUTER JOIN, 1335
NCHAR データ型, 1017
NDB$MAX(), 2402
NDB$MAX_DELETE_WIN(), 2402
NDB$OLD, 2402
ndb-batch-size オプション
mysqld, 2178
ndb-blob-read-batch-bytes オプション
mysqld, 2179
ndb-blob-write-batch-bytes オプション
mysqld, 2179
ndb-cluster-connection-pool オプション
mysqld, 2178
ndb-connectstring オプション
mysqld, 2180
ndb_config, 2242
ndb-connectstring オプション (MySQL Cluster プログラム), 2285
ndb-deferred-constraints オプション (MySQL Cluster), 2180
ndb-distribution オプション (MySQL Cluster), 2181
ndb-log-apply-status オプション
mysqld, 2182
ndb-log-empty-epochs オプション
mysqld, 2183
ndb-log-exclusive-reads オプション
mysqld, 2183
ndb-log-orig オプション
mysqld, 2184
ndb-log-transaction-id オプション
mysqld, 2184
ndb-log-update-as-write (mysqld オプション), 2400
ndb-mgmd-host オプション (MySQL Cluster), 2181
ndb-mgmd-host オプション (MySQL Cluster プログラム), 2286
ndb-nodegroup-map オプション (ndb_restore), 2268
ndb-nodeid オプション
mysqld, 2184
ndb-nodeid オプション (MySQL Cluster プログラム), 2286
ndb-optimized-node-selection オプション (MySQL Cluster), 2286
NDB API
および配布された権限, 2365
および配布された付与テーブル, 2365
NDB API _slave ステータス変数
および MySQL Cluster レプリケーション, 2376
NDB API カウンタ (MySQL Cluster), 2365
関連付けられたステータス変数, 2369
スコープ, 2367
タイプ, 2368
NDB
MAX_ROWS, 1270
大きなテーブルの作成, 1270
ndb_apply_status テーブル (MySQL Cluster レプリケーション), 2383, 2384, 2389
(参照 MySQL Cluster レプリケーション)
ndb_binlog_index テーブル (MySQL Cluster レプリケーション), 2382, 2389
(参照 MySQL Cluster レプリケーション)
ndb_blob_tool, 2222, 2238
check-orphans オプション, 2239
database オプション, 2239
delete-orphans オプション, 2239
dump-file オプション, 2239

verbose オプション, 2239
ndb_config, 2222, 2240
config-file オプション, 2242
config_from_node オプション, 2241
configinfo オプション, 2245
connections オプション, 2243
fields オプション, 2244
host オプション, 2243
id オプション, 2243
mycnf オプション, 2242
ndb-connectstring オプション, 2242
nodeid オプション, 2243
nodes オプション, 2243
query オプション, 2242, 2243
rows オプション, 2244
system オプション, 2243
type オプション, 2244
usage オプション, 2241
version オプション, 2242
xml オプション, 2245
ndb_cpcd, 2222, 2247
ndb_delete_all, 2222, 2247
transactional オプション, 2247
ndb_desc, 2222, 2248
blob-info オプション, 2251
database option, 2251
extra-node-info option, 2251
extra-partition-info option, 2251
retries option, 2251
table option, 2251
unqualified option, 2251
ndb_dist_priv.sql, 2362
ndb_drop_index, 2222, 2251
ndb_drop_table, 2222, 2252
ndb_error_reporter, 2222, 2253
オプション, 2253
ndb_index_stat, 2222, 2254
出力の解釈, 2254
例, 2254
ndb_log_empty_epochs システム変数, 2198
ndb_log_exclusive_reads (システム変数), 2409
ndb_log_exclusive_reads システム変数, 2198
ndb_log_orig システム変数, 2198
ndb_log_transaction_id システム変数, 2199
ndb_mgm (MySQL Cluster 管理ノードクライアント), 2061
ndb_mgm, 2222, 2237 (参照 mgm)
MySQL Cluster Manager での使用, 2289
ndb_mgmd (MySQL Cluster)
定義, 2011
(参照 管理ノード (MySQL Cluster))
[ndb_mgmd] (MySQL Cluster), 2153
ndb_mgmd (MySQL Cluster プロセス), 2230
ndb_mgmd, 2222 (参照 mgmd)
mycnf オプション, 2233
ndb_print_backup_file, 2222, 2258
ndb_print_file, 2222, 2259
ndb_print_schema_file, 2222, 2259
ndb_print_sys_file, 2222, 2260
ndb_replication システムテーブル, 2401
ndb_restore, 2261
append オプション, 2267
backup_path オプション, 2266
disable-indexes オプション, 2271
dont_ignore_systab_0 オプション, 2268
exclude-databases オプション, 2269
exclude-intermediate-sql-tables オプション, 2272
exclude-missing-columns オプション, 2270
exclude-missing-tables オプション, 2271
exclude-tables オプション, 2269
fields-enclosed-by オプション, 2267
fields-optionally-enclosed-by オプション, 2267
fields-terminated-by オプション, 2267, 2267
hex オプション, 2267
include-databases オプション, 2269
include-tables オプション, 2269
lossy-conversions オプション, 2265
ndb-nodegroup-map オプション, 2268
no-binlog オプション, 2268
no-restore-disk-objects オプション, 2268
no-upgrade オプション, 2266
parallelism オプション, 2268
print_data オプション, 2266
print_log オプション, 2268
print_meta オプション, 2268
print オプション, 2268
progress-frequency オプション, 2268
rebuild-indexes オプション, 2271
restore-privilege-tables オプション, 2266
rewrite-database オプション, 2271
skip-broken-objects オプション, 2271
skip-unknown-objects オプション, 2271
tab オプション, 2267
verbose オプション, 2268
一般的なオプションおよび必須のオプション, 2264
エラー, 2272
および循環レプリケーション, 2397
および配布された権限, 2364
属性昇格, 2264
ndb_schema テーブル (MySQL Cluster レプリケーション), 2384
(参照 MySQL Cluster レプリケーション)
ndb_select_all, 2222, 2272
database オプション, 2273
delimiter オプション, 2274
descending オプション, 2274
disk オプション, 2274
gci64 オプション, 2274
gci オプション, 2274
header オプション, 2274
lock オプション, 2273
nodata オプション, 2274
order オプション, 2274
parallelism オプション, 2273
rowid オプション, 2274
tupscan オプション, 2274
useHexFormat オプション, 2274
ndb_select_count, 2222, 2275
ndb_setup.py, 2222, 2276
browser-start-page オプション, 2277
ca-certs-file オプション, 2277
cert-file オプション, 2277
debug-level オプション, 2277
help オプション, 2277
key-file オプション, 2277
no-browser オプション, 2278
port オプション, 2278
server-log-file オプション, 2278
server-name オプション, 2278

use-https オプション, 2278
ndb_show_tables, 2222, 2278
database オプション, 2279
loops オプション, 2279
parsable オプション, 2279
show-temp-status オプション, 2279
type オプション, 2279
unqualified オプション, 2279
ndb_size.pl, 2222, 2279
ndb_size.pl スクリプト, 1055
ndb_waiter, 2222, 2282
no-contact オプション, 2283
not-started オプション, 2283
nowait-nodes オプション, 2283
single-user オプション, 2283
timeout オプション, 2283
wait-nodes オプション, 2283
ndbcluster オプション
mysqld, 2182
NDBCLUSTER ストレージエンジン (参照 MySQL Cluster)
[ndbd default] (MySQL Cluster), 2142
ndbd (MySQL Cluster)
定義, 2011
(参照 データノード (MySQL Cluster))
[ndbd] (MySQL Cluster), 2142
ndbd, 2222, 2222
ndbd_redo_log_reader, 2260
ndbinfo_select_all, 2222, 2228
ndbinfo データベース, 2319
およびクエリーキャッシュ, 2321
基本的な使用方法, 2322
サポートの判断, 2320
ndbmtd, 2222, 2229
MaxNoOfExecutionThreads, 2115
構成, 2119, 2119
トレースファイル, 2230, 2230
ndb オプション
perror, 391
NDB ストレージエンジン (参照 MySQL Cluster)
FAQ, 2850
NDB テーブル
および MySQL ルートユーザー, 2345
NDB 統計の変数
および NDB API カウンタ, 2369
NDB 統計の変数 (MySQL Cluster), 2365
スコープ, 2367
タイプ, 2368
NDB ユーティリティー
セキュリティー上の問題, 2347
Nested Loop 結合アルゴリズム, 789, 792
net_buffer_length システム変数, 508
net_buffer_length 変数, 276
net_read_timeout システム変数, 509
net_retry_count システム変数, 509
net_write_timeout システム変数, 509
new システム変数, 510
NFS
InnoDB, 1523, 1552
nice オプション
mysqld_safe, 248
no-auto-rehash オプション
mysql, 272
no-autocommit オプション
mysqldump, 314
no-beep オプション
mysql, 272
mysqladmin, 291
no-binlog オプション (ndb_restore), 2268
no-browser オプション
ndb_setup.py, 2278
no-contact オプション
ndb_waiter, 2283
no-create-db オプション
mysqldump, 306
no-create-info オプション
mysqldump, 306
no-data オプション
mysqldump, 312
no-defaults オプション, 241, 257
my_print_defaults, 389
myisamchk, 338
mysql, 272
mysql_plugin, 259
mysql_upgrade, 265
mysqladmin, 291
mysqlbinlog, 368
mysqlcheck, 298
mysqld, 441
mysqld_multi, 251
mysqld_safe, 248
mysqldump, 306
mysqlimport, 319
mysqlshow, 324
mysqlslap, 330
no-drop オプション
mysqlslap, 330
no-log オプション
mysqld_multi, 252
no-nodeid-checks オプション (ndb_mgmd), 2232
no-restore-disk-objects オプション (ndb_restore), 2268
no-set-names オプション
mysqldump, 308
no-symlinks オプション
myisamchk, 341
no-tablespaces オプション
mysqldump, 306
no-upgrade オプション (ndb_restore), 2266
NO_AUTO_CREATE_USER SQL モード, 588
NO_AUTO_VALUE_ON_ZERO SQL モード, 588
NO_BACKSLASH_ESCAPES SQL モード, 588
NO_DIR_IN_CREATE SQL モード, 588
NO_ENGINE_SUBSTITUTION SQL モード, 588
NO_FIELD_OPTIONS SQL モード, 589
NO_KEY_OPTIONS SQL モード, 589
NO_TABLE_OPTIONS SQL モード, 589
NO_UNSIGNED_SUBTRACTION SQL モード, 589
NO_ZERO_DATE SQL モード, 589
NO_ZERO_IN_DATE SQL モード, 590
nodaemon オプション (ndb_mgmd), 2234
nodata オプション
ndb_select_all, 2274
NodeGroup, 2082
NodeId1, 2133, 2136, 2138
NodeId2, 2133, 2136, 2138
NodeId, 2076, 2081, 2128
nodeid オプション
ndb_config, 2243
NODERESTART イベント (MySQL Cluster), 2303

nodes
  ndbinfo テーブル, 2335
nodes オプション
  ndb_config, 2243
noindices オプション
  mysqlhotcopy, 383
NoOfDiskPagesToDiskAfterRestartACC (非推奨), 2107
NoOfDiskPagesToDiskAfterRestartTUP (非推奨), 2105
NoOfDiskPagesToDiskDuringRestartACC, 2107
NoOfDiskPagesToDiskDuringRestartTUP, 2107
NoOfFragmentLogFiles, 2092
NoOfFragmentLogParts, 2119
NoOfReplicas, 2082
nopager コマンド
  mysql, 278
NoSQL, 2997
nostart オプション (ndbd), 2226
nostart オプション (ndbmtd), 2226
not-started オプション
  ndb_waiter, 2283
NOT BETWEEN, 1077
NOT EXISTS
  サブクエリーを使用した, 1348
NOT IN, 1078
NOT LIKE, 1098
NOT
  論理, 1079
NOT NULL
  制約, 33
NOT NULL 制約, 2997
NOT REGEXP, 1099
notee コマンド
  mysql, 278
Notifier, 88
NOW(), 1123
nowait-nodes オプション
  ndb_waiter, 2283
nowait-nodes オプション (ndb_mgmd), 2235
nowait-nodes オプション (ndbd), 2225
nowait-nodes オプション (ndbmtd), 2225
NOWAIT (START BACKUP コマンド), 2293
nowarning コマンド
  mysql, 278
NUL, 918, 1318
NULL, 2998
NULL, 209, 2962
  NULL かどうかのテスト, 1076, 1077, 1082
  NULL のテスト, 1075
  ORDER BY, 802, 1330
  testing for null, 1077
  スレッドの状態, 907
NULLIF(), 1083
NULL 値, 209, 922
  および AUTO_INCREMENT カラム, 2963
  および TIMESTAMP カラム, 2963
  とインデックス, 1265
  と空の値, 2962
Numa, 2115
number-char-cols オプション
  mysqlslap, 330
number-int-cols オプション
  mysqlslap, 330
number-of-queries オプション
  mysqlslap, 331
numeric-dump-file オプション
  resolve_stack_dump, 390
NUMERIC データ型, 1014
NumGeometries(), 1189
NumInteriorRings(), 1187
NumPoints(), 1186
NVARCHAR データ型, 1017

O

objects_summary_global_by_type テーブル
  performance_schema, 2625
OCT(), 1090
OCTET_LENGTH(), 1090
ODBC_INCLUDES= オプション
  CMake, 155
ODBC_LIB_DIR オプション
  CMake, 155
ODBC 互換性, 532, 925, 1263
ODBC との互換性, 1014, 1071, 1076
ODBC の互換性, 1337
ODirect, 2098
offset オプション
  mysqlbinlog, 368
OLAP, 1210
old-alter-table オプション
  mysqld, 441
old-style-user-limits オプション
  mysqld, 441
old_alter_table システム変数, 510
OLD_PASSWORD(), 1167
old_passwords システム変数, 510
old_server オプション
  mysqlaccess, 361
  mysqlhotcopy, 383
old システム変数, 510
OLTP, 2998
ON DUPLICATE KEY UPDATE, 1305
one-database オプション
  mysql, 272
one-thread オプション
  mysqld, 441
only-print オプション
  mysqlslap, 331
ONLY_FULL_GROUP_BY
  SQL モード, 1213
ONLY_FULL_GROUP_BY SQL モード, 590
open-files-limit オプション
  mysqld, 441
  mysqld_safe, 248
OPEN, 1394
open_files_limit variable, 371
open_files_limit システム変数, 511
OpenGIS, 1041
Opening master dump table
  スレッドの状態, 914
Opening mysql.ndb_apply_status
  スレッドの状態, 914
Opening table
  スレッドの状態, 908
Opening tables
  スレッドの状態, 908
OpenSSL, 715, 715
OPTIMIZE TABLE, 1434
  およびパーティショニング, 2450

optimizer_join_cache_level システム変数, 512
optimizer_prune_level システム変数, 512
optimizer_search_depth システム変数, 513
optimizer_switch システム変数, 513, 859
OPTIMIZER_TRACE
  INFORMATION_SCHEMA テーブル, 2509
optimizer_trace_features システム変数, 515
optimizer_trace_limit システム変数, 516
optimizer_trace_max_mem_size システム変数, 516
optimizer_trace_offset システム変数, 516
OPTIMIZER_TRACE オプション
  CMake, 158
optimizer_trace システム変数, 515
optimize オプション
  mysqlcheck, 298
optimizing
  スレッドの状態, 908
opt オプション
  mysqldump, 313
OR, 221, 781
  ビット単位, 1162
  論理, 1080
ORACLE SQL モード, 592
Oracle との互換性, 1210
Oracle の互換性, 27, 1231, 1488
ORD(), 1090
order-by-primary オプション
  mysqldump, 314
ORDER BY, 206, 1232, 1330
  NULL, 802, 1330
order オプション
  ndb_select_all, 2274
OR インデックスマージの最適化, 781
OS X
  インストール, 109
out_dir オプション
  comp_err, 254
out_file オプション
  comp_err, 255
OUTFILE, 1334
Overlaps(), 1192
OverloadLimit, 2134, 2137, 2140

P

PAD_CHAR_TO_FULL_LENGTH SQL モード, 590
page size, 2998
pager オプション
  mysql, 273
pager コマンド
  mysql, 278
PAM
  プラガブル認証, 700
parallel-recover オプション
  myisamchk, 341
parallelism オプション (ndb_restore), 2268
PARAMETERS
  INFORMATION_SCHEMA テーブル, 2509
parsable オプション
  ndb_show_tables, 2279
PARTITION BY LIST COLUMNS, 2422
PARTITION BY RANGE COLUMNS, 2422
PARTITION, 2411
PARTITIONS
  INFORMATION_SCHEMA テーブル, 2510
partition オプション
  mysqld, 442
PASSWORD(), 673, 687, 1168, 2952
password オプション, 232
  mysql, 273
  mysql_config_editor, 359
  mysql_convert_table_format, 384
  mysql_setpermission, 386
  mysql_upgrade, 265
  mysqlaccess, 361
  mysqladmin, 291
  mysqlbinlog, 368
  mysqlcheck, 298
  mysqld_multi, 252
  mysqldump, 304
  mysqlhotcopy, 383
  mysqlimport, 320
  mysqlshow, 324
  mysqlslap, 331
PATH 環境変数, 164, 192, 230
performance-schema-consumer-events-stages-current オプション
  mysqld, 2639
performance-schema-consumer-events-stages-history-long オプション
  mysqld, 2639
performance-schema-consumer-events-stages-history オプション
  mysqld, 2639
performance-schema-consumer-events-statements-current オプション
  mysqld, 2639
performance-schema-consumer-events-statements-history-long オプション
  mysqld, 2639
performance-schema-consumer-events-statements-history オプション
  mysqld, 2639
performance-schema-consumer-events-waits-current オプション
  mysqld, 2639
performance-schema-consumer-events-waits-history-long オプション
  mysqld, 2639
performance-schema-consumer-events-waits-history オプション
  mysqld, 2639
performance-schema-consumer-global-instrumentation オプション
  mysqld, 2639
performance-schema-consumer-statements-digest オプション
  mysqld, 2639
performance-schema-consumer-thread-instrumentation オプション
  mysqld, 2639
performance-schema-consumer-xxx オプション
  mysqld, 2639
performance-schema-instrument オプション
  mysqld, 2639
performance_schema
  accounts テーブル, 2618
  cond_instances テーブル, 2602
  events_stages_current テーブル, 2610

events_stages_history_long テーブル, 2611
events_stages_history テーブル, 2611
events_stages_summary_by_account_by_event_name テーブル, 2629
events_stages_summary_by_host_by_event_name テーブル, 2629
events_stages_summary_by_thread_by_event_name テーブル, 2623
events_stages_summary_by_user_by_event_name テーブル, 2629
events_stages_summary_global_by_event_name テーブル, 2623
events_statements_current テーブル, 2614
events_statements_history_long テーブル, 2617
events_statements_history テーブル, 2617
events_statements_summary_by_account_by_event_name テーブル, 2629
events_statements_summary_by_digest テーブル, 2623
events_statements_summary_by_host_by_event_name テーブル, 2629
events_statements_summary_by_thread_by_event_name テーブル, 2623
events_statements_summary_by_user_by_event_name テーブル, 2629
events_statements_summary_global_by_event_name テーブル, 2623
events_waits_current テーブル, 2607
events_waits_history_long テーブル, 2609
events_waits_history テーブル, 2609
events_waits_summary_by_account_by_event_name テーブル, 2629
events_waits_summary_by_host_by_event_name テーブル, 2629
events_waits_summary_by_instance テーブル, 2622
events_waits_summary_by_thread_by_event_name テーブル, 2622
events_waits_summary_by_user_by_event_name テーブル, 2629
events_waits_summary_global_by_event_name テーブル, 2622
file_instances テーブル, 2602
file_summary_by_event_name テーブル, 2625
file_summary_by_instance テーブル, 2625
host_cache テーブル, 2631
hosts テーブル, 2619
mutex_instances テーブル, 2603
objects_summary_global_by_type テーブル, 2625
performance_timers テーブル, 2633
rwlock_instances テーブル, 2604
session_account_connect_attrs テーブル, 2620
session_connect_attrs テーブル, 2620
setup_actors テーブル, 2598
setup_consumers テーブル, 2599
setup_instruments テーブル, 2599
setup_objects テーブル, 2600
setup_timers テーブル, 2601
socket_instances テーブル, 2604
socket_summary_by_event_name テーブル, 2630
socket_summary_by_instance テーブル, 2630
table_io_waits_summary_by_index_usage テーブル, 2627
table_io_waits_summary_by_table テーブル, 2627
table_lock_waits_summary_by_table テーブル, 2628
users テーブル, 2619
スレッドテーブル, 2634
performance_schema_accounts_size システム変数, 2640
performance_schema_digests_size システム変数, 2641
performance_schema_events_stages_history_long_size システム変数, 2641
performance_schema_events_stages_history_size システム変数, 2641
performance_schema_events_statements_history_long_size システム変数, 2642
performance_schema_events_statements_history_size システム変数, 2642
performance_schema_events_waits_history_long_size システム変数, 2642
performance_schema_events_waits_history_size システム変数, 2642
performance_schema_hosts_size システム変数, 2643
performance_schema_max_cond_classes システム変数, 2643
performance_schema_max_cond_instances システム変数, 2643
performance_schema_max_file_classes システム変数, 2643
performance_schema_max_file_handles システム変数, 2643
performance_schema_max_file_instances システム変数, 2644
performance_schema_max_mutex_classes システム変数, 2644
performance_schema_max_mutex_instances システム変数, 2644
performance_schema_max_rwlock_classes システム変数, 2644
performance_schema_max_rwlock_instances システム変数, 2645
performance_schema_max_socket_classes システム変数, 2645
performance_schema_max_socket_instances システム変数, 2645
performance_schema_max_stage_classes システム変数, 2645
performance_schema_max_statement_classes システム変数, 2646
performance_schema_max_table_handles システム変数, 2646
performance_schema_max_table_instances システム変数, 2646
performance_schema_max_thread_classes システム変数, 2646
performance_schema_max_thread_instances システム変数, 2647
performance_schema_session_connect_attrs_size システム変数, 2647
performance_schema_setup_actors_size システム変数, 2647
performance_schema_setup_objects_size システム変数, 2647
performance_schema_users_size システム変数, 2648
performance_schema システム変数, 2640
PERFORMANCE_SCHEMA ストレージエンジン, 2565
performance_schema データベース, 2565
  TRUNCATE TABLE, 2596, 2980
  制約, 2980
performance_timers テーブル
  performance_schema, 2633
PERIOD_ADD(), 1124
PERIOD_DIFF(), 1124
Perl API, 2763
Perl DBI/DBD

インストールに関する問題, 194
Perl
Windows にインストール, 194
インストール, 193
perror, 229, 390
help オプション, 391
ndb オプション, 391
silent オプション, 391
verbose オプション, 391
version オプション, 391
PI(), 1110
pid-file オプション
mysql.server, 250
mysqld, 442
mysqld_safe, 248
pid_file システム変数, 516
Ping
スレッドのコマンド, 904
PIPES_AS_CONCAT SQL モード, 591
pipe オプション, 232
mysql, 273, 298
mysql_upgrade, 265
mysqladmin, 291
mysqldump, 304
mysqlimport, 320
mysqlshow, 324
mysqlslap, 331
PITR, 2998
plan オプション
mysqlaccess, 361
plugin-dir オプション
mysql, 273
mysql_plugin, 260
mysql_upgrade, 265
mysqladmin, 292
mysqlbinlog, 368
mysqlcheck, 298
mysqld_safe, 248
mysqldump, 304
mysqlimport, 320
mysqlshow, 324
mysqlslap, 331
plugin-ini オプション
mysql_plugin, 260
plugin-load-add オプション
mysqld, 443
plugin-load オプション
mysqld, 443
plugin_dir システム変数, 516
plugindir オプション
mysql_config, 388
PLUGINS
INFORMATION_SCHEMA テーブル, 2512
plugin オプションプリフィクス
mysqld, 442
Point(), 1183
PointFromText(), 1182
PointFromWKB(), 1182
PointN(), 1186
POINT データ型, 1043
PolyFromText(), 1182
PolyFromWKB(), 1183
Polygon(), 1183
PolygonFromText(), 1182
PolygonFromWKB(), 1183
POLYGON データ型, 1043
port-open-timeout オプション
mysqld, 444
PortNumber, 2077
PortNumberStats, 2079
PortNumber (廃止), 2135
ports, 368, 1909
port オプション, 232
mysql, 273
mysql_config, 388
mysql_config_editor, 359
mysql_convert_table_format, 384
mysql_setpermission, 386
mysql_upgrade, 265
mysqladmin, 292
mysqlbinlog, 368
mysqlcheck, 298
mysqld, 444
mysqld_safe, 248
mysqldump, 305
mysqlhotcopy, 383
mysqlimport, 320
mysqlshow, 324
mysqlslap, 331
ndb_setup.py, 2278
port システム変数, 517
POSITION(), 1090
post-query オプション
mysqlslap, 331
post-system オプション
mysqlslap, 331
POSTGRESQL SQL モード, 592
PostgreSQL の互換性, 28
POW(), 1110
POWER(), 1110
pre-query オプション
mysqlslap, 331
pre-system オプション
mysqlslap, 331
preload_buffer_size システム変数, 517
Prepare
スレッドのコマンド, 905
PREPARE, 1383, 1386
XA トランザクション, 1369
preparing for alter table
スレッドの状態, 908
preparing
スレッドの状態, 908
preserve-trailing-spaces オプション (ndb_restore), 2265
preview オプション
mysqlaccess, 361
PRIMARY KEY, 1232, 1265
print-defaults オプション, 241
myisamchk, 338
mysql, 273
mysql_plugin, 260
mysql_upgrade, 265
mysqladmin, 292
mysqlbinlog, 368
mysqlcheck, 298
mysqld, 444
mysqldump, 306
mysqlimport, 320

mysqlshow, 325
mysqlslap, 331
print-full-config オプション (ndb_mgmd), 2234
print_data オプション (ndb_restore), 2266
print_log オプション (ndb_restore), 2268
print_meta オプション (ndb_restore), 2268
print オプション (ndb_restore), 2268
print コマンド
mysql, 279
PROCEDURE ANALYSE(), 833
PROCEDURE, 1332
Processing events from schema table
スレッドの状態, 914
Processing events
スレッドの状態, 914
Processlist
スレッドのコマンド, 905
PROCESSLIST, 1461
INFORMATION_SCHEMA テーブル, 2513
INFORMATION_SCHEMA テーブルとの不整合の可能性, 1691
PROFILING
INFORMATION_SCHEMA テーブル, 2514
profiling_history_size システム変数, 517
profiling システム変数, 517
progress-frequency オプション (ndb_restore), 2268
promote-attributes オプション (ndb_restore), 2264
prompt オプション
mysql, 273
prompt コマンド
mysql, 279
protocol_version システム変数, 518
protocol オプション, 232
mysql, 273
mysql_upgrade, 265
mysqladmin, 292
mysqlbinlog, 368
mysqlcheck, 298
mysqldump, 305
mysqlimport, 320
mysqlshow, 325
mysqlslap, 331
proxy_user システム変数, 518
pseudo_slave_mode システム変数, 518
pseudo_thread_id システム変数, 518
Pthreads, 2998
PURGE BINARY LOGS, 1371
PURGE MASTER LOGS, 1371
Purging old relay logs
スレッドの状態, 908
Python, 2657
サードパーティードライバ, 2764

Q

QUARTER(), 1124
query end
スレッドの状態, 908
Query
スレッドのコマンド, 905
query_alloc_block_size システム変数, 518
query_cache_limit システム変数, 519
query_cache_min_res_unit システム変数, 519
query_cache_size システム変数, 519
query_cache_type システム変数, 520
query_cache_wlock_invalidate システム変数, 520
query_prealloc_size システム変数, 521
query オプション
mysqlslap, 332
ndb_config, 2242, 2243
query オプション (ndb_index_stat), 2257
Queueing master event to the relay log
スレッドの状態, 913
quick オプション
myisamchk, 341
mysql, 273
mysqlcheck, 298
mysqldump, 313
quiet オプション
mysqlhotcopy, 383
Quit
スレッドのコマンド, 905
quit コマンド
mysql, 279
QUIT コマンド (MySQL Cluster), 2291
quote-names オプション
mysqldump, 310
QUOTE(), 919, 1090, 2711

R

RADIANS(), 1110
RAID, 2999
RAND(), 1110
rand_seed1 システム変数, 521
rand_seed2 システム変数, 521
random-passwords オプション
mysql_install_db, 257
RANDOM_BYTES(), 1169
range_alloc_block_size システム変数, 521
range 結合型
オプティマイザ, 851
RANGE パーティショニング, 2416, 2422
RANGE パーティション
管理, 2439
追加および削除, 2439
raw オプション
mysql, 273
mysqlbinlog, 368
RAW パーティション, 1538
raw バックアップ, 2999
read-from-remote-master オプション
mysqlbinlog, 369
read-from-remote-server オプション
mysqlbinlog, 369
read-only オプション
myisamchk, 340
mysqld, 1901
READ COMMITTED, 2999
READ COMMITTED
トランザクション分離レベル, 1367
READ UNCOMMITTED, 2999
READ UNCOMMITTED
トランザクション分離レベル, 1367
read_buffer_size myisamchk 変数, 338
read_buffer_size システム変数, 522
read_only システム変数, 522
read_rnd_buffer_size システム変数, 523
Reading event from the relay log
スレッドの状態, 913

Reading from net
スレッドの状態, 908
Reading master dump table data
スレッドの状態, 914
REAL_AS_FLOAT SQL モード, 591
RealtimeScheduler, 2114
REAL データ型, 1014
rebuild-indexes オプション (ndb_restore), 2271
Rebuilding the index on master dump table
スレッドの状態, 914
ReceiveBufferMemory, 2135
Reconnecting after a failed binlog dump request
スレッドの状態, 913
Reconnecting after a failed master event read
スレッドの状態, 913
reconnect オプション
mysql, 274
record_log_pos オプション
mysqlhotcopy, 383
RECOVER
XA トランザクション, 1369
recover オプション
myisamchk, 341
Redo, 2999
RedoBuffer, 2109
RedoOverCommitCounter
データノード, 2125
RedoOverCommitLimit
データノード, 2125
Redo ログ, 1536
Redo ログ, 2999
ref_or_null, 787
ref_or_null 結合型
オプティマイザ, 850
REFERENTIAL_CONSTRAINTS
INFORMATION_SCHEMA テーブル, 2515
Refresh
スレッドのコマンド, 905
ref 結合型
オプティマイザ, 850
REGEXP, 1099
REGEXP 演算子, 1099
regexp オプション
mysql_find_rows, 385
mysqlhotcopy, 383
Register Slave
スレッドのコマンド, 905
Registering slave on master
スレッドの状態, 913
rehash コマンド
mysql, 279
relative オプション
mysqladmin, 292
relay-log-index オプション
mysqld, 1902
relay-log-info-file オプション
mysqld, 1903
relay-log-info-repository オプション
mysqld, 1915
relay-log-purge オプション
mysqld, 1903
relay-log-recovery オプション
mysqld, 1903
relay-log-space-limit オプション
mysqld, 1904
relay-log オプション
mysqld, 1902
relay_log_basename システム変数, 1917
relay_log_index システム変数, 1917
relay_log_info_file システム変数, 1917
relay_log_info_repository システム変数, 1918
relay_log_purge システム変数, 524
relay_log_recovery システム変数, 1918
relay_log_space_limit システム変数, 524
relay_log システム変数, 1917
RELEASE SAVEPOINT, 1359
RELEASE_LOCK(), 1205
relnotes オプション
mysqlaccess, 361
reload オプション (ndb_mgmd), 2235
remove オプション
mysqld, 444
MySQLInstallerConsole, 88
remove オプション (ndb_mgmd), 2237
remove オプション (ndbd), 2226
remove オプション (ndbmtd), 2226
Removing duplicates
スレッドの状態, 908
removing tmp table
スレッドの状態, 908
rename
スレッドの状態, 908
rename result table
スレッドの状態, 908
RENAME TABLE, 1296
RENAME USER, 1427
Reopen tables
スレッドの状態, 908
Repair by sorting
スレッドの状態, 908
Repair done
スレッドの状態, 908
REPAIR TABLE, 1436
およびパーティショニング, 2450
およびレプリケーション, 1437, 1992
Repair with keycache
スレッドの状態, 909
repair オプション
mysqlcheck, 298
REPEAT, 1392
ラベル, 1387
REPEAT(), 1090
REPEATABLE READ, 2999
REPEATABLE READ
トランザクション分離レベル, 1366
replace, 229
REPLACE, 1326
REPLACE(), 1090
replace オプション
mysqldump, 306
mysqlimport, 320
replace ユーティリティー, 391
replicate-do-db オプション
mysqld, 1904
replicate-do-table オプション
mysqld, 1906
replicate-ignore-db オプション
mysqld, 1905

replicate-ignore-table オプション
  mysqld, 1906
replicate-rewrite-db オプション
  mysqld, 1907
replicate-same-server-id オプション
  mysqld, 1907
replicate-wild-do-table オプション
  mysqld, 1907
replicate-wild-ignore-table オプション
  mysqld, 1908
report-host オプション
  mysqld, 1908
report-password オプション
  mysqld, 1909
report-port オプション
  mysqld, 1909
report-user オプション
  mysqld, 1909
report_host システム変数, 524
report_password システム変数, 524
report_port システム変数, 525
report_user システム変数, 525
REPORT コマンド (MySQL Cluster), 2290
Requesting binlog dump
  スレッドの状態, 913
REQUIRE GRANT オプション, 1425
reschedule
  スレッドの状態, 911
ReservedSendBufferMemory, 2125
reset-slave.pl (参照 MySQL Cluster レプリケーション)
RESET MASTER, 1372
RESET SLAVE ALL, 1378
RESET SLAVE, 1378
Reset stmt
  スレッドのコマンド, 905
resetmaster オプション
  mysqlhotcopy, 383
resetslave オプション
  mysqlhotcopy, 383
RESIGNAL, 1400
resolve_stack_dump, 229, 390
  help オプション, 390
  numeric-dump-file オプション, 390
  symbols-file オプション, 390
  version オプション, 390
resolveip, 229, 392
  help オプション, 392
  silent オプション, 392
  version オプション, 392
resources
  ndbinfo テーブル, 2336
RestartOnErrorInsert, 2098
RESTART コマンド (MySQL Cluster), 2290
restore-privilege-tables オプション (ndb_restore), 2266
restore_connect オプション (ndb_restore), 2264
restore_data オプション (ndb_restore), 2266
restore_epoch オプション (ndb_restore), 2266
restore_meta オプション (ndb_restore), 2266
restore_nodeid オプション (ndb_restore), 2264
restore_skip-table-check オプション (ndb_restore), 2264
result-file オプション
  mysqlbinlog, 369
  mysqldump, 310
retries option
  ndb_desc, 2251
RETURN, 1393
REVERSE(), 1091
REVOKE, 1427
rewrite-database オプション (ndb_restore), 2271
rhost オプション
  mysqlaccess, 361
RIGHT JOIN, 1335
RIGHT OUTER JOIN, 1335
RIGHT(), 1091
RLIKE, 1099
ROLLBACK, 29, 1355
  XA トランザクション, 1369
ROLLBACK TO SAVEPOINT, 1359
rollback オプション
  mysqlaccess, 361
Rolling back
  スレッドの状態, 909
ROLLUP, 1210
root のパスワード, 173
root ユーザー, 644
  パスワードのリセット, 2954
ROUND(), 1112
ROUTINES
  INFORMATION_SCHEMA テーブル, 2515
routines オプション
  mysqldump, 312
ROW, 1347
ROW_COUNT(), 1178
ROW_FORMAT
  COMPACT, 1572
  COMPRESSED, 1555, 1572
  DYNAMIC, 1572
  REDUNDANT, 1572
rowid オプション
  ndb_select_all, 2274
rows オプション
  mysql_find_rows, 385
  ndb_config, 2244
RPAD(), 1091
Rpl_semi_sync_master_clients ステータス変数, 580
rpl_semi_sync_master_enabled システム変数, 525
Rpl_semi_sync_master_net_avg_wait_time ステータス変数, 580
Rpl_semi_sync_master_net_wait_time ステータス変数, 580
Rpl_semi_sync_master_net_waits ステータス変数, 580
Rpl_semi_sync_master_no_times ステータス変数, 580
Rpl_semi_sync_master_no_tx ステータス変数, 580
Rpl_semi_sync_master_status ステータス変数, 581
Rpl_semi_sync_master_timefunc_failures ステータス変数, 581
rpl_semi_sync_master_timeout システム変数, 525
rpl_semi_sync_master_trace_level システム変数, 525
Rpl_semi_sync_master_tx_avg_wait_time ステータス変数, 581
Rpl_semi_sync_master_tx_wait_time ステータス変数, 581
Rpl_semi_sync_master_tx_waits ステータス変数, 581
rpl_semi_sync_master_wait_no_slave システム変数, 526
Rpl_semi_sync_master_wait_pos_backtraverse ステータス変数, 581
Rpl_semi_sync_master_wait_sessions ステータス変数, 581
Rpl_semi_sync_master_yes_tx ステータス変数, 581
rpl_semi_sync_slave_enabled システム変数, 526
Rpl_semi_sync_slave_status ステータス変数, 581

rpl_semi_sync_slave_trace_level システム変数, 526
rpl_stop_slave_timeout システム変数, 1919
rpm オプション
  mysql_install_db, 258
RPM パッケージマネージャー, 129
RPM ファイル, 123, 127, 129
RTRIM(), 1091
Ruby API, 2764
rwlock_instances テーブル
  performance_schema, 2604
rw ロック (読み書きロック), 3000

S

safe-mode オプション
  mysqld, 445
safe-recover オプション
  myisamchk, 341
safe-updates オプション, 285
  mysql, 274
safe-user-create オプション
  mysqld, 445
Sakila, 7
SASL, 1731
SAVEPOINT, 1359
Saving state
  スレッドの状態, 909
SchedulerExecutionTimer, 2114
SchedulerSpinTimer, 2114
SCHEMA(), 1178
SCHEMA_PRIVILEGES
  INFORMATION_SCHEMA テーブル, 2517
SCHEMATA
  INFORMATION_SCHEMA テーブル, 2517
SCHEMA イベント (MySQL Cluster), 2305
[sci] (MySQL Cluster), 2157
SCI (スケーラブルコヒーラントインタフェース) (参照 MySQL Cluster)
Searching rows for update
  スレッドの状態, 909
SEC_TO_TIME(), 1124
SECOND(), 1124
secure-auth オプション
  mysql, 274
  mysqladmin, 292
  mysqlbinlog, 369
  mysqlcheck, 299
  mysqld, 445
  mysqldump, 305
  mysqlimport, 320
  mysqlshow, 325
  mysqlslap, 332
secure-file-priv オプション
  mysqld, 445
secure_auth システム変数, 527
secure_file_priv システム変数, 527
SELECT INTO TABLE, 28
SELECT
  INTO, 1333
  LIMIT, 1328
  クエリーキャッシュ, 868
  最適化, 846, 1488
select_limit 変数, 276
SELECT 速度, 775
SendBufferMemory, 2134
Sending binlog event to slave
  スレッドの状態, 912
sending cached result to client
  スレッドの状態, 912
SendLimit, 2140
SendSignalId, 2134, 2137, 2140
SEQUENCE, 221
SERIAL DEFAULT VALUE, 1055
SERIAL, 1011, 1013
SERIALIZABLE, 3000
SERIALIZABLE
  トランザクション分離レベル, 1367
server-id-bits オプション
  mysqlbinlog, 369
  mysqld, 2188
server-id オプション
  mysqlbinlog, 369
  mysqld, 1879
server-log-file オプション
  ndb_setup.py, 2278
server-name オプション
  ndb_setup.py, 2278
server-public-key-path オプション
  mysql, 274
server_id_bits システム変数, 2206
server_id システム変数, 527
server_operations
  ndbinfo テーブル, 2337
server_transactions
  ndbinfo テーブル, 2338
server_uuid システム変数
  mysqld, 1880
ServerPort, 2081
service-startup-timeout オプション
  mysql.server, 250
session_account_connect_attrs テーブル
  performance_schema, 2620
session_connect_attrs テーブル
  performance_schema, 2620
SESSION_STATUS
  INFORMATION_SCHEMA テーブル, 2507
SESSION_USER(), 1178
SESSION_VARIABLES
  INFORMATION_SCHEMA テーブル, 2508
set-auto-increment[オプション
  myisamchk, 342
set-character-set オプション
  myisamchk, 341
set-charset オプション
  mysqlbinlog, 369
  mysqldump, 308
set-collation オプション
  myisamchk, 342
set-gtid-purged オプション
  mysqldump, 309
SET GLOBAL sql_slave_skip_counter, 1378
Set option
  スレッドのコマンド, 905
SET PASSWORD, 1428
SET PASSWORD ステートメント, 687
SET, 1440
  CHARACTER SET, 954, 1442
  NAMES, 954, 956, 1442
  ONE_SHOT, 1443

サイズ, 1058
SET sql_log_bin, 1373
SET TRANSACTION, 1365
setup.bat
MySQL Cluster (Windows), 2276
setup
スレッドの状態, 909
setup_actors テーブル
performance_schema, 2598
setup_consumers テーブル
performance_schema, 2599
setup_instruments テーブル
performance_schema, 2599
setup_objects テーブル
performance_schema, 2600
setup_timers テーブル
performance_schema, 2601
SET ステートメント
割り当て演算子, 1081
SET データ型, 1019, 1040
SHA1(), 1169
SHA2(), 1169
sha256_password_private_key_path システム変数, 528
sha256_password_public_key_path システム変数, 528
sha256_password 認証プラグイン, 697
SHA(), 1169
shared-memory-base-name オプション, 233
mysql, 275
mysql_upgrade, 265
mysqladmin, 292
mysqlbinlog, 369
mysqlcheck, 299
mysqld, 446
mysqldump, 314
mysqlimport, 320
mysqlshow, 325
mysqlslap, 332
shared-memory オプション
mysqld, 446
shared_memory_base_name システム変数, 529
shared_memory システム変数, 528
SharedBufferSize, 2139
SharedGlobalMemory, 2121
[shm] (MySQL Cluster), 2157
ShmKey, 2137
ShmSize, 2137
short-form オプション
mysqlbinlog, 370
show-slave-auth-info オプション
mysqld, 1909
show-table-type オプション
mysqlshow, 325
show-temp-status オプション
ndb_show_tables, 2279
show-warnings オプション
mysql, 275
SHOW AUTHORS, 1443, 1444
SHOW BINARY LOGS, 1443, 1444
SHOW BINLOG EVENTS, 1443, 1444
SHOW CHARACTER SET, 1443, 1444
SHOW COLLATION, 1443, 1445
SHOW COLUMNS, 1443, 1445
SHOW CONTRIBUTORS, 1443, 1447
SHOW CREATE DATABASE, 1443, 1447
SHOW CREATE EVENT, 1443
SHOW CREATE FUNCTION, 1443, 1448
SHOW CREATE PROCEDURE, 1443, 1448
SHOW CREATE SCHEMA, 1443, 1447
SHOW CREATE TABLE, 1443, 1448
SHOW CREATE TRIGGER, 1443, 1448
SHOW CREATE VIEW, 1443, 1449
SHOW DATABASES, 1443, 1450
SHOW ENGINE INNODB STATUS, 1450
SHOW ENGINE INNODB ステータス
と innodb_use_sys_malloc, 1669
SHOW ENGINE NDB STATUS, 1450, 2317
SHOW ENGINE NDBCLUSTER STATUS, 1450, 2317
SHOW ENGINE, 1443, 1450
および MySQL Cluster, 2317
SHOW ENGINES, 1443, 1453
および MySQL Cluster, 2318
SHOW ERRORS, 1443, 1455
SHOW EVENTS, 1443, 1455
SHOW FIELDS, 1443, 1446
SHOW FUNCTION CODE, 1443, 1456
SHOW FUNCTION STATUS, 1443, 1456
SHOW GRANTS, 1443, 1457
SHOW INDEX, 1443, 1457
SHOW KEYS, 1443, 1457
SHOW MASTER LOGS, 1443, 1444
SHOW MASTER STATUS, 1443, 1458
SHOW OPEN TABLES, 1443, 1459
SHOW PLUGINS, 1443, 1459
SHOW PRIVILEGES, 1443, 1460
SHOW PROCEDURE CODE, 1443, 1460
SHOW PROCEDURE STATUS, 1443, 1461
SHOW PROCESSLIST, 1443, 1461
SHOW PROFILE, 1443, 1463
SHOW PROFILES, 1443, 1463, 1465
SHOW RELAYLOG EVENTS, 1466
SHOW SCHEDULER STATUS, 2485
SHOW SCHEMAS, 1450
SHOW
MySQL Cluster 管理クライアントでの, 2068
SHOW SLAVE HOSTS, 1443, 1466
SHOW SLAVE STATUS, 1443, 1467
SHOW STATUS, 1443
および MySQL Cluster, 2319
SHOW STORAGE ENGINES, 1453
SHOW TABLE STATUS, 1443
SHOW TABLES, 1443, 1476
SHOW TRIGGERS, 1443, 1476
SHOW VARIABLES, 1443
および MySQL Cluster, 2318
SHOW WARNINGS, 1443, 1478
SHOW 拡張, 2561
SHOW コマンド (MySQL Cluster), 2289
Shutdown
スレッドのコマンド, 905
shutdown_timeout 変数, 293
SHUTDOWN コマンド (MySQL Cluster), 2291
Shutting down
スレッドの状態, 914
sigint-ignore オプション
mysql, 275
SIGN(), 1113
SIGNAL, 1404
SigNum, 2138

silent オプション
myisamchk, 338
myisampack, 351
mysql, 275
mysqladmin, 292
mysqlcheck, 299
mysqld_multi, 252
mysqlimport, 321
mysqlslap, 332
perror, 391
resolveip, 392
simplified_binlog_gtid_recovery, 1949
SIN(), 1113
single-transaction オプション
mysqldump, 314
single-user オプション
ndb_waiter, 2283
SINGLEUSER イベント (MySQL Cluster), 2305
skip-broken-objects オプション (ndb_restore), 2271
skip-column-names オプション
mysql, 275
skip-comments オプション
mysqldump, 307
skip-concurrent-insert オプション
mysqld, 446
skip-database オプション
mysqlcheck, 299
skip-event-scheduler オプション
mysqld, 446
skip-grant-tables オプション
mysqld, 446
skip-gtids オプション
mysqlbinlog, 370
skip-host-cache オプション
mysqld, 447
skip-innodb オプション
mysqld, 447, 1611
skip-kill-mysqld オプション
mysqld_safe, 248
skip-line-numbers オプション
mysql, 275
skip-name-resolve オプション
mysql_install_db, 258
mysqld, 447
skip-ndbcluster オプション
mysqld, 2188
skip-networking オプション
mysqld, 447
skip-nodegroup option (ndb_error_reporter), 2254
skip-opt オプション
mysqldump, 313
skip-partition オプション
mysqld, 447
skip-show-database オプション
mysqld, 448
skip-slave-start オプション
mysqld, 1912
skip-ssl オプション, 718
skip-stack-trace オプション
mysqld, 448
skip-symbolic-links オプション
mysqld, 448
skip-syslog オプション
mysqld_safe, 248
skip-thread-priority オプション
mysqld, 448
skip-unknown-objects オプション (ndb_restore), 2271
skip-use-db オプション
mysql_find_rows, 385
skip_external_locking システム変数, 529
skip_name_resolve システム変数, 529
skip_networking システム変数, 529
skip_show_database システム変数, 529
slave-checkpoint-group オプション
mysqld, 1910
slave-checkpoint-period オプション
mysqld, 1910
slave-load-tmpdir オプション
mysqld, 1912
slave-max-allowed-packet (mysqld), 1912
slave-net-timeout オプション
mysqld, 1913
slave-parallel-workers オプション
mysqld, 1910
slave-pending-jobs-size-max オプション
mysqld, 1911
slave-rows-search-algorithms (mysqld), 1913
slave-skip-errors オプション
mysqld, 1914
slave-sql-verify-checksum オプション
mysqld, 1914
Slave has read all relay log; waiting for more updates
スレッドの状態, 914
slave_allow_batching, 2387
slave_checkpoint_group システム変数, 1919
slave_checkpoint_period システム変数, 1919
slave_compressed_protocol オプション
mysqld, 1912
slave_compressed_protocol システム変数, 1920
slave_exec_mode システム変数, 1920
slave_load_tmpdir システム変数, 1920
slave_max_allowed_packet システム変数, 1921
slave_net_timeout システム変数, 1921
slave_parallel_workers システム変数, 1921
slave_pending_jobs_size_max システム変数, 1922
slave_rows_search_algorithms システム変数, 1923
slave_skip_errors システム変数, 1924
slave_sql_verify_checksum システム変数, 1924
slave_transaction_retries システム変数, 1924
slave_type_conversions システム変数, 1925
Sleep
スレッドのコマンド, 905
SLEEP(), 1206
sleep オプション
mysqladmin, 292
slow-query-log オプション
mysqld, 448
slow-start-timeout オプション
mysqld, 449
slow_launch_time システム変数, 530
slow_query_log_file システム変数, 530
slow_query_log システム変数, 530
SMALLINT データ型, 1013
SNAPSHOTEND (START BACKUP コマンド), 2293
SNAPSHOTSTART (START BACKUP コマンド), 2293
socket_instances テーブル
performance_schema, 2604
socket_summary_by_event_name テーブル

performance_schema, 2630
socket_summary_by_instance テーブル
performance_schema, 2630
socket オプション, 233
mysql, 275
mysql_config, 388
mysql_config_editor, 359
mysql_convert_table_format, 384
mysql_setpermission, 386
mysql_upgrade, 265
mysqladmin, 292
mysqlbinlog, 370
mysqlcheck, 299
mysqld, 449
mysqld_safe, 248
mysqldump, 305
mysqlhotcopy, 383
mysqlimport, 321
mysqlshow, 325
mysqlslap, 332
socket システム変数, 530
Solaris
インストール, 139
Solaris x86_64 の問題, 839
Solaris トラブルシューティング, 163
Solaris のインストールの問題, 139
SOME, 1346
sort-index オプション
myisamchk, 342
sort-records オプション
myisamchk, 342
sort-recover オプション
myisamchk, 342
sort_buffer_size myisamchk 変数, 338
sort_buffer_size システム変数, 531
sort_key_blocks myisamchk 変数, 338
Sorting for group
スレッドの状態, 909
Sorting for order
スレッドの状態, 909
Sorting index
スレッドの状態, 909
Sorting result
スレッドの状態, 909
SOUNDEX(), 1091
SOUNDS LIKE, 1091
source (mysql クライアントコマンド), 217, 283
source コマンド
mysql, 279
SPACE(), 1092
spassword オプション
mysqlaccess, 361
sporadic-binlog-dump-fail オプション
mysqld, 1933
SQL, 3000
SQL-92
への拡張, 24
sql-mode オプション
mysqld, 449
SQL
定義, 4
sql_auto_is_null システム変数, 532
SQL_BIG_RESULT, 1332
sql_big_selects システム変数, 532
SQL_BUFFER_RESULT, 1333
sql_buffer_result システム変数, 532
SQL_CACHE, 870, 1333
SQL_CALC_FOUND_ROWS, 1333
sql_log_bin システム変数, 532
sql_log_off システム変数, 533
sql_mode システム変数, 533
SQL_NO_CACHE, 870, 1333
sql_notes システム変数, 534
sql_quote_show_create システム変数, 534
sql_safe_updates システム変数, 534
sql_select_limit システム変数, 534
sql_slave_skip_counter, 1378
sql_slave_skip_counter システム変数, 1925
SQL_SMALL_RESULT, 1332
SQL_THREAD_WAIT_AFTER_GTIDS(), 1194
sql_warnings システム変数, 534
SQL スクリプト, 266
SQL ステートメント
レプリケーションスレーブ, 1373
レプリケーションマスター, 1371
SQL ノード (MySQL Cluster), 2296
定義, 2011
SQL モード, 585
ALLOW_INVALID_DATES, 587
ANSI, 586, 592
ANSI_QUOTES, 587
DB2, 592
ERROR_FOR_DIVISION_BY_ZERO, 587
HIGH_NOT_PRECEDENCE, 587
IGNORE_SPACE, 588
MAXDB, 592
MSSQL, 592
MYSQL323, 592
MYSQL40, 592
NO_AUTO_CREATE_USER, 588
NO_AUTO_VALUE_ON_ZERO, 588
NO_BACKSLASH_ESCAPES, 588
NO_DIR_IN_CREATE, 588
NO_ENGINE_SUBSTITUTION, 588
NO_FIELD_OPTIONS, 589
NO_KEY_OPTIONS, 589
NO_TABLE_OPTIONS, 589
NO_UNSIGNED_SUBTRACTION, 589
NO_ZERO_DATE, 589
NO_ZERO_IN_DATE, 590
ONLY_FULL_GROUP_BY, 590, 1213
ORACLE, 592
PAD_CHAR_TO_FULL_LENGTH, 590
PIPES_AS_CONCAT, 591
POSTGRESQL, 592
REAL_AS_FLOAT, 591
strict, 587
STRICT_ALL_TABLES, 591
STRICT_TRANS_TABLES, 587, 591
TRADITIONAL, 587, 592
およびパーティショニング, 1997, 2461
およびレプリケーション, 1997
SQRT(), 1113
srcdir オプション
mysql_install_db, 258
SRID(), 1184
SSD, 3000
SSD, 1555

SSH, 656, 725
ssl-capath オプション, 719
ssl-ca オプション, 718
ssl-cert オプション, 719
ssl-cipher オプション, 719
ssl-crlpath オプション, 720
ssl-crl オプション, 720
ssl-key オプション, 721
ssl-verify-server-cert オプション, 721
SSL, 715
  X509 の基本, 715
  構成, 715
  コマンドオプション, 717
  接続の確立, 716
ssl_capath システム変数, 535
ssl_ca システム変数, 534
ssl_cert システム変数, 535
ssl_cipher システム変数, 535
ssl_crlpath システム変数, 535
ssl_crl システム変数, 535
ssl_key システム変数, 536
SSL オプション, 233
  mysql, 275
  mysql_upgrade, 265
  mysqladmin, 292
  mysqlcheck, 299
  mysqld, 447
  mysqldump, 305
  mysqlimport, 321
  mysqlshow, 325
  mysqlslap, 332
ssl オプション, 718
SSL 関連オプション, 1425
ST_Area(), 1188, 1188
ST_Centroid(), 1188
ST_Contains(), 1190
ST_Crosses(), 1190
ST_Difference(), 1189
ST_Disjoint(), 1191
ST_Distance(), 1191
ST_Envelope(), 1185
ST_Equals(), 1191
ST_Intersection(), 1189
ST_Intersects(), 1191
ST_Overlaps(), 1191
ST_SymDifference(), 1189
ST_Touches(), 1191
ST_Union(), 1190
ST_Within(), 1191
standalone オプション
  mysqld, 447
start-datetime オプション
  mysqlbinlog, 370
start-position オプション
  mysqlbinlog, 370
START BACKUP
  NOWAIT, 2293
  SNAPSHOTEND, 2293
  SNAPSHOTSTART, 2293
  WAIT COMPLETED, 2293
  WAIT STARTED, 2293
  構文, 2293
START SLAVE, 1379
START
  XA トランザクション, 1369
START TRANSACTION, 1355
start_row オプション
  mysql_find_rows, 385
StartConnectBackoffMaxTime, 2132
StartFailRetryDelay, 2126
StartFailureTimeout, 2100
StartNoNodeGroupTimeout, 2100
StartPartialTimeout, 2100
StartPartitionedTimeout, 2100
StartPoint(), 1186
StartupStatusReportFrequency, 2111
STARTUP イベント (MySQL Cluster), 2302
START コマンド (MySQL Cluster), 2290
statefile オプション
  comp_err, 255
Statistics
  スレッドのコマンド, 905
statistics
  スレッドの状態, 909
STATISTICS
  INFORMATION_SCHEMA テーブル, 2517
STATISTICS イベント (MySQL Cluster), 2304
stats_method myisamchk 変数, 338
stats オプション
  myisam_ftdump, 334
status オプション
  MySQLInstallerConsole, 88
  mysqlshow, 325
status コマンド
  mysql, 279
  結果, 288
STATUS コマンド (MySQL Cluster), 2290
STD(), 1210
STDDEV(), 1210
STDDEV_POP(), 1210
STDDEV_SAMP(), 1210
stop-datetime オプション
  mysqlbinlog, 370
stop-never-slave-server-id オプション
  mysqlbinlog, 370
stop-never オプション
  mysqlbinlog, 370
stop-position オプション
  mysqlbinlog, 371
STOP SLAVE, 1382
StopOnError, 2098
STOP コマンド (MySQL Cluster), 2290
storage_engine システム変数, 536
storing result in query cache
  スレッドの状態, 912
storing row into queue
  スレッドの状態, 911
STR_TO_DATE(), 1124
STRAIGHT_JOIN, 788, 789, 846, 856, 1332, 1335, 1489
STRCMP(), 1098
strict SQL モード, 587
STRICT_ALL_TABLES SQL モード, 591
STRICT_TRANS_TABLES SQL モード, 587, 591
StringMemory, 2086
SUBDATE(), 1125
SUBPARTITION BY KEY
  既知の問題, 2464
SUBSTR(), 1092

SUBSTRING(), 1092
SUBSTRING_INDEX(), 1092
SUBTIME(), 1125
suffix オプション
  mysqlhotcopy, 383
SUM(DISTINCT), 1210
SUM(), 1210
SUNPRO_CXX_LIBRARY オプション
  CMake, 160
super-large-pages オプション
  mysqld, 448
superuser オプション
  mysqlaccess, 361
symbolic-links オプション
  mysqld, 448
symbols-file オプション
  resolve_stack_dump, 390
sync_binlog システム変数, 1943
sync_frm システム変数, 537
sync_master_info システム変数, 1926
sync_relay_log_info システム変数, 1927
sync_relay_log システム変数, 1926
Syncing ndb table schema operation and binlog
  スレッドの状態, 914
sys-check オプション (ndb_index_stat), 2258
sys-create-if-not-exist オプション (ndb_index_stat), 2257
sys-create-if-not-valid オプション (ndb_index_stat), 2257
sys-create オプション (ndb_index_stat), 2257
sys-drop オプション (ndb_index_stat), 2257
sys-skip-events オプション (ndb_index_stat), 2258
sys-skip-tables オプション (ndb_index_stat), 2258
SYSCONFDIR オプション
  CMake, 155
sysdate-is-now オプション
  mysqld, 450
SYSDATE(), 1126
syslog-tag オプション
  mysqld_safe, 249
syslog オプション
  mysqld_safe, 248
System lock
  スレッドの状態, 909
system variable
  innodb_adaptive_hash_index, 1614
system_time_zone システム変数, 537
SYSTEM_USER(), 1178
system オプション
  ndb_config, 2243
system コマンド
  mysql, 279
system テーブル
  オプティマイザ, 849, 1332
sysvar_stored_program_cache システム変数, 536

T

Table Dump
  スレッドのコマンド, 905
「Table is full」エラー (MySQL Cluster), 2084
table option
  ndb_desc, 2251
table_definition_cache システム変数, 537
table_io_waits_summary_by_index_usage テーブル
  performance_schema, 2627
table_io_waits_summary_by_table テーブル
  performance_schema, 2627
table_lock_waits_summary_by_table テーブル
  performance_schema, 2628
table_open_cache, 834
table_open_cache_instances システム変数, 538
table_open_cache システム変数, 538
TABLE_PRIVILEGES
  INFORMATION_SCHEMA テーブル, 2520
tables
  圧縮, 350
TABLES
  INFORMATION_SCHEMA テーブル, 2518
TABLESPACE
  INFORMATION_SCHEMA テーブル, 2519
tables オプション
  mysqlcheck, 299
  mysqldump, 312
table オプション
  mysql, 275
  mysqlaccess, 361
tab オプション
  mysqldump, 310
tab オプション (ndb_restore), 2267
TAN(), 1113
tar
  Solaris に関する問題, 139, 139
tc-heuristic-recover オプション
  mysqld, 450
Tcl API, 2765
tcp-ip オプション
  mysqld_multi, 252
TCP/IP, 101, 105, 158, 191, 230, 248, 273, 368, 388, 444, 620, 656, 676, 894, 1909, 2604, 2942
[tcp] (MySQL Cluster), 2157
TCP_MAXSEG_SIZE, 2135
TCP_RCV_BUF_SIZE, 2135
TCP_SND_BUF_SIZE, 2135
tee オプション
  mysql, 275
tee コマンド
  mysql, 279
temp-pool オプション
  mysqld, 451
test_plugin_server 認証プラグイン, 711
test オプション
  myisampack, 351
TEXT
  サイズ, 1057
TEXT カラム
  インデックス設定, 826, 1266
  デフォルト値, 1036
TEXT データ型, 1018, 1036
thread_cache_size システム変数, 538
thread_concurrency システム変数, 539
thread_handling システム変数, 539
thread_pool_algorithm システム変数, 540
thread_pool_high_priority_connection システム変数, 540
thread_pool_max_unused_threads システム変数, 540
thread_pool_prio_kickup_timer システム変数, 541
thread_pool_size システム変数, 541
thread_pool_stall_limit システム変数, 542
thread_stack システム変数, 542
threadblocks
  ndbinfo テーブル, 2338

ThreadConfig, 2119
ThreadPool (参照 DiskIOThreadPool)
threadstat
  ndbinfo テーブル, 2339
Time
  スレッドのコマンド, 905
TIME(), 1126
TIME_FORMAT(), 1127
time_format システム変数, 542
TIME_TO_SEC(), 1127
time_zone システム変数, 543
TimeBetweenEpochs, 2103
TimeBetweenEpochsTimeout, 2103
TimeBetweenGlobalCheckpoints, 2102, 2124
TimeBetweenInactiveTransactionAbortCheck, 2104
TimeBetweenLocalCheckpoints, 2102
TimeBetweenWatchDogCheck, 2099
TimeBetweenWatchDogCheckInitial, 2099
timed_mutexes システム変数, 543
TIMEDIFF(), 1126
timeout, 474
  connect_timeout 変数, 276
timeout オプション
  ndb_waiter, 2283
TIMESTAMP
  および NULL 値, 2963
  およびレプリケーション, 1980
  初期化および更新機能, 1028
TIMESTAMP(), 1127
TIMESTAMPADD(), 1127
TIMESTAMPDIFF(), 1127
timestamp システム変数, 543
TIMESTAMP データ型, 1015, 1023
timezone オプション
  mysqld_safe, 249
TIME データ型, 1016, 1025
TINYBLOB データ型, 1018
TINYINT データ型, 1012
TINYTEXT データ型, 1018
tmp_table_size システム変数, 543
TMPDIR オプション
  CMake, 156
tmpdir オプション
  myisamchk, 342
  myisampack, 351
  mysql_upgrade, 266
  mysqld, 451
  mysqlhotcopy, 383
TMPDIR 環境変数, 169, 192, 229, 2958
tmpdir システム変数, 544
to-last-log オプション
  mysqlbinlog, 371
TO_BASE64(), 1092
TO_DAYS(), 1127
TO_SECONDS(), 1128
TotalSendBufferMemory
  API および SQL ノード, 2130
  管理ノード, 2080
  データノード, 2125
Touches(), 1192
TPS, 3000
trace DBI メソッド, 2825
TRADITIONAL SQL モード, 587, 592
transaction-isolation オプション
  mysqld, 451
transaction-read-only オプション
  mysqld, 451
transaction_alloc_block_size システム変数, 544
transaction_allow_batching セッション変数 (MySQL Cluster), 2206
transaction_prealloc_size システム変数, 544
transactional オプション
  ndb_delete_all, 2247
TransactionBufferMemory, 2089
TransactionDeadlockDetectionTimeout, 2104
TransactionInactiveTimeout, 2104
transporters
  ndbinfo テーブル, 2339
TRIGGERS
  INFORMATION_SCHEMA テーブル, 2520
triggers オプション
  mysqldump, 312
TRIM(), 1093
TRUE, 919, 922
  テスト, 1076, 1076
TRUNCATE TABLE, 1297
  performance_schema データベース, 2596, 2980
  および MySQL Cluster, 2025
  およびレプリケーション, 1999
TRUNCATE(), 1113
tupscan オプション
  ndb_select_all, 2274
two-phase commit, 1657
TwoPassInitialNodeRestartCopy, 2114
tx_isolation システム変数, 545
tx_read_only システム変数, 545
type オプション
  mysql_convert_table_format, 384
  ndb_config, 2244
  ndb_show_tables, 2279
tz-utc オプション
  mysqldump, 310
TZ 環境変数, 192, 2960

U

UCASE(), 1093
UCS-2, 945
ucs2 文字セット, 971
UDF, 1438, 1438
  コンパイル, 2819
  定義済み, 2812
  戻り値, 2818
ulimit, 2953
UMASK_DIR 環境変数, 192, 2954
UMASK 環境変数, 192, 2953
unbuffered オプション
  mysql, 275
UNCOMPRESS(), 1170
UNCOMPRESSED_LENGTH(), 1170
Undo, 3001
UndoDataBuffer, 2108
UndoIndexBuffer, 2108
Undo テーブルスペース, 3001
Undo ログ, 1528, 1535
Undo ログ, 3001
UNHEX(), 1093
Unicode Collation Algorithm, 979
Unicode, 945

UNINSTALL PLUGIN, 1440
UNION, 221, 1342
UNIQUE, 1232
unique_checks システム変数, 546
unique_subquery 結合型
    オプティマイザ, 850
Unix
    クライアントのコンパイル, 2664
UNIX_TIMESTAMP(), 1129
UNKNOWN
    テスト, 1076, 1076
UNLOCK TABLES, 1360
unpack オプション
    myisamchk, 342
unqualified option
    ndb_desc, 2251
unqualified オプション
    ndb_show_tables, 2279
UNSIGNED, 1011, 1019
UNTIL, 1392
updatable_views_with_limit システム変数, 546
update-state オプション
    myisamchk, 340
UPDATE, 28, 1353
update
    スレッドの状態, 909
UpdateXML(), 1155
update オプション
    MySQLInstallerConsole, 88
update オプション (ndb_index_stat), 2256
updating main table
    スレッドの状態, 909
updating reference tables
    スレッドの状態, 910
Updating
    スレッドの状態, 909
upgrade-system-tables オプション
    mysql_upgrade, 266
upgrade オプション
    MySQLInstallerConsole, 88
upgrading lock
    スレッドの状態, 911
UPPER(), 1093
usage オプション
    ndb_config, 2241
usage オプション (MySQL Cluster プログラム), 2285
use-frm オプション
    mysqlcheck, 299
use-https オプション
    ndb_setup.py, 2278
use-mysqld_safe オプション
    mysql.server, 250
use-threads オプション
    mysqlimport, 321
USE INDEX, 1340
USE KEY, 1340
USE, 1491
useHexFormat オプション
    ndb_select_all, 2274
User lock
    スレッドの状態, 910
User sleep
    スレッドの状態, 910
USER(), 1179
USER_PRIVILEGES
    INFORMATION_SCHEMA テーブル, 2522
users テーブル
    performance_schema, 2619
user オプション, 233
    mysql.server, 250
    mysql, 275
    mysql_config_editor, 359
    mysql_convert_table_format, 384
    mysql_install_db, 258
    mysql_setpermission, 386
    mysql_upgrade, 266
    mysqlaccess, 361
    mysqladmin, 292
    mysqlbinlog, 371
    mysqlcheck, 299
    mysqld, 452
    mysqld_multi, 252
    mysqld_safe, 249
    mysqldump, 305
    mysqlhotcopy, 383
    mysqlimport, 321
    mysqlshow, 325
    mysqlslap, 332
USER 環境変数, 192, 233
user テーブル
    ソート, 674
use コマンド
    mysql, 279
UTC_DATE(), 1129
UTC_TIME(), 1130
UTC_TIMESTAMP(), 1130
UTF-8, 945
utf16le 文字セット, 971
utf16_bin 照合順序, 981
utf16 文字セット, 971
utf32 文字セット, 971
utf8mb3 文字セット, 972
utf8mb4 文字セット, 973
utf8 文字セット, 972
UUID(), 1206
UUID_SHORT(), 1206

V

validate-password オプション
    mysqld, 654
validate_password_dictionary_file システム変数, 654
validate_password_length システム変数, 655
validate_password_mixed_case_count システム変数, 655
validate_password_number_count システム変数, 655
validate_password_policy システム変数, 656
validate_password_special_char_count システム変数, 656
validate_password プラグイン, 652
    インストール, 653
    構成, 654
    システム変数, 654
validate_user_plugins システム変数, 546
VALUES(), 1207
VAR_POP(), 1210
VAR_SAMP(), 1210
VARBINARY データ型, 1018, 1035
VARCHAR
    サイズ, 1057
VARCHARACTER データ型, 1017

VARCHAR データ型, 1017, 1033
VARIANCE(), 1210
verbose オプション
  my_print_defaults, 390
  myisam_ftdump, 334
  myisamchk, 338
  myisampack, 351
  mysql, 275
  mysql_config_editor, 359
  mysql_convert_table_format, 384
  mysql_install_db, 258
  mysql_plugin, 260
  mysql_upgrade, 266
  mysql_waitpid, 386
  mysqladmin, 293
  mysqlbinlog, 371
  mysqlcheck, 299
  mysqld, 452
  mysqld_multi, 252
  mysqldump, 307
  mysqldumpslow, 381
  mysqlimport, 321
  mysqlshow, 325
  mysqlslap, 332
  ndb_blob_tool, 2239
  perror, 391
verbose オプション (ndb_index_stat), 2258
verbose オプション (ndb_restore), 2268
verify-binlog-checksum オプション
  mysqlbinlog, 371
version-check オプション
  mysql_upgrade, 266
VERSION(), 1179
version_comment システム変数, 547
version_compile_machine システム変数, 547
version_compile_os システム変数, 547
version オプション
  comp_err, 255
  my_print_defaults, 390
  myisamchk, 338
  myisampack, 351
  mysql, 275
  mysql_config, 388
  mysql_config_editor, 359
  mysql_convert_table_format, 384
  mysql_plugin, 260
  mysql_waitpid, 386
  mysqlaccess, 362
  mysqladmin, 293
  mysqlbinlog, 371
  mysqlcheck, 299
  mysqld, 452
  mysqld_multi, 252
  mysqldump, 308
  mysqlimport, 321
  mysqlshow, 325
  mysqlslap, 332
  ndb_config, 2242
  perror, 391
  resolve_stack_dump, 390
  resolveip, 392
version オプション (MySQL Cluster プログラム), 2287
version システム変数, 547
VERSION ファイル
  CMake, 163
vertical オプション
  mysql, 275
  mysqladmin, 293
VIEWS
  INFORMATION_SCHEMA テーブル, 2522

W

wait-nodes オプション
  ndb_waiter, 2283
WAIT COMPLETED (START BACKUP コマンド), 2293
WAIT STARTED (START BACKUP コマンド), 2293
wait_timeout システム変数, 547
WAIT_UNTIL_SQL_THREAD_AFTER_GTIDS(), 1194
Waiting for allowed to take ndbcluster global schema lock
  スレッドの状態, 915
Waiting for commit lock
  スレッドの状態, 910
waiting for delay_list
  スレッドの状態, 911
Waiting for event from ndbcluster
  スレッドの状態, 915
Waiting for event metadata lock
  スレッドの状態, 910
Waiting for event read lock
  スレッドの状態, 910
Waiting for first event from ndbcluster
  スレッドの状態, 915
Waiting for global read lock
  スレッドの状態, 910
waiting for handler insert
  スレッドの状態, 911
waiting for handler lock
  スレッドの状態, 911
waiting for handler open
  スレッドの状態, 911
Waiting for INSERT
  スレッドの状態, 911
Waiting for master to send event
  スレッドの状態, 913
Waiting for master update
  スレッドの状態, 912
Waiting for ndbcluster binlog update to reach current position
  スレッドの状態, 915
Waiting for ndbcluster global schema lock
  スレッドの状態, 915
Waiting for ndbcluster to start
  スレッドの状態, 915
Waiting for next activation
  スレッドの状態, 915
Waiting for query cache lock
  スレッドの状態, 912
Waiting for scheduler to stop
  スレッドの状態, 915
Waiting for schema epoch
  スレッドの状態, 915
Waiting for schema metadata lock
  スレッドの状態, 910
Waiting for slave mutex on exit
  スレッドの状態, 913, 914
Waiting for stored function metadata lock
  スレッドの状態, 910
Waiting for stored procedure metadata lock
  スレッドの状態, 910

Waiting for table level lock
スレッドの状態, 910
Waiting for table metadata lock
スレッドの状態, 910
Waiting for table
スレッドの状態, 910
Waiting for tables
スレッドの状態, 910
Waiting for the next event in relay log
スレッドの状態, 913
Waiting for the slave SQL thread to free enough relay log space
スレッドの状態, 913
Waiting for trigger metadata lock
スレッドの状態, 910
Waiting on cond
スレッドの状態, 910
Waiting on empty queue
スレッドの状態, 915
Waiting to finalize termination
スレッドの状態, 912
Waiting to reconnect after a failed binlog dump request
スレッドの状態, 913
Waiting to reconnect after a failed master event read
スレッドの状態, 913
Waiting until MASTER_DELAY seconds after master executed event
スレッドの状態, 914
wait オプション
myisamchk, 338
myisampack, 351
mysql, 276
mysqladmin, 293
Wan, 2079, 2131
warning_count システム変数, 548
warnings コマンド
mysql, 279
warn オプション
mysql_config_editor, 360
WEEK(), 1130
WEEKDAY(), 1131
WEEKOFYEAR(), 1131
WEIGHT_STRING(), 1094
Well-Known Binary 形式, 1049
Well-Known Text 形式, 1048
WHERE, 776
SHOW を伴う, 2500, 2561
where オプション
mysqldump, 312
WHERE を伴う SHOW, 2500, 2561
WHILE, 1393
ラベル, 1387
Windows, 3001
Windows Server フェイルオーバークラスタリング, 1798
Windows
MySQL の制限, 2985, 2986
アップグレード, 107
クライアントのコンパイル, 2665
パス名区切り文字, 238
プラガブル認証, 706
windows オプション
mysql_install_db, 258
Windows フェイルオーバークラスタリング, 1798
WITH_ASAN オプション
CMake, 158
WITH_BUNDLED_LIBEVENT オプション
CMake, 161
WITH_BUNDLED_MEMCACHED オプション
CMake, 161
WITH_CLASSPATH オプション
CMake, 161
WITH_DEBUG オプション
CMake, 158
WITH_DEFAULT_COMPILER_OPTIONS オプション
CMake, 160
WITH_DEFAULT_FEATURE_SET オプション
CMake, 158
WITH_EDITLINE オプション
CMake, 158
WITH_EMBEDDED_SERVER オプション
CMake, 158
WITH_EMBEDDED_SHARED_LIBRARY オプション
CMake, 158
WITH_ERROR_INSERT オプション
CMake, 161
WITH_EXTRA_CHARSETS オプション
CMake, 159
WITH_INNODB_MEMCACHED オプション
CMake, 159
WITH_LIBEDIT オプション
CMake, 159
WITH_LIBEVENT オプション
CMake, 159
WITH_LIBWRAP オプション
CMake, 159
WITH_NDB_BINLOG オプション
CMake, 161
WITH_NDB_DEBUG オプション
CMake, 161
WITH_NDB_JAVA オプション
CMake, 162
WITH_NDB_PORT オプション
CMake, 162
WITH_NDB_TEST オプション
CMake, 162
WITH_NDBCLUSTER_STORAGE_ENGINE オプション
CMake, 161
WITH_NDBCLUSTER オプション
CMake, 161
WITH_NDBMTD オプション
CMake, 161
WITH_READLINE オプション
CMake, 159
WITH_SSL オプション
CMake, 159
WITH_UNIXODBC オプション
CMake, 159
WITH_VALGRIND オプション
CMake, 159
WITH_ZLIB オプション
CMake, 160
Within(), 1192
WITHOUT_SERVER オプション
CMake, 160
WKB 形式, 1049
WKT 形式, 1048
write-binlog オプション
mysql_upgrade, 266

mysqlcheck, 299
write_buffer_size myisamchk 変数, 338
Writing to net
スレッドの状態, 910

X

X509/証明書, 715
X(), 1185
XA, 3001
XA BEGIN, 1369
XA COMMIT, 1369
XA PREPARE, 1369
XA RECOVER, 1369
XA ROLLBACK, 1369
XA START, 1369
XA トランザクション, 1367
制約, 2979
トランザクション識別子, 1369
xid
XA トランザクション識別子, 1369
xml オプション
mysql, 276
mysqldump, 311
ndb_config, 2245
XOR
ビット単位, 1162
論理, 1080

Y

Y(), 1185
yaSSL, 715
YEAR(), 1131
YEARWEEK(), 1131
YEAR データ型, 1016, 1025

Z

ZEROFILL, 1011, 1019, 2759
ZFS, 1804

C 関数の索引

my_init()
セクション23.7.6「C API 関数の概要」
セクション23.7.12.1「my_init()」
セクション23.7.12.3「mysql_thread_init()」

mysql_affected_rows()
セクション23.7.5「C API データ構造」
セクション23.7.6「C API 関数の概要」
セクション13.2.1「CALL 構文」
セクション13.2.5「INSERT 構文」
セクション23.7.7.1「mysql_affected_rows()」
セクション23.7.7.46「mysql_next_result()」
セクション23.7.7.48「mysql_num_rows()」
セクション23.7.11.1「mysql_stmt_affected_rows()」
セクション23.7.7.72「mysql_use_result()」
セクション13.2.8「REPLACE 構文」
セクション23.7.15.2「クエリーからどんな結果を得ることができるか」
セクション12.14「情報関数」

mysql_autocommit()
セクション23.7.6「C API 関数の概要」

mysql_change_user()
セクション23.7.6「C API 関数の概要」
セクション23.7.7.3「mysql_change_user()」

mysql_character_set_name()
セクション23.7.6「C API 関数の概要」

mysql_client_find_plugin()
セクション23.7.6「C API 関数の概要」

mysql_client_register_plugin()
セクション23.7.6「C API 関数の概要」

mysql_close()
セクション23.7.6「C API 関数の概要」
セクション23.7.7.5「mysql_close()」
セクション23.7.7.6「mysql_commit()」
セクション23.7.7.7「mysql_connect()」
セクション23.7.7.36「mysql_init()」
セクション23.7.7.58「mysql_rollback()」
セクションB.5.2.11「通信エラーおよび中止された接続」

mysql_commit()
セクション23.7.6「C API 関数の概要」

mysql_connect()
セクション23.7.4.2「C API スレッドクライアントプログラムの作成」
セクション23.7.6「C API 関数の概要」
セクション23.7.12.1「my_init()」
セクション23.7.7.5「mysql_close()」
セクション23.7.7.7「mysql_connect()」
セクション23.7.7.49「mysql_options()」
セクション23.7.12.3「mysql_thread_init()」

mysql_create_db()
セクション23.7.6「C API 関数の概要」

mysql_data_seek()
セクション23.7.6「C API 関数の概要」
セクション23.7.7.9「mysql_data_seek()」
セクション23.7.7.59「mysql_row_seek()」
セクション23.7.7.72「mysql_use_result()」

mysql_debug()
セクション23.7.6「C API 関数の概要」
セクション23.7.7.10「mysql_debug()」

mysql_drop_db()
セクション23.7.6「C API 関数の概要」

mysql_dump_debug_info()
セクション23.7.6「C API 関数の概要」

mysql_eof()
セクション23.7.6「C API 関数の概要」
セクション23.7.7.13「mysql_eof()」

mysql_errno()
セクション23.7.6「C API 関数の概要」
セクション23.7.7「C API 関数の説明」
セクション23.7.14.1「mysql_client_find_plugin()」
セクション23.7.14.2「mysql_client_register_plugin()」
セクション23.7.7.7「mysql_connect()」
セクション23.7.7.13「mysql_eof()」
セクション23.7.7.14「mysql_errno()」
セクション23.7.7.22「mysql_field_count()」
セクション23.7.14.3「mysql_load_plugin()」
セクション23.7.7.47「mysql_num_fields()」
セクション23.7.15.1「mysql_query() が成功を返したあとに mysql_store_result() が NULL を返すことがあるのはなぜか」
セクション23.7.7.67「mysql_sqlstate()」
セクション23.7.7.70「mysql_store_result()」
セクション23.7.7.72「mysql_use_result()」
セクションB.2「エラー値のタイプ」
シグナルの条件情報項目
セクション24.2.4.8「監査プラグインの作成」
セクション6.3.12.3「監査ログファイル」

mysql_error()
セクション23.7.6「C API 関数の概要」
セクション23.7.7「C API 関数の説明」
セクション23.7.14.1「mysql_client_find_plugin()」
セクション23.7.14.2「mysql_client_register_plugin()」
セクション23.7.7.7「mysql_connect()」
セクション23.7.7.13「mysql_eof()」
セクション23.7.7.15「mysql_error()」
セクション23.7.14.3「mysql_load_plugin()」
セクション23.7.15.1「mysql_query() が成功を返したあとに mysql_store_result() が NULL を返すことがあるのはなぜか」
セクション23.7.7.70「mysql_store_result()」
セクション23.7.7.72「mysql_use_result()」
セクションB.2「エラー値のタイプ」

シグナルの条件情報項目
セクション24.2.4.8「監査プラグインの作成」

mysql_escape_string()

セクション23.7.6「C API 関数の概要」
セクション23.7.7.16「mysql_escape_string()」
セクション6.1.7「クライアントプログラミングのセキュリティーガイドライン」

mysql_fetch_field()

セクション23.7.5「C API データ構造」
セクション23.7.6「C API 関数の概要」
セクション23.7.7.17「mysql_fetch_field()」
セクション23.7.7.23「mysql_field_seek()」
セクション23.7.7.24「mysql_field_tell()」
セクション23.7.11.23「mysql_stmt_result_metadata()」

mysql_fetch_field_direct()

セクション23.7.6「C API 関数の概要」
セクション23.7.11.23「mysql_stmt_result_metadata()」

mysql_fetch_fields()

セクション23.7.6「C API 関数の概要」
セクション23.7.11.23「mysql_stmt_result_metadata()」

mysql_fetch_lengths()

セクション23.7.6「C API 関数の概要」
セクション23.7.7.20「mysql_fetch_lengths()」
セクション23.7.7.21「mysql_fetch_row()」

mysql_fetch_row()

セクション23.7.5「C API データ構造」
セクション23.7.6「C API 関数の概要」
セクション15.8.1「FEDERATED ストレージエンジンの概要」
セクション23.7.7.13「mysql_eof()」
セクション23.7.7.14「mysql_errno()」
セクション23.7.7.20「mysql_fetch_lengths()」
セクション23.7.7.21「mysql_fetch_row()」
セクション23.7.7.60「mysql_row_tell()」
セクション23.7.7.70「mysql_store_result()」
セクション23.7.7.72「mysql_use_result()」
セクション23.7.15.2「クエリーからどんな結果を得ることができるか」

mysql_field_count()

セクション23.7.6「C API 関数の概要」
セクション23.7.7.22「mysql_field_count()」
セクション23.7.7.47「mysql_num_fields()」
セクション23.7.7.52「mysql_query()」
セクション23.7.15.1「mysql_query() が成功を返したあとに mysql_store_result() が NULL を返すことがあるのはなぜか」
セクション23.7.7.55「mysql_real_query()」
セクション23.7.11.23「mysql_stmt_result_metadata()」
セクション23.7.7.70「mysql_store_result()」

mysql_field_seek()

セクション23.7.5「C API データ構造」
セクション23.7.6「C API 関数の概要」
セクション23.7.7.17「mysql_fetch_field()」
セクション23.7.7.24「mysql_field_tell()」
セクション23.7.11.23「mysql_stmt_result_metadata()」

mysql_field_tell()

セクション23.7.6「C API 関数の概要」
セクション23.7.11.23「mysql_stmt_result_metadata()」

mysql_free_result()

セクション23.7.10「C API プリペアドステートメント関数の概要」
セクション23.7.6「C API 関数の概要」
セクション23.7.7.25「mysql_free_result()」
セクション23.7.7.41「mysql_list_dbs()」
セクション23.7.7.42「mysql_list_fields()」
セクション23.7.7.43「mysql_list_processes()」
セクション23.7.7.44「mysql_list_tables()」
セクション23.7.7.46「mysql_next_result()」
セクション23.7.11.23「mysql_stmt_result_metadata()」
セクション23.7.7.70「mysql_store_result()」
セクション23.7.7.72「mysql_use_result()」
セクションB.5.2.14「コマンドは同期されていません」

mysql_get_character_set_info()

セクション23.7.6「C API 関数の概要」
セクション10.4.2「照合順序 ID の選択」

mysql_get_client_info()

セクション23.7.4.4「C API サーバーおよびクライアントライブラリのバージョン」
セクション23.7.6「C API 関数の概要」
セクション23.7.7.7「mysql_connect()」

mysql_get_client_version()

セクション23.7.4.4「C API サーバーおよびクライアントライブラリのバージョン」
セクション23.7.6「C API 関数の概要」

mysql_get_host_info()

セクション23.7.6「C API 関数の概要」

mysql_get_proto_info()

セクション23.7.6「C API 関数の概要」

mysql_get_server_info()

セクション23.7.4.4「C API サーバーおよびクライアントライブラリのバージョン」
セクション23.7.6「C API 関数の概要」

mysql_get_server_version()

セクション23.7.4.4「C API サーバーおよびクライアントライブラリのバージョン」
セクション23.7.6「C API 関数の概要」

mysql_get_ssl_cipher()

セクション23.7.6「C API 関数の概要」
セクション23.7.7.33「mysql_get_ssl_cipher()」
セクション6.3.10.3「SSL 接続の使用」

mysql_hex_string()

セクション23.7.6「C API 関数の概要」
セクション23.7.7.34「mysql_hex_string()」

mysql_info()

セクション13.1.7「ALTER TABLE 構文」
セクション23.7.6「C API 関数の概要」
セクション13.2.5.2「INSERT DELAYED 構文」
セクション13.2.5「INSERT 構文」
セクション13.2.6「LOAD DATA INFILE 構文」
セクション23.7.7.35「mysql_info()」
セクション23.7.7.49「mysql_options()」
セクション1.7.3.1「PRIMARY KEY および UNIQUE インデックス制約」
セクション13.2.11「UPDATE 構文」
セクション23.7.15.2「クエリーからどんな結果を得ることができるか」

mysql_init()

セクション23.7.4.2「C API スレッドクライアントプログラムの作成」
セクション23.7.6「C API 関数の概要」
セクション23.7.12.1「my_init()」
セクション23.7.7.5「mysql_close()」
セクション23.7.7.33「mysql_get_ssl_cipher()」
セクション23.7.7.36「mysql_init()」
セクション23.7.7.40「mysql_library_init()」
セクション23.7.7.49「mysql_options()」
セクション23.7.7.53「mysql_real_connect()」
セクション23.7.7.68「mysql_ssl_set()」
セクション23.7.12.3「mysql_thread_init()」

mysql_insert_id()

セクション3.6.9「AUTO_INCREMENT の使用」
セクション23.7.5「C API データ構造」
セクション23.7.6「C API 関数の概要」
セクション13.1.17「CREATE TABLE 構文」
セクション13.2.5「INSERT 構文」
セクション23.7.7.37「mysql_insert_id()」
セクション23.7.15.2「クエリーからどんな結果を得ることができるか」
セクション5.1.4「サーバーシステム変数」
セクション1.7.2.3「トランザクションおよびアトミック操作の違い」
セクション12.14「情報関数」
セクション23.7.15.3「最後に挿入された行の一意の ID を取得する方法」

mysql_kill()

セクション23.7.6「C API 関数の概要」
セクション23.7.7.38「mysql_kill()」
セクション23.7.7.71「mysql_thread_id()」
セクション23.7.16「自動再接続動作の制御」

mysql_library_end()

セクション23.7.13「C API 組み込みサーバー関数の説明」
セクション23.7.6「C API 関数の概要」
セクション23.7.7.39「mysql_library_end()」
セクション23.7.7.40「mysql_library_init()」
セクション23.7.13.2「mysql_server_end()」
セクション23.6「組み込み MySQL サーバーライブラリ libmysqld」

mysql_library_init()

セクション23.7.4.2「C API スレッドクライアントプログラムの作成」
セクション23.7.13「C API 組み込みサーバー関数の説明」
セクション23.7.6「C API 関数の概要」
セクション23.7.12.1「my_init()」
セクション23.7.7.36「mysql_init()」
セクション23.7.7.40「mysql_library_init()」
セクション23.7.14.3「mysql_load_plugin()」
セクション23.7.13.1「mysql_server_init()」
セクション23.7.12.3「mysql_thread_init()」
セクション23.6「組み込み MySQL サーバーライブラリ libmysqld」
セクション23.6.3「組み込みサーバーとオプション」

mysql_list_dbs()

セクション23.7.6「C API 関数の概要」
セクション23.7.7.25「mysql_free_result()」
セクション23.7.7.41「mysql_list_dbs()」

mysql_list_fields()

セクション23.7.5「C API データ構造」
セクション23.7.6「C API 関数の概要」
セクション23.7.7.42「mysql_list_fields()」

mysql_list_processes()

セクション23.7.6「C API 関数の概要」

mysql_list_tables()

セクション23.7.6「C API 関数の概要」
セクション23.7.7.44「mysql_list_tables()」

mysql_load_plugin()

セクション23.7.6「C API 関数の概要」
セクション23.7.14.3「mysql_load_plugin()」
セクション23.7.14.4「mysql_load_plugin_v()」
クライアントプラグインディスクリプタ

mysql_load_plugin_v()

セクション23.7.6「C API 関数の概要」
セクション23.7.14.3「mysql_load_plugin()」

mysql_more_results()

セクション23.7.6「C API 関数の概要」
セクション23.7.7.45「mysql_more_results()」
セクション23.7.7.46「mysql_next_result()」
セクション23.7.11.17「mysql_stmt_next_result()」
セクション23.7.17「複数ステートメント実行の C API サポート」

mysql_next_result()

セクション23.7.6「C API 関数の概要」
セクション13.2.1「CALL 構文」
セクション23.7.7.45「mysql_more_results()」
セクション23.7.7.46「mysql_next_result()」
セクション23.7.7.53「mysql_real_connect()」
セクション23.7.7.65「mysql_set_server_option()」
セクション23.7.7.70「mysql_store_result()」
セクション23.7.17「複数ステートメント実行の C API サポート」

mysql_num_fields()

セクション23.7.6「C API 関数の概要」
セクション23.7.7.18「mysql_fetch_field_direct()」

セクション23.7.7.21「mysql_fetch_row()」
セクション23.7.11.23「mysql_stmt_result_metadata()」

mysql_num_rows()

セクション23.7.5「C API データ構造」
セクション23.7.6「C API 関数の概要」
セクション23.7.7.1「mysql_affected_rows()」
セクション23.7.7.9「mysql_data_seek()」
セクション23.7.7.48「mysql_num_rows()」
セクション23.7.7.70「mysql_store_result()」
セクション23.7.7.72「mysql_use_result()」
セクション23.7.15.2「クエリーからどんな結果を得ることができるか」

mysql_options()

セクション23.7.14「C API クライアントプラグイン関数」
セクション23.7.9「C API プリペアドステートメントデータ構造」
セクション23.7.6「C API 関数の概要」
セクション6.1.6「LOAD DATA LOCAL のセキュリティーの問題」
セクションB.5.2.9「MySQL サーバーが存在しなくなりました」
セクション23.7.7.49「mysql_options()」
セクション23.7.7.50「mysql_options4()」
セクション23.7.7.51「mysql_ping()」
セクション23.7.7.53「mysql_real_connect()」
セクション23.7.11.11「mysql_stmt_fetch()」
セクション22.9.8.1「session_account_connect_attrs テーブル」
セクション22.9.8.2「session_connect_attrs テーブル」
クライアントプラグインディスクリプタ
セクション6.3.8.7「クライアント側のクリアテキスト認証プラグイン」
セクション6.3.6「パスワードの期限切れとサンドボックスモード」
セクション22.9.8「パフォーマンススキーマ接続属性テーブル」
セクション24.2.2「プラグイン API のコンポーネント」
セクション6.3.1「ユーザー名とパスワード」
セクション10.1.4「接続文字セットおよび照合順序」
セクション23.7.16「自動再接続動作の制御」
セクション5.3.4「複数サーバー環境でのクライアントプログラムの使用」
認証プラグインの使用

mysql_options4()

セクション23.7.6「C API 関数の概要」
セクション23.7.7.49「mysql_options()」
セクション23.7.7.50「mysql_options4()」
セクション22.9.8.1「session_account_connect_attrs テーブル」
セクション22.9.8.2「session_connect_attrs テーブル」
セクション22.9.8「パフォーマンススキーマ接続属性テーブル」

mysql_ping()

セクション23.7.6「C API 関数の概要」
セクションB.5.2.9「MySQL サーバーが存在しなくなりました」
セクション23.7.7.51「mysql_ping()」
セクション23.7.7.71「mysql_thread_id()」
セクション23.7.16「自動再接続動作の制御」

mysql_plugin_options()

セクション23.7.6「C API 関数の概要」

mysql_query()

セクション23.7.4.2「C API スレッドクライアントプログラムの作成」
セクション23.7.6「C API 関数の概要」
セクション13.2.1「CALL 構文」
セクション23.7.7.1「mysql_affected_rows()」
セクション23.7.7.8「mysql_create_db()」
セクション23.7.7.11「mysql_drop_db()」
セクション23.7.7.17「mysql_fetch_field()」
セクション23.7.7.38「mysql_kill()」
セクション23.7.7.46「mysql_next_result()」
セクション23.7.7.52「mysql_query()」
セクション23.7.15.1「mysql_query() が成功を返したあとに mysql_store_result() が NULL を返すことがあるのはなぜか」
セクション23.7.7.53「mysql_real_connect()」
セクション23.7.7.55「mysql_real_query()」
セクション23.7.7.57「mysql_reload()」
セクション23.7.7.64「mysql_set_local_infile_handler()」
セクション23.7.7.65「mysql_set_server_option()」
セクション23.7.7.70「mysql_store_result()」
セクション23.7.7.72「mysql_use_result()」
セクション23.7.15.3「最後に挿入された行の一意の ID を取得する方法」
セクション23.7.17「複数ステートメント実行の C API サポート」

mysql_real_connect()

セクション23.7.6「C API 関数の概要」
セクション13.2.1「CALL 構文」
セクション13.2.5.3「INSERT ... ON DUPLICATE KEY UPDATE 構文」
セクション13.2.5「INSERT 構文」
セクション23.7.7.1「mysql_affected_rows()」
セクション23.7.7.3「mysql_change_user()」
セクション23.7.7.7「mysql_connect()」
セクション23.7.7.36「mysql_init()」
セクション23.7.7.46「mysql_next_result()」
セクション23.7.7.49「mysql_options()」
セクション23.7.7.53「mysql_real_connect()」
セクション23.7.7.65「mysql_set_server_option()」
セクション23.7.7.67「mysql_sqlstate()」
セクション23.7.7.68「mysql_ssl_set()」
セクション6.3.10.3「SSL 接続の使用」
セクション5.1.4「サーバーシステム変数」
セクション20.2.1「ストアドルーチンの構文」
セクション6.3.6「パスワードの期限切れとサンドボックスモード」
セクション12.14「情報関数」
セクション13.5「準備済みステートメントのための SQL 構文」
セクション5.3.4「複数サーバー環境でのクライアントプログラムの使用」
セクション23.7.17「複数ステートメント実行の C API サポート」
第12章「関数と演算子」

mysql_real_escape_string()

セクション23.7.6「C API 関数の概要」
セクション23.7.7.16「mysql_escape_string()」
セクション23.7.7.54「mysql_real_escape_string()」

セクション23.7.7.62「mysql_set_character_set()」
セクション6.1.7「クライアントプログラミングのセキュリティーガイドライン」
セクション9.1.1「文字列リテラル」
セクション11.5.3.3「空間カラムへのデータ移入」

mysql_real_query()

セクション23.7.6「C API 関数の概要」
セクション13.2.1「CALL 構文」
セクション15.8.1「FEDERATED ストレージエンジンの概要」
セクション23.7.7.1「mysql_affected_rows()」
セクション23.7.7.46「mysql_next_result()」
セクション23.7.7.52「mysql_query()」
セクション23.7.7.53「mysql_real_connect()」
セクション23.7.7.55「mysql_real_query()」
セクション23.7.7.65「mysql_set_server_option()」
セクション23.7.7.70「mysql_store_result()」
セクション23.7.7.72「mysql_use_result()」
セクション23.7.17「複数ステートメント実行の C API サポート」

mysql_refresh()

セクション23.7.6「C API 関数の概要」

mysql_reload()

セクション23.7.6「C API 関数の概要」

mysql_rollback()

セクション23.7.6「C API 関数の概要」

mysql_row_seek()

セクション23.7.6「C API 関数の概要」
セクション23.7.7.59「mysql_row_seek()」
セクション23.7.7.60「mysql_row_tell()」
セクション23.7.7.70「mysql_store_result()」
セクション23.7.7.72「mysql_use_result()」

mysql_row_tell()

セクション23.7.6「C API 関数の概要」
セクション23.7.7.59「mysql_row_seek()」
セクション23.7.7.60「mysql_row_tell()」
セクション23.7.7.70「mysql_store_result()」
セクション23.7.7.72「mysql_use_result()」

mysql_select_db()

セクション23.7.6「C API 関数の概要」
セクション23.7.7.61「mysql_select_db()」

mysql_server_end()

セクション23.7.6「C API 関数の概要」
セクション23.7.13.2「mysql_server_end()」

mysql_server_init()

セクション23.7.6「C API 関数の概要」
セクション23.7.12.1「my_init()」
セクション23.7.13.1「mysql_server_init()」
セクション23.7.12.3「mysql_thread_init()」

mysql_set_character_set()

セクション23.7.6「C API 関数の概要」
セクション23.7.7.26「mysql_get_character_set_info()」
セクション23.7.7.54「mysql_real_escape_string()」

mysql_set_local_infile_default()

セクション23.7.6「C API 関数の概要」

mysql_set_local_infile_handler()

セクション23.7.6「C API 関数の概要」
セクション23.7.7.63「mysql_set_local_infile_default()」
セクション23.7.7.64「mysql_set_local_infile_handler()」

mysql_set_server_option()

セクション23.7.6「C API 関数の概要」
セクション23.7.7.65「mysql_set_server_option()」
セクション23.7.17「複数ステートメント実行の C API サポート」

mysql_shutdown()

セクション23.7.6「C API 関数の概要」
セクション23.7.7.66「mysql_shutdown()」

mysql_sqlstate()

セクション23.7.6「C API 関数の概要」
セクション23.7.7.14「mysql_errno()」
セクション23.7.7.67「mysql_sqlstate()」
セクションB.2「エラー値のタイプ」
シグナルの条件情報項目

mysql_ssl_set()

セクション23.7.6「C API 関数の概要」
セクション23.7.7.53「mysql_real_connect()」
セクション23.7.7.68「mysql_ssl_set()」
セクション6.3.10.3「SSL 接続の使用」

mysql_stat()

セクション23.7.6「C API 関数の概要」

mysql_stmt_affected_rows()

セクション23.7.10「C API プリペアドステートメント関数の概要」
セクション23.7.11.1「mysql_stmt_affected_rows()」
セクション23.7.11.10「mysql_stmt_execute()」
セクション23.7.11.17「mysql_stmt_next_result()」
セクション23.7.11.18「mysql_stmt_num_rows()」

mysql_stmt_attr_get()

セクション23.7.10「C API プリペアドステートメント関数の概要」

mysql_stmt_attr_set()

セクション23.7.5「C API データ構造」
セクション23.7.9.2「C API プリペアドステートメント型変換」
セクション23.7.10「C API プリペアドステートメント関数の概要」
セクション23.7.11.3「mysql_stmt_attr_set()」
セクション23.7.11.10「mysql_stmt_execute()」
セクション23.7.11.11「mysql_stmt_fetch()」
セクション23.7.11.28「mysql_stmt_store_result()」
セクションD.3「サーバー側のカーソルの制約」

mysql_stmt_bind_param()

セクション23.7.19「C API プリペアドステートメントの日時値の処理」
セクション23.7.9「C API プリペアドステートメントデータ構造」
セクション23.7.10「C API プリペアドステートメント関数の概要」
セクション23.7.11.4「mysql_stmt_bind_param()」
セクション23.7.11.10「mysql_stmt_execute()」
セクション23.7.11.21「mysql_stmt_prepare()」
セクション23.7.11.26「mysql_stmt_send_long_data()」

mysql_stmt_bind_result()

セクション23.7.19「C API プリペアドステートメントの日時値の処理」
セクション23.7.9「C API プリペアドステートメントデータ構造」
セクション23.7.10「C API プリペアドステートメント関数の概要」
セクション23.7.11.5「mysql_stmt_bind_result()」
セクション23.7.11.11「mysql_stmt_fetch()」
セクション23.7.11.12「mysql_stmt_fetch_column()」
セクション23.7.11.17「mysql_stmt_next_result()」
セクション23.7.11.28「mysql_stmt_store_result()」

mysql_stmt_close()

セクション23.7.9「C API プリペアドステートメントデータ構造」
セクション23.7.10「C API プリペアドステートメント関数の概要」
セクション23.7.11.6「mysql_stmt_close()」
セクション23.7.11.15「mysql_stmt_init()」

mysql_stmt_data_seek()

セクション23.7.10「C API プリペアドステートメント関数の概要」
セクション23.7.11.7「mysql_stmt_data_seek()」
セクション23.7.11.24「mysql_stmt_row_seek()」
セクション23.7.11.28「mysql_stmt_store_result()」

mysql_stmt_errno()

セクション23.7.10「C API プリペアドステートメント関数の概要」
セクション23.7.11.8「mysql_stmt_errno()」
セクション23.7.11.11「mysql_stmt_fetch()」
セクションB.2「エラー値のタイプ」

mysql_stmt_error()

セクション23.7.10「C API プリペアドステートメント関数の概要」
セクション23.7.11.9「mysql_stmt_error()」
セクション23.7.11.11「mysql_stmt_fetch()」
セクション23.7.11.21「mysql_stmt_prepare()」
セクションB.2「エラー値のタイプ」

mysql_stmt_execute()

セクション23.7.19「C API プリペアドステートメントの日時値の処理」
セクション23.7.9「C API プリペアドステートメントデータ構造」
セクション23.7.9.2「C API プリペアドステートメント型変換」
セクション23.7.10「C API プリペアドステートメント関数の概要」
セクション23.7.11.1「mysql_stmt_affected_rows()」
セクション23.7.11.3「mysql_stmt_attr_set()」
セクション23.7.11.10「mysql_stmt_execute()」
セクション23.7.11.11「mysql_stmt_fetch()」
セクション23.7.11.17「mysql_stmt_next_result()」
セクション23.7.11.26「mysql_stmt_send_long_data()」
セクション23.7.11.28「mysql_stmt_store_result()」
セクション8.9.3.1「クエリーキャッシュの動作」

mysql_stmt_fetch()

セクション23.7.9「C API プリペアドステートメントデータ構造」
セクション23.7.9.2「C API プリペアドステートメント型変換」
セクション23.7.10「C API プリペアドステートメント関数の概要」
セクション23.7.11.5「mysql_stmt_bind_result()」
セクション23.7.11.10「mysql_stmt_execute()」
セクション23.7.11.11「mysql_stmt_fetch()」
セクション23.7.11.23「mysql_stmt_result_metadata()」
セクション23.7.11.25「mysql_stmt_row_tell()」
セクション23.7.11.28「mysql_stmt_store_result()」

mysql_stmt_fetch_column()

セクション23.7.10「C API プリペアドステートメント関数の概要」
セクション23.7.11.11「mysql_stmt_fetch()」

mysql_stmt_field_count()

セクション23.7.10「C API プリペアドステートメント関数の概要」
セクション23.7.11.13「mysql_stmt_field_count()」

mysql_stmt_free_result()

セクション23.7.10「C API プリペアドステートメント関数の概要」
セクション23.7.11.3「mysql_stmt_attr_set()」
セクション23.7.11.14「mysql_stmt_free_result()」
セクション23.7.11.17「mysql_stmt_next_result()」

mysql_stmt_init()

セクション23.7.8「C API プリペアドステートメント」
セクション23.7.9「C API プリペアドステートメントデータ構造」
セクション23.7.10「C API プリペアドステートメント関数の概要」
セクション23.7.11「C API プリペアドステートメント関数の説明」
セクション23.7.11.10「mysql_stmt_execute()」
セクション23.7.11.21「mysql_stmt_prepare()」

mysql_stmt_insert_id()

セクション23.7.10「C API プリペアドステートメント関数の概要」

mysql_stmt_next_result()

セクション23.7.20「C API のプリペアド CALL ステートメントのサポート」
セクション23.7.10「C API プリペアドステートメント関数の概要」

セクション13.2.1「CALL 構文」
セクション23.7.11.17「mysql_stmt_next_result()」

mysql_stmt_num_rows()

セクション23.7.10「C API プリペアドステートメント関数の概要」
セクション23.7.11.7「mysql_stmt_data_seek()」
セクション23.7.11.18「mysql_stmt_num_rows()」

mysql_stmt_param_count()

セクション23.7.10「C API プリペアドステートメント関数の概要」
セクション23.7.11.10「mysql_stmt_execute()」

mysql_stmt_param_metadata()

セクション23.7.10「C API プリペアドステートメント関数の概要」

mysql_stmt_prepare()

セクション23.7.19「C API プリペアドステートメントの日時値の処理」
セクション23.7.9「C API プリペアドステートメントデータ構造」
セクション23.7.10「C API プリペアドステートメント関数の概要」
セクション23.7.11.4「mysql_stmt_bind_param()」
セクション23.7.11.10「mysql_stmt_execute()」
セクション23.7.11.13「mysql_stmt_field_count()」
セクション23.7.11.21「mysql_stmt_prepare()」
セクション23.7.11.22「mysql_stmt_reset()」
セクション23.7.11.23「mysql_stmt_result_metadata()」
セクション8.9.3.1「クエリーキャッシュの動作」
セクション8.9.4「プリペアドステートメントおよびストアドプログラムのキャッシュ」
セクション13.5「準備済みステートメントのための SQL 構文」

mysql_stmt_reset()

セクション23.7.10「C API プリペアドステートメント関数の概要」
セクション23.7.11.3「mysql_stmt_attr_set()」
セクション23.7.11.26「mysql_stmt_send_long_data()」

mysql_stmt_result_metadata()

セクション23.7.9.2「C API プリペアドステートメント型変換」
セクション23.7.10「C API プリペアドステートメント関数の概要」
セクション23.7.11.11「mysql_stmt_fetch()」
セクション23.7.11.23「mysql_stmt_result_metadata()」
セクション23.7.11.28「mysql_stmt_store_result()」

mysql_stmt_row_seek()

セクション23.7.10「C API プリペアドステートメント関数の概要」
セクション23.7.11.24「mysql_stmt_row_seek()」
セクション23.7.11.25「mysql_stmt_row_tell()」
セクション23.7.11.28「mysql_stmt_store_result()」

mysql_stmt_row_tell()

セクション23.7.10「C API プリペアドステートメント関数の概要」
セクション23.7.11.24「mysql_stmt_row_seek()」
セクション23.7.11.25「mysql_stmt_row_tell()」
セクション23.7.11.28「mysql_stmt_store_result()」

mysql_stmt_send_long_data()

セクション23.7.10「C API プリペアドステートメント関数の概要」
セクション23.7.11.22「mysql_stmt_reset()」
セクション23.7.11.26「mysql_stmt_send_long_data()」
セクション5.1.4「サーバーシステム変数」

mysql_stmt_sqlstate()

セクション23.7.10「C API プリペアドステートメント関数の概要」
セクション23.7.11.27「mysql_stmt_sqlstate()」
セクションB.2「エラー値のタイプ」

mysql_stmt_store_result()

セクション23.7.5「C API データ構造」
セクション23.7.10「C API プリペアドステートメント関数の概要」
セクション23.7.11.3「mysql_stmt_attr_set()」
セクション23.7.11.7「mysql_stmt_data_seek()」
セクション23.7.11.11「mysql_stmt_fetch()」
セクション23.7.11.18「mysql_stmt_num_rows()」
セクション23.7.11.24「mysql_stmt_row_seek()」
セクション23.7.11.25「mysql_stmt_row_tell()」
セクション23.7.11.28「mysql_stmt_store_result()」

mysql_store_result()

セクション23.7.4.2「C API スレッドクライアントプログラムの作成」
セクション23.7.5「C API データ構造」
セクション23.7.6「C API 関数の概要」
セクション15.8.1「FEDERATED ストレージエンジンの概要」
セクション4.5.1「mysql — MySQL コマンド行ツール」
セクション23.7.7.1「mysql_affected_rows()」
セクション23.7.7.9「mysql_data_seek()」
セクション23.7.7.13「mysql_eof()」
セクション23.7.7.17「mysql_fetch_field()」
セクション23.7.7.21「mysql_fetch_row()」
セクション23.7.7.22「mysql_field_count()」
セクション23.7.7.25「mysql_free_result()」
セクション23.7.7.46「mysql_next_result()」
セクション23.7.7.47「mysql_num_fields()」
セクション23.7.7.48「mysql_num_rows()」
セクション23.7.15.1「mysql_query() が成功を返したあとに mysql_store_result() が NULL を返すことがあるのはなぜか」
セクション23.7.7.59「mysql_row_seek()」
セクション23.7.7.60「mysql_row_tell()」
セクション23.7.11.10「mysql_stmt_execute()」
セクション23.7.11.23「mysql_stmt_result_metadata()」
セクション23.7.7.70「mysql_store_result()」
セクション23.7.7.72「mysql_use_result()」
セクション23.7.15.2「クエリーからどんな結果を得ることができるか」
セクションB.5.2.14「コマンドは同期されていません」

mysql_thread_end()

セクション23.7.4.2「C API スレッドクライアントプログラムの作成」

セクション23.7.6「C API 関数の概要」
セクション23.7.12.2「mysql_thread_end()」
セクション23.6「組み込み MySQL サーバーライブラリ libmysqld」

mysql_thread_id()

セクション23.7.6「C API 関数の概要」
セクション23.7.7.51「mysql_ping()」
セクション23.7.7.71「mysql_thread_id()」
セクション23.7.16「自動再接続動作の制御」

mysql_thread_init()

セクション23.7.4.2「C API スレッドクライアントプログラムの作成」
セクション23.7.6「C API 関数の概要」
セクション23.7.12.1「my_init()」
セクション23.7.12.2「mysql_thread_end()」
セクション23.7.12.3「mysql_thread_init()」
セクション23.6「組み込み MySQL サーバーライブラリ libmysqld」

mysql_thread_safe()

セクション23.7.6「C API 関数の概要」

mysql_use_result()

セクション23.7.4.2「C API スレッドクライアントプログラムの作成」
セクション23.7.5「C API データ構造」
セクション23.7.6「C API 関数の概要」
セクション4.5.1「mysql — MySQL コマンド行ツール」
セクション23.7.7.9「mysql_data_seek()」
セクション23.7.7.13「mysql_eof()」
セクション23.7.7.21「mysql_fetch_row()」
セクション23.7.7.25「mysql_free_result()」
セクション23.7.7.46「mysql_next_result()」
セクション23.7.7.47「mysql_num_fields()」
セクション23.7.7.48「mysql_num_rows()」
セクション23.7.7.59「mysql_row_seek()」
セクション23.7.7.60「mysql_row_tell()」
セクション23.7.11.10「mysql_stmt_execute()」
セクション23.7.7.70「mysql_store_result()」
セクション23.7.7.72「mysql_use_result()」
セクション23.7.15.2「クエリーからどんな結果を得ることができるか」
セクションB.5.2.14「コマンドは同期されていません」
セクションB.5.2.8「メモリー不足」

mysql_warning_count()

セクション23.7.6「C API 関数の概要」
セクション23.7.7.46「mysql_next_result()」
セクション13.7.5.41「SHOW WARNINGS 構文」
セクションB.2「エラー値のタイプ」

コマンドの索引

シンボル | A | B | C | D | E | F | G | H | I | K | L | M | N | O | P | R | S | T | U | V | W | Y | Z

シンボル

[索引の先頭]

「サービス」
セクション18.2.3.4「Windows サービスとしての MySQL Cluster プロセスのインストール」

「システム環境設定...」
セクション2.4.5「MySQL Preference Pane のインストールと使用」

このスタイルのテキスト
セクション1.2「表記規則および構文規則」

サービスコントロールマネージャー
セクション2.3.5.7「Windows のサービスとして MySQL を起動する」

サービス管理マネージャー
セクション2.3「Microsoft Windows に MySQL をインストールする」

ターミナル
セクション2.4「OS X に MySQL をインストールする」

ディレクトリユーティリティー
セクション2.4.1「OS X への MySQL のインストールに関する一般的な注記」

A

[索引の先頭]

Access
セクション13.2.2「DELETE 構文」

addgroup
セクション18.2.2.1「Linux での MySQL Cluster バイナリリリースのインストール」
セクション2.2「一般的なバイナリを使用した MySQL の Unix/Linux へのインストール」

addr2line
セクション24.4.1.5「スタックトレースの使用」

adduser
セクション18.2.2.1「Linux での MySQL Cluster バイナリリリースのインストール」
セクション2.2「一般的なバイナリを使用した MySQL の Unix/Linux へのインストール」

ALL STATUS
セクション18.5.8「MySQL Cluster のシングルユーザーモード」

APF
セクション18.5.11.1「MySQL Cluster のセキュリティーとネットワーク上の問題」

apt-get
セクション16.6.3.3「C および C++ での libmemcached の使用」
セクション16.6.1「memcached のインストール」
セクション2.5.7「ネイティブソフトウェアリポジトリから MySQL を Linux にインストールする」
セクション2.2「一般的なバイナリを使用した MySQL の Unix/Linux へのインストール」

B

[索引の先頭]

bash
セクション4.2.1「MySQL プログラムの起動」
セクション2.4.1「OS X への MySQL のインストールに関する一般的な注記」
セクション6.1.2.1「パスワードセキュリティーのためのエンドユーザーガイドライン」
セクション17.1.4.3「レプリケーションスレーブのオプションと変数」
セクション4.2.10「環境変数の設定」
セクション1.2「表記規則および構文規則」

bison
セクション2.9.5「MySQL のコンパイルに関する問題」
セクション1.8.1「MySQL への貢献者」
セクション2.9「ソースから MySQL をインストールする」

bzr
セクション2.9.3「開発ソースツリーを使用して MySQL をインストールする」

C

[索引の先頭]

c++filt
セクション24.4.1.5「スタックトレースの使用」

cat
セクション4.5.1.1「mysql のオプション」

chkconfig
セクション18.2.2.1「Linux での MySQL Cluster バイナリリリースのインストール」
セクション2.10.1.2「MySQL を自動的に起動および停止する」
セクション2.5.7「ネイティブソフトウェアリポジトリから MySQL を Linux にインストールする」

chroot
セクション4.6.10「mysqlhotcopy — データベースバックアッププログラム」

CMake
セクション15.5「ARCHIVE ストレージエンジン」

セクション15.6「BLACKHOLE ストレージエンジン」
セクション5.4「DTrace を使用した mysqld のトレース」
セクション15.9「EXAMPLE ストレージエンジン」
セクション15.8「FEDERATED ストレージエンジン」
セクション18.2.2.3「Linux でのソースからの MySQL Cluster のビルド」
セクション6.1.6「LOAD DATA LOCAL のセキュリティーの問題」
第18章「MySQL Cluster NDB 7.3 および MySQL Cluster NDB 7.4」
セクション18.1.4.1「MySQL Cluster NDB 7.3 での MySQL Cluster の開発」
セクション18.5.4「MySQL Cluster での MySQL サーバーの使用法」
セクションB.5.4.2「MySQL が繰り返しクラッシュする場合の対処方法」
セクションB.5.4.5「MySQL の UNIX ソケットファイルを保護または変更する方法」
セクション2.9.5「MySQL のコンパイルに関する問題」
セクション1.3.2「MySQL の主な機能」
セクション2.9.4「MySQL ソース構成オプション」
セクション6.3.10.2「SSL を使用するための MySQL の構成」
セクション5.3.3「Unix 上での複数の MySQL インスタンスの実行」
セクション18.2.3.2「Windows でのソースからの MySQL Cluster のコンパイルとインストール」
セクション10.1.5「アプリケーションの文字セットおよび照合順序の構成」
セクション4.2.6「オプションファイルの使用」
セクション5.1.4「サーバーシステム変数」
セクション10.1.3.1「サーバー文字セットおよび照合順序」
セクション2.9「ソースから MySQL をインストールする」
セクション22.2.1「パフォーマンススキーマビルド構成」
セクション24.2.5「プラグインのための MySQL サービス」
セクション24.2.4.3「プラグインライブラリのコンパイルおよびインストール」
セクション24.3.2.5「ユーザー定義関数のコンパイルおよびインストール」
セクション10.3「文字セットの追加」
セクションB.5.2.17「文字セットを初期化できません」
セクション2.9.2「標準ソース配布を使用して MySQL をインストールする」
セクション2.12「環境変数」
セクション2.9.3「開発ソースツリーを使用して MySQL をインストールする」

cmake

セクション14.18.4.1「SASL による memcached インタフェースのパスワード保護」
セクション24.3.2.5「ユーザー定義関数のコンパイルおよびインストール」
セクション2.9.2「標準ソース配布を使用して MySQL をインストールする」

cmd

root のパスワードのリセット: Windows システム

cmd.exe

セクション4.2.1「MySQL プログラムの起動」
セクション1.2「表記規則および構文規則」

command.com

セクション4.2.1「MySQL プログラムの起動」
セクション1.2「表記規則および構文規則」

comp_err

セクション4.4.1「comp_err — MySQL エラーメッセージファイルのコンパイル」
セクション4.1「MySQL プログラムの概要」

configure

セクション16.6.3.3「C および C++ での libmemcached の使用」
セクション16.6.1「memcached のインストール」
セクション18.4.27「MySQL Cluster プログラムに共通するオプション — MySQL Cluster プログラムに共通するオプション」
セクション16.6.3.6「PHP での MySQL と memcached の使用」
セクション1.2「表記規則および構文規則」
セクション1.6「質問またはバグをレポートする方法」

copy

セクション17.1.1.6「ローデータファイルを使用したデータスナップショットの作成」

coreadm

セクション2.7「Solaris および OpenSolaris に MySQL をインストールする」
セクション5.1.3「サーバーコマンドオプション」

cp

セクション17.3.1.2「スレーブからローデータをバックアップする」
セクション7.1「バックアップとリカバリの種類」
セクション17.1.1.6「ローデータファイルを使用したデータスナップショットの作成」
セクション17.1.1.9「既存のレプリケーション環境への追加スレーブの導入」

crash-me

セクション8.12.2「MySQL ベンチマークスイート」

cron

セクションB.5.2.2「[ローカルの] MySQL サーバーに接続できません」
セクション13.7.2.2「CHECK TABLE 構文」
セクション7.6.5「MyISAM テーブル保守スケジュールのセットアップ」
セクション15.2.1「MyISAM 起動オプション」
セクション5.2.7「サーバーログの保守」
セクション3.5「バッチモードでの MySQL の使用」

csh

セクション4.2.1「MySQL プログラムの起動」
セクション4.2.10「環境変数の設定」
セクション1.2「表記規則および構文規則」

D

[索引の先頭]

date
セクション4.3.2「mysqld_safe — MySQL サーバー起動スクリプト」

dd
セクション16.4.1「EC2 AMI での MySQL のセットアップ」

delete
セクション14.18.7「InnoDB memcached プラグインの内部構造」

df
セクションB.5.1「問題の原因を判別する方法」

drwtsn32.exe
セクション24.4.1.3「pdb を使用した Windows のクラッシュダンプの作成」

dump
セクション17.1.1.6「ローデータファイルを使用したデータスナップショットの作成」

E

[索引の先頭]

emerge
セクション16.6.1「memcached のインストール」
セクション2.5.7「ネイティブソフトウェアリポジトリから MySQL を Linux にインストールする」

EXIT SINGLE USER MODE
セクション18.5.8「MySQL Cluster のシングルユーザーモード」

F

[索引の先頭]

flush
セクション14.18.7「InnoDB memcached プラグインの内部構造」

G

[索引の先頭]

gcc
セクション23.6.1「libmysqld によるプログラムのコンパイル」
セクション1.8.4「MySQL の作成に使用されたツール」
セクション2.13.3「Perl DBI/DBD インタフェース使用の問題」
セクション24.3.2.5「ユーザー定義関数のコンパイルおよびインストール」

gdb
セクション24.4.1.4「gdb での mysqld のデバッグ」
セクションB.5.4.2「MySQL が繰り返しクラッシュする場合の対処方法」
セクション1.8.4「MySQL の作成に使用されたツール」
セクション24.4.1.1「デバッグのための MySQL のコンパイル」

get
セクション14.18.7「InnoDB memcached プラグインの内部構造」

git branch
セクション2.9.3「開発ソースツリーを使用して MySQL をインストールする」

git checkout
セクション2.9.3「開発ソースツリーを使用して MySQL をインストールする」

git checkout 5.7
セクション2.9.3「開発ソースツリーを使用して MySQL をインストールする」

gmake
セクション2.8「FreeBSD に MySQL をインストールする」
セクション2.9「ソースから MySQL をインストールする」
セクション2.9.2「標準ソース配布を使用して MySQL をインストールする」

GnuPG
セクション2.1.4.2「GnuPG を使用した署名確認」

gnutar
セクション2.9「ソースから MySQL をインストールする」
セクション2.2「一般的なバイナリを使用した MySQL の Unix/Linux へのインストール」

gogoc
セクション5.1.9.5「ブローカからの IPv6 アドレスの入手」

gpg
セクション2.1.4.2「GnuPG を使用した署名確認」

grep
セクション4.6.9「mysqldumpslow — スロークエリーログファイルの要約」
セクション3.3.4.7「パターンマッチング」

groupadd
セクション18.2.2.1「Linux での MySQL Cluster バイナリリリースのインストール」
セクション2.5.5「RPM パッケージを使用して MySQL を Linux にインストールする」
セクション2.7「Solaris および OpenSolaris に MySQL をインストールする」
セクション2.2「一般的なバイナリを使用した MySQL の Unix/Linux へのインストール」

gtar
セクション2.7「Solaris および OpenSolaris に MySQL をインストールする」
セクション2.9「ソースから MySQL をインストールする」
セクション2.2「一般的なバイナリを使用した MySQL の Unix/Linux へのインストール」

gunzip
セクション2.2「一般的なバイナリを使用した MySQL の Unix/Linux へのインストール」
セクション2.9.2「標準ソース配布を使用して MySQL をインストールする」

gzip
セクション18.3.2.6「MySQL Cluster データノードの定義」
セクション2.4「OS X に MySQL をインストールする」
セクション1.6「質問またはバグをレポートする方法」

H

[索引の先頭]

hdparm
セクション14.12「InnoDB の起動オプションおよびシステム変数」

help contents
セクション4.5.1.4「mysql サーバー側のヘルプ」

hostname
セクションB.5.2.2「[ローカルの] MySQL サーバーに接続できません」

I

[索引の先頭]

icc
セクション24.4「MySQL のデバッグおよび移植」
セクション2.1.6「コンパイラ固有のビルドの特徴」

ifconfig
セクション5.1.9.1「IPv6 用のシステムサポートの確認」

innochecksum
セクション13.7.2.2「CHECK TABLE 構文」
セクション4.6.1「innochecksum — オフライン InnoDB ファイルチェックサムユーティリティー」
セクション1.4「MySQL 5.6 の新機能」
セクション4.1「MySQL プログラムの概要」
MySQL 用語集

install.rb
セクション16.6.3.7「Ruby での MySQL と memcached の使用」

iptables
セクション18.5.11.1「MySQL Cluster のセキュリティーとネットワーク上の問題」

K

[索引の先頭]

kill
セクションB.5.2.2「[ローカルの] MySQL サーバーに接続できません」
セクション18.5.5「MySQL Cluster のローリング再起動の実行」
セクション18.4.1「ndbd — MySQL Cluster データノードデーモン」
セクションD.6「XA トランザクションの制約」

ksh
セクション4.2.1「MySQL プログラムの起動」
セクション4.2.10「環境変数の設定」

kswapd
セクション18.3.2.6「MySQL Cluster データノードの定義」

L

[索引の先頭]

ldconfig
セクション24.3.2.5「ユーザー定義関数のコンパイルおよびインストール」

less
セクション4.5.1.1「mysql のオプション」
セクション4.5.1.2「mysql コマンド」

libmemcached
libmemcached のコマンド行ユーティリティー

ln
Unix 上の MyISAM へのシンボリックリンクの使用

logger
セクション4.3.2「mysqld_safe — MySQL サーバー起動スクリプト」

lsof +L1
セクションB.5.4.4「MySQL が一時ファイルを格納する場所」

M

[索引の先頭]

m4
セクション2.9「ソースから MySQL をインストールする」

make
セクション2.8「FreeBSD に MySQL をインストールする」
セクション16.6.1「memcached のインストール」
セクション2.9.5「MySQL のコンパイルに関する問題」
セクション2.13.3「Perl DBI/DBD インタフェース使用の問題」
セクション2.9「ソースから MySQL をインストールする」
セクション24.2.4.3「プラグインライブラリのコンパイルおよびインストール」
セクション2.9.2「標準ソース配布を使用して MySQL をインストールする」

make && make install
セクション18.2.2.3「Linux でのソースからの MySQL Cluster のビルド」

make install

セクション18.2.2.3「Linux でのソースからの MySQL Cluster のビルド」
セクション16.6.1「memcached のインストール」
セクション2.9.4「MySQL ソース構成オプション」
セクション24.2.4.3「プラグインライブラリのコンパイルおよびインストール」

make package

セクション2.9.4「MySQL ソース構成オプション」
セクション2.9.2「標準ソース配布を使用して MySQL をインストールする」

make test

セクション24.1.2「MySQL テストスイート」
セクション2.13.1「Unix に Perl をインストールする」
セクション2.9.3「開発ソースツリーを使用して MySQL をインストールする」

make VERBOSE=1

セクション2.9.5「MySQL のコンパイルに関する問題」

md5

セクション2.1.4.1「MD5 チェックサムの確認」

md5.exe

セクション2.1.4.1「MD5 チェックサムの確認」

md5sum

セクション2.1.4.1「MD5 チェックサムの確認」

memcache

セクション16.6.2.4「memcached のハッシュ化/分布タイプ」
セクション16.6.3.5「Python での MySQL と memcached の使用」

memcached

セクション16.6.3.3「C および C++ での libmemcached の使用」
セクション16.4.1「EC2 AMI での MySQL のセットアップ」
セクション16.4.3「EC2 を使用した MySQL データベースの配備」
セクション14.18.5「InnoDB memcached インタフェース用のアプリケーションの作成」
セクション14.18.3.2「InnoDB memcached プラグインのインストールおよび構成」
セクション14.18.4「InnoDB memcached プラグインのセキュリティーに関する考慮事項」
セクション14.18.8「InnoDB memcached プラグインのトラブルシューティング」
セクション14.18.5.4「InnoDB memcached プラグインのトランザクション動作の制御」
セクション14.18.5.3「InnoDB memcached プラグインのパフォーマンスのチューニング」
セクション14.18.7「InnoDB memcached プラグインの内部構造」
セクション14.18.3.1「InnoDB memcached プラグインの前提条件」
セクション14.18.3「InnoDB Memcached プラグインの概要」
セクション14.18.2「InnoDB および memcached の統合のアーキテクチャー」
セクション14.18.3.3「InnoDB および memcached 設定の検証」
セクション14.18.1「InnoDB と memcached の組み合わせの利点」
セクション14.18「InnoDB と memcached の統合」
セクション14.12「InnoDB の起動オプションおよびシステム変数」
セクション16.6.3.8「Java での MySQL と memcached の使用」
libmemcached のコマンド行ユーティリティー
libmemcached の設定関数
セクション16.6.2.5「memcached と DTrace の使用」
セクション16.6.5「memcached の FAQ」
セクション16.6.3.9「memcached の TCP テキストプロトコルの使用」
セクション16.6.1「memcached のインストール」
セクション16.6.4.4「memcached のサイズ統計」
セクション16.6.4.2「memcached のスラブ統計」
セクション16.6.2.7「memcached のスレッドサポート」
セクション16.6.2.4「memcached のハッシュ化/分布タイプ」
セクション16.6.4.1「memcached の一般統計」
セクション16.6.2「memcached の使用」
セクション16.6.3.1「memcached の基本操作」
セクション16.6.4「memcached の統計の取得」
セクション16.6.4.5「memcached の詳細統計」
セクション16.6.2.1「memcached の配備」
セクション16.6.4.3「memcached の項目統計」
セクション14.18.5.1「memcached アプリケーションに対する既存の MySQL スキーマの改変」
セクション16.6.3「memcached アプリケーションの開発」
セクション16.6.2.8「memcached ログ」
セクション16.6.2.6「memcached 内でのメモリー割り当て」
セクション14.18.5.5「memcached 操作に合わせた DML ステートメントの改変」
セクション16.6.4.6「memcached-tool の使用」
セクション1.4「MySQL 5.6 の新機能」
セクション18.1.1「MySQL Cluster の主な概念」
セクション16.6「MySQL と memcached の併用」
セクション16.6.3.2「MySQL キャッシュレイヤーとしての memcached の使用」
セクション2.9.4「MySQL ソース構成オプション」
MySQL 用語集
セクション16.6.3.4「Perl での MySQL と memcached の使用」
セクション16.6.3.6「PHP での MySQL と memcached の使用」
セクション16.6.3.5「Python での MySQL と memcached の使用」
セクション16.6.3.7「Ruby での MySQL と memcached の使用」
セクション14.18.4.1「SASL による memcached インタフェースのパスワード保護」
セクション16.6.2.3「データ失効」
セクション14.18.5.6「ベースとなる InnoDB テーブルでの DML および DDL ステートメントの実行」
セクション14.18.6「レプリケーションでの InnoDB memcached プラグインの使用」
セクション16.6.2.2「名前空間の使用」
セクション14.18.5.2「統合型の memcached デーモンのための既存の memcached アプリケーションの改変」

memcached-1.2.5
セクション16.6.1「memcached のインストール」

memcached-tool
セクション16.6.4「memcached の統計の取得」
セクション16.6.4.6「memcached-tool の使用」

memcapable
セクション14.18.2「InnoDB および memcached の統合のアーキテクチャー」

memcat
セクション14.18.2「InnoDB および memcached の統合のアーキテクチャー」
セクション14.18.3.3「InnoDB および memcached 設定の検証」
libmemcached のコマンド行ユーティリティー

memcp
セクション14.18.2「InnoDB および memcached の統合のアーキテクチャー」
libmemcached のコマンド行ユーティリティー

memflush
セクション14.18.2「InnoDB および memcached の統合のアーキテクチャー」
libmemcached のコマンド行ユーティリティー

memrm
セクション14.18.2「InnoDB および memcached の統合のアーキテクチャー」
libmemcached のコマンド行ユーティリティー

memslap
セクション14.18.5.3「InnoDB memcached プラグインのパフォーマンスのチューニング」
libmemcached のコマンド行ユーティリティー

mgmd
セクション18.2「MySQL Cluster のインストール」

mkdir
セクション13.1.10「CREATE DATABASE 構文」

mklink
Windows 上のデータベースへのシンボリックリンクの使用

more
セクション4.5.1.1「mysql のオプション」
セクション4.5.1.2「mysql コマンド」

msql2mysql
セクション4.7.1「msql2mysql — mSQL プログラムを MySQL で使用するために変換」
セクション1.4「MySQL 5.6 の新機能」
セクション4.1「MySQL プログラムの概要」
セクション4.8.2「replace — 文字列置換ユーティリティー」

mv
セクション5.2.2「エラーログ」
セクション5.2.7「サーバーログの保守」
セクション5.2.3「一般クエリーログ」

my_print_defaults
セクション4.7.3「my_print_defaults — オプションファイルからのオプションの表示」
セクション4.1「MySQL プログラムの概要」
セクション4.7「MySQL プログラム開発ユーティリティー」
セクション4.4.4「mysql_plugin — MySQL サーバープラグインの構成」

myisam_ftdump
セクション4.6.2「myisam_ftdump — 全文インデックス情報の表示」
セクション4.1「MySQL プログラムの概要」
セクション12.9「全文検索関数」

myisamchk
セクション13.7.2.1「ANALYZE TABLE 構文」
セクション13.7.2.2「CHECK TABLE 構文」
セクション13.2.2「DELETE 構文」
セクション8.8.2「EXPLAIN 出力フォーマット」
セクション8.3.7「InnoDB および MyISAM インデックス統計コレクション」
セクション13.7.6.5「LOAD INDEX INTO CACHE 構文」
セクション8.6.1「MyISAM クエリーの最適化」
セクション15.2「MyISAM ストレージエンジン」
セクション7.6.2「MyISAM テーブルのエラーのチェック方法」
セクション15.2.3「MyISAM テーブルのストレージフォーマット」
セクション8.6.2「MyISAM テーブルの一括データロード」
セクション7.6「MyISAM テーブルの保守とクラッシュリカバリ」
セクション7.6.3「MyISAM テーブルの修復方法」
セクション7.6.4「MyISAM テーブルの最適化」
セクション15.2.4.1「MyISAM テーブルの破損」
セクション7.6.5「MyISAM テーブル保守スケジュールのセットアップ」
セクション15.2.1「MyISAM 起動オプション」
セクション4.6.3「myisamchk — MyISAM テーブルメンテナンスユーティリティー」
セクション4.6.3.5「myisamchk によるテーブル情報の取得」
セクション4.6.3.2「myisamchk のチェックオプション」
セクション4.6.3.1「myisamchk の一般オプション」
セクション4.6.3.3「myisamchk の修復オプション」
セクション4.6.3.6「myisamchk メモリー使用量」
セクション4.6.5「myisampack — 圧縮された読み取り専用の MyISAM テーブルの生成」
セクションB.5.4.2「MySQL が繰り返しクラッシュする場合の対処方法」
セクション1.3.2「MySQL の主な機能」
セクション12.9.6「MySQL の全文検索の微調整」
セクション24.4.1「MySQL サーバーのデバッグ」
セクション4.1「MySQL プログラムの概要」
セクション4.5.3「mysqlcheck — テーブル保守プログラム」
セクション24.4.1.6「mysqld でのエラーの原因を見つけるためのサーバーログの使用」
セクション4.6.10「mysqlhotcopy — データベースバックアッププログラム」
セクション8.6.3「REPAIR TABLE ステートメントの速度」
セクション13.7.2.5「REPAIR TABLE 構文」
セクション13.7.5.23「SHOW INDEX 構文」

セクション13.7.5.37「SHOW TABLE STATUS 構文」
Unix 上の MyISAM へのシンボリックリンクの使用
セクション4.6.3.4「その他の myisamchk オプション」
セクション8.2.5「その他の最適化のヒント」
セクション7.6.1「クラッシュリカバリへの myisamchk の使用」
セクション5.1.3「サーバーコマンドオプション」
セクション8.11.1「システム要素およびスタートアップパラメータのチューニング」
セクション24.4.1.7「テーブルが破損した場合のテストケースの作成」
セクションD.10.3「テーブルサイズの制限」
セクション7.2「データベースバックアップ方法」
セクション19.6「パーティショニングの制約と制限」
セクション19.3.4「パーティションの保守」
セクション15.2.3.2「動的テーブルの特徴」
セクション15.2.3.3「圧縮テーブルの特徴」
セクション8.10.5「外部ロック」
セクション10.5「文字セットの構成」
セクション1.6「質問またはバグをレポートする方法」
セクション15.2.4.2「適切に閉じられなかったテーブルの問題」
セクション15.2.3.1「静的 (固定長) テーブルの特長」

myisamchk *.MYI

セクション7.6.3「MyISAM テーブルの修復方法」

myisamchk tbl_name

セクション7.6.2「MyISAM テーブルのエラーのチェック方法」

myisamlog

セクション4.6.4「myisamlog — MyISAM ログファイルの内容の表示」
セクション4.1「MySQL プログラムの概要」

myisampack

セクション13.1.17「CREATE TABLE 構文」
セクション15.7「MERGE ストレージエンジン」
セクション15.7.1「MERGE テーブルの長所と短所」
セクション15.2「MyISAM ストレージエンジン」
セクション15.2.3「MyISAM テーブルのストレージフォーマット」
セクション8.6.2「MyISAM テーブルの一括データロード」
セクション4.6.3.5「myisamchk によるテーブル情報の取得」
セクション4.6.3.3「myisamchk の修復オプション」
セクション4.6.5「myisampack — 圧縮された読み取り専用の MyISAM テーブルの生成」
セクション4.1「MySQL プログラムの概要」
セクションD.10.3「テーブルサイズの制限」
セクション8.4.1「データサイズの最適化」
セクション19.6「パーティショニングの制約と制限」
セクション15.2.3.3「圧縮テーブルの特徴」
セクション8.10.5「外部ロック」
セクション13.1.17.3「暗黙のカラム指定の変更」

mysql

--safe-updates オプションの使用
セクション13.1.7.1「ALTER TABLE パーティション操作」
セクション13.6.1「BEGIN ... END 複合ステートメント構文」
セクション11.4.3「BLOB 型と TEXT 型」
セクション23.7.3「C API クライアントプログラムの例」
セクション13.1.15「CREATE PROCEDURE および CREATE FUNCTION 構文」
セクション13.6.7.3「GET DIAGNOSTICS 構文」
セクション13.7.1.4「GRANT 構文」
セクション13.8.3「HELP 構文」
セクション14.16「InnoDB のバックアップとリカバリ」
セクション14.5.1「InnoDB テーブルスペースの作成」
セクション14.19.3「InnoDB データディクショナリの操作のトラブルシューティング」
セクション14.15.2「InnoDB モニターの有効化」
セクション5.1.9.3「IPv6 ローカルホストアドレスを使用した接続」
セクション5.1.9.4「IPv6 非ローカルホストアドレスを使用した接続」
セクション8.2.1.19「LIMIT クエリーの最適化」
セクション13.2.6「LOAD DATA INFILE 構文」
セクション6.1.6「LOAD DATA LOCAL のセキュリティーの問題」
セクション13.2.7「LOAD XML 構文」
セクション14.6.4「MyISAM から InnoDB へのテーブルの変換」
セクションA.10「MySQL 5.6 FAQ: MySQL Cluster」
セクションA.11「MySQL 5.6 FAQ: MySQL の中国語、日本語、および韓国語の文字セット」
セクション1.4「MySQL 5.6 の新機能」
第18章「MySQL Cluster NDB 7.3 および MySQL Cluster NDB 7.4」
セクション18.5.4「MySQL Cluster での MySQL サーバーの使用法」
セクション18.1.6.8「MySQL Cluster に限定された問題」
セクション18.3「MySQL Cluster の構成」
セクション18.5「MySQL Cluster の管理」
セクション18.5.14「MySQL Cluster の配布された MySQL 権限」
セクション18.5.12.1「MySQL Cluster ディスクデータオブジェクト」
セクション18.5.13.1「MySQL Cluster データノードのオンライン追加: 一般的な問題」
セクション18.5.13.2「MySQL Cluster データノードのオンライン追加: 基本的な手順」
セクション18.5.13.3「MySQL Cluster データノードのオンライン追加: 詳細な例」
セクション18.6「MySQL Cluster レプリケーション」
セクション18.6.6「MySQL Cluster レプリケーションの起動 (レプリケーションチャネルが 1 つ)」
セクション18.6.9「MySQL Cluster レプリケーションを使用した MySQL Cluster バックアップ」
セクション18.6.4「MySQL Cluster レプリケーションスキーマとテーブル」
セクション18.3.2.7「MySQL Cluster 内の SQL ノードおよびその他の API ノードの定義」
セクション18.5.2「MySQL Cluster 管理クライアントのコマンド」
セクション10.6「MySQL Server でのタイムゾーンのサポート」
セクション4.5.1「mysql — MySQL コマンド行ツール」
セクション6.1.2.4「MySQL でのパスワードハッシュ」
セクション4.5.1.1「mysql のオプション」
セクション2.11.2「MySQL のダウングレード」
セクション4.5.1.6「mysql のヒント」
セクション4.5.1.3「mysql の口ギング」
mysql の自動再接続を無効にする
セクション1.8.1「MySQL への貢献者」

セクション2.10.1.2「MySQL を自動的に起動および停止する」
セクション6.1.5「MySQL を通常ユーザーとして実行する方法」
セクション24.4.2「MySQL クライアントのデバッグ」
セクション4.5.1.2「mysql コマンド」
セクション4.2.2「MySQL サーバーへの接続」
セクション4.5.1.4「mysql サーバー側のヘルプ」
セクション2.3.5.6「MySQL ツールの PATH をカスタマイズする」
セクション2.11.5「MySQL データベースのほかのマシンへのコピー」
セクション4.1「MySQL プログラムの概要」
セクション4.2.1「MySQL プログラムの起動」
セクション4.6.6「mysql_config_editor — MySQL 構成ユーティリティー」
セクション23.7.7.14「mysql_errno()」
セクション4.4.3「mysql_install_db — MySQL データディレクトリの初期化」
セクション2.10.1.1「mysql_install_db 実行の問題」
セクション23.7.7.67「mysql_sqlstate()」
セクション4.4.6「mysql_tzinfo_to_sql — タイムゾーンテーブルのロード」
セクション4.6.7「mysqlaccess — アクセス権限をチェックするクライアント」
セクション4.6.8「mysqlbinlog — バイナリログファイルを処理するためのユーティリティー」
セクション24.4.1.6「mysqld でのエラーの原因を見つけるためのサーバーログの使用」
セクション4.5.4「mysqldump — データベースバックアッププログラム」
セクション4.5.6「mysqlshow — データベース、テーブル、およびカラム情報の表示」
セクション18.5.15「NDB API 統計のカウンタと変数」
セクション18.4.5「ndb_mgm — MySQL Cluster 管理クライアント」
セクション18.5.10.17「ndbinfo memory_per_fragment テーブル」
セクション18.5.10「ndbinfo MySQL Cluster 情報データベース」
セクション18.5.10.24「ndbinfo transporters テーブル」
セクション2.4.1「OS X への MySQL のインストールに関する一般的な注記」
セクション19.2.3.1「RANGE COLUMNS パーティショニング」
セクション13.7.1.6「REVOKE 構文」
root のパスワードのリセット: 一般的な手順
セクション18.2.2.2「RPM からの MySQL Cluster のインストール」
セクション2.5.5「RPM パッケージを使用して MySQL を Linux にインストールする」
セクション13.2.9.1「SELECT ... INTO 構文」
セクション6.3.8.4「SHA-256 認証プラグイン」
セクション13.7.5.35「SHOW SLAVE STATUS 構文」
セクション13.6.7.5「SIGNAL 構文」
セクション7.4.2「SQL フォーマットバックアップのリロード」
セクション6.3.10.3「SSL 接続の使用」
セクション18.2.3.3「Windows での MySQL Cluster の初期起動」
セクション2.3.8「Windows でのインストール後の手順」
セクション18.2.3.1「Windows でのバイナリリリースからの MySQL Cluster のインストール」
Windows における Unicode のサポート
セクション2.3.5.7「Windows のサービスとして MySQL を起動する」
セクション12.11「XML 関数」
セクション6.2.7「アクセス拒否エラーの原因」
セクション10.1.5「アプリケーションの文字セットおよび照合順序の構成」
セクション20.4.2「イベントスケジューラの構成」
セクション4.2.9「オプションのデフォルト、値を想定するオプション、および = 記号」
セクション4.2.6「オプションファイルの使用」
セクション4.2.7「オプションファイルの処理に影響するコマンド行オプション」
セクション14.11.5「オンライン DDL の例」
セクション3.2「クエリーの入力」
セクション6.3.8.7「クライアント側のクリアテキスト認証プラグイン」
セクション4.2.4「コマンド行でのオプションの使用」
セクション1.7.2.5「コメントの先頭としての「--」」
セクション9.6「コメントの構文」
セクションB.3「サーバーのエラーコードおよびメッセージ」
セクション3.1「サーバーへの接続とサーバーからの切断」
セクション5.1.3「サーバーコマンドオプション」
セクション5.1.4「サーバーシステム変数」
セクション5.1.10「サーバー側のヘルプ」
セクション17.4.1.27「サーバー側ヘルプテーブルのレプリケーション」
セクション20.1「ストアドプログラムの定義」
セクション6.3.8.8「ソケットピア証明書認証プラグイン」
第3章「チュートリアル」
セクション4.5.1.5「テキストファイルから SQL ステートメントを実行する」
セクション18.2.6「テーブルとデータを含む MySQL Cluster の例」
セクション2.11.4「テーブルまたはインデックスの再作成または修復」
セクション18.5.12.2「ディスクデータオブジェクトでのシンボリックリンクの使用」
セクション14.2.11「デッドロックの対処方法」
セクション7.4.5.1「データベースのコピーの作成」
セクション3.3.1「データベースの作成と選択」
セクション20.3.1「トリガーの構文と例」
セクション7.5「バイナリログを使用したポイントインタイム (増分) リカバリ」
セクション4.6.8.3「バイナリログファイルのバックアップのための mysqlbinlog の使用」
セクション7.3「バックアップおよびリカバリ戦略の例」
セクション7.1「バックアップとリカバリの種類」
セクション7.4「バックアップへの mysqldump の使用」
セクション3.5「バッチモードでの MySQL の使用」
セクションB.5.2.10「パケットが大きすぎます」
セクション6.3.6「パスワードの期限切れとサンドボックスモード」
セクション6.1.2.1「パスワードセキュリティーのためのエンドユーザーガイドライン」
セクション19.3.3「パーティションとサブパーティションをテーブルと交換する」
セクション17.1.3.3「フェイルオーバーおよびスケールアウトでの GTID の使用」
セクションD.9「プラガブルな認証の制約」
セクション6.3.7「プラガブル認証」
プロキシユーザーとグループマッピングを使用した Unix パスワード認証
プロキシユーザーを使用しない Unix パスワード認証

セクション4.2.3「プログラムオプションの指定」
セクション4.2.5「プログラムオプション修飾子」
セクション4.2.8「プログラム変数の設定へのオプションの使用」
セクションB.5.2.8「メモリー不足」
セクションB.5.2.15「ユーザーを無視します」
セクション6.3.2「ユーザーアカウントの追加」
セクション7.3.2「リカバリへのバックアップの使用」
セクション18.6.5「レプリケーションのための MySQL Cluster の準備」
セクション3.6「一般的なクエリーの例」
入力行の編集
セクション7.4.4「区切りテキストフォーマットバックアップのリロード」
セクションB.5.1「問題の原因を判別する方法」
セクション8.12.1「式と関数の速度の測定」
セクション12.14「情報関数」
セクション10.1.4「接続文字セットおよび照合順序」
セクション6.1.3「攻撃者に対する MySQL のセキュアな状態の維持」
セクション9.1.1「文字列リテラル」
セクション2.10.2「最初の MySQL アカウントのセキュリティー設定」
セクション13.5「準備済みステートメントのための SQL 構文」
セクション2.12「環境変数」
セクション23.7.16「自動再接続動作の制御」
セクション1.2「表記規則および構文規則」
セクション1.6「質問またはバグをレポートする方法」
第12章「関数と演算子」
セクション14.13.16.2「非永続的オプティマイザ統計のパラメータの構成」

mysql ...

セクション24.4.1.1「デバッグのための MySQL のコンパイル」

mysql-server

セクション2.8「FreeBSD に MySQL をインストールする」

mysql-test-run.pl

セクション24.1.2「MySQL テストスイート」
セクション4.6.6「mysql_config_editor — MySQL 構成ユーティリティー」
セクション4.2.6「オプションファイルの使用」

mysql-test-run.pl test_name

セクション24.1.2「MySQL テストスイート」

mysql.exe

セクションA.11「MySQL 5.6 FAQ: MySQL の中国語、日本語、および韓国語の文字セット」
セクション18.2.3.3「Windows での MySQL Cluster の初期起動」
セクション18.2.3.1「Windows でのバイナリリリースからの MySQL Cluster のインストール」
Windows における Unicode のサポート

mysql.script

セクション2.4.1「OS X への MySQL のインストールに関する一般的な注記」

mysql.server

セクション2.5「Linux に MySQL をインストールする」
セクション2.10.1.2「MySQL を自動的に起動および停止する」
セクション4.1「MySQL プログラムの概要」
セクション4.3.3「mysql.server — MySQL サーバー起動スクリプト」
セクション4.6.9「mysqldumpslow — スロークエリーログファイルの要約」
セクション18.2.2.2「RPM からの MySQL Cluster のインストール」
セクション5.1.3「サーバーコマンドオプション」
セクションB.5.4.6「タイムゾーンの問題」
セクション6.1.3「攻撃者に対する MySQL のセキュアな状態の維持」

mysql.server stop

セクション2.10.1.2「MySQL を自動的に起動および停止する」

mysql_config

セクション23.7.4.1「C API クライアントプログラムの構築」
セクション23.6.1「libmysqld によるプログラムのコンパイル」
セクション23.7.1「MySQL C API の実装」
セクション2.9.5「MySQL のコンパイルに関する問題」
セクション4.1「MySQL プログラムの概要」
セクション4.7.2「mysql_config — クライアントのコンパイル用オプションの表示」
セクション24.2.2「プラグイン API のコンポーネント」

mysql_config_editor

セクション4.7.3「my_print_defaults — オプションファイルからのオプションの表示」
セクション4.6.3.1「myisamchk の一般オプション」
セクション1.4「MySQL 5.6 の新機能」
セクション4.5.1.1「mysql のオプション」
セクション4.1「MySQL プログラムの概要」
セクション4.6.6「mysql_config_editor — MySQL 構成ユーティリティー」
セクション4.4.7「mysql_upgrade — MySQL テーブルのチェックとアップグレード」
セクション4.5.2「mysqladmin — MySQL サーバーの管理を行うクライアント」
セクション4.6.8「mysqlbinlog — バイナリログファイルを処理するためのユーティリティー」
セクション4.5.3「mysqlcheck — テーブル保守プログラム」
セクション4.5.4「mysqldump — データベースバックアッププログラム」
セクション4.5.5「mysqlimport — データインポートプログラム」
セクション4.5.6「mysqlshow — データベース、テーブル、およびカラム情報の表示」
セクション4.5.7「mysqlslap — 負荷エミュレーションクライアント」
セクション4.2.6「オプションファイルの使用」
セクション4.2.7「オプションファイルの処理に影響するコマンド行オプション」
セクション5.1.3「サーバーコマンドオプション」
セクション6.1.2.1「パスワードセキュリティーのためのエンドユーザーガイドライン」
セクション2.12「環境変数」

mysql_convert_table_format

セクション1.4「MySQL 5.6 の新機能」
セクション4.1「MySQL プログラムの概要」
セクション4.6.11「mysql_convert_table_format — 指定されたストレージエンジンを使用するためのテーブルの変換」

mysql_find_rows

セクション1.4「MySQL 5.6 の新機能」
セクション4.1「MySQL プログラムの概要」
セクション4.6.12「mysql_find_rows — ファイルからの SQL ステートメントの抽出」

mysql_fix_extensions

セクション1.4「MySQL 5.6 の新機能」
セクション4.1「MySQL プログラムの概要」
セクション4.6.13「mysql_fix_extensions — テーブルファイル名の拡張子の正規化」

mysql_install_db

セクション14.6.7「InnoDB テーブル上の制限」
セクション2.7.2「IPS を使用して MySQL を OpenSolaris にインストールする」
セクション1.4「MySQL 5.6 の新機能」
セクション18.6.4「MySQL Cluster レプリケーションスキーマとテーブル」
セクション2.9.4「MySQL ソース構成オプション」
セクション4.1「MySQL プログラムの概要」
セクション4.4.3「mysql_install_db — MySQL データディレクトリの初期化」
セクション2.10.1.1「mysql_install_db 実行の問題」
セクション2.5.5「RPM パッケージを使用して MySQL を Linux にインストールする」
セクション2.10.1「Unix 類似システムでのインストール後の手順」
セクション6.2.7「アクセス拒否エラーの原因」
セクション17.1.4.5「グローバルトランザクション ID のオプションと変数」
セクション5.1.2.2「サンプルのデフォルトサーバー構成ファイルの使用」
セクション5.1.7「サーバー SQL モード」
セクション5.1.3「サーバーコマンドオプション」
セクション5.1.4「サーバーシステム変数」
セクション5.1.10「サーバー側のヘルプ」
セクション2.5.7「ネイティブソフトウェアリポジトリから MySQL を Linux にインストールする」
セクション2.4.2「ネイティブパッケージを使用して OS X に MySQL をインストールする」
セクション6.3.2「ユーザーアカウントの追加」
セクション2.2「一般的なバイナリを使用した MySQL の Unix/Linux へのインストール」
セクション2.10.2「最初の MySQL アカウントのセキュリティー設定」
セクション2.9.2「標準ソース配布を使用して MySQL をインストールする」
セクション5.3.1「複数のデータディレクトリのセットアップ」

mysql_plugin

セクション4.1「MySQL プログラムの概要」
セクション4.4.4「mysql_plugin — MySQL サーバープラグインの構成」

mysql_secure_installation

セクション2.7.2「IPS を使用して MySQL を OpenSolaris にインストールする」
セクション2.5.1「MySQL Yum リポジトリを使用して MySQL を Linux にインストールする」
セクション4.1「MySQL プログラムの概要」
セクション4.4.3「mysql_install_db — MySQL データディレクトリの初期化」
セクション4.4.5「mysql_secure_installation — MySQL インストールのセキュリティー改善」
セクション2.5.5「RPM パッケージを使用して MySQL を Linux にインストールする」
セクション2.7.1「Solaris PKG を使用して Solaris に MySQL をインストールする」
セクション2.5.7「ネイティブソフトウェアリポジトリから MySQL を Linux にインストールする」
セクション2.10.2「最初の MySQL アカウントのセキュリティー設定」

mysql_setpermission

セクション1.4「MySQL 5.6 の新機能」
セクション1.8.1「MySQL への貢献者」
セクション4.1「MySQL プログラムの概要」
セクション4.6.14「mysql_setpermission — 付与テーブルに許可をインタラクティブに設定」

mysql_setpermissions

セクション4.6.14「mysql_setpermission — 付与テーブルに許可をインタラクティブに設定」

mysql_stmt_execute()

セクション5.1.6「サーバーステータス変数」

mysql_stmt_prepare()

セクション5.1.6「サーバーステータス変数」

mysql_tzinfo_to_sql

セクション10.6「MySQL Server でのタイムゾーンのサポート」
セクション4.1「MySQL プログラムの概要」
セクション4.4.6「mysql_tzinfo_to_sql — タイムゾーンテーブルのロード」

mysql_upgrade

セクション6.3.8.3「4.1 よりも前のパスワードハッシュ方式と mysql_old_password プラグインからの移行」
セクション13.1.1「ALTER DATABASE 構文」
セクション17.1.3.4「GTID ベースレプリケーションの制約」
セクション2.11.1.3「MySQL 5.5 から 5.6 へのアップグレード」
セクション1.4「MySQL 5.6 の新機能」
セクション2.11.1.1「MySQL Yum リポジトリを使用する MySQL のアップグレード」
セクション6.1.2.4「MySQL でのパスワードハッシュ」
セクション2.11.1「MySQL のアップグレード」
セクション2.11.2「MySQL のダウングレード」
セクション4.1「MySQL プログラムの概要」
セクション4.4.7「mysql_upgrade — MySQL テーブルのチェックとアップグレード」
セクション4.5.3「mysqlcheck — テーブル保守プログラム」
セクション2.3.7「Windows 上の MySQL をアップグレードする」

セクション11.3.4「YEAR(2) の制限と YEAR(4) への移行」
セクション6.2.7「アクセス拒否エラーの原因」
セクション17.1.4.5「グローバルトランザクション ID のオプションと変数」
セクション2.5.2「サードパーティーの MySQL 配布を MySQL Yum リポジトリを使用して置換する」
セクション5.1.3「サーバーコマンドオプション」
セクション17.4.1.27「サーバー側ヘルプテーブルのレプリケーション」
セクション2.11.4「テーブルまたはインデックスの再作成または修復」
セクション2.11.3「テーブルまたはインデックスの再構築が必要かどうかのチェック」
セクション22.2.1「パフォーマンススキーマビルド構成」
セクション13.7.3.1「ユーザー定義関数のための CREATE FUNCTION 構文」

mysql_waitpid

セクション1.4「MySQL 5.6 の新機能」
セクション4.1「MySQL プログラムの概要」
セクション4.6.15「mysql_waitpid — プロセスを強制終了して終了を待機」

mysql_waitpid()

セクション4.6.15「mysql_waitpid — プロセスを強制終了して終了を待機」

mysql_zap

セクション1.4「MySQL 5.6 の新機能」
セクション4.1「MySQL プログラムの概要」
セクション4.6.16「mysql_zap — パターンに一致するプロセスを強制終了」

mysqlaccess

セクション1.4「MySQL 5.6 の新機能」
セクション1.8.1「MySQL への貢献者」
セクション4.1「MySQL プログラムの概要」
セクション4.6.7「mysqlaccess — アクセス権限をチェックするクライアント」
セクション1.6「質問またはバグをレポートする方法」

mysqladmin

セクションB.5.2.2「[ローカルの] MySQL サーバーに接続できません」
セクション23.7.3「C API クライアントプログラムの例」
セクション13.1.10「CREATE DATABASE 構文」
セクション13.1.21「DROP DATABASE 構文」
セクション13.7.6.3「FLUSH 構文」
セクション17.1.3.2「GTID を使用したレプリケーションのセットアップ」
セクション7.6.3「MyISAM テーブルの修復方法」
セクションA.11「MySQL 5.6 FAQ: MySQL の中国語、日本語、および韓国語の文字セット」
セクション5.2「MySQL Server ログ」
セクションB.5.4.2「MySQL が繰り返しクラッシュする場合の対処方法」
セクション6.2.1「MySQL で提供される権限」
セクション1.3.2「MySQL の主な機能」
セクション1.8.1「MySQL への貢献者」
セクション24.4.1「MySQL サーバーのデバッグ」
セクション4.2.2「MySQL サーバーへの接続」
セクション2.3.5.6「MySQL ツールの PATH をカスタマイズする」
セクション4.1「MySQL プログラムの概要」
セクション4.6.6「mysql_config_editor — MySQL 構成ユーティリティー」
セクション4.4.3「mysql_install_db — MySQL データディレクトリの初期化」
セクション4.5.2「mysqladmin — MySQL サーバーの管理を行うクライアント」
セクション4.3.4「mysqld_multi — 複数の MySQL サーバーの管理」
セクション17.3.1.1「mysqldump を使用してスレーブをバックアップする」
セクション2.4.1「OS X への MySQL のインストールに関する一般的な注記」
セクション2.5.5「RPM パッケージを使用して MySQL を Linux にインストールする」
セクション5.3.3「Unix 上での複数の MySQL インスタンスの実行」
セクション2.10.1「Unix 類似システムでのインストール後の手順」
セクション2.3.5.5「Windows のコマンド行からの MySQL の起動」
セクション2.3.5.7「Windows のサービスとして MySQL を起動する」
セクション2.3.7「Windows 上の MySQL をアップグレードする」
セクション6.3.5「アカウントパスワードの割り当て」
セクション4.2.6「オプションファイルの使用」
セクション6.3.8.7「クライアント側のクリアテキスト認証プラグイン」
セクション4.2.4「コマンド行でのオプションの使用」
セクション8.11.2「サーバーパラメータのチューニング」
セクション5.1.12「シャットダウンプロセス」
セクション6.3.6「パスワードの期限切れとサンドボックスモード」
セクションD.9「プラガブルな認証の制約」
セクションB.5.1「問題の原因を判別する方法」
セクション2.10.2「最初の MySQL アカウントのセキュリティー設定」
セクション1.6「質問またはバグをレポートする方法」

mysqladmin debug

セクション13.7.1.4「GRANT 構文」
セクション6.2.1「MySQL で提供される権限」
セクション24.4.1「MySQL サーバーのデバッグ」
セクション20.4.5「イベントスケジューラのステータス」

mysqladmin extended-status

セクション13.2.5.2「INSERT DELAYED 構文」
セクション13.7.5.36「SHOW STATUS 構文」

mysqladmin flush-hosts

セクション8.11.5.2「DNS ルックアップの最適化とホストキャッシュ」
セクション6.2.7「アクセス拒否エラーの原因」
セクション5.1.4「サーバーシステム変数」
セクションB.5.2.6「ホスト 'host_name' は拒否されました」

mysqladmin flush-logs

セクション5.2.2「エラーログ」
セクション5.2.7「サーバーログの保守」
セクション17.2.2.1「スレーブリレーログ」
セクション5.2.4「バイナリログ」
セクション7.3.1「バックアップポリシーの確立」

セクション7.3.3「バックアップ戦略サマリー」

mysqladmin flush-privileges

セクション2.11.5「MySQL データベースのほかのマシンへのコピー」
セクション2.10.1.1「mysql_install_db 実行の問題」
セクション4.5.2「mysqladmin — MySQL サーバーの管理を行うクライアント」
セクション6.2.7「アクセス拒否エラーの原因」
セクション5.1.3「サーバーコマンドオプション」
セクション6.2.2「権限システム付与テーブル」
セクション6.2.6「権限変更が有効化される時期」

mysqladmin flush-tables

セクション8.6.2「MyISAM テーブルの一括データロード」
セクション4.6.5「myisampack — 圧縮された読み取り専用の MyISAM テーブルの生成」
セクション8.4.3.1「MySQL でのテーブルのオープンとクローズの方法」
セクション8.11.4.1「MySQL のメモリーの使用方法」
セクション7.6.1「クラッシュリカバリへの myisamchk の使用」
セクション8.10.5「外部ロック」

mysqladmin flush-xxx

セクション6.3.2「ユーザーアカウントの追加」

mysqladmin kill

セクション13.7.6.4「KILL 構文」
セクションB.5.4.3「MySQL が満杯のディスクを処理する方法」
セクション6.2.1「MySQL で提供される権限」
セクションB.5.2.9「MySQL サーバーが存在しなくなりました」
セクション12.18「その他の関数」

mysqladmin password

セクション4.5.2「mysqladmin — MySQL サーバーの管理を行うクライアント」
セクション6.3.5「アカウントパスワードの割り当て」
セクション6.2.7「アクセス拒否エラーの原因」
セクション6.3.6「パスワードの期限切れとサンドボックスモード」

mysqladmin processlist

セクション13.7.6.4「KILL 構文」
セクション6.2.1「MySQL で提供される権限」
セクション24.1.1「MySQL のスレッド」
セクション23.7.7.43「mysql_list_processes()」
セクション13.7.5.30「SHOW PROCESSLIST 構文」
セクション8.12.5「スレッド情報の検査」
セクション6.3.2「ユーザーアカウントの追加」
セクション6.1.3「攻撃者に対する MySQL のセキュアな状態の維持」

mysqladmin processlist status

セクション24.4.1「MySQL サーバーのデバッグ」

mysqladmin refresh

セクション8.4.3.1「MySQL でのテーブルのオープンとクローズの方法」
セクション5.2.7「サーバーログの保守」
セクション6.3.2「ユーザーアカウントの追加」

mysqladmin reload

セクション2.10.1.1「mysql_install_db 実行の問題」
セクション4.6.7「mysqlaccess — アクセス権限をチェックするクライアント」
セクション6.3.4「アカウントリソース制限の設定」
セクション5.1.3「サーバーコマンドオプション」
セクション6.3.2「ユーザーアカウントの追加」
セクション6.2.2「権限システム付与テーブル」
セクション6.2.6「権限変更が有効化される時期」
セクション1.6「質問またはバグをレポートする方法」

mysqladmin reload version

セクション1.6「質問またはバグをレポートする方法」

mysqladmin shutdown

セクション13.7.1.4「GRANT 構文」
セクション14.5.1「InnoDB テーブルスペースの作成」
セクション7.6.3「MyISAM テーブルの修復方法」
セクションA.10「MySQL 5.6 FAQ: MySQL Cluster」
セクション18.2.7「MySQL Cluster の安全なシャットダウンと再起動」
セクション18.5.13.3「MySQL Cluster データノードのオンライン追加: 詳細な例」
セクションB.5.4.2「MySQL が繰り返しクラッシュする場合の対処方法」
セクション6.2.1「MySQL で提供される権限」
セクション2.10.1.2「MySQL を自動的に起動および停止する」
セクション6.1.5「MySQL を通常ユーザーとして実行する方法」
セクション4.5.2「mysqladmin — MySQL サーバーの管理を行うクライアント」
セクション2.3.5.7「Windows のサービスとして MySQL を起動する」
セクション6.2.5「アクセス制御、ステージ 2: リクエストの確認」
セクション5.1.12「シャットダウンプロセス」
セクション24.4.1.7「テーブルが破損した場合のテストケースの作成」
セクション24.4.1.2「トレースファイルの作成」
セクション2.4.2「ネイティブパッケージを使用して OS X に MySQL をインストールする」
セクション17.4.1.22「レプリケーションと一時テーブル」

mysqladmin status

セクション8.4.3.1「MySQL でのテーブルのオープンとクローズの方法」
セクション23.7.7.69「mysql_stat()」
セクション4.5.2「mysqladmin — MySQL サーバーの管理を行うクライアント」

mysqladmin variables

セクションB.5.2.9「MySQL サーバーが存在しなくなりました」
セクション13.7.5.40「SHOW VARIABLES 構文」

mysqladmin variables extended-status processlist

セクション1.6「質問またはバグをレポートする方法」

mysqladmin ver

セクション24.4.1.1「デバッグのための MySQL のコンパイル」

mysqladmin version

セクションB.5.2.2「[ローカルの] MySQL サーバーに接続できません」
セクションB.5.4.2「MySQL が繰り返しクラッシュする場合の対処方法」
セクションB.5.2.9「MySQL サーバーが存在しなくなりました」
セクション2.10.1「Unix 類似システムでのインストール後の手順」
セクション1.6「質問またはバグをレポートする方法」

mysqlanalyze

セクション4.5.3「mysqlcheck — テーブル保守プログラム」

mysqlbackup

セクション14.16「InnoDB のバックアップとリカバリ」
セクション4.5.4「mysqldump — データベースバックアッププログラム」
セクション7.1「バックアップとリカバリの種類」
セクション17.1.1.6「ローデータファイルを使用したデータスナップショットの作成」

mysqlbinlog

セクション13.7.6.1「BINLOG 構文」
セクション17.1.3.1「GTID の概念」
セクション17.1.3.4「GTID ベースレプリケーションの制約」
セクション14.16「InnoDB のバックアップとリカバリ」
セクション1.4「MySQL 5.6 の新機能」
セクション18.6.9.2「MySQL Cluster レプリケーションを使用したポイントインタイムリカバリ」
セクション18.3.4.2「MySQL Cluster 用の MySQL Server オプション」
セクション4.5.1.1「mysql のオプション」
セクションB.5.8「MySQL の既知の問題」
セクション4.1「MySQL プログラムの概要」
MySQL 用語集
セクション4.6.8.1「mysqlbinlog 16 進ダンプ形式」
セクション4.6.8「mysqlbinlog — バイナリログファイルを処理するためのユーティリティー」
セクション4.6.8.4「mysqlbinlog サーバー ID の指定」
セクション4.6.8.2「mysqlbinlog 行イベントの表示」
セクション13.7.5.3「SHOW BINLOG EVENTS 構文」
セクション13.7.5.33「SHOW RELAYLOG EVENTS 構文」
セクション13.4.2.5「START SLAVE 構文」
セクション12.18「その他の関数」
セクション7.5.2「イベントの位置を使用したポイントインタイムリカバリ」
セクション7.5.1「イベント時間を使用したポイントインタイムリカバリ」
セクション5.4.1.2「コマンドプローブ」
セクション17.1.2.1「ステートメントベースおよび行ベースレプリケーションのメリットとデメリット」
セクション17.2.2.1「スレーブリレーログ」
セクション5.2.4「バイナリログ」
セクション17.1.4.4「バイナリログのオプションと変数」
セクション7.5「バイナリログを使用したポイントインタイム (増分) リカバリ」
セクション4.6.8.3「バイナリログファイルのバックアップのための mysqlbinlog の使用」
セクション17.1.3.3「フェイルオーバーおよびスケールアウトでの GTID の使用」
セクション7.3.2「リカバリへのバックアップの使用」
セクション17.4.1.34「レプリケーションと変数」
セクション17.4.5「レプリケーションバグまたは問題を報告する方法」
セクション17.1.2.2「行ベースロギングおよびレプリケーションの使用」

mysqlbinlog binary-log-file | mysql

セクション24.4.1.7「テーブルが破損した場合のテストケースの作成」

mysqlbinlog|mysql

セクションB.5.8「MySQL の既知の問題」

mysqlbug

セクション1.4「MySQL 5.6 の新機能」
セクション4.4.2「mysqlbug — バグレポートの生成」

mysqlcheck

セクション13.1.1「ALTER DATABASE 構文」
セクション4.7.3「my_print_defaults — オプションファイルからのオプションの表示」
セクション15.2「MyISAM ストレージエンジン」
セクション7.6「MyISAM テーブルの保守とクラッシュリカバリ」
セクション1.3.2「MySQL の主な機能」
セクション4.1「MySQL プログラムの概要」
セクション4.4.7「mysql_upgrade — MySQL テーブルのチェックとアップグレード」
セクション4.5.3「mysqlcheck — テーブル保守プログラム」
セクション5.1.4「サーバーシステム変数」
セクション2.11.4「テーブルまたはインデックスの再作成または修復」
セクション2.11.3「テーブルまたはインデックスの再構築が必要かどうかのチェック」
セクション19.6「パーティショニングの制約と制限」
セクション19.3.4「パーティションの保守」
セクション9.2.3「識別子とファイル名のマッピング」

mysqld

セクションB.5.2.18「'File' が見つかりません、および同様のエラー」
セクション5.3「1 つのマシン上での複数の MySQL インスタンスの実行」
セクション18.6.7「2 つのレプリケーションチャネルを使用する MySQL Cluster レプリケーション」
セクションB.5.2.2「[ローカルの] MySQL サーバーに接続できません」
セクション15.6「BLACKHOLE ストレージエンジン」
セクション4.4.1「comp_err — MySQL エラーメッセージファイルのコンパイル」
セクション13.1.17「CREATE TABLE 構文」
セクション24.4.3「DBUG パッケージ」
セクション5.4「DTrace を使用した mysqld のトレース」
セクション24.4.1.4「gdb での mysqld のデバッグ」
セクション17.1.3.2「GTID を使用したレプリケーションのセットアップ」
セクション17.1.3.4「GTID ベースレプリケーションの制約」
セクション14.5.2「InnoDB File-Per-Table モード」

セクション14.18.3.2「InnoDB memcached プラグインのインストールおよび構成」
セクション14.18.5.4「InnoDB memcached プラグインのトランザクション動作の制御」
セクション14.18.2「InnoDB および memcached の統合のアーキテクチャー」
セクション14.19.1「InnoDB の I/O に関する問題のトラブルシューティング」
セクション14.19「InnoDB のトラブルシューティング」
セクション14.16「InnoDB のバックアップとリカバリ」
セクション14.19.2「InnoDB のリカバリの強制的な実行」
セクション14.3「InnoDB の構成」
セクション14.12「InnoDB の起動オプションおよびシステム変数」
セクション18.1.5「InnoDB を使用した MySQL Server と MySQL Cluster の比較」
セクション14.5.1「InnoDB テーブルスペースの作成」
セクション14.10.1「InnoDB ディスク I/O」
セクション14.15.2「InnoDB モニターの有効化」
セクション14.5.7「InnoDB ログファイルの数またはサイズの変更、および InnoDB テーブルスペースのサイズの変更」
セクション13.2.5.3「INSERT ... ON DUPLICATE KEY UPDATE 構文」
セクション13.2.5.2「INSERT DELAYED 構文」
セクション13.2.5「INSERT 構文」
セクション13.7.6.4「KILL 構文」
セクション18.2.2「Linux での MySQL Cluster のインストール」
セクション18.2.2.1「Linux での MySQL Cluster バイナリリリースのインストール」
セクション18.2.2.3「Linux でのソースからの MySQL Cluster のビルド」
セクション13.2.6「LOAD DATA INFILE 構文」
セクション6.1.6「LOAD DATA LOCAL のセキュリティーの問題」
セクション2.3.6「Microsoft Windows MySQL Server インストールのトラブルシューティング」
セクション2.3.1「Microsoft Windows 上での MySQL のインストールレイアウト」
セクション15.2「MyISAM ストレージエンジン」
セクション7.6.3「MyISAM テーブルの修復方法」
セクション15.2.4.1「MyISAM テーブルの破損」
セクション7.6.5「MyISAM テーブル保守スケジュールのセットアップ」
セクション15.2.1「MyISAM 起動オプション」
セクション4.6.3「myisamchk — MyISAM テーブルメンテナンスユーティリティー」
セクション4.6.3.2「myisamchk のチェックオプション」
セクション4.6.3.1「myisamchk の一般オプション」
セクション4.6.5「myisampack — 圧縮された読み取り専用の MyISAM テーブルの生成」
セクションA.10「MySQL 5.6 FAQ: MySQL Cluster」
セクションA.11「MySQL 5.6 FAQ: MySQL の中国語、日本語、および韓国語の文字セット」
セクションA.3「MySQL 5.6 FAQ: サーバー SQL モード」
セクションA.1「MySQL 5.6 FAQ: 全般」
MySQL Cluster Auto-Installer の「Define Processes」画面
MySQL Cluster Auto-Installer の「Deploy Cluster」画面
セクション18.2.1.1「MySQL Cluster Auto-Installer の要件」
第18章「MySQL Cluster NDB 7.3 および MySQL Cluster NDB 7.4」
セクション18.5.4「MySQL Cluster での MySQL サーバーの使用法」
セクション18.5.11.3「MySQL Cluster と MySQL のセキュリティー手順」
セクション18.1.6.8「MySQL Cluster に限定された問題」
セクション18.3.4.1「MySQL Cluster の mysqld オプションおよび変数のリファレンス」
セクション18.2「MySQL Cluster のインストール」
セクション18.3.4.3「MySQL Cluster のシステム変数」
セクション18.3.4.4「MySQL Cluster のステータス変数」
セクション18.1.2「MySQL Cluster のノード、ノードグループ、レプリカ、およびパーティション」
セクション18.5.5「MySQL Cluster のローリング再起動の実行」
セクション18.1.1「MySQL Cluster の主な概念」
セクション18.2.4「MySQL Cluster の初期構成」
セクション18.2.5「MySQL Cluster の初期起動」
セクション18.3.2.2「MySQL Cluster の推奨される初期構成」
セクション18.1「MySQL Cluster の概要」
セクション18.3「MySQL Cluster の構成」
セクション18.5「MySQL Cluster の管理」
セクション18.3.2.6「MySQL Cluster データノードの定義」
セクション18.4「MySQL Cluster プログラム」
セクション18.4.27「MySQL Cluster プログラムに共通するオプション — MySQL Cluster プログラムに共通するオプション」
セクション18.6「MySQL Cluster レプリケーション」
セクション18.6.10「MySQL Cluster レプリケーション: マルチマスターと循環レプリケーション」
セクション18.6.2「MySQL Cluster レプリケーションの一般要件」
セクション18.6.3「MySQL Cluster レプリケーションの既知の問題」
セクション18.6.11「MySQL Cluster レプリケーションの競合解決」
セクション18.6.6「MySQL Cluster レプリケーションの起動 (レプリケーションチャネルが 1 つ)」
セクション18.6.8「MySQL Cluster レプリケーションを使用したフェイルオーバーの実装」
セクション18.6.4「MySQL Cluster レプリケーションスキーマとテーブル」
セクション18.3.2.7「MySQL Cluster 内の SQL ノードおよびその他の API ノードの定義」
セクション18.3.2.1「MySQL Cluster 構成の基本的な例」
セクション18.3.3「MySQL Cluster 構成パラメータの概要」
セクション18.3.4.2「MySQL Cluster 用の MySQL Server オプション」
セクション18.5.2「MySQL Cluster 管理クライアントのコマンド」
セクション5.1「MySQL Server」
セクション10.6「MySQL Server でのタイムゾーンのサポート」
セクション5.2「MySQL Server ログ」
セクションB.5.4.4「MySQL が一時ファイルを格納する場所」
セクションB.5.4.2「MySQL が繰り返しクラッシュする場合の対処方法」
セクション8.4.3.1「MySQL でのテーブルのオープンとクローズの方法」
セクション2.11.1「MySQL のアップグレード」
セクション1.8.5「MySQL のサポータ」
セクション24.1.1「MySQL のスレッド」
セクション24.4「MySQL のデバッグおよび移植」
セクション8.11.4.1「MySQL のメモリーの使用方法」
セクション12.9.6「MySQL の全文検索の微調整」

セクション1.7「MySQL の標準への準拠」
セクション24.3「MySQL への新しい関数の追加」
セクション2.10.1.2「MySQL を自動的に起動および停止する」
セクション6.1.5「MySQL を通常ユーザーとして実行する方法」
セクション2.3.5.8「MySQL インストールのテスト」
セクション8.9.3「MySQL クエリーキャッシュ」
セクションB.5.2.9「MySQL サーバーが存在しなくなりました」
セクション23.7.2「MySQL サーバーと MySQL Connector/C の同時インストール」
セクション4.3「MySQL サーバーとサーバー起動プログラム」
セクション24.4.1「MySQL サーバーのデバッグ」
第5章「MySQL サーバーの管理」
セクション2.10.1.3「MySQL サーバーの起動とトラブルシューティング」
セクション2.3.5.3「MySQL サーバータイプの選択」
セクション2.9.4「MySQL ソース構成オプション」
セクション24.1.2「MySQL テストスイート」
セクション4.1「MySQL プログラムの概要」
MySQL 用語集
セクション4.3.3「mysql.server — MySQL サーバー起動スクリプト」
セクション23.7.7.1「mysql_affected_rows()」
セクション4.4.3「mysql_install_db — MySQL データディレクトリの初期化」
セクション2.10.1.1「mysql_install_db 実行の問題」
セクション23.7.7.49「mysql_options()」
セクション4.4.4「mysql_plugin — MySQL サーバープラグインの構成」
セクション4.6.8「mysqlbinlog — バイナリログファイルを処理するためのユーティリティー」
セクション4.5.3「mysqlcheck — テーブル保守プログラム」
セクション4.3.1「mysqld — MySQL サーバー」
セクション24.4.1.6「mysqld でのエラーの原因を見つけるためのサーバーログの使用」
セクション4.3.4「mysqld_multi — 複数の MySQL サーバーの管理」
セクション4.3.2「mysqld_safe — MySQL サーバー起動スクリプト」
セクション4.5.4「mysqldump — データベースバックアッププログラム」
セクション4.6.10「mysqlhotcopy — データベースバックアッププログラム」
セクション18.5.15「NDB API 統計のカウンタと変数」
セクション18.4.4「ndb_mgmd — MySQL Cluster 管理サーバーデーモン」
セクション18.4.20「ndb_restore — MySQL Cluster バックアップのリストア」
セクション18.4.24「ndb_show_tables — NDB テーブルのリストの表示」
セクション18.5.10「ndbinfo MySQL Cluster 情報データベース」
セクション13.7.2.4「OPTIMIZE TABLE 構文」
セクション24.4.1.3「pdb を使用した Windows のクラッシュダンプの作成」
セクション13.7.2.5「REPAIR TABLE 構文」
セクション13.4.2.3「RESET SLAVE 構文」
セクション4.7.4「resolve_stack_dump — 数値スタックトレースダンプをシンボルに解決」
root のパスワードのリセット: UNIX システム
root のパスワードのリセット: 一般的な手順
セクション18.2.2.2「RPM からの MySQL Cluster のインストール」
セクション2.5.5「RPM パッケージを使用して MySQL を Linux にインストールする」
セクション13.2.9.1「SELECT ... INTO 構文」
セクション13.3.6「SET TRANSACTION 構文」
セクション13.7.5.16「SHOW ENGINE 構文」
セクション13.7.5.40「SHOW VARIABLES 構文」
セクション2.7「Solaris および OpenSolaris に MySQL をインストールする」
セクション6.3.10.2「SSL を使用するための MySQL の構成」
Unix 上の MyISAM へのシンボリックリンクの使用
セクション2.10.1「Unix 類似システムでのインストール後の手順」
セクション18.2.3.1「Windows でのバイナリリリースからの MySQL Cluster のインストール」
セクション2.3.5.5「Windows のコマンド行からの MySQL の起動」
セクション2.3.5.7「Windows のサービスとして MySQL を起動する」
セクション5.3.2.1「Windows コマンド行での複数の MySQL インスタンスの起動」
セクション18.2.3.4「Windows サービスとしての MySQL Cluster プロセスのインストール」
セクション5.3.2.2「Windows サービスとして複数の MySQL インスタンスの起動」
セクション2.3.7「Windows 上の MySQL をアップグレードする」
Windows 上のデータベースへのシンボリックリンクの使用
セクション16.5.3「ZFS での MySQL リカバリーの扱い」
セクション12.18「その他の関数」
セクション6.2.7「アクセス拒否エラーの原因」
セクション2.1.2「インストールする MySQL のバージョンと配布の選択」
セクション10.2「エラーメッセージ言語の設定」
セクション5.2.2「エラーログ」
セクション4.2.6「オプションファイルの使用」
セクションB.5.6「オプティマイザ関連の問題」
セクション14.11.1「オンライン DDL の概要」
セクション8.9.3.3「クエリーキャッシュの構成」
セクションB.5.2.4「クライアントは認証プロトコルに対応できません」
セクション7.6.1「クラッシュリカバリへの myisamchk の使用」
セクション9.6「コメントの構文」
セクション5.1.3「サーバーコマンドオプション」
セクション5.1.4「サーバーシステム変数」
セクション5.1.6「サーバーステータス変数」
セクション8.11.2「サーバーパラメータのチューニング」
サーバープラグインのステータス変数およびシステム変数
セクション10.1.3.1「サーバー文字セットおよび照合順序」
セクション5.1.11「シグナルへのサーバー応答」
セクション24.4.1.5「スタックトレースの使用」
セクション5.2.5「スロークエリーログ」
セクション6.1.4「セキュリティー関連の mysqld オプションおよび変数」
セクションB.5.4.6「タイムゾーンの問題」
セクション10.6.1「タイムゾーンの変更による現在の時間の維持」
セクション24.4.1.7「テーブルが破損した場合のテストケースの作成」
セクションB.5.2.19「テーブルの破損の問題」
セクション8.10.2「テーブルロックの問題」

セクション24.4.1.1「デバッグのための MySQL のコンパイル」
セクション24.4.1.2「トレースファイルの作成」
セクション2.4.2「ネイティブパッケージを使用して OS X に MySQL をインストールする」
セクション5.2.4.1「バイナリロギング形式」
セクション5.2.4「バイナリログ」
セクション17.1.4.4「バイナリログのオプションと変数」
セクションB.5.2.10「パケットが大きすぎます」
セクション22.2.2「パフォーマンススキーマ起動構成」
セクションB.5.2.13「ファイルを作成/書き込みできない」
セクションB.5.3.1「ファイル権限の問題」
セクション17.3.6「フェイルオーバー中にマスターを切り替える」
セクション8.2.1.20「フルテーブルスキャンを回避する方法」
セクション24.2.4「プラグインの作成」
セクション4.2.5「プログラムオプション修飾子」
セクションB.5.2.6「ホスト 'host_name' は拒否されました」
セクションB.5.2.15「ユーザーを無視します」
セクション13.7.3.1「ユーザー定義関数のための CREATE FUNCTION 構文」
セクション24.3.2.5「ユーザー定義関数のコンパイルおよびインストール」
セクション24.3.2.6「ユーザー定義関数のセキュリティー上の予防措置」
セクション17.1.4.1「レプリケーション、バイナリロギングオプション、および変数のリファレンス」
セクション17.1.4「レプリケーションおよびバイナリロギングのオプションと変数」
セクション14.18.6「レプリケーションでの InnoDB memcached プラグインの使用」
セクション18.6.5「レプリケーションのための MySQL Cluster の準備」
セクション17.1.4.3「レプリケーションスレーブのオプションと変数」
セクション17.1.4.2「レプリケーションマスターのオプションと変数」
セクション17.2.2「レプリケーションリレーおよびステータスログ」
セクション5.2.3「一般クエリーログ」
セクション8.12.5.2「一般的なスレッドの状態」
セクション2.2「一般的なバイナリを使用した MySQL の Unix/Linux へのインストール」
セクション14.6.2「別のマシンへの InnoDB テーブルの移動またはコピー」
セクションB.5.1「問題の原因を判別する方法」
セクション8.10.5「外部ロック」
セクション12.14「情報関数」
セクションB.5.2.7「接続が多すぎます」
セクション6.1.3「攻撃者に対する MySQL のセキュアな状態の維持」
セクションB.5.2.17「文字セットを初期化できません」
セクション24.3.2「新しいユーザー定義関数の追加」
セクション6.2.2「権限システム付与テーブル」
セクション6.2.6「権限変更が有効化される時期」
セクション5.2.4.3「混合形式のバイナリロギング形式」
セクション2.12「環境変数」
セクション23.6.3「組み込みサーバーとオプション」
セクション1.2「表記規則および構文規則」
セクション9.2.2「識別子の大文字と小文字の区別」
セクション1.6「質問またはバグをレポートする方法」
セクションB.5.2.11「通信エラーおよび中止された接続」
セクション15.2.4.2「適切に閉じられなかったテーブルの問題」
セクションB.5.5.5「非トランザクションテーブルのロールバックの失敗」

mysqld mysqld.trace

セクション24.4.1.2「トレースファイルの作成」

mysqld-5.5

セクション2.11.1「MySQL のアップグレード」

mysqld-debug

セクション2.3.5.3「MySQL サーバータイプの選択」
セクション4.3.2「mysqld_safe — MySQL サーバー起動スクリプト」
セクション5.3.2.1「Windows コマンド行での複数の MySQL インスタンスの起動」
セクション24.4.1.2「トレースファイルの作成」
セクション2.2「一般的なバイナリを使用した MySQL の Unix/Linux へのインストール」

mysqld.exe

セクション18.2.3.3「Windows での MySQL Cluster の初期起動」
セクション18.2.3.1「Windows でのバイナリリリースからの MySQL Cluster のインストール」
セクション18.2.3.4「Windows サービスとしての MySQL Cluster プロセスのインストール」

mysqld_multi

セクション4.1「MySQL プログラムの概要」
セクション4.3.4「mysqld_multi — 複数の MySQL サーバーの管理」
セクション5.3.3「Unix 上での複数の MySQL インスタンスの実行」

mysqld_safe

セクションB.5.2.18「'File' が見つかりません、および同様のエラー」
セクション5.3「1 つのマシン上での複数の MySQL インスタンスの実行」
セクション17.1.3.2「GTID を使用したレプリケーションのセットアップ」
セクション14.19「InnoDB のトラブルシューティング」
セクション14.5.1「InnoDB テーブルスペースの作成」
セクションA.10「MySQL 5.6 FAQ: MySQL Cluster」
セクション18.5.11.3「MySQL Cluster と MySQL のセキュリティー手順」
セクション18.5.13.3「MySQL Cluster データノードのオンライン追加: 詳細な例」
セクション10.6「MySQL Server でのタイムゾーンのサポート」
セクションB.5.4.5「MySQL の UNIX ソケットファイルを保護または変更する方法」
セクション2.10.1.2「MySQL を自動的に起動および停止する」
セクション2.10.1.3「MySQL サーバーの起動とトラブルシューティング」
セクション4.1「MySQL プログラムの概要」
セクション4.3.3「mysql.server — MySQL サーバー起動スクリプト」
セクション4.4.3「mysql_install_db — MySQL データディレクトリの初期化」

セクション4.3.4「mysqld_multi — 複数の MySQL サーバーの管理」
セクション4.3.2「mysqld_safe — MySQL サーバー起動スクリプト」
セクション18.2.2.2「RPM からの MySQL Cluster のインストール」
セクション5.3.3「Unix 上での複数の MySQL インスタンスの実行」
セクション2.10.1「Unix 類似システムでのインストール後の手順」
セクション5.2.2「エラーログ」
セクション4.2.9「オプションのデフォルト、値を想定するオプション、および = 記号」
セクション4.2.6「オプションファイルの使用」
セクション5.1.2.2「サンプルのデフォルトサーバー構成ファイルの使用」
セクション5.1.3「サーバーコマンドオプション」
セクション5.1.4「サーバーシステム変数」
セクション8.11.2「サーバーパラメータのチューニング」
セクションB.5.4.6「タイムゾーンの問題」
セクション24.4.1.1「デバッグのための MySQL のコンパイル」
セクションB.5.2.10「パケットが大きすぎます」
セクションB.5.3.1「ファイル権限の問題」
セクション8.11.4.2「ラージページのサポートの有効化」
セクション2.2「一般的なバイナリを使用した MySQL の Unix/Linux へのインストール」
セクション6.1.3「攻撃者に対する MySQL のセキュアな状態の維持」

mysqldump

セクション11.4.3「BLOB 型と TEXT 型」
セクション13.1.17「CREATE TABLE 構文」
セクション16.4.2「EC2 インスタンスの制限」
セクション14.5.3「File-Per-Table モードの有効化および無効化」
セクション17.1.3.4「GTID ベースレプリケーションの制約」
セクション4.6.1「innochecksum — オフライン InnoDB ファイルチェックサムユーティリティー」
セクション14.16「InnoDB のバックアップとリカバリ」
セクション8.5.4「InnoDB テーブルの一括データロード」
セクション14.6.1「InnoDB テーブルの作成」
セクション8.9.1「InnoDB バッファープール」
セクション14.5.7「InnoDB ログファイルの数またはサイズの変更、および InnoDB テーブルスペースのサイズの変更」
セクション13.2.6「LOAD DATA INFILE 構文」
セクション13.2.7「LOAD XML 構文」
セクション2.11.1.3「MySQL 5.5 から 5.6 へのアップグレード」
セクション2.11.2.1「MySQL 5.5 へのダウングレード」
セクションA.10「MySQL 5.6 FAQ: MySQL Cluster」
セクション1.4「MySQL 5.6 の新機能」
セクション18.5.3「MySQL Cluster のオンラインバックアップ」
セクション18.5.5「MySQL Cluster のローリング再起動の実行」
セクション18.1「MySQL Cluster の概要」
セクション18.5.14「MySQL Cluster の配布された MySQL 権限」
セクション18.6.9.2「MySQL Cluster レプリケーションを使用したポイントインタイムリカバリ」
セクション5.2「MySQL Server ログ」
セクション4.5.1.1「mysql のオプション」
セクション2.11.2「MySQL のダウングレード」
セクション1.3.2「MySQL の主な機能」
セクション1.8.1「MySQL への貢献者」
セクション4.2.2「MySQL サーバーへの接続」
セクション2.3.5.6「MySQL ツールの PATH をカスタマイズする」
セクション2.11.5「MySQL データベースのほかのマシンへのコピー」
セクション4.1「MySQL プログラムの概要」
セクション4.5.4「mysqldump — データベースバックアッププログラム」
セクション7.4.1「mysqldump による SQL フォーマットでのデータのダンプ」
セクション7.4.3「mysqldump による区切りテキストフォーマットでのデータのダンプ」
セクション7.4.5「mysqldump のヒント」
セクション17.1.1.5「mysqldump を使用したデータスナップショットの作成」
セクション17.3.1.1「mysqldump を使用してスレーブをバックアップする」
セクション18.4.20「ndb_restore — MySQL Cluster バックアップのリストア」
セクション18.5.10「ndbinfo MySQL Cluster 情報データベース」
セクション7.4.2「SQL フォーマットバックアップのリロード」
セクション2.6「Unbreakable Linux Network (ULN) を使用した MySQL のインストール」
セクション13.7.3.4「UNINSTALL PLUGIN 構文」
セクション2.3.5.7「Windows のサービスとして MySQL を起動する」
セクション12.11「XML 関数」
セクション11.3.4「YEAR(2) の制限と YEAR(4) への移行」
セクション6.2.7「アクセス拒否エラーの原因」
セクション5.1.7「サーバー SQL モード」
セクション5.1.4「サーバーシステム変数」
セクション5.2.7「サーバーログの保守」
セクション7.4.5.2「サーバー間でのデータベースのコピー」
セクション7.4.5.3「ストアドプログラムのダンプ」
セクション18.2.6「テーブルとデータを含む MySQL Cluster の例」
セクション14.10.4「テーブルのデフラグ」
セクション2.11.4「テーブルまたはインデックスの再作成または修復」
セクション2.11.3「テーブルまたはインデックスの再構築が必要かどうかのチェック」
セクション14.5.5「テーブルスペースの別のサーバーへのコピー (トランスポータブルテーブルスペース)」
セクション7.4.5.4「テーブル定義と内容の個別のダンプ」
セクション7.4.5.1「データベースのコピーの作成」
セクション7.2「データベースバックアップ方法」
セクション4.6.8.3「バイナリログファイルのバックアップのための mysqlbinlog の使用」
セクション7.3「バックアップおよびリカバリ戦略の例」
第7章「バックアップとリカバリ」
セクション7.1「バックアップとリカバリの種類」
セクション7.4「バックアップへの mysqldump の使用」
セクション7.3.1「バックアップポリシーの確立」
セクション7.3.3「バックアップ戦略サマリー」
セクション17.3.1「バックアップ用にレプリケーションを使用する」
セクションD.8「パフォーマンススキーマの制約」
セクションD.5「ビューの制約」
セクション17.1.3.3「フェイルオーバーおよびスケールアウトでの GTID の使用」

セクション4.2.3「プログラムオプションの指定」
セクション17.3.1.3「マスターまたはスレーブを読み取り専用にしてバックアップする」
セクション14.18.6「レプリケーションでの InnoDB memcached プラグインの使用」
セクション18.6.5「レプリケーションのための MySQL Cluster の準備」
セクション17.1.1「レプリケーションのセットアップ方法」
セクション17.1.1.4「レプリケーションマスターバイナリログ座標の取得」
セクション17.2.2「レプリケーションリレーおよびステータスログ」
セクション17.1.1.6「ローデータファイルを使用したデータスナップショットの作成」
セクションB.5.5.7「一致する行がない場合の問題の解決」
セクション5.2.1「一般クエリーログおよびスロークエリーログの出力先の選択」
セクション10.1.11「以前の Unicode サポートから現在の Unicode サポートへのアップグレード」
セクション14.6.2「別のマシンへの InnoDB テーブルの移動またはコピー」
セクション7.4.4「区切りテキストフォーマットバックアップのリロード」
セクション1.7.2.4「外部キーの違い」
セクション13.1.17.2「外部キー制約の使用」
セクション17.1.1.8「既存のデータによるレプリケーションのセットアップ」
セクションB.5.5.8「浮動小数点値に関する問題」
セクション17.3.4「異なるデータベースを異なるスレーブに複製する」
セクション17.3.2「異なるマスターおよびスレーブストレージエンジンでレプリケーションを使用する」
セクション9.2.3「識別子とファイル名のマッピング」
セクション9.2.2「識別子の大文字と小文字の区別」
セクション1.6「質問またはバグをレポートする方法」

mysqldump mysql

セクション6.2.7「アクセス拒否エラーの原因」

mysqldumpslow

セクション4.1「MySQL プログラムの概要」
セクション4.6.9「mysqldumpslow — スロークエリーログファイルの要約」
セクション5.2.5「スロークエリーログ」

mysqlhotcopy

セクション1.4「MySQL 5.6 の新機能」
セクション1.8.1「MySQL への貢献者」
セクション4.1「MySQL プログラムの概要」
セクション4.5.4「mysqldump — データベースバックアッププログラム」
セクション4.6.10「mysqlhotcopy — データベースバックアッププログラム」
セクション7.2「データベースバックアップ方法」
第7章「バックアップとリカバリ」
セクション7.1「バックアップとリカバリの種類」

mysqlimport

セクション13.2.6「LOAD DATA INFILE 構文」
セクション6.1.6「LOAD DATA LOCAL のセキュリティーの問題」
セクション2.11.2「MySQL のダウングレード」
セクション2.11.5「MySQL データベースのほかのマシンへのコピー」
セクション4.1「MySQL プログラムの概要」
セクション4.5.5「mysqlimport — データインポートプログラム」
セクション7.2「データベースバックアップ方法」
セクション7.1「バックアップとリカバリの種類」
セクション7.4.4「区切りテキストフォーマットバックアップのリロード」

MySQLInstallerConsole

セクション2.3.3.2「MySQL Installer Console」

mysqloptimize

セクション4.5.3「mysqlcheck — テーブル保守プログラム」

mysqlrepair

セクション4.5.3「mysqlcheck — テーブル保守プログラム」

mysqlshow

セクション23.7.3「C API クライアントプログラムの例」
セクション2.3.5.8「MySQL インストールのテスト」
セクション4.2.2「MySQL サーバーへの接続」
セクション4.1「MySQL プログラムの概要」
セクション4.5.6「mysqlshow — データベース、テーブル、およびカラム情報の表示」
セクション13.7.5.15「SHOW DATABASES 構文」
セクション13.7.5.23「SHOW INDEX 構文」
セクション13.7.5.37「SHOW TABLE STATUS 構文」
セクション2.3.8「Windows でのインストール後の手順」

mysqlshow db_name

セクション13.7.5.38「SHOW TABLES 構文」

mysqlshow db_name tbl_name

セクション13.7.5.6「SHOW COLUMNS 構文」

mysqlshow mysql user

セクションB.5.2.15「ユーザーを無視します」

mysqlslap

セクション4.1「MySQL プログラムの概要」
セクション4.5.7「mysqlslap — 負荷エミュレーションクライアント」
セクション6.3.8.7「クライアント側のクリアテキスト認証プラグイン」
セクション8.12.3「独自のベンチマークの使用」

mysqltest

セクション24.1.2「MySQL テストスイート」
セクション6.3.8.4「SHA-256 認証プラグイン」
セクション6.3.6「パスワードの期限切れとサンドボックスモード」

N

[索引の先頭]

ndb_blob_tool

セクション18.4.6「ndb_blob_tool — MySQL Cluster テーブルの BLOB および TEXT カラムのチェックおよび修復」

ndb_config

セクション18.4「MySQL Cluster プログラム」
セクション18.4.7「ndb_config — MySQL Cluster 構成情報の抽出」

ndb_cpcd

セクション18.4.8「ndb_cpcd — NDB 開発のためのテストの自動化」

ndb_delete_all

セクション18.4.9「ndb_delete_all — NDB テーブルからのすべての行の削除」

ndb_desc

セクション21.30.1「INFORMATION_SCHEMA FILES テーブル」
セクション19.2.5「KEY パーティショニング」
セクション18.5.11.3「MySQL Cluster と MySQL のセキュリティー手順」
セクション18.5.12.1「MySQL Cluster ディスクデータオブジェクト」
セクション18.5.13.3「MySQL Cluster データノードのオンライン追加: 詳細な例」
セクション18.3.2.6「MySQL Cluster データノードの定義」
セクション18.3.4.2「MySQL Cluster 用の MySQL Server オプション」
セクション18.4.10「ndb_desc — NDB テーブルの表示」
セクション18.5.10.4「ndbinfo cluster_operations テーブル」
セクション18.5.10.20「ndbinfo server_operations テーブル」

ndb_drop_index

セクション18.4.11「ndb_drop_index — NDB テーブルからのインデックスの削除」

ndb_drop_table

セクション18.4.11「ndb_drop_index — NDB テーブルからのインデックスの削除」
セクション18.4.12「ndb_drop_table — NDB テーブルの削除」

ndb_error_reporter

セクション18.4.13「ndb_error_reporter — NDB エラーレポートユーティリティー」

ndb_index_stat

セクション18.4.14「ndb_index_stat — NDB インデックス統計ユーティリティー」

ndb_mgm

セクション18.2.2「Linux での MySQL Cluster のインストール」
セクション18.2.2.1「Linux での MySQL Cluster バイナリリリースのインストール」
セクション18.2.2.3「Linux でのソースからの MySQL Cluster のビルド」
セクションA.10「MySQL 5.6 FAQ: MySQL Cluster」
第18章「MySQL Cluster NDB 7.3 および MySQL Cluster NDB 7.4」
セクション18.5.6「MySQL Cluster で生成されたイベントレポート」
セクション18.5.3「MySQL Cluster のオンラインバックアップ」
セクション18.5.8「MySQL Cluster のシングルユーザーモード」
セクション18.5.11.1「MySQL Cluster のセキュリティーとネットワーク上の問題」
セクション18.1.2「MySQL Cluster のノード、ノードグループ、レプリカ、およびパーティション」
セクション18.5.6.1「MySQL Cluster のロギング管理コマンド」
セクション18.5.5「MySQL Cluster のローリング再起動の実行」
セクション18.1.1「MySQL Cluster の主な概念」
セクション18.2.5「MySQL Cluster の初期起動」
セクション18.2.7「MySQL Cluster の安全なシャットダウンと再起動」
セクション18.5.14「MySQL Cluster の配布された MySQL 権限」
セクション18.5.13.1「MySQL Cluster データノードのオンライン追加: 一般的な問題」
セクション18.5.13.3「MySQL Cluster データノードのオンライン追加: 詳細な例」
セクション18.3.2.6「MySQL Cluster データノードの定義」
セクション18.4「MySQL Cluster プログラム」
セクション18.6.9.2「MySQL Cluster レプリケーションを使用したポイントインタイムリカバリ」
セクション18.3.4.2「MySQL Cluster 用の MySQL Server オプション」
セクション18.5.2「MySQL Cluster 管理クライアントのコマンド」
セクション18.5.3.2「MySQL Cluster 管理クライアントを使用したバックアップの作成」
セクション18.4.5「ndb_mgm — MySQL Cluster 管理クライアント」
セクション18.4.4「ndb_mgmd — MySQL Cluster 管理サーバーデーモン」
セクション18.4.20「ndb_restore — MySQL Cluster バックアップのリストア」
セクション18.5.10.1「ndbinfo arbitrator_validity_detail テーブル」
セクション18.5.10.15「ndbinfo membership テーブル」
セクション18.5.10.16「ndbinfo memoryusage テーブル」
セクション18.5.10.18「ndbinfo nodes テーブル」
セクション18.5.10.24「ndbinfo transporters テーブル」
セクション18.2.2.2「RPM からの MySQL Cluster のインストール」
セクション18.2.3.3「Windows での MySQL Cluster の初期起動」

ndb_mgm.exe

セクション18.2.3.3「Windows での MySQL Cluster の初期起動」
セクション18.2.3.1「Windows でのバイナリリリースからの MySQL Cluster のインストール」

ndb_mgmd

セクション18.2.2「Linux での MySQL Cluster のインストール」
セクション18.2.2.1「Linux での MySQL Cluster バイナリリリースのインストール」
セクション18.2.2.3「Linux でのソースからの MySQL Cluster のビルド」
セクションA.10「MySQL 5.6 FAQ: MySQL Cluster」
MySQL Cluster Auto-Installer の「Define Processes」画面

セクション18.1.2「MySQL Cluster のノード、ノードグループ、レプリカ、およびパーティション」
セクション18.5.6.1「MySQL Cluster のロギング管理コマンド」
セクション18.5.5「MySQL Cluster のローリング再起動の実行」
セクション18.1.1「MySQL Cluster の主な概念」
セクション18.2.5「MySQL Cluster の初期起動」
セクション18.2.7「MySQL Cluster の安全なシャットダウンと再起動」
セクション18.3.2.3「MySQL Cluster の接続文字列」
セクション18.3.1「MySQL Cluster の簡易テストセットアップ」
セクション18.5.1「MySQL Cluster の起動フェーズのサマリー」
セクション18.3.2.6「MySQL Cluster データノードの定義」
セクション18.4「MySQL Cluster プログラム」
セクション18.3.2.1「MySQL Cluster 構成の基本的な例」
セクション18.3.3「MySQL Cluster 構成パラメータの概要」
セクション18.3.4.2「MySQL Cluster 用の MySQL Server オプション」
セクション18.5.2「MySQL Cluster 管理クライアントのコマンド」
セクション18.3.2.5「MySQL Cluster 管理サーバーの定義」
セクション2.9.4「MySQL ソース構成オプション」
セクション18.4.4「ndb_mgmd — MySQL Cluster 管理サーバーデーモン」
セクション18.4.1「ndbd — MySQL Cluster データノードデーモン」
セクション18.2.2.2「RPM からの MySQL Cluster のインストール」
セクション18.2.3.3「Windows での MySQL Cluster の初期起動」
セクション18.2.3.1「Windows でのバイナリリリースからの MySQL Cluster のインストール」

ndb_mgmd.exe

セクション18.2.3.3「Windows での MySQL Cluster の初期起動」
セクション18.2.3.1「Windows でのバイナリリリースからの MySQL Cluster のインストール」
セクション18.2.3.4「Windows サービスとしての MySQL Cluster プロセスのインストール」

ndb_print_backup_file

セクション18.4.27「MySQL Cluster プログラムに共通するオプション — MySQL Cluster プログラムに共通するオプション」
セクション18.4.15「ndb_print_backup_file — NDB バックアップファイルの内容の出力」
セクション18.4.17「ndb_print_schema_file — NDB スキーマファイル内容の出力」
セクション18.4.18「ndb_print_sys_file — NDB システムファイル内容の出力」
セクション18.4.19「ndbd_redo_log_reader — クラスタ Redo ログ内容のチェックおよび出力」

ndb_print_file

セクション18.4.16「ndb_print_file — NDB ディスクデータファイル内容の出力」

ndb_print_schema_file

セクション18.4.27「MySQL Cluster プログラムに共通するオプション — MySQL Cluster プログラムに共通するオプション」
セクション18.4.15「ndb_print_backup_file — NDB バックアップファイルの内容の出力」
セクション18.4.16「ndb_print_file — NDB ディスクデータファイル内容の出力」
セクション18.4.17「ndb_print_schema_file — NDB スキーマファイル内容の出力」
セクション18.4.18「ndb_print_sys_file — NDB システムファイル内容の出力」
セクション18.4.19「ndbd_redo_log_reader — クラスタ Redo ログ内容のチェックおよび出力」

ndb_print_sys_file

セクション18.4.27「MySQL Cluster プログラムに共通するオプション — MySQL Cluster プログラムに共通するオプション」
セクション18.4.15「ndb_print_backup_file — NDB バックアップファイルの内容の出力」
セクション18.4.16「ndb_print_file — NDB ディスクデータファイル内容の出力」
セクション18.4.17「ndb_print_schema_file — NDB スキーマファイル内容の出力」
セクション18.4.18「ndb_print_sys_file — NDB システムファイル内容の出力」

ndb_restore

セクションA.10「MySQL 5.6 FAQ: MySQL Cluster」
セクション18.5.3「MySQL Cluster のオンラインバックアップ」
セクション18.5.8「MySQL Cluster のシングルユーザーモード」
セクション18.5.5「MySQL Cluster のローリング再起動の実行」
セクション18.1.1「MySQL Cluster の主な概念」
セクション18.1「MySQL Cluster の概要」
セクション18.5.14「MySQL Cluster の配布された MySQL 権限」
セクション18.3.2.6「MySQL Cluster データノードの定義」
セクション18.4「MySQL Cluster プログラム」
セクション18.6.10「MySQL Cluster レプリケーション: マルチマスターと循環レプリケーション」
セクション18.6.9「MySQL Cluster レプリケーションを使用した MySQL Cluster バックアップ」
セクション18.6.9.2「MySQL Cluster レプリケーションを使用したポイントインタイムリカバリ」
セクション18.6.4「MySQL Cluster レプリケーションスキーマとテーブル」
セクション18.4.20「ndb_restore — MySQL Cluster バックアップのリストア」
セクション7.1「バックアップとリカバリの種類」

ndb_schema_backup_file

セクション18.4.17「ndb_print_schema_file — NDB スキーマファイル内容の出力」

ndb_select_all

セクション18.5.11.3「MySQL Cluster と MySQL のセキュリティー手順」

セクション18.4.21「ndb_select_all — NDB テーブルの行の出力」
セクション18.4.24「ndb_show_tables — NDB テーブルのリストの表示」

ndb_select_count

セクション18.4.22「ndb_select_count — NDB テーブルの行数の出力」

ndb_setup.py

MySQL Cluster Auto-Installer の「ようこそ」画面
セクション18.2.1.2「MySQL Cluster Auto-Installer の概要」
セクション18.2.1.1「MySQL Cluster Auto-Installer の要件」
MySQL Cluster Auto-Installer の起動
セクション18.4「MySQL Cluster プログラム」
セクション18.4.23「ndb_setup.py — MySQL Cluster のブラウザベース自動インストーラの開始」

ndb_show_tables

セクション18.5.11.3「MySQL Cluster と MySQL のセキュリティー手順」
セクション18.4「MySQL Cluster プログラム」
セクション18.6.4「MySQL Cluster レプリケーションスキーマとテーブル」
セクション18.3.4.2「MySQL Cluster 用の MySQL Server オプション」
セクション18.4.24「ndb_show_tables — NDB テーブルのリストの表示」
セクション18.5.10.4「ndbinfo cluster_operations テーブル」
セクション18.5.10.20「ndbinfo server_operations テーブル」

ndb_size.pl

セクションA.10「MySQL 5.6 FAQ: MySQL Cluster」
セクション18.3.4.2「MySQL Cluster 用の MySQL Server オプション」
セクション18.4.25「ndb_size.pl — NDBCLUSTER サイズ要件エスティメータ」
セクション11.7「データ型のストレージ要件」

ndb_waiter

セクション18.4.26「ndb_waiter — MySQL Cluster が指定したステータスになるまで待機する」

ndbd

セクション18.2.2「Linux での MySQL Cluster のインストール」
セクション18.2.2.1「Linux での MySQL Cluster バイナリリリースのインストール」
セクション18.2.2.3「Linux でのソースからの MySQL Cluster のビルド」
セクションA.10「MySQL 5.6 FAQ: MySQL Cluster」
MySQL Cluster Auto-Installer の「Define Processes」画面
セクション18.3.2.11「MySQL Cluster での SCI トランスポート接続」
セクション18.3.5「MySQL Cluster での高速インターコネクトの使用」
セクション18.2「MySQL Cluster のインストール」
セクション18.3.5.2「MySQL Cluster のインターコネクトとパフォーマンス」
セクション18.1.2「MySQL Cluster のノード、ノードグループ、レプリカ、およびパーティション」
セクション18.5.5「MySQL Cluster のローリング再起動の実行」
セクション18.1.1「MySQL Cluster の主な概念」
セクション18.2.5「MySQL Cluster の初期起動」
セクション18.2.7「MySQL Cluster の安全なシャットダウンと再起動」
セクション18.3.2.2「MySQL Cluster の推奨される初期構成」
セクション18.5「MySQL Cluster の管理」
セクション18.3.1「MySQL Cluster の簡易テストセットアップ」
セクション18.5.1「MySQL Cluster の起動フェーズのサマリー」
セクション18.5.13.2「MySQL Cluster データノードのオンライン追加: 基本的な手順」
セクション18.5.13.3「MySQL Cluster データノードのオンライン追加: 詳細な例」
セクション18.3.2.6「MySQL Cluster データノードの定義」
セクション18.3.3.1「MySQL Cluster データノードの構成パラメータ」
セクション18.4「MySQL Cluster プログラム」
セクション18.6.9「MySQL Cluster レプリケーションを使用した MySQL Cluster バックアップ」
セクション18.6.9.2「MySQL Cluster レプリケーションを使用したポイントインタイムリカバリ」
セクション18.3.2.1「MySQL Cluster 構成の基本的な例」
セクション18.3.3「MySQL Cluster 構成パラメータの概要」
セクション18.3.4.2「MySQL Cluster 用の MySQL Server オプション」
セクション18.5.6.3「MySQL Cluster 管理クライアントでの CLUSTERLOG STATISTICS の使用」
セクション18.5.2「MySQL Cluster 管理クライアントのコマンド」
セクション18.4.4「ndb_mgmd — MySQL Cluster 管理サーバーデーモン」
セクション18.4.20「ndb_restore — MySQL Cluster バックアップのリストア」
セクション18.4.26「ndb_waiter — MySQL Cluster が指定したステータスになるまで待機する」
セクション18.4.1「ndbd — MySQL Cluster データノードデーモン」
セクション18.5.10.18「ndbinfo nodes テーブル」
セクション18.4.3「ndbmtd — MySQL Cluster データノードデーモン (マルチスレッド)」
セクション18.2.2.2「RPM からの MySQL Cluster のインストール」

ndbd.exe

セクション18.2.3.3「Windows での MySQL Cluster の初期起動」
セクション18.2.3.1「Windows でのバイナリリリースからの MySQL Cluster のインストール」
セクション18.2.3.4「Windows サービスとしての MySQL Cluster プロセスのインストール」

ndbd_redo_log_reader

セクション18.4.19「ndbd_redo_log_reader — クラスタ Redo ログ内容のチェックおよび出力」

ndbinfo_select_all

セクション18.4.2「ndbinfo_select_all — ndbinfo テーブルからの選択」

ndbmtd
セクション18.2.2「Linux での MySQL Cluster のインストール」
セクション18.2.2.1「Linux での MySQL Cluster バイナリリリースのインストール」
セクション18.2.2.3「Linux でのソースからの MySQL Cluster のビルド」
MySQL Cluster Auto-Installer の「Define Processes」画面
セクション18.1.2「MySQL Cluster のノード、ノードグループ、レプリカ、およびパーティション」
セクション18.5.5「MySQL Cluster のローリング再起動の実行」
セクション18.1.1「MySQL Cluster の主な概念」
セクション18.2.7「MySQL Cluster の安全なシャットダウンと再起動」
セクション18.3.2.2「MySQL Cluster の推奨される初期構成」
セクション18.5.13.2「MySQL Cluster データノードのオンライン追加: 基本的な手順」
セクション18.5.13.3「MySQL Cluster データノードのオンライン追加: 詳細な例」
セクション18.3.2.6「MySQL Cluster データノードの定義」
セクション18.3.3.1「MySQL Cluster データノードの構成パラメータ」
セクション18.4「MySQL Cluster プログラム」
セクション18.6.9.2「MySQL Cluster レプリケーションを使用したポイントインタイムリカバリ」
セクション2.9.4「MySQL ソース構成オプション」
セクション18.4.4「ndb_mgmd — MySQL Cluster 管理サーバーデーモン」
セクション18.4.1「ndbd — MySQL Cluster データノードデーモン」
セクション18.5.10.18「ndbinfo nodes テーブル」
セクション18.4.3「ndbmtd — MySQL Cluster データノードデーモン (マルチスレッド)」
セクション18.2.2.2「RPM からの MySQL Cluster のインストール」

ndbmtd.exe
セクション18.2.3.3「Windows での MySQL Cluster の初期起動」
セクション18.2.3.1「Windows でのバイナリリリースからの MySQL Cluster のインストール」
セクション18.2.3.4「Windows サービスとしての MySQL Cluster プロセスのインストール」

NET
セクション2.3.5.7「Windows のサービスとして MySQL を起動する」

NET START
セクション18.2.3.4「Windows サービスとしての MySQL Cluster プロセスのインストール」
セクション5.3.2.2「Windows サービスとして複数の MySQL インスタンスの起動」

net start
セクション18.2.3.4「Windows サービスとしての MySQL Cluster プロセスのインストール」

NET START MySQL
セクション2.3.6「Microsoft Windows MySQL Server インストールのトラブルシューティング」
セクション2.3.5.7「Windows のサービスとして MySQL を起動する」
セクション2.3.7「Windows 上の MySQL をアップグレードする」

NET STOP
セクション18.2.3.4「Windows サービスとしての MySQL Cluster プロセスのインストール」
セクション5.3.2.2「Windows サービスとして複数の MySQL インスタンスの起動」

net stop
セクション18.2.3.4「Windows サービスとしての MySQL Cluster プロセスのインストール」

NET STOP MYSQL
セクション18.2.7「MySQL Cluster の安全なシャットダウンと再起動」

NET STOP MySQL
セクション2.3.5.7「Windows のサービスとして MySQL を起動する」

nm
セクション4.7.4「resolve_stack_dump — 数値スタックトレースダンプをシンボルに解決」
セクション24.4.1.5「スタックトレースの使用」

numactl
セクション18.3.2.6「MySQL Cluster データノードの定義」

O
[索引の先頭]

openssl
セクション6.3.10.5「MySQL での SSL 証明書および鍵の設定」

openssl md5 package_name
セクション2.1.4.1「MD5 チェックサムの確認」

P
[索引の先頭]

perror
セクションB.5.2.18「'File' が見つかりません、および同様のエラー」
セクション7.6.3「MyISAM テーブルの修復方法」
セクションA.10「MySQL 5.6 FAQ: MySQL Cluster」
セクション4.1「MySQL プログラムの概要」
セクション4.8.1「perror — エラーコードの説明」
セクションB.1「エラー情報のソース」
セクション14.19.6「オペレーティングシステムのエラーコード」
セクションB.5.2.13「ファイルを作成/書き込みできない」

pfexec
セクション2.7.2「IPS を使用して MySQL を OpenSolaris にインストールする」
セクション2.2「一般的なバイナリを使用した MySQL の Unix/Linux へのインストール」

PGP
セクション2.1.4.2「GnuPG を使用した署名確認」

ping6
セクション5.1.9.5「ブローカからの IPv6 アドレスの入手」

pkg
セクション2.7.2「IPS を使用して MySQL を OpenSolaris にインストールする」

pkgadd
セクション2.7.1「Solaris PKG を使用して Solaris に MySQL をインストールする」

pkgrm
セクション2.7.1「Solaris PKG を使用して Solaris に MySQL をインストールする」

ppm
セクション2.13「Perl のインストールに関する注釈」

ps
セクション8.11.4.1「MySQL のメモリーの使用方法」
セクション2.10.1.3「MySQL サーバーの起動とトラブルシューティング」
セクション4.6.16「mysql_zap — パターンに一致するプロセスを強制終了」
セクション6.1.2.1「パスワードセキュリティーのためのエンドユーザーガイドライン」
セクションB.5.1「問題の原因を判別する方法」

ps auxw
セクション4.2.2「MySQL サーバーへの接続」

ps xa | grep mysqld
セクションB.5.2.2「[ローカルの] MySQL サーバーに接続できません」

R
[索引の先頭]

rename
セクション5.2.2「エラーログ」
セクション5.2.7「サーバーログの保守」
セクション5.2.3「一般クエリーログ」

replace
セクション4.7.1「msql2mysql — mSQL プログラムを MySQL で使用するために変換」
セクション4.1「MySQL プログラムの概要」
セクション4.8.2「replace — 文字列置換ユーティリティー」
セクション1.7.2.5「コメントの先頭としての「--」」
セクション17.3.3「スケールアウトのためにレプリケーションを使用する」

resolve_stack_dump
セクション4.1「MySQL プログラムの概要」
セクション4.7.4「resolve_stack_dump — 数値スタックトレースダンプをシンボルに解決」
セクション24.4.1.5「スタックトレースの使用」

resolveip
セクション4.1「MySQL プログラムの概要」
セクション4.8.3「resolveip — ホスト名と IP アドレスの解決」

rm
セクション13.4.1.1「PURGE BINARY LOGS 構文」

rpm
セクション2.1.4.4「RPM を使用した署名確認」
セクション2.5.5「RPM パッケージを使用して MySQL を Linux にインストールする」
セクション2.9.2「標準ソース配布を使用して MySQL をインストールする」

rpmbuild
セクション2.9「ソースから MySQL をインストールする」
セクション2.9.2「標準ソース配布を使用して MySQL をインストールする」

rsync
セクション7.1「バックアップとリカバリの種類」
セクション17.1.1.6「ローデータファイルを使用したデータスナップショットの作成」
セクション17.1.1.9「既存のレプリケーション環境への追加スレーブの導入」

S
[索引の先頭]

scp
セクション7.1「バックアップとリカバリの種類」
セクション17.1.1.6「ローデータファイルを使用したデータスナップショットの作成」

sed
セクション3.3.4.7「パターンマッチング」

SELECT
セクション18.2.6「テーブルとデータを含む MySQL Cluster の例」

service
セクション2.5.7「ネイティブソフトウェアリポジトリから MySQL を Linux にインストールする」

Services
セクション2.3.5.7「Windows のサービスとして MySQL を起動する」

set
セクション14.18.7「InnoDB memcached プラグインの内部構造」

setenv
セクション4.2.10「環境変数の設定」

setrlimit
セクション16.6.2「memcached の使用」

setup.bat
MySQL Cluster Auto-Installer の起動

sh
セクションB.5.2.18「'File' が見つかりません、および同様のエラー」
セクション4.2.1「MySQL プログラムの起動」
セクション4.2.10「環境変数の設定」
セクション1.2「表記規則および構文規則」

SHOW
セクション18.3.1「MySQL Cluster の簡易テストセットアップ」
セクション18.4.20「ndb_restore — MySQL Cluster バックアップのリストア」

SHOW ERRORS
セクションA.10「MySQL 5.6 FAQ: MySQL Cluster」

SHOW WARNINGS
セクションA.10「MySQL 5.6 FAQ: MySQL Cluster」

sleep
セクション4.3.2「mysqld_safe — MySQL サーバー起動スクリプト」

ssh
セクション18.5.11.1「MySQL Cluster のセキュリティーとネットワーク上の問題」
セクション16.5.1「ZFS を使用したファイルシステムレプリケーション」

Start>Run>cmd.exe
セクション6.3.10.5「MySQL での SSL 証明書および鍵の設定」

strings
セクション6.1.1「セキュリティーガイドライン」

su root
セクション18.2.2.1「Linux での MySQL Cluster バイナリリリースのインストール」

sudo
セクション18.2.2.1「Linux での MySQL Cluster バイナリリリースのインストール」
セクション4.4.3「mysql_install_db — MySQL データディレクトリの初期化」
セクション2.2「一般的なバイナリを使用した MySQL の Unix/Linux へのインストール」

T
[索引の先頭]

tar
セクション23.7.2「MySQL サーバーと MySQL Connector/C の同時インストール」
セクション2.4「OS X に MySQL をインストールする」
セクション2.5.5「RPM パッケージを使用して MySQL を Linux にインストールする」
セクション2.7.1「Solaris PKG を使用して Solaris に MySQL をインストールする」
セクション2.7「Solaris および OpenSolaris に MySQL をインストールする」
セクション2.13.1「Unix に Perl をインストールする」
セクション2.1.2「インストールする MySQL のバージョンと配布の選択」
セクション17.3.1.2「スレーブからローデータをバックアップする」
セクション2.9「ソースから MySQL をインストールする」
セクション3.3「データベースの作成と使用」
セクション7.1「バックアップとリカバリの種類」
セクション17.1.1.6「ローデータファイルを使用したデータスナップショットの作成」
セクション2.2「一般的なバイナリを使用した MySQL の Unix/Linux へのインストール」
セクション17.1.1.9「既存のレプリケーション環境への追加スレーブの導入」
セクション2.9.2「標準ソース配布を使用して MySQL をインストールする」
セクション1.6「質問またはバグをレポートする方法」

tcpdump
セクション6.1.1「セキュリティーガイドライン」

tcsh
セクションB.5.2.18「'File' が見つかりません、および同様のエラー」
セクション4.2.1「MySQL プログラムの起動」
セクション2.4.1「OS X への MySQL のインストールに関する一般的な注記」
セクション4.2.10「環境変数の設定」
セクション1.2「表記規則および構文規則」

tee
セクション4.5.1.2「mysql コマンド」

Telnet
セクション16.6.4「memcached の統計の取得」

telnet
セクション14.18.2「InnoDB および memcached の統合のアーキテクチャー」
セクション14.18.3.3「InnoDB および memcached 設定の検証」
セクション16.6.4「memcached の統計の取得」
セクション6.1.1「セキュリティーガイドライン」

top
セクションB.5.1「問題の原因を判別する方法」

U
[索引の先頭]

ulimit
セクションB.5.2.18「'File' が見つかりません、および同様のエラー」
セクション16.6.2「memcached の使用」
セクション18.3.2.6「MySQL Cluster データノードの定義」
セクション18.4.27「MySQL Cluster プログラムに共通するオプション — MySQL Cluster プログラムに共通するオプション」
セクション4.3.2「mysqld_safe — MySQL サーバー起動スクリプト」
セクション5.1.3「サーバーコマンドオプション」
セクションB.5.2.10「パケットが大きすぎます」
セクション8.11.4.2「ラージページのサポートの有効化」

update-rc.d
セクション18.2.2.1「Linux での MySQL Cluster バイナリリリースのインストール」

useradd
セクション18.2.2.1「Linux での MySQL Cluster バイナリリリースのインストール」
セクション2.5.5「RPM パッケージを使用して MySQL を Linux にインストールする」
セクション2.7「Solaris および OpenSolaris に MySQL をインストールする」
セクション2.2「一般的なバイナリを使用した MySQL の Unix/Linux へのインストール」

usermod
セクション2.5.5「RPM パッケージを使用して MySQL を Linux にインストールする」

V

[索引の先頭]

vi
セクション18.2.4「MySQL Cluster の初期構成」
セクション4.5.1.2「mysql コマンド」
セクション3.3.4.7「パターンマッチング」

vmstat
セクション16.6.2「memcached の使用」

W

[索引の先頭]

WinDbg
セクション24.4.1.3「pdb を使用した Windows のクラッシュダンプの作成」

winMd5Sum
セクション2.1.4.1「MD5 チェックサムの確認」

WinZip
セクション17.3.1.2「スレーブからローデータをバックアップする」
セクション2.9「ソースから MySQL をインストールする」
セクション2.9.2「標準ソース配布を使用して MySQL をインストールする」

WordPad
セクション13.2.6「LOAD DATA INFILE 構文」

Y

[索引の先頭]

yacc
セクション2.9.5「MySQL のコンパイルに関する問題」
セクション9.3「予約語」

yum
セクション16.6.3.3「C および C++ での libmemcached の使用」
セクション16.4.1「EC2 AMI での MySQL のセットアップ」
セクション16.6.1「memcached のインストール」
セクション2.5.5「RPM パッケージを使用して MySQL を Linux にインストールする」
セクション2.5.7「ネイティブソフトウェアリポジトリから MySQL を Linux にインストールする」

yum install MySQL*rpm
セクション2.5.5「RPM パッケージを使用して MySQL を Linux にインストールする」

yum update
セクション2.5.1「MySQL Yum リポジトリを使用して MySQL を Linux にインストールする」
セクション2.5.2「サードパーティーの MySQL 配布を MySQL Yum リポジトリを使用して置換する」

yum update mysql-server
セクション2.5.2「サードパーティーの MySQL 配布を MySQL Yum リポジトリを使用して置換する」

yum-config-manager
セクション2.5.1「MySQL Yum リポジトリを使用して MySQL を Linux にインストールする」
セクション2.5.2「サードパーティーの MySQL 配布を MySQL Yum リポジトリを使用して置換する」

Z

[索引の先頭]

zfs recv
セクション16.5.1「ZFS を使用したファイルシステムレプリケーション」

zip
セクション17.1.1.6「ローデータファイルを使用したデータスナップショットの作成」
セクション1.6「質問またはバグをレポートする方法」

zsh
セクション4.2.10「環境変数の設定」

関数の索引

シンボル | A | B | C | D | E | F | G | H | I | L | M | N | O | P | Q | R | S | T | U | V | W | X | Y

シンボル

[索引の先頭]

%
セクション1.7.1「標準 SQL に対する MySQL 拡張機能」

A

[索引の先頭]

ABS()
セクション24.3「MySQL への新しい関数の追加」
セクション13.7.3.1「ユーザー定義関数のための CREATE FUNCTION 構文」
セクション12.6.2「数学関数」
セクション19.6.3「関数に関連するパーティショニング制限」

ACOS()
セクション12.6.2「数学関数」

add()
セクション16.6.3.1「memcached の基本操作」

ADDDATE()
セクション12.7「日付および時間関数」

addslashes()
セクション6.1.7「クライアントプログラミングのセキュリティーガイドライン」

ADDTIME()
セクション12.7「日付および時間関数」

AES_DECRYPT()
セクション12.17.4「Enterprise Encryption 関数の説明」
セクション8.9.3.1「クエリーキャッシュの動作」
セクション5.1.4「サーバーシステム変数」
セクション12.13「暗号化関数と圧縮関数」

AES_ENCRYPT()
セクション12.17.4「Enterprise Encryption 関数の説明」
セクション8.9.3.1「クエリーキャッシュの動作」
セクション5.1.4「サーバーシステム変数」
セクション12.13「暗号化関数と圧縮関数」

Area()
セクション12.15.7.6「MultiPolygon プロパティー関数」
セクション12.15.7.5「Polygon プロパティー関数」
セクション12.15.7「幾何プロパティー関数」

AsBinary()
セクション12.15.6「幾何形式変換関数」
セクション11.5.3.4「空間データのフェッチ」

ASCII()
セクション13.8.3「HELP 構文」
セクション12.5「文字列関数」

ASIN()
セクション12.6.2「数学関数」

AsText()
セクション12.15.6「幾何形式変換関数」
セクション11.5.3.4「空間データのフェッチ」

AsWKB()
セクション12.15.6「幾何形式変換関数」

AsWKT()
セクション12.15.6「幾何形式変換関数」

ASYMMETRIC_DECRYPT()
セクション12.17.4「Enterprise Encryption 関数の説明」

ASYMMETRIC_DERIVE()
セクション12.17.4「Enterprise Encryption 関数の説明」

ASYMMETRIC_ENCRYPT()
セクション12.17.4「Enterprise Encryption 関数の説明」

ASYMMETRIC_SIGN()
セクション12.17.4「Enterprise Encryption 関数の説明」

ASYMMETRIC_VERIFY()
セクション12.17.4「Enterprise Encryption 関数の説明」

ATAN()
セクション12.6.2「数学関数」

ATAN2()
セクション12.6.2「数学関数」

AVG()
セクション11.4.4「ENUM 型」
セクション12.19.1「GROUP BY (集約) 関数」
セクション1.3.2「MySQL の主な機能」
セクション11.4.5「SET 型」
ルースインデックススキャン
セクション11.1.2「日付と時間型の概要」

B

[索引の先頭]

BENCHMARK()
セクション13.2.10.8「FROM 句内のサブクエリー」
セクション8.9.3.1「クエリーキャッシュの動作」
セクション13.2.10.10「サブクエリーの最適化」
セクション8.12.1「式と関数の速度の測定」
セクション12.14「情報関数」

BIN()
セクション9.1.6「ビットフィールドリテラル」
セクション12.5「文字列関数」

BIT_AND()
セクション12.19.1「GROUP BY (集約) 関数」
セクション1.7.1「標準 SQL に対する MySQL 拡張機能」

BIT_COUNT()
セクション12.12「ビット関数」
セクション1.7.1「標準 SQL に対する MySQL 拡張機能」

BIT_LENGTH()
セクション12.5「文字列関数」

BIT_OR()
セクション12.19.1「GROUP BY (集約) 関数」
セクション1.7.1「標準 SQL に対する MySQL 拡張機能」

BIT_XOR()
セクション12.19.1「GROUP BY (集約) 関数」
セクション1.7.1「標準 SQL に対する MySQL 拡張機能」

Buffer()
セクション12.15.8「空間演算子関数」

C

[索引の先頭]

CAST()
セクション9.1.4「16 進数リテラル」
セクション10.1.7.7「BINARY 演算子」
セクション10.1.9.2「CONVERT() と CAST()」
セクション11.3.1「DATE、DATETIME、および TIMESTAMP 型」
セクション1.7.2「MySQL と標準 SQL との違い」
セクション12.10「キャスト関数と演算子」
セクション9.1.6「ビットフィールドリテラル」
セクション9.4「ユーザー定義変数」
セクション12.2「式評価での型変換」
セクション12.7「日付および時間関数」
セクション11.3.7「日付と時間型間での変換」
セクション12.3.2「比較関数と演算子」
セクション10.1.9.1「結果文字列」

CEIL()
セクション12.6.2「数学関数」

CEILING()
セクション19.2.4.1「LINEAR HASH パーティショニング」
セクション12.6.2「数学関数」
セクション19.6.3「関数に関連するパーティショニング制限」

Centroid()
セクション12.15.7.6「MultiPolygon プロパティー関数」

CHAR()
セクション12.10「キャスト関数と演算子」
セクション12.5「文字列関数」
セクション12.13「暗号化関数と圧縮関数」
セクション1.7.1「標準 SQL に対する MySQL 拡張機能」

CHAR_LENGTH()
セクション10.1.14.1「Unicode 文字セット」
セクション12.5「文字列関数」

CHARACTER_LENGTH()
セクション12.5「文字列関数」

CHARSET()
セクション12.14「情報関数」
セクション10.1.9.1「結果文字列」

COALESCE()
セクション13.2.9.2「JOIN 構文」
セクション12.3.2「比較関数と演算子」

COERCIBILITY()
セクション10.1.7.5「式の照合順序」
セクション12.14「情報関数」

COLLATION()
セクション12.14「情報関数」
セクションB.5.5.1「文字列検索での大文字/小文字の区別」
セクション10.1.9.1「結果文字列」

COMPRESS()
セクション2.9.4「MySQL ソース構成オプション」
セクション5.1.4「サーバーシステム変数」
セクション12.13「暗号化関数と圧縮関数」

CONCAT()
セクション12.19.1「GROUP BY (集約) 関数」
セクション21.28「INFORMATION_SCHEMA VIEWS テーブル」
セクション13.7.5.14「SHOW CREATE VIEW 構文」
セクション12.11「XML 関数」
セクション12.10「キャスト関数と演算子」
セクション5.1.7「サーバー SQL モード」
セクション13.7.3.1「ユーザー定義関数のための CREATE FUNCTION 構文」
セクション10.1.7.5「式の照合順序」
セクション12.2「式評価での型変換」
セクション10.1.8「文字列のレパートリー」
セクション12.5「文字列関数」
セクション1.7.1「標準 SQL に対する MySQL 拡張機能」
セクション10.1.9.1「結果文字列」

CONCAT_WS()
セクション12.19.1「GROUP BY (集約) 関数」
セクション12.5「文字列関数」

CONNECTION_ID()
セクション13.7.6.4「KILL 構文」
セクション13.7.5.30「SHOW PROCESSLIST 構文」
セクション8.9.3.1「クエリーキャッシュの動作」
セクション22.9.10.3「スレッドテーブル」
セクション17.1.2.3「バイナリロギングでの安全および安全でないステートメントの判断」

セクション12.14「情報関数」
セクション6.3.12.3「監査ログファイル」

Contains()
セクション1.4「MySQL 5.6 の新機能」
セクション12.15.9「幾何オブジェクト間の空間関係をテストする関数」
セクション12.15.9.2「最小外接矩形 (MBR) を使用する空間関係関数」

CONV()
セクション12.6.2「数学関数」
セクション12.5「文字列関数」
セクション10.1.9.1「結果文字列」

CONVERT()
セクション10.1.9.2「CONVERT() と CAST()」
セクションA.11「MySQL 5.6 FAQ: MySQL の中国語、日本語、および韓国語の文字セット」
セクション12.10「キャスト関数と演算子」
セクション10.1.3.5「文字列リテラルの文字セットおよび照合順序」
セクション12.3.2「比較関数と演算子」

CONVERT_TZ()
セクション8.9.3.1「クエリーキャッシュの動作」
セクション13.3.5.3「テーブルロックの制限と条件」
セクション17.4.1.30「レプリケーションとタイムゾーン」
セクション12.7「日付および時間関数」

COS()
セクション12.6.2「数学関数」

COT()
セクション12.6.2「数学関数」

COUNT()
セクション8.8.2「EXPLAIN 出力フォーマット」
セクション12.19.1「GROUP BY (集約) 関数」
セクション8.4.4「MySQL が内部一時テーブルを使用する仕組み」
セクション8.2.1.2「MySQL の WHERE 句の最適化の方法」
セクション19.1「MySQL のパーティショニングの概要」
セクション1.3.2「MySQL の主な機能」
セクションB.5.5.3「NULL 値に関する問題」
セクション5.1.7「サーバー SQL モード」
セクション20.5.2「ビュー処理アルゴリズム」
セクション13.7.3.1「ユーザー定義関数のための CREATE FUNCTION 構文」
ルースインデックススキャン
セクション20.5.3「更新可能および挿入可能なビュー」
セクション1.7.1「標準 SQL に対する MySQL 拡張機能」
セクション3.3.4.8「行のカウント」

CRC32()
セクション12.6.2「数学関数」

CREATE_ASYMMETRIC_PRIV_KEY()
セクション12.17.2「Enterprise Encryption の使用法と例」
セクション12.17.4「Enterprise Encryption 関数の説明」

CREATE_ASYMMETRIC_PUB_KEY()
セクション12.17.4「Enterprise Encryption 関数の説明」

CREATE_DH_PARAMETERS()
セクション12.17.2「Enterprise Encryption の使用法と例」
セクション12.17.4「Enterprise Encryption 関数の説明」

CREATE_DIGEST()
セクション12.17.4「Enterprise Encryption 関数の説明」

Crosses()
セクション12.15.9「幾何オブジェクト間の空間関係をテストする関数」
セクション12.15.9.2「最小外接矩形 (MBR) を使用する空間関係関数」

crypt()
セクション5.1.4「サーバーシステム変数」
セクション12.13「暗号化関数と圧縮関数」

CURDATE()
セクション8.9.3.1「クエリーキャッシュの動作」
セクション17.1.2.3「バイナリロギングでの安全および安全でないステートメントの判断」
セクション12.7「日付および時間関数」
セクション3.3.4.5「日付の計算」

CURRENT_DATE
セクション13.1.17「CREATE TABLE 構文」
セクション11.6「データ型デフォルト値」
セクション12.7「日付および時間関数」

CURRENT_DATE()
セクション8.9.3.1「クエリーキャッシュの動作」
セクション17.1.2.3「バイナリロギングでの安全および安全でないステートメントの判断」
セクション12.7「日付および時間関数」
セクション11.3.7「日付と時間型間での変換」

CURRENT_TIME
セクション12.7「日付および時間関数」

CURRENT_TIME()
セクション8.9.3.1「クエリーキャッシュの動作」
セクション17.1.2.3「バイナリロギングでの安全および安全でないステートメントの判断」
セクション12.7「日付および時間関数」

CURRENT_TIMESTAMP
セクション13.1.11「CREATE EVENT 構文」
セクション13.1.17「CREATE TABLE 構文」
セクション11.3.5「TIMESTAMP および DATETIME の自動初期化および更新機能」
セクション11.6「データ型デフォルト値」
セクション12.7「日付および時間関数」

CURRENT_TIMESTAMP()
セクション11.3.5「TIMESTAMP および DATETIME の自動初期化および更新機能」

セクション8.9.3.1「クエリーキャッシュの動作」
セクション17.1.2.3「バイナリロギングでの安全および安全でないステートメントの判断」
セクション12.7「日付および時間関数」

CURRENT_USER

セクション13.1.11「CREATE EVENT 構文」
セクション13.1.15「CREATE PROCEDURE および CREATE FUNCTION 構文」
セクション13.1.19「CREATE TRIGGER 構文」
セクション13.1.20「CREATE VIEW 構文」
セクション17.4.1.7「CURRENT_USER() のレプリケーション」
セクション6.2.3「アカウント名の指定」
セクション20.6「ストアドプログラムおよびビューのアクセスコントロール」
セクション17.4.1.15「レプリケーションとシステム関数」
セクション12.14「情報関数」
セクション6.2.2「権限システム付与テーブル」
セクション5.2.4.3「混合形式のバイナリロギング形式」

CURRENT_USER()

セクション13.1.11「CREATE EVENT 構文」
セクション13.1.15「CREATE PROCEDURE および CREATE FUNCTION 構文」
セクション13.1.19「CREATE TRIGGER 構文」
セクション13.1.20「CREATE VIEW 構文」
セクション17.4.1.7「CURRENT_USER() のレプリケーション」
セクション13.7.1.7「SET PASSWORD 構文」
セクション6.3.13「SQL ベースの MySQL アカウントアクティビティーの監査」
セクション6.2.3「アカウント名の指定」
セクション6.2.4「アクセス制御、ステージ 1: 接続の検証」
サーバー側認証プラグインの作成
セクション6.3.9「プロキシユーザー」
セクション10.1.12「メタデータ用の UTF-8」
セクション17.4.1.15「レプリケーションとシステム関数」
セクション12.14「情報関数」
セクション5.2.4.3「混合形式のバイナリロギング形式」
認証プラグインでのプロキシユーザーサポートの実装

CURTIME()

セクション10.6「MySQL Server でのタイムゾーンのサポート」
セクション8.9.3.1「クエリーキャッシュの動作」
セクション17.1.2.3「バイナリロギングでの安全および安全でないステートメントの判断」
セクション17.4.1.13「レプリケーションと小数秒サポート」
セクション12.7「日付および時間関数」
セクション11.3.6「時間値での小数秒」

D

[索引の先頭]

DATABASE()

セクション13.1.21「DROP DATABASE 構文」
セクションB.5.8「MySQL の既知の問題」
セクション8.9.3.1「クエリーキャッシュの動作」
セクション3.4「データベースとテーブルに関する情報の取得」
セクション3.3.1「データベースの作成と選択」
セクション17.1.4.4「バイナリログのオプションと変数」
セクション10.1.12「メタデータ用の UTF-8」
セクション12.14「情報関数」

DATE()

セクション12.7「日付および時間関数」

DATE_ADD()

セクション13.1.11「CREATE EVENT 構文」
セクション9.5「式の構文」
セクション12.7「日付および時間関数」
セクション11.3「日付と時間型」
セクション3.3.4.5「日付の計算」
セクション12.6.1「算術演算子」

DATE_FORMAT()

セクション23.7.18「C API プリペアドステートメントの問題」
セクション10.7「MySQL Server のロケールサポート」
セクション5.1.4「サーバーシステム変数」
セクション12.7「日付および時間関数」

DATE_SUB()

セクション12.7「日付および時間関数」
セクション11.3「日付と時間型」

DATEDIFF()

セクション12.7「日付および時間関数」
セクション19.6.3「関数に関連するパーティショニング制限」

DAY()

セクション12.7「日付および時間関数」
セクション19.6.3「関数に関連するパーティショニング制限」

DAYNAME()

セクション10.7「MySQL Server のロケールサポート」
セクション5.1.4「サーバーシステム変数」
セクション12.7「日付および時間関数」

DAYOFMONTH()

セクション12.7「日付および時間関数」
セクション3.3.4.5「日付の計算」
セクション19.6.3「関数に関連するパーティショニング制限」

DAYOFWEEK()

セクション12.7「日付および時間関数」
セクション19.6.3「関数に関連するパーティショニング制限」

DAYOFYEAR()

セクション19.2「パーティショニングタイプ」
セクション12.7「日付および時間関数」
セクション19.6.3「関数に関連するパーティショニング制限」

DECODE()

セクション12.13「暗号化関数と圧縮関数」

セクション1.7.1「標準 SQL に対する MySQL 拡張機能」

decr()
セクション16.6.3.1「memcached の基本操作」

DEFAULT()
セクション13.2.5「INSERT 構文」
セクション13.2.8「REPLACE 構文」
セクション12.18「その他の関数」
セクション11.6「データ型デフォルト値」

DEGREES()
セクション12.6.2「数学関数」

delete()
セクション16.6.3.1「memcached の基本操作」

DES_DECRYPT()
セクション5.1.3「サーバーコマンドオプション」
セクション12.13「暗号化関数と圧縮関数」

DES_ENCRYPT()
セクション5.1.3「サーバーコマンドオプション」
セクション12.13「暗号化関数と圧縮関数」

Dimension()
セクション12.15.7.1「一般的な幾何プロパティー関数」

Disjoint()
セクション12.15.9.2「最小外接矩形 (MBR) を使用する空間関係関数」

E

[索引の先頭]

ELT()
セクションB.5.8「MySQL の既知の問題」
セクション12.5「文字列関数」
セクション1.7.1「標準 SQL に対する MySQL 拡張機能」
セクション10.1.9.1「結果文字列」

ENCODE()
セクション12.13「暗号化関数と圧縮関数」
セクション1.7.1「標準 SQL に対する MySQL 拡張機能」

ENCRYPT()
セクション1.8.1「MySQL への貢献者」
セクション8.9.3.1「クエリーキャッシュの動作」
セクション5.1.4「サーバーシステム変数」
セクション6.3.1「ユーザー名とパスワード」
セクションD.7「文字セットの制約」
セクション12.13「暗号化関数と圧縮関数」
セクション1.7.1「標準 SQL に対する MySQL 拡張機能」

EndPoint()
セクション12.15.7.3「LineString プロパティー関数」
セクション12.15.7.4「MultiLineString プロパティー関数」
セクション12.15.8「空間演算子関数」

Envelope()
セクション12.15.7.1「一般的な幾何プロパティー関数」
セクション12.15.8「空間演算子関数」

Equals()
セクション12.15.9.2「最小外接矩形 (MBR) を使用する空間関係関数」

EXP()
セクション12.6.2「数学関数」

EXPORT_SET()
セクション12.5「文字列関数」

expr IN ()
セクション12.3.2「比較関数と演算子」

expr NOT IN ()
セクション12.3.2「比較関数と演算子」

ExteriorRing()
セクション12.15.7.5「Polygon プロパティー関数」
セクション12.15.8「空間演算子関数」

EXTRACT()
セクション12.10「キャスト関数と演算子」
セクション12.7「日付および時間関数」
セクション19.6.3「関数に関連するパーティショニング制限」

ExtractValue()
セクション12.11「XML 関数」

F

[索引の先頭]

FIELD()
セクション12.5「文字列関数」

FIND_IN_SET()
セクション11.4.5「SET 型」
セクション12.5「文字列関数」

FLOOR()
セクション12.6.2「数学関数」
セクション12.6.1「算術演算子」
セクション19.6.3「関数に関連するパーティショニング制限」

flush_all
セクション16.6.3.1「memcached の基本操作」

FORMAT()
セクション10.7「MySQL Server のロケールサポート」
セクション12.18「その他の関数」
セクション12.6.2「数学関数」
セクション12.5「文字列関数」

セクション1.7.1「標準 SQL に対する MySQL 拡張機能」
セクション10.1.9.1「結果文字列」

FOUND_ROWS()
セクション8.9.3.1「クエリーキャッシュの動作」
セクション17.1.2.1「ステートメントベースおよび行ベースレプリケーションのメリットとデメリット」
セクション17.1.2.3「バイナリロギングでの安全および安全でないステートメントの判断」
セクション17.4.1.15「レプリケーションとシステム関数」
セクション12.14「情報関数」
セクション5.2.4.3「混合形式のバイナリロギング形式」

FROM_BASE64()
セクション12.5「文字列関数」

FROM_DAYS()
セクション12.7「日付および時間関数」
セクション1.7.1「標準 SQL に対する MySQL 拡張機能」

FROM_UNIXTIME()
セクション1.8.1「MySQL への貢献者」
セクション17.4.1.30「レプリケーションとタイムゾーン」
セクション12.7「日付および時間関数」
セクション6.3.12.4「監査ログプラグインのロギング制御」

G

[索引の先頭]

GeomCollFromText()
セクション12.15.3「WKT 値から幾何値を作成する関数」

GeomCollFromWKB()
セクション12.15.4「WKB 値から幾何値を作成する関数」

GeometryCollection()
セクション12.15.5「幾何値を作成する MySQL 固有の関数」

GeometryCollectionFromText()
セクション12.15.3「WKT 値から幾何値を作成する関数」

GeometryCollectionFromWKB()
セクション12.15.4「WKB 値から幾何値を作成する関数」

GeometryFromText()
セクション12.15.3「WKT 値から幾何値を作成する関数」

GeometryFromWKB()
セクション12.15.4「WKB 値から幾何値を作成する関数」

GeometryN()
セクション12.15.7.7「GeometryCollection プロパティー関数」
セクション12.15.8「空間演算子関数」

GeometryType()
セクション12.15.7.1「一般的な幾何プロパティー関数」

GeomFromText
セクション12.15.7.3「LineString プロパティー関数」

GeomFromText()
セクション12.15.3「WKT 値から幾何値を作成する関数」
セクション12.15.6「幾何形式変換関数」
セクション11.5.3.3「空間カラムへのデータ移入」

GeomFromWKB()
セクション12.15.4「WKB 値から幾何値を作成する関数」
セクション12.15.6「幾何形式変換関数」

get()
セクション16.6.3.1「memcached の基本操作」

GET_FORMAT()
セクション10.7「MySQL Server のロケールサポート」
セクション12.7「日付および時間関数」

GET_LOCK()
セクション13.1.11「CREATE EVENT 構文」
セクション13.7.6.4「KILL 構文」
セクション23.7.7.3「mysql_change_user()」
セクション12.18「その他の関数」
セクション20.4.1「イベントスケジューラの概要」
セクション8.9.3.1「クエリーキャッシュの動作」
セクション17.1.2.1「ステートメントベースおよび行ベースレプリケーションのメリットとデメリット」
セクション13.3.5.3「テーブルロックの制限と条件」
セクション17.1.2.3「バイナリロギングでの安全および安全でないステートメントの判断」
セクション17.4.1.15「レプリケーションとシステム関数」
セクション8.12.5.2「一般的なスレッドの状態」
セクション23.7.16「自動再接続動作の制御」

gethostbyaddr()
セクション8.11.5.2「DNS ルックアップの最適化とホストキャッシュ」

gethostbyaddr_r()
セクション8.11.5.2「DNS ルックアップの最適化とホストキャッシュ」

gethostbyname()
セクション8.11.5.2「DNS ルックアップの最適化とホストキャッシュ」

gethostbyname_r()
セクション8.11.5.2「DNS ルックアップの最適化とホストキャッシュ」

getrusage()
セクション18.5.10.23「ndbinfo threadstat テーブル」

gettimeofday()
セクション18.5.10.23「ndbinfo threadstat テーブル」

GLength()
セクション12.15.7.3「LineString プロパティー関数」

セクション12.15.7.4「MultiLineString プロパティー関数」
セクション12.5「文字列関数」
セクション11.5「空間データの拡張」

GREATEST()
セクション12.3.2「比較関数と演算子」
セクション10.1.9.1「結果文字列」

GROUP_CONCAT()
セクション12.19.1「GROUP BY (集約) 関数」
セクション8.4.4「MySQL が内部一時テーブルを使用する仕組み」
セクション1.3.2「MySQL の主な機能」
セクションB.5.8「MySQL の既知の問題」
セクション5.1.4「サーバーシステム変数」
セクション1.7.1「標準 SQL に対する MySQL 拡張機能」

GTID_SUBSET()
セクション17.1.3.1「GTID の概念」
セクション12.16「グローバルトランザクション ID とともに使用される関数」

GTID_SUBTRACT()
セクション17.1.3.1「GTID の概念」
セクション12.16「グローバルトランザクション ID とともに使用される関数」
セクション17.1.4.5「グローバルトランザクション ID のオプションと変数」

H

[索引の先頭]

HEX()
セクション9.1.4「16 進数リテラル」
セクションA.11「MySQL 5.6 FAQ: MySQL の中国語、日本語、および韓国語の文字セット」
セクション12.18「その他の関数」
セクション12.6.2「数学関数」
セクション10.1.3.5「文字列リテラルの文字セットおよび照合順序」
セクション12.5「文字列関数」
セクション10.1.9.1「結果文字列」

HOUR()
セクション12.7「日付および時間関数」
セクション19.6.3「関数に関連するパーティショニング制限」

I

[索引の先頭]

IF()
セクション13.6.5.2「IF 構文」
セクション14.14.3「InnoDB INFORMATION_SCHEMA システムテーブル」
セクションB.5.8「MySQL の既知の問題」
セクション12.4「制御フロー関数」
セクション1.7.1「標準 SQL に対する MySQL 拡張機能」
セクション10.1.9.1「結果文字列」

IFNULL()
セクションB.5.5.3「NULL 値に関する問題」
セクション12.4「制御フロー関数」

IN
セクション12.3.1「演算子の優先順位」

IN()
セクション8.8.2「EXPLAIN 出力フォーマット」
セクションD.4「サブクエリーの制約」
シングルパートインデックスの range アクセスメソッド
セクション12.2「式評価での型変換」

incr()
セクション16.6.3.1「memcached の基本操作」

INET6_ATON()
セクション5.1.9「IPv6 サポート」
セクション12.18「その他の関数」

INET6_NTOA()
セクション5.1.9「IPv6 サポート」
セクション12.18「その他の関数」

INET_ATON()
セクション5.1.9「IPv6 サポート」
セクション12.18「その他の関数」

INET_NTOA()
セクション5.1.9「IPv6 サポート」
セクション12.18「その他の関数」

INSERT()
セクション12.5「文字列関数」

INSTR()
セクション12.5「文字列関数」
セクション10.1.9.1「結果文字列」

InteriorRingN()
セクション12.15.7.5「Polygon プロパティー関数」
セクション12.15.8「空間演算子関数」

Intersects()
セクション12.15.9.2「最小外接矩形 (MBR) を使用する空間関係関数」

INTERVAL()
セクション12.3.2「比較関数と演算子」

IS_FREE_LOCK()
セクション12.18「その他の関数」
セクション17.1.2.1「ステートメントベースおよび行ベースレプリケーションのメリットとデメリット」
セクション17.1.2.3「バイナリロギングでの安全および安全でないステートメントの判断」
セクション17.4.1.15「レプリケーションとシステム関数」

IS_IPV4()
セクション12.18「その他の関数」

IS_IPV4_COMPAT()
セクション12.18「その他の関数」

IS_IPV4_MAPPED()
セクション12.18「その他の関数」

IS_IPV6()
セクション12.18「その他の関数」

IS_USED_LOCK()
セクション12.18「その他の関数」
セクション17.1.2.1「ステートメントベースおよび行ベースレプリケーションのメリットとデメリット」
セクション17.1.2.3「バイナリロギングでの安全および安全でないステートメントの判断」
セクション17.4.1.15「レプリケーションとシステム関数」

IsClosed()
セクション12.15.7.3「LineString プロパティー関数」
セクション12.15.7.4「MultiLineString プロパティー関数」

IsEmpty()
セクション12.15.7.1「一般的な幾何プロパティー関数」
セクション12.15.9「幾何オブジェクト間の空間関係をテストする関数」

ISNULL()
セクション12.3.2「比較関数と演算子」

IsSimple()
セクション12.15.7.1「一般的な幾何プロパティー関数」

L

[索引の先頭]

LAST_DAY()
セクション12.7「日付および時間関数」

LAST_INSERT_ID()
セクション3.6.9「AUTO_INCREMENT の使用」
セクション13.1.17「CREATE TABLE 構文」
セクション13.2.5.3「INSERT ... ON DUPLICATE KEY UPDATE 構文」
セクション13.2.5.2「INSERT DELAYED 構文」
セクション13.2.5「INSERT 構文」
セクション23.7.7.37「mysql_insert_id()」
セクション23.7.11.16「mysql_stmt_insert_id()」
セクション4.6.8.1「mysqlbinlog 16 進ダンプ形式」
セクション8.9.3.1「クエリーキャッシュの動作」
セクション5.1.4「サーバーシステム変数」
セクション20.2.4「ストアドプロシージャー、関数、トリガー、および LAST_INSERT_ID()」
セクション13.3.5.3「テーブルロックの制限と条件」
セクション1.7.2.3「トランザクションおよびアトミック操作の違い」
セクション17.1.2.3「バイナリロギングでの安全および安全でないステートメントの判断」
セクション17.4.1.1「レプリケーションとAUTO_INCREMENT」
セクション17.4.1.15「レプリケーションとシステム関数」
セクション17.4.4「レプリケーションのトラブルシューティング」
セクション12.14「情報関数」
セクション20.5.3「更新可能および挿入可能なビュー」
セクション23.7.15.3「最後に挿入された行の一意の ID を取得する方法」
セクション1.7.1「標準 SQL に対する MySQL 拡張機能」
セクション12.3.2「比較関数と演算子」
セクション23.7.16「自動再接続動作の制御」

LCASE()
セクション12.5「文字列関数」
セクション10.1.9.1「結果文字列」

LEAST()
セクション12.3.2「比較関数と演算子」
セクション10.1.9.1「結果文字列」

LEFT()
セクション12.10「キャスト関数と演算子」
セクション12.5「文字列関数」

LENGTH()
セクション12.5「文字列関数」

Length()
セクション12.15.7.3「LineString プロパティー関数」
セクション12.15.7.4「MultiLineString プロパティー関数」
セクション11.5「空間データの拡張」

LineFromText()
セクション12.15.3「WKT 値から幾何値を作成する関数」
セクション12.15.6「幾何形式変換関数」

LineFromWKB()
セクション12.15.4「WKB 値から幾何値を作成する関数」
セクション12.15.6「幾何形式変換関数」

LineString
セクション12.15.7.3「LineString プロパティー関数」

LineString()
セクション12.15.5「幾何値を作成する MySQL 固有の関数」

LineStringFromText()
セクション12.15.3「WKT 値から幾何値を作成する関数」

LineStringFromWKB()
セクション12.15.4「WKB 値から幾何値を作成する関数」

LN()
セクション12.6.2「数学関数」

LOAD_FILE()
セクション12.17.2「Enterprise Encryption の使用法と例」

セクション13.2.7「LOAD XML 構文」
セクション6.2.1「MySQL で提供される権限」
セクション8.9.3.1「クエリーキャッシュの動作」
セクション5.1.3「サーバーコマンドオプション」
セクション5.1.4「サーバーシステム変数」
セクション17.1.2.1「ステートメントベースおよび行ベースレプリケーションのメリットとデメリット」
セクション17.1.2.3「バイナリロギングでの安全および安全でないステートメントの判断」
セクション17.4.1.15「レプリケーションとシステム関数」
セクション12.5「文字列関数」
セクション5.2.4.3「混合形式のバイナリロギング形式」

LOCALTIME

セクション11.3.5「TIMESTAMP および DATETIME の自動初期化および更新機能」
セクション12.7「日付および時間関数」

LOCALTIME()

セクション11.3.5「TIMESTAMP および DATETIME の自動初期化および更新機能」
セクション17.1.2.3「バイナリロギングでの安全および安全でないステートメントの判断」
セクション12.7「日付および時間関数」

LOCALTIMESTAMP

セクション11.3.5「TIMESTAMP および DATETIME の自動初期化および更新機能」
セクション12.7「日付および時間関数」

LOCALTIMESTAMP()

セクション11.3.5「TIMESTAMP および DATETIME の自動初期化および更新機能」
セクション17.1.2.3「バイナリロギングでの安全および安全でないステートメントの判断」
セクション12.7「日付および時間関数」

LOCATE()

セクション12.5「文字列関数」

LOG()

セクション19.2.4.1「LINEAR HASH パーティショニング」
セクション12.6.2「数学関数」

LOG10()

セクション12.6.2「数学関数」

LOG2()

セクション12.6.2「数学関数」

LOWER()

セクション10.1.14.1「Unicode 文字セット」
セクション12.10「キャスト関数と演算子」
セクション12.5「文字列関数」
セクション10.1.7.9「照合順序と INFORMATION_SCHEMA 検索」
セクション10.1.9.1「結果文字列」

LPAD()

セクション12.5「文字列関数」

LTRIM()

セクション12.5「文字列関数」
セクション10.1.9.1「結果文字列」

M

[索引の先頭]

MAKE_SET()

セクション12.5「文字列関数」

MAKEDATE()

セクション12.7「日付および時間関数」

MAKETIME()

セクション12.7「日付および時間関数」

MASTER_POS_WAIT()

セクション13.4.2.1「CHANGE MASTER TO 構文」
セクションA.13「MySQL 5.6 FAQ: レプリケーション」
セクション12.18「その他の関数」
セクション8.9.3.1「クエリーキャッシュの動作」
セクション17.1.2.1「ステートメントベースおよび行ベースレプリケーションのメリットとデメリット」
セクション17.1.2.3「バイナリロギングでの安全および安全でないステートメントの判断」

MATCH

セクション9.5「式の構文」

MATCH ()

セクション12.9「全文検索関数」

MATCH()

セクション14.2.13.3「FULLTEXT インデックス」
セクション1.4「MySQL 5.6 の新機能」
セクション12.9.6「MySQL の全文検索の微調整」
MySQL 用語集
セクション12.9.2「ブール全文検索」
セクション12.9.5「全文制限」
セクション12.9「全文検索関数」
セクション12.9.1「自然言語全文検索」

MAX()

セクション3.6.9「AUTO_INCREMENT の使用」
セクション8.8.2「EXPLAIN 出力フォーマット」
セクション12.19.1「GROUP BY (集約) 関数」
セクション21.31.2「INFORMATION_SCHEMA TP_THREAD_GROUP_STATE テーブル」
セクション12.19.3「MySQL での GROUP BY の処理」
セクション8.3.1「MySQL のインデックスの使用の仕組み」
セクション1.3.2「MySQL の主な機能」
セクションB.5.8「MySQL の既知の問題」
セクション8.2.1.7「インデックス拡張の使用」
セクション13.2.10.10「サブクエリーの最適化」
セクション5.1.7「サーバー SQL モード」
セクション20.5.2「ビュー処理アルゴリズム」
ルースインデックススキャン
セクション11.1.1「数値型の概要」
セクション11.3.8「日付での 2 桁の年」
セクション20.5.3「更新可能および挿入可能なビュー」

MBRContains()
セクション12.15.9.3「最小外接矩形 (MBR) を使用する MySQL 固有の空間関係関数」
セクション11.5.3.7「空間インデックスの使用」

MBRDisjoint()
セクション12.15.9.3「最小外接矩形 (MBR) を使用する MySQL 固有の空間関係関数」

MBREqual()
セクション12.15.9.3「最小外接矩形 (MBR) を使用する MySQL 固有の空間関係関数」

MBRIntersects()
セクション12.15.9.3「最小外接矩形 (MBR) を使用する MySQL 固有の空間関係関数」

MBROverlaps()
セクション12.15.9.3「最小外接矩形 (MBR) を使用する MySQL 固有の空間関係関数」

MBRTouches()
セクション12.15.9.3「最小外接矩形 (MBR) を使用する MySQL 固有の空間関係関数」

MBRWithin()
セクション12.15.9.3「最小外接矩形 (MBR) を使用する MySQL 固有の空間関係関数」
セクション11.5.3.7「空間インデックスの使用」

MD5()
セクション19.2.5「KEY パーティショニング」
セクション6.1.2.5「MySQL 4.1 でのパスワードハッシュ変更がアプリケーションプログラムに及ぼす影響」
セクション9.2「スキーマオブジェクト名」
セクション6.1.1「セキュリティーガイドライン」
セクション12.13「暗号化関数と圧縮関数」
セクション1.7.1「標準 SQL に対する MySQL 拡張機能」

MICROSECOND()
セクション12.7「日付および時間関数」
セクション19.6.3「関数に関連するパーティショニング制限」

MID()
セクション12.5「文字列関数」
セクション10.1.9.1「結果文字列」

MIN()
セクション23.7.18「C API プリペアドステートメントの問題」
セクション8.8.2「EXPLAIN 出力フォーマット」
セクション12.19.1「GROUP BY (集約) 関数」
セクション12.19.3「MySQL での GROUP BY の処理」
セクション8.2.1.2「MySQL の WHERE 句の最適化の方法」
セクション8.3.1「MySQL のインデックスの使用の仕組み」
セクション1.3.2「MySQL の主な機能」
セクションB.5.8「MySQL の既知の問題」
セクションB.5.5.3「NULL 値に関する問題」
セクション8.2.1.7「インデックス拡張の使用」
セクション13.2.10.10「サブクエリーの最適化」
セクション20.5.2「ビュー処理アルゴリズム」
ルースインデックススキャン
セクション11.1.1「数値型の概要」
セクション11.3.8「日付での 2 桁の年」
セクション20.5.3「更新可能および挿入可能なビュー」

MINUTE()
セクション12.7「日付および時間関数」
セクション19.6.3「関数に関連するパーティショニング制限」

MLineFromText()
セクション12.15.3「WKT 値から幾何値を作成する関数」

MLineFromWKB()
セクション12.15.4「WKB 値から幾何値を作成する関数」

MOD()
セクション5.1.7「サーバー SQL モード」
セクション12.6.2「数学関数」
セクション3.3.4.5「日付の計算」
セクション1.7.1「標準 SQL に対する MySQL 拡張機能」
セクション12.6.1「算術演算子」
セクション19.6.3「関数に関連するパーティショニング制限」

MONTH()
セクション19.2「パーティショニングタイプ」
セクション12.7「日付および時間関数」
セクション3.3.4.5「日付の計算」
セクション19.6.3「関数に関連するパーティショニング制限」

MONTHNAME()
セクション10.7「MySQL Server の ロケールサポート」
セクション5.1.4「サーバーシステム変数」
セクション12.7「日付および時間関数」

MPointFromText()
セクション12.15.3「WKT 値から幾何値を作成する関数」

MPointFromWKB()
セクション12.15.4「WKB 値から幾何値を作成する関数」

MPolyFromText()
セクション12.15.3「WKT 値から幾何値を作成する関数」

MPolyFromWKB()
セクション12.15.4「WKB 値から幾何値を作成する関数」

MultiLineString()
セクション12.15.5「幾何値を作成する MySQL 固有の関数」

MultiLineStringFromText()
セクション12.15.3「WKT 値から幾何値を作成する関数」

MultiLineStringFromWKB()
セクション12.15.4「WKB 値から幾何値を作成する関数」

MultiPoint()
セクション12.15.5「幾何値を作成する MySQL 固有の関数」

MultiPointFromText()
セクション12.15.3「WKT 値から幾何値を作成する関数」

MultiPointFromWKB()
セクション12.15.4「WKB 値から幾何値を作成する関数」

MultiPolygon()
セクション12.15.5「幾何値を作成する MySQL 固有の関数」

MultiPolygonFromText()
セクション12.15.3「WKT 値から幾何値を作成する関数」

MultiPolygonFromWKB()
セクション12.15.4「WKB 値から幾何値を作成する関数」

my_open()
セクション5.1.6「サーバーステータス変数」

N

[索引の先頭]

NAME_CONST()
セクション12.18「その他の関数」
セクション20.7「ストアドプログラムのバイナリロギング」

NOW()
セクション13.1.15「CREATE PROCEDURE および CREATE FUNCTION 構文」
セクション13.1.17「CREATE TABLE 構文」
セクションA.1「MySQL 5.6 FAQ: 全般」
セクション10.6「MySQL Server でのタイムゾーンのサポート」
セクション11.3.5「TIMESTAMP および DATETIME の自動初期化および更新機能」
セクション11.3.3「YEAR 型」
セクション8.9.3.1「クエリーキャッシュの動作」
セクション5.1.3「サーバーコマンドオプション」
セクション5.1.4「サーバーシステム変数」
セクション17.1.2.1「ステートメントベースおよび行ベースレプリケーションのメリットとデメリット」
セクション10.6.2「タイムゾーンのうるう秒のサポート」
セクション11.6「データ型デフォルト値」
セクション17.1.2.3「バイナリロギングでの安全および安全でないステートメントの判断」
セクション17.4.1.15「レプリケーションとシステム関数」
セクション17.4.1.30「レプリケーションとタイムゾーン」
セクション12.7「日付および時間関数」
セクション11.3.6「時間値での小数秒」

NULLIF()
セクション12.4「制御フロー関数」

NumGeometries()
セクション12.15.7.7「GeometryCollection プロパティー関数」

NumInteriorRings()
セクション12.15.7.5「Polygon プロパティー関数」

NumPoints()
セクション12.15.7.3「LineString プロパティー関数」

O

[索引の先頭]

OCT()
セクション12.5「文字列関数」

OCTET_LENGTH()
セクション12.5「文字列関数」

OLD_PASSWORD()
セクション13.7.1.2「CREATE USER 構文」
セクション6.1.2.5「MySQL 4.1 でのパスワードハッシュ変更がアプリケーションプログラムに及ぼす影響」
セクション2.11.1.3「MySQL 5.5 から 5.6 へのアップグレード」
セクション1.4「MySQL 5.6 の新機能」
セクション6.1.2.4「MySQL でのパスワードハッシュ」
セクション13.7.1.7「SET PASSWORD 構文」
セクションB.5.2.4「クライアントは認証プロトコルに対応できません」
セクション5.1.4「サーバーシステム変数」
セクション24.2.3.8「パスワード検証プラグイン」
セクション6.1.2.6「パスワード検証プラグイン」
セクション12.13「暗号化関数と圧縮関数」

ORD()
セクション12.5「文字列関数」

Overlaps()
セクション12.15.9.2「最小外接矩形 (MBR) を使用する空間関係関数」

P

[索引の先頭]

PASSWORD ()
セクション6.3.8.4「SHA-256 認証プラグイン」

PASSWORD()
セクション13.7.1.2「CREATE USER 構文」
セクション19.2.5「KEY パーティショニング」
セクション6.1.2.5「MySQL 4.1 でのパスワードハッシュ変更がアプリケーションプログラムに及ぼす影響」
セクション2.11.1.3「MySQL 5.5 から 5.6 へのアップグレード」
セクション1.4「MySQL 5.6 の新機能」
セクション6.1.2.4「MySQL でのパスワードハッシュ」
セクション13.7.1.7「SET PASSWORD 構文」
セクション6.3.8.4「SHA-256 認証プラグイン」
セクション6.3.5「アカウントパスワードの割り当て」
セクション6.2.4「アクセス制御、ステージ 1: 接続の検証」

セクション6.2.7「アクセス拒否エラーの原因」
セクション8.9.3.1「クエリーキャッシュの動作」
セクション5.1.3「サーバーコマンドオプション」
セクション5.1.4「サーバーシステム変数」
セクション17.1.2.3「バイナリロギングでの安全および安全でないステートメントの判断」
セクション6.1.2.3「パスワードおよびロギング」
セクション24.2.3.8「パスワード検証プラグイン」
セクション6.1.2.6「パスワード検証プラグイン」
セクションB.5.2.15「ユーザーを無視します」
セクション6.3.2「ユーザーアカウントの追加」
セクション6.3.1「ユーザー名とパスワード」
セクション12.13「暗号化関数と圧縮関数」
セクション2.10.2「最初の MySQL アカウントのセキュリティー設定」
セクション1.7.1「標準 SQL に対する MySQL 拡張機能」

PERIOD_ADD()
セクション12.7「日付および時間関数」
セクション1.7.1「標準 SQL に対する MySQL 拡張機能」

PERIOD_DIFF()
セクション12.7「日付および時間関数」
セクション1.7.1「標準 SQL に対する MySQL 拡張機能」

PI()
セクション12.6.2「数学関数」
セクション9.2.4「関数名の構文解析と解決」

Point()
WKT (Well-Known Text) 形式
セクション12.15.5「幾何値を作成する MySQL 固有の関数」

PointFromText()
セクション12.15.3「WKT 値から幾何値を作成する関数」
セクション12.15.6「幾何形式変換関数」

PointFromWKB()
セクション12.15.4「WKB 値から幾何値を作成する関数」
セクション12.15.6「幾何形式変換関数」

PointN()
セクション12.15.7.3「LineString プロパティー関数」
セクション12.15.8「空間演算子関数」

PolyFromText()
セクション12.15.3「WKT 値から幾何値を作成する関数」

PolyFromWKB()
セクション12.15.4「WKB 値から幾何値を作成する関数」

Polygon()
セクション12.15.5「幾何値を作成する MySQL 固有の関数」

PolygonFromText()
セクション12.15.3「WKT 値から幾何値を作成する関数」

PolygonFromWKB()
セクション12.15.4「WKB 値から幾何値を作成する関数」

POSITION()
セクション12.5「文字列関数」

POW()
セクション19.2.4「HASH パーティショニング」
セクション12.6.2「数学関数」

POWER()
セクション19.2.4.1「LINEAR HASH パーティショニング」
セクション12.6.2「数学関数」

pthread_mutex()
セクション1.8.1「MySQL への貢献者」

Q

[索引の先頭]

QUARTER()
セクション12.7「日付および時間関数」
セクション19.6.3「関数に関連するパーティショニング制限」

QUOTE()
セクション23.7.7.54「mysql_real_escape_string()」
セクション9.1.1「文字列リテラル」
セクション12.5「文字列関数」

R

[索引の先頭]

RADIANS()
セクション12.6.2「数学関数」

RAND()
セクション13.1.15「CREATE PROCEDURE および CREATE FUNCTION 構文」
セクション4.6.8.1「mysqlbinlog 16 進ダンプ形式」
セクション8.9.3.1「クエリーキャッシュの動作」
セクション5.1.4「サーバーシステム変数」
セクション17.1.2.1「ステートメントベースおよび行ベースレプリケーションのメリットとデメリット」
セクション17.1.2.3「バイナリロギングでの安全および安全でないステートメントの判断」
セクション17.4.1.15「レプリケーションとシステム関数」
セクション12.6.2「数学関数」

RANDOM_BYTES()
セクション8.9.3.1「クエリーキャッシュの動作」
セクション12.13「暗号化関数と圧縮関数」

RELEASE_LOCK()
セクション13.2.3「DO 構文」
セクション12.18「その他の関数」
セクション8.9.3.1「クエリーキャッシュの動作」
セクション17.1.2.1「ステートメントベースおよび行ベースレプリケーションのメリットとデメリット」
セクション13.3.5.3「テーブルロックの制限と条件」

セクション17.1.2.3「バイナリロギングでの安全および安全でないステートメントの判断」
セクション17.4.1.15「レプリケーションとシステム関数」

REPEAT()
セクション12.5「文字列関数」
セクション10.1.9.1「結果文字列」

REPLACE()
セクション12.5「文字列関数」
セクション10.1.9.1「結果文字列」

replace()
セクション16.6.3.1「memcached の基本操作」

REVERSE()
セクション12.5「文字列関数」
セクション10.1.9.1「結果文字列」

RIGHT()
セクション12.5「文字列関数」
セクション10.1.9.1「結果文字列」

ROUND()
セクション14.14.3「InnoDB INFORMATION_SCHEMA システムテーブル」
セクション12.20.4「丸め動作」
セクション12.6.2「数学関数」
セクション12.20「高精度計算」
セクション12.20.5「高精度計算の例」

ROW_COUNT()
セクション13.2.1「CALL 構文」
セクション13.2.2「DELETE 構文」
セクション13.2.5「INSERT 構文」
セクション23.7.7.1「mysql_affected_rows()」
セクション17.1.2.3「バイナリロギングでの安全および安全でないステートメントの判断」
セクション17.4.1.15「レプリケーションとシステム関数」
セクション12.14「情報関数」
セクション5.2.4.3「混合形式のバイナリロギング形式」
診断領域の情報項目

RPAD()
セクション12.5「文字列関数」
セクション10.1.9.1「結果文字列」

RTRIM()
セクション12.5「文字列関数」
セクション10.1.9.1「結果文字列」

S

[索引の先頭]

SCHEMA()
セクション12.14「情報関数」

SEC_TO_TIME()
セクション12.7「日付および時間関数」

SECOND()
セクション12.7「日付および時間関数」
セクション19.6.3「関数に関連するパーティショニング制限」

SESSION_USER()
セクション17.1.2.3「バイナリロギングでの安全および安全でないステートメントの判断」
セクション10.1.12「メタデータ用の UTF-8」
セクション12.14「情報関数」

set()
セクション16.6.3.1「memcached の基本操作」

setrlimit()
セクション5.1.3「サーバーコマンドオプション」

SHA()
セクション12.13「暗号化関数と圧縮関数」

SHA1()
セクション6.1.2.5「MySQL 4.1 でのパスワードハッシュ変更がアプリケーションプログラムに及ぼす影響」
セクション6.1.1「セキュリティーガイドライン」
セクション12.13「暗号化関数と圧縮関数」

SHA2()
セクション6.1.2.5「MySQL 4.1 でのパスワードハッシュ変更がアプリケーションプログラムに及ぼす影響」
セクション6.1.1「セキュリティーガイドライン」
セクション12.13「暗号化関数と圧縮関数」

SIGN()
セクション12.6.2「数学関数」

SIN()
セクション24.3.2.3「UDF 引数の処理」
セクション12.6.2「数学関数」

SLEEP()
セクション21.31.3「INFORMATION_SCHEMA TP_THREAD_GROUP_STATS テーブル」
セクション12.18「その他の関数」
セクション8.9.3.1「クエリーキャッシュの動作」
セクション17.1.2.1「ステートメントベースおよび行ベースレプリケーションのメリットとデメリット」
セクション17.1.2.3「バイナリロギングでの安全および安全でないステートメントの判断」
セクション17.4.1「レプリケーションの機能と問題」
セクション8.12.5.2「一般的なスレッドの状態」

SOUNDEX()
セクション24.3「MySQL への新しい関数の追加」
セクション12.5「文字列関数」
セクション10.1.9.1「結果文字列」

SPACE()
セクション12.5「文字列関数」
セクション10.1.9.1「結果文字列」

SQL_THREAD_WAIT_AFTER_GTIDS()
セクション12.16「グローバルトランザクション ID とともに使用される関数」

SQRT()
セクション12.6.2「数学関数」

SRID()
セクション12.15.7.1「一般的な幾何プロパティー関数」

ST_Area()
セクション12.15.7.6「MultiPolygon プロパティー関数」
セクション12.15.7.5「Polygon プロパティー関数」

ST_Centroid()
セクション12.15.7.6「MultiPolygon プロパティー関数」

ST_Contains()
セクション1.4「MySQL 5.6 の新機能」
セクション12.15.9.1「オブジェクト形状を使用する空間関係関数」
セクション12.15.9「幾何オブジェクト間の空間関係をテストする関数」

ST_Crosses()
セクション12.15.9.1「オブジェクト形状を使用する空間関係関数」

ST_Difference()
セクション12.15.8「空間演算子関数」

ST_Disjoint()
セクション12.15.9.1「オブジェクト形状を使用する空間関係関数」

ST_Distance()
セクション12.15.9.1「オブジェクト形状を使用する空間関係関数」
セクション12.15.9「幾何オブジェクト間の空間関係をテストする関数」

ST_Envelope()
セクション12.15.7.1「一般的な幾何プロパティー関数」

ST_Equals()
セクション12.15.9.1「オブジェクト形状を使用する空間関係関数」

ST_Intersection()
セクション12.15.8「空間演算子関数」

ST_Intersects()
セクション12.15.9.1「オブジェクト形状を使用する空間関係関数」

ST_Overlaps()
セクション12.15.9.1「オブジェクト形状を使用する空間関係関数」

ST_SymDifference()
セクション12.15.8「空間演算子関数」

ST_Touches()
セクション12.15.9.1「オブジェクト形状を使用する空間関係関数」

ST_Union()
セクション12.15.8「空間演算子関数」

ST_Within()
セクション12.15.9.1「オブジェクト形状を使用する空間関係関数」

StartPoint()
セクション12.15.7.3「LineString プロパティー関数」
セクション12.15.7.4「MultiLineString プロパティー関数」
セクション12.15.8「空間演算子関数」

STD()
セクション12.19.1「GROUP BY (集約) 関数」
セクション1.3.2「MySQL の主な機能」
セクション1.7.1「標準 SQL に対する MySQL 拡張機能」

STDDEV()
セクション12.19.1「GROUP BY (集約) 関数」

STDDEV_POP()
セクション12.19.1「GROUP BY (集約) 関数」

STDDEV_SAMP()
セクション12.19.1「GROUP BY (集約) 関数」

STR_TO_DATE()
セクション10.7「MySQL Server のロケールサポート」
セクション12.7「日付および時間関数」

STRCMP()
セクションB.5.5.2「DATE カラムの使用に関する問題」
セクション12.5.1「文字列比較関数」

SUBDATE()
セクション12.7「日付および時間関数」

SUBSTR()
セクション12.5「文字列関数」

SUBSTRING()
セクション12.5「文字列関数」
セクション10.1.9.1「結果文字列」

SUBSTRING_INDEX()
セクション6.3.13「SQL ベースの MySQL アカウントアクティビティーの監査」
セクション12.5「文字列関数」

SUBTIME()
セクション12.7「日付および時間関数」

SUM()
セクション11.4.4「ENUM 型」
セクション12.19.1「GROUP BY (集約) 関数」
セクション19.1「MySQL のパーティショニングの概要」
セクション1.3.2「MySQL の主な機能」
セクションB.5.5.3「NULL 値に関する問題」
セクション11.4.5「SET 型」
セクション20.5.2「ビュー処理アルゴリズム」
ルースインデックススキャン
セクション24.3.2「新しいユーザー定義関数の追加」
セクション11.1.2「日付と時間型の概要」
セクション20.5.3「更新可能および挿入可能なビュー」

SYSDATE()
セクション8.9.3.1「クエリーキャッシュの動作」
セクション5.1.3「サーバーコマンドオプション」
セクション5.1.4「サーバーシステム変数」
セクション17.1.2.1「ステートメントベースおよび行ベースレプリケーションのメリットとデメリット」
セクション17.1.2.3「バイナリロギングでの安全および安全でないステートメントの判断」
セクション17.4.1.15「レプリケーションとシステム関数」
セクション17.4.1.13「レプリケーションと小数秒サポート」
セクション12.7「日付および時間関数」
セクション11.3.6「時間値での小数秒」

SYSTEM_USER()
セクション17.1.2.3「バイナリロギングでの安全および安全でないステートメントの判断」
セクション10.1.12「メタデータ用の UTF-8」
セクション12.14「情報関数」

T

[索引の先頭]

TAN()
セクション12.6.2「数学関数」

thr_setconcurrency()
セクション5.1.4「サーバーシステム変数」

TIME()
セクション12.7「日付および時間関数」

TIME_FORMAT()
セクション12.7「日付および時間関数」

TIME_TO_SEC()
セクション12.7「日付および時間関数」
セクション19.6.3「関数に関連するパーティショニング制限」

TIMEDIFF()
セクション12.7「日付および時間関数」

TIMESTAMP()
セクション12.7「日付および時間関数」

TIMESTAMPADD()
セクション12.7「日付および時間関数」

TIMESTAMPDIFF()
セクション12.7「日付および時間関数」
セクション3.3.4.5「日付の計算」

TO_BASE64()
セクション12.5「文字列関数」

TO_DAYS()
セクション19.2.4「HASH パーティショニング」
セクション19.2「パーティショニングタイプ」
セクション19.4「パーティションプルーニング」
セクション12.7「日付および時間関数」
セクション1.7.1「標準 SQL に対する MySQL 拡張機能」
セクション19.6.3「関数に関連するパーティショニング制限」

TO_SECONDS()
セクション19.2「パーティショニングタイプ」
セクション19.4「パーティションプルーニング」
セクション12.7「日付および時間関数」
セクション19.6.3「関数に関連するパーティショニング制限」

Touches()
セクション12.15.9.2「最小外接矩形 (MBR) を使用する空間関係関数」

TRIM()
セクション10.1.13「カラム文字セットの変換」
セクション12.5「文字列関数」
セクション1.7.1「標準 SQL に対する MySQL 拡張機能」
セクション10.1.9.1「結果文字列」

TRUNCATE()
セクション12.6.2「数学関数」

U

[索引の先頭]

UCASE()
セクション12.5「文字列関数」
セクション10.1.9.1「結果文字列」

UNCOMPRESS()
セクション2.9.4「MySQL ソース構成オプション」
セクション5.1.4「サーバーシステム変数」
セクション12.13「暗号化関数と圧縮関数」

UNCOMPRESSED_LENGTH()
セクション12.13「暗号化関数と圧縮関数」

UNHEX()
セクション12.5「文字列関数」
セクション12.13「暗号化関数と圧縮関数」

UNIX_TIMESTAMP()
セクション19.2.1「RANGE パーティショニング」

セクション8.9.3.1「クエリーキャッシュの動作」
セクション5.1.4「サーバーシステム変数」
セクションB.5.4.6「タイムゾーンの問題」
セクション17.1.2.3「バイナリロギングでの安全および安全でないステートメントの判断」
セクション12.7「日付および時間関数」
セクション19.6.3「関数に関連するパーティショニング制限」

UpdateXML()

セクション12.11「XML 関数」

UPPER()

セクション10.1.14.1「Unicode 文字セット」
セクション12.10「キャスト関数と演算子」
セクション10.1.8「文字列のレパートリー」
セクション12.5「文字列関数」
セクション10.1.7.9「照合順序と INFORMATION_SCHEMA 検索」
セクション10.1.9.1「結果文字列」

USER()

セクション6.3.13「SQL ベースの MySQL アカウントアクティビティーの監査」
セクション8.9.3.1「クエリーキャッシュの動作」
サーバー側認証プラグインの作成
セクション17.1.2.1「ステートメントベースおよび行ベースレプリケーションのメリットとデメリット」
セクション17.1.2.3「バイナリロギングでの安全および安全でないステートメントの判断」
セクション6.3.9「プロキシユーザー」
セクション10.1.12「メタデータ用の UTF-8」
セクション17.4.1.15「レプリケーションとシステム関数」
セクション10.1.7.5「式の照合順序」
セクション12.14「情報関数」
セクション5.2.4.3「混合形式のバイナリロギング形式」
認証プラグインでのプロキシユーザーサポートの実装

UTC_DATE

セクション12.7「日付および時間関数」

UTC_DATE()

セクション17.1.2.3「バイナリロギングでの安全および安全でないステートメントの判断」
セクション12.7「日付および時間関数」

UTC_TIME

セクション12.7「日付および時間関数」

UTC_TIME()

セクション17.1.2.3「バイナリロギングでの安全および安全でないステートメントの判断」
セクション12.7「日付および時間関数」

UTC_TIMESTAMP

セクション12.7「日付および時間関数」

UTC_TIMESTAMP()

セクション10.6「MySQL Server でのタイムゾーンのサポート」
セクション17.1.2.3「バイナリロギングでの安全および安全でないステートメントの判断」
セクション17.4.1.13「レプリケーションと小数秒サポート」
セクション12.7「日付および時間関数」
セクション11.3.6「時間値での小数秒」

UUID()

セクション12.18「その他の関数」
セクション8.9.3.1「クエリーキャッシュの動作」
セクション17.1.2.1「ステートメントベースおよび行ベースレプリケーションのメリットとデメリット」
セクション20.7「ストアドプログラムのバイナリロギング」
セクション17.1.2.3「バイナリロギングでの安全および安全でないステートメントの判断」
セクション17.1.4.4「バイナリログのオプションと変数」
セクション5.2.4.2「バイナリログ形式の設定」
セクション17.4.1.15「レプリケーションとシステム関数」
セクション5.2.4.3「混合形式のバイナリロギング形式」

UUID_SHORT()

セクション12.18「その他の関数」
セクション8.9.3.1「クエリーキャッシュの動作」
セクション17.1.2.1「ステートメントベースおよび行ベースレプリケーションのメリットとデメリット」
セクション17.1.2.3「バイナリロギングでの安全および安全でないステートメントの判断」

V

[索引の先頭]

VALIDATE_PASSWORD_STRENGTH()

セクション1.4「MySQL 5.6 の新機能」
セクション24.2.3.8「パスワード検証プラグイン」
セクション6.1.2.6「パスワード検証プラグイン」
パスワード検証プラグインのオプションおよび変数
セクション12.13「暗号化関数と圧縮関数」

VALUES()

セクション13.2.5.3「INSERT ... ON DUPLICATE KEY UPDATE 構文」
セクション12.18「その他の関数」

VAR_POP()

セクション12.19.1「GROUP BY (集約) 関数」

VAR_SAMP()

セクション12.19.1「GROUP BY (集約) 関数」

VARIANCE()

セクション12.19.1「GROUP BY (集約) 関数」

VERSION()

セクション17.1.2.1「ステートメントベースおよび行ベースレプリケーションのメリットとデメリット」
セクション10.1.12「メタデータ用の UTF-8」
セクション17.4.1.15「レプリケーションとシステム関数」
セクション10.1.7.5「式の照合順序」
セクション12.14「情報関数」
セクションB.5.5.1「文字列検索での大文字/小文字の区別」
セクション6.3.12.3「監査ログファイル」

W

[索引の先頭]

WAIT_UNTIL_SQL_THREAD_AFTER_GTIDS()
セクション12.16「グローバルトランザクション ID とともに使用される関数」

WEEK()
セクション5.1.4「サーバーシステム変数」
セクション12.7「日付および時間関数」

WEEKDAY()
セクション19.2「パーティショニングタイプ」
セクション12.7「日付および時間関数」
セクション1.7.1「標準 SQL に対する MySQL 拡張機能」
セクション19.6.3「関数に関連するパーティショニング制限」

WEEKOFYEAR()
セクション12.7「日付および時間関数」

WEIGHT_STRING()
セクション10.1.14.1「Unicode 文字セット」
セクション10.4「文字セットへの照合順序の追加」
セクションB.5.5.1「文字列検索での大文字/小文字の区別」
セクション12.5「文字列関数」

Within()
セクション12.15.9.2「最小外接矩形 (MBR) を使用する空間関係関数」

X

[索引の先頭]

X()
セクション12.15.7.2「点プロパティー関数」

Y

[索引の先頭]

Y()
セクション12.15.7.2「点プロパティー関数」

YEAR()
セクション19.2.4「HASH パーティショニング」
セクション19.2.7「MySQL パーティショニングによる NULL の扱い」
セクション19.3.1「RANGE および LIST パーティションの管理」
セクション19.2.1「RANGE パーティショニング」
セクション19.2「パーティショニングタイプ」
セクション19.4「パーティションプルーニング」
セクション12.7「日付および時間関数」
セクション3.3.4.5「日付の計算」
セクション19.6.3「関数に関連するパーティショニング制限」

YEARWEEK()
セクション12.7「日付および時間関数」
セクション19.6.3「関数に関連するパーティショニング制限」

INFORMATION_SCHEMA の索引

C | E | F | G | I | K | N | O | P | R | S | T | U | V

C

[索引の先頭]

CHARACTER_SETS

セクション21.1「INFORMATION_SCHEMA CHARACTER_SETS テーブル」
セクション10.1.9.3「SHOW ステートメントと INFORMATION_SCHEMA」

COLLATION_CHARACTER_SET_APPLICABILITY

セクション21.3「INFORMATION_SCHEMA COLLATION_CHARACTER_SET_APPLICABILITY テーブル」

COLLATIONS

セクション23.7.5「C API データ構造」
セクション21.2「INFORMATION_SCHEMA COLLATIONS テーブル」
セクション8.2.4「INFORMATION_SCHEMA クエリーの最適化」
セクション10.1.9.3「SHOW ステートメントと INFORMATION_SCHEMA」

COLUMN_PRIVILEGES

セクション21.5「INFORMATION_SCHEMA COLUMN_PRIVILEGES テーブル」

COLUMNS

セクション21.4「INFORMATION_SCHEMA COLUMNS テーブル」
セクション21.12「INFORMATION_SCHEMA PARAMETERS テーブル」
セクション21.18「INFORMATION_SCHEMA ROUTINES テーブル」
セクション8.2.4「INFORMATION_SCHEMA クエリーの最適化」

E

[索引の先頭]

ENGINES

セクション21.6「INFORMATION_SCHEMA ENGINES テーブル」
セクション13.7.5.17「SHOW ENGINES 構文」
セクション18.5.9「クイックリファレンス: MySQL Cluster の SQL ステートメント」
セクション5.1.4「サーバーシステム変数」

EVENTS

セクション21.7「INFORMATION_SCHEMA EVENTS テーブル」
セクション20.4.2「イベントスケジューラの構成」
セクション20.4.4「イベントメタデータ」
セクション17.4.1.11「呼び出される機能のレプリケーション」

F

[索引の先頭]

FILES

セクション21.30.1「INFORMATION_SCHEMA FILES テーブル」
セクション21.30「MySQL Cluster の INFORMATION_SCHEMA テーブル」
セクション18.5.12.1「MySQL Cluster ディスクデータオブジェクト」
セクション18.5.10「ndbinfo MySQL Cluster 情報データベース」

G

[索引の先頭]

GLOBAL_STATUS

セクション21.8「INFORMATION_SCHEMA GLOBAL_STATUS および SESSION_STATUS テーブル」
セクション18.5「MySQL Cluster の管理」
セクション18.6「MySQL Cluster レプリケーション」
セクション18.5.15「NDB API 統計のカウンタと変数」

GLOBAL_VARIABLES

セクション21.9「INFORMATION_SCHEMA GLOBAL_VARIABLES および SESSION_VARIABLES テーブル」

I

[索引の先頭]

INFORMATION_SCHEMA GLOBAL_STATUS

セクション21.8「INFORMATION_SCHEMA GLOBAL_STATUS および SESSION_STATUS テーブル」

INFORMATION_SCHEMA GLOBAL_VARIABLES

セクション21.9「INFORMATION_SCHEMA GLOBAL_VARIABLES および SESSION_VARIABLES テーブル」

INFORMATION_SCHEMA.CHARACTER_SE

セクションA.11「MySQL 5.6 FAQ: MySQL の中国語、日本語、および韓国語の文字セット」

INFORMATION_SCHEMA.COLLATIONS

セクション10.4.2「照合順序 ID の選択」

INFORMATION_SCHEMA.COLUMNS

セクションA.11「MySQL 5.6 FAQ: MySQL の中国語、日本語、および韓国語の文字セット」
セクション18.4.14「ndb_index_stat — NDB インデックス統計ユーティリティー」

セクション22.1「パフォーマンススキーマクイックスタート」

INFORMATION_SCHEMA.ENGINES
セクション18.5.9「クイックリファレンス: MySQL Cluster の SQL ステートメント」
セクション22.1「パフォーマンススキーマクイックスタート」

INFORMATION_SCHEMA.EVENTS
セクション13.7.5.19「SHOW EVENTS 構文」
セクション20.4.6「イベントスケジューラと MySQL 権限」
セクション20.4.4「イベントメタデータ」
セクションD.1「ストアドプログラムの制約」
セクション17.4.1.11「呼び出される機能のレプリケーション」

INFORMATION_SCHEMA.FILES
セクション13.1.3「ALTER LOGFILE GROUP 構文」
セクション13.1.8「ALTER TABLESPACE 構文」
セクション13.1.14「CREATE LOGFILE GROUP 構文」
セクション13.1.18「CREATE TABLESPACE 構文」
セクション18.5.12.3「MySQL Cluster ディスクデータストレージの要件」
セクション18.4.10「ndb_desc — NDB テーブルの表示」

INFORMATION_SCHEMA.GLOBAL_STATUS
セクション18.5.15「NDB API 統計のカウンタと変数」

INFORMATION_SCHEMA.INNODB_CMP
セクション14.7.3「InnoDB テーブルの圧縮の調整」
MySQL 用語集
セクション14.14.1.3「圧縮情報スキーマテーブルの使用」

INFORMATION_SCHEMA.INNODB_CMP_PER_INDEX
セクション14.12「InnoDB の起動オプションおよびシステム変数」
セクション14.7.3「InnoDB テーブルの圧縮の調整」

INFORMATION_SCHEMA.INNODB_CMPMEM
セクション14.14.1.3「圧縮情報スキーマテーブルの使用」

INFORMATION_SCHEMA.INNODB_FT_CONFIG
セクション14.2.13.3「FULLTEXT インデックス」

INFORMATION_SCHEMA.INNODB_FT_DEFAULT_STOPWORD
セクション12.9.4「全文ストップワード」

INFORMATION_SCHEMA.INNODB_FT_INDEX_CACHE
セクション14.2.13.3「FULLTEXT インデックス」

INFORMATION_SCHEMA.INNODB_FT_INDEX_TABLE
セクション12.9.4「全文ストップワード」

INFORMATION_SCHEMA.INNODB_LOCK_WAITS
セクション14.14.2.1「InnoDB トランザクションおよびロックテーブルの使用例」

INFORMATION_SCHEMA.INNODB_LOCKS
セクション14.14.2.1「InnoDB トランザクションおよびロックテーブルの使用例」

INFORMATION_SCHEMA.INNODB_METRICS
セクション14.12「InnoDB の起動オプションおよびシステム変数」

INFORMATION_SCHEMA.INNODB_SYS_INDE
セクション14.2.13.3「FULLTEXT インデックス」

INFORMATION_SCHEMA.INNODB_SYS_TAB
セクション14.14.3「InnoDB INFORMATION_SCHEMA システムテーブル」
セクション14.2.13.7「物理的な行構造」

INFORMATION_SCHEMA.INNODB_SYS_TAB
セクション14.14.3「InnoDB INFORMATION_SCHEMA システムテーブル」

INFORMATION_SCHEMA.INNODB_SYS_TAB
セクション14.14.3「InnoDB INFORMATION_SCHEMA システムテーブル」

INFORMATION_SCHEMA.INNODB_TRX
セクション14.12「InnoDB の起動オプションおよびシステム変数」
セクション14.14.2.1「InnoDB トランザクションおよびロックテーブルの使用例」

INFORMATION_SCHEMA.KEY_COLUMN_USA
セクション1.7.3.2「FOREIGN KEY の制約」
セクション14.6.6「InnoDB と FOREIGN KEY 制約」
セクション2.11.1.3「MySQL 5.5 から 5.6 へのアップグレード」
セクション13.1.17.2「外部キー制約の使用」

INFORMATION_SCHEMA.OPTIMIZER_TRACE
セクション1.4「MySQL 5.6 の新機能」

INFORMATION_SCHEMA.PARTITIONS
セクション21.13「INFORMATION_SCHEMA PARTITIONS テーブル」
セクション19.2.5「KEY パーティショニング」
セクション19.2.7「MySQL パーティショニングによる NULL の扱い」
セクション19.2.3.1「RANGE COLUMNS パーティショニング」
セクション5.1.3「サーバーコマンドオプション」
セクション19.3.3「パーティションとサブパーティションをテーブルと交換する」
セクション19.3.5「パーティションに関する情報を取得する」

INFORMATION_SCHEMA.PLUGINS
セクション24.2.4.6「INFORMATION_SCHEMA プラグインの作成」
セクション13.7.3.3「INSTALL PLUGIN 構文」
PAM 認証プラグインのインストール
Windows 認証プラグインのインストール
サーバープラグインライブラリおよびプラグインディスクリプタ
セクション5.1.8.2「サーバープラグイン情報の取得」
サーバー側認証プラグインの作成

セクション8.11.6.1「スレッドプールコンポーネントとインストール」
セクション24.2.4.5「デーモンプラグインの作成」
パスワード検証プラグインのインストール
セクション24.2.4.10「パスワード検証プラグインの作成」
第19章「パーティション化」
セクション24.2.2「プラグイン API のコンポーネント」
セクション24.2.1「プラグイン API の特徴」
セクション5.1.8.1「プラグインのインストールおよびアンインストール」
セクション24.2.4.4「全文パーサープラグインの作成」
セクション17.3.8.2「準同期レプリケーションのインストールと構成」
セクション24.2.4.8「監査プラグインの作成」
セクション6.3.12.1「監査ログプラグインのインストール」

INFORMATION_SCHEMA.PROCESSLIST

セクション21.15「INFORMATION_SCHEMA PROCESSLIST テーブル」
セクション14.14.2.1「InnoDB トランザクションおよびロックテーブルの使用例」
セクション13.7.6.4「KILL 構文」
セクション13.7.5.30「SHOW PROCESSLIST 構文」
セクション22.9.10.3「スレッドテーブル」
セクション8.12.5「スレッド情報の検査」
セクション22.4「パフォーマンススキーマインストゥルメント命名規則」
セクション22.9.5「パフォーマンススキーマステージイベントテーブル」
セクション12.14「情報関数」

INFORMATION_SCHEMA.REFERENTIAL_CONSTRAINTS

セクション1.7.2.4「外部キーの違い」

INFORMATION_SCHEMA.ROUTINES

セクション21.18「INFORMATION_SCHEMA ROUTINES テーブル」
第21章「INFORMATION_SCHEMA テーブル」
セクションA.4「MySQL 5.6 FAQ: ストアドプロシージャーおよびストアドファンクション」

INFORMATION_SCHEMA.SESSION_STATUS

セクション18.5.15「NDB API 統計のカウンタと変数」

INFORMATION_SCHEMA.STATISTICS

セクション18.4.14「ndb_index_stat — NDB インデックス統計ユーティリティー」
セクション14.13.16.2「非永続的オプティマイザ統計のパラメータの構成」

INFORMATION_SCHEMA.TABLE_CONSTRAINTS

セクション21.17「INFORMATION_SCHEMA REFERENTIAL_CONSTRAINTS テーブル」
セクション2.11.1.3「MySQL 5.5 から 5.6 へのアップグレード」
セクション1.7.2.4「外部キーの違い」

information_schema.table_constraints

セクション14.11.1「オンライン DDL の概要」

INFORMATION_SCHEMA.TABLES

第21章「INFORMATION_SCHEMA テーブル」
セクション18.5.13.3「MySQL Cluster データノードのオンライン追加: 詳細な例」
セクション5.1.3「サーバーコマンドオプション」
セクション14.13.16.2「非永続的オプティマイザ統計のパラメータの構成」

INFORMATION_SCHEMA.TRIGGERS

セクションA.5「MySQL 5.6 FAQ: トリガー」

INFORMATION_SCHEMA.VIEWS

セクション20.5.3「更新可能および挿入可能なビュー」

INNODB_BUFFER_PAGE

セクション21.29.17「INFORMATION_SCHEMA INNODB_BUFFER_PAGE_LRU テーブル」
セクション14.14.5「InnoDB INFORMATION_SCHEMA バッファープールテーブル」

INNODB_BUFFER_PAGE_LRU

セクション21.29.17「INFORMATION_SCHEMA INNODB_BUFFER_PAGE_LRU テーブル」
セクション14.14.5「InnoDB INFORMATION_SCHEMA バッファープールテーブル」
セクション14.13.1.5「再起動を高速化するための InnoDB バッファープールのプリロード」

INNODB_BUFFER_POOL_STATS

セクション14.14.5「InnoDB INFORMATION_SCHEMA バッファープールテーブル」
セクション8.9.1「InnoDB バッファープール」

INNODB_CMP

セクション14.14.1.1「INNODB_CMP および INNODB_CMP_RESET」
セクション14.14.1.2「INNODB_CMPMEM および INNODB_CMPMEM_RESET」
セクション14.14.1「圧縮に関する InnoDB INFORMATION_SCHEMA テーブル」
セクション14.14.1.3「圧縮情報スキーマテーブルの使用」
セクション14.7.4「実行時の圧縮のモニタリング」

INNODB_CMP_PER_INDEX

セクション14.14.1.3「圧縮情報スキーマテーブルの使用」
セクション14.7.4「実行時の圧縮のモニタリング」

INNODB_CMP_RESET

セクション14.14.1.1「INNODB_CMP および INNODB_CMP_RESET」
セクション14.14.1.2「INNODB_CMPMEM および INNODB_CMPMEM_RESET」
セクション14.14.1「圧縮に関する InnoDB INFORMATION_SCHEMA テーブル」

INNODB_CMPMEM

セクション14.14.1.2「INNODB_CMPMEM および INNODB_CMPMEM_RESET」
セクション14.14.1「圧縮に関する InnoDB INFORMATION_SCHEMA テーブル」
セクション14.14.1.3「圧縮情報スキーマテーブルの使用」

INNODB_CMPMEM_RESET

セクション14.14.1.2「INNODB_CMPMEM および INNODB_CMPMEM_RESET」

INNODB_FT_BEING_DELETED

セクション14.2.13.3「FULLTEXT インデックス」
セクション14.14.4「InnoDB INFORMATION_SCHEMA FULLTEXT インデックステーブル」
セクション14.12「InnoDB の起動オプションおよびシステム変数」

INNODB_FT_CONFIG

セクション14.2.13.3「FULLTEXT インデックス」
セクション14.14.4「InnoDB INFORMATION_SCHEMA FULLTEXT インデックステーブル」
セクション14.12「InnoDB の起動オプションおよびシステム変数」

INNODB_FT_DEFAULT_STOPWORD

セクション14.2.13.3「FULLTEXT インデックス」
セクション14.14.4「InnoDB INFORMATION_SCHEMA FULLTEXT インデックステーブル」
セクション12.9.4「全文ストップワード」

INNODB_FT_DELETED

セクション14.2.13.3「FULLTEXT インデックス」
セクション21.29.25「INFORMATION_SCHEMA INNODB_FT_BEING_DELETED テーブル」
セクション14.14.4「InnoDB INFORMATION_SCHEMA FULLTEXT インデックステーブル」
セクション14.12「InnoDB の起動オプションおよびシステム変数」

INNODB_FT_INDEX_CACHE

セクション14.2.13.3「FULLTEXT インデックス」
セクション14.14.4「InnoDB INFORMATION_SCHEMA FULLTEXT インデックステーブル」
セクション14.12「InnoDB の起動オプションおよびシステム変数」

INNODB_FT_INDEX_TABLE

セクション14.2.13.3「FULLTEXT インデックス」
セクション14.14.4「InnoDB INFORMATION_SCHEMA FULLTEXT インデックステーブル」
セクション14.12「InnoDB の起動オプションおよびシステム変数」

INNODB_LOCK_WAITS

セクション14.14.2「InnoDB INFORMATION_SCHEMA トランザクションおよびロックテーブル」
セクション14.14.2.3「InnoDB トランザクションおよびロックテーブルのデータ永続性および一貫性」
セクション14.14.2.2「INNODB_LOCKS と INNODB_LOCK_WAITS のデータ」
PROCESSLIST データとの不整合の可能性

INNODB_LOCKS

セクション14.14.2「InnoDB INFORMATION_SCHEMA トランザクションおよびロックテーブル」
セクション14.14.2.3「InnoDB トランザクションおよびロックテーブルのデータ永続性および一貫性」
セクション14.14.2.2「INNODB_LOCKS と INNODB_LOCK_WAITS のデータ」
MySQL 用語集
PROCESSLIST データとの不整合の可能性

INNODB_METRICS

セクション14.14.6「InnoDB INFORMATION_SCHEMA メトリックテーブル」

innodb_metrics

MySQL 用語集

INNODB_SYS_COLUMNS

セクション14.14.3「InnoDB INFORMATION_SCHEMA システムテーブル」
セクション14.19.3「InnoDB データディクショナリの操作のトラブルシューティング」

INNODB_SYS_DATAFILES

セクション14.14.3「InnoDB INFORMATION_SCHEMA システムテーブル」

INNODB_SYS_FIELDS

セクション14.14.3「InnoDB INFORMATION_SCHEMA システムテーブル」

INNODB_SYS_FOREIGN

セクション1.7.3.2「FOREIGN KEY の制約」
セクション14.14.3「InnoDB INFORMATION_SCHEMA システムテーブル」
セクション14.6.6「InnoDB と FOREIGN KEY 制約」
セクション13.1.17.2「外部キー制約の使用」

INNODB_SYS_FOREIGN_COLS

セクション1.7.3.2「FOREIGN KEY の制約」
セクション14.14.3「InnoDB INFORMATION_SCHEMA システムテーブル」
セクション14.6.6「InnoDB と FOREIGN KEY 制約」
セクション13.1.17.2「外部キー制約の使用」

INNODB_SYS_INDEXES

セクション14.2.13.3「FULLTEXT インデックス」
セクション14.14.3「InnoDB INFORMATION_SCHEMA システムテーブル」
セクション14.19.3「InnoDB データディクショナリの操作のトラブルシューティング」

INNODB_SYS_TABLES

セクション14.2.13.3「FULLTEXT インデックス」
セクション14.14.3「InnoDB INFORMATION_SCHEMA システムテーブル」
セクション14.19.3「InnoDB データディクショナリの操作のトラブルシューティング」

INNODB_SYS_TABLESPACES

セクション21.29.15「INFORMATION_SCHEMA INNODB_SYS_TABLESPACES テーブル」
セクション14.14.3「InnoDB INFORMATION_SCHEMA システムテーブル」

INNODB_SYS_TABLESTATS
セクション14.14.3「InnoDB INFORMATION_SCHEMA システムテーブル」

INNODB_TRX
セクション14.14.2「InnoDB INFORMATION_SCHEMA トランザクションおよびロックテーブル」
セクション14.14.2.3「InnoDB トランザクションおよびロックテーブルのデータ永続性および一貫性」
MySQL 用語集
PROCESSLIST データとの不整合の可能性

K

[索引の先頭]

KEY_COLUMN_USAGE
セクション21.10「INFORMATION_SCHEMA KEY_COLUMN_USAGE テーブル」
セクション8.2.4「INFORMATION_SCHEMA クエリーの最適化」

N

[索引の先頭]

NDB_TRANSID_MYSQL_CONNECTION_MAP
セクション18.5.10.20「ndbinfo server_operations テーブル」
セクション18.5.10.21「ndbinfo server_transactions テーブル」

ndb_transid_mysql_connection_map
セクション21.30「MySQL Cluster の INFORMATION_SCHEMA テーブル」
セクション18.3.4.2「MySQL Cluster 用の MySQL Server オプション」

O

[索引の先頭]

OPTIMIZER_TRACE
セクション21.11「INFORMATION_SCHEMA OPTIMIZER_TRACE テーブル」

P

[索引の先頭]

PARAMETERS
セクション21.12「INFORMATION_SCHEMA PARAMETERS テーブル」
セクション21.18「INFORMATION_SCHEMA ROUTINES テーブル」

PARTITIONS
セクション21.13「INFORMATION_SCHEMA PARTITIONS テーブル」
セクション8.2.4「INFORMATION_SCHEMA クエリーの最適化」
セクション19.2.7「MySQL パーティショニングによる NULL の扱い」
セクション19.3.3「パーティションとサブパーティションをテーブルと交換する」
セクション19.3.5「パーティションに関する情報を取得する」
第19章「パーティション化」

PLUGINS
セクション21.14「INFORMATION_SCHEMA PLUGINS テーブル」
セクション1.4「MySQL 5.6 の新機能」
セクション5.1.8.2「サーバープラグイン情報の取得」

PROCESSLIST
セクション21.15「INFORMATION_SCHEMA PROCESSLIST テーブル」
PROCESSLIST データとの不整合の可能性
セクション13.7.5.30「SHOW PROCESSLIST 構文」
セクション8.12.5「スレッド情報の検査」

PROFILING
セクション21.16「INFORMATION_SCHEMA PROFILING テーブル」
セクション13.7.5.31「SHOW PROFILE 構文」

R

[索引の先頭]

REFERENTIAL_CONSTRAINTS
セクション21.17「INFORMATION_SCHEMA REFERENTIAL_CONSTRAINTS テーブル」
セクション8.2.4「INFORMATION_SCHEMA クエリーの最適化」

ROUTINES
セクション21.12「INFORMATION_SCHEMA PARAMETERS テーブル」
セクション21.18「INFORMATION_SCHEMA ROUTINES テーブル」
セクションA.4「MySQL 5.6 FAQ: ストアドプロシージャーおよびストアドファンクション」
セクション13.7.5.29「SHOW PROCEDURE STATUS 構文」
セクション20.2.3「ストアドルーチンのメタデータ」

S

[索引の先頭]

SCHEMA_PRIVILEGES
セクション21.20「INFORMATION_SCHEMA SCHEMA_PRIVILEGES テーブル」

SCHEMATA
セクション21.19「INFORMATION_SCHEMA SCHEMATA テーブル」
セクション6.2.2「権限システム付与テーブル」

SESSION_STATUS
セクション21.8「INFORMATION_SCHEMA GLOBAL_STATUS および SESSION_STATUS テーブル」
セクション18.5「MySQL Cluster の管理」
セクション18.6「MySQL Cluster レプリケーション」
セクション18.5.15「NDB API 統計のカウンタと変数」

SESSION_VARIABLES
セクション21.9「INFORMATION_SCHEMA GLOBAL_VARIABLES および SESSION_VARIABLES テーブル」

STATISTICS
セクション21.21「INFORMATION_SCHEMA STATISTICS テーブル」
セクション8.2.4「INFORMATION_SCHEMA クエリーの最適化」
セクション14.12「InnoDB の起動オプションおよびシステム変数」

T

[索引の先頭]

TABLE_CONSTRAINTS
セクション21.24「INFORMATION_SCHEMA TABLE_CONSTRAINTS テーブル」
セクション8.2.4「INFORMATION_SCHEMA クエリーの最適化」

TABLE_PRIVILEGES
セクション21.25「INFORMATION_SCHEMA TABLE_PRIVILEGES テーブル」

TABLES
セクション21.22「INFORMATION_SCHEMA TABLES テーブル」
セクション8.2.4「INFORMATION_SCHEMA クエリーの最適化」
第21章「INFORMATION_SCHEMA テーブル」
セクション14.12「InnoDB の起動オプションおよびシステム変数」

TABLESPACES
セクション21.23「INFORMATION_SCHEMA TABLESPACES テーブル」

TP_THREAD_GROUP_STATE
セクション21.31.2「INFORMATION_SCHEMA TP_THREAD_GROUP_STATE テーブル」
セクション21.31「スレッドプールの INFORMATION_SCHEMA テーブル」
セクション8.11.6.1「スレッドプールコンポーネントとインストール」

TP_THREAD_GROUP_STATS
セクション21.31.1「INFORMATION_SCHEMA TP_THREAD_STATE テーブル」
セクション21.31「スレッドプールの INFORMATION_SCHEMA テーブル」
セクション8.11.6.1「スレッドプールコンポーネントとインストール」

TP_THREAD_STATE
セクション21.31「スレッドプールの INFORMATION_SCHEMA テーブル」
セクション8.11.6.1「スレッドプールコンポーネントとインストール」

TRIGGERS
セクション21.26「INFORMATION_SCHEMA TRIGGERS テーブル」
セクション8.2.4「INFORMATION_SCHEMA クエリーの最適化」
セクション13.7.5.13「SHOW CREATE TRIGGER 構文」
セクション13.7.5.39「SHOW TRIGGERS 構文」
セクション20.3.2「トリガーのメタデータ」

U

[索引の先頭]

USER_PRIVILEGES
セクション21.27「INFORMATION_SCHEMA USER_PRIVILEGES テーブル」

V

[索引の先頭]

VIEWS
セクション21.28「INFORMATION_SCHEMA VIEWS テーブル」
セクション8.2.4「INFORMATION_SCHEMA クエリーの最適化」
セクション13.7.5.14「SHOW CREATE VIEW 構文」
セクション20.5.4「ビューのメタデータ」

結合型の索引

A | C | E | F | I | R | S | U

A

[索引の先頭]

ALL

セクション8.8.2「EXPLAIN 出力フォーマット」
セクション8.2.1.10「Nested Loop 結合アルゴリズム」
セクション8.2.1.20「フルテーブルスキャンを回避する方法」
外部結合と準結合の Block Nested Loop アルゴリズム

C

[索引の先頭]

const

セクション8.8.3「EXPLAIN EXTENDED 出力フォーマット」
セクション8.8.2「EXPLAIN 出力フォーマット」
セクション18.3.4.3「MySQL Cluster のシステム変数」
セクション8.2.1.15「ORDER BY の最適化」
セクション13.2.9「SELECT 構文」
シングルパートインデックスの range アクセスメソッド

E

[索引の先頭]

eq_ref

Batched Key Access 結合
EXISTS 戦略によるサブクエリーの最適化
セクション8.8.2「EXPLAIN 出力フォーマット」
セクション15.7.1「MERGE テーブルの長所と短所」
セクション18.3.4.3「MySQL Cluster のシステム変数」
セクション8.2.1.6「インデックスコンディションプッシュダウンの最適化」

F

[索引の先頭]

fulltext

セクション8.8.2「EXPLAIN 出力フォーマット」

I

[索引の先頭]

index

セクション8.8.2「EXPLAIN 出力フォーマット」
セクション8.2.1.10「Nested Loop 結合アルゴリズム」
外部結合と準結合の Block Nested Loop アルゴリズム

index_merge

セクション8.8.2「EXPLAIN 出力フォーマット」
セクション8.2.1.4「インデックスマージの最適化」

index_subquery

EXISTS 戦略によるサブクエリーの最適化
セクション8.8.2「EXPLAIN 出力フォーマット」
セクション13.2.10.10「サブクエリーの最適化」

R

[索引の先頭]

range

セクション8.8.2「EXPLAIN 出力フォーマット」
セクション8.2.1.10「Nested Loop 結合アルゴリズム」
セクション8.2.1.3「range の最適化」
セクション8.2.1.6「インデックスコンディションプッシュダウンの最適化」
セクション8.2.1.4「インデックスマージの最適化」
シングルパートインデックスの range アクセスメソッド
ルースインデックススキャン
外部結合と準結合の Block Nested Loop アルゴリズム

ref

Batched Key Access 結合
EXISTS 戦略によるサブクエリーの最適化
セクション8.8.3「EXPLAIN EXTENDED 出力フォーマット」
セクション8.8.2「EXPLAIN 出力フォーマット」
FROM 句内のサブクエリー (派生テーブル) の最適化
セクション8.3.7「InnoDB および MyISAM インデックス統計コレクション」
セクション15.7.1「MERGE テーブルの長所と短所」
セクション18.3.4.3「MySQL Cluster のシステム変数」
セクション8.2.1.6「インデックスコンディションプッシュダウンの最適化」

ref_or_null

EXISTS 戦略によるサブクエリーの最適化
セクション8.8.2「EXPLAIN 出力フォーマット」
セクション8.2.1.8「IS NULL の最適化」
セクション8.2.1.6「インデックスコンディションプッシュダウンの最適化」

S

[索引の先頭]

system

セクション8.8.3「EXPLAIN EXTENDED 出力フォーマット」
セクション8.8.2「EXPLAIN 出力フォーマット」
セクション13.2.9「SELECT 構文」
シングルパートインデックスの range アクセスメソッド

U

[索引の先頭]

unique_subquery

EXISTS 戦略によるサブクエリーの最適化
セクション8.8.2「EXPLAIN 出力フォーマット」
セクション13.2.10.10「サブクエリーの最適化」

演算子の索引

シンボル | A | B | C | D | E | I | L | N | O | R | X

シンボル

[索引の先頭]

-

セクション12.10「キャスト関数と演算子」
セクション19.6「パーティショニングの制約と制限」
セクション11.1.1「数値型の概要」
セクション12.7「日付および時間関数」
セクション12.6.1「算術演算子」

!

セクション9.5「式の構文」
セクション12.3.1「演算子の優先順位」
セクション12.3.3「論理演算子」

!=

シングルパートインデックスの range アクセスメソッド
マルチパートインデックスの range アクセスメソッド
セクション12.3.2「比較関数と演算子」
セクション12.3.1「演算子の優先順位」

%

セクション12.6.1「算術演算子」

&

セクション13.1.17「CREATE TABLE 構文」
セクション19.6「パーティショニングの制約と制限」
セクション12.12「ビット関数」

&&

セクション1.7.1「標準 SQL に対する MySQL 拡張機能」
セクション12.3.3「論理演算子」

>

セクション8.3.8「B ツリーインデックスとハッシュインデックスの比較」
セクション8.8.2「EXPLAIN 出力フォーマット」
シングルパートインデックスの range アクセスメソッド
マルチパートインデックスの range アクセスメソッド
セクション1.7.1「標準 SQL に対する MySQL 拡張機能」
セクション12.3.2「比較関数と演算子」
セクション12.3.1「演算子の優先順位」

>>

セクション19.6「パーティショニングの制約と制限」
セクション12.12「ビット関数」
セクション1.7.1「標準 SQL に対する MySQL 拡張機能」

>=

セクション8.3.8「B ツリーインデックスとハッシュインデックスの比較」
セクション8.8.2「EXPLAIN 出力フォーマット」
シングルパートインデックスの range アクセスメソッド
マルチパートインデックスの range アクセスメソッド
セクション1.7.1「標準 SQL に対する MySQL 拡張機能」
セクション12.3.2「比較関数と演算子」
セクション12.3.1「演算子の優先順位」

<

セクション8.3.8「B ツリーインデックスとハッシュインデックスの比較」
セクション8.8.2「EXPLAIN 出力フォーマット」
セクション3.3.4.6「NULL 値の操作」
シングルパートインデックスの range アクセスメソッド
マルチパートインデックスの range アクセスメソッド
セクション1.7.1「標準 SQL に対する MySQL 拡張機能」
セクション12.3.2「比較関数と演算子」
セクション12.3.1「演算子の優先順位」

<>

セクション8.8.2「EXPLAIN 出力フォーマット」
セクション3.3.4.6「NULL 値の操作」
シングルパートインデックスの range アクセスメソッド
マルチパートインデックスの range アクセスメソッド
セクション1.7.1「標準 SQL に対する MySQL 拡張機能」
セクション12.3.2「比較関数と演算子」
セクション12.3.1「演算子の優先順位」

<<

セクション19.6「パーティショニングの制約と制限」
セクション12.12「ビット関数」
セクション1.7.1「標準 SQL に対する MySQL 拡張機能」

<=

セクション8.3.8「B ツリーインデックスとハッシュインデックスの比較」
セクション8.8.2「EXPLAIN 出力フォーマット」
シングルパートインデックスの range アクセスメソッド
マルチパートインデックスの range アクセスメソッド
セクション1.7.1「標準 SQL に対する MySQL 拡張機能」
セクション12.3.2「比較関数と演算子」
セクション12.3.1「演算子の優先順位」

<=>

セクション8.8.2「EXPLAIN 出力フォーマット」
シングルパートインデックスの range アクセスメソッド
マルチパートインデックスの range アクセスメソッド
セクション12.2「式評価での型変換」
セクション1.7.1「標準 SQL に対する MySQL 拡張機能」
セクション12.3.2「比較関数と演算子」
セクション12.3.1「演算子の優先順位」

*

セクション19.6「パーティショニングの制約と制限」
セクション11.1.1「数値型の概要」
セクション12.6.1「算術演算子」

+

セクション12.10「キャスト関数と演算子」
セクション19.6「パーティショニングの制約と制限」
セクション11.1.1「数値型の概要」
セクション12.7「日付および時間関数」
セクション12.6.1「算術演算子」

/

セクション5.1.4「サーバーシステム変数」

セクション19.6「パーティショニングの制約と制限」
セクション12.6.1「算術演算子」

:=

セクション13.7.4「SET 構文」
セクション9.4「ユーザー定義変数」
セクション12.3.4「割り当て演算子」
セクション12.3.1「演算子の優先順位」

=

セクション8.3.8「B ツリーインデックスとハッシュインデックスの比較」
セクション8.8.2「EXPLAIN 出力フォーマット」
セクション3.3.4.6「NULL 値の操作」
セクション13.7.4「SET 構文」
セクションD.4「サブクエリーの制約」
シングルパートインデックスの range アクセスメソッド
マルチパートインデックスの range アクセスメソッド
セクション9.4「ユーザー定義変数」
セクション12.3.4「割り当て演算子」
セクション12.5.1「文字列比較関数」
セクション1.7.1「標準 SQL に対する MySQL 拡張機能」
セクション12.3.2「比較関数と演算子」
セクション12.3.1「演算子の優先順位」

^

セクション19.6「パーティショニングの制約と制限」
セクション12.12「ビット関数」
セクション9.5「式の構文」
セクション12.3.1「演算子の優先順位」

|

セクション19.6「パーティショニングの制約と制限」
セクション12.12「ビット関数」

||

セクション10.1.7.3「COLLATE 句の優先順位」
セクション5.1.7「サーバー SQL モード」
セクション9.5「式の構文」
セクション1.7.1「標準 SQL に対する MySQL 拡張機能」
セクション12.3.1「演算子の優先順位」
セクション10.1.9.1「結果文字列」
セクション12.3.3「論理演算子」

~

セクション19.6「パーティショニングの制約と制限」
セクション12.12「ビット関数」

A

[索引の先頭]

AND

セクション3.6.7「2 つのキーを使用した検索」
セクション8.3.8「B ツリーインデックスとハッシュインデックスの比較」
セクション13.1.17「CREATE TABLE 構文」
EXISTS 戦略によるサブクエリーの最適化
セクション8.2.1.4「インデックスマージの最適化」
インデックスマージ共通集合アクセスアルゴリズム
セクションD.4「サブクエリーの制約」
シングルパートインデックスの range アクセスメソッド
セクション20.5.2「ビュー処理アルゴリズム」
マルチパートインデックスの range アクセスメソッド
セクション12.5.1「文字列比較関数」
セクション1.7.1「標準 SQL に対する MySQL 拡張機能」
セクション3.3.4.2「特定の行の選択」
セクション12.3.3「論理演算子」

B

[索引の先頭]

BETWEEN

セクション8.3.8「B ツリーインデックスとハッシュインデックスの比較」
セクション8.8.2「EXPLAIN 出力フォーマット」
シングルパートインデックスの range アクセスメソッド
マルチパートインデックスの range アクセスメソッド
セクション12.2「式評価での型変換」
セクション12.3.2「比較関数と演算子」

BINARY

セクション10.1.7.7「BINARY 演算子」
セクション12.10「キャスト関数と演算子」
セクション3.3.4.7「パターンマッチング」
セクション3.3.4.4「行のソート」

BINARY str

セクション12.10「キャスト関数と演算子」

C

[索引の先頭]

CASE

セクション13.6.5.1「CASE 構文」
セクション12.4「制御フロー関数」
セクション9.5「式の構文」
セクション1.7.1「標準 SQL に対する MySQL 拡張機能」

CASE value WHEN END

セクション12.4「制御フロー関数」

CASE WHEN END

セクション12.4「制御フロー関数」

CASE WHEN expr1 = expr2 THEN NULL ELSE expr1 END

セクション12.4「制御フロー関数」

D

[索引の先頭]

DIV

セクション19.6「パーティショニングの制約と制限」
セクション12.6.1「算術演算子」

E

[索引の先頭]

expr BETWEEN min AND max
セクション12.3.2「比較関数と演算子」

expr LIKE pat
セクション12.5.1「文字列比較関数」

expr NOT BETWEEN min AND max
セクション12.3.2「比較関数と演算子」

expr NOT LIKE pat
セクション12.5.1「文字列比較関数」

expr NOT REGEXP pat
セクション12.5.2「正規表現」

expr NOT RLIKE pat
セクション12.5.2「正規表現」

expr REGEXP pat
セクション12.5.2「正規表現」

expr RLIKE pat
セクション12.5.2「正規表現」

expr1 SOUNDS LIKE expr2
セクション12.5「文字列関数」

I

[索引の先頭]

IS
セクション12.3.1「演算子の優先順位」

IS boolean_value
セクション12.3.2「比較関数と演算子」

IS NOT boolean_value
セクション12.3.2「比較関数と演算子」

IS NOT NULL
セクションB.5.5.3「NULL 値に関する問題」
セクション3.3.4.6「NULL 値の操作」
シングルパートインデックスの range アクセスメソッド
セクション12.3.2「比較関数と演算子」

IS NULL
EXISTS 戦略によるサブクエリーの最適化
セクション8.8.2「EXPLAIN 出力フォーマット」
セクション8.2.1.8「IS NULL の最適化」
セクションB.5.5.3「NULL 値に関する問題」
セクション3.3.4.6「NULL 値の操作」
セクション5.1.4「サーバーシステム変数」
シングルパートインデックスの range アクセスメソッド
マルチパートインデックスの range アクセスメソッド
セクション12.3.2「比較関数と演算子」

L

[索引の先頭]

LIKE
セクション8.3.8「B ツリーインデックスとハッシュインデックスの比較」
セクション11.4.1「CHAR および VARCHAR 型」
セクション13.8.3「HELP 構文」
セクションA.11「MySQL 5.6 FAQ: MySQL の中国語、日本語、および韓国語の文字セット」
セクション18.6.11「MySQL Cluster レプリケーションの競合解決」
セクション4.5.1.4「mysql サーバー側のヘルプ」
セクション11.4.5「SET 型」
セクション13.7.5.4「SHOW CHARACTER SET 構文」
セクション13.7.5.5「SHOW COLLATION 構文」
セクション13.7.5.6「SHOW COLUMNS 構文」
セクション13.7.5.15「SHOW DATABASES 構文」
セクション13.7.5.19「SHOW EVENTS 構文」
セクション13.7.5.25「SHOW OPEN TABLES 構文」
セクション13.7.5.29「SHOW PROCEDURE STATUS 構文」
セクション13.7.5.36「SHOW STATUS 構文」
セクション13.7.5.37「SHOW TABLE STATUS 構文」
セクション13.7.5.38「SHOW TABLES 構文」
セクション13.7.5.39「SHOW TRIGGERS 構文」
セクション13.7.5.40「SHOW VARIABLES 構文」
セクション10.1.9.3「SHOW ステートメントと INFORMATION_SCHEMA」
セクション21.32「SHOW ステートメントの拡張」
セクション6.2.3「アカウント名の指定」
セクション6.2.5「アクセス制御、ステージ 2: リクエストの確認」
インストゥルメントによる事前フィルタリング
セクション12.10「キャスト関数と演算子」
セクション18.5.9「クイックリファレンス: MySQL Cluster の SQL ステートメント」
セクション5.1.5「システム変数の使用」
シングルパートインデックスの range アクセスメソッド
セクション3.3.4.7「パターンマッチング」
マルチパートインデックスの range アクセスメソッド
セクション17.1.4.3「レプリケーションスレーブのオプションと変数」
セクション9.1.1「文字列リテラル」
セクション12.5.1「文字列比較関数」
セクション5.1.5.1「構造化システム変数」
セクション1.7.1「標準 SQL に対する MySQL 拡張機能」
セクション12.3.1「演算子の優先順位」

LIKE '_A%'
セクションA.11「MySQL 5.6 FAQ: MySQL の中国語、日本語、および韓国語の文字セット」

LIKE 'pattern'
セクション13.7.5「SHOW 構文」
マルチパートインデックスの range アクセスメソッド

LIKE ... ESCAPE
セクションB.5.8「MySQL の既知の問題」

N

[索引の先頭]

N % M

セクション12.6.2「数学関数」
セクション12.6.1「算術演算子」

N MOD M

セクション12.6.2「数学関数」
セクション12.6.1「算術演算子」

NOT

セクション5.1.7「サーバー SQL モード」
セクション12.3.3「論理演算子」

NOT LIKE

セクション3.3.4.7「パターンマッチング」
セクション12.5.1「文字列比較関数」

NOT REGEXP

セクション3.3.4.7「パターンマッチング」
セクション12.5.1「文字列比較関数」
セクション1.7.1「標準 SQL に対する MySQL 拡張機能」

NOT RLIKE

セクション3.3.4.7「パターンマッチング」
セクション12.5.1「文字列比較関数」

O

[索引の先頭]

OR

セクション3.6.7「2 つのキーを使用した検索」
EXISTS 戦略によるサブクエリーの最適化
セクション13.7.1.4「GRANT 構文」
セクション8.2.1.4「インデックスマージの最適化」
インデックスマージソート和集合アクセスアルゴリズム
インデックスマージ和集合アクセスアルゴリズム
セクション5.1.7「サーバー SQL モード」
シングルパートインデックスの range アクセスメソッド
マルチパートインデックスの range アクセスメソッド
セクション9.5「式の構文」
セクション12.5.1「文字列比較関数」
セクション1.7.1「標準 SQL に対する MySQL 拡張機能」
セクション12.3.1「演算子の優先順位」
セクション3.3.4.2「特定の行の選択」
セクション12.3.3「論理演算子」

R

[索引の先頭]

REGEXP

セクション3.3.4.7「パターンマッチング」
セクションD.7「文字セットの制約」
セクション1.7.1「標準 SQL に対する MySQL 拡張機能」
セクション12.5.2「正規表現」
セクション12.3.1「演算子の優先順位」

RLIKE

セクション3.3.4.7「パターンマッチング」
セクションD.7「文字セットの制約」
セクション12.5.2「正規表現」

X

[索引の先頭]

XOR

セクション12.19.1「GROUP BY (集約) 関数」
セクション12.3.3「論理演算子」

オプションの索引

シンボル | A | B | C | D | E | F | G | H | I | J | K | L | M | N | O | P | Q | R | S | T | U | V | W | X | Y | Z

シンボル

[索引の先頭]

--

セクション4.8.2「replace — 文字列置換ユーティリティー」
セクション1.7.2.5「コメントの先頭としての「--」」

-#

セクション4.4.1「comp_err — MySQL エラーメッセージファイルのコンパイル」
セクション24.4.3「DBUG パッケージ」
セクション4.7.3「my_print_defaults — オプションファイルからのオプションの表示」
セクション4.6.3.1「myisamchk の一般オプション」
セクション4.6.5「myisampack — 圧縮された読み取り専用の MyISAM テーブルの生成」
セクション4.5.1.1「mysql のオプション」
セクション4.6.6「mysql_config_editor — MySQL 構成ユーティリティー」
セクション4.4.7「mysql_upgrade — MySQL テーブルのチェックとアップグレード」
セクション4.5.2「mysqladmin — MySQL サーバーの管理を行うクライアント」
セクション4.6.8「mysqlbinlog — バイナリログファイルを処理するためのユーティリティー」
セクション4.5.3「mysqlcheck — テーブル保守プログラム」
セクション4.5.4「mysqldump — データベースバックアッププログラム」
セクション4.5.5「mysqlimport — データインポートプログラム」
セクション4.5.6「mysqlshow — データベース、テーブル、およびカラム情報の表示」
セクション4.5.7「mysqlslap — 負荷エミュレーションクライアント」
セクション4.8.2「replace — 文字列置換ユーティリティー」
セクション5.1.3「サーバーコマンドオプション」

/opt/mysql/server-5.6

セクション2.5.6「オラクルからの Debian パッケージを使用して MySQL を Linux にインストールする」

-1

セクション4.5.3「mysqlcheck — テーブル保守プログラム」

-?

セクション4.4.1「comp_err — MySQL エラーメッセージファイルのコンパイル」
セクション4.7.3「my_print_defaults — オプションファイルからのオプションの表示」
セクション4.6.2「myisam_ftdump — 全文インデックス情報の表示」
セクション4.6.3.1「myisamchk の一般オプション」
セクション4.6.4「myisamlog — MyISAM ログファイルの内容の表示」
セクション4.6.5「myisampack — 圧縮された読み取り専用の MyISAM テーブルの生成」
セクション18.4.27「MySQL Cluster プログラムに共通するオプション — MySQL Cluster プログラムに共通するオプション」
セクション4.5.1.1「mysql のオプション」
セクション1.3.2「MySQL の主な機能」
セクション4.6.6「mysql_config_editor — MySQL 構成ユーティリティー」
セクション4.4.4「mysql_plugin — MySQL サーバープラグインの構成」
セクション4.6.15「mysql_waitpid — プロセスを強制終了して終了を待機」
セクション4.6.16「mysql_zap — パターンに一致するプロセスを強制終了」
セクション4.6.7「mysqlaccess — アクセス権限をチェックするクライアント」
セクション4.5.2「mysqladmin — MySQL サーバーの管理を行うクライアント」
セクション4.6.8「mysqlbinlog — バイナリログファイルを処理するためのユーティリティー」
セクション4.5.3「mysqlcheck — テーブル保守プログラム」
セクション4.5.4「mysqldump — データベースバックアッププログラム」
セクション4.6.10「mysqlhotcopy — データベースバックアッププログラム」
セクション4.5.5「mysqlimport — データインポートプログラム」
セクション4.5.6「mysqlshow — データベース、テーブル、およびカラム情報の表示」
セクション4.5.7「mysqlslap — 負荷エミュレーションクライアント」
セクション18.4.7「ndb_config — MySQL Cluster 構成情報の抽出」
セクション18.4.16「ndb_print_file — NDB ディスクデータファイル内容の出力」
セクション4.8.1「perror — エラーコードの説明」
セクション4.8.2「replace — 文字列置換ユーティリティー」
セクション4.8.3「resolveip — ホスト名と IP アドレスの解決」
セクション4.2.4「コマンド行でのオプションの使用」
セクション5.1.3「サーバーコマンドオプション」

このスタイルのテキスト

セクション1.2「表記規則および構文規則」

A

[索引の先頭]

-A

セクション4.5.1.1「mysql のオプション」
セクション4.5.3「mysqlcheck — テーブル保守プログラム」
セクション4.5.4「mysqldump — データベースバックアッププログラム」
セクション18.4.20「ndb_restore — MySQL Cluster バックアップのリストア」
セクション4.6.3.4「その他の myisamchk オプション」

-a

セクション16.6.2「memcached の使用」
セクション7.6.4「MyISAM テーブルの最適化」
セクション18.5.2「MySQL Cluster 管理クライアントのコマンド」
セクション4.5.3「mysqlcheck — テーブル保守プログラム」

セクション4.6.9「mysqldumpslow — スロークエリーログファイルの要約」
セクション4.5.7「mysqlslap — 負荷エミュレーションクライアント」
セクション18.4.23「ndb_setup.py — MySQL Cluster のブラウザベース自動インストーラの開始」
セクション4.6.3.4「その他の myisamchk オプション」

--abort-slave-event-count
セクション17.1.4.3「レプリケーションスレーブのオプションと変数」

--add-drop-database
セクション4.5.4「mysqldump — データベースバックアッププログラム」
セクション7.4.1「mysqldump による SQL フォーマットでのデータのダンプ」

--add-drop-table
セクション4.5.4「mysqldump — データベースバックアッププログラム」

--add-drop-trigger
セクション4.5.4「mysqldump — データベースバックアッププログラム」

--add-locks
セクション4.5.4「mysqldump — データベースバックアッププログラム」

--addtodest
セクション4.6.10「mysqlhotcopy — データベースバックアッププログラム」

--all
セクション4.6.6「mysql_config_editor — MySQL 構成ユーティリティー」

--all-databases
セクション2.11.1.3「MySQL 5.5 から 5.6 へのアップグレード」
セクション4.4.7「mysql_upgrade — MySQL テーブルのチェックとアップグレード」
セクション4.5.3「mysqlcheck — テーブル保守プログラム」
セクション4.5.4「mysqldump — データベースバックアッププログラム」
セクション7.4.1「mysqldump による SQL フォーマットでのデータのダンプ」
セクション18.5.10「ndbinfo MySQL Cluster 情報データベース」
セクション7.4.2「SQL フォーマットバックアップのリロード」
セクション2.11.4「テーブルまたはインデックスの再作成または修復」
セクション4.6.8.3「バイナリログファイルのバックアップのための mysqlbinlog の使用」
セクション9.2.3「識別子とファイル名のマッピング」

--all-in-1
セクション4.5.3「mysqlcheck — テーブル保守プログラム」

--all-tablespaces
セクション4.5.4「mysqldump — データベースバックアッププログラム」

--allow-keywords
セクション4.5.4「mysqldump — データベースバックアッププログラム」

--allow-suspicious-udfs
セクション5.1.3「サーバーコマンドオプション」
セクション24.3.2.6「ユーザー定義関数のセキュリティー上の予防措置」

--allowold
セクション4.6.10「mysqlhotcopy — データベースバックアッププログラム」

--analyze
セクション7.6.4「MyISAM テーブルの最適化」
セクション4.6.3.1「myisamchk の一般オプション」
セクション4.5.3「mysqlcheck — テーブル保守プログラム」
セクション4.6.3.4「その他の myisamchk オプション」

--ansi
セクション1.7「MySQL の標準への準拠」
セクション5.1.3「サーバーコマンドオプション」

antonio
プロキシユーザーとグループマッピングを使用した Unix パスワード認証

--append
セクション18.4.20「ndb_restore — MySQL Cluster バックアップのリストア」

--apply-slave-statements
セクション4.5.4「mysqldump — データベースバックアッププログラム」

--audit-log
セクション6.3.12.1「監査ログプラグインのインストール」
セクション6.3.12.6「監査ログプラグインのオプションおよびシステム変数」

--auto-generate-sql
セクション4.5.7「mysqlslap — 負荷エミュレーションクライアント」

--auto-generate-sql-add-autoincrement
セクション4.5.7「mysqlslap — 負荷エミュレーションクライアント」

--auto-generate-sql-execute-number
セクション4.5.7「mysqlslap — 負荷エミュレーションクライアント」

--auto-generate-sql-guid-primary
セクション4.5.7「mysqlslap — 負荷エミュレーションクライアント」

--auto-generate-sql-load-type
セクション4.5.7「mysqlslap — 負荷エミュレーションクライアント」

--auto-generate-sql-secondary-indexes
セクション4.5.7「mysqlslap — 負荷エミュレーションクライアント」

--auto-generate-sql-unique-query-number
セクション4.5.7「mysqlslap — 負荷エミュレーションクライアント」

--auto-generate-sql-unique-write-number
セクション4.5.7「mysqlslap — 負荷エミュレーションクライアント」

--auto-generate-sql-write-number
セクション4.5.7「mysqlslap — 負荷エミュレーションクライアント」

--auto-rehash
セクション4.5.1.1「mysql のオプション」
セクション4.5.1.2「mysql コマンド」
セクション14.13.16.2「非永続的オプティマイザ統計のパラメータの構成」

--auto-repair
セクション4.5.3「mysqlcheck — テーブル保守プログラム」

--auto-vertical-output
セクション4.5.1.1「mysql のオプション」

--autocommit
セクション5.1.4「サーバーシステム変数」

B

[索引の先頭]

-B
セクション16.6.2「memcached の使用」
セクション4.6.3.3「myisamchk の修復オプション」
セクション4.5.1.1「mysql のオプション」
セクション4.5.3「mysqlcheck — テーブル保守プログラム」
セクション4.5.4「mysqldump — データベースバックアッププログラム」

-b
セクション16.6.2「memcached の使用」
セクション4.6.5「myisampack — 圧縮された読み取り専用の MyISAM テーブルの生成」
セクション4.5.1.1「mysql のオプション」
セクション4.4.4「mysql_plugin — MySQL サーバープラグインの構成」
セクション4.6.7「mysqlaccess — アクセス権限をチェックするクライアント」
セクション4.5.2「mysqladmin — MySQL サーバーの管理を行うクライアント」
セクション18.4.10「ndb_desc — NDB テーブルの表示」
セクション18.4.20「ndb_restore — MySQL Cluster バックアップのリストア」
セクション4.6.3.4「その他の myisamchk オプション」
セクション5.1.3「サーバーコマンドオプション」

--back_log
セクション2.7「Solaris および OpenSolaris に MySQL をインストールする」

--backup
セクション4.6.3.3「myisamchk の修復オプション」
セクション4.6.5「myisampack — 圧縮された読み取り専用の MyISAM テーブルの生成」

--backup_path
セクション18.4.20「ndb_restore — MySQL Cluster バックアップのリストア」

backup_path
セクション18.4.20「ndb_restore — MySQL Cluster バックアップのリストア」

--backupid
セクション18.4.20「ndb_restore — MySQL Cluster バックアップのリストア」

--base64-output
セクション17.1.3.1「GTID の概念」
セクション4.6.8「mysqlbinlog — バイナリログファイルを処理するためのユーティリティー」
セクション4.6.8.2「mysqlbinlog 行イベントの表示」
セクション17.1.2.1「ステートメントベースおよび行ベースレプリケーションのメリットとデメリット」
セクション17.1.2.2「行ベースロギングおよびレプリケーションの使用」

--basedir
セクション5.3「1 つのマシン上での複数の MySQL インスタンスの実行」
セクション18.1.6.1「MySQL Cluster での SQL 構文の不適合」
セクション2.10.1.3「MySQL サーバーの起動とトラブルシューティング」
セクション2.9.4「MySQL ソース構成オプション」
セクション4.3.3「mysql.server — MySQL サーバー起動スクリプト」
セクション4.4.3「mysql_install_db — MySQL データディレクトリの初期化」
セクション4.4.4「mysql_plugin — MySQL サーバープラグインの構成」
セクション4.4.7「mysql_upgrade — MySQL テーブルのチェックとアップグレード」
セクション4.3.2「mysqld_safe — MySQL サーバー起動スクリプト」

セクション2.10.1「Unix 類似システムでのインストール後の手順」
セクション5.1.3「サーバーコマンドオプション」
セクション5.1.4「サーバーシステム変数」

basedir

セクション2.3.6「Microsoft Windows MySQL Server インストールのトラブルシューティング」
セクション2.10.1.2「MySQL を自動的に起動および停止する」
セクション2.3.5.2「オプションファイルの作成」

--batch

セクション4.5.1.1「mysql のオプション」
セクション4.5.1.3「mysql のロギング」

--big-tables

セクション5.1.3「サーバーコマンドオプション」

--binary-mode

セクション4.5.1.1「mysql のオプション」
セクション4.5.1.2「mysql コマンド」
セクション4.6.8「mysqlbinlog — バイナリログファイルを処理するためのユーティリティー」

--bind-address

セクション5.3「1 つのマシン上での複数の MySQL インスタンスの実行」
セクションB.5.2.2「[ローカルの] MySQL サーバーに接続できません」
セクション5.1.9「IPv6 サポート」
セクション5.1.9.3「IPv6 ローカルホストアドレスを使用した接続」
セクション5.1.9.2「IPv6 接続を許可するための MySQL Server の構成」
セクション5.1.9.4「IPv6 非ローカルホストアドレスを使用した接続」
セクション4.5.1.1「mysql のオプション」
セクション4.5.2「mysqladmin — MySQL サーバーの管理を行うクライアント」
セクション4.6.8「mysqlbinlog — バイナリログファイルを処理するためのユーティリティー」
セクション4.5.3「mysqlcheck — テーブル保守プログラム」
セクション4.3.4「mysqld_multi — 複数の MySQL サーバーの管理」
セクション4.5.4「mysqldump — データベースバックアッププログラム」
セクション4.5.5「mysqlimport — データインポートプログラム」
セクション4.5.6「mysqlshow — データベース、テーブル、およびカラム情報の表示」
セクション18.4.4「ndb_mgmd — MySQL Cluster 管理サーバーデーモン」
セクション18.4.1「ndbd — MySQL Cluster データノードデーモン」
セクション6.2.7「アクセス拒否エラーの原因」
セクション5.1.3「サーバーコマンドオプション」
セクション5.1.4「サーバーシステム変数」
セクション5.1.9.5「ブローカからの IPv6 アドレスの入手」

--binlog-checksum

セクション17.1.4.4「バイナリログのオプションと変数」

--binlog-do-db

セクション18.6.3「MySQL Cluster レプリケーションの既知の問題」
セクション4.6.8「mysqlbinlog — バイナリログファイルを処理するためのユーティリティー」
セクション17.2.3「サーバーがレプリケーションフィルタリングルールをどのように評価するか」
セクション17.2.3.1「データベースレベルレプリケーションオプションおよびバイナリロギングオプションの評価」
セクション5.2.4「バイナリログ」
セクション17.1.4.4「バイナリログのオプションと変数」
セクション17.1.4.3「レプリケーションスレーブのオプションと変数」

--binlog-format

セクションA.4「MySQL 5.6 FAQ: ストアドプロシージャーおよびストアドファンクション」
セクション18.1.6.6「MySQL Cluster の未サポート機能または欠落している機能」
セクション18.6.2「MySQL Cluster レプリケーションの一般要件」
セクション18.6.6「MySQL Cluster レプリケーションの起動 (レプリケーションチャネルが 1 つ)」
セクション5.1.3「サーバーコマンドオプション」
セクション5.2.4.1「バイナリロギング形式」
セクション17.1.4.4「バイナリログのオプションと変数」
セクション5.2.4.2「バイナリログ形式の設定」

--binlog-ignore-db

セクション18.6.3「MySQL Cluster レプリケーションの既知の問題」
セクション17.2.3「サーバーがレプリケーションフィルタリングルールをどのように評価するか」
セクション17.2.3.1「データベースレベルレプリケーションオプションおよびバイナリロギングオプションの評価」
セクション5.2.4「バイナリログ」
セクション17.1.4.4「バイナリログのオプションと変数」
セクション17.1.4.3「レプリケーションスレーブのオプションと変数」

--binlog-row-event-max-size

セクション2.11.1.3「MySQL 5.5 から 5.6 へのアップグレード」
セクション4.6.8「mysqlbinlog — バイナリログファイルを処理するためのユーティリティー」
セクション5.1.2.1「サーバーのデフォルト値への変更」
セクション17.1.4.4「バイナリログのオプションと変数」
セクション5.2.4.2「バイナリログ形式の設定」

--binlog-rows-query-log-events

セクション17.1.4.4「バイナリログのオプションと変数」

--blob-info

セクション18.4.10「ndb_desc — NDB テーブルの表示」

--block-search

セクション4.6.3.4「その他の myisamchk オプション」

--bootstrap

セクション5.1.3「サーバーコマンドオプション」

--brief
セクション4.6.7「mysqlaccess — アクセス権限をチェックするクライアント」

--browser-start-page
セクション18.4.23「ndb_setup.py — MySQL Cluster のブラウザベース自動インストーラの開始」

--builddir
セクション4.4.3「mysql_install_db — MySQL データディレクトリの初期化」

C

[索引の先頭]

-C
セクション4.4.1「comp_err — MySQL エラーメッセージファイルのコンパイル」
セクション16.6.2「memcached の使用」
セクション4.6.3.2「myisamchk のチェックオプション」
セクション4.5.1.1「mysql のオプション」
セクション4.5.2「mysqladmin — MySQL サーバーの管理を行うクライアント」
セクション4.5.3「mysqlcheck — テーブル保守プログラム」
セクション4.5.4「mysqldump — データベースバックアッププログラム」
セクション4.5.5「mysqlimport — データインポートプログラム」
セクション4.5.6「mysqlshow — データベース、テーブル、およびカラム情報の表示」
セクション4.5.7「mysqlslap — 負荷エミュレーションクライアント」
セクション5.1.3「サーバーコマンドオプション」

-c
セクション4.6.1「innochecksum — オフライン InnoDB ファイルチェックサムユーティリティー」
セクション16.6.2「memcached の使用」
セクション4.7.3「my_print_defaults — オプションファイルからのオプションの表示」
セクション4.6.2「myisam_ftdump — 全文インデックス情報の表示」
セクション4.6.3.2「myisamchk のチェックオプション」
セクション4.6.4「myisamlog — MyISAM ログファイルの内容の表示」
セクションA.10「MySQL 5.6 FAQ: MySQL Cluster」
セクション18.4.27「MySQL Cluster プログラムに共通するオプション — MySQL Cluster プログラムに共通するオプション」
セクション4.5.1.1「mysql のオプション」
セクション4.5.2「mysqladmin — MySQL サーバーの管理を行うクライアント」
セクション4.6.8「mysqlbinlog — バイナリログファイルを処理するためのユーティリティー」
セクション4.5.3「mysqlcheck — テーブル保守プログラム」
セクション4.5.4「mysqldump — データベースバックアッププログラム」
セクション4.5.5「mysqlimport — データインポートプログラム」
セクション4.5.7「mysqlslap — 負荷エミュレーションクライアント」
セクション18.4.7「ndb_config — MySQL Cluster 構成情報の抽出」
セクション18.4.20「ndb_restore — MySQL Cluster バックアップのリストア」
セクション18.4.23「ndb_setup.py — MySQL Cluster のブラウザベース自動インストーラの開始」
セクション18.2.3.1「Windows でのバイナリリリースからの MySQL Cluster のインストール」

--ca-certs-file
セクション18.2.1.1「MySQL Cluster Auto-Installer の要件」
セクション18.4.23「ndb_setup.py — MySQL Cluster のブラウザベース自動インストーラの開始」

--cert-file
セクション18.2.1.1「MySQL Cluster Auto-Installer の要件」
セクション18.4.23「ndb_setup.py — MySQL Cluster のブラウザベース自動インストーラの開始」

--cflags
セクション2.9.5「MySQL のコンパイルに関する問題」
セクション4.7.2「mysql_config — クライアントのコンパイル用オプションの表示」

--character-set-client-handshake
cp932 文字セット
セクションA.11「MySQL 5.6 FAQ: MySQL の中国語、日本語、および韓国語の文字セット」
セクション5.1.3「サーバーコマンドオプション」

--character-set-filesystem
セクション5.1.3「サーバーコマンドオプション」

--character-set-server
セクションA.11「MySQL 5.6 FAQ: MySQL の中国語、日本語、および韓国語の文字セット」
セクション10.1.5「アプリケーションの文字セットおよび照合順序の構成」
セクション5.1.3「サーバーコマンドオプション」
セクション10.1.3.1「サーバー文字セットおよび照合順序」
セクション17.4.1.3「レプリケーションと文字セット」
セクション10.5「文字セットの構成」

--character-sets-dir
セクション4.6.3.3「myisamchk の修復オプション」
セクション4.6.5「myisampack — 圧縮された読み取り専用の MyISAM テーブルの生成」
セクション18.4.27「MySQL Cluster プログラムに共通するオプション — MySQL Cluster プログラムに共通するオプション」
セクション4.5.1.1「mysql のオプション」
セクション4.4.7「mysql_upgrade — MySQL テーブルのチェックとアップグレード」
セクション4.5.2「mysqladmin — MySQL サーバーの管理を行うクライアント」
セクション4.6.8「mysqlbinlog — バイナリログファイルを処理するためのユーティリティー」
セクション4.5.3「mysqlcheck — テーブル保守プログラム」
セクション4.5.4「mysqldump — データベースバックアッププログラム」
セクション4.5.5「mysqlimport — データインポートプログラム」

セクション4.5.6「mysqlshow — データベース、テーブル、およびカラム情報の表示」
セクション5.1.3「サーバーコマンドオプション」
セクション10.5「文字セットの構成」
セクションB.5.2.17「文字セットを初期化できません」

--character_set_server
セクション2.9.4「MySQL ソース構成オプション」

--charset
セクション4.4.1「comp_err — MySQL エラーメッセージファイルのコンパイル」

--check
セクション4.6.3.2「myisamchk のチェックオプション」
セクション4.5.3「mysqlcheck — テーブル保守プログラム」

--check-only-changed
セクション4.6.3.2「myisamchk のチェックオプション」
セクション4.5.3「mysqlcheck — テーブル保守プログラム」

--check-orphans
セクション18.4.6「ndb_blob_tool — MySQL Cluster テーブルの BLOB および TEXT カラムのチェックおよび修復」

--check-upgrade
セクション4.4.7「mysql_upgrade — MySQL テーブルのチェックとアップグレード」
セクション4.5.3「mysqlcheck — テーブル保守プログラム」

--checkpoint
セクション4.6.10「mysqlhotcopy — データベースバックアッププログラム」

--chroot
セクション4.6.10「mysqlhotcopy — データベースバックアッププログラム」
セクション5.1.3「サーバーコマンドオプション」

CMAKE_BUILD_TYPE
セクション2.9.4「MySQL ソース構成オプション」

CMAKE_C_FLAGS
セクション2.9.5「MySQL のコンパイルに関する問題」
セクション2.9.4「MySQL ソース構成オプション」
セクション24.4.1.1「デバッグのための MySQL のコンパイル」

CMAKE_C_FLAGS_build_type
セクション2.9.4「MySQL ソース構成オプション」

CMAKE_C_FLAGS_RELWITHDEBINFO
セクション2.9.4「MySQL ソース構成オプション」

CMAKE_CXX_FLAGS
セクション2.9.5「MySQL のコンパイルに関する問題」
セクション2.9.4「MySQL ソース構成オプション」
セクション24.4.1.1「デバッグのための MySQL のコンパイル」

CMAKE_CXX_FLAGS_build_type
セクション2.9.4「MySQL ソース構成オプション」

CMAKE_CXX_FLAGS_RELWITHDEBINFO
セクション2.9.4「MySQL ソース構成オプション」

CMAKE_INSTALL_PREFIX
セクション2.9.4「MySQL ソース構成オプション」
セクション5.3.3「Unix 上での複数の MySQL インスタンスの実行」
セクション24.2.4.3「プラグインライブラリのコンパイルおよびインストール」
セクション2.9.3「開発ソースツリーを使用して MySQL をインストールする」

CMAKE_PREFIX_PATH
セクション2.9.4「MySQL ソース構成オプション」

--collation-server
セクション10.1.5「アプリケーションの文字セットおよび照合順序の構成」
セクション5.1.3「サーバーコマンドオプション」
セクション10.1.3.1「サーバー文字セットおよび照合順序」
セクション17.4.1.3「レプリケーションと文字セット」
セクション10.5「文字セットの構成」

--collation_server
セクション2.9.4「MySQL ソース構成オプション」

--column-names
セクション4.5.1.1「mysql のオプション」
セクション4.2.5「プログラムオプション修飾子」

--column-type-info
セクション4.5.1.1「mysql のオプション」

--columns
セクション4.5.5「mysqlimport — データインポートプログラム」

--comments
セクション4.5.1.1「mysql のオプション」
セクション4.5.4「mysqldump — データベースバックアッププログラム」

--commit
セクション4.6.7「mysqlaccess — アクセス権限をチェックするクライアント」
セクション4.5.7「mysqlslap — 負荷エミュレーションクライアント」

--comp
セクション4.2.3「プログラムオプションの指定」

--compact
セクション4.5.4「mysqldump — データベースバックアッププログラム」

--compatible
セクション4.5.4「mysqldump — データベースバックアッププログラム」

COMPILATION_COMMENT
セクション5.1.4「サーバーシステム変数」

--complete-insert
セクション4.5.4「mysqldump — データベースバックアッププログラム」

--compr
セクション4.2.3「プログラムオプションの指定」

--compress
セクション13.2.6「LOAD DATA INFILE 構文」
セクション4.5.1.1「mysql のオプション」
セクション4.4.7「mysql_upgrade — MySQL テーブルのチェックとアップグレード」
セクション4.5.2「mysqladmin — MySQL サーバーの管理を行うクライアント」
セクション4.5.3「mysqlcheck — テーブル保守プログラム」
セクション4.5.4「mysqldump — データベースバックアッププログラム」
セクション4.5.5「mysqlimport — データインポートプログラム」
セクション4.5.6「mysqlshow — データベース、テーブル、およびカラム情報の表示」
セクション4.5.7「mysqlslap — 負荷エミュレーションクライアント」
セクション4.2.3「プログラムオプションの指定」

--concurrency
セクション4.5.7「mysqlslap — 負荷エミュレーションクライアント」

--config-cache
セクション18.3.2「MySQL Cluster の構成ファイル」
セクション18.4.4「ndb_mgmd — MySQL Cluster 管理サーバーデーモン」

--config-dir
セクション18.4.4「ndb_mgmd — MySQL Cluster 管理サーバーデーモン」

--config-file
セクション4.7.3「my_print_defaults — オプションファイルからのオプションの表示」
セクションA.10「MySQL 5.6 FAQ: MySQL Cluster」
セクション18.2.5「MySQL Cluster の初期起動」
セクション18.3.2.1「MySQL Cluster 構成の基本的な例」
セクション18.4.7「ndb_config — MySQL Cluster 構成情報の抽出」
セクション18.4.4「ndb_mgmd — MySQL Cluster 管理サーバーデーモン」
セクション18.2.3.3「Windows での MySQL Cluster の初期起動」
セクション18.2.3.4「Windows サービスとしての MySQL Cluster プロセスのインストール」

--config_from_node
セクション18.4.7「ndb_config — MySQL Cluster 構成情報の抽出」

--configdir
セクション18.3.2「MySQL Cluster の構成ファイル」
セクション18.4.4「ndb_mgmd — MySQL Cluster 管理サーバーデーモン」

--configinfo
セクション18.4.7「ndb_config — MySQL Cluster 構成情報の抽出」

--connect-delay
セクション18.4.1「ndbd — MySQL Cluster データノードデーモン」

--connect-expired-password
セクション4.5.1.1「mysql のオプション」

--connect-retries
セクション18.4.1「ndbd — MySQL Cluster データノードデーモン」

--connect-string
セクション18.4.27「MySQL Cluster プログラムに共通するオプション — MySQL Cluster プログラムに共通するオプション」

--connection-server-id
セクション4.6.8「mysqlbinlog — バイナリログファイルを処理するためのユーティリティー」

--connection-timeout
セクション18.4.13「ndb_error_reporter — NDB エラーレポートユーティリティー」

--connections
セクション18.4.7「ndb_config — MySQL Cluster 構成情報の抽出」

--console
セクション14.19「InnoDB のトラブルシューティング」
セクション14.5.1「InnoDB テーブルスペースの作成」
セクション14.15.2「InnoDB モニターの有効化」
root のパスワードのリセット: Windows システム
セクション18.2.3.3「Windows での MySQL Cluster の初期起動」
セクション2.3.5.5「Windows のコマンド行からの MySQL の起動」
セクション5.2.2「エラーログ」
セクション2.3.5.4「サーバーをはじめて起動する」
セクション5.1.3「サーバーコマンドオプション」

--copy
セクション4.6.7「mysqlaccess — アクセス権限をチェックするクライアント」

--core-file
セクション24.4.1.4「gdb での mysqld のデバッグ」

セクション18.4.27「MySQL Cluster プログラムに共通するオプション — MySQL Cluster プログラムに共通するオプション」
セクション5.1.3「サーバーコマンドオプション」
セクション5.1.4「サーバーシステム変数」

--core-file-size

セクション4.3.2「mysqld_safe — MySQL サーバー起動スクリプト」
セクション5.1.3「サーバーコマンドオプション」

--correct-checksum

セクション4.6.3.3「myisamchk の修復オプション」

--count

セクション4.6.2「myisam_ftdump — 全文インデックス情報の表示」
セクション4.5.2「mysqladmin — MySQL サーバーの管理を行うクライアント」
セクション4.5.6「mysqlshow — データベース、テーブル、およびカラム情報の表示」

--create

セクション4.5.7「mysqlslap — 負荷エミュレーションクライアント」

--create-options

セクション4.5.4「mysqldump — データベースバックアッププログラム」

--create-schema

セクション4.5.7「mysqlslap — 負荷エミュレーションクライアント」

--cross-bootstrap

セクション4.4.3「mysql_install_db — MySQL データディレクトリの初期化」

--csv

セクション4.5.7「mysqlslap — 負荷エミュレーションクライアント」

--cxxflags

セクション2.9.5「MySQL のコンパイルに関する問題」
セクション4.7.2「mysql_config — クライアントのコンパイル用オプションの表示」

D

[索引の先頭]

-D

セクション15.5「ARCHIVE ストレージエンジン」
セクション15.6「BLACKHOLE ストレージエンジン」
セクション4.4.1「comp_err — MySQL エラーメッセージファイルのコンパイル」
セクション5.4「DTrace を使用した mysqld のトレース」
セクション15.9「EXAMPLE ストレージエンジン」
セクション15.8「FEDERATED ストレージエンジン」
セクション14.18.3.1「InnoDB memcached プラグインの前提条件」
セクション23.6.1「libmysqld によるプログラムのコンパイル」
セクション18.2.2.3「Linux でのソースからの MySQL Cluster のビルド」
セクション6.1.6「LOAD DATA LOCAL のセキュリティーの問題」
セクション16.6.2「memcached の使用」
セクション4.6.3.3「myisamchk の修復オプション」
セクション18.5.4「MySQL Cluster での MySQL サーバーの使用法」
セクションB.5.4.2「MySQL が繰り返しクラッシュする場合の対処方法」
セクション4.5.1.1「mysql のオプション」
セクション24.1.1「MySQL のスレッド」
セクション24.4.2「MySQL クライアントのデバッグ」
セクション2.9.4「MySQL ソース構成オプション」
セクション4.6.8「mysqlbinlog — バイナリログファイルを処理するためのユーティリティー」
セクション4.5.5「mysqlimport — データインポートプログラム」
セクション18.4.21「ndb_select_all — NDB テーブルの行の出力」
セクション18.2.3.2「Windows でのソースからの MySQL Cluster のコンパイルとインストール」
セクション2.1.2「インストールする MySQL のバージョンと配布の選択」
セクション5.1.3「サーバーコマンドオプション」
セクション5.1.4「サーバーシステム変数」
セクション24.4.1.1「デバッグのための MySQL のコンパイル」
セクション22.2.1「パフォーマンススキーマビルド構成」
第19章「パーティション化」
セクション17.1.4.3「レプリケーションスレーブのオプションと変数」
セクション10.3「文字セットの追加」
セクションB.5.2.17「文字セットを初期化できません」
セクション2.9.2「標準ソース配布を使用して MySQL をインストールする」

-d

セクション4.6.1「innochecksum — オフライン InnoDB ファイルチェックサムユーティリティー」
セクション16.6.2「memcached の使用」
セクション4.6.2「myisam_ftdump — 全文インデックス情報の表示」
セクション4.6.3.1「myisamchk の一般オプション」
セクション4.4.4「mysql_plugin — MySQL サーバープラグインの構成」
セクション4.6.7「mysqlaccess — アクセス権限をチェックするクライアント」
セクション4.6.8「mysqlbinlog — バイナリログファイルを処理するためのユーティリティー」
セクション4.5.4「mysqldump — データベースバックアッププログラム」
セクション4.6.9「mysqldumpslow — スロークエリーログファイルの要約」
セクション18.4.6「ndb_blob_tool — MySQL Cluster テーブルの BLOB および TEXT カラムのチェックおよび修復」
セクション18.4.10「ndb_desc — NDB テーブルの表示」
セクション18.4.14「ndb_index_stat — NDB インデックス統計ユーティリティー」
セクション18.4.4「ndb_mgmd — MySQL Cluster 管理サーバーデーモン」

セクション18.4.20「ndb_restore — MySQL Cluster バックアップのリストア」
セクション18.4.21「ndb_select_all — NDB テーブルの行の出力」
セクション18.4.23「ndb_setup.py — MySQL Cluster のブラウザベース自動インストーラの開始」
セクション18.4.24「ndb_show_tables — NDB テーブルのリストの表示」
セクション18.4.1「ndbd — MySQL Cluster データノードデーモン」
セクション4.6.3.4「その他の myisamchk オプション」
セクション5.1.4「サーバーシステム変数」

--daemon

セクション18.4.4「ndb_mgmd — MySQL Cluster 管理サーバーデーモン」
セクション18.4.1「ndbd — MySQL Cluster データノードデーモン」

--data-file-length

セクション4.6.3.3「myisamchk の修復オプション」

--database

セクション4.5.1.1「mysql のオプション」
セクション4.6.8「mysqlbinlog — バイナリログファイルを処理するためのユーティリティー」
セクション18.4.6「ndb_blob_tool — MySQL Cluster テーブルの BLOB および TEXT カラムのチェックおよび修復」
セクション18.4.10「ndb_desc — NDB テーブルの表示」
セクション18.4.14「ndb_index_stat — NDB インデックス統計ユーティリティー」
セクション18.4.21「ndb_select_all — NDB テーブルの行の出力」
セクション18.4.24「ndb_show_tables — NDB テーブルのリストの表示」
セクション18.4.25「ndb_size.pl — NDBCLUSTER サイズ要件エスティメータ」

--databases

セクション4.5.3「mysqlcheck — テーブル保守プログラム」
セクション4.5.4「mysqldump — データベースバックアッププログラム」
セクション7.4.1「mysqldump による SQL フォーマットでのデータのダンプ」
セクション18.5.10「ndbinfo MySQL Cluster 情報データベース」
セクション7.4.2「SQL フォーマットバックアップのリロード」
セクション7.4.5.2「サーバー間でのデータベースのコピー」
セクション2.11.4「テーブルまたはインデックスの再作成または修復」
セクション7.4.5.1「データベースのコピーの作成」

--datadir

セクション5.3「1 つのマシン上での複数の MySQL インスタンスの実行」
セクション18.5.11.3「MySQL Cluster と MySQL のセキュリティー手順」
セクション2.10.1.3「MySQL サーバーの起動とトラブルシューティング」
セクション2.9.4「MySQL ソース構成オプション」
セクション4.3.3「mysql.server — MySQL サーバー起動スクリプト」
セクション4.4.3「mysql_install_db — MySQL データディレクトリの初期化」
セクション4.4.4「mysql_plugin — MySQL サーバープラグインの構成」
セクション4.4.7「mysql_upgrade — MySQL テーブルのチェックとアップグレード」
セクション4.3.2「mysqld_safe — MySQL サーバー起動スクリプト」
セクション5.3.3「Unix 上での複数の MySQL インスタンスの実行」
セクション2.10.1「Unix 類似システムでのインストール後の手順」
セクション2.3.5.2「オプションファイルの作成」
セクション4.2.6「オプションファイルの使用」
セクション5.1.3「サーバーコマンドオプション」
セクション5.1.4「サーバーシステム変数」
セクション5.3.1「複数のデータディレクトリのセットアップ」

datadir

セクション16.4.1「EC2 AMI での MySQL のセットアップ」
セクション2.3.6「Microsoft Windows MySQL Server インストールのトラブルシューティング」
セクション2.10.1.2「MySQL を自動的に起動および停止する」
セクション2.4.1「OS X への MySQL のインストールに関する一般的な注記」
セクションD.10.6「Windows プラットフォームの制限」
セクション2.3.5.2「オプションファイルの作成」

--db

セクション4.6.7「mysqlaccess — アクセス権限をチェックするクライアント」

--debug

セクション4.4.1「comp_err — MySQL エラーメッセージファイルのコンパイル」
セクション24.4.3「DBUG パッケージ」
セクション4.7.3「my_print_defaults — オプションファイルからのオプションの表示」
セクション4.6.3.1「myisamchk の一般オプション」
セクション4.6.5「myisampack — 圧縮された読み取り専用の MyISAM テーブルの生成」
セクション18.4.27「MySQL Cluster プログラムに共通するオプション — MySQL Cluster プログラムに共通するオプション」
セクション4.5.1.1「mysql のオプション」
セクション2.10.1.3「MySQL サーバーの起動とトラブルシューティング」
セクション2.9.4「MySQL ソース構成オプション」
セクション4.6.6「mysql_config_editor — MySQL 構成ユーティリティー」
セクション4.4.7「mysql_upgrade — MySQL テーブルのチェックとアップグレード」
セクション4.6.7「mysqlaccess — アクセス権限をチェックするクライアント」
セクション4.5.2「mysqladmin — MySQL サーバーの管理を行うクライアント」
セクション4.6.8「mysqlbinlog — バイナリログファイルを処理するためのユーティリティー」
セクション4.5.3「mysqlcheck — テーブル保守プログラム」
セクション4.5.4「mysqldump — データベースバックアッププログラム」

セクション4.6.9「mysqldumpslow — スロークエリーログファイルの要約」
セクション4.6.10「mysqlhotcopy — データベースバックアッププログラム」
セクション4.5.5「mysqlimport — データインポートプログラム」
セクション4.5.6「mysqlshow — データベース、テーブル、およびカラム情報の表示」
セクション4.5.7「mysqlslap — 負荷エミュレーションクライアント」
セクション2.3.5.5「Windows のコマンド行からの MySQL の起動」
セクション6.2.7「アクセス拒否エラーの原因」
セクション5.1.3「サーバーコマンドオプション」
セクション5.1.4「サーバーシステム変数」
セクション24.4.1.1「デバッグのための MySQL のコンパイル」

--debug-check

セクション4.5.1.1「mysql のオプション」
セクション4.4.7「mysql_upgrade — MySQL テーブルのチェックとアップグレード」
セクション4.5.2「mysqladmin — MySQL サーバーの管理を行うクライアント」
セクション4.6.8「mysqlbinlog — バイナリログファイルを処理するためのユーティリティー」
セクション4.5.3「mysqlcheck — テーブル保守プログラム」
セクション4.5.4「mysqldump — データベースバックアッププログラム」
セクション4.5.5「mysqlimport — データインポートプログラム」
セクション4.5.6「mysqlshow — データベース、テーブル、およびカラム情報の表示」
セクション4.5.7「mysqlslap — 負荷エミュレーションクライアント」

--debug-info

セクション4.4.1「comp_err — MySQL エラーメッセージファイルのコンパイル」
セクション4.5.1.1「mysql のオプション」
セクション4.4.7「mysql_upgrade — MySQL テーブルのチェックとアップグレード」
セクション4.5.2「mysqladmin — MySQL サーバーの管理を行うクライアント」
セクション4.6.8「mysqlbinlog — バイナリログファイルを処理するためのユーティリティー」
セクション4.5.3「mysqlcheck — テーブル保守プログラム」
セクション4.5.4「mysqldump — データベースバックアッププログラム」
セクション4.5.5「mysqlimport — データインポートプログラム」
セクション4.5.6「mysqlshow — データベース、テーブル、およびカラム情報の表示」
セクション4.5.7「mysqlslap — 負荷エミュレーションクライアント」

--debug-level

セクション18.4.23「ndb_setup.py — MySQL Cluster のブラウザベース自動インストーラの開始」

--debug-sync-timeout

セクション2.9.4「MySQL ソース構成オプション」
セクション5.1.3「サーバーコマンドオプション」
セクション5.1.4「サーバーシステム変数」

--default-auth

セクション23.7.14「C API クライアントプラグイン関数」
セクション4.5.1.1「mysql のオプション」
セクション4.4.7「mysql_upgrade — MySQL テーブルのチェックとアップグレード」
セクション4.5.2「mysqladmin — MySQL サーバーの管理を行うクライアント」
セクション4.6.8「mysqlbinlog — バイナリログファイルを処理するためのユーティリティー」
セクション4.5.3「mysqlcheck — テーブル保守プログラム」
セクション4.5.4「mysqldump — データベースバックアッププログラム」
セクション4.5.5「mysqlimport — データインポートプログラム」
セクション4.5.6「mysqlshow — データベース、テーブル、およびカラム情報の表示」
セクション4.5.7「mysqlslap — 負荷エミュレーションクライアント」
クライアントプラグインディスクリプタ
セクション6.3.8.1「ネイティブ認証プラグイン」
セクション6.3.7「プラガブル認証」
セクション6.3.8.2「古いネイティブ認証プラグイン」
認証プラグインの使用

--default-authentication-plugin

セクション13.7.1.2「CREATE USER 構文」
セクション5.1.3「サーバーコマンドオプション」
セクション5.1.4「サーバーシステム変数」
セクション6.3.7「プラガブル認証」

--default-character-set

セクション13.2.6「LOAD DATA INFILE 構文」
セクション4.5.1.1「mysql のオプション」
セクション4.4.7「mysql_upgrade — MySQL テーブルのチェックとアップグレード」
セクション4.5.2「mysqladmin — MySQL サーバーの管理を行うクライアント」
セクション4.5.3「mysqlcheck — テーブル保守プログラム」
セクション4.5.4「mysqldump — データベースバックアッププログラム」
セクション4.5.5「mysqlimport — データインポートプログラム」
セクション4.5.6「mysqlshow — データベース、テーブル、およびカラム情報の表示」
Windows における Unicode のサポート
セクション10.1.5「アプリケーションの文字セットおよび照合順序の構成」
セクション5.1.4「サーバーシステム変数」
セクション4.5.1.5「テキストファイルから SQL ステートメントを実行する」
セクション6.3.1「ユーザー名とパスワード」
セクション10.1.4「接続文字セットおよび照合順序」
セクション10.5「文字セットの構成」

--default-storage-engine

セクション14.1.3「InnoDB の無効化」
セクション14.12「InnoDB の起動オプションおよびシステム変数」
セクション5.1.3「サーバーコマンドオプション」
セクション5.1.4「サーバーシステム変数」
セクション15.1「ストレージエンジンの設定」

default-storage-engine
セクション15.1「ストレージエンジンの設定」

--default-time-zone
セクション10.6「MySQL Server でのタイムゾーンのサポート」
セクション5.1.3「サーバーコマンドオプション」
セクション5.1.4「サーバーシステム変数」
セクション17.4.1.30「レプリケーションとタイムゾーン」

--default-tmp-storage-engine
セクション14.1.3「InnoDB の無効化」
セクション14.12「InnoDB の起動オプションおよびシステム変数」
セクション5.1.3「サーバーコマンドオプション」

--default.key_buffer_size
セクション5.1.5.1「構造化システム変数」

DEFAULT_CHARSET
セクション10.1.5「アプリケーションの文字セットおよび照合順序の構成」
セクション10.1.3.1「サーバー文字セットおよび照合順序」

DEFAULT_COLLATION
セクション10.1.5「アプリケーションの文字セットおよび照合順序の構成」
セクション10.1.3.1「サーバー文字セットおよび照合順序」

--defaults-extra-file
セクション4.7.3「my_print_defaults — オプションファイルからのオプションの表示」
セクション4.6.3.1「myisamchk の一般オプション」
セクション4.5.1.1「mysql のオプション」
セクション4.4.3「mysql_install_db — MySQL データディレクトリの初期化」
セクション4.4.7「mysql_upgrade — MySQL テーブルのチェックとアップグレード」
セクション4.5.2「mysqladmin — MySQL サーバーの管理を行うクライアント」
セクション4.6.8「mysqlbinlog — バイナリログファイルを処理するためのユーティリティー」
セクション4.5.3「mysqlcheck — テーブル保守プログラム」
セクション4.3.4「mysqld_multi — 複数の MySQL サーバーの管理」
セクション4.3.2「mysqld_safe — MySQL サーバー起動スクリプト」
セクション4.5.4「mysqldump — データベースバックアッププログラム」
セクション4.5.5「mysqlimport — データインポートプログラム」
セクション4.5.6「mysqlshow — データベース、テーブル、およびカラム情報の表示」
セクション4.5.7「mysqlslap — 負荷エミュレーションクライアント」
セクション4.2.6「オプションファイルの使用」
セクション4.2.7「オプションファイルの処理に影響するコマンド行オプション」
セクション5.1.3「サーバーコマンドオプション」

--defaults-file
セクション5.3「1 つのマシン上での複数の MySQL インスタンスの実行」
セクション14.3「InnoDB の構成」
セクション4.7.3「my_print_defaults — オプションファイルからのオプションの表示」
セクション4.6.3.1「myisamchk の一般オプション」
セクション4.5.1.1「mysql のオプション」
セクション2.9.4「MySQL ソース構成オプション」
セクション4.4.3「mysql_install_db — MySQL データディレクトリの初期化」
セクション4.4.7「mysql_upgrade — MySQL テーブルのチェックとアップグレード」
セクション4.5.2「mysqladmin — MySQL サーバーの管理を行うクライアント」
セクション4.6.8「mysqlbinlog — バイナリログファイルを処理するためのユーティリティー」
セクション4.5.3「mysqlcheck — テーブル保守プログラム」
セクション4.3.4「mysqld_multi — 複数の MySQL サーバーの管理」
セクション4.3.2「mysqld_safe — MySQL サーバー起動スクリプト」
セクション4.5.4「mysqldump — データベースバックアッププログラム」
セクション4.5.5「mysqlimport — データインポートプログラム」
セクション4.5.6「mysqlshow — データベース、テーブル、およびカラム情報の表示」
セクション4.5.7「mysqlslap — 負荷エミュレーションクライアント」
root のパスワードのリセット: Windows システム
セクション5.3.3「Unix 上での複数の MySQL インスタンスの実行」
セクション2.3.5.7「Windows のサービスとして MySQL を起動する」
セクション5.3.2.1「Windows コマンド行での複数の MySQL インスタンスの起動」
セクション5.3.2.2「Windows サービスとして複数の MySQL インスタンスの起動」
セクション4.2.7「オプションファイルの処理に影響するコマンド行オプション」
セクション5.1.3「サーバーコマンドオプション」
セクション6.1.2.1「パスワードセキュリティーのためのエンドユーザーガイドライン」
セクション23.6.3「組み込みサーバーとオプション」

--defaults-group-suffix
セクション4.7.3「my_print_defaults — オプションファイルからのオプションの表示」
セクション4.6.3.1「myisamchk の一般オプション」
セクション4.5.1.1「mysql のオプション」
セクション4.4.7「mysql_upgrade — MySQL テーブルのチェックとアップグレード」
セクション4.5.2「mysqladmin — MySQL サーバーの管理を行うクライアント」
セクション4.6.8「mysqlbinlog — バイナリログファイルを処理するためのユーティリティー」
セクション4.5.3「mysqlcheck — テーブル保守プログラム」
セクション4.5.4「mysqldump — データベースバックアッププログラム」
セクション4.5.5「mysqlimport — データインポートプログラム」

セクション4.5.6「mysqlshow — データベース、テーブル、およびカラム情報の表示」
セクション4.5.7「mysqlslap — 負荷エミュレーションクライアント」
セクション4.2.7「オプションファイルの処理に影響するコマンド行オプション」
セクション5.1.3「サーバーコマンドオプション」
セクション2.12「環境変数」

--delay
セクション18.4.2「ndbinfo_select_all — ndbinfo テーブルからの選択」

--delay-key-write
セクション15.2.1「MyISAM 起動オプション」
セクションA.13「MySQL 5.6 FAQ: レプリケーション」
セクションB.5.4.2「MySQL が繰り返しクラッシュする場合の対処方法」
セクション5.1.3「サーバーコマンドオプション」
セクション8.10.5「外部ロック」

--delay_key_write
セクション5.1.4「サーバーシステム変数」
セクション5.1.5「システム変数の使用」

--delayed-insert
セクション4.5.4「mysqldump — データベースバックアッププログラム」

--delete
セクション4.5.5「mysqlimport — データインポートプログラム」
セクション18.4.14「ndb_index_stat — NDB インデックス統計ユーティリティー」

--delete-master-logs
セクション4.5.4「mysqldump — データベースバックアッププログラム」

--delete-orphans
セクション18.4.6「ndb_blob_tool — MySQL Cluster テーブルの BLOB および TEXT カラムのチェックおよび修復」

--delimiter
セクション4.5.1.1「mysql のオプション」
セクション4.5.7「mysqlslap — 負荷エミュレーションクライアント」
セクション18.4.21「ndb_select_all — NDB テーブルの行の出力」

--demangle
セクション24.4.1.5「スタックトレースの使用」

--des-key-file
セクション13.7.6.3「FLUSH 構文」
セクション5.1.3「サーバーコマンドオプション」
セクション12.13「暗号化関数と圧縮関数」

--descending
セクション18.4.21「ndb_select_all — NDB テーブルの行の出力」

--description
セクション4.6.3.4「その他の myisamchk オプション」

--detach
セクション4.5.7「mysqlslap — 負荷エミュレーションクライアント」

--disable
セクション4.2.5「プログラムオプション修飾子」

--disable-auto-rehash
セクション4.5.1.1「mysql のオプション」
セクション14.13.16.2「非永続的オプティマイザ統計のパラメータの構成」

--disable-gtid-unsafe-statements
セクション17.1.3.2「GTID を使用したレプリケーションのセットアップ」
セクション17.1.4.5「グローバルトランザクション ID のオプションと変数」
セクション17.1.2.2「行ベースロギングおよびレプリケーションの使用」

--disable-indexes
セクション18.4.20「ndb_restore — MySQL Cluster バックアップのリストア」

--disable-innodb
セクション14.1.3「InnoDB の無効化」
セクション1.4「MySQL 5.6 の新機能」

--disable-keys
セクション4.5.4「mysqldump — データベースバックアッププログラム」

--disable-log-bin
セクション4.6.8「mysqlbinlog — バイナリログファイルを処理するためのユーティリティー」

--disable-named-commands
セクション4.5.1.1「mysql のオプション」

--disable-plugin_name
セクション5.1.8.1「プラグインのインストールおよびアンインストール」

--disable-ssl
セクション6.3.10.4「SSL コマンドのオプション」
セクション6.3.10.3「SSL 接続の使用」

--disconnect-slave-event-count
セクション17.1.4.3「レプリケーションスレーブのオプションと変数」

--disk
セクション18.4.21「ndb_select_all — NDB テーブルの行の出力」

--dont_ignore_systab_0
セクション18.4.20「ndb_restore — MySQL Cluster バックアップのリストア」

--dry-scp
セクション18.4.13「ndb_error_reporter — NDB エラーレポートユーティリティー」

--dryrun
セクション4.6.10「mysqlhotcopy — データベースバックアッププログラム」

--dump
セクション4.6.2「myisam_ftdump — 全文インデックス情報の表示」
セクション18.4.14「ndb_index_stat — NDB インデックス統計ユーティリティー」

--dump-date
セクション4.5.4「mysqldump — データベースバックアッププログラム」

--dump-file
セクション18.4.6「ndb_blob_tool — MySQL Cluster テーブルの BLOB および TEXT カラムのチェックおよび修復」

--dump-slave
セクション4.5.4「mysqldump — データベースバックアッププログラム」

E

[索引の先頭]

-E
セクション4.5.1.1「mysql のオプション」
セクション4.5.2「mysqladmin — MySQL サーバーの管理を行うクライアント」
セクション4.5.4「mysqldump — データベースバックアッププログラム」

-e
セクション4.6.1「innochecksum — オフライン InnoDB ファイルチェックサムユーティリティー」
セクション13.2.7「LOAD XML 構文」
セクション4.7.3「my_print_defaults — オプションファイルからのオプションの表示」
セクション7.6.2「MyISAM テーブルのエラーのチェック方法」
セクション4.6.3.5「myisamchk によるテーブル情報の取得」
セクション4.6.3.2「myisamchk のチェックオプション」
セクション4.6.3.1「myisamchk の一般オプション」
セクション4.6.3.3「myisamchk の修復オプション」
セクションA.10「MySQL 5.6 FAQ: MySQL Cluster」
セクション18.2.7「MySQL Cluster の安全なシャットダウンと再起動」
セクション18.6.9「MySQL Cluster レプリケーションを使用した MySQL Cluster バックアップ」
セクション18.5.3.2「MySQL Cluster 管理クライアントを使用したバックアップの作成」
セクション4.5.1.1「mysql のオプション」
セクション4.5.3「mysqlcheck — テーブル保守プログラム」
セクション4.5.4「mysqldump — データベースバックアッププログラム」
セクション4.5.7「mysqlslap — 負荷エミュレーションクライアント」
セクション18.4.5「ndb_mgm — MySQL Cluster 管理クライアント」
セクション18.4.20「ndb_restore — MySQL Cluster バックアップのリストア」
セクション4.2.4「コマンド行でのオプションの使用」

--embedded
セクション4.7.2「mysql_config — クライアントのコンパイル用オプションの表示」

--enable-64bit
セクション16.6.1「memcached のインストール」

--enable-cleartext-plugin
セクション4.5.1.1「mysql のオプション」
セクション4.5.2「mysqladmin — MySQL サーバーの管理を行うクライアント」
セクション4.5.7「mysqlslap — 負荷エミュレーションクライアント」
セクション6.3.8.7「クライアント側のクリアテキスト認証プラグイン」

--enable-dtrace
セクション16.6.2.5「memcached と DTrace の使用」
セクション16.6.1「memcached のインストール」

--enable-memcache
セクション16.6.3.6「PHP での MySQL と memcached の使用」

--enable-named-pipe
セクションB.5.2.2「[ローカルの] MySQL サーバーに接続できません」
セクション1.3.2「MySQL の主な機能」
セクション4.2.2「MySQL サーバーへの接続」
セクション2.3.5.3「MySQL サーバータイプの選択」
セクション5.1.3「サーバーコマンドオプション」

--enable-plugin_name
セクション5.1.8.1「プラグインのインストールおよびアンインストール」

--enable-threads
セクション16.6.1「memcached のインストール」

enabled
セクション2.5.1「MySQL Yum リポジトリを使用して MySQL を Linux にインストールする」

--enforce-gtid-consistency
セクション17.1.3.2「GTID を使用したレプリケーションのセットアップ」
セクション17.1.3.4「GTID ベースレプリケーションの制約」
セクション1.4「MySQL 5.6 の新機能」
セクション17.1.4.5「グローバルトランザクション ID のオプションと変数」

--engine
セクション4.5.7「mysqlslap — 負荷エミュレーションクライアント」

--engine-condition-pushdown
セクション18.3.2.2「MySQL Cluster の推奨される初期構成」

--event-scheduler
セクション20.4.2「イベントスケジューラの構成」
セクション5.1.3「サーバーコマンドオプション」

event-scheduler
セクション20.4.2「イベントスケジューラの構成」

--events
セクション2.11.1.3「MySQL 5.5 から 5.6 へのアップグレード」
セクション4.5.4「mysqldump — データベースバックアッププログラム」
セクション7.4.5.3「ストアドプログラムのダンプ」
セクション7.4.5.4「テーブル定義と内容の個別のダンプ」
セクション4.6.8.3「バイナリログファイルのバックアップのための mysqlbinlog の使用」

--example
セクション4.3.4「mysqld_multi — 複数の MySQL サーバーの管理」

--exclude-*
セクション18.4.20「ndb_restore — MySQL Cluster バックアップのリストア」

--exclude-databases
セクション18.4.20「ndb_restore — MySQL Cluster バックアップのリストア」

--exclude-gtids
セクション4.6.8「mysqlbinlog — バイナリログファイルを処理するためのユーティリティー」

--exclude-intermediate-sql-tables
セクション18.4.20「ndb_restore — MySQL Cluster バックアップのリストア」

--exclude-missing-columns
セクション18.4.20「ndb_restore — MySQL Cluster バックアップのリストア」

--exclude-missing-tables
セクション18.4.20「ndb_restore — MySQL Cluster バックアップのリストア」

--exclude-tables
セクション18.4.20「ndb_restore — MySQL Cluster バックアップのリストア」

--excludedbs
セクション18.4.25「ndb_size.pl — NDBCLUSTER サイズ要件エスティメータ」

--excludetables
セクション18.4.25「ndb_size.pl — NDBCLUSTER サイズ要件エスティメータ」

--execute
セクション18.5.3.2「MySQL Cluster 管理クライアントを使用したバックアップの作成」
セクション4.5.1.1「mysql のオプション」
セクション4.5.1.3「mysql のロギング」
セクション18.4.5「ndb_mgm — MySQL Cluster 管理クライアント」
セクション4.2.4「コマンド行でのオプションの使用」

--exit-info
セクション5.1.3「サーバーコマンドオプション」

--extend-check
セクション4.6.3.2「myisamchk のチェックオプション」
セクション4.6.3.1「myisamchk の一般オプション」
セクション4.6.3.3「myisamchk の修復オプション」

--extended
セクション4.5.3「mysqlcheck — テーブル保守プログラム」

--extended-insert
セクション4.5.4「mysqldump — データベースバックアッププログラム」

--external-locking
セクション15.2.1「MyISAM 起動オプション」
セクション5.1.3「サーバーコマンドオプション」
セクション5.1.4「サーバーシステム変数」
セクション8.11.1「システム要素およびスタートアップパラメータのチューニング」
セクション8.10.5「外部ロック」

--extra-file
セクション4.7.3「my_print_defaults — オプションファイルからのオプションの表示」

--extra-node-info
セクション18.4.10「ndb_desc — NDB テーブルの表示」

--extra-partition-info
セクション18.4.10「ndb_desc — NDB テーブルの表示」
セクション18.5.10.4「ndbinfo cluster_operations テーブル」
セクション18.5.10.20「ndbinfo server_operations テーブル」

F

[索引の先頭]

-F
セクション4.4.1「comp_err — MySQL エラーメッセージファイルのコンパイル」
セクション4.6.3.2「myisamchk のチェックオプション」
セクション4.5.1.2「mysql コマンド」
セクション4.6.8「mysqlbinlog — バイナリログファイルを処理するためのユーティリティー」

セクション4.5.3「mysqlcheck — テーブル保守プログラム」
セクション4.5.4「mysqldump — データベースバックアッププログラム」
セクション4.5.7「mysqlslap — 負荷エミュレーションクライアント」

-f

セクション16.6.2「memcached の使用」
セクション4.6.3.2「myisamchk のチェックオプション」
セクション4.6.3.3「myisamchk の修復オプション」
セクション4.6.4「myisamlog — MyISAM ログファイルの内容の表示」
セクション4.6.5「myisampack — 圧縮された読み取り専用の MyISAM テーブルの生成」
セクションA.10「MySQL 5.6 FAQ: MySQL Cluster」
セクション18.2.5「MySQL Cluster の初期起動」
セクション18.5.2「MySQL Cluster 管理クライアントのコマンド」
セクション4.5.1.1「mysql のオプション」
セクション4.6.16「mysql_zap — パターンに一致するプロセスを強制終了」
セクション4.5.2「mysqladmin — MySQL サーバーの管理を行うクライアント」
セクション4.6.8「mysqlbinlog — バイナリログファイルを処理するためのユーティリティー」
セクション4.5.3「mysqlcheck — テーブル保守プログラム」
セクション4.5.4「mysqldump — データベースバックアッププログラム」
セクション4.5.5「mysqlimport — データインポートプログラム」
セクション18.4.7「ndb_config — MySQL Cluster 構成情報の抽出」
セクション18.4.4「ndb_mgmd — MySQL Cluster 管理サーバーデーモン」
セクション18.2.3.3「Windows での MySQL Cluster の初期起動」
セクション24.4.1.5「スタックトレースの使用」

--fast

セクション4.6.3.2「myisamchk のチェックオプション」
セクション4.5.3「mysqlcheck — テーブル保守プログラム」

--federated

セクション15.8「FEDERATED ストレージエンジン」

--fields

セクション18.4.7「ndb_config — MySQL Cluster 構成情報の抽出」

--fields-enclosed-by

セクション4.5.4「mysqldump — データベースバックアッププログラム」
セクション7.4.3「mysqldump による区切りテキストフォーマットでのデータのダンプ」
セクション4.5.5「mysqlimport — データインポートプログラム」
セクション18.4.20「ndb_restore — MySQL Cluster バックアップのリストア」

--fields-escaped-by

セクション4.5.4「mysqldump — データベースバックアッププログラム」
セクション7.4.3「mysqldump による区切りテキストフォーマットでのデータのダンプ」
セクション4.5.5「mysqlimport — データインポートプログラム」

--fields-optionally-enclosed-by

セクション4.5.4「mysqldump — データベースバックアッププログラム」
セクション7.4.3「mysqldump による区切りテキストフォーマットでのデータのダンプ」
セクション4.5.5「mysqlimport — データインポートプログラム」
セクション18.4.20「ndb_restore — MySQL Cluster バックアップのリストア」

--fields-terminated-by

セクション4.5.4「mysqldump — データベースバックアッププログラム」
セクション7.4.3「mysqldump による区切りテキストフォーマットでのデータのダンプ」
セクション4.5.5「mysqlimport — データインポートプログラム」
セクション18.4.20「ndb_restore — MySQL Cluster バックアップのリストア」

--fields-xxx

セクション4.5.4「mysqldump — データベースバックアッププログラム」

--fix-db-names

セクション4.5.3「mysqlcheck — テーブル保守プログラム」

--fix-table-names

セクション4.5.3「mysqlcheck — テーブル保守プログラム」

--flush

セクションB.5.4.2「MySQL が繰り返しクラッシュする場合の対処方法」
セクション5.1.3「サーバーコマンドオプション」
セクション5.1.4「サーバーシステム変数」

--flush-logs

セクション5.2「MySQL Server ログ」
セクション4.5.4「mysqldump — データベースバックアッププログラム」
セクション7.3.1「バックアップポリシーの確立」

--flush-privileges

セクション4.5.4「mysqldump — データベースバックアッププログラム」

--flush_time

セクション24.1.1「MySQL のスレッド」

--flushlog

セクション4.6.10「mysqlhotcopy — データベースバックアッププログラム」

--force

セクション4.6.3.2「myisamchk のチェックオプション」

セクション4.6.3.3「myisamchk の修復オプション」
セクション4.6.5「myisampack — 圧縮された読み取り専用の MyISAM テーブルの生成」
セクション4.5.1.1「mysql のオプション」
セクション24.1.2「MySQL テストスイート」
セクション4.6.11「mysql_convert_table_format — 指定されたストレージエンジンを使用するためのテーブルの変換」
セクション4.4.3「mysql_install_db — MySQL データディレクトリの初期化」
セクション4.4.7「mysql_upgrade — MySQL テーブルのチェックとアップグレード」
セクション4.5.2「mysqladmin — MySQL サーバーの管理を行うクライアント」
セクション4.5.3「mysqlcheck — テーブル保守プログラム」
セクション4.5.4「mysqldump — データベースバックアッププログラム」
セクション4.5.5「mysqlimport — データインポートプログラム」
セクション3.5「バッチモードでの MySQL の使用」

--force-if-open

セクション4.6.8「mysqlbinlog — バイナリログファイルを処理するためのユーティリティー」

--force-read

セクション4.6.8「mysqlbinlog — バイナリログファイルを処理するためのユーティリティー」

--foreground

セクション18.4.1「ndbd — MySQL Cluster データノードデーモン」

--format

セクション18.4.25「ndb_size.pl — NDBCLUSTER サイズ要件エスティメータ」

--fs

セクション18.4.13「ndb_error_reporter — NDB エラーレポートユーティリティー」

G

[索引の先頭]

-G

セクション4.5.1.1「mysql のオプション」
セクション4.6.6「mysql_config_editor — MySQL 構成ユーティリティー」

-g

セクション4.7.3「my_print_defaults — オプションファイルからのオプションの表示」
セクション4.5.3「mysqlcheck — テーブル保守プログラム」
セクション4.6.9「mysqldumpslow — スロークエリーログファイルの要約」
セクション24.4.1.1「デバッグのための MySQL のコンパイル」

--gci

セクション18.4.21「ndb_select_all — NDB テーブルの行の出力」

--gci64

セクション18.4.21「ndb_select_all — NDB テーブルの行の出力」

--gdb

セクション24.4.1.4「gdb での mysqld のデバッグ」
セクション5.1.3「サーバーコマンドオプション」

--general-log

セクション5.1.3「サーバーコマンドオプション」

--general_log

セクション1.4「MySQL 5.6 の新機能」
セクション5.1.3「サーバーコマンドオプション」
セクション5.1.4「サーバーシステム変数」
セクション5.2.3「一般クエリーログ」
セクション5.2.1「一般クエリーログおよびスロークエリーログの出力先の選択」

--general_log_file

セクション5.3「1 つのマシン上での複数の MySQL インスタンスの実行」
セクション1.4「MySQL 5.6 の新機能」
セクション5.1.3「サーバーコマンドオプション」
セクション5.1.4「サーバーシステム変数」
セクション5.2.3「一般クエリーログ」

--gtid-mode

セクション17.1.3.2「GTID を使用したレプリケーションのセットアップ」
セクション17.1.3.4「GTID ベースレプリケーションの制約」
セクション1.4「MySQL 5.6 の新機能」
セクション2.11.1「MySQL のアップグレード」
セクション17.1.4.5「グローバルトランザクション ID のオプションと変数」
セクション17.1.3.3「フェイルオーバーおよびスケールアウトでの GTID の使用」
セクション17.1.4.3「レプリケーションスレーブのオプションと変数」

--gtid_mode

セクション17.1.3.4「GTID ベースレプリケーションの制約」

H

[索引の先頭]

-H

セクション4.4.1「comp_err — MySQL エラーメッセージファイルのコンパイル」
セクション4.6.3.1「myisamchk の一般オプション」
セクション4.5.1.1「mysql のオプション」
セクション4.6.7「mysqlaccess — アクセス権限をチェックするクライアント」
セクション4.6.8「mysqlbinlog — バイナリログファイルを処理するためのユーティリティー」

-h

セクション16.6.2「memcached の使用」
セクション4.6.2「myisam_ftdump — 全文インデックス情報の表示」

セクション4.5.1.1「mysql のオプション」
セクション4.2.2「MySQL サーバーへの接続」
セクション4.2.1「MySQL プログラムの起動」
セクション4.6.6「mysql_config_editor — MySQL 構成ユーティリティー」
セクション4.4.7「mysql_upgrade — MySQL テーブルのチェックとアップグレード」
セクション4.6.7「mysqlaccess — アクセス権限をチェックするクライアント」
セクション4.5.2「mysqladmin — MySQL サーバーの管理を行うクライアント」
セクション4.6.8「mysqlbinlog — バイナリログファイルを処理するためのユーティリティー」
セクション4.5.3「mysqlcheck — テーブル保守プログラム」
セクション4.5.4「mysqldump — データベースバックアッププログラム」
セクション4.6.9「mysqldumpslow — スロークエリーログファイルの要約」
セクション4.6.10「mysqlhotcopy — データベースバックアッププログラム」
セクション4.5.5「mysqlimport — データインポートプログラム」
セクション4.5.6「mysqlshow — データベース、テーブル、およびカラム情報の表示」
セクション4.5.7「mysqlslap — 負荷エミュレーションクライアント」
セクション18.4.16「ndb_print_file — NDB ディスクデータファイル内容の出力」
セクション18.4.23「ndb_setup.py — MySQL Cluster のブラウザベース自動インストーラの開始」
セクション4.7.4「resolve_stack_dump — 数値スタックトレースダンプをシンボルに解決」
セクション4.2.4「コマンド行でのオプションの使用」
セクション5.1.3「サーバーコマンドオプション」
セクション1.2「表記規則および構文規則」

--header

セクション18.4.21「ndb_select_all — NDB テーブルの行の出力」

--header_file

セクション4.4.1「comp_err — MySQL エラーメッセージファイルのコンパイル」

--HELP

セクション4.6.3.1「myisamchk の一般オプション」

--help

セクション4.4.1「comp_err — MySQL エラーメッセージファイルのコンパイル」
セクション4.7.3「my_print_defaults — オプションファイルからのオプションの表示」
セクション4.6.2「myisam_ftdump — 全文インデックス情報の表示」
セクション4.6.3.1「myisamchk の一般オプション」
セクション4.6.5「myisampack — 圧縮された読み取り専用の MyISAM テーブルの生成」
セクション18.4.27「MySQL Cluster プログラムに共通するオプション — MySQL Cluster プログラムに共通するオプション」
セクション4.5.1.1「mysql のオプション」
セクション1.3.2「MySQL の主な機能」
セクション2.10.1.3「MySQL サーバーの起動とトラブルシューティング」
セクション4.1「MySQL プログラムの概要」
セクション4.6.6「mysql_config_editor — MySQL 構成ユーティリティー」
セクション4.6.11「mysql_convert_table_format — 指定されたストレージエンジンを使用するためのテーブルの変換」
セクション4.6.12「mysql_find_rows — ファイルからの SQL ステートメントの抽出」
セクション4.4.3「mysql_install_db — MySQL データディレクトリの初期化」
セクション4.4.4「mysql_plugin — MySQL サーバープラグインの構成」
セクション4.6.14「mysql_setpermission — 付与テーブルに許可をインタラクティブに設定」
セクション4.4.7「mysql_upgrade — MySQL テーブルのチェックとアップグレード」
セクション4.6.15「mysql_waitpid — プロセスを強制終了して終了を待機」
セクション4.6.16「mysql_zap — パターンに一致するプロセスを強制終了」
セクション4.6.7「mysqlaccess — アクセス権限をチェックするクライアント」
セクション4.5.2「mysqladmin — MySQL サーバーの管理を行うクライアント」
セクション4.6.8「mysqlbinlog — バイナリログファイルを処理するためのユーティリティー」
セクション4.5.3「mysqlcheck — テーブル保守プログラム」
セクション4.3.4「mysqld_multi — 複数の MySQL サーバーの管理」
セクション4.3.2「mysqld_safe — MySQL サーバー起動スクリプト」
セクション4.5.4「mysqldump — データベースバックアッププログラム」
セクション4.6.9「mysqldumpslow — スロークエリーログファイルの要約」
セクション4.6.10「mysqlhotcopy — データベースバックアッププログラム」
セクション4.5.5「mysqlimport — データインポートプログラム」
セクション4.5.6「mysqlshow — データベース、テーブル、およびカラム情報の表示」
セクション4.5.7「mysqlslap — 負荷エミュレーションクライアント」
セクション18.4.6「ndb_blob_tool — MySQL Cluster テーブルの BLOB および TEXT カラムのチェックおよび修復」
セクション18.4.7「ndb_config — MySQL Cluster 構成情報の抽出」
セクション18.4.13「ndb_error_reporter — NDB エラーレポートユーティリティー」
セクション18.4.16「ndb_print_file — NDB ディスクデータファイル内容の出力」
セクション18.4.23「ndb_setup.py — MySQL Cluster のブラウザベース自動インストーラの開始」
セクション18.4.19「ndbd_redo_log_reader — クラスタ Redo ログ内容のチェックおよび出力」
セクション4.8.1「perror — エラーコードの説明」
セクション4.7.4「resolve_stack_dump — 数値スタックトレースダンプをシンボルに解決」
セクション4.8.3「resolveip — ホスト名と IP アドレスの解決」
セクション2.10.1「Unix 類似システムでのインストール後の手順」

セクション18.2.3.4「Windows サービスとしての MySQL Cluster プロセスのインストール」
セクション4.2.6「オプションファイルの使用」
セクション4.2.4「コマンド行でのオプションの使用」
セクション5.1.3「サーバーコマンドオプション」
セクション8.11.2「サーバーパラメータのチューニング」
第3章「チュートリアル」

--hex

セクション18.4.20「ndb_restore — MySQL Cluster バックアップのリストア」

--hex-blob

セクション4.5.4「mysqldump — データベースバックアッププログラム」

--hexdump

セクション4.6.8.1「mysqlbinlog 16 進ダンプ形式」
セクション4.6.8「mysqlbinlog — バイナリログファイルを処理するためのユーティリティー」

--histignore

セクション4.5.1.1「mysql のオプション」
セクション4.5.1.3「mysql の口ギング」

--host

セクション5.1.9.2「IPv6 接続を許可するための MySQL Server の構成」
セクション4.5.1.1「mysql のオプション」
セクション4.2.2「MySQL サーバーへの接続」
セクション4.2.1「MySQL プログラムの起動」
セクション4.6.6「mysql_config_editor — MySQL 構成ユーティリティー」
セクション4.6.11「mysql_convert_table_format — 指定されたストレージエンジンを使用するためのテーブルの変換」
セクション4.6.14「mysql_setpermission — 付与テーブルに許可をインタラクティブに設定」
セクション4.4.7「mysql_upgrade — MySQL テーブルのチェックとアップグレード」
セクション4.6.7「mysqlaccess — アクセス権限をチェックするクライアント」
セクション4.5.2「mysqladmin — MySQL サーバーの管理を行うクライアント」
セクション4.6.8「mysqlbinlog — バイナリログファイルを処理するためのユーティリティー」
セクション4.5.3「mysqlcheck — テーブル保守プログラム」
セクション4.5.4「mysqldump — データベースバックアッププログラム」
セクション4.6.10「mysqlhotcopy — データベースバックアッププログラム」
セクション4.5.5「mysqlimport — データインポートプログラム」
セクション4.5.6「mysqlshow — データベース、テーブル、およびカラム情報の表示」
セクション4.5.7「mysqlslap — 負荷エミュレーションクライアント」
セクション18.4.7「ndb_config — MySQL Cluster 構成情報の抽出」
セクション6.2.7「アクセス拒否エラーの原因」
セクション4.2.9「オプションのデフォルト、値を想定するオプション、および = 記号」
セクション4.2.6「オプションファイルの使用」
セクション4.2.4「コマンド行でのオプションの使用」
セクション5.1.3「サーバーコマンドオプション」
セクション4.6.8.3「バイナリログファイルのバックアップのための mysqlbinlog の使用」
セクション1.2「表記規則および構文規則」
セクション5.3.4「複数サーバー環境でのクライアントプログラムの使用」

host

セクション4.7.3「my_print_defaults — オプションファイルからのオプションの表示」
セクション4.5.1.1「mysql のオプション」
セクション4.6.6「mysql_config_editor — MySQL 構成ユーティリティー」
セクション4.4.7「mysql_upgrade — MySQL テーブルのチェックとアップグレード」
セクション4.5.2「mysqladmin — MySQL サーバーの管理を行うクライアント」
セクション4.6.8「mysqlbinlog — バイナリログファイルを処理するためのユーティリティー」
セクション4.5.3「mysqlcheck — テーブル保守プログラム」
セクション4.5.4「mysqldump — データベースバックアッププログラム」
セクション4.5.5「mysqlimport — データインポートプログラム」
セクション4.5.6「mysqlshow — データベース、テーブル、およびカラム情報の表示」
セクション4.5.7「mysqlslap — 負荷エミュレーションクライアント」
セクション4.2.6「オプションファイルの使用」
セクション4.2.7「オプションファイルの処理に影響するコマンド行オプション」

--hostname

セクション18.4.25「ndb_size.pl — NDBCLUSTER サイズ要件エスティメータ」

--howto

セクション4.6.7「mysqlaccess — アクセス権限をチェックするクライアント」

--html

セクション4.5.1.1「mysql のオプション」

I

[索引の先頭]

-I

セクション23.7.4.1「C API クライアントプログラムの構築」
セクション16.6.5「memcached の FAQ」
セクション16.6.2「memcached の使用」
セクション4.6.4「myisamlog — MyISAM ログファイルの内容の表示」
セクション4.6.15「mysql_waitpid — プロセスを強制終了して終了を待機」
セクション4.6.16「mysql_zap — パターンに一致するプロセスを強制終了」
セクション4.8.1「perror — エラーコードの説明」
セクション4.8.2「replace — 文字列置換ユーティリティー」
セクション4.8.3「resolveip — ホスト名と IP アドレスの解決」

-i
セクション16.6.2「memcached の使用」
セクション7.6.2「MyISAM テーブルのエラーのチェック方法」
セクション4.6.3.2「myisamchk のチェックオプション」
セクション4.6.4「myisamlog — MyISAM ログファイルの内容の表示」
セクション18.5.2「MySQL Cluster 管理クライアントのコマンド」
セクション4.5.1.1「mysql のオプション」
セクション4.4.4「mysql_plugin — MySQL サーバープラグインの構成」
セクション4.5.2「mysqladmin — MySQL サーバーの管理を行うクライアント」
セクション4.5.4「mysqldump — データベースバックアッププログラム」
セクション4.6.9「mysqldumpslow — スロークエリーログファイルの要約」
セクション4.5.5「mysqlimport — データインポートプログラム」
セクション4.5.6「mysqlshow — データベース、テーブル、およびカラム情報の表示」
セクション4.5.7「mysqlslap — 負荷エミュレーションクライアント」
セクション24.4.1.3「pdb を使用した Windows のクラッシュダンプの作成」
セクション2.5.5「RPM パッケージを使用して MySQL を Linux にインストールする」

--i-am-a-dummy
--safe-updates オプションの使用
セクション4.5.1.1「mysql のオプション」

--id
セクション18.4.7「ndb_config — MySQL Cluster 構成情報の抽出」

--ignore
セクション4.5.5「mysqlimport — データインポートプログラム」

--ignore-builtin-innodb
セクション14.12「InnoDB の起動オプションおよびシステム変数」

--ignore-db-dir
セクション5.1.3「サーバーコマンドオプション」
セクション5.1.4「サーバーシステム変数」

--ignore-lines
セクション4.5.5「mysqlimport — データインポートプログラム」

--ignore-spaces
セクション4.5.1.1「mysql のオプション」

--ignore-table
セクション4.5.4「mysqldump — データベースバックアッププログラム」

--in_file
セクション4.4.1「comp_err — MySQL エラーメッセージファイルのコンパイル」

--include
セクション4.7.2「mysql_config — クライアントのコンパイル用オプションの表示」

--include-*
セクション18.4.20「ndb_restore — MySQL Cluster バックアップのリストア」

--include-databases
セクション18.4.20「ndb_restore — MySQL Cluster バックアップのリストア」

--include-gtids
セクション4.6.8「mysqlbinlog — バイナリログファイルを処理するためのユーティリティー」

--include-master-host-port
セクション4.5.4「mysqldump — データベースバックアッププログラム」

--include-tables
セクション18.4.20「ndb_restore — MySQL Cluster バックアップのリストア」

--info
セクション4.8.1「perror — エラーコードの説明」
セクション4.8.3「resolveip — ホスト名と IP アドレスの解決」

--Information
セクション4.6.12「mysql_find_rows — ファイルからの SQL ステートメントの抽出」

--information
セクション4.6.3.2「myisamchk のチェックオプション」

--init-command
セクション4.5.1.1「mysql のオプション」
セクション17.4.1.27「サーバー側ヘルプテーブルのレプリケーション」

--init-file
セクション15.3「MEMORY ストレージエンジン」
root のパスワードのリセット: UNIX システム
root のパスワードのリセット: Windows システム
セクション5.1.3「サーバーコマンドオプション」
セクション5.1.4「サーバーシステム変数」
セクション22.2.3「パフォーマンススキーマ実行時構成」

--init-rpl-role
セクション1.4「MySQL 5.6 の新機能」

--init_connect
セクション10.1.5「アプリケーションの文字セットおよび照合順序の構成」

--initial
セクション18.3.3.3「MySQL Cluster SQL ノードおよび API ノードの構成パラメータ」
セクション18.5.5「MySQL Cluster のローリング再起動の実行」
セクション18.3.2「MySQL Cluster の構成ファイル」
セクション18.5.1「MySQL Cluster の起動フェーズのサマリー」
セクション18.5.14「MySQL Cluster の配布された MySQL 権限」
セクション18.5.12.3「MySQL Cluster ディスクデータストレージの要件」
セクション18.5.13.2「MySQL Cluster データノードのオンライン追加: 基本的な手順」
セクション18.5.13.3「MySQL Cluster データノードのオンライン追加: 詳細な例」
セクション18.3.2.6「MySQL Cluster データノードの定義」
セクション18.3.3.1「MySQL Cluster データノードの構成パラメータ」
セクション18.6.3「MySQL Cluster レプリケーションの既知の問題」
セクション18.6.9.2「MySQL Cluster レプリケーションを使用したポイントインタイムリカバリ」
セクション18.3.3「MySQL Cluster 構成パラメータの概要」
セクション18.5.2「MySQL Cluster 管理クライアントのコマンド」
セクション18.3.3.2「MySQL Cluster 管理ノードの構成パラメータ」
セクション18.4.7「ndb_config — MySQL Cluster 構成情報の抽出」
セクション18.4.4「ndb_mgmd — MySQL Cluster 管理サーバーデーモン」
セクション18.4.20「ndb_restore — MySQL Cluster バックアップのリストア」
セクション18.4.1「ndbd — MySQL Cluster データノードデーモン」
セクション18.4.3「ndbmtd — MySQL Cluster データノードデーモン (マルチスレッド)」
セクション18.2.3.3「Windows での MySQL Cluster の初期起動」
セクション18.3.3.4「その他の MySQL Cluster 構成パラメータ」
セクション18.1.6.10「複数の MySQL Cluster ノードに関する制限」

--initial-start
セクション18.4.1「ndbd — MySQL Cluster データノードデーモン」

--innodb
セクション14.1.3「InnoDB の無効化」
セクション14.12「InnoDB の起動オプションおよびシステム変数」
セクション1.4「MySQL 5.6 の新機能」

--innodb-status-file
セクション14.12「InnoDB の起動オプションおよびシステム変数」

innodb-status-file
セクション14.15.2「InnoDB モニターの有効化」

--innodb-xxx
セクション5.1.3「サーバーコマンドオプション」

--innodb_adaptive_hash_index
セクション14.12「InnoDB の起動オプションおよびシステム変数」

--innodb_file_per_table
セクション14.5.3「File-Per-Table モードの有効化および無効化」
Windows 上のデータベースへのシンボリックリンクの使用
セクション5.1.3「サーバーコマンドオプション」

innodb_file_per_table
セクション5.1.3「サーバーコマンドオプション」
セクション17.1.1.6「ローデータファイルを使用したデータスナップショットの作成」

--innodb_rollback_on_timeout
セクション14.19.4「InnoDB のエラー処理」
セクション14.12「InnoDB の起動オプションおよびシステム変数」

--innodb_support_xa
セクション5.2.4「バイナリログ」

--insert-ignore
セクション4.5.4「mysqldump — データベースバックアッププログラム」

--install
セクション18.4.4「ndb_mgmd — MySQL Cluster 管理サーバーデーモン」
セクション18.4.1「ndbd — MySQL Cluster データノードデーモン」
セクション2.3.5.7「Windows のサービスとして MySQL を起動する」
セクション18.2.3.4「Windows サービスとしての MySQL Cluster プロセスのインストール」
セクション5.3.2.2「Windows サービスとして複数の MySQL インスタンスの起動」
セクション4.2.7「オプションファイルの処理に影響するコマンド行オプション」
セクション5.1.3「サーバーコマンドオプション」

--install-manual
セクション2.3.5.7「Windows のサービスとして MySQL を起動する」
セクション5.3.2.2「Windows サービスとして複数の MySQL インスタンスの起動」
セクション5.1.3「サーバーコマンドオプション」

--interactive
セクション18.4.4「ndb_mgmd — MySQL Cluster 管理サーバーデーモン」

--iterations
セクション4.5.7「mysqlslap — 負荷エミュレーションクライアント」

J

[索引の先頭]

-j

セクション4.6.5「myisampack — 圧縮された読み取り専用のMyISAM テーブルの生成」
セクション4.6.8「mysqlbinlog — バイナリログファイルを処理するためのユーティリティー」

--join

セクション4.6.5「myisampack — 圧縮された読み取り専用のMyISAM テーブルの生成」

K

[索引の先頭]

-K

セクション4.5.4「mysqldump — データベースバックアッププログラム」

-k

セクション16.6.2「memcached の使用」
セクション4.6.3.3「myisamchk の修復オプション」
セクション4.4.7「mysql_upgrade — MySQL テーブルのチェックとアップグレード」
セクション4.5.6「mysqlshow — データベース、テーブル、およびカラム情報の表示」

--keep-my-cnf

セクション4.4.3「mysql_install_db — MySQL データディレクトリの初期化」

--keep_files_on_create

セクション13.1.17「CREATE TABLE 構文」

--keepold

セクション4.6.10「mysqlhotcopy — データベースバックアッププログラム」

--key-file

セクション18.4.23「ndb_setup.py — MySQL Cluster のブラウザベース自動インストーラの開始」

--key_buffer_size

セクション5.1.3「サーバーコマンドオプション」

--keys

セクション4.5.6「mysqlshow — データベース、テーブル、およびカラム情報の表示」

--keys-used

セクション4.6.3.3「myisamchk の修復オプション」

L

[索引の先頭]

-L

セクション23.7.4.1「C API クライアントプログラムの構築」
セクション6.1.6「LOAD DATA LOCAL のセキュリティーの問題」
セクション16.6.2「memcached の使用」
セクション4.5.1.1「mysql のオプション」
セクション4.5.5「mysqlimport — データインポートプログラム」
セクション18.4.20「ndb_restore — MySQL Cluster バックアップのリストア」
セクション2.13.3「Perl DBI/DBD インタフェース使用の問題」

-l

セクション23.7.4.1「C API クライアントプログラムの構築」
セクション23.7.13「C API 組み込みサーバー関数の説明」
セクション23.7.6「C API 関数の概要」
セクション16.6.2「memcached の使用」
セクション4.7.3「my_print_defaults — オプションファイルからのオプションの表示」
セクション4.6.2「myisam_ftdump — 全文インデックス情報の表示」
セクション4.6.3.3「myisamchk の修復オプション」
セクション23.7.7.39「mysql_library_end()」
セクション4.6.8「mysqlbinlog — バイナリログファイルを処理するためのユーティリティー」
セクション4.5.4「mysqldump — データベースバックアッププログラム」
セクション4.6.9「mysqldumpslow — スロークエリーログファイルの要約」
セクション4.5.5「mysqlimport — データインポートプログラム」
セクション18.4.21「ndb_select_all — NDB テーブルの行の出力」
セクション18.4.24「ndb_show_tables — NDB テーブルのリストの表示」
セクション18.4.2「ndbinfo_select_all — ndbinfo テーブルからの選択」
セクション5.1.3「サーバーコマンドオプション」

--language

セクション1.4「MySQL 5.6 の新機能」
セクション5.1.3「サーバーコマンドオプション」

--large-pages

セクション5.1.3「サーバーコマンドオプション」
セクション5.1.4「サーバーシステム変数」
セクション8.11.4.2「ラージページのサポートの有効化」

--lc-messages

セクション1.4「MySQL 5.6 の新機能」
セクション5.1.3「サーバーコマンドオプション」

--lc-messages-dir

セクション1.4「MySQL 5.6 の新機能」
セクション5.1.3「サーバーコマンドオプション」

--ldata

セクション4.4.3「mysql_install_db — MySQL データディレクトリの初期化」

--ledir
セクション4.3.2「mysqld_safe — MySQL サーバー起動スクリプト」

--length
セクション4.6.2「myisam_ftdump — 全文インデックス情報の表示」

--libmysqld-libs
セクション4.7.2「mysql_config — クライアントのコンパイル用オプションの表示」

--libs
セクション4.7.2「mysql_config — クライアントのコンパイル用オプションの表示」

--libs_r
セクション4.7.2「mysql_config — クライアントのコンパイル用オプションの表示」

--line-numbers
セクション4.5.1.1「mysql のオプション」

--lines-terminated-by
セクション4.5.4「mysqldump — データベースバックアッププログラム」
セクション7.4.3「mysqldump による区切りテキストフォーマットでのデータのダンプ」
セクション4.5.5「mysqlimport — データインポートプログラム」

--loadqueries
セクション18.4.25「ndb_size.pl — NDBCLUSTER サイズ要件エスティメータ」

--local
セクション13.2.6「LOAD DATA INFILE 構文」
セクション6.1.6「LOAD DATA LOCAL のセキュリティーの問題」
セクション4.5.5「mysqlimport — データインポートプログラム」

--local-infile
セクション6.1.6「LOAD DATA LOCAL のセキュリティーの問題」
セクション13.2.7「LOAD XML 構文」
セクション4.5.1.1「mysql のオプション」

--local-load
セクション4.6.8「mysqlbinlog — バイナリログファイルを処理するためのユーティリティー」

--local-service
セクション2.3.5.7「Windows のサービスとして MySQL を起動する」
セクション5.1.3「サーバーコマンドオプション」

--lock
セクション18.4.21「ndb_select_all — NDB テーブルの行の出力」

--lock-all-tables
セクション4.5.4「mysqldump — データベースバックアッププログラム」

--lock-tables
セクション4.5.4「mysqldump — データベースバックアッププログラム」
セクション4.5.5「mysqlimport — データインポートプログラム」

--log
セクション1.4「MySQL 5.6 の新機能」
セクション4.3.4「mysqld_multi — 複数の MySQL サーバーの管理」
セクション5.1.3「サーバーコマンドオプション」

--log-bin
セクション5.3「1 つのマシン上での複数の MySQL インスタンスの実行」
セクション13.4.2.1「CHANGE MASTER TO 構文」
セクション1.4「MySQL 5.6 の新機能」
セクション18.1.6.1「MySQL Cluster での SQL 構文の不適合」
セクションB.5.8「MySQL の既知の問題」
セクション13.4.1.1「PURGE BINARY LOGS 構文」
セクション17.1.4.5「グローバルトランザクション ID のオプションと変数」
セクション20.7「ストアドプログラムのバイナリロギング」
セクション7.2「データベースバックアップ方法」
セクション5.2.4「バイナリログ」
セクション17.1.4.4「バイナリログのオプションと変数」
セクション7.5「バイナリログを使用したポイントインタイム (増分) リカバリ」
セクション7.3.1「バックアップポリシーの確立」
セクション7.3.3「バックアップ戦略サマリー」
セクション17.3.6「フェイルオーバー中にマスターを切り替える」
セクション7.3.2「リカバリへのバックアップの使用」
セクション17.4.4「レプリケーションのトラブルシューティング」
セクション17.1.4.3「レプリケーションスレーブのオプションと変数」
セクション17.4.3「レプリケーションセットアップをアップグレードする」
セクション17.4.5「レプリケーションバグまたは問題を報告する方法」

--log-bin-index
セクション18.1.6.1「MySQL Cluster での SQL 構文の不適合」
セクション5.2.4「バイナリログ」
セクション17.1.4.4「バイナリログのオプションと変数」

--log-bin-trust-function-creators
セクション20.7「ストアドプログラムのバイナリロギング」
セクション17.1.4.4「バイナリログのオプションと変数」

--log-bin-use-v1-events
セクション17.1.4.4「バイナリログのオプションと変数」

--log-bin-use-v1-row-events
セクション2.11.2.1「MySQL 5.5 へのダウングレード」

セクション18.6.11「MySQL Cluster レプリケーションの競合解決」
セクション18.3.4.2「MySQL Cluster 用の MySQL Server オプション」
セクション17.1.4.4「バイナリログのオプションと変数」

--log-error

セクション5.3「1 つのマシン上での複数の MySQL インスタンスの実行」
セクション4.3.2「mysqld_safe — MySQL サーバー起動スクリプト」
セクション4.5.4「mysqldump — データベースバックアッププログラム」
セクション2.3.5.5「Windows のコマンド行からの MySQL の起動」
セクション5.2.2「エラーログ」
セクション4.2.9「オプションのデフォルト、値を想定するオプション、および = 記号」
セクション2.3.5.4「サーバーをはじめて起動する」
セクション5.1.3「サーバーコマンドオプション」
セクション5.2.7「サーバーログの保守」

--log-isam

セクション4.6.4「myisamlog — MyISAM ログファイルの内容の表示」
セクション5.1.3「サーバーコマンドオプション」

--log-name

セクション18.4.4「ndb_mgmd — MySQL Cluster 管理サーバーデーモン」

--log-output

セクション5.1.3「サーバーコマンドオプション」
セクション5.2.5「スロークエリーログ」
セクション5.2.3「一般クエリーログ」
セクション5.2.1「一般クエリーログおよびスロークエリーログの出力先の選択」

--log-queries-not-using-indexes

セクション5.1.3「サーバーコマンドオプション」

--log-raw

セクション5.1.3「サーバーコマンドオプション」
セクション6.1.2.3「パスワードおよびロギング」
セクション5.2.3「一般クエリーログ」

--log-short-format

セクション5.1.3「サーバーコマンドオプション」
セクション5.2.5「スロークエリーログ」
セクション17.1.4.4「バイナリログのオプションと変数」

--log-slave-updates

セクション18.6.10「MySQL Cluster レプリケーション: マルチマスターと循環レプリケーション」
セクション18.6.3「MySQL Cluster レプリケーションの既知の問題」
セクション18.3.4.2「MySQL Cluster 用の MySQL Server オプション」
セクション17.1.4.5「グローバルトランザクション ID のオプションと変数」
セクション5.2.4「バイナリログ」
セクション17.3.6「フェイルオーバー中にマスターを切り替える」
セクション17.1.4.3「レプリケーションスレーブのオプションと変数」
セクション17.4.5「レプリケーションバグまたは問題を報告する方法」
セクション17.3.5「レプリケーションパフォーマンスを改善する」

--log-slow-admin-statements

セクション5.1.3「サーバーコマンドオプション」
セクション5.1.4「サーバーシステム変数」

--log-slow-queries

セクション1.4「MySQL 5.6 の新機能」
セクション5.1.3「サーバーコマンドオプション」

--log-slow-slave-statements

セクション17.1.4.3「レプリケーションスレーブのオプションと変数」

--log-tc

セクション5.1.3「サーバーコマンドオプション」

--log-tc-size

セクション5.1.3「サーバーコマンドオプション」
セクション5.1.6「サーバーステータス変数」

--log-warnings

セクションB.5.2.9「MySQL サーバーが存在しなくなりました」
セクション5.2.2「エラーログ」
セクション5.1.3「サーバーコマンドオプション」
セクション17.1.4.3「レプリケーションスレーブのオプションと変数」
セクション5.2.4.3「混合形式のバイナリロギング形式」
セクションB.5.2.11「通信エラーおよび中止された接続」

--login-path

セクション4.7.3「my_print_defaults — オプションファイルからのオプションの表示」
セクション4.5.1.1「mysql のオプション」
セクション4.6.6「mysql_config_editor — MySQL 構成ユーティリティー」
セクション4.4.7「mysql_upgrade — MySQL テーブルのチェックとアップグレード」
セクション4.5.2「mysqladmin — MySQL サーバーの管理を行うクライアント」
セクション4.6.8「mysqlbinlog — バイナリログファイルを処理するためのユーティリティー」
セクション4.5.3「mysqlcheck — テーブル保守プログラム」
セクション4.5.4「mysqldump — データベースバックアッププログラム」
セクション4.5.5「mysqlimport — データインポートプログラム」
セクション4.5.6「mysqlshow — データベース、テーブル、およびカラム情報の表示」
セクション4.5.7「mysqlslap — 負荷エミュレーションクライアント」
セクション4.2.6「オプションファイルの使用」

セクション4.2.7「オプションファイルの処理に影響するコマンド行オプション」

--loops

セクション18.4.14「ndb_index_stat — NDB インデックス統計ユーティリティー」
セクション18.4.24「ndb_show_tables — NDB テーブルのリストの表示」
セクション18.4.2「ndbinfo_select_all — ndbinfo テーブルからの選択」

--loose

セクション4.2.5「プログラムオプション修飾子」

--loose-opt_name

セクション4.2.6「オプションファイルの使用」

--lossy-conversions

セクション18.4.20「ndb_restore — MySQL Cluster バックアップのリストア」

--low-priority

セクション4.5.5「mysqlimport — データインポートプログラム」

--low-priority-updates

セクション13.2.5「INSERT 構文」
セクションA.13「MySQL 5.6 FAQ: レプリケーション」
セクション5.1.3「サーバーコマンドオプション」
セクション8.10.2「テーブルロックの問題」
セクション8.10.3「同時挿入」

--lower-case-table-names

セクション9.2.2「識別子の大文字と小文字の区別」

M

[索引の先頭]

-M

セクション16.6.2「memcached の使用」

-m

セクション16.6.2「memcached の使用」
セクション4.6.3.2「myisamchk のチェックオプション」
セクション18.6.9「MySQL Cluster レプリケーションを使用した MySQL Cluster バックアップ」
セクション4.5.1.1「mysql のオプション」
セクション4.5.3「mysqlcheck — テーブル保守プログラム」
セクション18.4.20「ndb_restore — MySQL Cluster バックアップのリストア」

--malloc-lib

セクション4.3.2「mysqld_safe — MySQL サーバー起動スクリプト」

--master-connect-retry

セクション13.7.5.35「SHOW SLAVE STATUS 構文」
セクション17.1.4.3「レプリケーションスレーブのオプションと変数」

--master-data

セクション5.2「MySQL Server ログ」
セクション4.5.4「mysqldump — データベースバックアッププログラム」
セクション17.1.1.5「mysqldump を使用したデータスナップショットの作成」
セクション4.6.8.3「バイナリログファイルのバックアップのための mysqlbinlog の使用」
セクション7.3.1「バックアップポリシーの確立」
セクション17.1.3.3「フェイルオーバーおよびスケールアウトでの GTID の使用」
セクション17.2.2「レプリケーションリレーおよびステータスログ」

--master-host

セクション17.1.4.3「レプリケーションスレーブのオプションと変数」

--master-info-file

セクション17.2.2.2「スレーブステータスログ」
セクション17.1.4.3「レプリケーションスレーブのオプションと変数」

--master-info-repository

セクション13.4.2.1「CHANGE MASTER TO 構文」
セクション14.12「InnoDB の起動オプションおよびシステム変数」
セクション1.4「MySQL 5.6 の新機能」
セクション17.2.2.2「スレーブステータスログ」
セクション17.1.4.3「レプリケーションスレーブのオプションと変数」
セクション17.2.2「レプリケーションリレーおよびステータスログ」
セクション17.1.1.8「既存のデータによるレプリケーションのセットアップ」

--master-password

セクション17.1.4.3「レプリケーションスレーブのオプションと変数」

--master-port

セクション17.1.4.3「レプリケーションスレーブのオプションと変数」

--master-retry-count

セクション13.4.2.1「CHANGE MASTER TO 構文」
セクション13.7.5.35「SHOW SLAVE STATUS 構文」
セクション17.1.4.3「レプリケーションスレーブのオプションと変数」

--master-ssl

セクション17.1.4.3「レプリケーションスレーブのオプションと変数」

--master-ssl-ca

セクション17.1.4.3「レプリケーションスレーブのオプションと変数」

--master-ssl-capath
セクション17.1.4.3「レプリケーションスレーブのオプションと変数」

--master-ssl-cert
セクション17.1.4.3「レプリケーションスレーブのオプションと変数」

--master-ssl-cipher
セクション17.1.4.3「レプリケーションスレーブのオプションと変数」

--master-ssl-key
セクション17.1.4.3「レプリケーションスレーブのオプションと変数」

--master-user
セクション17.1.4.3「レプリケーションスレーブのオプションと変数」

--master-verify-checksum
セクション17.1.4.4「バイナリログのオプションと変数」

--max
セクション4.2.8「プログラム変数の設定へのオプションの使用」

--max-binlog-dump-events
セクション17.1.4.4「バイナリログのオプションと変数」

--max-binlog-size
セクション17.1.4.3「レプリケーションスレーブのオプションと変数」

--max-record-length
セクション4.6.3.3「myisamchk の修復オプション」
セクション13.7.2.5「REPAIR TABLE 構文」

--max-relay-log-size
セクション17.1.4.3「レプリケーションスレーブのオプションと変数」

--max-seeks-for-key
セクションB.5.6「オプティマイザ関連の問題」
セクション8.2.1.20「フルテーブルスキャンを回避する方法」

--max_a
セクション4.2.8「プログラム変数の設定へのオプションの使用」

--max_join_size
--safe-updates オプションの使用

--maximum
セクション4.2.5「プログラムオプション修飾子」

--maximum-query_cache_size
セクション8.9.3.3「クエリーキャッシュの構成」
セクション5.1.5「システム変数の使用」
セクション4.2.5「プログラムオプション修飾子」

--maximum-var_name
セクション5.1.3「サーバーコマンドオプション」
セクション5.1.5「システム変数の使用」

--medium-check
セクション4.6.3.2「myisamchk のチェックオプション」
セクション4.5.3「mysqlcheck — テーブル保守プログラム」

--memlock
セクション5.1.3「サーバーコマンドオプション」
セクション5.1.4「サーバーシステム変数」
セクション14.5.8「共有テーブルスペースでの RAW ディスクパーティションの使用」

--method
セクション4.6.10「mysqlhotcopy — データベースバックアッププログラム」

--min-examined-row-limit
セクション5.1.3「サーバーコマンドオプション」

--my-plugin
セクション5.1.8.1「プラグインのインストールおよびアンインストール」

--my-print-defaults
セクション4.4.4「mysql_plugin — MySQL サーバープラグインの構成」

--my_plugin
セクション5.1.8.1「プラグインのインストールおよびアンインストール」

--mycnf
セクション18.4.7「ndb_config — MySQL Cluster 構成情報の抽出」
セクション18.4.4「ndb_mgmd — MySQL Cluster 管理サーバーデーモン」

--myisam-block-size
セクション8.9.2.5「キーキャッシュブロックサイズ」
セクション5.1.3「サーバーコマンドオプション」

--myisam-recover
セクション5.1.3「サーバーコマンドオプション」

--myisam-recover-options
セクション8.6.1「MyISAM クエリーの最適化」
セクション15.2「MyISAM ストレージエンジン」
セクション7.6.5「MyISAM テーブル保守スケジュールのセットアップ」
セクション15.2.1「MyISAM 起動オプション」
セクション24.4.1.6「mysqld でのエラーの原因を見つけるためのサーバーログの使用」
セクション8.2.5「その他の最適化のヒント」
セクション5.1.3「サーバーコマンドオプション」
セクション5.1.4「サーバーシステム変数」

セクションB.5.2.19「テーブルの破損の問題」

--myisam_sort_buffer_size

セクション4.6.3.6「myisamchk メモリー使用量」

MYSQL_MAINTAINER_MODE

セクション2.9.5「MySQL のコンパイルに関する問題」

MYSQL_TCP_PORT

セクション2.9.4「MySQL ソース構成オプション」
セクション2.9.3「開発ソースツリーを使用して MySQL をインストールする」

MYSQL_UNIX_ADDR

セクションB.5.4.5「MySQL の UNIX ソケットファイルを保護または変更する方法」
セクション2.9.4「MySQL ソース構成オプション」
セクション2.9.3「開発ソースツリーを使用して MySQL をインストールする」

--mysqladmin

セクション4.3.4「mysqld_multi — 複数の MySQL サーバーの管理」

--mysqld

セクション4.4.4「mysql_plugin — MySQL サーバープラグインの構成」
セクション4.3.4「mysqld_multi — 複数の MySQL サーバーの管理」
セクション4.3.2「mysqld_safe — MySQL サーバー起動スクリプト」

--mysqld-version

セクション4.3.2「mysqld_safe — MySQL サーバー起動スクリプト」

N

[索引の先頭]

-N

セクション4.4.1「comp_err — MySQL エラーメッセージファイルのコンパイル」
セクション4.5.1.1「mysql のオプション」
セクション4.5.4「mysqldump — データベースバックアッププログラム」
セクション18.4.23「ndb_setup.py — MySQL Cluster のブラウザベース自動インストーラの開始」

-n

セクション16.6.2「memcached の使用」
セクション4.7.3「my_print_defaults — オプションファイルからのオプションの表示」
セクション4.6.3.3「myisamchk の修復オプション」
セクション18.5.2「MySQL Cluster 管理クライアントのコマンド」
セクション4.5.1.1「mysql のオプション」
セクション4.5.4「mysqldump — データベースバックアッププログラム」
セクション4.6.9「mysqldumpslow — スロークエリーログファイルの要約」
セクション4.6.10「mysqlhotcopy — データベースバックアッププログラム」
セクション18.4.10「ndb_desc — NDB テーブルの表示」
セクション18.4.20「ndb_restore — MySQL Cluster バックアップのリストア」
セクション18.4.23「ndb_setup.py — MySQL Cluster のブラウザベース自動インストーラの開始」
セクション18.4.26「ndb_waiter — MySQL Cluster が指定したステータスになるまで待機する」
セクション18.4.1「ndbd — MySQL Cluster データノードデーモン」
セクション18.4.19「ndbd_redo_log_reader — クラスタ Redo ログ内容のチェックおよび出力」
セクション4.7.4「resolve_stack_dump — 数値スタックトレースダンプをシンボルに解決」

--name_file

セクション4.4.1「comp_err — MySQL エラーメッセージファイルのコンパイル」

--named-commands

セクション4.5.1.1「mysql のオプション」

--ndb

セクション4.8.1「perror — エラーコードの説明」

--ndb-batch-size

セクション18.3.4.2「MySQL Cluster 用の MySQL Server オプション」

--ndb-blob-read-batch-bytes

セクション18.3.4.2「MySQL Cluster 用の MySQL Server オプション」

--ndb-blob-write-batch-bytes

セクション18.3.4.2「MySQL Cluster 用の MySQL Server オプション」

--ndb-cluster

セクションA.10「MySQL 5.6 FAQ: MySQL Cluster」

--ndb-cluster-connection-pool

セクション18.3.4.2「MySQL Cluster 用の MySQL Server オプション」

--ndb-connectstring

セクションA.10「MySQL 5.6 FAQ: MySQL Cluster」
セクション18.5.4「MySQL Cluster での MySQL サーバーの使用法」
セクション18.5.11.2「MySQL Cluster と MySQL 権限」
セクション18.1.1「MySQL Cluster の主な概念」
セクション18.3.2.2「MySQL Cluster の推奨される初期構成」
セクション18.4.27「MySQL Cluster プログラムに共通するオプション — MySQL Cluster プログラムに共通するオプション」
セクション18.3.4.2「MySQL Cluster 用の MySQL Server オプション」
セクション18.4.7「ndb_config — MySQL Cluster 構成情報の抽出」

セクション18.2.3.1「Windows でのバイナリリリースからの MySQL Cluster のインストール」
セクション18.6.5「レプリケーションのための MySQL Cluster の準備」

--ndb-deferred-constraints

セクション18.3.4.2「MySQL Cluster 用の MySQL Server オプション」

--ndb-distribution

セクション18.3.4.2「MySQL Cluster 用の MySQL Server オプション」

--ndb-force-send

セクション18.3.2.2「MySQL Cluster の推奨される初期構成」

--ndb-index-stat-enable

セクション18.3.2.2「MySQL Cluster の推奨される初期構成」

--ndb-log-apply-status

セクション18.3.4.3「MySQL Cluster のシステム変数」
セクション18.3.4.2「MySQL Cluster 用の MySQL Server オプション」

--ndb-log-empty-epochs

セクション18.6.4「MySQL Cluster レプリケーションスキーマとテーブル」
セクション18.3.4.2「MySQL Cluster 用の MySQL Server オプション」

--ndb-log-exclusive-reads

セクション18.3.4.2「MySQL Cluster 用の MySQL Server オプション」

--ndb-log-orig

セクション18.3.4.3「MySQL Cluster のシステム変数」
セクション18.6.4「MySQL Cluster レプリケーションスキーマとテーブル」
セクション18.3.4.2「MySQL Cluster 用の MySQL Server オプション」

--ndb-log-transaction-id

セクション18.3.4.3「MySQL Cluster のシステム変数」
セクション18.6.11「MySQL Cluster レプリケーションの競合解決」
セクション18.3.4.2「MySQL Cluster 用の MySQL Server オプション」
セクション17.1.4.4「バイナリログのオプションと変数」

--ndb-log-update-as-write

セクション18.6.3「MySQL Cluster レプリケーションの既知の問題」
セクション18.6.11「MySQL Cluster レプリケーションの競合解決」

--ndb-log-updated-only

セクション18.6.11「MySQL Cluster レプリケーションの競合解決」

--ndb-mgmd-host

セクション18.4.27「MySQL Cluster プログラムに共通するオプション — MySQL Cluster プログラムに共通するオプション」
セクション18.3.4.2「MySQL Cluster 用の MySQL Server オプション」

--ndb-nodegroup-map

セクション18.4.20「ndb_restore — MySQL Cluster バックアップのリストア」

--ndb-nodeid

セクション18.4.27「MySQL Cluster プログラムに共通するオプション — MySQL Cluster プログラムに共通するオプション」
セクション18.3.4.2「MySQL Cluster 用の MySQL Server オプション」
セクション18.4.4「ndb_mgmd — MySQL Cluster 管理サーバーデーモン」
セクション18.4.1「ndbd — MySQL Cluster データノードデーモン」

--ndb-optimized-node-selection

セクション18.4.27「MySQL Cluster プログラムに共通するオプション — MySQL Cluster プログラムに共通するオプション」

--ndb-recv-thread-activation-threshold

セクション18.3.4.3「MySQL Cluster のシステム変数」
セクション18.3.4.2「MySQL Cluster 用の MySQL Server オプション」

--ndb-recv-thread-cpu-mask

セクション18.3.4.3「MySQL Cluster のシステム変数」
セクション18.3.4.2「MySQL Cluster 用の MySQL Server オプション」

--ndb-transid-mysql-connection-map

セクション21.30.2「INFORMATION_SCHEMA ndb_transid_mysql_connection_map テーブル」

ndb-transid-mysql-connection-map

セクション18.3.4.2「MySQL Cluster 用の MySQL Server オプション」

--ndb-use-exact-count

セクション18.3.2.2「MySQL Cluster の推奨される初期構成」

--ndb-wait-connected

セクション18.3.4.2「MySQL Cluster 用の MySQL Server オプション」

--ndb-wait-setup

セクション18.3.4.2「MySQL Cluster 用の MySQL Server オプション」

--ndb_optimization_delay

セクション18.3.4.2「MySQL Cluster 用の MySQL Server オプション」

--ndbcluster
セクションA.10「MySQL 5.6 FAQ: MySQL Cluster」
セクション18.5.4「MySQL Cluster での MySQL サーバーの使用法」
セクション18.5.11.2「MySQL Cluster と MySQL 権限」
セクション18.1.1「MySQL Cluster の主な概念」
セクション18.3.2.2「MySQL Cluster の推奨される初期構成」
セクション18.3「MySQL Cluster の構成」
セクション18.3.4.2「MySQL Cluster 用の MySQL Server オプション」
セクション18.5.10「ndbinfo MySQL Cluster 情報データベース」
セクション13.7.5.17「SHOW ENGINES 構文」
セクション18.2.3.1「Windows でのバイナリリリースからの MySQL Cluster のインストール」

--ndbinfo-database
セクション18.3.4.3「MySQL Cluster のシステム変数」

--ndbinfo-table-prefix
セクション18.3.4.3「MySQL Cluster のシステム変数」

net_retry_count
セクション17.2.1「レプリケーション実装の詳細」

net_write_timeout
セクション17.2.1「レプリケーション実装の詳細」

--new
セクション4.2.6「オプションファイルの使用」

--nice
セクション4.3.2「mysqld_safe — MySQL サーバー起動スクリプト」

--no-auto-rehash
セクション4.5.1.1「mysql のオプション」

--no-autocommit
セクション4.5.4「mysqldump — データベースバックアッププログラム」

--no-beep
セクション4.5.1.1「mysql のオプション」
セクション4.5.2「mysqladmin — MySQL サーバーの管理を行うクライアント」

--no-binlog
セクション18.4.20「ndb_restore — MySQL Cluster バックアップのリストア」

--no-browser
セクション18.4.23「ndb_setup.py — MySQL Cluster のブラウザベース自動インストーラの開始」

--no-contact
セクション18.4.26「ndb_waiter — MySQL Cluster が指定したステータスになるまで待機する」

--no-create-db
セクション4.5.4「mysqldump — データベースバックアッププログラム」

--no-create-info
セクション4.5.4「mysqldump — データベースバックアッププログラム」
セクション7.4.5.4「テーブル定義と内容の個別のダンプ」

--no-data
セクション4.5.4「mysqldump — データベースバックアッププログラム」
セクション7.4.5.4「テーブル定義と内容の個別のダンプ」

--no-defaults
セクション4.7.3「my_print_defaults — オプションファイルからのオプションの表示」
セクション4.6.3.1「myisamchk の一般オプション」
セクション4.5.1.1「mysql のオプション」
セクション4.6.6「mysql_config_editor — MySQL 構成ユーティリティー」
セクション4.4.3「mysql_install_db — MySQL データディレクトリの初期化」
セクション4.4.4「mysql_plugin — MySQL サーバープラグインの構成」
セクション4.4.7「mysql_upgrade — MySQL テーブルのチェックとアップグレード」
セクション4.5.2「mysqladmin — MySQL サーバーの管理を行うクライアント」
セクション4.6.8「mysqlbinlog — バイナリログファイルを処理するためのユーティリティー」
セクション4.5.3「mysqlcheck — テーブル保守プログラム」
セクション4.3.4「mysqld_multi — 複数の MySQL サーバーの管理」
セクション4.3.2「mysqld_safe — MySQL サーバー起動スクリプト」
セクション4.5.4「mysqldump — データベースバックアッププログラム」
セクション4.5.5「mysqlimport — データインポートプログラム」
セクション4.5.6「mysqlshow — データベース、テーブル、およびカラム情報の表示」
セクション4.5.7「mysqlslap — 負荷エミュレーションクライアント」
セクション6.2.7「アクセス拒否エラーの原因」
セクション4.2.7「オプションファイルの処理に影響するコマンド行オプション」
セクション5.1.3「サーバーコマンドオプション」

--no-drop
セクション4.5.7「mysqlslap — 負荷エミュレーションクライアント」

--no-log
セクション4.3.4「mysqld_multi — 複数の MySQL サーバーの管理」

--no-nodeid-checks
セクション18.4.4「ndb_mgmd — MySQL Cluster 管理サーバーデーモン」

--no-restore-disk-objects
セクション18.4.20「ndb_restore — MySQL Cluster バックアップのリストア」

--no-set-names
セクション4.5.4「mysqldump — データベースバックアッププログラム」

--no-symlinks
セクション4.6.3.3「myisamchk の修復オプション」

--no-tablespaces
セクション4.5.4「mysqldump — データベースバックアッププログラム」

--no-upgrade
セクション18.4.20「ndb_restore — MySQL Cluster バックアップのリストア」

--nodaemon
セクション18.4.4「ndb_mgmd — MySQL Cluster 管理サーバーデーモン」
セクション18.4.1「ndbd — MySQL Cluster データノードデーモン」

--nodata
セクション18.4.21「ndb_select_all — NDB テーブルの行の出力」

--nodeid
セクション18.4.7「ndb_config — MySQL Cluster 構成情報の抽出」
セクション18.4.20「ndb_restore — MySQL Cluster バックアップのリストア」

--nodes
セクション18.4.7「ndb_config — MySQL Cluster 構成情報の抽出」

--noindices
セクション4.6.10「mysqlhotcopy — データベースバックアッププログラム」

--nostart
セクション18.5.2「MySQL Cluster 管理クライアントのコマンド」
セクション18.4.1「ndbd — MySQL Cluster データノードデーモン」

--not-started
セクション18.4.26「ndb_waiter — MySQL Cluster が指定したステータスになるまで待機する」

--nowait-nodes
セクション18.5.13.3「MySQL Cluster データノードのオンライン追加: 詳細な例」
セクション18.3.2.6「MySQL Cluster データノードの定義」
セクション18.4.4「ndb_mgmd — MySQL Cluster 管理サーバーデーモン」
セクション18.4.26「ndb_waiter — MySQL Cluster が指定したステータスになるまで待機する」
セクション18.4.1「ndbd — MySQL Cluster データノードデーモン」

--number-char-cols
セクション4.5.7「mysqlslap — 負荷エミュレーションクライアント」

--number-int-cols
セクション4.5.7「mysqlslap — 負荷エミュレーションクライアント」

--number-of-queries
セクション4.5.7「mysqlslap — 負荷エミュレーションクライアント」

--numeric-dump-file
セクション4.7.4「resolve_stack_dump — 数値スタックトレースダンプをシンボルに解決」

O

[索引の先頭]

-O
セクション4.4.1「comp_err — MySQL エラーメッセージファイルのコンパイル」
セクション2.9.4「MySQL ソース構成オプション」

-o
セクション23.6.1「libmysqld によるプログラムのコンパイル」
セクション4.6.3.3「myisamchk の修復オプション」
セクション4.6.4「myisamlog — MyISAM ログファイルの内容の表示」
セクション4.5.1.1「mysql のオプション」
セクション4.6.8「mysqlbinlog — バイナリログファイルを処理するためのユーティリティー」
セクション4.5.3「mysqlcheck — テーブル保守プログラム」
セクション18.4.21「ndb_select_all — NDB テーブルの行の出力」
セクション18.4.23「ndb_setup.py — MySQL Cluster のブラウザベース自動インストーラの開始」
セクション8.11.3「ディスク I/O の最適化」

--offset
セクション4.6.8「mysqlbinlog — バイナリログファイルを処理するためのユーティリティー」

--old-alter-table
セクション5.1.3「サーバーコマンドオプション」

--old-style-user-limits
セクション6.3.4「アカウントリソース制限の設定」
セクション5.1.3「サーバーコマンドオプション」

--old_server
セクション4.6.7「mysqlaccess — アクセス権限をチェックするクライアント」

セクション4.6.10「mysqlhotcopy — データベースバックアッププログラム」

ON
セクション3.3.4.9「複数のテーブルの使用」

--one-database
セクション4.5.1.1「mysql のオプション」

--one-thread
セクション1.4「MySQL 5.6 の新機能」
セクション5.1.3「サーバーコマンドオプション」

--only-print
セクション4.5.7「mysqlslap — 負荷エミュレーションクライアント」

--open-files-limit
セクションB.5.2.18「'File' が見つかりません、および同様のエラー」
セクション14.12「InnoDB の起動オプションおよびシステム変数」
セクション8.4.3.1「MySQL でのテーブルのオープンとクローズの方法」
セクション4.3.2「mysqld_safe — MySQL サーバー起動スクリプト」
セクション5.1.3「サーバーコマンドオプション」

open-files-limit
セクションB.5.2.7「接続が多すぎます」

--opt
セクション8.5.4「InnoDB テーブルの一括データロード」
セクション4.5.4「mysqldump — データベースバックアッププログラム」

--opt_name
セクション4.2.6「オプションファイルの使用」

--optimize
セクション4.5.3「mysqlcheck — テーブル保守プログラム」

--order
セクション18.4.21「ndb_select_all — NDB テーブルの行の出力」

--order-by-primary
セクション4.5.4「mysqldump — データベースバックアッププログラム」

--out_dir
セクション4.4.1「comp_err — MySQL エラーメッセージファイルのコンパイル」

--out_file
セクション4.4.1「comp_err — MySQL エラーメッセージファイルのコンパイル」

P
[索引の先頭]

-P
セクション16.6.2「memcached の使用」
セクション4.5.1.1「mysql のオプション」
セクション4.2.2「MySQL サーバーへの接続」
セクション4.2.1「MySQL プログラムの起動」
セクション4.6.6「mysql_config_editor — MySQL 構成ユーティリティー」
セクション4.4.4「mysql_plugin — MySQL サーバープラグインの構成」
セクション4.4.7「mysql_upgrade — MySQL テーブルのチェックとアップグレード」
セクション4.6.7「mysqlaccess — アクセス権限をチェックするクライアント」
セクション4.5.2「mysqladmin — MySQL サーバーの管理を行うクライアント」
セクション4.6.8「mysqlbinlog — バイナリログファイルを処理するためのユーティリティー」
セクション4.5.3「mysqlcheck — テーブル保守プログラム」
セクション4.5.4「mysqldump — データベースバックアッププログラム」
セクション4.6.10「mysqlhotcopy — データベースバックアッププログラム」
セクション4.5.5「mysqlimport — データインポートプログラム」
セクション4.5.6「mysqlshow — データベース、テーブル、およびカラム情報の表示」
セクション4.5.7「mysqlslap — 負荷エミュレーションクライアント」
セクション18.4.4「ndb_mgmd — MySQL Cluster 管理サーバーデーモン」
セクション5.1.3「サーバーコマンドオプション」

-p
セクション4.6.1「innochecksum — オフライン InnoDB ファイルチェックサムユーティリティー」
セクション19.2.5「KEY パーティショニング」
セクション16.6.2「memcached の使用」
セクション4.6.3.3「myisamchk の修復オプション」
セクション4.6.4「myisamlog — MyISAM ログファイルの内容の表示」
セクション18.5.13.3「MySQL Cluster データノードのオンライン追加: 詳細な例」
セクション4.5.1.1「mysql のオプション」
セクション2.3.5.8「MySQL インストールのテスト」
セクション4.2.2「MySQL サーバーへの接続」
セクション4.2.1「MySQL プログラムの起動」
セクション4.6.6「mysql_config_editor — MySQL 構成ユーティリティー」
セクション4.4.4「mysql_plugin — MySQL サーバープラグインの構成」
セクション4.4.7「mysql_upgrade — MySQL テーブルのチェックとアップグレード」
セクション4.6.7「mysqlaccess — アクセス権限をチェックするクライアント」
セクション4.5.2「mysqladmin — MySQL サーバーの管理を行うクライアント」
セクション4.6.8「mysqlbinlog — バイナリログファイルを処理するためのユーティリティー」
セクション4.5.3「mysqlcheck — テーブル保守プログラム」
セクション4.5.4「mysqldump — データベースバックアッププログラム」
セクション4.6.10「mysqlhotcopy — データベースバックアッププログラム」

セクション4.5.5「mysqlimport — データインポートプログラム」
セクション4.5.6「mysqlshow — データベース、テーブル、およびカラム情報の表示」
セクション4.5.7「mysqlslap — 負荷エミュレーションクライアント」
セクション18.4.10「ndb_desc — NDB テーブルの表示」
セクション18.4.20「ndb_restore — MySQL Cluster バックアップのリストア」
セクション18.4.21「ndb_select_all — NDB テーブルの行の出力」
セクション18.4.23「ndb_setup.py — MySQL Cluster のブラウザベース自動インストーラの開始」
セクション18.4.24「ndb_show_tables — NDB テーブルのリストの表示」
セクション18.5.10.4「ndbinfo cluster_operations テーブル」
セクション18.5.10.20「ndbinfo server_operations テーブル」
セクション2.3.8「Windows でのインストール後の手順」
セクション2.3.5.5「Windows のコマンド行からの MySQL の起動」
セクション2.3.5.7「Windows のサービスとして MySQL を起動する」
セクション2.3.7「Windows 上の MySQL をアップグレードする」
セクション6.2.7「アクセス拒否エラーの原因」
セクション4.2.4「コマンド行でのオプションの使用」
セクションB.5.2.5「パスワードをインタラクティブに入力すると失敗する」
セクション6.1.2.1「パスワードセキュリティーのためのエンドユーザーガイドライン」
セクション6.3.2「ユーザーアカウントの追加」
セクション6.3.1「ユーザー名とパスワード」
セクション2.10.2「最初の MySQL アカウントのセキュリティー設定」

--pager

セクション4.5.1.1「mysql のオプション」
セクション4.5.1.2「mysql コマンド」

--parallel-recover

セクション4.6.3.3「myisamchk の修復オプション」

--parallelism

セクション18.4.20「ndb_restore — MySQL Cluster バックアップのリストア」

parallelism

セクション18.4.21「ndb_select_all — NDB テーブルの行の出力」

--parsable

セクション18.4.24「ndb_show_tables — NDB テーブルのリストの表示」

--partition

セクション5.1.3「サーバーコマンドオプション」

--password

セクション4.5.1.1「mysql のオプション」
セクション4.2.2「MySQL サーバーへの接続」
セクション4.2.1「MySQL プログラムの起動」
セクション4.6.6「mysql_config_editor — MySQL 構成ユーティリティー」
セクション4.6.11「mysql_convert_table_format — 指定されたストレージエンジンを使用するためのテーブルの変換」
セクション4.6.14「mysql_setpermission — 付与テーブルに許可をインタラクティブに設定」
セクション4.4.7「mysql_upgrade — MySQL テーブルのチェックとアップグレード」
セクション4.6.7「mysqlaccess — アクセス権限をチェックするクライアント」
セクション4.5.2「mysqladmin — MySQL サーバーの管理を行うクライアント」
セクション4.6.8「mysqlbinlog — バイナリログファイルを処理するためのユーティリティー」
セクション4.5.3「mysqlcheck — テーブル保守プログラム」
セクション4.3.4「mysqld_multi — 複数の MySQL サーバーの管理」
セクション4.5.4「mysqldump — データベースバックアッププログラム」
セクション4.6.10「mysqlhotcopy — データベースバックアッププログラム」
セクション4.5.5「mysqlimport — データインポートプログラム」
セクション4.5.6「mysqlshow — データベース、テーブル、およびカラム情報の表示」
セクション4.5.7「mysqlslap — 負荷エミュレーションクライアント」
セクション6.2.7「アクセス拒否エラーの原因」
セクション4.2.4「コマンド行でのオプションの使用」
セクション6.3.8.9「テスト認証プラグイン」
セクション4.6.8.3「バイナリログファイルのバックアップのための mysqlbinlog の使用」
セクション7.3「バックアップおよびリカバリ戦略の例」
セクションB.5.2.5「パスワードをインタラクティブに入力すると失敗する」
セクション6.1.2.1「パスワードセキュリティーのためのエンドユーザーガイドライン」
セクション6.3.7「プラガブル認証」
セクション6.3.2「ユーザーアカウントの追加」
セクション6.3.1「ユーザー名とパスワード」

password

セクション4.7.3「my_print_defaults — オプションファイルからのオプションの表示」
セクション4.5.1.1「mysql のオプション」
セクション4.6.6「mysql_config_editor — MySQL 構成ユーティリティー」
セクション4.4.7「mysql_upgrade — MySQL テーブルのチェックとアップグレード」
セクション4.5.2「mysqladmin — MySQL サーバーの管理を行うクライアント」
セクション4.6.8「mysqlbinlog — バイナリログファイルを処理するためのユーティリティー」
セクション4.5.3「mysqlcheck — テーブル保守プログラム」
セクション4.5.4「mysqldump — データベースバックアッププログラム」
セクション4.5.5「mysqlimport — データインポートプログラム」
セクション4.5.6「mysqlshow — データベース、テーブル、およびカラム情報の表示」
セクション4.5.7「mysqlslap — 負荷エミュレーションクライアント」
セクション4.2.6「オプションファイルの使用」

セクション4.2.7「オプションファイルの処理に影響するコマンド行オプション」

--performance-schema-consumer-consumer_name
セクション22.11「パフォーマンススキーマコマンドオプション」

--performance-schema-consumer-events-stages-current
セクション22.11「パフォーマンススキーマコマンドオプション」

--performance-schema-consumer-events-stages-history
セクション22.11「パフォーマンススキーマコマンドオプション」

--performance-schema-consumer-events-stages-history-long
セクション22.11「パフォーマンススキーマコマンドオプション」

--performance-schema-consumer-events-statements-current
セクション22.11「パフォーマンススキーマコマンドオプション」

--performance-schema-consumer-events-statements-history
セクション22.11「パフォーマンススキーマコマンドオプション」

--performance-schema-consumer-events-statements-history-long
セクション22.11「パフォーマンススキーマコマンドオプション」

--performance-schema-consumer-events-waits-current
セクション22.11「パフォーマンススキーマコマンドオプション」

--performance-schema-consumer-events-waits-history
セクション22.11「パフォーマンススキーマコマンドオプション」

--performance-schema-consumer-events-waits-history-long
セクション22.11「パフォーマンススキーマコマンドオプション」

--performance-schema-consumer-global-instrumentation
セクション22.11「パフォーマンススキーマコマンドオプション」

--performance-schema-consumer-statements-digest
セクション22.11「パフォーマンススキーマコマンドオプション」

--performance-schema-consumer-thread-instrumentation
セクション22.11「パフォーマンススキーマコマンドオプション」

--performance-schema-instrument
セクション22.11「パフォーマンススキーマコマンドオプション」
セクション22.2.2「パフォーマンススキーマ起動構成」

--performance-schema-xxx
セクション5.1.3「サーバーコマンドオプション」

--performance_schema_max_mutex_classes
セクション22.5「パフォーマンススキーマステータスモニタリング」

--performance_schema_max_mutex_instances
セクション22.5「パフォーマンススキーマステータスモニタリング」

--pid-file
セクション5.3「1 つのマシン上での複数の MySQL インスタンスの実行」
セクション4.3.3「mysql.server — MySQL サーバー起動スクリプト」
セクション4.3.4「mysqld_multi — 複数の MySQL サーバーの管理」
セクション4.3.2「mysqld_safe — MySQL サーバー起動スクリプト」
セクション5.1.3「サーバーコマンドオプション」
セクション5.1.4「サーバーシステム変数」

pid-file
セクション2.10.1.2「MySQL を自動的に起動および停止する」

--pipe
セクション4.5.1.1「mysql のオプション」
セクション2.3.5.8「MySQL インストールのテスト」
セクション4.2.2「MySQL サーバーへの接続」
セクション4.4.7「mysql_upgrade — MySQL テーブルのチェックとアップグレード」
セクション4.5.2「mysqladmin — MySQL サーバーの管理を行うクライアント」
セクション4.5.3「mysqlcheck — テーブル保守プログラム」

セクション4.5.4「mysqldump — データベースバックアッププログラム」
セクション4.5.5「mysqlimport — データインポートプログラム」
セクション4.5.6「mysqlshow — データベース、テーブル、およびカラム情報の表示」
セクション4.5.7「mysqlslap — 負荷エミュレーションクライアント」

--plan

セクション4.6.7「mysqlaccess — アクセス権限をチェックするクライアント」

--plugin

セクション5.1.3「サーバーコマンドオプション」

--plugin-dir

セクション23.7.14「C API クライアントプラグイン関数」
セクション4.5.1.1「mysql のオプション」
セクション23.7.14.3「mysql_load_plugin()」
セクション4.4.4「mysql_plugin — MySQL サーバープラグインの構成」
セクション4.4.7「mysql_upgrade — MySQL テーブルのチェックとアップグレード」
セクション4.5.2「mysqladmin — MySQL サーバーの管理を行うクライアント」
セクション4.6.8「mysqlbinlog — バイナリログファイルを処理するためのユーティリティー」
セクション4.5.3「mysqlcheck — テーブル保守プログラム」
セクション4.3.2「mysqld_safe — MySQL サーバー起動スクリプト」
セクション4.5.4「mysqldump — データベースバックアッププログラム」
セクション4.5.5「mysqlimport — データインポートプログラム」
セクション4.5.6「mysqlshow — データベース、テーブル、およびカラム情報の表示」
セクション4.5.7「mysqlslap — 負荷エミュレーションクライアント」
クライアントプラグインディスクリプタ
セクションD.9「プラガブルな認証の制約」
セクション6.3.7「プラガブル認証」
認証プラグインの使用

--plugin-ini

セクション4.4.4「mysql_plugin — MySQL サーバープラグインの構成」

--plugin-innodb_file_per_table

セクション5.1.3「サーバーコマンドオプション」

--plugin-load

セクション13.7.3.3「INSTALL PLUGIN 構文」
セクション2.9.4「MySQL ソース構成オプション」
セクション24.2「MySQL プラグイン API」
セクション4.4.4「mysql_plugin — MySQL サーバープラグインの構成」
PAM 認証プラグインのインストール
Windows 認証プラグインのインストール
セクション5.1.3「サーバーコマンドオプション」
サーバープラグインライブラリおよびプラグインディスクリプタ
セクション8.11.6.1「スレッドプールコンポーネントとインストール」
パスワード検証プラグインのインストール
パスワード検証プラグインのオプションおよび変数
セクション6.3.7「プラガブル認証」
セクション24.2.2「プラグイン API のコンポーネント」
セクション5.1.8.1「プラグインのインストールおよびアンインストール」
セクション24.2.4.2「プラグインのデータ構造体」
セクション6.3.12.1「監査ログプラグインのインストール」
セクション6.3.12.6「監査ログプラグインのオプションおよびシステム変数」
認証プラグインの使用

--plugin-load-add

セクション5.1.3「サーバーコマンドオプション」
セクション5.1.8.1「プラグインのインストールおよびアンインストール」

--plugin-sql-mode

セクション5.1.3「サーバーコマンドオプション」

--plugin-xxx

セクション5.1.3「サーバーコマンドオプション」

--plugin_dir

セクション2.9.4「MySQL ソース構成オプション」
セクション24.2.2「プラグイン API のコンポーネント」

--plugin_name

セクション4.4.4「mysql_plugin — MySQL サーバープラグインの構成」
セクション5.1.8.1「プラグインのインストールおよびアンインストール」

--plugindir

セクション4.7.2「mysql_config — クライアントのコンパイル用オプションの表示」

--port

セクション5.3「1 つのマシン上での複数の MySQL インスタンスの実行」
セクション4.5.1.1「mysql のオプション」
セクション2.10.1.3「MySQL サーバーの起動とトラブルシューティング」
セクション4.2.2「MySQL サーバーへの接続」
セクション2.9.4「MySQL ソース構成オプション」
セクション4.2.1「MySQL プログラムの起動」
セクション4.7.2「mysql_config — クライアントのコンパイル用オプションの表示」
セクション4.6.6「mysql_config_editor — MySQL 構成ユーティリティー」
セクション4.6.11「mysql_convert_table_format — 指定されたストレージエンジンを使用するためのテーブルの変換」
セクション4.6.14「mysql_setpermission — 付与テーブルに許可をインタラクティブに設定」
セクション4.4.7「mysql_upgrade — MySQL テーブルのチェックとアップグレード」
セクション4.5.2「mysqladmin — MySQL サーバーの管理を行うクライアント」
セクション4.6.8「mysqlbinlog — バイナリログファイルを処理するためのユーティリティー」

セクション4.5.3「mysqlcheck — テーブル保守プログラム」
セクション4.3.2「mysqld_safe — MySQL サーバー起動スクリプト」
セクション4.5.4「mysqldump — データベースバックアッププログラム」
セクション4.6.10「mysqlhotcopy — データベースバックアッププログラム」
セクション4.5.5「mysqlimport — データインポートプログラム」
セクション4.5.6「mysqlshow — データベース、テーブル、およびカラム情報の表示」
セクション4.5.7「mysqlslap — 負荷エミュレーションクライアント」
セクション18.4.23「ndb_setup.py — MySQL Cluster のブラウザベース自動インストーラの開始」
セクション5.3.3「Unix 上での複数の MySQL インスタンスの実行」
セクション6.2.7「アクセス拒否エラーの原因」
セクション5.1.3「サーバーコマンドオプション」
セクション5.1.4「サーバーシステム変数」
セクション5.3.4「複数サーバー環境でのクライアントプログラムの使用」

--port-open-timeout

セクション5.1.3「サーバーコマンドオプション」

--post-query

セクション4.5.7「mysqlslap — 負荷エミュレーションクライアント」

--post-system

セクション4.5.7「mysqlslap — 負荷エミュレーションクライアント」

--pre-query

セクション4.5.7「mysqlslap — 負荷エミュレーションクライアント」

--pre-system

セクション4.5.7「mysqlslap — 負荷エミュレーションクライアント」

PREFIX

セクション18.2.2.3「Linux でのソースからの MySQL Cluster のビルド」

--prefix

セクション16.6.1「memcached のインストール」

--preserve-trailing-spaces

セクション18.4.20「ndb_restore — MySQL Cluster バックアップのリストア」

--preview

セクション4.6.7「mysqlaccess — アクセス権限をチェックするクライアント」

--print

セクション18.4.20「ndb_restore — MySQL Cluster バックアップのリストア」

--print-defaults

セクション4.6.3.1「myisamchk の一般オプション」
セクション2.11.1「MySQL のアップグレード」
セクション4.5.1.1「mysql のオプション」
セクション4.4.4「mysql_plugin — MySQL サーバープラグインの構成」
セクション4.4.7「mysql_upgrade — MySQL テーブルのチェックとアップグレード」
セクション4.5.2「mysqladmin — MySQL サーバーの管理を行うクライアント」
セクション4.6.8「mysqlbinlog — バイナリログファイルを処理するためのユーティリティー」
セクション4.5.3「mysqlcheck — テーブル保守プログラム」
セクション4.5.4「mysqldump — データベースバックアッププログラム」
セクション4.5.5「mysqlimport — データインポートプログラム」
セクション4.5.6「mysqlshow — データベース、テーブル、およびカラム情報の表示」
セクション4.5.7「mysqlslap — 負荷エミュレーションクライアント」
セクション4.2.7「オプションファイルの処理に影響するコマンド行オプション」
セクション5.1.3「サーバーコマンドオプション」

--print-full-config

セクション18.4.4「ndb_mgmd — MySQL Cluster 管理サーバーデーモン」

--print_*

セクション18.4.20「ndb_restore — MySQL Cluster バックアップのリストア」

--print_data

セクション18.4.20「ndb_restore — MySQL Cluster バックアップのリストア」

--print_log

セクション18.4.20「ndb_restore — MySQL Cluster バックアップのリストア」

--print_meta

セクション18.4.20「ndb_restore — MySQL Cluster バックアップのリストア」

--progress-frequency

セクション18.4.20「ndb_restore — MySQL Cluster バックアップのリストア」

--promote-attributes

セクション18.4.20「ndb_restore — MySQL Cluster バックアップのリストア」

--prompt

セクション4.5.1.1「mysql のオプション」
セクション4.5.1.2「mysql コマンド」

--protocol

セクション4.5.1.1「mysql のオプション」
セクション1.3.2「MySQL の主な機能」

セクション2.3.5.8「MySQL インストールのテスト」
セクション4.2.2「MySQL サーバーへの接続」
セクション4.4.7「mysql_upgrade — MySQL テーブルのチェックとアップグレード」
セクション4.5.2「mysqladmin — MySQL サーバーの管理を行うクライアント」
セクション4.6.8「mysqlbinlog — バイナリログファイルを処理するためのユーティリティー」
セクション4.5.3「mysqlcheck — テーブル保守プログラム」
セクション4.5.4「mysqldump — データベースバックアッププログラム」
セクション4.5.5「mysqlimport — データインポートプログラム」
セクション4.5.6「mysqlshow — データベース、テーブル、およびカラム情報の表示」
セクション4.5.7「mysqlslap — 負荷エミュレーションクライアント」
セクション5.3.3「Unix 上での複数の MySQL インスタンスの実行」
セクション2.3.5.4「サーバーをはじめて起動する」
セクション5.3.4「複数サーバー環境でのクライアントプログラムの使用」

Q

[索引の先頭]

-Q

セクション4.5.4「mysqldump — データベースバックアッププログラム」

-q

セクション4.6.3.3「myisamchk の修復オプション」
セクション4.5.1.1「mysql のオプション」
セクション4.5.3「mysqlcheck — テーブル保守プログラム」
セクション4.5.4「mysqldump — データベースバックアッププログラム」
セクション4.6.10「mysqlhotcopy — データベースバックアッププログラム」
セクション4.5.7「mysqlslap — 負荷エミュレーションクライアント」
セクション18.4.7「ndb_config — MySQL Cluster 構成情報の抽出」
セクション18.4.16「ndb_print_file — NDB ディスクデータファイル内容の出力」

--query

セクション4.5.7「mysqlslap — 負荷エミュレーションクライアント」
セクション18.4.7「ndb_config — MySQL Cluster 構成情報の抽出」
セクション18.4.14「ndb_index_stat — NDB インデックス統計ユーティリティー」

--query-cache-size

セクション8.10.5「外部ロック」

--quick

セクション4.6.3.3「myisamchk の修復オプション」
セクション4.6.3.6「myisamchk メモリー使用量」
セクション4.5.1「mysql — MySQL コマンド行ツール」
セクション4.5.1.1「mysql のオプション」
セクション4.5.3「mysqlcheck — テーブル保守プログラム」
セクション4.5.4「mysqldump — データベースバックアッププログラム」
セクション4.2.6「オプションファイルの使用」
セクション7.6.1「クラッシュリカバリへの myisamchk の使用」
セクションB.5.2.8「メモリー不足」

--quiet

セクション4.6.10「mysqlhotcopy — データベースバックアッププログラム」

--quote-names

セクション4.5.4「mysqldump — データベースバックアッププログラム」

R

[索引の先頭]

-R

セクション16.6.4.1「memcached の一般統計」
セクション16.6.2「memcached の使用」
セクション7.6.4「MyISAM テーブルの最適化」
セクション4.6.4「myisamlog — MyISAM ログファイルの内容の表示」
セクション4.6.8「mysqlbinlog — バイナリログファイルを処理するためのユーティリティー」
セクション4.5.4「mysqldump — データベースバックアッププログラム」
セクション18.4.20「ndb_restore — MySQL Cluster バックアップのリストア」
セクション4.6.3.4「その他の myisamchk オプション」
セクション4.6.8.3「バイナリログファイルのバックアップのための mysqlbinlog の使用」

-r

セクション16.6.2「memcached の使用」
セクション7.6.3「MyISAM テーブルの修復方法」
セクション4.6.3.2「myisamchk のチェックオプション」
セクション4.6.3.3「myisamchk の修復オプション」
セクション4.6.4「myisamlog — MyISAM ログファイルの内容の表示」
セクション4.5.1.1「mysql のオプション」
セクション4.5.2「mysqladmin — MySQL サーバーの管理を行うクライアント」
セクション4.6.8「mysqlbinlog — バイナリログファイルを処理するためのユーティリティー」
セクション4.5.3「mysqlcheck — テーブル保守プログラム」
セクション4.5.4「mysqldump — データベースバックアッププログラム」
セクション4.6.9「mysqldumpslow — スロークエリーログファイルの要約」
セクション4.5.5「mysqlimport — データインポートプログラム」
セクション18.4.7「ndb_config — MySQL Cluster 構成情報の抽出」
セクション18.4.10「ndb_desc — NDB テーブルの表示」
セクション5.1.3「サーバーコマンドオプション」
セクション2.2「一般的なバイナリを使用した MySQL の Unix/Linux へのインストール」
セクション24.3.2「新しいユーザー定義関数の追加」

--random-passwords

セクション1.4「MySQL 5.6 の新機能」
セクション4.4.3「mysql_install_db — MySQL データディレクトリの初期化」
セクション2.5.5「RPM パッケージを使用して MySQL を Linux にインストールする」
セクション2.10.1「Unix 類似システムでのインストール後の手順」

--raw

セクション1.4「MySQL 5.6 の新機能」
セクション4.5.1.1「mysql のオプション」
セクション4.6.8「mysqlbinlog — バイナリログファイルを処理するためのユーティリティー」
セクション4.6.8.3「バイナリログファイルのバックアップのための mysqlbinlog の使用」
セクション17.1.3.3「フェイルオーバーおよびスケールアウトでの GTID の使用」

--read-from-remote-master

セクション4.6.8「mysqlbinlog — バイナリログファイルを処理するためのユーティリティー」
セクション17.1.3.3「フェイルオーバーおよびスケールアウトでの GTID の使用」

--read-from-remote-server

セクション1.4「MySQL 5.6 の新機能」
セクション4.6.8「mysqlbinlog — バイナリログファイルを処理するためのユーティリティー」
セクション4.6.8.4「mysqlbinlog サーバー ID の指定」
セクション4.6.8.3「バイナリログファイルのバックアップのための mysqlbinlog の使用」
セクション17.1.3.3「フェイルオーバーおよびスケールアウトでの GTID の使用」

--read-only

セクション4.6.3.2「myisamchk のチェックオプション」
セクション17.1.4.3「レプリケーションスレーブのオプションと変数」

--rebuild-indexes

セクション18.3.2.6「MySQL Cluster データノードの定義」
セクション18.4.20「ndb_restore — MySQL Cluster バックアップのリストア」

--reconnect

セクション4.5.1.1「mysql のオプション」

--record_log_pos

セクション4.6.10「mysqlhotcopy — データベースバックアッププログラム」

--recover

セクション4.6.3.2「myisamchk のチェックオプション」
セクション4.6.3.1「myisamchk の一般オプション」
セクション4.6.3.3「myisamchk の修復オプション」
セクション4.6.3.6「myisamchk メモリー使用量」

--regexp

セクション4.6.12「mysql_find_rows — ファイルからの SQL ステートメントの抽出」
セクション4.6.10「mysqlhotcopy — データベースバックアッププログラム」

--relative

セクション4.5.2「mysqladmin — MySQL サーバーの管理を行うクライアント」

--relay-log

セクション13.4.2.1「CHANGE MASTER TO 構文」
セクション4.2.9「オプションのデフォルト、値を想定するオプション、および = 記号」
セクション17.2.2.1「スレーブリレーログ」
セクション17.1.4.3「レプリケーションスレーブのオプションと変数」
セクション17.3.5「レプリケーションパフォーマンスを改善する」
セクション17.1.1.9「既存のレプリケーション環境への追加スレーブの導入」

--relay-log-index

セクション17.2.2.1「スレーブリレーログ」
セクション17.1.4.3「レプリケーションスレーブのオプションと変数」
セクション17.1.1.9「既存のレプリケーション環境への追加スレーブの導入」

--relay-log-info-file

セクション17.2.2.2「スレーブステータスログ」
セクション17.1.4.3「レプリケーションスレーブのオプションと変数」

--relay-log-info-repository

セクション14.12「InnoDB の起動オプションおよびシステム変数」
セクション1.4「MySQL 5.6 の新機能」
セクション17.2.2.2「スレーブステータスログ」
セクション17.1.4.3「レプリケーションスレーブのオプションと変数」
セクション17.2.2「レプリケーションリレーおよびステータスログ」

--relay-log-purge

セクション17.1.4.3「レプリケーションスレーブのオプションと変数」

relay-log-purge

セクション17.1.4.3「レプリケーションスレーブのオプションと変数」

--relay-log-recovery

セクション13.7.5.35「SHOW SLAVE STATUS 構文」
セクション5.1.12「シャットダウンプロセス」
セクション17.1.4.3「レプリケーションスレーブのオプションと変数」
セクション17.2.2「レプリケーションリレーおよびステータスログ」

--relay-log-space-limit

セクション17.1.4.3「レプリケーションスレーブのオプションと変数」

--relnotes

セクション4.6.7「mysqlaccess — アクセス権限をチェックするクライアント」

--reload

セクション18.5.5「MySQL Cluster のローリング再起動の実行」
セクション18.3.2「MySQL Cluster の構成ファイル」
セクション18.5.13.2「MySQL Cluster データノードのオンライン追加: 基本的な手順」
セクション18.5.13.3「MySQL Cluster データノードのオンライン追加: 詳細な例」
セクション18.4.4「ndb_mgmd — MySQL Cluster 管理サーバーデーモン」
セクション18.2.3.3「Windows での MySQL Cluster の初期起動」
セクション18.1.6.10「複数の MySQL Cluster ノードに関する制限」

--remove

セクション18.4.4「ndb_mgmd — MySQL Cluster 管理サーバーデーモン」
セクション18.4.1「ndbd — MySQL Cluster データノードデーモン」
セクション2.3.5.7「Windows のサービスとして MySQL を起動する」
セクション18.2.3.4「Windows サービスとしての MySQL Cluster プロセスのインストール」
セクション5.3.2.2「Windows サービスとして複数の MySQL インスタンスの起動」
セクション5.1.3「サーバーコマンドオプション」

--repair

セクション4.5.3「mysqlcheck — テーブル保守プログラム」

--replace

セクション4.5.4「mysqldump — データベースバックアッププログラム」
セクション4.5.5「mysqlimport — データインポートプログラム」

--replicate-*

セクション17.2.3「サーバーがレプリケーションフィルタリングルールをどのように評価するか」
セクション17.1.4.3「レプリケーションスレーブのオプションと変数」
セクション17.2.3.3「レプリケーションルールの適用」

--replicate-*-db

セクションD.1「ストアドプログラムの制約」
セクション17.1.4.3「レプリケーションスレーブのオプションと変数」
セクション17.2.3.3「レプリケーションルールの適用」

--replicate-*-table

セクション17.2.3.3「レプリケーションルールの適用」

--replicate-do-*

セクション18.6.3「MySQL Cluster レプリケーションの既知の問題」

--replicate-do-db

セクション18.6.3「MySQL Cluster レプリケーションの既知の問題」
セクション13.7.5.35「SHOW SLAVE STATUS 構文」
セクション13.3.1「START TRANSACTION、COMMIT、および ROLLBACK 構文」
セクション17.2.3「サーバーがレプリケーションフィルタリングルールをどのように評価するか」
セクション17.2.3.1「データベースレベルレプリケーションオプションおよびバイナリロギングオプションの評価」
セクション5.2.4「バイナリログ」
セクション17.1.4.4「バイナリログのオプションと変数」
セクション17.4.1.22「レプリケーションと一時テーブル」
セクション17.4.1.25「レプリケーションと予約語」
セクション17.1.4.3「レプリケーションスレーブのオプションと変数」
セクション17.3.4「異なるデータベースを異なるスレーブに複製する」
セクション17.1.2.2「行ベースロギングおよびレプリケーションの使用」

--replicate-do-table

セクション15.6「BLACKHOLE ストレージエンジン」
セクション18.6.3「MySQL Cluster レプリケーションの既知の問題」
セクション13.7.5.35「SHOW SLAVE STATUS 構文」
セクション17.2.3.2「テーブルレベルレプリケーションオプションの評価」
セクション17.4.1.15「レプリケーションとシステム関数」
セクション17.4.1.22「レプリケーションと一時テーブル」
セクション17.4.1.25「レプリケーションと予約語」
セクション17.1.4.3「レプリケーションスレーブのオプションと変数」
セクション17.2.3.3「レプリケーションルールの適用」
セクション17.1.2.2「行ベースロギングおよびレプリケーションの使用」

--replicate-ignore-*

セクション18.6.3「MySQL Cluster レプリケーションの既知の問題」

--replicate-ignore-db

セクション18.6.3「MySQL Cluster レプリケーションの既知の問題」
セクション13.7.5.35「SHOW SLAVE STATUS 構文」
セクション13.3.1「START TRANSACTION、COMMIT、および ROLLBACK 構文」
セクション17.2.3「サーバーがレプリケーションフィルタリングルールをどのように評価するか」
セクション17.2.3.1「データベースレベルレプリケーションオプションおよびバイナリロギングオプションの評価」
セクション5.2.4「バイナリログ」
セクション17.1.4.4「バイナリログのオプションと変数」
セクション17.4.1.15「レプリケーションとシステム関数」
セクション17.4.1.25「レプリケーションと予約語」
セクション17.1.4.3「レプリケーションスレーブのオプションと変数」
セクション17.2.3.3「レプリケーションルールの適用」
セクション17.1.2.2「行ベースロギングおよびレプリケーションの使用」

--replicate-ignore-table

セクション15.6「BLACKHOLE ストレージエンジン」
セクション18.6.3「MySQL Cluster レプリケーションの既知の問題」
セクション13.7.5.35「SHOW SLAVE STATUS 構文」
セクション17.2.3.2「テーブルレベルレプリケーションオプションの評価」
セクション17.4.1.22「レプリケーションと一時テーブル」
セクション17.4.1.25「レプリケーションと予約語」
セクション17.1.4.3「レプリケーションスレーブのオプションと変数」
セクション17.1.2.2「行ベースロギングおよびレプリケーションの使用」

--replicate-rewrite-db

セクション17.2.3「サーバーがレプリケーションフィルタリングルールをどのように評価するか」
セクション17.1.4.3「レプリケーションスレーブのオプションと変数」
セクション17.1.2.2「行ベースロギングおよびレプリケーションの使用」

--replicate-same-server-id

セクション13.4.2.1「CHANGE MASTER TO 構文」
セクション17.1.4「レプリケーションおよびバイナリロギングのオプションと変数」
セクション17.1.4.3「レプリケーションスレーブのオプションと変数」

--replicate-wild-do-table

セクション13.7.5.35「SHOW SLAVE STATUS 構文」
セクション17.2.3「サーバーがレプリケーションフィルタリングルールをどのように評価するか」
セクションD.1「ストアドプログラムの制約」
セクション17.2.3.2「テーブルレベルレプリケーションオプションの評価」
セクション17.4.1.22「レプリケーションと一時テーブル」
セクション17.1.4.3「レプリケーションスレーブのオプションと変数」
セクション17.3.4「異なるデータベースを異なるスレーブに複製する」

--replicate-wild-ignore-table

セクションA.13「MySQL 5.6 FAQ: レプリケーション」
セクション18.6.3「MySQL Cluster レプリケーションの既知の問題」
セクション13.7.5.35「SHOW SLAVE STATUS 構文」
セクション17.2.3.2「テーブルレベルレプリケーションオプションの評価」
セクション17.4.1.22「レプリケーションと一時テーブル」
セクション17.1.4.3「レプリケーションスレーブのオプションと変数」

replication-ignore-table

セクション17.4.1.35「レプリケーションとビュー」

--replication-rewrite-db

セクション17.1.4.3「レプリケーションスレーブのオプションと変数」

--report-host

セクション13.7.5.34「SHOW SLAVE HOSTS 構文」
セクション5.1.4「サーバーシステム変数」
セクション17.1.5.1「レプリケーションステータスの確認」
セクション17.1.4.3「レプリケーションスレーブのオプションと変数」

--report-password

セクション5.1.4「サーバーシステム変数」
セクション17.1.4.3「レプリケーションスレーブのオプションと変数」

--report-port

セクション13.7.5.34「SHOW SLAVE HOSTS 構文」
セクション5.1.4「サーバーシステム変数」
セクション17.1.4.3「レプリケーションスレーブのオプションと変数」

--report-user

セクション5.1.4「サーバーシステム変数」
セクション17.1.4.3「レプリケーションスレーブのオプションと変数」

--resetmaster

セクション4.6.10「mysqlhotcopy — データベースバックアッププログラム」

--resetslave

セクション4.6.10「mysqlhotcopy — データベースバックアッププログラム」

--restore-privilege-tables

セクション18.5.14「MySQL Cluster の配布された MySQL 権限」
セクション18.4.20「ndb_restore — MySQL Cluster バックアップのリストア」

--restore_data

セクション18.4.20「ndb_restore — MySQL Cluster バックアップのリストア」

--restore_epoch

セクション18.6.9「MySQL Cluster レプリケーションを使用した MySQL Cluster バックアップ」
セクション18.6.9.2「MySQL Cluster レプリケーションを使用したポイントインタイムリカバリ」
セクション18.4.20「ndb_restore — MySQL Cluster バックアップのリストア」

--restore_meta

セクション18.4.20「ndb_restore — MySQL Cluster バックアップのリストア」

--result-file

セクション4.6.8「mysqlbinlog — バイナリログファイルを処理するためのユーティリティー」
セクション4.5.4「mysqldump — データベースバックアッププログラム」
セクション4.6.8.3「バイナリログファイルのバックアップのための mysqlbinlog の使用」

--retries

セクション18.4.10「ndb_desc — NDB テーブルの表示」

--rewrite-database
セクション18.4.20「ndb_restore — MySQL Cluster バックアップのリストア」

--rhost
セクション4.6.7「mysqlaccess — アクセス権限をチェックするクライアント」

--rollback
セクション4.6.7「mysqlaccess — アクセス権限をチェックするクライアント」

--routines
セクション2.11.1.3「MySQL 5.5 から 5.6 へのアップグレード」
セクション4.5.4「mysqldump — データベースバックアッププログラム」
セクション7.4.5.3「ストアドプログラムのダンプ」
セクション7.4.5.4「テーブル定義と内容の個別のダンプ」
セクション4.6.8.3「バイナリログファイルのバックアップのための mysqlbinlog の使用」

--rowid
セクション18.4.21「ndb_select_all — NDB テーブルの行の出力」

--rows
セクション4.6.12「mysql_find_rows — ファイルからの SQL ステートメントの抽出」
セクション18.4.7「ndb_config — MySQL Cluster 構成情報の抽出」

--rpl-recovery-rank
セクション1.4「MySQL 5.6 の新機能」

--rpm
セクション4.4.3「mysql_install_db — MySQL データディレクトリの初期化」

S

[索引の先頭]

-S
セクション4.4.1「comp_err — MySQL エラーメッセージファイルのコンパイル」
セクション7.6.4「MyISAM テーブルの最適化」
セクション4.5.1.1「mysql のオプション」
セクション4.5.1.2「mysql コマンド」
セクション4.2.2「MySQL サーバーへの接続」
セクション4.2.1「MySQL プログラムの起動」
セクション4.6.6「mysql_config_editor — MySQL 構成ユーティリティー」
セクション4.4.7「mysql_upgrade — MySQL テーブルのチェックとアップグレード」
セクション4.5.2「mysqladmin — MySQL サーバーの管理を行うクライアント」
セクション4.6.8「mysqlbinlog — バイナリログファイルを処理するためのユーティリティー」
セクション4.5.3「mysqlcheck — テーブル保守プログラム」
セクション4.5.4「mysqldump — データベースバックアッププログラム」
セクション4.6.10「mysqlhotcopy — データベースバックアッププログラム」
セクション4.5.5「mysqlimport — データインポートプログラム」
セクション4.5.6「mysqlshow — データベース、テーブル、およびカラム情報の表示」
セクション4.5.7「mysqlslap — 負荷エミュレーションクライアント」
セクション18.4.23「ndb_setup.py — MySQL Cluster のブラウザベース自動インストーラの開始」
セクション4.6.3.4「その他の myisamchk オプション」

-s
セクション4.6.1「innochecksum — オフライン InnoDB ファイルチェックサムユーティリティー」
セクション16.6.2「memcached の使用」
セクション7.6.2「MyISAM テーブルのエラーのチェック方法」
セクション7.6.3「MyISAM テーブルの修復方法」
セクション7.6.5「MyISAM テーブル保守スケジュールのセットアップ」
セクション4.6.2「myisam_ftdump — 全文インデックス情報の表示」
セクション4.6.3.1「myisamchk の一般オプション」
セクション4.6.5「myisampack — 圧縮された読み取り専用の MyISAM テーブルの生成」
セクション4.5.1.1「mysql のオプション」
セクション4.4.7「mysql_upgrade — MySQL テーブルのチェックとアップグレード」
セクション4.6.16「mysql_zap — パターンに一致するプロセスを強制終了」
セクション4.5.2「mysqladmin — MySQL サーバーの管理を行うクライアント」
セクション4.6.8「mysqlbinlog — バイナリログファイルを処理するためのユーティリティー」
セクション4.5.3「mysqlcheck — テーブル保守プログラム」
セクション4.6.9「mysqldumpslow — スロークエリーログファイルの要約」
セクション4.5.5「mysqlimport — データインポートプログラム」
セクション4.5.7「mysqlslap — 負荷エミュレーションクライアント」
セクション18.4.20「ndb_restore — MySQL Cluster バックアップのリストア」
セクション18.4.23「ndb_setup.py — MySQL Cluster のブラウザベース自動インストーラの開始」
セクション4.8.1「perror — エラーコードの説明」
セクション4.8.2「replace — 文字列置換ユーティリティー」
セクション4.7.4「resolve_stack_dump — 数値スタックトレースダンプをシンボルに解決」
セクション4.8.3「resolveip — ホスト名と IP アドレスの解決」
セクション5.1.3「サーバーコマンドオプション」

--safe-mode
セクション1.4「MySQL 5.6 の新機能」
セクション5.1.3「サーバーコマンドオプション」

--safe-recover
セクション4.6.3.1「myisamchk の一般オプション」
セクション4.6.3.3「myisamchk の修復オプション」

セクション4.6.3.6「myisamchk メモリー使用量」

--safe-updates

--safe-updates オプションの使用
セクション4.5.1.1「mysql のオプション」
セクション4.5.1.2「mysql コマンド」

--safe-user-create

セクション5.1.3「サーバーコマンドオプション」

--savequeries

セクション18.4.25「ndb_size.pl — NDBCLUSTER サイズ要件エスティメータ」

--secure-auth

セクション6.3.8.3「4.1 よりも前のパスワードハッシュ方式と mysql_old_password プラグインからの移行」
セクション6.1.2.4「MySQL でのパスワードハッシュ」
セクション4.5.1.1「mysql のオプション」
セクション4.5.2「mysqladmin — MySQL サーバーの管理を行うクライアント」
セクション4.6.8「mysqlbinlog — バイナリログファイルを処理するためのユーティリティー」
セクション4.5.3「mysqlcheck — テーブル保守プログラム」
セクション4.5.4「mysqldump — データベースバックアッププログラム」
セクション4.5.5「mysqlimport — データインポートプログラム」
セクション4.5.6「mysqlshow — データベース、テーブル、およびカラム情報の表示」
セクション4.5.7「mysqlslap — 負荷エミュレーションクライアント」
セクション5.1.3「サーバーコマンドオプション」

--secure-file-priv

セクション2.10.1「Unix 類似システムでのインストール後の手順」
セクション5.1.3「サーバーコマンドオプション」
セクション5.1.4「サーバーシステム変数」
セクション6.1.3「攻撃者に対する MySQL のセキュアな状態の維持」

--select_limit

--safe-updates オプションの使用

--server-id

セクション18.6.2「MySQL Cluster レプリケーションの一般要件」
セクション4.6.8「mysqlbinlog — バイナリログファイルを処理するためのユーティリティー」
セクション13.7.5.34「SHOW SLAVE HOSTS 構文」
セクション5.1.4「サーバーシステム変数」
セクション17.1.4「レプリケーションおよびバイナリロギングのオプションと変数」
セクション17.4.4「レプリケーションのトラブルシューティング」

server-id

セクション17.1.4.3「レプリケーションスレーブのオプションと変数」
セクション17.1.1.2「レプリケーションスレーブ構成の設定」
セクション17.1.4.2「レプリケーションマスターのオプションと変数」
セクション17.1.1.1「レプリケーションマスター構成の設定」
セクション17.1「レプリケーション構成」
セクション17.1.1.8「既存のデータによるレプリケーションのセットアップ」
セクション17.1.1.9「既存のレプリケーション環境への追加スレーブの導入」

--server-id-bits

セクション18.3.4.3「MySQL Cluster のシステム変数」
セクション18.3.4.2「MySQL Cluster 用の MySQL Server オプション」
セクション4.6.8「mysqlbinlog — バイナリログファイルを処理するためのユーティリティー」

--server-log-file

セクション18.4.23「ndb_setup.py — MySQL Cluster のブラウザベース自動インストーラの開始」

--server-name

セクション18.4.23「ndb_setup.py — MySQL Cluster のブラウザベース自動インストーラの開始」

--server-public-key

セクション4.5.1.1「mysql のオプション」
セクション6.3.8.4「SHA-256 認証プラグイン」

--server-public-key-path

セクション4.5.1.1「mysql のオプション」
セクション6.3.8.4「SHA-256 認証プラグイン」

--service-startup-timeout

セクション4.3.3「mysql.server — MySQL サーバー起動スクリプト」

--set-auto-increment

セクション4.6.3.4「その他の myisamchk オプション」

--set-character-set

セクション4.6.3.3「myisamchk の修復オプション」

--set-charset

セクション4.6.8「mysqlbinlog — バイナリログファイルを処理するためのユーティリティー」
セクション4.5.4「mysqldump — データベースバックアッププログラム」

--set-collation

セクション4.6.3.3「myisamchk の修復オプション」

--set-gtid-purged

セクション4.5.4「mysqldump — データベースバックアッププログラム」
セクション17.1.3.3「フェイルオーバーおよびスケールアウトでの GTID の使用」

--shared-memory

セクション4.5.1.1「mysql のオプション」

セクション1.3.2「MySQL の主な機能」
セクション4.2.2「MySQL サーバーへの接続」
セクション4.4.7「mysql_upgrade — MySQL テーブルのチェックとアップグレード」
セクション4.5.2「mysqladmin — MySQL サーバーの管理を行うクライアント」
セクション4.6.8「mysqlbinlog — バイナリログファイルを処理するためのユーティリティー」
セクション4.5.3「mysqlcheck — テーブル保守プログラム」
セクション4.5.4「mysqldump — データベースバックアッププログラム」
セクション4.5.5「mysqlimport — データインポートプログラム」
セクション4.5.6「mysqlshow — データベース、テーブル、およびカラム情報の表示」
セクション5.3.2.1「Windows コマンド行での複数の MySQL インスタンスの起動」
セクション2.3.5.4「サーバーをはじめて起動する」
セクション5.1.3「サーバーコマンドオプション」

--shared-memory-base-name

セクション5.3「1 つのマシン上での複数の MySQL インスタンスの実行」
セクション4.5.1.1「mysql のオプション」
セクション4.2.2「MySQL サーバーへの接続」
セクション23.7.7.49「mysql_options()」
セクション4.4.7「mysql_upgrade — MySQL テーブルのチェックとアップグレード」
セクション4.5.2「mysqladmin — MySQL サーバーの管理を行うクライアント」
セクション4.6.8「mysqlbinlog — バイナリログファイルを処理するためのユーティリティー」
セクション4.5.3「mysqlcheck — テーブル保守プログラム」
セクション4.5.4「mysqldump — データベースバックアッププログラム」
セクション4.5.5「mysqlimport — データインポートプログラム」
セクション4.5.6「mysqlshow — データベース、テーブル、およびカラム情報の表示」
セクション4.5.7「mysqlslap — 負荷エミュレーションクライアント」
セクション5.3.2.1「Windows コマンド行での複数の MySQL インスタンスの起動」
セクション5.1.3「サーバーコマンドオプション」
セクション5.3.4「複数サーバー環境でのクライアントプログラムの使用」

--short-form

セクション4.6.8「mysqlbinlog — バイナリログファイルを処理するためのユーティリティー」

--show-slave-auth-info

セクション17.1.4.3「レプリケーションスレーブのオプションと変数」

--show-table-type

セクション4.5.6「mysqlshow — データベース、テーブル、およびカラム情報の表示」

--show-temp-status

セクション18.4.24「ndb_show_tables — NDB テーブルのリストの表示」

--show-warnings

セクション4.5.1.1「mysql のオプション」

--sigint-ignore

セクション4.5.1.1「mysql のオプション」

--silent

セクション7.6.5「MyISAM テーブル保守スケジュールのセットアップ」
セクション4.6.3.1「myisamchk の一般オプション」
セクション4.6.5「myisampack — 圧縮された読み取り専用の MyISAM テーブルの生成」
セクション4.5.1.1「mysql のオプション」
セクション4.5.2「mysqladmin — MySQL サーバーの管理を行うクライアント」
セクション4.5.3「mysqlcheck — テーブル保守プログラム」
セクション4.3.4「mysqld_multi — 複数の MySQL サーバーの管理」
セクション4.5.5「mysqlimport — データインポートプログラム」
セクション4.5.7「mysqlslap — 負荷エミュレーションクライアント」
セクション4.8.1「perror — エラーコードの説明」
セクション4.8.3「resolveip — ホスト名と IP アドレスの解決」

--single-transaction

セクション14.16「InnoDB のバックアップとリカバリ」
セクション4.5.4「mysqldump — データベースバックアッププログラム」
セクション7.2「データベースバックアップ方法」
セクション7.3.1「バックアップポリシーの確立」

--single-user

セクション18.4.26「ndb_waiter — MySQL Cluster が指定したステータスになるまで待機する」

--skip

セクション4.5.4「mysqldump — データベースバックアッププログラム」
セクション5.1.3「サーバーコマンドオプション」
セクション4.2.5「プログラムオプション修飾子」

--skip-add-drop-table

セクション4.5.4「mysqldump — データベースバックアッププログラム」

--skip-add-locks

セクション4.5.4「mysqldump — データベースバックアッププログラム」

--skip-auto-rehash

セクション14.19.3「InnoDB データディクショナリの操作のトラブルシューティング」
セクション4.5.1.1「mysql のオプション」

--skip-broken-objects

セクション18.4.20「ndb_restore — MySQL Cluster バックアップのリストア」

--skip-character-set-client-handshake
cp932 文字セット
セクションA.11「MySQL 5.6 FAQ: MySQL の中国語、日本語、および韓国語の文字セット」
セクション5.1.3「サーバーコマンドオプション」
セクション5.1.4「サーバーシステム変数」

--skip-column-names
セクション4.5.1.1「mysql のオプション」

--skip-comments
セクション4.5.4「mysqldump — データベースバックアッププログラム」

--skip-concurrent-insert
セクション5.1.3「サーバーコマンドオプション」

--skip-config-cache
セクション18.4.4「ndb_mgmd — MySQL Cluster 管理サーバーデーモン」

--skip-database
セクション4.5.3「mysqlcheck — テーブル保守プログラム」

--skip-disable-keys
セクション4.5.4「mysqldump — データベースバックアッププログラム」

--skip-dump-date
セクション4.5.4「mysqldump — データベースバックアッププログラム」

--skip-engine_name
セクション13.7.5.17「SHOW ENGINES 構文」

--skip-event-scheduler
セクション5.1.3「サーバーコマンドオプション」

--skip-events
セクション7.4.5.3「ストアドプログラムのダンプ」

--skip-extended-insert
セクション4.5.4「mysqldump — データベースバックアッププログラム」

--skip-external-locking
セクションB.5.4.2「MySQL が繰り返しクラッシュする場合の対処方法」
セクション5.1.3「サーバーコマンドオプション」
セクション5.1.4「サーバーシステム変数」
セクション8.11.1「システム要素およびスタートアップパラメータのチューニング」
セクション8.12.5.2「一般的なスレッドの状態」
セクション8.10.5「外部ロック」

--skip-federated
セクション17.3.2「異なるマスターおよびスレーブストレージエンジンでレプリケーションを使用する」

--skip-grant-tables
セクション13.7.3.3「INSTALL PLUGIN 構文」
セクション18.5.14「MySQL Cluster の配布された MySQL 権限」
セクション2.10.1.1「mysql_install_db 実行の問題」
セクション4.5.2「mysqladmin — MySQL サーバーの管理を行うクライアント」
root のパスワードのリセット: 一般的な手順
セクション6.2.7「アクセス拒否エラーの原因」
セクション20.4.2「イベントスケジューラの構成」
セクション4.2.4「コマンド行でのオプションの使用」
セクション5.1.3「サーバーコマンドオプション」
セクション6.3.7「プラガブル認証」
セクション5.1.8.1「プラグインのインストールおよびアンインストール」
セクション13.7.3.1「ユーザー定義関数のための CREATE FUNCTION 構文」
セクション24.3.2.5「ユーザー定義関数のコンパイルおよびインストール」
セクション6.2.6「権限変更が有効化される時期」

--skip-gtids
セクション4.6.8「mysqlbinlog — バイナリログファイルを処理するためのユーティリティー」
セクション7.5「バイナリログを使用したポイントインタイム (増分) リカバリ」

--skip-host-cache
セクション6.2.7「アクセス拒否エラーの原因」
セクション5.1.3「サーバーコマンドオプション」
セクション5.1.4「サーバーシステム変数」

--skip-innodb
セクション14.1.3「InnoDB の無効化」
セクション14.12「InnoDB の起動オプションおよびシステム変数」
セクションA.13「MySQL 5.6 FAQ: レプリケーション」
セクション1.4「MySQL 5.6 の新機能」
セクション5.1.3「サーバーコマンドオプション」
セクション5.1.4「サーバーシステム変数」
セクション5.1.8.1「プラグインのインストールおよびアンインストール」

--skip-innodb-checksums
セクション14.12「InnoDB の起動オプションおよびシステム変数」

--skip-innodb_adaptive_hash_index
セクション14.12「InnoDB の起動オプションおよびシステム変数」

--skip-innodb_doublewrite
セクション14.12「InnoDB の起動オプションおよびシステム変数」

--skip-kill-mysqld
セクション4.3.2「mysqld_safe — MySQL サーバー起動スクリプト」

--skip-line-numbers
セクション4.5.1.1「mysql のオプション」

--skip-lock-tables
セクション4.5.3「mysqlcheck — テーブル保守プログラム」
セクション4.5.4「mysqldump — データベースバックアッププログラム」

--skip-name-resolve
セクション8.11.5.2「DNS ルックアップの最適化とホストキャッシュ」
セクション2.3.5.8「MySQL インストールのテスト」
セクション4.4.3「mysql_install_db — MySQL データディレクトリの初期化」
セクション6.2.7「アクセス拒否エラーの原因」
セクション5.1.3「サーバーコマンドオプション」
セクション5.1.4「サーバーシステム変数」
セクション2.10.2「最初の MySQL アカウントのセキュリティー設定」

--skip-named-commands
セクション4.5.1.1「mysql のオプション」

--skip-ndbcluster
セクション18.3.4.1「MySQL Cluster の mysqld オプションおよび変数のリファレンス」
セクション18.3.4.3「MySQL Cluster のシステム変数」
セクション18.3.4.2「MySQL Cluster 用の MySQL Server オプション」

--skip-networking
セクションB.5.2.2「[ローカルの] MySQL サーバーに接続できません」
セクション8.11.5.2「DNS ルックアップの最適化とホストキャッシュ」
セクションB.5.2.9「MySQL サーバーが存在しなくなりました」
root のパスワードのリセット: 一般的な手順
セクション6.2.7「アクセス拒否エラーの原因」
セクション5.1.3「サーバーコマンドオプション」
セクション5.1.4「サーバーシステム変数」
セクション6.3.7「プラガブル認証」
セクション17.4.3「レプリケーションセットアップをアップグレードする」

skip-networking
セクションA.13「MySQL 5.6 FAQ: レプリケーション」
セクション17.4.4「レプリケーションのトラブルシューティング」
セクション17.1.1.1「レプリケーションマスター構成の設定」

--skip-new
セクション24.4.1「MySQL サーバーのデバッグ」
セクション13.7.2.4「OPTIMIZE TABLE 構文」
セクション5.1.4「サーバーシステム変数」

--skip-nodegroup
セクション18.4.13「ndb_error_reporter — NDB エラーレポートユーティリティー」

--skip-opt
セクション4.5.4「mysqldump — データベースバックアッププログラム」

--skip-pager
セクション4.5.1.1「mysql のオプション」

--skip-partition
セクション5.1.3「サーバーコマンドオプション」
第19章「パーティション化」

--skip-plugin-innodb_file_per_table
セクション5.1.3「サーバーコマンドオプション」

--skip-plugin_name
セクション5.1.8.1「プラグインのインストールおよびアンインストール」

--skip-quick
セクション4.5.4「mysqldump — データベースバックアッププログラム」

--skip-quote-names
セクション4.5.4「mysqldump — データベースバックアッププログラム」

--skip-reconnect
セクション4.5.1.1「mysql のオプション」
mysql の自動再接続を無効にする
セクション23.7.16「自動再接続動作の制御」

--skip-routines
セクション7.4.5.3「ストアドプログラムのダンプ」

--skip-secure-auth
セクション4.5.1.1「mysql のオプション」
セクション4.5.2「mysqladmin — MySQL サーバーの管理を行うクライアント」
セクション4.6.8「mysqlbinlog — バイナリログファイルを処理するためのユーティリティー」
セクション4.5.3「mysqlcheck — テーブル保守プログラム」
セクション4.5.4「mysqldump — データベースバックアッププログラム」
セクション4.5.5「mysqlimport — データインポートプログラム」
セクション4.5.6「mysqlshow — データベース、テーブル、およびカラム情報の表示」
セクション4.5.7「mysqlslap — 負荷エミュレーションクライアント」
セクション5.1.3「サーバーコマンドオプション」

--skip-set-charset
セクション4.5.4「mysqldump — データベースバックアッププログラム」

--skip-show-database
セクション6.2.1「MySQL で提供される権限」
セクション1.8.5「MySQL のサポータ」
セクション13.7.5.15「SHOW DATABASES 構文」
セクション5.1.3「サーバーコマンドオプション」

--skip-slave-start
セクション13.4.2.1「CHANGE MASTER TO 構文」

セクション17.1.3.2「GTID を使用したレプリケーションのセットアップ」
セクション18.6.6「MySQL Cluster レプリケーションの起動 (レプリケーションチャネルが 1 つ)」
セクション18.6.9「MySQL Cluster レプリケーションを使用した MySQL Cluster バックアップ」
セクション17.3.7「SSL を使用してレプリケーションをセットアップする」
セクション13.4.2.5「START SLAVE 構文」
セクション18.6.5「レプリケーションのための MySQL Cluster の準備」
セクション17.4.4「レプリケーションのトラブルシューティング」
セクション17.1.4.3「レプリケーションスレーブのオプションと変数」
セクション17.4.3「レプリケーションセットアップをアップグレードする」
セクション17.1.1.8「既存のデータによるレプリケーションのセットアップ」

--skip-ssl

セクション6.3.10.4「SSL コマンドのオプション」
セクション6.3.10.3「SSL 接続の使用」

--skip-stack-trace

セクション24.4.1.4「gdb での mysqld のデバッグ」
セクション5.1.3「サーバーコマンドオプション」

--skip-super-large-pages

セクション5.1.3「サーバーコマンドオプション」
セクション8.11.4.2「ラージページのサポートの有効化」

--skip-symbolic-links

セクション13.1.17「CREATE TABLE 構文」
Unix 上の MyISAM へのシンボリックリンクの使用
Windows 上のデータベースへのシンボリックリンクの使用
セクション5.1.3「サーバーコマンドオプション」
セクション5.1.4「サーバーシステム変数」
セクション6.1.3「攻撃者に対する MySQL のセキュアな状態の維持」

--skip-syslog

セクション4.3.2「mysqld_safe — MySQL サーバー起動スクリプト」
セクション5.2.2「エラーログ」

--skip-table-check

セクション18.4.20「ndb_restore — MySQL Cluster バックアップのリストア」

--skip-thread-priority

セクション1.4「MySQL 5.6 の新機能」
セクション5.1.3「サーバーコマンドオプション」

--skip-triggers

セクション4.5.4「mysqldump — データベースバックアッププログラム」
セクション7.4.5.3「ストアドプログラムのダンプ」

--skip-tz-utc

セクション4.5.4「mysqldump — データベースバックアッププログラム」

--skip-unknown-objects

セクション18.4.20「ndb_restore — MySQL Cluster バックアップのリストア」

--skip-use-db

セクション4.6.12「mysql_find_rows — ファイルからの SQL ステートメントの抽出」

--skip-version-check

セクション4.4.7「mysql_upgrade — MySQL テーブルのチェックとアップグレード」

--skip-warn

セクション4.6.6「mysql_config_editor — MySQL 構成ユーティリティー」

--skip-write-binlog

セクション4.4.7「mysql_upgrade — MySQL テーブルのチェックとアップグレード」
セクション4.5.3「mysqlcheck — テーブル保守プログラム」
セクション11.3.4「YEAR(2) の制限と YEAR(4) への移行」

--skip_grant_tables

セクション4.2.4「コマンド行でのオプションの使用」

--slave-checkpoint-group

セクション17.1.4.3「レプリケーションスレーブのオプションと変数」

--slave-checkpoint-period

セクション17.1.4.3「レプリケーションスレーブのオプションと変数」

--slave-load-tmpdir

セクションB.5.4.4「MySQL が一時ファイルを格納する場所」
セクション17.3.1.2「スレーブからローデータをバックアップする」
セクション7.2「データベースバックアップ方法」
セクション17.1.4.3「レプリケーションスレーブのオプションと変数」

--slave-max-allowed-packet

セクション17.1.4.3「レプリケーションスレーブのオプションと変数」

slave-max-allowed-packet

セクション17.1.4.3「レプリケーションスレーブのオプションと変数」

--slave-net-timeout

セクション17.1.4.3「レプリケーションスレーブのオプションと変数」

--slave-parallel-workers

セクション17.1.4.3「レプリケーションスレーブのオプションと変数」

--slave-pending-jobs-size-max
セクション17.1.4.3「レプリケーションスレーブのオプションと変数」

slave-rows-search-algorithms
セクション17.1.4.3「レプリケーションスレーブのオプションと変数」

--slave-skip-errors
セクション18.6.8「MySQL Cluster レプリケーションを使用したフェイルオーバーの実装」
セクション17.1.4.3「レプリケーションスレーブのオプションと変数」
セクション17.4.1.26「レプリケーション中のスレーブエラー」

--slave-sql-verify-checksum
セクション1.4「MySQL 5.6 の新機能」
セクション17.1.4.4「バイナリログのオプションと変数」
セクション17.1.4.3「レプリケーションスレーブのオプションと変数」

--slave_compressed_protocol
セクション17.1.4.3「レプリケーションスレーブのオプションと変数」

--sleep
セクション4.5.2「mysqladmin — MySQL サーバーの管理を行うクライアント」

--slow-query-log
セクション5.1.3「サーバーコマンドオプション」

--slow-start-timeout
セクション5.1.3「サーバーコマンドオプション」

--slow_query_log
セクション1.4「MySQL 5.6 の新機能」
セクション5.1.3「サーバーコマンドオプション」
セクション5.1.4「サーバーシステム変数」
セクション5.2.5「スロークエリーログ」
セクション5.2.1「一般クエリーログおよびスロークエリーログの出力先の選択」

--slow_query_log_file
セクション5.3「1 つのマシン上での複数の MySQL インスタンスの実行」
セクション1.4「MySQL 5.6 の新機能」
セクション5.1.3「サーバーコマンドオプション」
セクション5.2.5「スロークエリーログ」

--socket
セクション5.3「1 つのマシン上での複数の MySQL インスタンスの実行」
セクションB.5.2.2「[ローカルの] MySQL サーバーに接続できません」
セクションB.5.4.5「MySQL の UNIX ソケットファイルを保護または変更する方法」
セクション4.5.1.1「mysql のオプション」
セクション2.3.5.8「MySQL インストールのテスト」
セクション4.2.2「MySQL サーバーへの接続」
セクション2.9.4「MySQL ソース構成オプション」
セクション4.2.1「MySQL プログラムの起動」
セクション4.7.2「mysql_config — クライアントのコンパイル用オプションの表示」
セクション4.6.6「mysql_config_editor — MySQL 構成ユーティリティー」
セクション4.6.11「mysql_convert_table_format — 指定されたストレージエンジンを使用するためのテーブルの変換」
セクション4.6.14「mysql_setpermission — 付与テーブルに許可をインタラクティブに設定」
セクション4.4.7「mysql_upgrade — MySQL テーブルのチェックとアップグレード」
セクション4.5.2「mysqladmin — MySQL サーバーの管理を行うクライアント」
セクション4.6.8「mysqlbinlog — バイナリログファイルを処理するためのユーティリティー」
セクション4.5.3「mysqlcheck — テーブル保守プログラム」
セクション4.3.2「mysqld_safe — MySQL サーバー起動スクリプト」
セクション4.5.4「mysqldump — データベースバックアッププログラム」
セクション4.6.10「mysqlhotcopy — データベースバックアッププログラム」
セクション4.5.5「mysqlimport — データインポートプログラム」
セクション4.5.6「mysqlshow — データベース、テーブル、およびカラム情報の表示」
セクション4.5.7「mysqlslap — 負荷エミュレーションクライアント」
セクション5.3.3「Unix 上での複数の MySQL インスタンスの実行」
セクション6.2.7「アクセス拒否エラーの原因」
セクション5.1.3「サーバーコマンドオプション」
サーバープラグインライブラリおよびプラグインディスクリプタ
セクション5.3.4「複数サーバー環境でのクライアントプログラムの使用」

--sort-index
セクション7.6.4「MyISAM テーブルの最適化」
セクション4.6.3.4「その他の myisamchk オプション」

--sort-records
セクション7.6.4「MyISAM テーブルの最適化」
セクション4.6.3.4「その他の myisamchk オプション」

--sort-recover
セクション4.6.3.1「myisamchk の一般オプション」
セクション4.6.3.3「myisamchk の修復オプション」
セクション4.6.3.6「myisamchk メモリー使用量」

--spassword
セクション4.6.7「mysqlaccess — アクセス権限をチェックするクライアント」

--sporadic-binlog-dump-fail
セクション17.1.4.4「バイナリログのオプションと変数」

--sql-mode
セクションA.3「MySQL 5.6 FAQ: サーバー SQL モード」
セクション5.1.7「サーバー SQL モード」
セクション5.1.3「サーバーコマンドオプション」
第12章「関数と演算子」

sql-mode
セクション5.1.7「サーバー SQL モード」

--srcdir
セクション4.4.3「mysql_install_db — MySQL データディレクトリの初期化」

--ssl
セクション4.5.1.1「mysql のオプション」
セクション4.2.2「MySQL サーバーへの接続」
セクション4.4.7「mysql_upgrade — MySQL テーブルのチェックとアップグレード」
セクション4.5.2「mysqladmin — MySQL サーバーの管理を行うクライアント」
セクション4.5.3「mysqlcheck — テーブル保守プログラム」
セクション4.5.4「mysqldump — データベースバックアッププログラム」
セクション4.5.5「mysqlimport — データインポートプログラム」
セクション4.5.6「mysqlshow — データベース、テーブル、およびカラム情報の表示」
セクション4.5.7「mysqlslap — 負荷エミュレーションクライアント」
セクション6.3.10.4「SSL コマンドのオプション」
セクション6.3.10.3「SSL 接続の使用」
セクション5.1.3「サーバーコマンドオプション」

--ssl*
セクション4.5.1.1「mysql のオプション」
セクション4.2.2「MySQL サーバーへの接続」
セクション4.4.7「mysql_upgrade — MySQL テーブルのチェックとアップグレード」
セクション4.5.2「mysqladmin — MySQL サーバーの管理を行うクライアント」
セクション4.5.3「mysqlcheck — テーブル保守プログラム」
セクション4.5.4「mysqldump — データベースバックアッププログラム」
セクション4.5.5「mysqlimport — データインポートプログラム」
セクション4.5.6「mysqlshow — データベース、テーブル、およびカラム情報の表示」
セクション4.5.7「mysqlslap — 負荷エミュレーションクライアント」
セクション5.1.3「サーバーコマンドオプション」

--ssl-ca
セクション13.7.1.4「GRANT 構文」
セクション6.3.10.5「MySQL での SSL 証明書および鍵の設定」
セクション6.3.10.4「SSL コマンドのオプション」
セクション6.3.10.3「SSL 接続の使用」

--ssl-capath
セクション13.7.1.4「GRANT 構文」
セクション6.3.10.4「SSL コマンドのオプション」

--ssl-cert
セクション13.7.1.4「GRANT 構文」
セクション6.3.10.5「MySQL での SSL 証明書および鍵の設定」
セクション6.3.10.4「SSL コマンドのオプション」
セクション6.3.10.3「SSL 接続の使用」

--ssl-cipher
セクション6.3.10.4「SSL コマンドのオプション」

--ssl-crl
セクション6.3.10.4「SSL コマンドのオプション」

--ssl-crlpath
セクション6.3.10.4「SSL コマンドのオプション」

--ssl-key
セクション13.7.1.4「GRANT 構文」
セクション6.3.10.5「MySQL での SSL 証明書および鍵の設定」
セクション6.3.10.4「SSL コマンドのオプション」
セクション6.3.10.3「SSL 接続の使用」

--ssl-verify-server-cert
セクション6.3.10.4「SSL コマンドのオプション」

--ssl-xxx
セクション13.4.2.1「CHANGE MASTER TO 構文」
セクション6.3.10.2「SSL を使用するための MySQL の構成」
セクション6.3.10.4「SSL コマンドのオプション」
セクション5.1.4「サーバーシステム変数」

--standalone
セクション2.3.5.5「Windows のコマンド行からの MySQL の起動」
セクション5.1.3「サーバーコマンドオプション」
セクション24.4.1.2「トレースファイルの作成」

--start-datetime
セクション4.6.8「mysqlbinlog — バイナリログファイルを処理するためのユーティリティー」
セクション7.5.1「イベント時間を使用したポイントインタイムリカバリ」

--start-position
セクション4.6.8「mysqlbinlog — バイナリログファイルを処理するためのユーティリティー」
セクション7.5.2「イベントの位置を使用したポイントインタイムリカバリ」

--start_row
セクション4.6.12「mysql_find_rows — ファイルからの SQL ステートメントの抽出」

--statefile
セクション4.4.1「comp_err — MySQL エラーメッセージファイルのコンパイル」

--stats
セクション4.6.2「myisam_ftdump — 全文インデックス情報の表示」

--status
セクション4.5.6「mysqlshow — データベース、テーブル、およびカラム情報の表示」

--stop-datetime
セクション4.6.8「mysqlbinlog — バイナリログファイルを処理するためのユーティリティー」
セクション7.5.1「イベント時間を使用したポイントインタイムリカバリ」

--stop-never
セクション4.6.8「mysqlbinlog — バイナリログファイルを処理するためのユーティリティー」
セクション4.6.8.4「mysqlbinlog サーバー ID の指定」
セクション4.6.8.3「バイナリログファイルのバックアップのための mysqlbinlog の使用」

--stop-never-slave-server-id
セクション4.6.8「mysqlbinlog — バイナリログファイルを処理するためのユーティリティー」
セクション4.6.8.4「mysqlbinlog サーバー ID の指定」
セクション4.6.8.3「バイナリログファイルのバックアップのための mysqlbinlog の使用」

--stop-position
セクション4.6.8「mysqlbinlog — バイナリログファイルを処理するためのユーティリティー」
セクション7.5.2「イベントの位置を使用したポイントインタイムリカバリ」

--suffix
セクション4.6.10「mysqlhotcopy — データベースバックアッププログラム」

--super-large-pages
セクション5.1.3「サーバーコマンドオプション」
セクション8.11.4.2「ラージページのサポートの有効化」

--superuser
セクション4.6.7「mysqlaccess — アクセス権限をチェックするクライアント」

--symbolic-links
セクション5.1.3「サーバーコマンドオプション」

--symbols-file
セクション4.7.4「resolve_stack_dump — 数値スタックトレースダンプをシンボルに解決」

--sys-*
セクション18.4.14「ndb_index_stat — NDB インデックス統計ユーティリティー」

--sys-check
セクション18.4.14「ndb_index_stat — NDB インデックス統計ユーティリティー」

--sys-create
セクション18.4.14「ndb_index_stat — NDB インデックス統計ユーティリティー」

sys-create-if-not-exist
セクション18.4.14「ndb_index_stat — NDB インデックス統計ユーティリティー」

--sys-create-if-not-valid
セクション18.4.14「ndb_index_stat — NDB インデックス統計ユーティリティー」

--sys-drop
セクション18.4.14「ndb_index_stat — NDB インデックス統計ユーティリティー」

--sys-skip-events
セクション18.4.14「ndb_index_stat — NDB インデックス統計ユーティリティー」

--sys-skip-tables
セクション18.4.14「ndb_index_stat — NDB インデックス統計ユーティリティー」

SYSCONFDIR
セクション4.2.6「オプションファイルの使用」

--sysdate-is-now
セクション5.1.3「サーバーコマンドオプション」
セクション5.1.4「サーバーシステム変数」
セクション17.1.2.1「ステートメントベースおよび行ベースレプリケーションのメリットとデメリット」
セクション17.4.1.15「レプリケーションとシステム関数」
セクション12.7「日付および時間関数」

--syslog
セクション4.3.2「mysqld_safe — MySQL サーバー起動スクリプト」
セクション5.2.2「エラーログ」

--syslog-tag
セクション4.3.2「mysqld_safe — MySQL サーバー起動スクリプト」

--system
セクション18.4.7「ndb_config — MySQL Cluster 構成情報の抽出」

T
[索引の先頭]

-T
セクション4.4.1「comp_err — MySQL エラーメッセージファイルのコンパイル」
セクション4.6.3.2「myisamchk のチェックオプション」
セクション4.6.5「myisampack — 圧縮された読み取り専用の MyISAM テーブルの生成」
セクション4.5.1.1「mysql のオプション」
セクション4.4.7「mysql_upgrade — MySQL テーブルのチェックとアップグレード」
セクション4.5.4「mysqldump — データベースバックアッププログラム」
セクション4.5.7「mysqlslap — 負荷エミュレーションクライアント」
セクション18.4.20「ndb_restore — MySQL Cluster バックアップのリストア」
セクション5.1.3「サーバーコマンドオプション」

-t

セクション16.6.2「memcached の使用」
セクション4.6.3.3「myisamchk の修復オプション」
セクション4.6.5「myisampack — 圧縮された読み取り専用の MyISAM テーブルの生成」
セクション4.5.1.1「mysql のオプション」
セクション4.4.7「mysql_upgrade — MySQL テーブルのチェックとアップグレード」
セクション4.6.16「mysql_zap — パターンに一致するプロセスを強制終了」
セクション4.6.7「mysqlaccess — アクセス権限をチェックするクライアント」
セクション4.6.8「mysqlbinlog — バイナリログファイルを処理するためのユーティリティー」
セクション4.5.4「mysqldump — データベースバックアッププログラム」
セクション4.6.9「mysqldumpslow — スロークエリーログファイルの要約」
セクション4.5.6「mysqlshow — データベース、テーブル、およびカラム情報の表示」
セクション18.4.9「ndb_delete_all — NDB テーブルからのすべての行の削除」
セクション18.4.10「ndb_desc — NDB テーブルの表示」
セクション18.4.21「ndb_select_all — NDB テーブルの行の出力」
セクション18.4.24「ndb_show_tables — NDB テーブルのリストの表示」
セクション18.4.26「ndb_waiter — MySQL Cluster が指定したステータスになるまで待機する」
セクション5.1.3「サーバーコマンドオプション」

--tab

セクション4.5.4「mysqldump — データベースバックアッププログラム」
セクション7.4.3「mysqldump による区切りテキストフォーマットでのデータのダンプ」
セクション18.4.20「ndb_restore — MySQL Cluster バックアップのリストア」
セクション7.2「データベースバックアップ方法」
セクション7.1「バックアップとリカバリの種類」
セクション7.4「バックアップへの mysqldump の使用」

--table

セクション4.5.1.1「mysql のオプション」
セクション4.6.7「mysqlaccess — アクセス権限をチェックするクライアント」
セクション18.4.10「ndb_desc — NDB テーブルの表示」

--table-cache

セクション1.4「MySQL 5.6 の新機能」

--tables

セクション4.5.3「mysqlcheck — テーブル保守プログラム」
セクション4.5.4「mysqldump — データベースバックアッププログラム」

--tc-heuristic-recover

セクション5.1.3「サーバーコマンドオプション」

--tcp-ip

セクション4.3.4「mysqld_multi — 複数の MySQL サーバーの管理」

--tee

セクション4.5.1.1「mysql のオプション」
セクション4.5.1.2「mysql コマンド」

--temp-pool

セクション5.1.3「サーバーコマンドオプション」

--test

セクション4.6.5「myisampack — 圧縮された読み取り専用の MyISAM テーブルの生成」

--thread_cache_size

セクション24.4.1.4「gdb での mysqld のデバッグ」

--thread_handling

セクション1.4「MySQL 5.6 の新機能」
セクション5.1.3「サーバーコマンドオプション」

--thread_stack

セクション8.11.5.1「MySQL のクライアント接続のためのスレッドの使用方法」

--timeout

セクション18.4.26「ndb_waiter — MySQL Cluster が指定したステータスになるまで待機する」

--timezone

セクション10.6「MySQL Server でのタイムゾーンのサポート」
セクション4.3.2「mysqld_safe — MySQL サーバー起動スクリプト」
セクション5.1.4「サーバーシステム変数」
セクションB.5.4.6「タイムゾーンの問題」
セクション17.4.1.30「レプリケーションとタイムゾーン」

--tmpdir

セクション5.3「1 つのマシン上での複数の MySQL インスタンスの実行」
セクション4.6.3.3「myisamchk の修復オプション」
セクション4.6.3.6「myisamchk メモリー使用量」
セクション4.6.5「myisampack — 圧縮された読み取り専用の MyISAM テーブルの生成」
セクションB.5.4.4「MySQL が一時ファイルを格納する場所」
セクション4.4.7「mysql_upgrade — MySQL テーブルのチェックとアップグレード」
セクション4.6.10「mysqlhotcopy — データベースバックアッププログラム」
セクション5.1.3「サーバーコマンドオプション」
セクションB.5.2.13「ファイルを作成/書き込みできない」

tmpdir

セクション2.3「Microsoft Windows に MySQL をインストールする」

--to-last-log

セクション4.6.8「mysqlbinlog — バイナリログファイルを処理するためのユーティリティー」
セクション4.6.8.4「mysqlbinlog サーバー ID の指定」

セクション4.6.8.3「バイナリログファイルのバックアップのための mysqlbinlog の使用」

--transaction-isolation

セクション14.2.2「InnoDB のトランザクションモデルおよびロック」
セクション13.3.6「SET TRANSACTION 構文」
セクション5.1.3「サーバーコマンドオプション」
セクション5.1.4「サーバーシステム変数」

--transaction-read-only

セクション13.3.6「SET TRANSACTION 構文」
セクション5.1.3「サーバーコマンドオプション」
セクション5.1.4「サーバーシステム変数」

--transactional

セクション18.4.9「ndb_delete_all — NDB テーブルからのすべての行の削除」

--triggers

セクション4.5.4「mysqldump — データベースバックアッププログラム」
セクション7.4.5.3「ストアドプログラムのダンプ」

--try-reconnect

セクション18.4.5「ndb_mgm — MySQL Cluster 管理クライアント」

--tupscan

セクション18.4.21「ndb_select_all — NDB テーブルの行の出力」

--type

セクション4.6.11「mysql_convert_table_format — 指定されたストレージエンジンを使用するためのテーブルの変換」
セクション18.4.7「ndb_config — MySQL Cluster 構成情報の抽出」
セクション18.4.24「ndb_show_tables — NDB テーブルのリストの表示」

--tz-utc

セクション4.5.4「mysqldump — データベースバックアッププログラム」

U

[索引の先頭]

-U

セクション16.6.2「memcached の使用」
セクション4.6.3.2「myisamchk のチェックオプション」
セクション4.5.1.1「mysql のオプション」
セクション4.6.7「mysqlaccess — アクセス権限をチェックするクライアント」
セクション2.5.5「RPM パッケージを使用して MySQL を Linux にインストールする」

-u

セクション16.6.2「memcached の使用」
セクション4.6.3.3「myisamchk の修復オプション」
セクション4.6.4「myisamlog — MyISAM ログファイルの内容の表示」
セクション4.5.1.1「mysql のオプション」
セクション2.3.5.8「MySQL インストールのテスト」
セクション4.2.2「MySQL サーバーへの接続」
セクション4.2.1「MySQL プログラムの起動」
セクション4.6.6「mysql_config_editor — MySQL 構成ユーティリティー」
セクション4.4.7「mysql_upgrade — MySQL テーブルのチェックとアップグレード」
セクション4.6.7「mysqlaccess — アクセス権限をチェックするクライアント」
セクション4.5.2「mysqladmin — MySQL サーバーの管理を行うクライアント」
セクション4.6.8「mysqlbinlog — バイナリログファイルを処理するためのユーティリティー」
セクション4.5.3「mysqlcheck — テーブル保守プログラム」
セクション4.5.4「mysqldump — データベースバックアッププログラム」
セクション4.6.10「mysqlhotcopy — データベースバックアッププログラム」
セクション4.5.5「mysqlimport — データインポートプログラム」
セクション4.5.6「mysqlshow — データベース、テーブル、およびカラム情報の表示」
セクション4.5.7「mysqlslap — 負荷エミュレーションクライアント」
セクション18.4.10「ndb_desc — NDB テーブルの表示」
セクション18.4.20「ndb_restore — MySQL Cluster バックアップのリストア」
セクション18.4.24「ndb_show_tables — NDB テーブルのリストの表示」
セクション2.3.8「Windows でのインストール後の手順」
セクション5.1.3「サーバーコマンドオプション」
セクション6.3.1「ユーザー名とパスワード」

--unbuffered

セクション4.5.1.1「mysql のオプション」

--unpack

セクション15.2.3「MyISAM テーブルのストレージフォーマット」
セクション4.6.3.3「myisamchk の修復オプション」
セクション4.6.5「myisampack — 圧縮された読み取り専用の MyISAM テーブルの生成」

--unqualified

セクション18.4.10「ndb_desc — NDB テーブルの表示」
セクション18.4.24「ndb_show_tables — NDB テーブルのリストの表示」

--update

セクション18.4.14「ndb_index_stat — NDB インデックス統計ユーティリティー」

--update-state

セクション15.2「MyISAM ストレージエンジン」
セクション7.6.3「MyISAM テーブルの修復方法」
セクション4.6.3.2「myisamchk のチェックオプション」

--upgrade-system-tables

セクション4.4.7「mysql_upgrade — MySQL テーブルのチェックとアップグレード」

--usage
セクション18.4.27「MySQL Cluster プログラムに共通するオプション — MySQL Cluster プログラムに共通するオプション」
セクション18.4.7「ndb_config — MySQL Cluster 構成情報の抽出」

--use-frm
セクション4.5.3「mysqlcheck — テーブル保守プログラム」

--use-https
セクション18.2.1.1「MySQL Cluster Auto-Installer の要件」
セクション18.4.23「ndb_setup.py — MySQL Cluster のブラウザベース自動インストーラの開始」

--use-mysqld_safe
セクション4.3.3「mysql.server — MySQL サーバー起動スクリプト」

--use-threads
セクション4.5.5「mysqlimport — データインポートプログラム」

--useHexFormat
セクション18.4.21「ndb_select_all — NDB テーブルの行の出力」

--user
セクションB.5.2.18「'File' が見つかりません、および同様のエラー」
セクション18.5.11.3「MySQL Cluster と MySQL のセキュリティー手順」
セクション4.5.1.1「mysql のオプション」
セクション6.1.5「MySQL を通常ユーザーとして実行する方法」
セクション4.2.2「MySQL サーバーへの接続」
セクション4.2.1「MySQL プログラムの起動」
セクション4.3.3「mysql.server — MySQL サーバー起動スクリプト」
セクション4.6.6「mysql_config_editor — MySQL 構成ユーティリティー」
セクション4.6.11「mysql_convert_table_format — 指定されたストレージエンジンを使用するためのテーブルの変換」
セクション4.4.3「mysql_install_db — MySQL データディレクトリの初期化」
セクション4.6.14「mysql_setpermission — 付与テーブルに許可をインタラクティブに設定」
セクション4.4.7「mysql_upgrade — MySQL テーブルのチェックとアップグレード」
セクション4.6.7「mysqlaccess — アクセス権限をチェックするクライアント」
セクション4.5.2「mysqladmin — MySQL サーバーの管理を行うクライアント」
セクション4.6.8「mysqlbinlog — バイナリログファイルを処理するためのユーティリティー」
セクション4.5.3「mysqlcheck — テーブル保守プログラム」
セクション4.3.4「mysqld_multi — 複数の MySQL サーバーの管理」
セクション4.3.2「mysqld_safe — MySQL サーバー起動スクリプト」
セクション4.5.4「mysqldump — データベースバックアッププログラム」
セクション4.6.10「mysqlhotcopy — データベースバックアッププログラム」
セクション4.5.5「mysqlimport — データインポートプログラム」
セクション4.5.6「mysqlshow — データベース、テーブル、およびカラム情報の表示」
セクション4.5.7「mysqlslap — 負荷エミュレーションクライアント」
root のパスワードのリセット: UNIX システム
セクション2.10.1「Unix 類似システムでのインストール後の手順」
セクション4.2.9「オプションのデフォルト、値を想定するオプション、および = 記号」
セクション4.2.6「オプションファイルの使用」
セクション5.1.3「サーバーコマンドオプション」
セクション6.3.8.8「ソケットピア証明書認証プラグイン」
セクション6.3.8.9「テスト認証プラグイン」
セクション4.6.8.3「バイナリログファイルのバックアップのための mysqlbinlog の使用」
セクション7.3「バックアップおよびリカバリ戦略の例」
セクション6.3.7「プラガブル認証」
セクション6.3.1「ユーザー名とパスワード」
セクション6.1.3「攻撃者に対する MySQL のセキュアな状態の維持」

user
セクション4.7.3「my_print_defaults — オプションファイルからのオプションの表示」
セクション4.5.1.1「mysql のオプション」
セクション4.6.6「mysql_config_editor — MySQL 構成ユーティリティー」
セクション4.4.7「mysql_upgrade — MySQL テーブルのチェックとアップグレード」
セクション4.5.2「mysqladmin — MySQL サーバーの管理を行うクライアント」
セクション4.6.8「mysqlbinlog — バイナリログファイルを処理するためのユーティリティー」
セクション4.5.3「mysqlcheck — テーブル保守プログラム」
セクション4.5.4「mysqldump — データベースバックアッププログラム」
セクション4.5.5「mysqlimport — データインポートプログラム」
セクション4.5.6「mysqlshow — データベース、テーブル、およびカラム情報の表示」
セクション4.5.7「mysqlslap — 負荷エミュレーションクライアント」
セクション4.2.6「オプションファイルの使用」
セクション4.2.7「オプションファイルの処理に影響するコマンド行オプション」

V

[索引の先頭]

-V
セクション4.4.1「comp_err — MySQL エラーメッセージファイルのコンパイル」
セクション4.7.3「my_print_defaults — オプションファイルからのオプションの表示」
セクション4.6.3.1「myisamchk の一般オプション」
セクション4.6.4「myisamlog — MyISAM ログファイルの内容の表示」
セクション4.6.5「myisampack — 圧縮された読み取り専用の MyISAM テーブルの生成」

セクション18.4.27「MySQL Cluster プログラムに共通するオプション — MySQL Cluster プログラムに共通するオプション」
セクション4.5.1.1「mysql のオプション」
セクション4.6.6「mysql_config_editor — MySQL 構成ユーティリティー」
セクション4.4.4「mysql_plugin — MySQL サーバープラグインの構成」
セクション4.6.15「mysql_waitpid — プロセスを強制終了して終了を待機」
セクション4.5.2「mysqladmin — MySQL サーバーの管理を行うクライアント」
セクション4.6.8「mysqlbinlog — バイナリログファイルを処理するためのユーティリティー」
セクション4.5.3「mysqlcheck — テーブル保守プログラム」
セクション4.5.4「mysqldump — データベースバックアッププログラム」
セクション4.5.5「mysqlimport — データインポートプログラム」
セクション4.5.6「mysqlshow — データベース、テーブル、およびカラム情報の表示」
セクション4.5.7「mysqlslap — 負荷エミュレーションクライアント」
セクション18.4.7「ndb_config — MySQL Cluster 構成情報の抽出」
セクション4.8.1「perror — エラーコードの説明」
セクション4.8.2「replace — 文字列置換ユーティリティー」
セクション4.7.4「resolve_stack_dump — 数値スタックトレースダンプをシンボルに解決」
セクション4.8.3「resolveip — ホスト名と IP アドレスの解決」
セクション4.2.4「コマンド行でのオプションの使用」
セクション5.1.3「サーバーコマンドオプション」

-v

セクション4.6.1「innochecksum — オフライン InnoDB ファイルチェックサムユーティリティー」
セクション16.6.2「memcached の使用」
セクション16.6.2.8「memcached ログ」
セクション4.7.3「my_print_defaults — オプションファイルからのオプションの表示」
セクション7.6.2「MyISAM テーブルのエラーのチェック方法」
セクション4.6.2「myisam_ftdump — 全文インデックス情報の表示」
セクション4.6.3.5「myisamchk によるテーブル情報の取得」
セクション4.6.3.1「myisamchk の一般オプション」
セクション4.6.4「myisamlog — MyISAM ログファイルの内容の表示」
セクション4.6.5「myisampack — 圧縮された読み取り専用の MyISAM テーブルの生成」
セクション4.5.1.1「mysql のオプション」
セクション4.6.6「mysql_config_editor — MySQL 構成ユーティリティー」
セクション4.4.4「mysql_plugin — MySQL サーバープラグインの構成」
セクション4.6.15「mysql_waitpid — プロセスを強制終了して終了を待機」
セクション4.6.7「mysqlaccess — アクセス権限をチェックするクライアント」
セクション4.5.2「mysqladmin — MySQL サーバーの管理を行うクライアント」
セクション4.6.8「mysqlbinlog — バイナリログファイルを処理するためのユーティリティー」
セクション4.6.8.2「mysqlbinlog 行イベントの表示」
セクション4.5.3「mysqlcheck — テーブル保守プログラム」
セクション4.5.4「mysqldump — データベースバックアッププログラム」
セクション4.6.9「mysqldumpslow — スロークエリーログファイルの要約」
セクション4.5.5「mysqlimport — データインポートプログラム」
セクション4.5.6「mysqlshow — データベース、テーブル、およびカラム情報の表示」
セクション4.5.7「mysqlslap — 負荷エミュレーションクライアント」
セクション18.4.16「ndb_print_file — NDB ディスクデータファイル内容の出力」
セクション4.8.1「perror — エラーコードの説明」
セクション4.8.2「replace — 文字列置換ユーティリティー」
セクション4.2.4「コマンド行でのオプションの使用」
セクション5.1.3「サーバーコマンドオプション」

--validate-password

パスワード検証プラグインのインストール
パスワード検証プラグインのオプションおよび変数

--var_name

セクション14.12「InnoDB の起動オプションおよびシステム変数」
セクション4.6.3.1「myisamchk の一般オプション」
セクション4.5.1.1「mysql のオプション」
セクション4.5.2「mysqladmin — MySQL サーバーの管理を行うクライアント」
セクション4.6.8「mysqlbinlog — バイナリログファイルを処理するためのユーティリティー」
セクション4.5.4「mysqldump — データベースバックアッププログラム」
セクション5.1.3「サーバーコマンドオプション」

--verbose

セクション4.7.3「my_print_defaults — オプションファイルからのオプションの表示」
セクション4.6.2「myisam_ftdump — 全文インデックス情報の表示」
セクション4.6.3.1「myisamchk の一般オプション」
セクション4.6.5「myisampack — 圧縮された読み取り専用の MyISAM テーブルの生成」
セクション4.5.1.1「mysql のオプション」
セクション2.10.1.3「MySQL サーバーの起動とトラブルシューティング」
セクション4.6.6「mysql_config_editor — MySQL 構成ユーティリティー」
セクション4.6.11「mysql_convert_table_format — 指定されたストレージエンジンを使用するためのテーブルの変換」
セクション4.4.3「mysql_install_db — MySQL データディレクトリの初期化」
セクション4.4.4「mysql_plugin — MySQL サーバープラグインの構成」
セクション4.4.7「mysql_upgrade — MySQL テーブルのチェックとアップグレード」
セクション4.6.15「mysql_waitpid — プロセスを強制終了して終了を待機」
セクション4.5.2「mysqladmin — MySQL サーバーの管理を行うクライアント」
セクション4.6.8「mysqlbinlog — バイナリログファイルを処理するためのユーティリティー」

セクション4.6.8.2「mysqlbinlog 行イベントの表示」
セクション4.5.3「mysqlcheck — テーブル保守プログラム」
セクション4.3.4「mysqld_multi — 複数の MySQL サーバーの管理」
セクション4.5.4「mysqldump — データベースバックアッププログラム」
セクション4.6.9「mysqldumpslow — スロークエリーログファイルの要約」
セクション4.5.5「mysqlimport — データインポートプログラム」
セクション4.5.6「mysqlshow — データベース、テーブル、およびカラム情報の表示」
セクション4.5.7「mysqlslap — 負荷エミュレーションクライアント」
セクション18.4.6「ndb_blob_tool — MySQL Cluster テーブルの BLOB および TEXT カラムのチェックおよび修復」
セクション18.4.14「ndb_index_stat — NDB インデックス統計ユーティリティー」
セクション18.4.20「ndb_restore — MySQL Cluster バックアップのリストア」
セクション4.8.1「perror — エラーコードの説明」
セクション4.6.3.4「その他の myisamchk オプション」
セクション4.2.6「オプションファイルの使用」
セクション4.2.4「コマンド行でのオプションの使用」
セクション5.1.3「サーバーコマンドオプション」
セクション8.11.2「サーバーパラメータのチューニング」
セクション17.1.2.1「ステートメントベースおよび行ベースレプリケーションのメリットとデメリット」
セクション4.5.1.5「テキストファイルから SQL ステートメントを実行する」
セクション17.1.2.2「行ベースロギングおよびレプリケーションの使用」

--verify-binlog-checksum

セクション4.6.8「mysqlbinlog — バイナリログファイルを処理するためのユーティリティー」

--version

セクション4.4.1「comp_err — MySQL エラーメッセージファイルのコンパイル」
セクション4.7.3「my_print_defaults — オプションファイルからのオプションの表示」
セクション4.6.3.1「myisamchk の一般オプション」
セクション4.6.5「myisampack — 圧縮された読み取り専用の MyISAM テーブルの生成」
セクション18.4.27「MySQL Cluster プログラムに共通するオプション — MySQL Cluster プログラムに共通するオプション」
セクション4.5.1.1「mysql のオプション」
セクション4.7.2「mysql_config — クライアントのコンパイル用オプションの表示」
セクション4.6.6「mysql_config_editor — MySQL 構成ユーティリティー」
セクション4.6.11「mysql_convert_table_format — 指定されたストレージエンジンを使用するためのテーブルの変換」
セクション4.4.4「mysql_plugin — MySQL サーバープラグインの構成」
セクション4.6.15「mysql_waitpid — プロセスを強制終了して終了を待機」
セクション4.6.7「mysqlaccess — アクセス権限をチェックするクライアント」
セクション4.5.2「mysqladmin — MySQL サーバーの管理を行うクライアント」
セクション4.6.8「mysqlbinlog — バイナリログファイルを処理するためのユーティリティー」
セクション4.5.3「mysqlcheck — テーブル保守プログラム」
セクション4.3.4「mysqld_multi — 複数の MySQL サーバーの管理」
セクション4.5.4「mysqldump — データベースバックアッププログラム」
セクション4.5.5「mysqlimport — データインポートプログラム」
セクション4.5.6「mysqlshow — データベース、テーブル、およびカラム情報の表示」
セクション4.5.7「mysqlslap — 負荷エミュレーションクライアント」
セクション18.4.7「ndb_config — MySQL Cluster 構成情報の抽出」
セクション4.8.1「perror — エラーコードの説明」
セクション4.7.4「resolve_stack_dump — 数値スタックトレースダンプをシンボルに解決」
セクション4.8.3「resolveip — ホスト名と IP アドレスの解決」
セクション4.2.4「コマンド行でのオプションの使用」
セクション5.1.3「サーバーコマンドオプション」

--version-check

セクション4.4.7「mysql_upgrade — MySQL テーブルのチェックとアップグレード」

--vertical

セクション4.5.1.1「mysql のオプション」
セクション4.5.2「mysqladmin — MySQL サーバーの管理を行うクライアント」
セクション1.6「質問またはバグをレポートする方法」

W

[索引の先頭]

-W

セクション4.5.1.1「mysql のオプション」
セクション4.2.2「MySQL サーバーへの接続」
セクション4.4.7「mysql_upgrade — MySQL テーブルのチェックとアップグレード」
セクション4.5.2「mysqladmin — MySQL サーバーの管理を行うクライアント」
セクション4.5.3「mysqlcheck — テーブル保守プログラム」
セクション4.5.4「mysqldump — データベースバックアッププログラム」
セクション4.5.5「mysqlimport — データインポートプログラム」
セクション4.5.6「mysqlshow — データベース、テーブル、およびカラム情報の表示」
セクション4.5.7「mysqlslap — 負荷エミュレーションクライアント」
セクション5.1.3「サーバーコマンドオプション」

-w

セクション4.6.3.1「myisamchk の一般オプション」
セクション4.6.4「myisamlog — MyISAM ログファイルの内容の表示」
セクション4.6.5「myisampack — 圧縮された読み取り専用の MyISAM テーブルの生成」
セクション4.5.1.1「mysql のオプション」

セクション4.6.6「mysql_config_editor — MySQL 構成ユーティリティー」
セクション4.5.2「mysqladmin — MySQL サーバーの管理を行うクライアント」
セクション4.5.4「mysqldump — データベースバックアッププログラム」
セクション18.4.26「ndb_waiter — MySQL Cluster が指定したステータスになるまで待機する」

--wait
セクション4.6.3.1「myisamchk の一般オプション」
セクション4.6.5「myisampack — 圧縮された読み取り専用の MyISAM テーブルの生成」
セクション4.5.1.1「mysql のオプション」
セクション4.5.2「mysqladmin — MySQL サーバーの管理を行うクライアント」

--wait-nodes
セクション18.4.26「ndb_waiter — MySQL Cluster が指定したステータスになるまで待機する」

--warn
セクション4.6.6「mysql_config_editor — MySQL 構成ユーティリティー」

--where
セクション4.5.4「mysqldump — データベースバックアッププログラム」

--windows
セクション4.4.3「mysql_install_db — MySQL データディレクトリの初期化」

--with-libevent
セクション16.6.1「memcached のインストール」

--with-ndb-sci
セクション18.3.2.11「MySQL Cluster での SCI トランスポート接続」

WITH_BUNDLED_MEMCACHED
セクション2.9.4「MySQL ソース構成オプション」

WITH_CLASSPATH
セクション18.2.2.3「Linux でのソースからの MySQL Cluster のビルド」
セクション18.2.3.2「Windows でのソースからの MySQL Cluster のコンパイルとインストール」

WITH_DEBUG
セクション2.9.4「MySQL ソース構成オプション」
セクション13.7.5.16「SHOW ENGINE 構文」

WITH_EDITLINE
セクション2.9.4「MySQL ソース構成オプション」

WITH_LIBEDIT
セクション2.9.4「MySQL ソース構成オプション」

WITH_NDB_JAVA
セクション18.2.2.3「Linux でのソースからの MySQL Cluster のビルド」
セクション18.2.3.2「Windows でのソースからの MySQL Cluster のコンパイルとインストール」

WITH_NDBCLUSTER
セクション18.2.3.2「Windows でのソースからの MySQL Cluster のコンパイルとインストール」

WITH_NDBCLUSTER_STORAGE_ENGINE
セクション18.2.2.3「Linux でのソースからの MySQL Cluster のビルド」
セクション2.9.4「MySQL ソース構成オプション」
セクション18.2.3.2「Windows でのソースからの MySQL Cluster のコンパイルとインストール」

WITH_PERFSCHEMA_STORAGE_ENGINE
セクション22.2.1「パフォーマンススキーマビルド構成」

WITH_ZLIB
セクション2.9.4「MySQL ソース構成オプション」

--write-binlog
セクション17.1.3.4「GTID ベースレプリケーションの制約」
セクション4.4.7「mysql_upgrade — MySQL テーブルのチェックとアップグレード」
セクション4.5.3「mysqlcheck — テーブル保守プログラム」
セクション17.1.4.5「グローバルトランザクション ID のオプションと変数」

X

[索引の先頭]

-X
セクション4.5.1.1「mysql のオプション」
セクション4.5.1.2「mysql コマンド」
セクション4.5.4「mysqldump — データベースバックアッププログラム」

-x
セクション4.5.4「mysqldump — データベースバックアッププログラム」
セクション4.5.7「mysqlslap — 負荷エミュレーションクライアント」
セクション18.4.21「ndb_select_all — NDB テーブルの行の出力」

--xml
セクション13.2.7「LOAD XML 構文」
セクション4.5.1.1「mysql のオプション」
セクション4.5.4「mysqldump — データベースバックアッププログラム」
セクション18.4.7「ndb_config — MySQL Cluster 構成情報の抽出」
セクション12.11「XML 関数」

Y

[索引の先頭]

-Y
セクション4.5.4「mysqldump — データベースバックアッププログラム」

-y
セクション4.5.4「mysqldump — データベースバックアッププログラム」
セクション4.5.7「mysqlslap — 負荷エミュレーションクライアント」

Z

[索引の先頭]

-z
セクション18.4.20「ndb_restore — MySQL Cluster バックアップのリストア」
セクション18.4.21「ndb_select_all — NDB テーブルの行の出力」

権限の索引

A | C | D | E | F | G | I | L | P | R | S | T | U

A

[索引の先頭]

ALL
セクション13.7.1.4「GRANT 構文」
セクション6.2.1「MySQL で提供される権限」

ALL PRIVILEGES
セクション6.2.1「MySQL で提供される権限」

ALTER
セクション13.1.1「ALTER DATABASE 構文」
セクション13.1.7「ALTER TABLE 構文」
セクション13.7.1.4「GRANT 構文」
セクション21.25「INFORMATION_SCHEMA TABLE_PRIVILEGES テーブル」
セクション6.2.1「MySQL で提供される権限」
セクション13.1.32「RENAME TABLE 構文」
セクション19.3.3「パーティションとサブパーティションをテーブルと交換する」

ALTER ROUTINE
セクション13.1.4「ALTER FUNCTION 構文」
セクション13.1.5「ALTER PROCEDURE 構文」
セクション13.1.15「CREATE PROCEDURE および CREATE FUNCTION 構文」
セクション13.1.26「DROP PROCEDURE および DROP FUNCTION 構文」
セクション13.7.1.4「GRANT 構文」
セクション6.2.1「MySQL で提供される権限」
セクション5.1.4「サーバーシステム変数」
セクション20.7「ストアドプログラムのバイナリロギング」
セクション20.2.2「ストアドルーチンと MySQL 権限」

C

[索引の先頭]

CREATE
セクション13.1.7「ALTER TABLE 構文」
セクション13.1.10「CREATE DATABASE 構文」
セクション13.1.17「CREATE TABLE 構文」
セクション13.7.1.4「GRANT 構文」
セクション6.2.1「MySQL で提供される権限」
セクション13.1.32「RENAME TABLE 構文」
セクション19.3.3「パーティションとサブパーティションをテーブルと交換する」

CREATE ROUTINE
セクション13.1.15「CREATE PROCEDURE および CREATE FUNCTION 構文」
セクション13.7.1.4「GRANT 構文」
セクションA.4「MySQL 5.6 FAQ: ストアドプロシージャーおよびストアドファンクション」
セクション6.2.1「MySQL で提供される権限」
セクション5.1.4「サーバーシステム変数」
セクション20.7「ストアドプログラムのバイナリロギング」
セクション20.2.2「ストアドルーチンと MySQL 権限」

CREATE TABLESPACE
セクション13.7.1.4「GRANT 構文」
セクション6.2.1「MySQL で提供される権限」

CREATE TEMPORARY TABLES
セクション13.1.17「CREATE TABLE 構文」
セクション13.7.1.4「GRANT 構文」
セクション6.2.1「MySQL で提供される権限」

CREATE USER
セクション13.7.1.1「ALTER USER 構文」
セクション13.7.1.2「CREATE USER 構文」
セクション13.7.1.3「DROP USER 構文」
セクション13.7.1.4「GRANT 構文」
セクション6.2.1「MySQL で提供される権限」
セクション13.7.1.5「RENAME USER 構文」
セクション13.7.1.6「REVOKE 構文」

CREATE VIEW
セクション13.1.9「ALTER VIEW 構文」
セクション13.1.20「CREATE VIEW 構文」
セクション13.7.1.4「GRANT 構文」
セクション21.25「INFORMATION_SCHEMA TABLE_PRIVILEGES テーブル」
セクション6.2.1「MySQL で提供される権限」
セクションD.5「ビューの制約」

D

[索引の先頭]

DELETE
セクション13.1.17「CREATE TABLE 構文」
セクション13.2.2「DELETE 構文」
セクション13.7.3.2「DROP FUNCTION 構文」
セクション13.7.1.3「DROP USER 構文」
セクション13.7.1.4「GRANT 構文」
セクション15.7「MERGE ストレージエンジン」
セクション18.5.11.2「MySQL Cluster と MySQL 権限」
セクション6.2.1「MySQL で提供される権限」
セクション13.2.8「REPLACE 構文」
セクション22.9.2.4「setup_objects テーブル」
セクション13.7.3.4「UNINSTALL PLUGIN 構文」
セクション6.2.5「アクセス制御、ステージ 2: リクエストの確認」
セクション5.1.8.1「プラグインのインストールおよびアンインストール」
セクション24.3.2.5「ユーザー定義関数のコンパイルおよびインストール」
セクション24.3.2.6「ユーザー定義関数のセキュリティー上の予防措置」

DROP
セクション13.1.7「ALTER TABLE 構文」
セクション13.1.9「ALTER VIEW 構文」
セクション13.1.20「CREATE VIEW 構文」
セクション13.1.21「DROP DATABASE 構文」
セクション13.1.28「DROP TABLE 構文」
セクション13.1.31「DROP VIEW 構文」

セクション12.17.1「Enterprise Encryption のインストール」
セクション13.7.1.4「GRANT 構文」
セクション22.9.10.1「host_cache テーブル」
セクション21.25「INFORMATION_SCHEMA TABLE_PRIVILEGES テーブル」
セクション6.2.1「MySQL で提供される権限」
セクション6.2「MySQL アクセス権限システム」
セクション19.3.1「RANGE および LIST パーティションの管理」
セクション13.1.32「RENAME TABLE 構文」
セクション13.1.33「TRUNCATE TABLE 構文」
セクション22.8「パフォーマンススキーマの一般的なテーブル特性」
セクション19.3.3「パーティションとサブパーティションをテーブルと交換する」

E

[索引の先頭]

EVENT

セクション13.1.2「ALTER EVENT 構文」
セクション13.1.11「CREATE EVENT 構文」
セクション13.1.22「DROP EVENT 構文」
セクション13.7.1.4「GRANT 構文」
セクション6.2.1「MySQL で提供される権限」
セクション13.7.5.9「SHOW CREATE EVENT 構文」
セクション13.7.5.19「SHOW EVENTS 構文」
セクション20.4.6「イベントスケジューラと MySQL 権限」
セクション20.4.1「イベントスケジューラの概要」
セクション20.4.3「イベント構文」

EXECUTE

セクション13.1.15「CREATE PROCEDURE および CREATE FUNCTION 構文」
セクション13.1.26「DROP PROCEDURE および DROP FUNCTION 構文」
セクション13.7.1.4「GRANT 構文」
セクション6.2.1「MySQL で提供される権限」
セクション5.1.4「サーバーシステム変数」
セクション20.6「ストアドプログラムおよびビューのアクセスコントロール」
セクション20.2.2「ストアドルーチンと MySQL 権限」

F

[索引の先頭]

FILE

セクション11.4.3「BLOB 型と TEXT 型」
セクション12.17.2「Enterprise Encryption の使用法と例」
セクション13.7.1.4「GRANT 構文」
セクション13.2.6「LOAD DATA INFILE 構文」
セクション13.2.7「LOAD XML 構文」
セクション6.2.1「MySQL で提供される権限」
セクション4.5.4「mysqldump — データベースバックアッププログラム」
セクション7.4.3「mysqldump による区切りテキストフォーマットでのデータのダンプ」
セクション13.2.9.1「SELECT ... INTO 構文」
セクション6.2.7「アクセス拒否エラーの原因」
セクション6.1.3「攻撃者に対する MySQL のセキュアな状態の維持」
セクション12.5「文字列関数」
セクション6.2.2「権限システム付与テーブル」

G

[索引の先頭]

GRANT OPTION

セクション13.7.1.4「GRANT 構文」
セクション21.5「INFORMATION_SCHEMA COLUMN_PRIVILEGES テーブル」
セクション6.2.1「MySQL で提供される権限」
セクション13.7.1.6「REVOKE 構文」

I

[索引の先頭]

INDEX

セクション13.1.7「ALTER TABLE 構文」
セクション13.7.1.4「GRANT 構文」
セクション21.25「INFORMATION_SCHEMA TABLE_PRIVILEGES テーブル」
セクション6.2.1「MySQL で提供される権限」

INSERT

セクション13.1.7「ALTER TABLE 構文」
セクション13.7.1.1「ALTER USER 構文」
セクション13.7.2.1「ANALYZE TABLE 構文」
セクション13.7.1.2「CREATE USER 構文」
セクション13.1.20「CREATE VIEW 構文」
セクション12.17.1「Enterprise Encryption のインストール」
セクション13.7.1.4「GRANT 構文」
セクション21.5「INFORMATION_SCHEMA COLUMN_PRIVILEGES テーブル」
セクション21.25「INFORMATION_SCHEMA TABLE_PRIVILEGES テーブル」
セクション13.2.5「INSERT 構文」
セクション13.7.3.3「INSTALL PLUGIN 構文」
セクション6.2.1「MySQL で提供される権限」
セクション13.7.2.4「OPTIMIZE TABLE 構文」
セクション13.1.32「RENAME TABLE 構文」
セクション13.7.2.5「REPAIR TABLE 構文」
セクション13.2.8「REPLACE 構文」
セクション22.9.2.4「setup_objects テーブル」
セクション6.2.5「アクセス制御、ステージ 2: リクエストの確認」
セクション20.4.6「イベントスケジューラと MySQL 権限」
セクション5.1.3「サーバーコマンドオプション」
セクション20.6「ストアドプログラムおよびビューのアクセスコントロール」
セクション19.3.3「パーティションとサブパーティションをテーブルと交換する」
セクション15.11.1「プラガブルストレージエンジンのアーキテクチャー」
セクション5.1.8.1「プラグインのインストールおよびアンインストール」
セクション6.3.2「ユーザーアカウントの追加」
セクション13.7.3.1「ユーザー定義関数のための CREATE FUNCTION 構文」
セクション24.3.2.5「ユーザー定義関数のコンパイルおよびインストール」

セクション24.3.2.6「ユーザー定義関数のセキュリティー上の予防措置」

L

[索引の先頭]

LOCK TABLES

セクション13.7.6.3「FLUSH 構文」
セクション13.7.1.4「GRANT 構文」
セクション13.3.5「LOCK TABLES および UNLOCK TABLES 構文」
セクション6.2.1「MySQL で提供される権限」
セクション4.5.4「mysqldump — データベースバックアッププログラム」
セクション4.6.10「mysqlhotcopy — データベースバックアッププログラム」

P

[索引の先頭]

PROCESS

セクション13.7.1.4「GRANT 構文」
セクション21.15「INFORMATION_SCHEMA PROCESSLIST テーブル」
第21章「INFORMATION_SCHEMA テーブル」
セクション14.15.2「InnoDB モニターの有効化」
セクション13.7.6.4「KILL 構文」
セクション18.5.4「MySQL Cluster での MySQL サーバーの使用法」
セクション6.2.1「MySQL で提供される権限」
セクション13.7.5.16「SHOW ENGINE 構文」
セクション13.7.5.30「SHOW PROCESSLIST 構文」
セクション20.4.2「イベントスケジューラの構成」
セクション22.9.10.3「スレッドテーブル」
セクション8.12.5「スレッド情報の検査」
セクション6.3.2「ユーザーアカウントの追加」
セクション6.1.3「攻撃者に対する MySQL のセキュアな状態の維持」

PROXY

セクション13.7.1.4「GRANT 構文」
セクション22.9.10.1「host_cache テーブル」
セクション6.2.1「MySQL で提供される権限」
Windows 認証プラグインの使用
セクション6.3.9「プロキシユーザー」
プロキシユーザーとグループマッピングを使用した Unix パスワード認証
セクション6.2.2「権限システム付与テーブル」
認証プラグインでのプロキシユーザーサポートの実装

PROXY ... WITH GRANT OPTION

セクション6.3.9「プロキシユーザー」

R

[索引の先頭]

REFERENCES

セクション13.7.1.4「GRANT 構文」
セクション21.5「INFORMATION_SCHEMA COLUMN_PRIVILEGES テーブル」
セクション21.25「INFORMATION_SCHEMA TABLE_PRIVILEGES テーブル」
セクション6.2.1「MySQL で提供される権限」

RELOAD

セクション13.7.6.3「FLUSH 構文」
セクション13.7.1.4「GRANT 構文」
セクション22.9.10.1「host_cache テーブル」
セクション6.2.1「MySQL で提供される権限」
セクション23.7.7.56「mysql_refresh()」
セクション23.7.7.57「mysql_reload()」
セクション4.5.4「mysqldump — データベースバックアッププログラム」
セクション4.6.10「mysqlhotcopy — データベースバックアッププログラム」
セクション13.7.6.6「RESET 構文」
セクション6.2.5「アクセス制御、ステージ 2: リクエストの確認」
セクション6.3.2「ユーザーアカウントの追加」
セクション12.13「暗号化関数と圧縮関数」
セクション6.2.2「権限システム付与テーブル」

REPLICATION CLIENT

セクション13.7.1.4「GRANT 構文」
セクション6.2.1「MySQL で提供される権限」
セクション13.7.5.2「SHOW BINARY LOGS 構文」
セクション13.7.5.24「SHOW MASTER STATUS 構文」
セクション13.7.5.35「SHOW SLAVE STATUS 構文」

REPLICATION SLAVE

セクション13.7.1.4「GRANT 構文」
セクション6.2.1「MySQL で提供される権限」
セクション17.3.7「SSL を使用してレプリケーションをセットアップする」
セクション17.1.1.3「レプリケーション用ユーザーの作成」

S

[索引の先頭]

SELECT

セクション13.7.2.1「ANALYZE TABLE 構文」
セクション13.7.2.3「CHECKSUM TABLE 構文」
セクション13.1.15「CREATE PROCEDURE および CREATE FUNCTION 構文」
セクション13.1.17「CREATE TABLE 構文」
セクション13.1.19「CREATE TRIGGER 構文」
セクション13.1.20「CREATE VIEW 構文」
セクション13.2.2「DELETE 構文」
セクション13.7.1.4「GRANT 構文」
セクション21.5「INFORMATION_SCHEMA COLUMN_PRIVILEGES テーブル」
セクション21.25「INFORMATION_SCHEMA TABLE_PRIVILEGES テーブル」
セクション13.2.5「INSERT 構文」
セクション13.3.5「LOCK TABLES および UNLOCK TABLES 構文」
セクション15.7「MERGE ストレージエンジン」
セクション18.5.11.2「MySQL Cluster と MySQL 権限」
セクション6.2.1「MySQL で提供される権限」

セクション6.2「MySQL アクセス権限システム」
セクション4.5.4「mysqldump — データベースバックアッププログラム」
セクション4.6.10「mysqlhotcopy — データベースバックアッププログラム」
セクション13.7.2.4「OPTIMIZE TABLE 構文」
セクション13.7.2.5「REPAIR TABLE 構文」
セクション13.7.5.14「SHOW CREATE VIEW 構文」
セクション13.7.5.22「SHOW GRANTS 構文」
セクション13.2.11「UPDATE 構文」
セクション6.2.5「アクセス制御、ステージ 2: リクエストの確認」
セクション20.4.6「イベントスケジューラと MySQL 権限」
セクション20.6「ストアドプログラムおよびビューのアクセスコントロール」
セクション20.3.1「トリガーの構文と例」
セクション22.8「パフォーマンススキーマの一般的なテーブル特性」
セクションD.5「ビューの制約」

SHOW DATABASES

セクション13.7.1.4「GRANT 構文」
セクション6.2.1「MySQL で提供される権限」
セクション13.7.5.15「SHOW DATABASES 構文」
セクション5.1.4「サーバーシステム変数」

SHOW VIEW

セクション13.7.1.4「GRANT 構文」
セクション21.28「INFORMATION_SCHEMA VIEWS テーブル」
セクション6.2.1「MySQL で提供される権限」
セクション4.5.4「mysqldump — データベースバックアッププログラム」
セクション13.7.5.14「SHOW CREATE VIEW 構文」
セクションD.5「ビューの制約」

SHUTDOWN

セクション13.7.1.4「GRANT 構文」
セクション17.1.3.2「GTID を使用したレプリケーションのセットアップ」
セクション6.2.1「MySQL で提供される権限」
セクション23.7.7.66「mysql_shutdown()」
セクション4.3.4「mysqld_multi — 複数の MySQL サーバーの管理」
セクション6.2.5「アクセス制御、ステージ 2: リクエストの確認」
セクション5.1.12「シャットダウンプロセス」
セクション6.2.2「権限システム付与テーブル」

SUPER

セクション13.1.4「ALTER FUNCTION 構文」
セクション13.1.6「ALTER SERVER 構文」
セクション13.1.9「ALTER VIEW 構文」
セクション13.7.6.1「BINLOG 構文」
セクション13.1.11「CREATE EVENT 構文」
セクション13.1.15「CREATE PROCEDURE および CREATE FUNCTION 構文」
セクション13.1.16「CREATE SERVER 構文」
セクション13.1.19「CREATE TRIGGER 構文」
セクション13.1.20「CREATE VIEW 構文」
セクション24.4.3「DBUG パッケージ」
セクション13.1.27「DROP SERVER 構文」
セクション13.7.1.4「GRANT 構文」
セクション17.1.3.2「GTID を使用したレプリケーションのセットアップ」
セクション13.7.6.4「KILL 構文」
セクションA.4「MySQL 5.6 FAQ: ストアドプロシージャーおよびストアドファンクション」
セクション10.6「MySQL Server でのタイムゾーンのサポート」
セクション10.7「MySQL Server の ロケールサポート」
セクション6.2.1「MySQL で提供される権限」
セクション12.9.6「MySQL の全文検索の微調整」
セクション23.7.7.12「mysql_dump_debug_info()」
セクション4.6.8「mysqlbinlog — バイナリログファイルを処理するためのユーティリティー」
セクション13.7.1.7「SET PASSWORD 構文」
セクション13.4.1.3「SET sql_log_bin 構文」
セクション13.3.6「SET TRANSACTION 構文」
セクション13.7.4「SET 構文」
セクション13.7.5.2「SHOW BINARY LOGS 構文」
セクション13.7.5.24「SHOW MASTER STATUS 構文」
セクション13.7.5.30「SHOW PROCESSLIST 構文」
セクション13.7.5.35「SHOW SLAVE STATUS 構文」
セクション13.4.2.5「START SLAVE 構文」
セクション13.3.1「START TRANSACTION、COMMIT、および ROLLBACK 構文」
セクション13.4.2.6「STOP SLAVE 構文」
セクション6.3.5「アカウントパスワードの割り当て」
セクション10.1.5「アプリケーションの文字セットおよび照合順序の構成」
セクション20.4.6「イベントスケジューラと MySQL 権限」
セクション17.1.4.5「グローバルトランザクション ID のオプションと変数」
セクション5.1.7「サーバー SQL モード」
セクション5.1.4「サーバーシステム変数」
セクション5.1.5「システム変数の使用」
セクション20.6「ストアドプログラムおよびビューのアクセスコントロール」
セクション20.7「ストアドプログラムのバイナリロギング」
セクション5.2.4「バイナリログ」
セクション17.1.4.4「バイナリログのオプションと変数」
セクション5.2.4.2「バイナリログ形式の設定」
セクション17.1.1「レプリケーションのセットアップ方法」
セクション17.1.4.3「レプリケーションスレーブのオプションと変数」
セクション17.1.2「レプリケーション形式」
セクションB.5.2.7「接続が多すぎます」
セクション6.1.3「攻撃者に対する MySQL のセキュアな状態の維持」
セクション12.13「暗号化関数と圧縮関数」
セクション17.3.8.2「準同期レプリケーションのインストールと構成」

T

[索引の先頭]

TRIGGER

セクション13.1.19「CREATE TRIGGER 構文」
セクション13.1.30「DROP TRIGGER 構文」
セクション13.7.1.4「GRANT 構文」
セクション21.26「INFORMATION_SCHEMA TRIGGERS テーブル」
セクション6.2.1「MySQL で提供される権限」
セクション4.5.4「mysqldump — データベースバックアッププログラム」

セクション13.7.5.39「SHOW TRIGGERS 構文」
セクション20.6「ストアドプログラムおよびビューのアクセスコントロール」

U

[索引の先頭]

UPDATE

セクション13.1.17「CREATE TABLE 構文」
セクション13.1.19「CREATE TRIGGER 構文」
セクション13.7.1.4「GRANT 構文」
セクション21.5「INFORMATION_SCHEMA COLUMN_PRIVILEGES テーブル」
セクション21.25「INFORMATION_SCHEMA TABLE_PRIVILEGES テーブル」
セクション13.2.5「INSERT 構文」
セクション15.7「MERGE ストレージエンジン」
セクション18.5.11.2「MySQL Cluster と MySQL 権限」
セクション6.2.1「MySQL で提供される権限」
セクション13.7.1.5「RENAME USER 構文」
セクション13.7.1.6「REVOKE 構文」
セクション22.9.2.4「setup_objects テーブル」
セクション13.2.11「UPDATE 構文」
セクション6.3.5「アカウントパスワードの割り当て」
セクション20.6「ストアドプログラムおよびビューのアクセスコントロール」
セクション20.3.1「トリガーの構文と例」
セクション22.8「パフォーマンススキーマの一般的なテーブル特性」
セクション22.9.2「パフォーマンススキーマセットアップテーブル」
セクション22.2.3「パフォーマンススキーマ実行時構成」

USAGE

セクション13.7.1.4「GRANT 構文」
セクション6.2.1「MySQL で提供される権限」

SQL モードの索引

A | D | E | H | I | M | N | O | P | R | S | T

A

[索引の先頭]

ALLOW_INVALID_DATES

セクションB.5.5.2「DATE カラムの使用に関する問題」
セクション11.3.1「DATE、DATETIME、および TIMESTAMP 型」
セクション5.1.7「サーバー SQL モード」
セクション12.7「日付および時間関数」
セクション11.3「日付と時間型」

ANSI

セクション21.28「INFORMATION_SCHEMA VIEWS テーブル」
セクション13.7.5.14「SHOW CREATE VIEW 構文」
セクション5.1.7「サーバー SQL モード」
セクション9.2.4「関数名の構文解析と解決」

ANSI_QUOTES

セクション4.5.4「mysqldump — データベースバックアッププログラム」
セクション5.1.7「サーバー SQL モード」
セクション9.2「スキーマオブジェクト名」
セクション13.1.17.2「外部キー制約の使用」
セクション9.1.1「文字列リテラル」
セクション1.7.1「標準 SQL に対する MySQL 拡張機能」

D

[索引の先頭]

DB2

セクション5.1.7「サーバー SQL モード」

E

[索引の先頭]

ERROR_FOR_DIVISION_BY_ZERO

セクション1.4「MySQL 5.6 の新機能」
セクション5.1.7「サーバー SQL モード」
セクション12.20.3「式の処理」
セクション12.20.5「高精度計算の例」

H

[索引の先頭]

HIGH_NOT_PRECEDENCE

セクション5.1.7「サーバー SQL モード」
セクション9.5「式の構文」
セクション12.3.1「演算子の優先順位」

I

[索引の先頭]

IGNORE_SPACE

セクション13.1.15「CREATE PROCEDURE および CREATE FUNCTION 構文」
セクション4.5.1.1「mysql のオプション」
セクション5.1.7「サーバー SQL モード」
セクション9.2.4「関数名の構文解析と解決」

M

[索引の先頭]

MAXDB

セクション11.3.1「DATE、DATETIME、および TIMESTAMP 型」
セクション5.1.7「サーバー SQL モード」
セクション11.1.2「日付と時間型の概要」

MSSQL

セクション5.1.7「サーバー SQL モード」

MYSQL323

セクション5.1.7「サーバー SQL モード」

MYSQL40

セクション5.1.7「サーバー SQL モード」

N

[索引の先頭]

NO_AUTO_CREATE_USER

セクション13.7.1.4「GRANT 構文」
セクション5.1.7「サーバー SQL モード」
セクション6.3.2「ユーザーアカウントの追加」

NO_AUTO_VALUE_ON_ZERO

セクション3.6.9「AUTO_INCREMENT の使用」
セクション13.1.17「CREATE TABLE 構文」
セクション5.1.7「サーバー SQL モード」

NO_BACKSLASH_ESCAPES

セクション5.1.7「サーバー SQL モード」
セクション9.1.1「文字列リテラル」
セクション12.5.1「文字列比較関数」

NO_DIR_IN_CREATE

セクション13.1.17「CREATE TABLE 構文」
セクション19.2.6「サブパーティショニング」
セクション5.1.7「サーバー SQL モード」
セクション5.2.4「バイナリログ」
セクション17.4.1.10「レプリケーションと DIRECTORY テーブルオプション」
セクション17.4.1.34「レプリケーションと変数」

NO_ENGINE_SUBSTITUTION

セクション13.1.7「ALTER TABLE 構文」
セクション13.1.17「CREATE TABLE 構文」
セクション5.1.2.2「サンプルのデフォルトサーバー構成ファイルの使用」
セクション5.1.7「サーバー SQL モード」

セクション5.1.3「サーバーコマンドオプション」
セクション5.1.4「サーバーシステム変数」
セクション15.1「ストレージエンジンの設定」
セクション5.1.8.1「プラグインのインストールおよびアンインストール」
セクション17.3.2「異なるマスターおよびスレーブストレージエンジンでレプリケーションを使用する」

NO_FIELD_OPTIONS

セクション5.1.7「サーバー SQL モード」

NO_KEY_OPTIONS

セクション5.1.7「サーバー SQL モード」

NO_TABLE_OPTIONS

セクション5.1.7「サーバー SQL モード」

NO_UNSIGNED_SUBTRACTION

セクション12.10「キャスト関数と演算子」
セクション5.1.7「サーバー SQL モード」
セクション19.6「パーティショニングの制約と制限」
セクション11.1.1「数値型の概要」
セクション12.6.1「算術演算子」
セクション11.2.6「範囲外およびオーバーフローの処理」

NO_ZERO_DATE

セクション13.1.17「CREATE TABLE 構文」
セクションB.5.5.2「DATE カラムの使用に関する問題」
セクション1.4「MySQL 5.6 の新機能」
セクション11.3.5「TIMESTAMP および DATETIME の自動初期化および更新機能」
セクション12.10「キャスト関数と演算子」
セクション5.1.7「サーバー SQL モード」
セクション11.3「日付と時間型」

NO_ZERO_IN_DATE

セクション13.1.17「CREATE TABLE 構文」
セクションB.5.5.2「DATE カラムの使用に関する問題」
セクション1.4「MySQL 5.6 の新機能」
セクション5.1.7「サーバー SQL モード」
セクション11.3「日付と時間型」

O

[索引の先頭]

ONLY_FULL_GROUP_BY

セクション12.19.2「GROUP BY 修飾子」
セクション12.19.3「MySQL での GROUP BY の処理」
セクション5.1.7「サーバー SQL モード」
セクション3.3.4.8「行のカウント」

ORACLE

セクション5.1.7「サーバー SQL モード」

P

[索引の先頭]

PAD_CHAR_TO_FULL_LENGTH

セクション11.4.1「CHAR および VARCHAR 型」
セクション5.1.7「サーバー SQL モード」
セクション11.1.3「文字列型の概要」

PIPES_AS_CONCAT

セクション5.1.7「サーバー SQL モード」
セクション9.5「式の構文」
セクション12.3.1「演算子の優先順位」

POSTGRESQL

セクション5.1.7「サーバー SQL モード」

R

[索引の先頭]

REAL_AS_FLOAT

セクション5.1.7「サーバー SQL モード」
セクション11.2「数値型」
セクション11.1.1「数値型の概要」

S

[索引の先頭]

STRICT_ALL_TABLES

セクションA.3「MySQL 5.6 FAQ: サーバー SQL モード」
セクション1.4「MySQL 5.6 の新機能」
セクション5.1.7「サーバー SQL モード」
セクション6.3.2「ユーザーアカウントの追加」
セクション17.4.3「レプリケーションセットアップをアップグレードする」
セクション12.20.3「式の処理」
セクション1.7.3.3「無効データの制約」

STRICT_TRANS_TABLES

セクションA.3「MySQL 5.6 FAQ: サーバー SQL モード」
セクション1.4「MySQL 5.6 の新機能」
セクション5.1.2.2「サンプルのデフォルトサーバー構成ファイルの使用」
セクション5.1.7「サーバー SQL モード」
セクション6.3.2「ユーザーアカウントの追加」
セクション17.4.3「レプリケーションセットアップをアップグレードする」
セクション12.20.3「式の処理」
セクション1.7.3.3「無効データの制約」

T

[索引の先頭]

TRADITIONAL

セクションA.3「MySQL 5.6 FAQ: サーバー SQL モード」
セクション11.3.5「TIMESTAMP および DATETIME の自動初期化および更新機能」
セクション5.1.7「サーバー SQL モード」
セクション12.20.3「式の処理」

ステートメント/構文の索引

A | B | C | D | E | F | G | H | I | K | L | O | P | R | S | T | U | W | X

A

[索引の先頭]

ADD FULLTEXT INDEX

セクション14.11.1「オンライン DDL の概要」

ADD INDEX

セクション14.11.1「オンライン DDL の概要」

ALTER DATABASE

セクション13.1.1「ALTER DATABASE 構文」
セクション4.6.8「mysqlbinlog — バイナリログファイルを処理するためのユーティリティー」
セクション4.5.4「mysqldump — データベースバックアッププログラム」
セクション17.2.3「サーバーがレプリケーションフィルタリングルールをどのように評価するか」
セクション17.2.3.1「データベースレベルレプリケーションオプションおよびバイナリロギングオプションの評価」
セクション10.1.3.2「データベース文字セットおよび照合順序」
セクション17.1.4.3「レプリケーションスレーブのオプションと変数」
セクション1.7.1「標準 SQL に対する MySQL 拡張機能」

ALTER EVENT

セクション13.1.2「ALTER EVENT 構文」
セクション13.1.11「CREATE EVENT 構文」
セクション17.4.1.7「CURRENT_USER() のレプリケーション」
セクション21.7「INFORMATION_SCHEMA EVENTS テーブル」
セクション20.4.6「イベントスケジューラと MySQL 権限」
セクション20.4.1「イベントスケジューラの概要」
セクション20.4.4「イベントメタデータ」
セクション20.4.3「イベント構文」
セクション20.7「ストアドプログラムのバイナリロギング」
セクションD.1「ストアドプログラムの制約」
セクション17.4.1.11「呼び出される機能のレプリケーション」
セクション12.14「情報関数」
セクション13.3.3「暗黙的なコミットを発生させるステートメント」

ALTER EVENT event_name ENABLED

セクション17.4.1.11「呼び出される機能のレプリケーション」

ALTER FUNCTION

セクション13.1.4「ALTER FUNCTION 構文」
セクション20.7「ストアドプログラムのバイナリロギング」
セクション20.2.1「ストアドルーチンの構文」
セクション13.3.3「暗黙的なコミットを発生させるステートメント」

ALTER LOGFILE GROUP

セクション13.1.3「ALTER LOGFILE GROUP 構文」
セクション21.30.1「INFORMATION_SCHEMA FILES テーブル」
セクション18.1.6.8「MySQL Cluster に限定された問題」
セクション18.5.12.1「MySQL Cluster ディスクデータオブジェクト」
セクション18.3.2.6「MySQL Cluster データノードの定義」

ALTER ONLINE TABLE

セクション18.5.13.3「MySQL Cluster データノードのオンライン追加: 詳細な例」

ALTER ONLINE TABLE ... REORGANIZE PARTITION

セクション18.5.13.1「MySQL Cluster データノードのオンライン追加: 一般的な問題」
セクション18.5.13.2「MySQL Cluster データノードのオンライン追加: 基本的な手順」
セクション18.5.13.3「MySQL Cluster データノードのオンライン追加: 詳細な例」
セクション18.5.2「MySQL Cluster 管理クライアントのコマンド」

ALTER ONLINE TABLE tbl REORGANIZE PARTITION

セクション18.5.13.2「MySQL Cluster データノードのオンライン追加: 基本的な手順」

ALTER PROCEDURE

セクション13.1.5「ALTER PROCEDURE 構文」
セクション20.7「ストアドプログラムのバイナリロギング」
セクション20.2.1「ストアドルーチンの構文」
セクション13.3.3「暗黙的なコミットを発生させるステートメント」

ALTER SCHEMA

セクション13.1.1「ALTER DATABASE 構文」

ALTER SERVER

セクション17.4.1.6「CREATE SERVER、ALTER SERVER、および DROP SERVER のレプリケーション」
セクション17.1.4.5「グローバルトランザクション ID のオプションと変数」
セクション13.3.3「暗黙的なコミットを発生させるステートメント」

ALTER TABLE

セクションB.5.7.1「ALTER TABLE での問題」
セクション13.1.7.3「ALTER TABLE の例」
セクション13.1.7.1「ALTER TABLE パーティション操作」
セクション13.1.7「ALTER TABLE 構文」
セクション3.6.9「AUTO_INCREMENT の使用」
セクション13.7.2.2「CHECK TABLE 構文」
セクション13.1.13「CREATE INDEX 構文」
セクション13.1.17「CREATE TABLE 構文」
セクション14.11.4「DDL ステートメントの結合または分離」
セクション5.2.6「DDL ログ」
セクション13.1.24「DROP INDEX 構文」

セクション8.8.2「EXPLAIN 出力フォーマット」
セクション15.8.3「FEDERATED ストレージエンジンの注記とヒント」
セクション1.7.3.2「FOREIGN KEY の制約」
セクション14.2.13.3「FULLTEXT インデックス」
セクション13.7.1.4「GRANT 構文」
セクション19.3.2「HASH および KEY パーティションの管理」
セクション21.29.7「INFORMATION_SCHEMA INNODB_SYS_TABLES テーブル」
セクション21.29.15「INFORMATION_SCHEMA INNODB_SYS_TABLESPACES テーブル」
セクション21.13「INFORMATION_SCHEMA PARTITIONS テーブル」
セクション14.5.2「InnoDB File-Per-Table モード」
セクション14.18.5.4「InnoDB memcached プラグインのトランザクション動作の制御」
セクション8.3.7「InnoDB および MyISAM インデックス統計コレクション」
セクション14.6.6「InnoDB と FOREIGN KEY 制約」
セクション14.11「InnoDB とオンライン DDL」
セクション14.19.5「InnoDB のエラーコード」
セクション14.19.2「InnoDB のリカバリの強制的な実行」
セクション14.12「InnoDB の起動オプションおよびシステム変数」
セクション14.7.5「InnoDB テーブルでの圧縮の動作」
セクション14.13.17「InnoDB テーブルに対する ANALYZE TABLE の複雑さの推定」
セクション14.6.1「InnoDB テーブルの作成」
セクション14.19.3「InnoDB データディクショナリの操作のトラブルシューティング」
セクション14.9.1「InnoDB 行ストレージの概要」
セクション13.2.5.2「INSERT DELAYED 構文」
セクション13.7.6.4「KILL 構文」
セクション13.3.5「LOCK TABLES および UNLOCK TABLES 構文」
セクション15.3「MEMORY ストレージエンジン」
セクション15.7.2「MERGE テーブルの問題点」
セクション14.6.4「MyISAM から InnoDB へのテーブルの変換」
セクション15.2「MyISAM ストレージエンジン」
セクション15.2.3「MyISAM テーブルのストレージフォーマット」
セクション7.6.3「MyISAM テーブルの修復方法」
セクション15.2.1「MyISAM 起動オプション」
セクション4.6.3.1「myisamchk の一般オプション」
セクション2.11.1.3「MySQL 5.5 から 5.6 へのアップグレード」
セクションA.10「MySQL 5.6 FAQ: MySQL Cluster」
セクション1.4「MySQL 5.6 の新機能」
セクション13.1.7.2「MySQL Cluster での ALTER TABLE オンライン操作」
セクション18.1.6.1「MySQL Cluster での SQL 構文の不適合」
セクション18.1.6.3「MySQL Cluster でのトランザクション処理に関する制限」
セクション18.1.6.8「MySQL Cluster に限定された問題」
セクション18.3.4.3「MySQL Cluster のシステム変数」
セクション18.2.4「MySQL Cluster の初期構成」
セクション18.1.6.2「MySQL Cluster の制限と標準の MySQL の制限との違い」
セクション18.1.6.6「MySQL Cluster の未サポート機能または欠落している機能」
セクション18.5.12.1「MySQL Cluster ディスクデータオブジェクト」
セクション18.5.13.3「MySQL Cluster データノードのオンライン追加: 詳細な例」
セクション18.3.2.6「MySQL Cluster データノードの定義」
セクション18.6.4「MySQL Cluster レプリケーションスキーマとテーブル」
セクション18.3.2.1「MySQL Cluster 構成の基本的な例」
セクションB.5.4.4「MySQL が一時ファイルを格納する場所」
セクションB.5.4.3「MySQL が満杯のディスクを処理する方法」
セクションB.5.4.2「MySQL が繰り返しクラッシュする場合の対処方法」
セクション6.2.1「MySQL で提供される権限」
セクション19.1「MySQL のパーティショニングの概要」
セクション12.9.6「MySQL の全文検索の微調整」
セクションB.5.8「MySQL の既知の問題」
MySQL 用語集
セクション23.7.7.35「mysql_info()」
セクション4.5.4「mysqldump — データベースバックアッププログラム」
セクション18.4.20「ndb_restore — MySQL Cluster バックアップのリストア」
セクション13.7.2.4「OPTIMIZE TABLE 構文」
セクション19.2.3.1「RANGE COLUMNS パーティショニング」
セクション19.3.1「RANGE および LIST パーティションの管理」
セクション19.2.1「RANGE パーティショニング」
セクション13.1.32「RENAME TABLE 構文」
セクション13.7.5.16「SHOW ENGINE 構文」
セクション13.7.5.41「SHOW WARNINGS 構文」
セクション14.7.7「SQL 圧縮構文の警告とエラー」
セクションB.5.7.2「TEMPORARY テーブルに関する問題」
Unix 上の MyISAM へのシンボリックリンクの使用
セクション11.3.4「YEAR(2) の制限と YEAR(4) への移行」
セクション14.11.7「オンライン DDL でのクラッシュリカバリの動作のしくみ」
セクション14.11.2「オンライン DDL でのパフォーマンスと並列性に関する考慮事項」
セクション14.11.3「オンライン DDL の SQL 構文」
セクション14.11.5「オンライン DDL の例」
セクション14.11.9「オンライン DDL の制限」
セクション14.11.6「オンライン DDL の実装の詳細」
セクション14.11.1「オンライン DDL の概要」
セクション10.1.3.4「カラム文字セットおよび照合順序」
セクション10.1.13「カラム文字セットの変換」
セクション8.9.3.1「クエリーキャッシュの動作」
セクション5.1.7「サーバー SQL モード」
セクション5.1.3「サーバーコマンドオプション」
セクション5.1.4「サーバーシステム変数」
セクション19.6.2「ストレージエンジンに関連するパーティショニング制限」
セクション15.1「ストレージエンジンの設定」
セクション5.2.5「スロークエリーログ」
セクション18.2.6「テーブルとデータを含む MySQL Cluster の例」
セクション14.10.4「テーブルのデフラグ」
セクション3.3.2「テーブルの作成」
セクション14.9.2「テーブルの行フォーマットの指定」
セクション2.11.4「テーブルまたはインデックスの再作成または修復」
セクションD.10.3「テーブルサイズの制限」

セクション14.5.4「テーブルスペースの位置の指定」
セクション14.7.2「テーブル圧縮の有効化」
セクション14.7.1「テーブル圧縮の概要」
セクション10.1.3.3「テーブル文字セットおよび照合順序」
セクション17.1.4.4「バイナリログのオプションと変数」
セクション5.2.4.2「バイナリログ形式の設定」
セクション19.6.4「パーティショニングとロック」
セクション19.6「パーティショニングの制約と制限」
セクション19.6.1「パーティショニングキー、主キー、および一意キー」
セクション19.3.3「パーティションとサブパーティションをテーブルと交換する」
セクション19.3.4「パーティションの保守」
セクション14.11.8「パーティション化された InnoDB テーブルに対するオンライン DDL」
セクション19.3「パーティション管理」
セクションD.5「ビューの制約」
セクション14.8.2「ファイル形式の互換性の確認」
マスターまたはスレーブでカラムが多い場合のレプリケーション
セクション17.4.1.1「レプリケーションと AUTO_INCREMENT」
セクション17.4.1.25「レプリケーションと予約語」
セクション17.2.2「レプリケーションリレーおよびステータスログ」
セクション5.2.1「一般クエリーログおよびスロークエリーログの出力先の選択」
セクション8.12.5.2「一般的なスレッドの状態」
セクション14.2.4「一貫性非ロック読み取り」
セクション10.1.11「以前の Unicode サポートから現在の Unicode サポートへのアップグレード」
セクション12.9「全文検索関数」
セクション14.6.2「別のマシンへの InnoDB テーブルの移動またはコピー」
セクション13.1.17.2「外部キー制約の使用」
セクション14.6.5.1「従来の InnoDB の自動インクリメントロック」
セクション12.14「情報関数」
セクション11.1.3「文字列型の概要」
セクション13.1.17.3「暗黙のカラム指定の変更」
セクション13.3.3「暗黙的なコミットを発生させるステートメント」
セクション1.7.1「標準 SQL に対する MySQL 拡張機能」
セクション14.13.16.1「永続的オプティマイザ統計のパラメータの構成」
セクション17.3.2「異なるマスターおよびスレーブストレージエンジンでレプリケーションを使用する」
セクション11.5.3.6「空間インデックスの作成」
セクション11.5.3.2「空間カラムの作成」
セクション11.2.6「範囲外およびオーバーフローの処理」
セクション18.1.6.10「複数の MySQL Cluster ノードに関する制限」

ALTER TABLE ... ADD FOREIGN KEY

セクション13.1.7「ALTER TABLE 構文」
セクション13.1.17.2「外部キー制約の使用」

ALTER TABLE ... ALGORITHM=COPY

セクション13.1.7「ALTER TABLE 構文」
セクション14.11.9「オンライン DDL の制限」
セクション13.1.17.2「外部キー制約の使用」

ALTER TABLE ... ALGORITHM=INPLACE

セクション13.1.7「ALTER TABLE 構文」
セクション14.11.9「オンライン DDL の制限」
セクション13.1.17.2「外部キー制約の使用」

ALTER TABLE ... DISCARD TABLESPACE

MySQL 用語集
セクション14.5.5「テーブルスペースの別のサーバーへのコピー (トランスポータブルテーブルスペース)」

ALTER TABLE ... DROP FOREIGN KEY

セクション13.1.7「ALTER TABLE 構文」
セクション13.1.17.2「外部キー制約の使用」

ALTER TABLE ... ENGINE = MEMORY

セクション17.4.1.21「レプリケーションと MEMORY テーブル」

ALTER TABLE ... ENGINE = NDB

セクション18.5.14「MySQL Cluster の配布された MySQL 権限」

ALTER TABLE ... ENGINE=...

セクション1.4「MySQL 5.6 の新機能」

ALTER TABLE ... ENGINE=INNODB

セクション2.11.1.3「MySQL 5.5 から 5.6 へのアップグレード」
セクション1.4「MySQL 5.6 の新機能」

ALTER TABLE ... ENGINE=InnoDB

セクション17.2.2「レプリケーションリレーおよびステータスログ」

ALTER TABLE ... EXCHANGE PARTITION

セクション13.1.7.1「ALTER TABLE パーティション操作」
セクション1.4「MySQL 5.6 の新機能」
セクション19.6.4「パーティショニングとロック」
セクション19.3.3「パーティションとサブパーティションをテーブルと交換する」

ALTER TABLE ... FORCE

セクション1.4「MySQL 5.6 の新機能」
セクション13.7.2.4「OPTIMIZE TABLE 構文」

ALTER TABLE ... IMPORT TABLESPACE

セクション13.1.7「ALTER TABLE 構文」
セクション2.11.1.3「MySQL 5.5 から 5.6 へのアップグレード」

MySQL 用語集
セクション14.5.5「テーブルスペースの別のサーバーへのコピー (トランスポータブルテーブルスペース)」
セクション14.6.2「別のマシンへの InnoDB テーブルの移動またはコピー」

ALTER TABLE ... OPTIMIZE PARTITION

セクション19.6.2「ストレージエンジンに関連するパーティショニング制限」
セクション19.3.4「パーティションの保守」

ALTER TABLE ... PARTITION BY

セクション19.6.1「パーティショニングキー、主キー、および一意キー」

ALTER TABLE ... PARTITION BY ...

セクション19.3.1「RANGE および LIST パーティションの管理」
セクション19.6「パーティショニングの制約と制限」

ALTER TABLE ... RENAME

Unix 上の MyISAM へのシンボリックリンクの使用

ALTER TABLE ... REORGANIZE PARTITION

セクション18.5.13.1「MySQL Cluster データノードのオンライン追加: 一般的な問題」
セクション18.3.2.6「MySQL Cluster データノードの定義」
セクション18.3.2.7「MySQL Cluster 内の SQL ノードおよびその他の API ノードの定義」

ALTER TABLE ... REPAIR PARTITION

セクション19.3.4「パーティションの保守」

ALTER TABLE ... TRUNCATE PARTITION

セクション19.6.4「パーティショニングとロック」
セクション19.3.4「パーティションの保守」
セクション19.3「パーティション管理」

ALTER TABLE EXCHANGE PARTITION

セクション19.3.3「パーティションとサブパーティションをテーブルと交換する」

ALTER TABLE IGNORE

セクション13.1.7「ALTER TABLE 構文」
セクション13.1.17.2「外部キー制約の使用」

ALTER TABLE mysql.ndb_apply_status ENGINE=MyISAM

セクション18.6.3「MySQL Cluster レプリケーションの既知の問題」

ALTER TABLE t TRUNCATE PARTITION ()

セクション13.2.2「DELETE 構文」

ALTER TABLE t3 DROP PARTITION p2

セクション5.2.6「DDL ログ」

ALTER TABLE table_name REORGANIZE PARTITION

セクション18.5.13.3「MySQL Cluster データノードのオンライン追加: 詳細な例」

ALTER TABLE tbl_name ENGINE=INNODB

セクション13.1.7「ALTER TABLE 構文」
セクション14.10.4「テーブルのデフラグ」

ALTER TABLE tbl_name FORCE

セクション13.1.7「ALTER TABLE 構文」
セクション14.10.4「テーブルのデフラグ」

ALTER TABLESPACE

セクション13.1.8「ALTER TABLESPACE 構文」
セクション13.1.18「CREATE TABLESPACE 構文」
セクション21.30.1「INFORMATION_SCHEMA FILES テーブル」
セクション18.1.6.8「MySQL Cluster に限定された問題」
セクション18.5.12.1「MySQL Cluster ディスクデータオブジェクト」
セクション18.3.2.6「MySQL Cluster データノードの定義」

ALTER USER

セクション6.3.8.3「4.1 よりも前のパスワードハッシュ方式と mysql_old_password プラグインからの移行」
セクション13.7.1.1「ALTER USER 構文」
セクション1.4「MySQL 5.6 の新機能」
セクション6.3.6「パスワードの期限切れとサンドボックスモード」
セクション6.2.2「権限システム付与テーブル」

ALTER VIEW

セクション13.1.9「ALTER VIEW 構文」
セクション13.1.20「CREATE VIEW 構文」
セクション17.4.1.7「CURRENT_USER() のレプリケーション」
セクションD.1「ストアドプログラムの制約」
セクション20.5.1「ビューの構文」
セクション20.5.2「ビュー処理アルゴリズム」
セクション12.14「情報関数」
セクション13.3.3「暗黙的なコミットを発生させるステートメント」

ANALYZE TABLE

セクション13.1.7「ALTER TABLE 構文」
セクション13.7.2.1「ANALYZE TABLE 構文」
セクション13.1.13「CREATE INDEX 構文」
セクション13.1.17「CREATE TABLE 構文」

セクション8.8.1「EXPLAIN によるクエリーの最適化」
セクション8.8.2「EXPLAIN 出力フォーマット」
セクション13.8.2「EXPLAIN 構文」
セクション8.3.7「InnoDB および MyISAM インデックス統計コレクション」
セクション14.12「InnoDB の起動オプションおよびシステム変数」
InnoDB オプティマイザ統計でサンプリングされるページの数の構成
セクション14.13.17「InnoDB テーブルに対する ANALYZE TABLE の複雑さの推定」
セクション14.6.7「InnoDB テーブル上の制限」
InnoDB 永続的統計テーブルの例
セクション15.7.2「MERGE テーブルの問題点」
セクション8.6.1「MyISAM クエリーの最適化」
セクション7.6「MyISAM テーブルの保守とクラッシュリカバリ」
セクション4.6.3.1「myisamchk の一般オプション」
セクション6.2.1「MySQL で提供される権限」
セクション12.9.6「MySQL の全文検索の微調整」
セクションB.5.8「MySQL の既知の問題」
MySQL 用語集
セクション4.5.3「mysqlcheck — テーブル保守プログラム」
セクション8.2.1.1「SELECT ステートメントの速度」
セクション13.7.5.23「SHOW INDEX 構文」
セクション17.1.4.5「グローバルトランザクション ID のオプションと変数」
セクション5.1.3「サーバーコマンドオプション」
セクション5.1.4「サーバーシステム変数」
セクション5.2.5「スロークエリーログ」
セクション19.6「パーティショニングの制約と制限」
セクション19.3.4「パーティションの保守」
セクション17.4.1.14「レプリケーションと FLUSH」
セクション8.12.5.2「一般的なスレッドの状態」
セクション13.3.3「暗黙的なコミットを発生させるステートメント」
セクション1.7.1「標準 SQL に対する MySQL 拡張機能」
セクション14.13.16.1「永続的オプティマイザ統計のパラメータの構成」
複数値比較の等価範囲の最適化
セクション14.13.16.2「非永続的オプティマイザ統計のパラメータの構成」

autocommit = 0 の場合、

セクション14.12「InnoDB の起動オプションおよびシステム変数」

B

[索引の先頭]

BEGIN

セクション13.6.1「BEGIN ... END 複合ステートメント構文」
セクション14.19.4「InnoDB のエラー処理」
セクション14.2.2「InnoDB のトランザクションモデルおよびロック」
セクション18.6.11「MySQL Cluster レプリケーションの競合解決」
セクション5.1.4「サーバーシステム変数」
セクション20.7「ストアドプログラムのバイナリロギング」
セクションD.1「ストアドプログラムの制約」
セクション17.4.1.31「レプリケーションとトランザクション」
セクション17.1.4.3「レプリケーションスレーブのオプションと変数」
セクション13.3.3「暗黙的なコミットを発生させるステートメント」

BEGIN ... END

セクション13.6.1「BEGIN ... END 複合ステートメント構文」
セクション13.6.5.1「CASE 構文」
セクション13.1.19「CREATE TRIGGER 構文」
セクション13.6.7.2「DECLARE ... HANDLER 構文」
セクション13.6.3「DECLARE 構文」
セクション13.6.5.4「LEAVE 構文」
セクション13.6「MySQL 複合ステートメント構文」
セクション13.3.1「START TRANSACTION、COMMIT、および ROLLBACK 構文」
セクション20.4.1「イベントスケジューラの概要」
セクション13.6.6.1「カーソルの CLOSE 構文」
セクション13.6.2「ステートメントラベルの構文」
セクションD.1「ストアドプログラムの制約」
セクション20.1「ストアドプログラムの定義」
セクション20.3.1「トリガーの構文と例」
セクション13.6.7.6「ハンドラのスコープに関するルール」
セクション13.6.4.1「ローカル変数の DECLARE 構文」
セクション13.6.4.2「ローカル変数のスコープと解決」
セクション13.3.3「暗黙的なコミットを発生させるステートメント」

BINLOG

セクション13.7.6.1「BINLOG 構文」
セクション4.6.8「mysqlbinlog — バイナリログファイルを処理するためのユーティリティー」
セクション4.6.8.2「mysqlbinlog 行イベントの表示」

C

[索引の先頭]

CACHE INDEX

セクション13.7.6.2「CACHE INDEX 構文」
セクション13.7.6.5「LOAD INDEX INTO CACHE 構文」
セクション8.9.2.4「インデックスプリロード」
セクション17.1.4.5「グローバルトランザクション ID のオプションと変数」
セクション19.6「パーティショニングの制約と制限」
セクション13.3.3「暗黙的なコミットを発生させるステートメント」
セクション8.9.2.2「複合キーキャッシュ」

CALL

セクション23.7.20「C API のプリペアド CALL ステートメントのサポート」
セクション23.7.5「C API データ構造」
セクション23.7.18「C API プリペアドステートメントの問題」
セクション13.2.1「CALL 構文」
セクション13.1.15「CREATE PROCEDURE および CREATE FUNCTION 構文」
セクション23.7.7.1「mysql_affected_rows()」
セクション23.7.7.37「mysql_insert_id()」
セクション23.7.7.45「mysql_more_results()」
セクション23.7.7.46「mysql_next_result()」
セクション23.7.7.53「mysql_real_connect()」

セクション23.7.7.65「mysql_set_server_option()」
セクション23.7.11.17「mysql_stmt_next_result()」
第20章「ストアドプログラムおよびビュー」
セクション20.6「ストアドプログラムおよびビューのアクセスコントロール」
セクション20.7「ストアドプログラムのバイナリロギング」
セクション20.2.1「ストアドルーチンの構文」
セクション20.3.1「トリガーの構文と例」
セクション13.5「準備済みステートメントのための SQL 構文」
セクション23.7.17「複数ステートメント実行の C API サポート」

CALL p()

条件値とオプションの新しいシグナル情報を含む RESIGNAL

CALL stored_procedure()

セクション19.6.4「パーティショニングとロック」

CASE

セクション13.6.5.1「CASE 構文」
セクション13.6.5「フロー制御ステートメント」
セクション8.9.4「プリペアドステートメントおよびストアドプログラムのキャッシュ」
セクション12.4「制御フロー関数」

CHANGE MASTER TO

セクション13.4.2.1「CHANGE MASTER TO 構文」
セクション13.7.1.4「GRANT 構文」
セクション17.1.3.1「GTID の概念」
セクション17.1.3.2「GTID を使用したレプリケーションのセットアップ」
セクション1.4「MySQL 5.6 の新機能」
セクション18.6.10「MySQL Cluster レプリケーション: マルチマスターと循環レプリケーション」
セクション18.6.9「MySQL Cluster レプリケーションを使用した MySQL Cluster バックアップ」
セクション6.2.1「MySQL で提供される権限」
セクション4.5.4「mysqldump — データベースバックアッププログラム」
セクション17.1.1.5「mysqldump を使用したデータスナップショットの作成」
セクション13.4.2.3「RESET SLAVE 構文」
セクション13.7.1.7「SET PASSWORD 構文」
セクション13.7.5.35「SHOW SLAVE STATUS 構文」
セクション17.3.7「SSL を使用してレプリケーションをセットアップする」
セクション13.4.2.5「START SLAVE 構文」
セクション17.1.4.5「グローバルトランザクション ID のオプションと変数」
セクション5.1.6「サーバーステータス変数」
セクション17.3.1.2「スレーブからローデータをバックアップする」
セクション17.1.1.10「スレーブでのマスター構成の設定」
セクション17.2.2.2「スレーブステータスログ」
セクション17.1.3.3「フェイルオーバーおよびスケールアウトでの GTID の使用」
セクション17.3.6「フェイルオーバー中にマスターを切り替える」
セクション17.1.4「レプリケーションおよびバイナリロギングのオプションと変数」
セクション17.4.1.19「レプリケーションとマスターまたはスレーブシャットダウン」
セクション18.6.5「レプリケーションのための MySQL Cluster の準備」
セクション8.12.5.7「レプリケーションスレーブ SQL スレッドの状態」
セクション8.12.5.6「レプリケーションスレーブの I/O スレッド状態」
セクション17.1.4.3「レプリケーションスレーブのオプションと変数」
セクション8.12.5.8「レプリケーションスレーブ接続スレッドの状態」
セクション17.1「レプリケーション構成」
セクション17.1.1.7「新しいマスターとスレーブを使用したレプリケーションのセットアップ」
セクション17.1.1.8「既存のデータによるレプリケーションのセットアップ」
セクション13.3.3「暗黙的なコミットを発生させるステートメント」
セクション17.1.2.2「行ベースロギングおよびレプリケーションの使用」
セクション17.3.9「遅延レプリケーション」

CHANGE MASTER TO ... MASTER_LOG_FILE = ... MASTER_LOG_POS = ...

セクション17.1.4.5「グローバルトランザクション ID のオプションと変数」

CHECK TABLE

セクション13.1.7.1「ALTER TABLE パーティション操作」
セクション15.5「ARCHIVE ストレージエンジン」
セクション13.7.2.2「CHECK TABLE 構文」
セクション13.1.15「CREATE PROCEDURE および CREATE FUNCTION 構文」
セクション13.1.20「CREATE VIEW 構文」
セクション4.6.1「innochecksum — オフライン InnoDB ファイルチェックサムユーティリティー」
セクション14.19「InnoDB のトラブルシューティング」
セクション14.16「InnoDB のバックアップとリカバリ」
セクション15.7「MERGE ストレージエンジン」
セクション7.6「MyISAM テーブルの保守とクラッシュリカバリ」
セクション7.6.3「MyISAM テーブルの修復方法」
セクション15.2.4.1「MyISAM テーブルの破損」
セクション7.6.5「MyISAM テーブル保守スケジュールのセットアップ」
セクション4.6.3「myisamchk — MyISAM テーブルメンテナンスユーティリティー」
セクション2.11.1.3「MySQL 5.5 から 5.6 へのアップグレード」
セクションA.6「MySQL 5.6 FAQ: ビュー」
セクションB.5.2.9「MySQL サーバーが存在しなくなりました」
セクション23.7.7.70「mysql_store_result()」
セクション4.4.7「mysql_upgrade — MySQL テーブルのチェックとアップグレード」
セクション23.7.7.72「mysql_use_result()」
セクション4.5.3「mysqlcheck — テーブル保守プログラム」
セクション11.3.4「YEAR(2) の制限と YEAR(4) への移行」
セクション17.1.4.5「グローバルトランザクション ID のオプションと変数」
セクション5.1.3「サーバーコマンドオプション」
セクション5.1.4「サーバーシステム変数」

セクションD.3「サーバー側のカーソルの制約」
セクションD.1「ストアドプログラムの制約」
セクション5.2.5「スロークエリーログ」
セクション2.11.4「テーブルまたはインデックスの再作成または修復」
セクション2.11.3「テーブルまたはインデックスの再構築が必要かどうかのチェック」
セクション19.6「パーティショニングの制約と制限」
セクション19.3.4「パーティションの保守」
セクションD.5「ビューの制約」
セクション5.2.1「一般クエリーログおよびスロークエリーログの出力先の選択」
セクション8.10.5「外部ロック」
セクション13.3.3「暗黙的なコミットを発生させるステートメント」
セクション1.7.1「標準 SQL に対する MySQL 拡張機能」
セクション1.6「質問またはバグをレポートする方法」
セクション15.2.4.2「適切に閉じられなかったテーブルの問題」

CHECK TABLE ... EXTENDED

セクション13.7.2.2「CHECK TABLE 構文」

CHECK TABLE ... FOR UPGRADE

セクション13.7.2.5「REPAIR TABLE 構文」
セクション2.11.3「テーブルまたはインデックスの再構築が必要かどうかのチェック」

CHECKSUM TABLE

セクション13.7.2.3「CHECKSUM TABLE 構文」
セクション13.1.17「CREATE TABLE 構文」

COMMIT

セクション3.6.9「AUTO_INCREMENT の使用」
セクション13.1.15「CREATE PROCEDURE および CREATE FUNCTION 構文」
セクション14.19.4「InnoDB のエラー処理」
セクション14.2.2「InnoDB のトランザクションモデルおよびロック」
セクション14.13.14「InnoDB の読み取り専用トランザクションの最適化」
セクション14.12「InnoDB の起動オプションおよびシステム変数」
セクション8.5.4「InnoDB テーブルの一括データロード」
セクション14.6.7「InnoDB テーブル上の制限」
セクション14.6.4「MyISAM から InnoDB へのテーブルの変換」
セクション18.3.4.3「MySQL Cluster のシステム変数」
セクション14.2.1「MySQL および ACID モデル」
セクション13.3「MySQL トランザクションおよびロックステートメント」
セクション4.5.4「mysqldump — データベースバックアッププログラム」
セクション13.3.4「SAVEPOINT、ROLLBACK TO SAVEPOINT、および RELEASE SAVEPOINT 構文」
セクション13.3.1「START TRANSACTION、COMMIT、および ROLLBACK 構文」
セクション5.1.4「サーバーシステム変数」
セクション5.1.6「サーバーステータス変数」
セクション20.7「ストアドプログラムのバイナリロギング」
セクション1.7.2.3「トランザクションおよびアトミック操作の違い」
セクション14.6.3「トランザクションを使用した DML 操作のグループ化」
セクション20.3.1「トリガーの構文と例」
セクション5.2.4「バイナリログ」
セクション17.4.1.31「レプリケーションとトランザクション」
セクション17.1.4.3「レプリケーションスレーブのオプションと変数」
セクション17.1.1.4「レプリケーションマスターバイナリログ座標の取得」
セクション13.3.3「暗黙的なコミットを発生させるステートメント」
セクション14.2.9「暗黙的なトランザクションコミットとロールバック」

CREATE DATABASE

セクション23.7.6「C API 関数の概要」
セクション13.1.10「CREATE DATABASE 構文」
セクション18.1.6.8「MySQL Cluster に限定された問題」
セクション18.6.9「MySQL Cluster レプリケーションを使用した MySQL Cluster バックアップ」
セクション23.7.7.8「mysql_create_db()」
セクション4.6.8「mysqlbinlog — バイナリログファイルを処理するためのユーティリティー」
セクション4.5.4「mysqldump — データベースバックアッププログラム」
セクション7.4.1「mysqldump による SQL フォーマットでのデータのダンプ」
セクション13.7.5.8「SHOW CREATE DATABASE 構文」
セクション10.1.9.3「SHOW ステートメントと INFORMATION_SCHEMA」
セクション7.4.2「SQL フォーマットバックアップのリロード」
セクション10.1.5「アプリケーションの文字セットおよび照合順序の構成」
セクション17.2.3「サーバーがレプリケーションフィルタリングルールをどのように評価するか」
セクションB.3「サーバーのエラーコードおよびメッセージ」
セクション10.1.3.1「サーバー文字セットおよび照合順序」
セクション7.4.5.2「サーバー間でのデータベースのコピー」
セクション17.2.3.1「データベースレベルレプリケーションオプションおよびバイナリロギングオプションの評価」
セクション10.1.3.2「データベース文字セットおよび照合順序」
セクション7.1「バックアップとリカバリの種類」
セクション22.4「パフォーマンススキーマインストゥルメント命名規則」
セクション17.1.4.3「レプリケーションスレーブのオプションと変数」
セクション13.3.3「暗黙的なコミットを発生させるステートメント」
セクション1.7.1「標準 SQL に対する MySQL 拡張機能」
セクション9.2.2「識別子の大文字と小文字の区別」

CREATE DATABASE dbx

セクション17.2.3「サーバーがレプリケーションフィルタリングルールをどのように評価するか」

CREATE DATABASE IF NOT EXISTS

セクション17.4.1.4「CREATE ... IF NOT EXISTS ステートメントのレプリケーション」

CREATE EVENT

セクション13.1.2「ALTER EVENT 構文」
セクション13.1.11「CREATE EVENT 構文」
セクション17.4.1.7「CURRENT_USER() のレプリケーション」
セクション21.7「INFORMATION_SCHEMA EVENTS テーブル」
セクション13.7.5.9「SHOW CREATE EVENT 構文」
セクション20.4.6「イベントスケジューラと MySQL 権限」
セクション20.4.4「イベントメタデータ」
セクション20.4.3「イベント構文」
セクション20.7「ストアドプログラムのバイナリロギング」
セクションD.1「ストアドプログラムの制約」
セクション17.4.1.11「呼び出される機能のレプリケーション」
セクション12.14「情報関数」
セクション13.3.3「暗黙的なコミットを発生させるステートメント」

CREATE EVENT IF NOT EXISTS

セクション17.4.1.4「CREATE ... IF NOT EXISTS ステートメントのレプリケーション」

CREATE FULLTEXT INDEX

セクション8.5.4「InnoDB テーブルの一括データロード」

CREATE FUNCTION

セクション13.1.4「ALTER FUNCTION 構文」
セクション13.1.12「CREATE FUNCTION 構文」
セクション13.1.15「CREATE PROCEDURE および CREATE FUNCTION 構文」
セクション17.4.1.7「CURRENT_USER() のレプリケーション」
セクション13.7.3.2「DROP FUNCTION 構文」
セクション12.17.1「Enterprise Encryption のインストール」
セクション2.11.1「MySQL のアップグレード」
セクション24.3「MySQL への新しい関数の追加」
セクション1.8.1「MySQL への貢献者」
セクション4.5.4「mysqldump — データベースバックアッププログラム」
セクション20.7「ストアドプログラムのバイナリロギング」
セクション20.2.1「ストアドルーチンの構文」
セクション13.7.3.1「ユーザー定義関数のための CREATE FUNCTION 構文」
セクション24.3.2.5「ユーザー定義関数のコンパイルおよびインストール」
セクション24.3.2.6「ユーザー定義関数のセキュリティー上の予防措置」
セクション24.3.2.1「単純な関数のための UDF の呼び出しシーケンス」
セクション17.4.1.11「呼び出される機能のレプリケーション」
セクション12.14「情報関数」
セクション13.3.3「暗黙的なコミットを発生させるステートメント」
セクション9.2.4「関数名の構文解析と解決」

CREATE INDEX

セクション13.1.13「CREATE INDEX 構文」
セクション14.2.13.3「FULLTEXT インデックス」
セクション14.18.5.4「InnoDB memcached プラグインのトランザクション動作の制御」
セクション14.19.5「InnoDB のエラーコード」
セクション14.12「InnoDB の起動オプションおよびシステム変数」
セクション14.7.5「InnoDB テーブルでの圧縮の動作」
セクション8.7「MEMORY テーブルの最適化」
セクション13.1.7.2「MySQL Cluster での ALTER TABLE オンライン操作」
セクション18.1.6.6「MySQL Cluster の未サポート機能または欠落している機能」
MySQL 用語集
セクション14.11.7「オンライン DDL でのクラッシュリカバリの動作のしくみ」
セクション14.11.2「オンライン DDL でのパフォーマンスと並列性に関する考慮事項」
セクション14.11.5「オンライン DDL の例」
セクション14.11.1「オンライン DDL の概要」
セクション5.1.3「サーバーコマンドオプション」
セクション5.1.4「サーバーシステム変数」
セクション5.2.5「スロークエリーログ」
セクション14.7.2「テーブル圧縮の有効化」
セクション12.9「全文検索関数」
セクション13.3.3「暗黙的なコミットを発生させるステートメント」
セクション11.5.3.6「空間インデックスの作成」

CREATE LOGFILE GROUP

セクション13.1.3「ALTER LOGFILE GROUP 構文」
セクション13.1.14「CREATE LOGFILE GROUP 構文」
セクション13.1.18「CREATE TABLESPACE 構文」
セクション21.30.1「INFORMATION_SCHEMA FILES テーブル」
セクション18.1.6.8「MySQL Cluster に限定された問題」
セクション18.5.12.1「MySQL Cluster ディスクデータオブジェクト」
セクション18.3.2.6「MySQL Cluster データノードの定義」
セクション4.5.4「mysqldump — データベースバックアッププログラム」

CREATE OR REPLACE VIEW

セクション13.1.9「ALTER VIEW 構文」
セクション13.1.20「CREATE VIEW 構文」
セクションD.5「ビューの制約」

CREATE PROCEDURE

セクション13.1.5「ALTER PROCEDURE 構文」
セクション13.2.1「CALL 構文」
セクション13.1.15「CREATE PROCEDURE および CREATE FUNCTION 構文」
セクション17.4.1.7「CURRENT_USER() のレプリケーション」
セクション4.5.4「mysqldump — データベースバックアッププログラム」
セクション20.7「ストアドプログラムのバイナリロギング」
セクション20.2.1「ストアドルーチンの構文」
セクション17.4.1.11「呼び出される機能のレプリケーション」
セクション12.14「情報関数」
セクション13.3.3「暗黙的なコミットを発生させるステートメント」
セクション6.2.2「権限システム付与テーブル」

CREATE SCHEMA

セクション13.1.10「CREATE DATABASE 構文」

セクション18.1.6.8「MySQL Cluster に限定された問題」
セクション18.6.9「MySQL Cluster レプリケーションを使用した MySQL Cluster バックアップ」

CREATE SERVER

セクション13.1.6「ALTER SERVER 構文」
セクション15.8.2.2「CREATE SERVER を使用した FEDERATED テーブルの作成」
セクション17.4.1.6「CREATE SERVER、ALTER SERVER、および DROP SERVER のレプリケーション」
セクション15.8.3「FEDERATED ストレージエンジンの注記とヒント」
セクション15.8.2「FEDERATED テーブルの作成方法」
セクション13.7.6.3「FLUSH 構文」
セクション8.11.4.1「MySQL のメモリーの使用方法」
セクション17.1.4.5「グローバルトランザクション ID のオプションと変数」
セクション13.3.3「暗黙的なコミットを発生させるステートメント」

CREATE TABLE

セクション13.1.7.3「ALTER TABLE の例」
セクション13.1.7.1「ALTER TABLE パーティション操作」
セクション13.1.7「ALTER TABLE 構文」
セクション15.5「ARCHIVE ストレージエンジン」
セクション3.6.9「AUTO_INCREMENT の使用」
セクション15.8.2.1「CONNECTION を使用した FEDERATED テーブルの作成」
セクション13.1.11「CREATE EVENT 構文」
セクション13.1.13「CREATE INDEX 構文」
セクション13.1.16「CREATE SERVER 構文」
セクション17.4.1.5「CREATE TABLE ... SELECT ステートメントのレプリケーション」
セクション13.1.17.1「CREATE TABLE ... SELECT 構文」
セクション13.1.17「CREATE TABLE 構文」
セクション11.4.4「ENUM 型」
セクション1.7.3.2「FOREIGN KEY の制約」
セクション14.2.13.3「FULLTEXT インデックス」
セクション19.2.4「HASH パーティショニング」
セクション13.8.3「HELP 構文」
セクション21.29.14「INFORMATION_SCHEMA INNODB_SYS_DATAFILES テーブル」
セクション21.29.7「INFORMATION_SCHEMA INNODB_SYS_TABLES テーブル」
セクション21.29.15「INFORMATION_SCHEMA INNODB_SYS_TABLESPACES テーブル」
セクション21.13「INFORMATION_SCHEMA PARTITIONS テーブル」
セクション8.5.6「InnoDB DDL 操作の最適化」
セクション14.14.3「InnoDB INFORMATION_SCHEMA システムテーブル」
セクション14.6.6「InnoDB と FOREIGN KEY 制約」
セクション14.17「InnoDB と MySQL レプリケーション」
セクション14.19.5「InnoDB のエラーコード」
セクション14.19「InnoDB のトラブルシューティング」
セクション14.9「InnoDB の行ストレージと行フォーマット」
セクション14.12「InnoDB の起動オプションおよびシステム変数」
セクション14.7.5「InnoDB テーブルでの圧縮の動作」
セクション14.13.17「InnoDB テーブルに対する ANALYZE TABLE の複雑さの推定」
セクション14.6.1「InnoDB テーブルの作成」
セクション14.19.3「InnoDB データディクショナリの操作のトラブルシューティング」
セクション14.15.2「InnoDB モニターの有効化」
セクション14.1「InnoDB 入門」
セクション14.9.1「InnoDB 行ストレージの概要」
セクション19.2.5「KEY パーティショニング」
セクション19.2.2「LIST パーティショニング」
セクション13.2.7「LOAD XML 構文」
セクション15.3「MEMORY ストレージエンジン」
セクション14.6.4「MyISAM から InnoDB へのテーブルの変換」
セクション15.2「MyISAM ストレージエンジン」
セクション15.2.3「MyISAM テーブルのストレージフォーマット」
セクション2.11.1.3「MySQL 5.5 から 5.6 へのアップグレード」
セクションA.10「MySQL 5.6 FAQ: MySQL Cluster」
セクション1.4「MySQL 5.6 の新機能」
セクション13.1.7.2「MySQL Cluster での ALTER TABLE オンライン操作」
セクション18.1.6.1「MySQL Cluster での SQL 構文の不適合」
セクション18.1.6.3「MySQL Cluster でのトランザクション処理に関する制限」
セクション18.1.6.8「MySQL Cluster に限定された問題」
セクション18.2.4「MySQL Cluster の初期構成」
セクション18.1.6.6「MySQL Cluster の未サポート機能または欠落している機能」
セクション18.5.12.1「MySQL Cluster ディスクデータオブジェクト」
セクション18.3.2.6「MySQL Cluster データノードの定義」
セクション18.3.2.1「MySQL Cluster 構成の基本的な例」
セクション8.4.4「MySQL が内部一時テーブルを使用する仕組み」
セクション6.2.1「MySQL で提供される権限」
セクション4.5.1.1「mysql のオプション」
セクション19.1「MySQL のパーティショニングの概要」
セクション5.2.4.4「mysql データベーステーブルへの変更に対するロギング形式」
セクション19.2.7「MySQL パーティショニングによる NULL の扱い」
MySQL 用語集
セクション4.5.4「mysqldump — データベースバックアッププログラム」
セクション7.4.3「mysqldump による区切りテキストフォーマットでのデータのダンプ」
セクション18.5.15「NDB API 統計のカウンタと変数」
セクション18.4.6「ndb_blob_tool — MySQL Cluster テーブルの BLOB および TEXT カラムのチェックおよび修復」
セクション19.2.3.1「RANGE COLUMNS パーティショニング」
セクション19.3.1「RANGE および LIST パーティションの管理」
セクション19.2.1「RANGE パーティショニング」
セクション13.2.8「REPLACE 構文」
セクション13.7.5.6「SHOW COLUMNS 構文」
セクション13.7.5.12「SHOW CREATE TABLE 構文」
セクション13.7.5.16「SHOW ENGINE 構文」
セクション13.7.5.37「SHOW TABLE STATUS 構文」
セクション13.7.5.41「SHOW WARNINGS 構文」
セクション10.1.9.3「SHOW ステートメントと INFORMATION_SCHEMA」
セクション14.7.7「SQL 圧縮構文の警告とエラー」
セクション13.1.33「TRUNCATE TABLE 構文」

セクション13.7.3.4「UNINSTALL PLUGIN 構文」
Unix 上の MyISAM へのシンボリックリンクの使用
セクションD.10.6「Windows プラットフォームの制限」
セクションB.1「エラー情報のソース」
セクション14.11.5「オンライン DDL の例」
セクション14.11.1「オンライン DDL の概要」
セクション8.3.4「カラムインデックス」
セクション10.1.3.4「カラム文字セットおよび照合順序」
セクション19.2.6「サブパーティショニング」
セクション5.1.7「サーバー SQL モード」
セクション5.1.3「サーバーコマンドオプション」
セクション5.1.4「サーバーシステム変数」
セクション8.11.3.1「シンボリックリンクの使用」
セクション13.2.10.1「スカラーオペランドとしてのサブクエリー」
セクション19.6.2「ストレージエンジンに関連するパーティショニング制限」
セクション15.1「ストレージエンジンの設定」
セクション3.3.2「テーブルの作成」
セクション14.9.2「テーブルの行フォーマットの指定」
セクション3.3.3「テーブルへのデータのロード」
セクション2.11.4「テーブルまたはインデックスの再作成または修復」
セクションD.10.3「テーブルサイズの制限」
セクション14.5.4「テーブルスペースの位置の指定」
セクション13.3.5.3「テーブルロックの制限と条件」
セクション14.7.2「テーブル圧縮の有効化」
セクション14.7.1「テーブル圧縮の概要」
セクション10.1.3.3「テーブル文字セットおよび照合順序」
セクション8.4.1「データサイズの最適化」
セクション3.4「データベースとテーブルに関する情報の取得」
セクション7.2「データベースバックアップ方法」
セクション10.1.3.2「データベース文字セットおよび照合順序」
セクション17.1.4.4「バイナリログのオプションと変数」
セクション5.2.4.2「バイナリログ形式の設定」
セクション7.1「バックアップとリカバリの種類」
セクション7.4「バックアップへの mysqldump の使用」
セクション19.6「パーティショニングの制約と制限」
セクション19.6.1「パーティショニングキー、主キー、および一意キー」
セクション19.2「パーティショニングタイプ」
セクション19.3「パーティション管理」
マスターまたはスレーブでカラムが多い場合のレプリケーション
セクション17.4.1.1「レプリケーションと AUTO_INCREMENT」
セクション17.4.1.10「レプリケーションと DIRECTORY テーブルオプション」
セクション17.4.1.15「レプリケーションとシステム関数」
セクション17.4.1.13「レプリケーションと小数秒サポート」
セクション17.4.1.3「レプリケーションと文字セット」
セクション5.2.1「一般クエリーログおよびスロークエリーログの出力先の選択」
第15章「代替ストレージエンジン」
セクション12.9「全文検索関数」
セクション7.4.4「区切りテキストフォーマットバックアップのリロード」
セクション1.7.2.4「外部キーの違い」
セクション13.1.17.2「外部キー制約の使用」
セクション14.6.5.1「従来の InnoDB の自動インクリメントロック」
セクション12.14「情報関数」
セクション11.1.3「文字列型の概要」
セクション13.1.17.3「暗黙のカラム指定の変更」
セクション13.3.3「暗黙的なコミットを発生させるステートメント」
セクション1.7.1「標準 SQL に対する MySQL 拡張機能」
セクション14.13.16.1「永続的オプティマイザ統計のパラメータの構成」
セクション17.3.2「異なるマスターおよびスレーブストレージエンジンでレプリケーションを使用する」
セクション11.5.3.6「空間インデックスの作成」
セクション11.5.3.2「空間カラムの作成」
セクション3.3.4.9「複数のテーブルの使用」
セクション9.2.2「識別子の大文字と小文字の区別」
セクション19.6.3「関数に関連するパーティショニング制限」

CREATE TABLE ... KEY ()

セクション2.11.1.3「MySQL 5.5 から 5.6 へのアップグレード」

CREATE TABLE ... LIKE

セクション13.1.17「CREATE TABLE 構文」
セクション13.3.5.3「テーブルロックの制限と条件」

CREATE TABLE ... SELECT

セクション17.4.1.5「CREATE TABLE ... SELECT ステートメントのレプリケーション」
セクション13.1.17.1「CREATE TABLE ... SELECT 構文」
セクション17.1.3.4「GTID ベースレプリケーションの制約」
セクションB.5.8「MySQL の既知の問題」
セクション5.2.4.4「mysql データベーステーブルへの変更に対するロギング形式」
セクション17.4.2「MySQL バージョン間のレプリケーション互換性」
セクション1.7.2.1「SELECT INTO TABLE の違い」
セクション13.2.9「SELECT 構文」
セクション12.10「キャスト関数と演算子」
セクション17.1.4.5「グローバルトランザクション ID のオプションと変数」
セクション17.1.2.1「ステートメントベースおよび行ベースレプリケーションのメリットとデメリット」
セクション20.7「ストアドプログラムのバイナリロギング」
セクション14.2.4「一貫性非ロック読み取り」
セクション13.3.3「暗黙的なコミットを発生させるステートメント」

CREATE TABLE ... SELECT ...

セクション14.2.8「InnoDB のさまざまな SQL ステートメントで設定されたロック」
セクション19.3.1「RANGE および LIST パーティションの管理」

CREATE TABLE IF NOT EXISTS

セクション17.4.1.4「CREATE ... IF NOT EXISTS ステートメントのレプリケーション」
セクション13.1.17.1「CREATE TABLE ... SELECT 構文」

CREATE TABLE IF NOT EXISTS ... LIKE

セクション17.4.1.4「CREATE ... IF NOT EXISTS ステートメントのレプリケーション」

CREATE TABLE IF NOT EXISTS ... SELECT

セクション17.4.1.4「CREATE ... IF NOT EXISTS ステートメントのレプリケーション」
セクション13.1.17.1「CREATE TABLE ... SELECT 構文」

CREATE TABLE new_table SELECT ... FROM old_table ...

セクション13.1.17.1「CREATE TABLE ... SELECT 構文」
セクション13.2.9「SELECT 構文」

CREATE TABLESPACE

セクション13.1.8「ALTER TABLESPACE 構文」
セクション13.1.17「CREATE TABLE 構文」
セクション13.1.18「CREATE TABLESPACE 構文」
セクション13.1.29「DROP TABLESPACE 構文」
セクション21.30.1「INFORMATION_SCHEMA FILES テーブル」
セクション18.1.6.8「MySQL Cluster に限定された問題」
セクション18.5.12.1「MySQL Cluster ディスクデータオブジェクト」
セクション18.3.2.6「MySQL Cluster データノードの定義」
セクション4.5.4「mysqldump — データベースバックアッププログラム」

CREATE TEMPORARY TABLE

セクション13.7.1.4「GRANT 構文」
セクション17.1.3.4「GTID ベースレプリケーションの制約」
セクションA.10「MySQL 5.6 FAQ: MySQL Cluster」
セクション6.2.1「MySQL で提供される権限」
セクション4.6.8「mysqlbinlog — バイナリログファイルを処理するためのユーティリティー」
セクション17.1.4.5「グローバルトランザクション ID のオプションと変数」
セクション5.1.4「サーバーシステム変数」
セクション15.1「ストレージエンジンの設定」
セクション7.5「バイナリログを使用したポイントインタイム (増分) リカバリ」
セクション13.3.3「暗黙的なコミットを発生させるステートメント」

CREATE TRIGGER

セクション13.1.19「CREATE TRIGGER 構文」
セクション17.4.1.7「CURRENT_USER() のレプリケーション」
EXISTS 戦略によるサブクエリーの最適化
セクションA.5「MySQL 5.6 FAQ: トリガー」
セクション4.5.4「mysqldump — データベースバックアッププログラム」
セクション13.7.5.13「SHOW CREATE TRIGGER 構文」
セクション20.7「ストアドプログラムのバイナリロギング」
セクション20.3.1「トリガーの構文と例」
セクション17.4.1.11「呼び出される機能のレプリケーション」
セクション12.14「情報関数」
セクション13.3.3「暗黙的なコミットを発生させるステートメント」

CREATE USER

セクション13.7.1.2「CREATE USER 構文」
セクション13.7.6.3「FLUSH 構文」
セクション13.7.1.4「GRANT 構文」
セクション5.1.9「IPv6 サポート」
セクション5.1.9.3「IPv6 ローカルホストアドレスを使用した接続」
セクション1.4「MySQL 5.6 の新機能」
セクション18.5.11.2「MySQL Cluster と MySQL 権限」
セクション6.1.2.4「MySQL でのパスワードハッシュ」
セクション6.2.1「MySQL で提供される権限」
セクション8.11.4.1「MySQL のメモリーの使用方法」
セクション6.2「MySQL アクセス権限システム」
セクション5.2.4.4「mysql データベーステーブルへの変更に対するロギング形式」
PAM 認証プラグインのインストール
PAM 認証プラグインの使用
セクション6.3.8.4「SHA-256 認証プラグイン」
Windows 認証プラグインのインストール
Windows 認証プラグインの使用
セクション6.3.5「アカウントパスワードの割り当て」
セクション6.2.3「アカウント名の指定」
セクション6.2.7「アクセス拒否エラーの原因」
セクション5.1.3「サーバーコマンドオプション」
セクション5.1.4「サーバーシステム変数」
セクション6.1.2.3「パスワードおよびロギング」
セクション6.1.2.1「パスワードセキュリティーのためのエンドユーザーガイドライン」
セクション24.2.3.8「パスワード検証プラグイン」
セクション6.1.2.6「パスワード検証プラグイン」
セクション6.3.7「プラガブル認証」
セクション6.3.9「プロキシユーザー」
セクション6.3.2「ユーザーアカウントの追加」
セクション6.3.1「ユーザー名とパスワード」
セクション17.1.1.3「レプリケーション用ユーザーの作成」
セクション12.13「暗号化関数と圧縮関数」
セクション13.3.3「暗黙的なコミットを発生させるステートメント」
セクション6.2.2「権限システム付与テーブル」
認証プラグインでのプロキシユーザーサポートの実装

CREATE VIEW

セクション13.1.9「ALTER VIEW 構文」
セクション13.1.20「CREATE VIEW 構文」
セクション17.4.1.7「CURRENT_USER() のレプリケーション」
セクション21.28「INFORMATION_SCHEMA VIEWS テーブル」
セクション6.2.1「MySQL で提供される権限」
セクション13.7.5.14「SHOW CREATE VIEW 構文」
セクション9.2「スキーマオブジェクト名」
セクション13.3.5.3「テーブルロックの制限と条件」
セクション19.6.4「パーティショニングとロック」
セクションD.5「ビューの制約」
セクション20.5.1「ビューの構文」
セクション20.5.2「ビュー処理アルゴリズム」
セクション8.12.5.2「一般的なスレッドの状態」
セクション12.14「情報関数」
セクション13.3.3「暗黙的なコミットを発生させるステートメント」
セクション20.5.3「更新可能および挿入可能なビュー」

D

[索引の先頭]

DEALLOCATE PREPARE

セクション13.5.3「DEALLOCATE PREPARE 構文」
セクション13.5.1「PREPARE 構文」
セクション5.1.6「サーバーステータス変数」
セクションD.1「ストアドプログラムの制約」
セクション13.5「準備済みステートメントのための SQL 構文」

DECLARE

セクション13.1.15「CREATE PROCEDURE および CREATE FUNCTION 構文」
セクション13.6.3「DECLARE 構文」
セクション13.6.7.3「GET DIAGNOSTICS 構文」
セクション13.6.7.5「SIGNAL 構文」
セクション13.6.4「ストアドプログラム内の変数」

DECLARE ... CONDITION

セクション13.6.7.1「DECLARE ... CONDITION 構文」
セクション13.6.7.2「DECLARE ... HANDLER 構文」
セクション13.6.7.5「SIGNAL 構文」
セクション13.6.7「条件の処理」

DECLARE ... HANDLER

セクション13.6.7.1「DECLARE ... CONDITION 構文」
セクション13.6.7.2「DECLARE ... HANDLER 構文」
ハンドラ、カーソル、およびステートメントに対するシグナルの影響
セクション13.6.7「条件の処理」

DELETE

--safe-updates オプションの使用
セクション13.1.7.1「ALTER TABLE パーティション操作」
セクション15.5「ARCHIVE ストレージエンジン」
セクション3.6.9「AUTO_INCREMENT の使用」
セクション15.6「BLACKHOLE ストレージエンジン」
セクション23.7.10「C API プリペアドステートメント関数の概要」
セクション23.7.6「C API 関数の概要」
セクション13.1.15「CREATE PROCEDURE および CREATE FUNCTION 構文」
セクション13.1.19「CREATE TRIGGER 構文」
セクション13.1.20「CREATE VIEW 構文」
セクション13.2.2「DELETE 構文」
セクション8.2.2「DML ステートメントの最適化」
セクション8.8.3「EXPLAIN EXTENDED 出力フォーマット」
セクション8.8.1「EXPLAIN によるクエリーの最適化」
セクション8.8.2「EXPLAIN 出力フォーマット」
セクション13.8.2「EXPLAIN 構文」
セクション15.8.3「FEDERATED ストレージエンジンの注記とヒント」
セクション13.7.1.4「GRANT 構文」
セクション21.28「INFORMATION_SCHEMA VIEWS テーブル」
第21章「INFORMATION_SCHEMA テーブル」
セクション14.17「InnoDB と MySQL レプリケーション」
セクション14.2.8「InnoDB のさまざまな SQL ステートメントで設定されたロック」
セクション14.19.2「InnoDB のリカバリの強制的な実行」
セクション14.12「InnoDB の起動オプションおよびシステム変数」
セクション13.2.9.2「JOIN 構文」
セクション13.7.6.4「KILL 構文」
セクション19.2.2「LIST パーティショニング」
セクション15.3「MEMORY ストレージエンジン」
セクション15.7「MERGE ストレージエンジン」
セクション15.7.2「MERGE テーブルの問題点」
セクション14.6.4「MyISAM から InnoDB へのテーブルの変換」
セクション8.6.2「MyISAM テーブルの一括データロード」
セクション1.4「MySQL 5.6 の新機能」
セクション18.1.6.3「MySQL Cluster でのトランザクション処理に関する制限」
セクション18.5.11.3「MySQL Cluster と MySQL のセキュリティー手順」
セクション18.5.5「MySQL Cluster のローリング再起動の実行」
セクション18.1.6.2「MySQL Cluster の制限と標準の MySQL の制限との違い」
セクション18.5.12.1「MySQL Cluster ディスクデータオブジェクト」
セクション6.2.1「MySQL で提供される権限」
セクション8.2.1.2「MySQL の WHERE 句の最適化の方法」
セクション4.5.1.1「mysql のオプション」
セクション19.1「MySQL のパーティショニングの概要」
セクション1.3.2「MySQL の主な機能」
セクションB.5.8「MySQL の既知の問題」
セクション6.2「MySQL アクセス権限システム」
セクション5.2.4.4「mysql データベーステーブルへの変更に対する口ギング形式」
MySQL 用語集
セクション23.7.7.1「mysql_affected_rows()」
セクション23.7.7.48「mysql_num_rows()」
セクション23.7.15.1「mysql_query() が成功を返したあとに mysql_store_result() が NULL を返すことがあるのはなぜか」
セクション23.7.11.10「mysql_stmt_execute()」
セクション23.7.11.13「mysql_stmt_field_count()」
セクション23.7.11.18「mysql_stmt_num_rows()」
セクション18.4.9「ndb_delete_all — NDB テーブルからのすべての行の削除」
セクション14.7.6「OLTP ワークロードの圧縮」
セクション13.7.2.4「OPTIMIZE TABLE 構文」
セクション19.3.1「RANGE および LIST パーティションの管理」
セクション19.2.1「RANGE パーティショニング」
セクション13.7.1.6「REVOKE 構文」
セクション13.3.6「SET TRANSACTION 構文」
セクション13.1.33「TRUNCATE TABLE 構文」
セクション3.3.4.1「すべてのデータの選択」
セクション6.2.7「アクセス拒否エラーの原因」
セクション14.11.5「オンライン DDL の例」
セクション14.11.1「オンライン DDL の概要」
セクション23.7.15.2「クエリーからどんな結果を得ることができるか」
セクション8.9.3.1「クエリーキャッシュの動作」
セクション8.12.5.4「クエリーキャッシュスレッドの状態」
セクション13.2.10.9「サブクエリーのエラー」
セクション13.2.10.11「サブクエリーの結合としての書き換え」
セクション13.2.10「サブクエリー構文」
セクション5.1.3「サーバーコマンドオプション」
セクション5.1.4「サーバーシステム変数」
セクション5.1.6「サーバーステータス変数」
セクション5.4.1.12「ステートメントプローブ」
セクション17.1.2.1「ステートメントベースおよび行ベースレプリケーションのメリットとデメリット」

セクション8.10.2「テーブルロックの問題」
セクション20.3.1「トリガーの構文と例」
セクション5.2.4「バイナリログ」
セクション17.1.4.4「バイナリログのオプションと変数」
セクション19.4「パーティションプルーニング」
セクション19.5「パーティション選択」
セクションD.5「ビューの制約」
セクション6.3.2「ユーザーアカウントの追加」
セクション17.4.1.16「レプリケーションと LIMIT」
セクション17.4.1.21「レプリケーションと MEMORY テーブル」
セクション17.4.1.24「レプリケーションとクエリーオプティマイザ」
セクション17.1.4.3「レプリケーションスレーブのオプションと変数」
セクション5.2.1「一般クエリーログおよびスロークエリーログの出力先の選択」
セクション8.12.5.2「一般的なスレッドの状態」
セクション14.2.4「一貫性非ロック読み取り」
セクション9.3「予約語」
セクション12.9.5「全文制限」
セクション8.10.1「内部ロック方法」
セクション13.1.17.2「外部キー制約の使用」
セクション12.14「情報関数」
セクション20.5.3「更新可能および挿入可能なビュー」
セクション1.7.1「標準 SQL に対する MySQL 拡張機能」
セクション6.2.2「権限システム付与テーブル」
セクション6.2.6「権限変更が有効化される時期」
セクション24.2.4.8「監査プラグインの作成」
セクション17.1.2.2「行ベースロギングおよびレプリケーションの使用」
第12章「関数と演算子」
セクションB.5.5.6「関連するテーブルからの行の削除」

DELETE FROM ... WHERE ...

セクション14.2.8「InnoDB のさまざまな SQL ステートメントで設定されたロック」

DELETE FROM a.t

セクション17.1.4.3「レプリケーションスレーブのオプションと変数」

DELETE FROM t1,t2

セクション5.4.1.12「ステートメントプローブ」

DESCRIBE

セクション23.7.5「C API データ構造」
セクション23.7.6「C API 関数の概要」
セクション13.1.17「CREATE TABLE 構文」
セクション13.8.1「DESCRIBE 構文」
セクション13.8.2「EXPLAIN 構文」
セクション21.29.16「INFORMATION_SCHEMA INNODB_BUFFER_PAGE テーブル」
セクション21.29.17「INFORMATION_SCHEMA INNODB_BUFFER_PAGE_LRU テーブル」
セクション21.29.18「INFORMATION_SCHEMA INNODB_BUFFER_POOL_STATS テーブル」
セクション21.29.1「INFORMATION_SCHEMA INNODB_CMP および INNODB_CMP_RESET テーブル」
セクション21.29.2「INFORMATION_SCHEMA INNODB_CMP_PER_INDEX および INNODB_CMP_PER_INDEX_RESET テーブル」
セクション21.29.3「INFORMATION_SCHEMA INNODB_CMPMEM および INNODB_CMPMEM_RESET テーブル」
セクション21.29.25「INFORMATION_SCHEMA INNODB_FT_BEING_DELETED テーブル」
セクション21.29.20「INFORMATION_SCHEMA INNODB_FT_CONFIG テーブル」
セクション21.29.21「INFORMATION_SCHEMA INNODB_FT_DEFAULT_STOPWORD テーブル」
セクション21.29.24「INFORMATION_SCHEMA INNODB_FT_DELETED テーブル」
セクション21.29.23「INFORMATION_SCHEMA INNODB_FT_INDEX_CACHE テーブル」
セクション21.29.22「INFORMATION_SCHEMA INNODB_FT_INDEX_TABLE テーブル」
セクション21.29.6「INFORMATION_SCHEMA INNODB_LOCK_WAITS テーブル」
セクション21.29.5「INFORMATION_SCHEMA INNODB_LOCKS テーブル」
セクション21.29.19「INFORMATION_SCHEMA INNODB_METRICS テーブル」
セクション21.29.9「INFORMATION_SCHEMA INNODB_SYS_COLUMNS テーブル」
セクション21.29.14「INFORMATION_SCHEMA INNODB_SYS_DATAFILES テーブル」
セクション21.29.10「INFORMATION_SCHEMA INNODB_SYS_FIELDS テーブル」
セクション21.29.11「INFORMATION_SCHEMA INNODB_SYS_FOREIGN テーブル」
セクション21.29.12「INFORMATION_SCHEMA INNODB_SYS_FOREIGN_COLS テーブル」
セクション21.29.8「INFORMATION_SCHEMA INNODB_SYS_INDEXES テーブル」
セクション21.29.7「INFORMATION_SCHEMA INNODB_SYS_TABLES テーブル」
セクション21.29.15「INFORMATION_SCHEMA INNODB_SYS_TABLESPACES テーブル」
セクション21.29.13「INFORMATION_SCHEMA INNODB_SYS_TABLESTATS ビュー」
セクション21.29.4「INFORMATION_SCHEMA INNODB_TRX テーブル」
セクション8.2.1.19「LIMIT クエリーの最適化」
セクション23.7.11.28「mysql_stmt_store_result()」
セクション23.7.7.70「mysql_store_result()」
セクション23.7.7.72「mysql_use_result()」
セクション13.7.5.6「SHOW COLUMNS 構文」
セクション21.32「SHOW ステートメントの拡張」
セクション3.3.2「テーブルの作成」
セクション3.4「データベースとテーブルに関する情報の取得」
セクション10.1.12「メタデータ用の UTF-8」
セクション3.6.6「外部キーの使用」
セクション13.1.17.3「暗黙のカラム指定の変更」

DO

セクション13.1.2「ALTER EVENT 構文」
セクション13.1.11「CREATE EVENT 構文」
セクション13.2.3「DO 構文」
セクション21.7「INFORMATION_SCHEMA EVENTS テーブル」
セクション5.2.4.4「mysql データベーステーブルへの変更に対するロギング形式」
セクション12.18「その他の関数」
セクション13.2.10「サブクエリー構文」

セクション20.7「ストアドプログラムのバイナリロギング」
セクションD.1「ストアドプログラムの制約」
セクション19.6.4「パーティショニングとロック」
セクション1.7.1「標準 SQL に対する MySQL 拡張機能」

DROP DATABASE

セクション23.7.6「C API 関数の概要」
セクション13.1.21「DROP DATABASE 構文」
セクション18.1.6.8「MySQL Cluster に限定された問題」
セクション23.7.7.11「mysql_drop_db()」
セクション4.6.8「mysqlbinlog — バイナリログファイルを処理するためのユーティリティー」
セクション4.5.4「mysqldump — データベースバックアッププログラム」
セクション7.4.1「mysqldump による SQL フォーマットでのデータのダンプ」
セクションD.10.6「Windows プラットフォームの制限」
セクション8.9.3.1「クエリーキャッシュの動作」
セクション17.2.3「サーバーがレプリケーションフィルタリングルールをどのように評価するか」
セクション17.2.3.1「データベースレベルレプリケーションオプションおよびバイナリロギングオプションの評価」
セクション7.5「バイナリログを使用したポイントインタイム (増分) リカバリ」
セクション17.1.4.3「レプリケーションスレーブのオプションと変数」
セクション13.3.3「暗黙的なコミットを発生させるステートメント」
セクション1.7.1「標準 SQL に対する MySQL 拡張機能」

DROP DATABASE IF EXISTS

セクション17.4.1.8「DROP ... IF EXISTS ステートメントのレプリケーション」

DROP EVENT

セクション20.4.6「イベントスケジューラと MySQL 権限」
セクション20.4.3「イベント構文」
セクション20.7「ストアドプログラムのバイナリロギング」
セクション17.4.1.11「呼び出される機能のレプリケーション」
セクション13.3.3「暗黙的なコミットを発生させるステートメント」

DROP FUNCTION

セクション13.1.4「ALTER FUNCTION 構文」
セクション13.1.23「DROP FUNCTION 構文」
セクション13.7.3.2「DROP FUNCTION 構文」
セクション13.1.26「DROP PROCEDURE および DROP FUNCTION 構文」
セクション12.17.1「Enterprise Encryption のインストール」
セクション2.11.1「MySQL のアップグレード」
セクション24.3「MySQL への新しい関数の追加」
セクション1.8.1「MySQL への貢献者」
セクション20.7「ストアドプログラムのバイナリロギング」
セクション20.2.1「ストアドルーチンの構文」
セクション13.7.3.1「ユーザー定義関数のための CREATE FUNCTION 構文」
セクション24.3.2.5「ユーザー定義関数のコンパイルおよびインストール」
セクション24.3.2.6「ユーザー定義関数のセキュリティー上の予防措置」
セクション17.4.1.11「呼び出される機能のレプリケーション」
セクション13.3.3「暗黙的なコミットを発生させるステートメント」
セクション9.2.4「関数名の構文解析と解決」

DROP INDEX

セクション13.1.7「ALTER TABLE 構文」
セクション13.1.24「DROP INDEX 構文」
セクション13.1.7.2「MySQL Cluster での ALTER TABLE オンライン操作」
MySQL 用語集
セクション14.11.5「オンライン DDL の例」
セクション14.11.1「オンライン DDL の概要」
セクション5.1.3「サーバーコマンドオプション」
セクション5.1.4「サーバーシステム変数」
セクション5.2.5「スロークエリーログ」
セクション13.3.3「暗黙的なコミットを発生させるステートメント」
セクション1.7.1「標準 SQL に対する MySQL 拡張機能」
セクション11.5.3.6「空間インデックスの作成」

DROP LOGFILE GROUP

セクション13.1.25「DROP LOGFILE GROUP 構文」
セクション18.1.6.8「MySQL Cluster に限定された問題」

DROP PROCEDURE

セクション13.1.5「ALTER PROCEDURE 構文」
セクション20.7「ストアドプログラムのバイナリロギング」
セクション20.2.1「ストアドルーチンの構文」
セクション17.4.1.11「呼び出される機能のレプリケーション」
セクション13.3.3「暗黙的なコミットを発生させるステートメント」

DROP SCHEMA

セクション13.1.21「DROP DATABASE 構文」
セクション18.1.6.8「MySQL Cluster に限定された問題」
セクション5.1.4「サーバーシステム変数」

DROP SERVER

セクション17.4.1.6「CREATE SERVER、ALTER SERVER、および DROP SERVER のレプリケーション」
セクション13.7.6.3「FLUSH 構文」
セクション8.11.4.1「MySQL のメモリーの使用方法」
セクション17.1.4.5「グローバルトランザクション ID のオプションと変数」
セクション13.3.3「暗黙的なコミットを発生させるステートメント」

DROP TABLE

セクション13.1.7「ALTER TABLE 構文」
セクション13.1.19「CREATE TRIGGER 構文」
セクション5.2.6「DDL ログ」
セクション13.1.28「DROP TABLE 構文」
セクション15.8.3「FEDERATED ストレージエンジンの注記とヒント」
セクション8.5.6「InnoDB DDL 操作の最適化」
セクション14.5.2「InnoDB File-Per-Table モード」
セクション14.12「InnoDB の起動オプションおよびシステム変数」
セクション14.19.3「InnoDB データディクショナリの操作のトラブルシューティング」
セクション13.3.5「LOCK TABLES および UNLOCK TABLES 構文」

セクション15.3「MEMORY ストレージエンジン」
セクション15.7「MERGE ストレージエンジン」
セクション15.7.2「MERGE テーブルの問題点」
セクション14.6.4「MyISAM から InnoDB へのテーブルの変換」
セクション18.1.6.8「MySQL Cluster に限定された問題」
セクション18.1.6.2「MySQL Cluster の制限と標準の MySQL の制限との違い」
セクション18.5.13.1「MySQL Cluster データノードのオンライン追加: 一般的な問題」
セクション18.6.9.2「MySQL Cluster レプリケーションを使用したポイントインタイムリカバリ」
セクション18.5.2「MySQL Cluster 管理クライアントのコマンド」
セクション6.2.1「MySQL で提供される権限」
セクション4.5.1.1「mysql のオプション」
MySQL 用語集
セクション4.5.4「mysqldump — データベースバックアッププログラム」
セクション18.4.11「ndb_drop_index — NDB テーブルからのインデックスの削除」
セクション18.4.12「ndb_drop_table — NDB テーブルの削除」
セクション13.6.7.5「SIGNAL 構文」
セクション13.4.2.5「START SLAVE 構文」
セクション13.1.33「TRUNCATE TABLE 構文」
セクション13.7.3.4「UNINSTALL PLUGIN 構文」
セクション8.9.3.1「クエリーキャッシュの動作」
セクション17.1.4.5「グローバルトランザクション ID のオプションと変数」
セクション5.1.3「サーバーコマンドオプション」
セクション5.1.4「サーバーシステム変数」
セクション14.5.4「テーブルスペースの位置の指定」
セクション13.6.7.6「ハンドラのスコープに関するルール」
セクション5.2.4.2「バイナリログ形式の設定」
セクションD.5「ビューの制約」
セクション5.2.1「一般クエリーログおよびスロークエリーログの出力先の選択」
セクション14.2.4「一貫性非ロック読み取り」
セクション24.2.3.2「全文パーサープラグイン」
セクション13.1.17.2「外部キー制約の使用」
セクション12.14「情報関数」
セクション13.3.3「暗黙的なコミットを発生させるステートメント」
セクション1.7.1「標準 SQL に対する MySQL 拡張機能」
診断領域が移入される方法

DROP TABLE IF EXISTS

セクション17.4.1.8「DROP ... IF EXISTS ステートメントのレプリケーション」

DROP TABLE IF EXISTS mysql.user mysql.db mysql.tables_priv mysql.columns_priv mysql.procs_priv

セクション18.5.14「MySQL Cluster の配布された MySQL 権限」

DROP TABLESPACE

セクション13.1.29「DROP TABLESPACE 構文」
セクション18.1.6.8「MySQL Cluster に限定された問題」

DROP TEMPORARY TABLE

セクション17.1.3.4「GTID ベースレプリケーションの制約」

DROP TEMPORARY TABLE IF EXISTS

セクション17.1.2.2「行ベースロギングおよびレプリケーションの使用」

DROP TRIGGER

セクション13.1.30「DROP TRIGGER 構文」
セクションA.5「MySQL 5.6 FAQ: トリガー」
セクション4.5.4「mysqldump — データベースバックアッププログラム」
セクション20.3.1「トリガーの構文と例」
セクション17.4.1.11「呼び出される機能のレプリケーション」
セクション13.3.3「暗黙的なコミットを発生させるステートメント」

DROP USER

セクション17.4.1.7「CURRENT_USER() のレプリケーション」
セクション13.7.1.3「DROP USER 構文」
セクション13.7.6.3「FLUSH 構文」
セクション13.7.1.4「GRANT 構文」
セクション18.5.11.2「MySQL Cluster と MySQL 権限」
セクション6.2.1「MySQL で提供される権限」
セクション8.11.4.1「MySQL のメモリーの使用方法」
セクション13.7.1.6「REVOKE 構文」
セクション20.4.6「イベントスケジューラと MySQL 権限」
セクション6.3.3「ユーザーアカウントの削除」
セクション12.14「情報関数」
セクション13.3.3「暗黙的なコミットを発生させるステートメント」

DROP USER 'x'@'localhost'

認証プラグインの使用

DROP VIEW

セクション13.1.31「DROP VIEW 構文」
セクション13.3.5.3「テーブルロックの制限と条件」
セクションD.5「ビューの制約」
セクション20.5.1「ビューの構文」
セクション13.3.3「暗黙的なコミットを発生させるステートメント」

DROP VIEW IF EXISTS

セクション17.4.1.8「DROP ... IF EXISTS ステートメントのレプリケーション」

E

[索引の先頭]

EXECUTE

セクション23.7.20「C API のプリペアド CALL ステートメントのサポート」
セクション13.2.1「CALL 構文」
セクション13.5.2「EXECUTE 構文」
セクション13.5.1「PREPARE 構文」

セクション5.1.6「サーバーステータス変数」
セクションD.1「ストアドプログラムの制約」
セクションD.5「ビューの制約」
セクション13.5「準備済みステートメントのための SQL 構文」

EXPLAIN

セクション13.1.7「ALTER TABLE 構文」
Batched Key Access 結合
セクション23.7.5「C API データ構造」
セクション23.7.6「C API 関数の概要」
セクション13.1.15「CREATE PROCEDURE および CREATE FUNCTION 構文」
セクション13.8.1「DESCRIBE 構文」
セクション8.2.1.17「DISTINCT の最適化」
EXISTS 戦略によるサブクエリーの最適化
セクション8.8.3「EXPLAIN EXTENDED 出力フォーマット」
セクション8.8.1「EXPLAIN によるクエリーの最適化」
セクション8.8.2「EXPLAIN 出力フォーマット」
セクション13.8.2「EXPLAIN 構文」
セクション13.2.10.8「FROM 句内のサブクエリー」
FROM 句内のサブクエリー (派生テーブル) の最適化
セクション8.2.4「INFORMATION_SCHEMA クエリーの最適化」
第21章「INFORMATION_SCHEMA テーブル」
InnoDB オプティマイザ統計でサンプリングされるページの数の構成
セクション8.2.1.8「IS NULL の最適化」
セクション8.2.1.13「Multi-Range Read の最適化」
セクション1.4「MySQL 5.6 の新機能」
セクション18.3.4.3「MySQL Cluster のシステム変数」
セクション18.3.4.4「MySQL Cluster のステータス変数」
セクション8.4.4「MySQL が内部一時テーブルを使用する仕組み」
セクション1.3.2「MySQL の主な機能」
セクション24.4.1「MySQL サーバーのデバッグ」
第22章「MySQL パフォーマンススキーマ」
セクション23.7.11.28「mysql_stmt_store_result()」
セクション23.7.7.70「mysql_store_result()」
セクション23.7.7.72「mysql_use_result()」
セクション24.4.1.6「mysqld でのエラーの原因を見つけるためのサーバーログの使用」
セクション8.2.1.15「ORDER BY の最適化」
セクション8.2.1.1「SELECT ステートメントの速度」
セクション13.2.9「SELECT 構文」
セクション13.7.5.41「SHOW WARNINGS 構文」
セクション8.2.5「その他の最適化のヒント」
セクション8.3.6「インデックスの使用の確認」
セクション8.2.1.6「インデックスコンディションプッシュダウンの最適化」
セクション13.2.9.3「インデックスヒントの構文」
セクション8.2.1.4「インデックスマージの最適化」
インデックスマージ共通集合アクセスアルゴリズム
セクション8.2.1.7「インデックス拡張の使用」
セクション8.2.1.5「エンジンコンディションプッシュダウンの最適化」
セクションB.5.6「オプティマイザ関連の問題」
セクション8.8「クエリー実行プランの理解」
セクション13.2.10.10「サブクエリーの最適化」
サブクエリー実体化によるサブクエリーの最適化
セクションB.3「サーバーのエラーコードおよびメッセージ」
セクションD.1「ストアドプログラムの制約」
セクション19.3.5「パーティションに関する情報を取得する」
セクション8.2.1.20「フルテーブルスキャンを回避する方法」
マルチパートインデックスの range アクセスメソッド
ルースインデックススキャン
セクションB.5.5.7「一致する行がない場合の問題の解決」
外部結合と準結合の Block Nested Loop アルゴリズム
準結合変換によるサブクエリーの最適化
セクション11.5.3.7「空間インデックスの使用」

EXPLAIN ... SELECT

セクション19.3.5「パーティションに関する情報を取得する」

EXPLAIN EXTENDED

EXISTS 戦略によるサブクエリーの最適化
セクション8.8.3「EXPLAIN EXTENDED 出力フォーマット」
セクション8.8.1「EXPLAIN によるクエリーの最適化」
セクション8.8.2「EXPLAIN 出力フォーマット」
セクション13.8.2「EXPLAIN 構文」
セクション13.7.5.41「SHOW WARNINGS 構文」
セクション8.2.1.5「エンジンコンディションプッシュダウンの最適化」
サブクエリー実体化によるサブクエリーの最適化
準結合変換によるサブクエリーの最適化

EXPLAIN PARTITIONS

セクション8.8.1「EXPLAIN によるクエリーの最適化」
セクション13.8.2「EXPLAIN 構文」
セクション19.3.5「パーティションに関する情報を取得する」

EXPLAIN PARTITIONS SELECT

セクション19.3.5「パーティションに関する情報を取得する」

EXPLAIN PARTITIONS SELECT COUNT()

セクション19.2.1「RANGE パーティショニング」

EXPLAIN SELECT

セクション8.8.2「EXPLAIN 出力フォーマット」
セクション13.2.10.8「FROM 句内のサブクエリー」
FROM 句内のサブクエリー (派生テーブル) の最適化
セクション14.2.11「デッドロックの対処方法」
セクション19.3.5「パーティションに関する情報を取得する」
セクション1.7.1「標準 SQL に対する MySQL 拡張機能」
セクション1.6「質問またはバグをレポートする方法」

EXPLAIN SELECT ... ORDER BY

セクション8.2.1.15「ORDER BY の最適化」

EXPLAIN tbl_name

セクション8.8.1「EXPLAIN によるクエリーの最適化」

F

[索引の先頭]

FETCH
セクション13.6.6.2「カーソルの DECLARE 構文」
セクション13.6.6.3「カーソルの FETCH 構文」
セクションD.1「ストアドプログラムの制約」

FETCH ... INTO var_list
セクション13.6.4「ストアドプログラム内の変数」

FLUSH
セクション13.7.6.3「FLUSH 構文」
セクション13.7.1.4「GRANT 構文」
セクション6.2.1「MySQL で提供される権限」
セクション4.5.4「mysqldump — データベースバックアッププログラム」
セクション13.7.6.6「RESET 構文」
root のパスワードのリセット: UNIX システム
root のパスワードのリセット: Windows システム
root のパスワードのリセット: 一般的な手順
セクション17.1.4.5「グローバルトランザクション ID のオプションと変数」
セクション5.1.11「シグナルへのサーバー応答」
セクションD.1「ストアドプログラムの制約」
セクション7.3.1「バックアップポリシーの確立」
セクション17.4.1.14「レプリケーションと FLUSH」
セクション13.3.3「暗黙的なコミットを発生させるステートメント」
セクション2.10.2「最初の MySQL アカウントのセキュリティー設定」
セクション1.7.1「標準 SQL に対する MySQL 拡張機能」

FLUSH BINARY LOGS
セクション5.2.7「サーバーログの保守」

FLUSH DES_KEY_FILE
セクション12.13「暗号化関数と圧縮関数」

FLUSH HOSTS
セクション8.11.5.2「DNS ルックアップの最適化とホストキャッシュ」
セクション22.9.10.1「host_cache テーブル」
セクション23.7.7.56「mysql_refresh()」
セクション5.1.4「サーバーシステム変数」
セクションB.5.2.6「ホスト 'host_name' は拒否されました」

FLUSH LOGS
セクション13.7.6.3「FLUSH 構文」
セクション5.2「MySQL Server ログ」
セクション23.7.7.56「mysql_refresh()」
セクション5.2.2「エラーログ」
セクション5.1.6「サーバーステータス変数」
セクション5.2.7「サーバーログの保守」
セクション17.2.2.1「スレーブリレーログ」
セクション7.2「データベースバックアップ方法」
セクション7.3.1「バックアップポリシーの確立」
セクション7.3.3「バックアップ戦略サマリー」
セクション17.1.3.3「フェイルオーバーおよびスケールアウトでの GTID の使用」
セクション17.4.1.14「レプリケーションと FLUSH」
セクション5.2.1「一般クエリーログおよびスロークエリーログの出力先の選択」

FLUSH MASTER
セクション17.4.1.14「レプリケーションと FLUSH」

FLUSH PRIVILEGES
セクション13.7.6.3「FLUSH 構文」
セクション18.5.11.3「MySQL Cluster と MySQL のセキュリティー手順」
セクション18.5.14「MySQL Cluster の配布された MySQL 権限」
セクション8.11.4.1「MySQL のメモリーの使用方法」
セクション23.7.7.56「mysql_refresh()」
セクション23.7.7.57「mysql_reload()」
セクション4.5.4「mysqldump — データベースバックアッププログラム」
セクション6.3.5「アカウントパスワードの割り当て」
セクション6.3.4「アカウントリソース制限の設定」
セクション6.2.7「アクセス拒否エラーの原因」
セクション5.1.3「サーバーコマンドオプション」
セクション6.3.2「ユーザーアカウントの追加」
セクション17.4.1.14「レプリケーションと FLUSH」
セクション6.2.2「権限システム付与テーブル」
セクション6.2.6「権限変更が有効化される時期」
セクション1.2「表記規則および構文規則」

FLUSH QUERY CACHE
セクション13.7.6.3「FLUSH 構文」
セクション8.9.3.4「クエリーキャッシュのステータスと保守」

FLUSH SLAVE
セクション17.4.1.14「レプリケーションと FLUSH」

FLUSH STATUS
セクション23.7.7.56「mysql_refresh()」
セクション8.2.1.7「インデックス拡張の使用」
セクション5.1.6「サーバーステータス変数」

FLUSH TABLE
セクション8.6.2「MyISAM テーブルの一括データロード」
セクション1.4「MySQL 5.6 の新機能」
セクション8.2.1.7「インデックス拡張の使用」

FLUSH TABLES
セクション13.7.6.3「FLUSH 構文」
セクション13.2.4「HANDLER 構文」
セクション13.2.5.2「INSERT DELAYED 構文」
セクション15.7.2「MERGE テーブルの問題点」
セクション8.6.2「MyISAM テーブルの一括データロード」
セクション4.6.3「myisamchk — MyISAM テーブルメンテナンスユーティリティー」
セクション8.4.3.1「MySQL でのテーブルのオープンとクローズの方法」
セクション8.11.4.1「MySQL のメモリーの使用方法」
セクション23.7.7.56「mysql_refresh()」
セクション4.6.10「mysqlhotcopy — データベースバックアッププログラム」
セクション8.9.3.4「クエリーキャッシュのステータスと保守」
セクション5.1.4「サーバーシステム変数」
セクション5.1.6「サーバーステータス変数」
セクション19.6「パーティショニングの制約と制限」

セクション8.9.4「プリペアドステートメントおよびストアドプログラムのキャッシュ」
セクション17.4.1.14「レプリケーションと FLUSH」
セクション17.1.1.4「レプリケーションマスターバイナリログ座標の取得」
セクション5.2.1「一般クエリーログおよびスロークエリーログの出力先の選択」
セクション8.12.5.2「一般的なスレッドの状態」
セクション15.2.4.2「適切に閉じられなかったテーブルの問題」

FLUSH TABLES ... FOR EXPORT

セクション13.7.6.3「FLUSH 構文」
MySQL 用語集
セクション14.5.4「テーブルスペースの位置の指定」
セクション14.5.5「テーブルスペースの別のサーバーへのコピー (トランスポータブルテーブルスペース)」
セクション14.6.2「別のマシンへの InnoDB テーブルの移動またはコピー」

FLUSH TABLES tbl_list WITH READ LOCK

セクション4.6.10「mysqlhotcopy — データベースバックアッププログラム」

FLUSH TABLES tbl_name ... FOR EXPORT

セクション13.7.6.3「FLUSH 構文」

FLUSH TABLES tbl_name ... WITH READ LOCK

セクション13.7.6.3「FLUSH 構文」

FLUSH TABLES tbl_name WITH READ LOCK

セクション13.2.4「HANDLER 構文」

FLUSH TABLES WITH READ LOCK

セクション13.7.6.3「FLUSH 構文」
セクション13.3.5「LOCK TABLES および UNLOCK TABLES 構文」
セクション4.5.4「mysqldump — データベースバックアッププログラム」
セクション17.1.1.5「mysqldump を使用したデータスナップショットの作成」
セクション13.3.1「START TRANSACTION、COMMIT、および ROLLBACK 構文」
セクション5.1.4「サーバーシステム変数」
セクション13.3.5.1「テーブルロックとトランザクションの通信」
セクション7.2「データベースバックアップ方法」
セクション7.3.1「バックアップポリシーの確立」
セクション17.4.1.14「レプリケーションと FLUSH」
セクション17.1.1.4「レプリケーションマスターバイナリログ座標の取得」
セクション5.2.1「一般クエリーログおよびスロークエリーログの出力先の選択」
セクション8.12.5.2「一般的なスレッドの状態」
セクション13.3.3「暗黙的なコミットを発生させるステートメント」

FLUSH USER_RESOURCES

セクション13.7.6.3「FLUSH 構文」
セクション6.3.4「アカウントリソース制限の設定」

G

[索引の先頭]

GET DIAGNOSTICS

セクション13.6.7.3「GET DIAGNOSTICS 構文」
セクション1.4「MySQL 5.6 の新機能」
セクション13.6.7.4「RESIGNAL 構文」
セクション13.7.5.41「SHOW WARNINGS 構文」
セクション13.6.7.5「SIGNAL 構文」
セクションB.1「エラー情報のソース」
セクション5.1.4「サーバーシステム変数」
シグナルの条件情報項目
セクションD.1「ストアドプログラムの制約」
セクション13.6.7「条件の処理」
セクションD.2「条件処理の制約」
診断領域が移入される方法

GRANT

セクション13.1.11「CREATE EVENT 構文」
セクション13.1.15「CREATE PROCEDURE および CREATE FUNCTION 構文」
セクション13.1.19「CREATE TRIGGER 構文」
セクション13.7.1.2「CREATE USER 構文」
セクション13.1.20「CREATE VIEW 構文」
セクション17.4.1.7「CURRENT_USER() のレプリケーション」
セクション13.7.6.3「FLUSH 構文」
セクション13.7.1.4「GRANT 構文」
セクション5.1.9「IPv6 サポート」
セクション5.1.9.3「IPv6 ローカルホストアドレスを使用した接続」
セクションA.13「MySQL 5.6 FAQ: レプリケーション」
セクション1.4「MySQL 5.6 の新機能」
セクション18.5.11.2「MySQL Cluster と MySQL 権限」
セクション6.1.2.4「MySQL でのパスワードハッシュ」
セクション6.2.1「MySQL で提供される権限」
セクション8.11.4.1「MySQL のメモリーの使用方法」
セクション6.2「MySQL アクセス権限システム」
セクション17.4.1.23「mysql システムデータベースのレプリケーション」
セクション5.2.4.4「mysql データベーステーブルへの変更に対するロギング形式」
MySQL 用語集
セクション2.10.1.1「mysql_install_db 実行の問題」
PAM 認証プラグインのインストール
PAM 認証プラグインの使用
セクション13.7.1.6「REVOKE 構文」
セクション13.7.5.22「SHOW GRANTS 構文」
セクション6.3.10.4「SSL コマンドのオプション」
セクション6.3.10.3「SSL 接続の使用」
Windows 認証プラグインのインストール
Windows 認証プラグインの使用
セクション6.3.5「アカウントパスワードの割り当て」
セクション6.3.4「アカウントリソース制限の設定」
セクション6.2.3「アカウント名の指定」
セクション6.2.5「アクセス制御、ステージ 2: リクエストの確認」
セクション6.2.7「アクセス拒否エラーの原因」

セクション20.4.6「イベントスケジューラと MySQL 権限」
セクション5.1.7「サーバー SQL モード」
セクション5.1.3「サーバーコマンドオプション」
セクション5.1.4「サーバーシステム変数」
セクション17.1.2.1「ステートメントベースおよび行ベースレプリケーションのメリットとデメリット」
セクション6.3.10「セキュアな接続のための SSL の使用」
セクション6.1.1「セキュリティーガイドライン」
セクション8.2.3「データベース権限の最適化」
セクション6.1.2.3「パスワードおよびロギング」
セクション6.1.2.1「パスワードセキュリティーのためのエンドユーザーガイドライン」
セクション24.2.3.8「パスワード検証プラグイン」
セクション6.1.2.6「パスワード検証プラグイン」
セクション6.3.9「プロキシユーザー」
セクション6.3.2「ユーザーアカウントの追加」
セクション6.3.1「ユーザー名とパスワード」
セクション17.4.1.14「レプリケーションと FLUSH」
セクション18.6.5「レプリケーションのための MySQL Cluster の準備」
セクション17.1.1.3「レプリケーション用ユーザーの作成」
セクション12.14「情報関数」
セクション6.1.3「攻撃者に対する MySQL のセキュアな状態の維持」
セクション12.13「暗号化関数と圧縮関数」
セクション13.3.3「暗黙的なコミットを発生させるステートメント」
セクション6.2.2「権限システム付与テーブル」
セクション6.2.6「権限変更が有効化される時期」
認証プラグインでのプロキシユーザーサポートの実装
セクション14.3.1「読み取り専用操作用の InnoDB の構成」

GRANT ALL

セクション13.7.1.4「GRANT 構文」

GRANT EVENT

セクション20.4.6「イベントスケジューラと MySQL 権限」

GRANT USAGE

セクション13.7.1.4「GRANT 構文」
セクション6.3.5「アカウントパスワードの割り当て」
セクション6.3.4「アカウントリソース制限の設定」

H

[索引の先頭]

HANDLER

セクション15.8.3「FEDERATED ストレージエンジンの注記とヒント」
セクション13.7.6.3「FLUSH 構文」
セクションA.4「MySQL 5.6 FAQ: ストアドプロシージャーおよびストアドファンクション」
セクション1.7「MySQL の標準への準拠」
セクション23.7.7.3「mysql_change_user()」
セクション5.1.4「サーバーシステム変数」
セクション19.6「パーティショニングの制約と制限」
セクション23.7.16「自動再接続動作の制御」

HANDLER ... CLOSE

セクション13.7.5.25「SHOW OPEN TABLES 構文」

HANDLER ... OPEN

セクション13.7.5.25「SHOW OPEN TABLES 構文」

HANDLER ... READ

セクションD.1「ストアドプログラムの制約」

HANDLER OPEN

セクション13.2.4「HANDLER 構文」
セクション13.1.33「TRUNCATE TABLE 構文」

HELP

セクション13.8.3「HELP 構文」
セクション5.1.10「サーバー側のヘルプ」
セクション17.4.1.27「サーバー側ヘルプテーブルのレプリケーション」
セクション13.3.5.3「テーブルロックの制限と条件」

I

[索引の先頭]

IF

セクション13.6.7.2「DECLARE ... HANDLER 構文」
セクション13.6.5.2「IF 構文」
セクション13.6.5「フロー制御ステートメント」
セクション8.9.4「プリペアドステートメントおよびストアドプログラムのキャッシュ」
セクション12.4「制御フロー関数」

INSERT

セクション10.1.7.6「_bin および binary 照合順序」
セクション15.5「ARCHIVE ストレージエンジン」
セクション3.6.9「AUTO_INCREMENT の使用」
セクション15.6「BLACKHOLE ストレージエンジン」
セクション23.7.10「C API プリペアドステートメント関数の概要」
セクション23.7.6「C API 関数の概要」
セクション15.8.2.1「CONNECTION を使用した FEDERATED テーブルの作成」
セクション13.1.13「CREATE INDEX 構文」
セクション13.1.15「CREATE PROCEDURE および CREATE FUNCTION 構文」
セクション13.1.17.1「CREATE TABLE ... SELECT 構文」
セクション13.1.19「CREATE TRIGGER 構文」
セクション13.1.20「CREATE VIEW 構文」
セクション13.6.7.2「DECLARE ... HANDLER 構文」
セクション13.2.2「DELETE 構文」
セクション8.2.2「DML ステートメントの最適化」
セクション12.17.2「Enterprise Encryption の使用法と例」
セクション8.8.3「EXPLAIN EXTENDED 出力フォーマット」
セクション8.8.1「EXPLAIN によるクエリーの最適化」
セクション8.8.2「EXPLAIN 出力フォーマット」
セクション13.8.2「EXPLAIN 構文」
セクション15.8.3「FEDERATED ストレージエンジンの注記とヒント」
セクション13.7.1.4「GRANT 構文」
セクション21.28「INFORMATION_SCHEMA VIEWS テーブル」
第21章「INFORMATION_SCHEMA テーブル」
セクション14.18.8「InnoDB memcached プラグインのトラブルシューティング」

セクション14.2.8「InnoDB のさまざまな SQL ステートメントで設定されたロック」
セクション14.19.5「InnoDB のエラーコード」
セクション14.19.2「InnoDB のリカバリの強制的な実行」
セクション14.2.6「InnoDB のレコード、ギャップ、およびネクストキーロック」
セクション14.12「InnoDB の起動オプションおよびシステム変数」
セクション8.5.4「InnoDB テーブルの一括データロード」
セクション13.2.5.3「INSERT ... ON DUPLICATE KEY UPDATE 構文」
セクション13.2.5.1「INSERT ... SELECT 構文」
セクション13.2.5.2「INSERT DELAYED 構文」
セクション8.2.2.1「INSERT ステートメントの速度」
セクション13.2.5「INSERT 構文」
セクション19.2.2「LIST パーティショニング」
セクション13.2.6「LOAD DATA INFILE 構文」
セクション13.3.5「LOCK TABLES および UNLOCK TABLES 構文」
セクション15.7「MERGE ストレージエンジン」
セクション15.7.2「MERGE テーブルの問題点」
セクション14.6.4「MyISAM から InnoDB へのテーブルの変換」
セクション8.6.1「MyISAM クエリーの最適化」
セクション15.2「MyISAM ストレージエンジン」
セクション8.6.2「MyISAM テーブルの一括データロード」
セクション2.11.1.3「MySQL 5.5 から 5.6 へのアップグレード」
セクションA.5「MySQL 5.6 FAQ: トリガー」
セクションA.6「MySQL 5.6 FAQ: ビュー」
セクションA.1「MySQL 5.6 FAQ: 全般」
セクション1.4「MySQL 5.6 の新機能」
セクション18.1.6.3「MySQL Cluster でのトランザクション処理に関する制限」
セクション18.5.5「MySQL Cluster のローリング再起動の実行」
セクション18.5.12.1「MySQL Cluster ディスクデータオブジェクト」
セクション18.6.11「MySQL Cluster レプリケーションの競合解決」
セクション6.2.1「MySQL で提供される権限」
セクション4.5.1.1「mysql のオプション」
セクション19.1「MySQL のパーティショニングの概要」
セクション1.3.2「MySQL の主な機能」
セクション6.2「MySQL アクセス権限システム」
セクション8.9.3「MySQL クエリーキャッシュ」
セクションB.5.2.9「MySQL サーバーが存在しなくなりました」
セクション5.2.4.4「mysql データベーステーブルへの変更に対するロギング形式」
MySQL 用語集
セクション23.7.7.1「mysql_affected_rows()」
セクション23.7.7.37「mysql_insert_id()」
セクション23.7.7.48「mysql_num_rows()」
セクション23.7.15.1「mysql_query() が成功を返したあとに mysql_store_result() が NULL を返すことがあるのはなぜか」
セクション23.7.11.10「mysql_stmt_execute()」
セクション23.7.11.13「mysql_stmt_field_count()」
セクション23.7.11.16「mysql_stmt_insert_id()」
セクション23.7.11.18「mysql_stmt_num_rows()」
セクション23.7.11.21「mysql_stmt_prepare()」
セクション23.7.7.70「mysql_store_result()」
セクション4.6.8「mysqlbinlog — バイナリログファイルを処理するためのユーティリティー」
セクション4.5.4「mysqldump — データベースバックアッププログラム」
セクション18.5.15「NDB API 統計のカウンタと変数」
セクション14.7.6「OLTP ワークロードの圧縮」
セクション13.7.2.4「OPTIMIZE TABLE 構文」
セクション1.7.3.1「PRIMARY KEY および UNIQUE インデックス制約」
セクション19.3.1「RANGE および LIST パーティションの管理」
セクション19.2.1「RANGE パーティショニング」
セクション13.2.8「REPLACE 構文」
セクション13.7.5.28「SHOW PROCEDURE CODE 構文」
セクション13.7.5.41「SHOW WARNINGS 構文」
セクション13.2.11「UPDATE 構文」
セクション8.2.5「その他の最適化のヒント」
セクション12.18「その他の関数」
セクション6.2.5「アクセス制御、ステージ 2: リクエストの確認」
セクション6.2.7「アクセス拒否エラーの原因」
セクション14.11.1「オンライン DDL の概要」
セクション10.1.13「カラム文字セットの変換」
セクション23.7.15.2「クエリーからどんな結果を得ることができるか」
セクション8.9.3.1「クエリーキャッシュの動作」
セクション8.12.5.4「クエリーキャッシュスレッドの状態」
セクション13.2.10「サブクエリー構文」
セクション5.1.7「サーバー SQL モード」
セクション5.1.3「サーバーコマンドオプション」
セクション5.1.4「サーバーシステム変数」
セクション5.1.12「シャットダウンプロセス」
セクション5.4.1.12「ステートメントプローブ」
セクション17.1.2.1「ステートメントベースおよび行ベースレプリケーションのメリットとデメリット」
セクション20.7「ストアドプログラムのバイナリロギング」
セクションD.1「ストアドプログラムの制約」
セクション18.2.6「テーブルとデータを含む MySQL Cluster の例」
セクション3.3.3「テーブルへのデータのロード」
セクション8.10.2「テーブルロックの問題」
セクション11.6「データ型デフォルト値」
セクション14.6.3「トランザクションを使用した DML 操作のグループ化」
セクション20.3「トリガーの使用」
セクション20.3.1「トリガーの構文と例」
セクション17.1.2.3「バイナリロギングでの安全および安全でないステートメントの判断」
セクション5.2.4「バイナリログ」
セクション7.1「バックアップとリカバリの種類」
セクション7.3.1「バックアップポリシーの確立」
セクション22.9.6「パフォーマンススキーマステートメントイベントテーブル」
セクション19.6.4「パーティショニングとロック」
セクション19.6「パーティショニングの制約と制限」
セクション19.4「パーティションプルーニング」
セクション19.5「パーティション選択」
セクション8.9.4「プリペアドステートメントおよびストアドプログラムのキャッシュ」
セクション6.3.2「ユーザーアカウントの追加」
セクション17.4.1.1「レプリケーションと AUTO_INCREMENT」
セクション17.4.1.28「レプリケーションとサーバー SQL モード」
セクション17.4.1.15「レプリケーションとシステム関数」
セクション17.4.1.34「レプリケーションと変数」

セクション17.1.4.2「レプリケーションマスターのオプションと変数」
セクション17.2.3.3「レプリケーションルールの適用」
セクション17.4.1.26「レプリケーション中のスレーブエラー」
セクション5.2.1「一般クエリーログおよびスロークエリーログの出力先の選択」
セクション8.12.5.2「一般的なスレッドの状態」
セクション12.9.5「全文制限」
セクション8.10.1「内部ロック方法」
セクション8.10.3「同時挿入」
セクション13.1.17.2「外部キー制約の使用」
セクション12.20.3「式の処理」
セクション14.6.5.1「従来の InnoDB の自動インクリメントロック」
セクション12.14「情報関数」
セクション11.1.2「日付と時間型の概要」
セクション20.5.3「更新可能および挿入可能なビュー」
セクション23.7.15.3「最後に挿入された行の一意の ID を取得する方法」
セクション14.6.5.2「構成可能な InnoDB の自動インクリメントロック」
セクション1.7.1「標準 SQL に対する MySQL 拡張機能」
セクション6.2.2「権限システム付与テーブル」
セクション6.2.6「権限変更が有効化される時期」
セクション1.7.3.3「無効データの制約」
セクション11.5.3.3「空間カラムへのデータ移入」
セクション11.2.6「範囲外およびオーバーフローの処理」
セクション5.4.1.7「行レベルプローブ」
セクション23.7.17「複数ステートメント実行の C API サポート」

INSERT ... ()

セクション5.4.1.12「ステートメントプローブ」

INSERT ... ON DUPLICATE KEY UPDATE

セクション15.8.3「FEDERATED ストレージエンジンの注記とヒント」
セクション14.2.8「InnoDB のさまざまな SQL ステートメントで設定されたロック」
セクション13.2.5.3「INSERT ... ON DUPLICATE KEY UPDATE 構文」
セクション13.2.5.2「INSERT DELAYED 構文」
セクション13.2.5「INSERT 構文」
セクション15.7.2「MERGE テーブルの問題点」
セクション18.6.3「MySQL Cluster レプリケーションの既知の問題」
MySQL 用語集
セクション23.7.7.1「mysql_affected_rows()」
セクション23.7.7.37「mysql_insert_id()」
セクション12.18「その他の関数」
セクション17.1.2.3「バイナリロギングでの安全および安全でないステートメントの判断」
セクション19.6.4「パーティショニングとロック」
セクション12.14「情報関数」
セクション14.6.5.2「構成可能な InnoDB の自動インクリメントロック」

INSERT ... SELECT

セクション13.1.17.1「CREATE TABLE ... SELECT 構文」
セクション14.2.8「InnoDB のさまざまな SQL ステートメントで設定されたロック」
セクション13.2.5.3「INSERT ... ON DUPLICATE KEY UPDATE 構文」
セクション13.2.5.1「INSERT ... SELECT 構文」
セクション13.2.5.2「INSERT DELAYED 構文」
セクション13.2.5「INSERT 構文」
セクション18.3.4.3「MySQL Cluster のシステム変数」
セクションB.5.8「MySQL の既知の問題」
セクション23.7.7.37「mysql_insert_id()」
セクション5.1.4「サーバーシステム変数」
セクション5.4.1.12「ステートメントプローブ」
セクション17.1.2.1「ステートメントベースおよび行ベースレプリケーションのメリットとデメリット」
セクション5.2.4「バイナリログ」
セクション19.6.4「パーティショニングとロック」
セクション19.5「パーティション選択」
セクション17.4.1.16「レプリケーションと LIMIT」
セクション8.10.3「同時挿入」
セクション14.6.5.2「構成可能な InnoDB の自動インクリメントロック」

INSERT ... SELECT ON DUPLICATE KEY UPDATE

セクション13.2.5.3「INSERT ... ON DUPLICATE KEY UPDATE 構文」
セクション13.2.5.1「INSERT ... SELECT 構文」

INSERT ... SET

セクション13.2.5「INSERT 構文」

INSERT ... VALUES

セクション13.2.5「INSERT 構文」
セクション23.7.7.35「mysql_info()」

INSERT DELAYED

セクション13.1.7「ALTER TABLE 構文」
セクション15.5「ARCHIVE ストレージエンジン」
セクション13.2.5.2「INSERT DELAYED 構文」
セクション13.2.5「INSERT 構文」
セクション13.7.6.4「KILL 構文」
セクション15.3「MEMORY ストレージエンジン」
セクション15.7.2「MERGE テーブルの問題点」
セクション8.6.1「MyISAM クエリーの最適化」
セクション8.6.2「MyISAM テーブルの一括データロード」
セクション24.1.1「MySQL のスレッド」
セクションB.5.8「MySQL の既知の問題」
セクション4.5.4「mysqldump — データベースバックアッププログラム」
セクション5.1.4「サーバーシステム変数」
セクション5.1.6「サーバーステータス変数」
セクションD.1「ストアドプログラムの制約」
セクション22.9.10.3「スレッドテーブル」
セクション13.3.5.3「テーブルロックの制限と条件」
セクション8.10.2「テーブルロックの問題」
セクション1.7.2.3「トランザクションおよびアトミック操作の違い」
セクション19.6.4「パーティショニングとロック」
セクション19.6「パーティショニングの制約と制限」
セクション20.5.3「更新可能および挿入可能なビュー」
セクション5.2.4.3「混合形式のバイナリロギング形式」
セクション8.12.5.3「遅延挿入スレッドの状態」

INSERT IGNORE
セクション13.1.17.1「CREATE TABLE ... SELECT 構文」
セクション1.7.3.4「ENUM および SET の制約」
セクション13.2.5「INSERT 構文」
セクション4.5.4「mysqldump — データベースバックアッププログラム」
セクション5.1.7「サーバー SQL モード」
セクション12.14「情報関数」
セクション1.7.3.3「無効データの制約」

INSERT IGNORE ... SELECT
セクション13.2.5.1「INSERT ... SELECT 構文」

INSERT INTO ... SELECT
セクション13.1.11「CREATE EVENT 構文」
セクション17.4.1.5「CREATE TABLE ... SELECT ステートメントのレプリケーション」
セクション13.2.5「INSERT 構文」
セクション15.3「MEMORY ストレージエンジン」
セクション1.7.2.1「SELECT INTO TABLE の違い」
セクション6.2.5「アクセス制御、ステージ 2: リクエストの確認」
セクション14.2.4「一貫性非ロック読み取り」
セクション1.7.3.3「無効データの制約」

INSERT INTO ... SELECT ...
セクション15.8.3「FEDERATED ストレージエンジンの注記とヒント」
セクション23.7.7.35「mysql_info()」
セクション23.7.15.2「クエリーからどんな結果を得ることができるか」

INSERT INTO ... SELECT FROM memory_table
セクション17.4.1.21「レプリケーションと MEMORY テーブル」

INSERT LOW_PRIORITY
セクション8.2.5「その他の最適化のヒント」

INSTALL PLUGIN
セクション13.7.6.3「FLUSH 構文」
セクション21.14「INFORMATION_SCHEMA PLUGINS テーブル」
セクション24.2.4.6「INFORMATION_SCHEMA プラグインの作成」
セクション13.7.3.3「INSTALL PLUGIN 構文」
セクション8.11.4.1「MySQL のメモリーの使用方法」
セクション2.9.4「MySQL ソース構成オプション」
セクション24.2「MySQL プラグイン API」
セクション4.4.4「mysql_plugin — MySQL サーバープラグインの構成」
セクション13.7.5.26「SHOW PLUGINS 構文」
セクション5.1.3「サーバーコマンドオプション」
サーバープラグインライブラリおよびプラグインディスクリプタ
セクション5.1.8.2「サーバープラグイン情報の取得」
サーバー側認証プラグインの作成
セクション24.2.4.5「デーモンプラグインの作成」
パスワード検証プラグインのインストール
パスワード検証プラグインのオプションおよび変数
セクション24.2.4.10「パスワード検証プラグインの作成」
セクション15.11.1「プラガブルストレージエンジンのアーキテクチャー」
セクション6.3.7「プラガブル認証」
セクション24.2.2「プラグイン API のコンポーネント」
セクション5.1.8.1「プラグインのインストールおよびアンインストール」
セクション24.2.4.2「プラグインのデータ構造体」
セクション24.2.4.4「全文パーサープラグインの作成」
セクション17.3.8.2「準同期レプリケーションのインストールと構成」
セクション17.3.8.1「準同期レプリケーション管理インタフェース」
セクション24.2.4.8「監査プラグインの作成」
セクション6.3.12.1「監査ログプラグインのインストール」
セクション6.3.12.6「監査ログプラグインのオプションおよびシステム変数」

ITERATE
セクション13.6.7.2「DECLARE ... HANDLER 構文」
セクション13.6.5.3「ITERATE 構文」
セクション13.6.2「ステートメントラベルの構文」
セクション13.6.5「フロー制御ステートメント」

K
[索引の先頭]

KILL
セクション13.7.1.4「GRANT 構文」
セクション13.7.6.4「KILL 構文」
セクション6.2.1「MySQL で提供される権限」
セクションB.5.2.9「MySQL サーバーが存在しなくなりました」
セクション23.7.7.38「mysql_kill()」
セクション4.6.16「mysql_zap — パターンに一致するプロセスを強制終了」
セクション13.7.5.30「SHOW PROCESSLIST 構文」
セクション13.3.5.3「テーブルロックの制限と条件」
セクション8.12.5.2「一般的なスレッドの状態」

KILL CONNECTION
セクション13.7.6.4「KILL 構文」
セクション13.4.2.6「STOP SLAVE 構文」
セクション5.1.12「シャットダウンプロセス」

KILL QUERY
セクション13.7.6.4「KILL 構文」
セクション13.4.2.6「STOP SLAVE 構文」
セクション5.1.12「シャットダウンプロセス」

L
[索引の先頭]

LEAVE
セクション13.6.7.2「DECLARE ... HANDLER 構文」
セクション13.6.5.4「LEAVE 構文」
セクション13.6.5.5「LOOP 構文」
セクション13.6.5.7「RETURN 構文」
セクション13.6.2「ステートメントラベルの構文」
セクションD.1「ストアドプログラムの制約」

セクション13.6.5「フロー制御ステートメント」

LOAD DATA

セクション13.1.19「CREATE TRIGGER 構文」
セクション11.4.4「ENUM 型」
セクション13.2.6「LOAD DATA INFILE 構文」
セクション6.1.6「LOAD DATA LOCAL のセキュリティーの問題」
セクション13.2.7「LOAD XML 構文」
セクション1.4「MySQL 5.6 の新機能」
セクション19.1「MySQL のパーティショニングの概要」
セクションB.5.8「MySQL の既知の問題」
セクション3.3.4.1「すべてのデータの選択」
セクション5.1.3「サーバーコマンドオプション」
セクション5.1.4「サーバーシステム変数」
セクションD.1「ストアドプログラムの制約」
セクション3.3.3「テーブルへのデータのロード」
セクション10.1.3.2「データベース文字セットおよび照合順序」
セクション20.3「トリガーの使用」
セクション19.6.4「パーティショニングとロック」
セクション19.6「パーティショニングの制約と制限」
セクション19.5「パーティション選択」
セクション9.4「ユーザー定義変数」
セクション8.10.3「同時挿入」
セクション13.1.17.2「外部キー制約の使用」
セクション6.1.3「攻撃者に対する MySQL のセキュアな状態の維持」
セクション14.6.5.2「構成可能な InnoDB の自動インクリメントロック」

LOAD DATA INFILE

セクション8.2.2.1「INSERT ステートメントの速度」
セクション13.2.6「LOAD DATA INFILE 構文」
セクション15.3「MEMORY ストレージエンジン」
セクション8.6.2「MyISAM テーブルの一括データロード」
セクション15.2.1「MyISAM 起動オプション」
セクション18.1.6.3「MySQL Cluster でのトランザクション処理に関する制限」
セクション18.5.5「MySQL Cluster のローリング再起動の実行」
セクションB.5.4.4「MySQL が一時ファイルを格納する場所」
セクションB.5.4.3「MySQL が満杯のディスクを処理する方法」
セクション6.2.1「MySQL で提供される権限」
セクション4.5.1.1「mysql のオプション」
セクションB.5.8「MySQL の既知の問題」
セクション6.2「MySQL アクセス権限システム」
セクション2.9.4「MySQL ソース構成オプション」
セクション5.2.4.4「mysql データベーステーブルへの変更に対する口ギング形式」
セクション17.4.2「MySQL バージョン間のレプリケーション互換性」
セクション4.1「MySQL プログラムの概要」
セクション4.6.8.1「mysqlbinlog 16 進ダンプ形式」
セクション4.6.8「mysqlbinlog — バイナリログファイルを処理するためのユーティリティー」
セクション4.5.4「mysqldump — データベースバックアッププログラム」
セクション4.5.5「mysqlimport — データインポートプログラム」
セクション18.4.24「ndb_show_tables — NDB テーブルのリストの表示」
セクション9.1.7「NULL 値」
セクションB.5.5.3「NULL 値に関する問題」
セクション13.2.9.1「SELECT ... INTO 構文」
セクション13.7.5.41「SHOW WARNINGS 構文」
セクションD.10.6「Windows プラットフォームの制限」
セクション8.2.5「その他の最適化のヒント」
セクション6.2.7「アクセス拒否エラーの原因」
セクション13.2.10「サブクエリー構文」
セクション5.1.3「サーバーコマンドオプション」
セクション5.1.4「サーバーシステム変数」
セクション13.2.10.1「スカラーオペランドとしてのサブクエリー」
セクション17.3.1.2「スレーブからローデータをバックアップする」
セクション7.2「データベースバックアップ方法」
セクション17.1.2.3「バイナリロギングでの安全および安全でないステートメントの判断」
セクション7.1「バックアップとリカバリの種類」
セクション17.4.1.17「レプリケーションと LOAD DATA INFILE」
セクション8.12.5.7「レプリケーションスレーブ SQL スレッドの状態」
セクション17.1.4.3「レプリケーションスレーブのオプションと変数」
セクション7.4.4「区切りテキストフォーマットバックアップのリロード」
セクション8.10.3「同時挿入」
セクション12.14「情報関数」
セクションD.7「文字セットの制約」
セクション13.3.3「暗黙的なコミットを発生させるステートメント」
セクション1.7.1「標準 SQL に対する MySQL 拡張機能」
セクション6.3.12.4「監査ログプラグインのロギング制御」
セクション6.3.12.8「監査ログプラグインの制限事項」
セクション11.2.6「範囲外およびオーバーフローの処理」

LOAD DATA INFILE ...

セクション23.7.7.35「mysql_info()」
セクション23.7.15.2「クエリーからどんな結果を得ることができるか」

LOAD DATA LOCAL

セクション13.2.6「LOAD DATA INFILE 構文」
セクション6.1.6「LOAD DATA LOCAL のセキュリティーの問題」
セクション23.7.7.49「mysql_options()」
セクション23.7.7.53「mysql_real_connect()」
セクション4.6.8「mysqlbinlog — バイナリログファイルを処理するためのユーティリティー」

LOAD DATA LOCAL INFILE

セクション23.7.6「C API 関数の概要」
セクション13.2.6「LOAD DATA INFILE 構文」
セクション23.7.7.64「mysql_set_local_infile_handler()」
セクション4.6.8「mysqlbinlog — バイナリログファイルを処理するためのユーティリティー」

LOAD INDEX INTO CACHE

セクション13.7.6.2「CACHE INDEX 構文」
セクション13.7.6.5「LOAD INDEX INTO CACHE 構文」
セクション8.9.2.4「インデックスプリロード」
セクション17.1.4.5「グローバルトランザクション ID のオプションと変数」

セクション19.6「パーティショニングの制約と制限」
セクション13.3.3「暗黙的なコミットを発生させるステートメント」

LOAD INDEX INTO CACHE ... IGNORE LEAVES

セクション13.7.6.5「LOAD INDEX INTO CACHE 構文」

LOAD XML

セクション13.2.7「LOAD XML 構文」
セクション1.4「MySQL 5.6 の新機能」
セクション19.1「MySQL のパーティショニングの概要」
セクション19.5「パーティション選択」

LOAD XML INFILE

セクション13.2.7「LOAD XML 構文」

LOAD XML LOCAL

セクション13.2.7「LOAD XML 構文」

LOAD XML LOCAL INFILE

セクション13.2.7「LOAD XML 構文」

LOCK TABLE

セクションB.5.7.1「ALTER TABLE での問題」
セクション8.12.5.2「一般的なスレッドの状態」
セクション8.10.3「同時挿入」

LOCK TABLES

セクション13.1.10「CREATE DATABASE 構文」
セクション13.1.17「CREATE TABLE 構文」
セクション13.1.19「CREATE TRIGGER 構文」
セクション13.7.6.3「FLUSH 構文」
セクション13.7.1.4「GRANT 構文」
セクション14.2.8「InnoDB のさまざまな SQL ステートメントで設定されたロック」
セクション14.12「InnoDB の起動オプションおよびシステム変数」
セクション14.6.7「InnoDB テーブル上の制限」
セクション13.3.5「LOCK TABLES および UNLOCK TABLES 構文」
セクション13.3.5.2「LOCK TABLES とトリガー」
セクション15.7.2「MERGE テーブルの問題点」
セクション8.6.2「MyISAM テーブルの一括データロード」
セクション6.2.1「MySQL で提供される権限」
セクション4.5.4「mysqldump — データベースバックアッププログラム」
セクション4.6.10「mysqlhotcopy — データベースバックアッププログラム」
セクション13.3.1「START TRANSACTION、COMMIT、および ROLLBACK 構文」
セクション5.1.4「サーバーシステム変数」
セクション8.11.1「システム要素およびスタートアップパラメータのチューニング」
セクションD.1「ストアドプログラムの制約」
セクション13.3.5.1「テーブルロックとトランザクションの通信」
セクション13.3.5.3「テーブルロックの制限と条件」
セクション8.10.2「テーブルロックの問題」
セクション14.2.11「デッドロックの対処方法」
セクション14.2.10「デッドロックの検出とロールバック」
セクション1.7.2.3「トランザクションおよびアトミック操作の違い」
セクション19.6.4「パーティショニングとロック」
セクション5.2.1「一般クエリーログおよびスロークエリーログの出力先の選択」
セクション8.10.1「内部ロック方法」
セクション13.3.3「暗黙的なコミットを発生させるステートメント」
セクション18.1.6.10「複数の MySQL Cluster ノードに関する制限」
セクション15.2.4.2「適切に閉じられなかったテーブルの問題」

LOCK TABLES ... READ

セクション13.7.6.3「FLUSH 構文」
セクション14.12「InnoDB の起動オプションおよびシステム変数」
セクション14.6.7「InnoDB テーブル上の制限」

LOCK TABLES ... WRITE

セクション14.12「InnoDB の起動オプションおよびシステム変数」
セクション14.6.7「InnoDB テーブル上の制限」

LOOP

セクション13.6.5.3「ITERATE 構文」
セクション13.6.5.4「LEAVE 構文」
セクション13.6.5.5「LOOP 構文」
セクション13.6.2「ステートメントラベルの構文」
セクション13.6.5「フロー制御ステートメント」

O

[索引の先頭]

OPTIMIZE TABLE

セクション15.5「ARCHIVE ストレージエンジン」
セクション13.2.2「DELETE 構文」
セクション21.29.25「INFORMATION_SCHEMA INNODB_FT_BEING_DELETED テーブル」
セクション21.29.20「INFORMATION_SCHEMA INNODB_FT_CONFIG テーブル」
セクション21.29.24「INFORMATION_SCHEMA INNODB_FT_DELETED テーブル」
セクション21.29.23「INFORMATION_SCHEMA INNODB_FT_INDEX_CACHE テーブル」
セクション21.29.22「INFORMATION_SCHEMA INNODB_FT_INDEX_TABLE テーブル」
セクション14.5.2「InnoDB File-Per-Table モード」
セクション14.14.4「InnoDB INFORMATION_SCHEMA FULLTEXT インデックステーブル」
セクション14.12「InnoDB の起動オプションおよびシステム変数」
セクション13.7.6.4「KILL 構文」
セクション15.7.2「MERGE テーブルの問題点」
セクション8.6.1「MyISAM クエリーの最適化」
セクション7.6「MyISAM テーブルの保守とクラッシュリカバリ」
セクション7.6.4「MyISAM テーブルの最適化」
セクション7.6.5「MyISAM テーブル保守スケジュールのセットアップ」
セクション4.6.3.1「myisamchk の一般オプション」
セクション1.4「MySQL 5.6 の新機能」

セクション18.1.6.2「MySQL Cluster の制限と標準の MySQL の制限との違い」
セクション18.5.12.3「MySQL Cluster ディスクデータストレージの要件」
セクション18.5.13.2「MySQL Cluster データノードのオンライン追加: 基本的な手順」
セクション18.5.13.3「MySQL Cluster データノードのオンライン追加: 詳細な例」
セクション18.3.4.2「MySQL Cluster 用の MySQL Server オプション」
セクションB.5.4.3「MySQL が満杯のディスクを処理する方法」
セクション6.2.1「MySQL で提供される権限」
セクション12.9.6「MySQL の全文検索の微調整」
セクションB.5.8「MySQL の既知の問題」
セクション24.4.1「MySQL サーバーのデバッグ」
セクション4.5.3「mysqlcheck — テーブル保守プログラム」
セクション13.7.2.4「OPTIMIZE TABLE 構文」
Unix 上の MyISAM へのシンボリックリンクの使用
セクション8.2.2.2「UPDATE ステートメントの速度」
セクション8.2.5「その他の最適化のヒント」
セクション14.11.9「オンライン DDL の制限」
セクション14.11.1「オンライン DDL の概要」
セクション17.1.4.5「グローバルトランザクション ID のオプションと変数」
セクション5.1.3「サーバーコマンドオプション」
セクション5.1.4「サーバーシステム変数」
セクション5.1.12「シャットダウンプロセス」
セクション5.2.5「スロークエリーログ」
セクション19.6「パーティショニングの制約と制限」
セクション19.3.4「パーティションの保守」
セクション17.4.1.14「レプリケーションと FLUSH」
セクション8.12.5.2「一般的なスレッドの状態」
セクション15.2.3.2「動的テーブルの特徴」
セクション13.3.3「暗黙的なコミットを発生させるステートメント」
セクション1.7.1「標準 SQL に対する MySQL 拡張機能」
セクション15.2.3.1「静的 (固定長) テーブルの特長」

P

[索引の先頭]

PREPARE

セクション23.7.20「C API のプリペアド CALL ステートメントのサポート」
セクション13.2.1「CALL 構文」
セクション13.5.3「DEALLOCATE PREPARE 構文」
セクション13.5.2「EXECUTE 構文」
セクション13.5.1「PREPARE 構文」
セクション5.1.6「サーバーステータス変数」
セクションD.1「ストアドプログラムの制約」
セクションD.5「ビューの制約」
セクション8.9.4「プリペアドステートメントおよびストアドプログラムのキャッシュ」
セクション8.10.4「メタデータのロック」
セクション13.5「準備済みステートメントのための SQL 構文」
セクション9.2.2「識別子の大文字と小文字の区別」

PURGE BINARY LOGS

セクション13.7.1.4「GRANT 構文」
セクション6.2.1「MySQL で提供される権限」
セクション4.5.4「mysqldump — データベースバックアッププログラム」
セクション13.4.1.1「PURGE BINARY LOGS 構文」
セクション13.4.1.2「RESET MASTER 構文」
セクション5.1.4「サーバーシステム変数」
セクション5.2.7「サーバーログの保守」
セクション5.2.4「バイナリログ」
セクション7.3.1「バックアップポリシーの確立」
セクション17.1.3.3「フェイルオーバーおよびスケールアウトでの GTID の使用」

R

[索引の先頭]

RELEASE SAVEPOINT

セクション13.3.4「SAVEPOINT、ROLLBACK TO SAVEPOINT、および RELEASE SAVEPOINT 構文」

RENAME TABLE

セクション13.1.7「ALTER TABLE 構文」
セクション13.2.2「DELETE 構文」
セクション4.5.4「mysqldump — データベースバックアッププログラム」
セクション13.1.32「RENAME TABLE 構文」
Unix 上の MyISAM へのシンボリックリンクの使用
セクション5.2.1「一般クエリーログおよびスロークエリーログの出力先の選択」
セクション8.12.5.2「一般的なスレッドの状態」
セクション14.6.2「別のマシンへの InnoDB テーブルの移動またはコピー」
セクション13.3.3「暗黙的なコミットを発生させるステートメント」
セクション1.7.1「標準 SQL に対する MySQL 拡張機能」
セクション9.2.2「識別子の大文字と小文字の区別」

RENAME USER

セクション17.4.1.7「CURRENT_USER() のレプリケーション」
セクション13.7.1.4「GRANT 構文」
セクション6.2.1「MySQL で提供される権限」
セクション5.2.4.4「mysql データベーステーブルへの変更に対する口ギング形式」
セクション13.7.1.5「RENAME USER 構文」
セクション20.4.6「イベントスケジューラと MySQL 権限」
セクション12.14「情報関数」
セクション13.3.3「暗黙的なコミットを発生させるステートメント」
セクション6.2.6「権限変更が有効化される時期」

REPAIR TABLE

セクションB.5.7.1「ALTER TABLE での問題」
セクション13.1.7.1「ALTER TABLE パーティション操作」
セクション13.1.7「ALTER TABLE 構文」
セクション15.5「ARCHIVE ストレージエンジン」
セクション13.7.2.2「CHECK TABLE 構文」
セクション13.7.6.4「KILL 構文」
セクション13.2.6「LOAD DATA INFILE 構文」
セクション15.7.2「MERGE テーブルの問題点」
セクション7.6「MyISAM テーブルの保守とクラッシュリカバリ」
セクション7.6.3「MyISAM テーブルの修復方法」
セクション15.2.4.1「MyISAM テーブルの破損」

セクション7.6.5「MyISAM テーブル保守スケジュールのセットアップ」
セクション15.2.1「MyISAM 起動オプション」
セクション4.6.3「myisamchk — MyISAM テーブルメンテナンスユーティリティー」
セクション4.6.3.1「myisamchk の一般オプション」
セクション2.11.1.3「MySQL 5.5 から 5.6 へのアップグレード」
セクションB.5.4.3「MySQL が満杯のディスクを処理する方法」
セクション6.2.1「MySQL で提供される権限」
セクション12.9.6「MySQL の全文検索の微調整」
セクションB.5.8「MySQL の既知の問題」
セクション4.5.3「mysqlcheck — テーブル保守プログラム」
セクション8.6.3「REPAIR TABLE ステートメントの速度」
セクション13.7.2.5「REPAIR TABLE 構文」
Unix 上の MyISAM へのシンボリックリンクの使用
セクション11.3.4「YEAR(2) の制限と YEAR(4) への移行」
セクション16.5.3「ZFS での MySQL リカバリーの扱い」
セクション17.1.4.5「グローバルトランザクション ID のオプションと変数」
セクション5.1.3「サーバーコマンドオプション」
セクション5.1.4「サーバーシステム変数」
セクション5.1.12「シャットダウンプロセス」
セクション5.2.5「スロークエリーログ」
セクション2.11.4「テーブルまたはインデックスの再作成または修復」
セクション7.2「データベースバックアップ方法」
セクション19.6「パーティショニングの制約と制限」
セクション19.3.4「パーティションの保守」
セクション17.4.1.14「レプリケーションと FLUSH」
セクション17.4.1.18「レプリケーションと REPAIR TABLE」
セクション8.12.5.2「一般的なスレッドの状態」
セクション8.10.5「外部ロック」
セクション13.3.3「暗黙的なコミットを発生させるステートメント」
セクション1.7.1「標準 SQL に対する MySQL 拡張機能」
セクション1.6「質問またはバグをレポートする方法」
セクション15.2.4.2「適切に閉じられなかったテーブルの問題」

REPEAT

セクション13.6.7.2「DECLARE ... HANDLER 構文」
セクション13.6.5.3「ITERATE 構文」
セクション13.6.5.4「LEAVE 構文」
セクション13.6.5.6「REPEAT 構文」
セクション13.6.2「ステートメントラベルの構文」
セクション20.1「ストアドプログラムの定義」
セクション13.6.5「フロー制御ステートメント」

REPLACE

セクション15.5「ARCHIVE ストレージエンジン」
セクション13.1.17.1「CREATE TABLE ... SELECT 構文」
セクション13.1.19「CREATE TRIGGER 構文」
セクション8.8.3「EXPLAIN EXTENDED 出力フォーマット」
セクション8.8.1「EXPLAIN によるクエリーの最適化」
セクション8.8.2「EXPLAIN 出力フォーマット」
セクション13.8.2「EXPLAIN 構文」
セクション14.2.8「InnoDB のさまざまな SQL ステートメントで設定されたロック」
セクション13.2.5.2「INSERT DELAYED 構文」
セクション13.2.5「INSERT 構文」
セクション15.7.2「MERGE テーブルの問題点」
セクション1.4「MySQL 5.6 の新機能」
セクション19.1「MySQL のパーティショニングの概要」
セクション1.3.2「MySQL の主な機能」
セクションB.5.8「MySQL の既知の問題」
セクションB.5.2.9「MySQL サーバーが存在しなくなりました」
セクション5.2.4.4「mysql データベーステーブルへの変更に対するロギング形式」
セクション23.7.7.1「mysql_affected_rows()」
セクション4.5.4「mysqldump — データベースバックアッププログラム」
セクション13.2.8「REPLACE 構文」
セクション13.2.11「UPDATE 構文」
セクション13.2.10「サブクエリー構文」
セクション5.1.3「サーバーコマンドオプション」
セクション17.1.2.1「ステートメントベースおよび行ベースレプリケーションのメリットとデメリット」
セクション11.6「データ型デフォルト値」
セクション19.6.4「パーティショニングとロック」
セクション19.6「パーティショニングの制約と制限」
セクション19.5「パーティション選択」
セクション12.14「情報関数」
セクション14.6.5.2「構成可能な InnoDB の自動インクリメントロック」
セクション1.7.1「標準 SQL に対する MySQL 拡張機能」

REPLACE ... SELECT

セクションB.5.8「MySQL の既知の問題」
セクション14.6.5.2「構成可能な InnoDB の自動インクリメントロック」

RESET

セクション13.7.6.3「FLUSH 構文」
セクション13.7.6.6「RESET 構文」
セクション17.1.4.5「グローバルトランザクション ID のオプションと変数」
セクション13.3.3「暗黙的なコミットを発生させるステートメント」
セクション1.7.1「標準 SQL に対する MySQL 拡張機能」

RESET MASTER

セクション1.4「MySQL 5.6 の新機能」
セクション18.6.3「MySQL Cluster レプリケーションの既知の問題」
セクション23.7.7.56「mysql_refresh()」
セクション13.4.1.2「RESET MASTER 構文」
セクション13.7.5.35「SHOW SLAVE STATUS 構文」
セクション17.1.4.5「グローバルトランザクション ID のオプションと変数」
セクション5.2.4「バイナリログ」
セクション17.3.6「フェイルオーバー中にマスターを切り替える」

RESET QUERY CACHE

セクション8.12.5.4「クエリーキャッシュスレッドの状態」

RESET SLAVE

セクション13.4.2.1「CHANGE MASTER TO 構文」
セクション1.4「MySQL 5.6 の新機能」
セクション18.6.3「MySQL Cluster レプリケーションの既知の問題」
セクション23.7.7.56「mysql_refresh()」

セクション13.4.1.2「RESET MASTER 構文」
セクション13.4.2.3「RESET SLAVE 構文」
セクション13.7.5.35「SHOW SLAVE STATUS 構文」
セクション17.1.4「レプリケーションおよびバイナリロギングのオプションと変数」
セクション13.3.3「暗黙的なコミットを発生させるステートメント」
セクション17.3.9「遅延レプリケーション」

RESET SLAVE ALL

セクション13.4.2.1「CHANGE MASTER TO 構文」

RESIGNAL

セクション13.6.7.1「DECLARE ... CONDITION 構文」
セクション13.6.7.2「DECLARE ... HANDLER 構文」
RESIGNAL には条件ハンドラのコンテキストが必要
RESIGNAL のみ
セクション13.6.7.4「RESIGNAL 構文」
シグナルの条件情報項目
セクションD.1「ストアドプログラムの制約」
セクション13.6.7.6「ハンドラのスコープに関するルール」
新しいシグナル情報を含む RESIGNAL
セクション13.6.7「条件の処理」
条件値とオプションの新しいシグナル情報を含む RESIGNAL
セクションD.2「条件処理の制約」
診断領域が移入される方法
診断領域の情報項目
診断領域関連のシステム変数

RETURN

セクション13.1.15「CREATE PROCEDURE および CREATE FUNCTION 構文」
セクション13.6.5.5「LOOP 構文」
セクション13.6.5.7「RETURN 構文」
セクションD.1「ストアドプログラムの制約」
セクション13.6.5「フロー制御ステートメント」
セクション8.9.4「プリペアドステートメントおよびストアドプログラムのキャッシュ」

REVOKE

セクション17.4.1.7「CURRENT_USER() のレプリケーション」
セクション13.7.6.3「FLUSH 構文」
セクション13.7.1.4「GRANT 構文」
セクション5.1.9「IPv6 サポート」
セクションA.13「MySQL 5.6 FAQ: レプリケーション」
セクション18.5.11.2「MySQL Cluster と MySQL 権限」
セクション6.2.1「MySQL で提供される権限」
セクション1.7.2「MySQL と標準 SQL との違い」
セクション8.11.4.1「MySQL のメモリーの使用方法」
セクション6.2「MySQL アクセス権限システム」
セクション17.4.1.23「mysql システムデータベースのレプリケーション」
セクション5.2.4.4「mysql データベーステーブルへの変更に対する口ギング形式」
MySQL 用語集
セクション2.10.1.1「mysql_install_db 実行の問題」
セクション13.7.1.6「REVOKE 構文」
セクション20.4.6「イベントスケジューラと MySQL 権限」
セクション5.1.4「サーバーシステム変数」
セクション17.1.2.1「ステートメントベースおよび行ベースレプリケーションのメリットとデメリット」
セクション6.1.1「セキュリティーガイドライン」
セクション6.3.9「プロキシユーザー」
セクション6.3.1「ユーザー名とパスワード」
セクション12.14「情報関数」
セクション13.3.3「暗黙的なコミットを発生させるステートメント」
セクション6.2.2「権限システム付与テーブル」
セクション6.2.6「権限変更が有効化される時期」
セクション14.3.1「読み取り専用操作用の InnoDB の構成」

REVOKE ALL PRIVILEGES

セクション13.7.1.4「GRANT 構文」
セクション6.2.1「MySQL で提供される権限」

ROLLBACK

セクション14.19.4「InnoDB のエラー処理」
セクション14.2.2「InnoDB のトランザクションモデルおよびロック」
セクション14.6.4「MyISAM から InnoDB へのテーブルの変換」
セクション14.2.1「MySQL および ACID モデル」
セクション13.3「MySQL トランザクションおよびロックステートメント」
セクション23.7.7.3「mysql_change_user()」
セクション13.3.4「SAVEPOINT、ROLLBACK TO SAVEPOINT、および RELEASE SAVEPOINT 構文」
セクション13.3.1「START TRANSACTION、COMMIT、および ROLLBACK 構文」
セクション5.1.4「サーバーシステム変数」
セクション20.7「ストアドプログラムのバイナリロギング」
セクション13.3.5.1「テーブルロックとトランザクションの通信」
セクション14.2.10「デッドロックの検出とロールバック」
セクション1.7.2.3「トランザクションおよびアトミック操作の違い」
セクション14.6.3「トランザクションを使用した DML 操作のグループ化」
セクション20.3.1「トリガーの構文と例」
セクション5.2.4「バイナリログ」
セクション17.4.1.31「レプリケーションとトランザクション」
セクション17.1.4.3「レプリケーションスレーブのオプションと変数」
セクション13.3.2「ロールバックできないステートメント」
セクション12.14「情報関数」
セクション13.3.3「暗黙的なコミットを発生させるステートメント」
セクションB.5.5.5「非トランザクションテーブルのロールバックの失敗」

ROLLBACK TO SAVEPOINT

セクション13.3.4「SAVEPOINT、ROLLBACK TO SAVEPOINT、および RELEASE SAVEPOINT 構文」

S

[索引の先頭]

SAVEPOINT

セクション13.3.4「SAVEPOINT、ROLLBACK TO SAVEPOINT、および RELEASE SAVEPOINT 構文」

SELECT

--safe-updates オプションの使用

セクション3.6.7「2 つのキーを使用した検索」
セクション13.1.7「ALTER TABLE 構文」
セクション13.1.9「ALTER VIEW 構文」
セクション15.5「ARCHIVE ストレージエンジン」
セクション8.3.8「B ツリーインデックスとハッシュインデックスの比較」
セクション23.7.20「C API のプリペアド CALL ステートメントのサポート」
セクション23.7.5「C API データ構造」
セクション23.7.10「C API プリペアドステートメント関数の概要」
セクション23.7.6「C API 関数の概要」
セクション13.2.1「CALL 構文」
セクション15.8.2.1「CONNECTION を使用した FEDERATED テーブルの作成」
セクション17.4.1.4「CREATE ... IF NOT EXISTS ステートメントのレプリケーション」
セクション13.1.11「CREATE EVENT 構文」
セクション13.1.15「CREATE PROCEDURE および CREATE FUNCTION 構文」
セクション13.1.17.1「CREATE TABLE ... SELECT 構文」
セクション13.1.17「CREATE TABLE 構文」
セクション13.1.20「CREATE VIEW 構文」
セクション15.4.1「CSV テーブルの修復と確認」
セクションB.5.5.2「DATE カラムの使用に関する問題」
セクション13.2.2「DELETE 構文」
セクション8.11.5.2「DNS ルックアップの最適化とホストキャッシュ」
セクション13.2.3「DO 構文」
セクション12.17.2「Enterprise Encryption の使用法と例」
セクション11.4.4「ENUM 型」
セクション13.2.10.6「EXISTS または NOT EXISTS を使用したサブクエリー」
EXISTS 戦略によるサブクエリーの最適化
セクション8.8.3「EXPLAIN EXTENDED 出力フォーマット」
セクション8.8.1「EXPLAIN によるクエリーの最適化」
セクション8.8.2「EXPLAIN 出力フォーマット」
セクション13.8.2「EXPLAIN 構文」
セクション15.8.3「FEDERATED ストレージエンジンの注記とヒント」
セクション13.7.6.3「FLUSH 構文」
セクション13.2.10.8「FROM 句内のサブクエリー」
FROM 句内のサブクエリー (派生テーブル) の最適化
セクション13.7.1.4「GRANT 構文」
セクション12.19.1「GROUP BY (集約) 関数」
セクション13.2.4「HANDLER 構文」
セクション22.9.10.1「host_cache テーブル」
セクション21.4「INFORMATION_SCHEMA COLUMNS テーブル」
セクション21.7「INFORMATION_SCHEMA EVENTS テーブル」
セクション21.9「INFORMATION_SCHEMA GLOBAL_VARIABLES および SESSION_VARIABLES テーブル」
セクション21.28「INFORMATION_SCHEMA VIEWS テーブル」
第21章「INFORMATION_SCHEMA テーブル」
セクション14.2.8「InnoDB のさまざまな SQL ステートメントで設定されたロック」
セクション14.19.2「InnoDB のリカバリの強制的な実行」
セクション14.13.14「InnoDB の読み取り専用トランザクションの最適化」
セクション14.12「InnoDB の起動オプションおよびシステム変数」
セクション8.5.2「InnoDB トランザクション管理の最適化」
セクション13.2.5.3「INSERT ... ON DUPLICATE KEY UPDATE 構文」
セクション13.2.5.1「INSERT ... SELECT 構文」
セクション13.2.5.2「INSERT DELAYED 構文」
セクション13.2.5「INSERT 構文」
セクション13.2.9.2「JOIN 構文」
セクション13.7.6.4「KILL 構文」
セクション13.2.7「LOAD XML 構文」
セクション15.7「MERGE ストレージエンジン」
セクション15.7.2「MERGE テーブルの問題点」
セクション14.6.4「MyISAM から InnoDB へのテーブルの変換」
セクション8.6.1「MyISAM クエリーの最適化」
セクション8.6.2「MyISAM テーブルの一括データロード」
セクション7.6.4「MyISAM テーブルの最適化」
セクションA.11「MySQL 5.6 FAQ: MySQL の中国語、日本語、および韓国語の文字セット」
セクションA.4「MySQL 5.6 FAQ: ストアドプロシージャーおよびストアドファンクション」
セクションA.13「MySQL 5.6 FAQ: レプリケーション」
セクション1.4「MySQL 5.6 の新機能」
セクション18.1.6.3「MySQL Cluster でのトランザクション処理に関する制限」
セクション18.3.4.3「MySQL Cluster のシステム変数」
セクション18.5.12.1「MySQL Cluster ディスクデータオブジェクト」
セクション18.6.11「MySQL Cluster レプリケーションの競合解決」
セクション18.6.4「MySQL Cluster レプリケーションスキーマとテーブル」
セクションB.5.4.4「MySQL が一時ファイルを格納する場所」
セクション8.4.4「MySQL が内部一時テーブルを使用する仕組み」
セクション6.2.1「MySQL で提供される権限」
セクション8.2.1.2「MySQL の WHERE 句の最適化の方法」
セクション4.5.1.1「mysql のオプション」
セクション1.3.2「MySQL の主な機能」
セクションB.5.8「MySQL の既知の問題」
セクション6.2「MySQL アクセス権限システム」
セクション8.9.3「MySQL クエリーキャッシュ」
セクション5.2.4.4「mysql データベーステーブルへの変更に対するロギング形式」
第22章「MySQL パフォーマンススキーマ」
セクション19.2.7「MySQL パーティショニングによる NULL の扱い」
MySQL 用語集
セクション23.7.7.1「mysql_affected_rows()」
セクション23.7.7.17「mysql_fetch_field()」
セクション23.7.7.22「mysql_field_count()」
セクション23.7.7.47「mysql_num_fields()」
セクション23.7.7.48「mysql_num_rows()」
セクション23.7.11.10「mysql_stmt_execute()」
セクション23.7.11.11「mysql_stmt_fetch()」
セクション23.7.11.18「mysql_stmt_num_rows()」
セクション23.7.11.28「mysql_stmt_store_result()」
セクション23.7.7.70「mysql_store_result()」
セクション23.7.7.72「mysql_use_result()」
セクション24.4.1.6「mysqld でのエラーの原因を見つけるためのサーバーログの使用」

セクション4.5.4「mysqldump — データベースバックアッププログラム」
セクション4.5.7「mysqlslap — 負荷エミュレーションクライアント」
セクション18.4.21「ndb_select_all — NDB テーブルの行の出力」
セクション18.5.10「ndbinfo MySQL Cluster 情報データベース」
セクション18.5.10.18「ndbinfo nodes テーブル」
セクション8.4.2.4「PROCEDURE ANALYSE の使用」
セクション19.2.3.1「RANGE COLUMNS パーティショニング」
セクション19.3.1「RANGE および LIST パーティションの管理」
セクション13.2.8「REPLACE 構文」
セクション13.2.9.1「SELECT ... INTO 構文」
セクション8.2.1「SELECT ステートメントの最適化」
セクション8.2.1.1「SELECT ステートメントの速度」
セクション13.2.9「SELECT 構文」
セクション13.3.6「SET TRANSACTION 構文」
セクション13.7.4「SET 構文」
セクション13.7.5.3「SHOW BINLOG EVENTS 構文」
セクション13.7.5.11「SHOW CREATE PROCEDURE 構文」
セクション13.7.5.14「SHOW CREATE VIEW 構文」
セクション13.7.5.18「SHOW ERRORS 構文」
セクション13.7.5.28「SHOW PROCEDURE CODE 構文」
セクション13.7.5.30「SHOW PROCESSLIST 構文」
セクション13.7.5.33「SHOW RELAYLOG EVENTS 構文」
セクション13.7.5.40「SHOW VARIABLES 構文」
セクション13.7.5.41「SHOW WARNINGS 構文」
セクション13.7.5「SHOW 構文」
セクション13.3.1「START TRANSACTION、COMMIT、および ROLLBACK 構文」
セクション13.2.9.4「UNION 構文」
セクション2.10.1「Unix 類似システムでのインストール後の手順」
セクション8.2.2.2「UPDATE ステートメントの速度」
セクション13.2.11「UPDATE 構文」
セクション3.3.4.1「すべてのデータの選択」
セクション4.6.3.4「その他の myisamchk オプション」
セクション8.2.5「その他の最適化のヒント」
セクション20.4.2「イベントスケジューラの構成」
セクション13.2.9.3「インデックスヒントの構文」
セクションB.5.6「オプティマイザ関連の問題」
セクション14.11.1「オンライン DDL の概要」
セクション13.6.6.2「カーソルの DECLARE 構文」
セクション13.6.6.3「カーソルの FETCH 構文」
セクション18.5.9「クイックリファレンス: MySQL Cluster の SQL ステートメント」
セクション3.2「クエリーの入力」
セクション8.9.3.2「クエリーキャッシュ SELECT オプション」
セクション8.9.3.4「クエリーキャッシュのステータスと保守」
セクション8.9.3.1「クエリーキャッシュの動作」
セクション8.12.5.4「クエリーキャッシュスレッドの状態」
セクション13.2.10.9「サブクエリーのエラー」
セクション13.2.10「サブクエリー構文」
セクション5.1.7「サーバー SQL モード」
セクション5.1.4「サーバーシステム変数」
セクション5.1.5「システム変数の使用」
シングルパートインデックスの range アクセスメソッド
セクション13.2.10.1「スカラーオペランドとしてのサブクエリー」
セクション5.4.1.12「ステートメントプローブ」
セクション20.7「ストアドプログラムのバイナリロギング」
セクションD.1「ストアドプログラムの制約」
セクション20.2.1「ストアドルーチンの構文」
セクション22.9.10.3「スレッドテーブル」
セクション3.3.4「テーブルからの情報の取り出し」
セクション18.2.6「テーブルとデータを含む MySQL Cluster の例」
セクション13.3.5.3「テーブルロックの制限と条件」
セクション8.10.2「テーブルロックの問題」
セクション14.2.11「デッドロックの対処方法」
セクション14.2.10「デッドロックの検出とロールバック」
セクション3.3.1「データベースの作成と選択」
セクション14.6.3「トランザクションを使用した DML 操作のグループ化」
セクション20.3.1「トリガーの構文と例」
セクション14.2.7「ネクストキーロックによるファントム問題の回避」
セクション5.2.4「バイナリログ」
セクション17.1.4.4「バイナリログのオプションと変数」
セクション22.4「パフォーマンススキーマインストゥルメント命名規則」
セクション19.6.4「パーティショニングとロック」
セクション19.3.5「パーティションに関する情報を取得する」
セクション19.4「パーティションプルーニング」
セクション19.5「パーティション選択」
セクションD.5「ビューの制約」
セクション20.5.1「ビューの構文」
セクション8.9.4「プリペアドステートメントおよびストアドプログラムのキャッシュ」
セクション8.3.5「マルチカラムインデックス」
セクション10.1.12「メタデータ用の UTF-8」
セクション9.4「ユーザー定義変数」
セクション17.2「レプリケーションの実装」
セクション17.1.4.3「レプリケーションスレーブのオプションと変数」
セクション17.1.4.2「レプリケーションマスターのオプションと変数」
セクション14.2.5「ロック読み取り (SELECT ... FOR UPDATE および SELECT ... LOCK IN SHARE MODE)」
セクション13.6.4.2「ローカル変数のスコープと解決」
セクションB.5.5.7「一致する行がない場合の問題の解決」
セクション8.12.5.2「一般的なスレッドの状態」
セクション14.2.4「一貫性非ロック読み取り」
セクション9.3「予約語」
セクション8.10.1「内部ロック方法」
セクション12.3.4「割り当て演算子」
セクション5.1.5.2「動的システム変数」
セクション8.4.3.2「同じデータベースに大量のテーブルを作成することの短所」
セクション8.10.3「同時挿入」
セクション17.4.1.11「呼び出される機能のレプリケーション」
セクション1.7.2.4「外部キーの違い」
セクション12.2「式評価での型変換」
セクション12.14「情報関数」
セクション10.1.4「接続文字セットおよび照合順序」
セクション6.1.3「攻撃者に対する MySQL のセキュアな状態の維持」
セクション9.1.1「文字列リテラル」

セクション2.10.2「最初の MySQL アカウントのセキュリティー設定」
セクション8.3「最適化とインデックス」
セクション14.6.5.2「構成可能な InnoDB の自動インクリメントロック」
セクション1.7.1「標準 SQL に対する MySQL 拡張機能」
セクション12.3.2「比較関数と演算子」
セクションB.5.5.8「浮動小数点値に関する問題」
準結合変換によるサブクエリーの最適化
セクション10.1.7.8「照合順序の効果の例」
セクション3.3.4.2「特定の行の選択」
セクション24.2.4.8「監査プラグインの作成」
セクション6.3.12.4「監査ログプラグインのロギング制御」
セクション11.5.3.7「空間インデックスの使用」
セクション12.9.1「自然言語全文検索」
セクション1.2「表記規則および構文規則」
セクション23.7.17「複数ステートメント実行の C API サポート」
セクション9.2.1「識別子の修飾子」
セクション1.6「質問またはバグをレポートする方法」
第12章「関数と演算子」

SELECT *

セクション11.4.3「BLOB 型と TEXT 型」

SELECT * FROM

セクション18.5.10.17「ndbinfo memory_per_fragment テーブル」

SELECT * FROM t PARTITION ()

セクション19.1「MySQL のパーティショニングの概要」

SELECT * INTO OUTFILE 'file_name' FROM tbl_name

セクション7.2「データベースバックアップ方法」

SELECT ... FOR UPDATE

セクション14.2.8「InnoDB のさまざまな SQL ステートメントで設定されたロック」
セクション14.2.3「InnoDB のロックモード」
MySQL 用語集
セクション14.2.11「デッドロックの対処方法」
セクション14.2.5「ロック読み取り (SELECT ... FOR UPDATE および SELECT ... LOCK IN SHARE MODE)」

SELECT ... FROM

セクション14.2.8「InnoDB のさまざまな SQL ステートメントで設定されたロック」

SELECT ... FROM ... FOR UPDATE

セクション14.2.8「InnoDB のさまざまな SQL ステートメントで設定されたロック」

SELECT ... FROM ... LOCK IN SHARE MODE

セクション14.2.8「InnoDB のさまざまな SQL ステートメントで設定されたロック」

SELECT ... INTO

セクション13.1.11「CREATE EVENT 構文」
セクション13.2.9.1「SELECT ... INTO 構文」
セクション1.7.2.1「SELECT INTO TABLE の違い」
セクション13.2.9「SELECT 構文」
セクション17.4.1.15「レプリケーションとシステム関数」
セクション13.6.4.2「ローカル変数のスコープと解決」

SELECT ... INTO DUMPFILE

セクション2.10.1「Unix 類似システムでのインストール後の手順」
セクション5.1.4「サーバーシステム変数」
セクション6.1.3「攻撃者に対する MySQL のセキュアな状態の維持」

SELECT ... INTO OUTFILE

セクション14.19.2「InnoDB のリカバリの強制的な実行」
セクション13.2.6「LOAD DATA INFILE 構文」
セクション6.2.1「MySQL で提供される権限」
セクション7.4.3「mysqldump による区切りテキストフォーマットでのデータのダンプ」
セクション9.1.7「NULL 値」
セクション13.2.9.1「SELECT ... INTO 構文」
セクション1.7.2.1「SELECT INTO TABLE の違い」
セクションD.10.6「Windows プラットフォームの制限」
セクション6.2.7「アクセス拒否エラーの原因」
セクション5.1.3「サーバーコマンドオプション」
セクション5.1.4「サーバーシステム変数」
セクション7.1「バックアップとリカバリの種類」
セクション6.1.3「攻撃者に対する MySQL のセキュアな状態の維持」
セクション1.2「表記規則および構文規則」

SELECT ... INTO OUTFILE 'file_name'

セクション13.2.9.1「SELECT ... INTO 構文」

SELECT ... INTO var_list

セクションD.1「ストアドプログラムの制約」
セクション13.6.4「ストアドプログラム内の変数」

SELECT ... LOCK IN SHARE MODE

セクション14.2.8「InnoDB のさまざまな SQL ステートメントで設定されたロック」
セクション14.2.3「InnoDB のロックモード」
セクション13.3.6「SET TRANSACTION 構文」
セクション14.2.5「ロック読み取り (SELECT ... FOR UPDATE および SELECT ... LOCK IN SHARE MODE)」

SELECT ... PARTITION

セクション1.4「MySQL 5.6 の新機能」

SELECT DISTINCT

InnoDB オプティマイザ統計でサンプリングされるページの数の構成
セクションD.4「サブクエリーの制約」
セクション8.12.5.2「一般的なスレッドの状態」
準結合変換によるサブクエリーの最適化

SELECT HIGH_PRIORITY

セクション8.2.5「その他の最適化のヒント」

SET

--safe-updates オプションの使用
セクション13.7.1.1「ALTER USER 構文」

セクション12.17.2「Enterprise Encryption の使用法と例」
セクション14.12「InnoDB の起動オプションおよびシステム変数」
セクション1.4「MySQL 5.6 の新機能」
セクション18.6.11「MySQL Cluster レプリケーションの競合解決」
セクション4.6.12「mysql_find_rows — ファイルからの SQL ステートメントの抽出」
セクション4.6.8「mysqlbinlog — バイナリログファイルを処理するためのユーティリティー」
セクション13.7.4「SET 構文」
セクション13.7.5.40「SHOW VARIABLES 構文」
セクション20.4.2「イベントスケジューラの構成」
セクション8.9.3.3「クエリーキャッシュの構成」
セクション13.2.10「サブクエリー構文」
セクション5.1.7「サーバー SQL モード」
セクション5.1.3「サーバーコマンドオプション」
セクション5.1.4「サーバーシステム変数」
セクション5.1.5「システム変数の使用」
セクション20.7「ストアドプログラムのバイナリロギング」
セクション20.1「ストアドプログラムの定義」
セクション13.6.4「ストアドプログラム内の変数」
セクション20.3.1「トリガーの構文と例」
セクション17.1.4.4「バイナリログのオプションと変数」
セクション6.3.6「パスワードの期限切れとサンドボックスモード」
セクション19.6.4「パーティショニングとロック」
セクション4.2.8「プログラム変数の設定へのオプションの使用」
セクション9.4「ユーザー定義変数」
セクション17.1.4.3「レプリケーションスレーブのオプションと変数」
セクション17.1.4.2「レプリケーションマスターのオプションと変数」
セクション12.3.4「割り当て演算子」
セクション5.1.5.2「動的システム変数」
セクション12.14「情報関数」
セクション1.7.1「標準 SQL に対する MySQL 拡張機能」
セクション12.3「演算子」
診断領域が移入される方法
第12章「関数と演算子」
セクション12.1「関数と演算子のリファレンス」

SET @@GLOBAL.gtid_purged

セクション4.5.4「mysqldump — データベースバックアッププログラム」

SET @@GLOBAL.ndb_slave_conflict_role = 'NONE'

セクション18.3.4.3「MySQL Cluster のシステム変数」

SET autocommit

セクション8.5.4「InnoDB テーブルの一括データロード」
セクション13.3「MySQL トランザクションおよびロックステートメント」

SET autocommit = 0

セクション17.3.8「準同期レプリケーション」

SET GLOBAL

セクション13.7.1.4「GRANT 構文」
セクション6.2.1「MySQL で提供される権限」
MySQL 用語集
セクション13.7.4「SET 構文」
セクション5.1.5「システム変数の使用」
セクション5.1.5.2「動的システム変数」
セクション17.3.8.2「準同期レプリケーションのインストールと構成」
セクション8.9.2.2「複合キーキャッシュ」

SET GLOBAL sql_slave_skip_counter

セクション13.4.2.4「SET GLOBAL sql_slave_skip_counter 構文」

SET GLOBAL TRANSACTION

セクション13.3.6「SET TRANSACTION 構文」

SET NAMES

セクション10.1.6「エラーメッセージの文字セット」
セクション12.2「式評価での型変換」

SET PASSWORD

セクション13.7.1.1「ALTER USER 構文」
セクション13.7.1.2「CREATE USER 構文」
セクション17.4.1.7「CURRENT_USER() のレプリケーション」
セクション2.11.1.3「MySQL 5.5 から 5.6 へのアップグレード」
セクション1.4「MySQL 5.6 の新機能」
セクション6.1.2.4「MySQL でのパスワードハッシュ」
セクション5.2.4.4「mysql データベーステーブルへの変更に対するロギング形式」
セクション4.4.3「mysql_install_db — MySQL データディレクトリの初期化」
セクション2.5.5「RPM パッケージを使用して MySQL を Linux にインストールする」
セクション13.7.1.7「SET PASSWORD 構文」
セクション13.7.4「SET 構文」
セクション6.3.8.4「SHA-256 認証プラグイン」
セクション6.3.5「アカウントパスワードの割り当て」
セクション6.2.3「アカウント名の指定」
セクション6.2.7「アクセス拒否エラーの原因」
セクションB.5.2.4「クライアントは認証プロトコルに対応できません」
セクション5.1.4「サーバーシステム変数」
セクション6.1.2.3「パスワードおよびロギング」
セクション6.3.6「パスワードの期限切れとサンドボックスモード」
セクション6.1.2.1「パスワードセキュリティーのためのエンドユーザーガイドライン」
セクション24.2.3.8「パスワード検証プラグイン」
セクション6.1.2.6「パスワード検証プラグイン」
セクション17.4.1.34「レプリケーションと変数」
セクション12.14「情報関数」
セクション13.3.3「暗黙的なコミットを発生させるステートメント」
セクション2.10.2「最初の MySQL アカウントのセキュリティー設定」
セクション6.2.2「権限システム付与テーブル」
セクション6.2.6「権限変更が有効化される時期」

SET SESSION

セクション13.7.4「SET 構文」
セクション5.1.5「システム変数の使用」

セクション5.1.5.2「動的システム変数」

SET SESSION TRANSACTION

セクション13.3.6「SET TRANSACTION 構文」

SET sql_mode='modes'

セクションA.3「MySQL 5.6 FAQ: サーバー SQL モード」

SET TIMESTAMP = value

セクション8.12.5「スレッド情報の検査」

SET TRANSACTION

セクション14.2.2「InnoDB のトランザクションモデルおよびロック」
セクション13.3.6「SET TRANSACTION 構文」
セクション13.3.1「START TRANSACTION、COMMIT、および ROLLBACK 構文」
セクション5.1.3「サーバーコマンドオプション」
セクション5.1.4「サーバーシステム変数」

SET TRANSACTION ISOLATION LEVEL

セクション13.7.4「SET 構文」
セクション13.3.1「START TRANSACTION、COMMIT、および ROLLBACK 構文」

SHOW

セクション23.7.5「C API データ構造」
セクション23.7.6「C API 関数の概要」
セクション13.1.11「CREATE EVENT 構文」
セクション13.1.15「CREATE PROCEDURE および CREATE FUNCTION 構文」
セクション21.1「INFORMATION_SCHEMA CHARACTER_SETS テーブル」
セクション21.3「INFORMATION_SCHEMA COLLATION_CHARACTER_SET_APPLICABILITY テーブル」
セクション21.2「INFORMATION_SCHEMA COLLATIONS テーブル」
セクション21.5「INFORMATION_SCHEMA COLUMN_PRIVILEGES テーブル」
セクション21.4「INFORMATION_SCHEMA COLUMNS テーブル」
セクション21.6「INFORMATION_SCHEMA ENGINES テーブル」
セクション21.7「INFORMATION_SCHEMA EVENTS テーブル」
セクション21.30.1「INFORMATION_SCHEMA FILES テーブル」
セクション21.8「INFORMATION_SCHEMA GLOBAL_STATUS および SESSION_STATUS テーブル」
セクション21.9「INFORMATION_SCHEMA GLOBAL_VARIABLES および SESSION_VARIABLES テーブル」
セクション21.10「INFORMATION_SCHEMA KEY_COLUMN_USAGE テーブル」
セクション21.30.2「INFORMATION_SCHEMA ndb_transid_mysql_connection_map テーブル」
セクション21.13「INFORMATION_SCHEMA PARTITIONS テーブル」
セクション21.14「INFORMATION_SCHEMA PLUGINS テーブル」
セクション21.15「INFORMATION_SCHEMA PROCESSLIST テーブル」
セクション21.16「INFORMATION_SCHEMA PROFILING テーブル」
セクション21.17「INFORMATION_SCHEMA REFERENTIAL_CONSTRAINTS テーブル」
セクション21.20「INFORMATION_SCHEMA SCHEMA_PRIVILEGES テーブル」
セクション21.19「INFORMATION_SCHEMA SCHEMATA テーブル」
セクション21.21「INFORMATION_SCHEMA STATISTICS テーブル」
セクション21.24「INFORMATION_SCHEMA TABLE_CONSTRAINTS テーブル」
セクション21.25「INFORMATION_SCHEMA TABLE_PRIVILEGES テーブル」
セクション21.22「INFORMATION_SCHEMA TABLES テーブル」
セクション21.23「INFORMATION_SCHEMA TABLESPACES テーブル」
セクション21.26「INFORMATION_SCHEMA TRIGGERS テーブル」
セクション21.27「INFORMATION_SCHEMA USER_PRIVILEGES テーブル」
セクション21.28「INFORMATION_SCHEMA VIEWS テーブル」
第21章「INFORMATION_SCHEMA テーブル」
セクションA.13「MySQL 5.6 FAQ: レプリケーション」
セクション18.6.4「MySQL Cluster レプリケーションスキーマとテーブル」
セクション1.3.2「MySQL の主な機能」
セクション23.7.11.28「mysql_stmt_store_result()」
セクション23.7.7.70「mysql_store_result()」
セクション23.7.7.72「mysql_use_result()」
セクション4.5.6「mysqlshow — データベース、テーブル、およびカラム情報の表示」
セクション13.7.5.6「SHOW COLUMNS 構文」
セクション13.7.5.23「SHOW INDEX 構文」
セクション13.7.5.25「SHOW OPEN TABLES 構文」
セクション13.7.5.38「SHOW TABLES 構文」
セクション10.1.9.3「SHOW ステートメントと INFORMATION_SCHEMA」
セクション21.32「SHOW ステートメントの拡張」
セクション13.7.5「SHOW 構文」
セクション13.6.6.2「カーソルの DECLARE 構文」
セクションD.1「ストアドプログラムの制約」
セクション3.3「データベースの作成と使用」
セクション5.2.4「バイナリログ」
セクション22.1「パフォーマンススキーマクイックスタート」
セクション13.4.1「マスターサーバーを制御するための SQL ステートメント」
セクション10.1.12「メタデータ用の UTF-8」
セクション1.7.1「標準 SQL に対する MySQL 拡張機能」
セクション9.2.3「識別子とファイル名のマッピング」

SHOW AUTHORS

セクション13.7.5.1「SHOW AUTHORS 構文」

SHOW BINARY LOGS

セクション6.2.1「MySQL で提供される権限」
セクション13.4.1.1「PURGE BINARY LOGS 構文」
セクション13.7.5.2「SHOW BINARY LOGS 構文」

セクション4.6.8.3「バイナリログファイルのバックアップのための mysqlbinlog の使用」
セクション13.4.1「マスターサーバーを制御するための SQL ステートメント」

SHOW BINLOG EVENTS

セクション17.1.3.1「GTID の概念」
セクション18.6.4「MySQL Cluster レプリケーションスキーマとテーブル」
セクション13.7.5.3「SHOW BINLOG EVENTS 構文」
セクション13.4.2.5「START SLAVE 構文」
セクションD.3「サーバー側のカーソルの制約」
セクション13.4.1「マスターサーバーを制御するための SQL ステートメント」

SHOW CHARACTER SET

セクション13.1.1「ALTER DATABASE 構文」
セクション10.1.2「MySQL での文字セットと照合順序」
セクション10.1.14「MySQL でサポートされる文字セットと照合順序」
セクション13.7.5.4「SHOW CHARACTER SET 構文」
セクション10.1.9.3「SHOW ステートメントと INFORMATION_SCHEMA」
セクション21.32「SHOW ステートメントの拡張」

SHOW COLLATION

セクション13.1.1「ALTER DATABASE 構文」
セクション23.7.5「C API データ構造」
セクション21.3「INFORMATION_SCHEMA COLLATION_CHARACTER_SET_APPLICABILITY テーブル」
セクション21.2「INFORMATION_SCHEMA COLLATIONS テーブル」
セクション10.1.2「MySQL での文字セットと照合順序」
セクション2.9.4「MySQL ソース構成オプション」
セクション13.7.5.5「SHOW COLLATION 構文」
セクション10.1.9.3「SHOW ステートメントと INFORMATION_SCHEMA」
セクション10.1.3.4「カラム文字セットおよび照合順序」
セクション10.1.3.3「テーブル文字セットおよび照合順序」
セクション10.1.3.2「データベース文字セットおよび照合順序」
セクション10.5「文字セットの構成」
セクション10.1.3.5「文字列リテラルの文字セットおよび照合順序」
セクション10.4.2「照合順序 ID の選択」

SHOW COLUMNS

セクション13.8.2「EXPLAIN 構文」
セクション21.29.16「INFORMATION_SCHEMA INNODB_BUFFER_PAGE テーブル」
セクション21.29.17「INFORMATION_SCHEMA INNODB_BUFFER_PAGE_LRU テーブル」
セクション21.29.18「INFORMATION_SCHEMA INNODB_BUFFER_POOL_STATS テーブル」
セクション21.29.1「INFORMATION_SCHEMA INNODB_CMP および INNODB_CMP_RESET テーブル」
セクション21.29.2「INFORMATION_SCHEMA INNODB_CMP_PER_INDEX および INNODB_CMP_PER_INDEX_RESET テーブル」
セクション21.29.3「INFORMATION_SCHEMA INNODB_CMPMEM および INNODB_CMPMEM_RESET テーブル」
セクション21.29.25「INFORMATION_SCHEMA INNODB_FT_BEING_DELETED テーブル」
セクション21.29.20「INFORMATION_SCHEMA INNODB_FT_CONFIG テーブル」
セクション21.29.21「INFORMATION_SCHEMA INNODB_FT_DEFAULT_STOPWORD テーブル」
セクション21.29.24「INFORMATION_SCHEMA INNODB_FT_DELETED テーブル」
セクション21.29.23「INFORMATION_SCHEMA INNODB_FT_INDEX_CACHE テーブル」
セクション21.29.22「INFORMATION_SCHEMA INNODB_FT_INDEX_TABLE テーブル」
セクション21.29.6「INFORMATION_SCHEMA INNODB_LOCK_WAITS テーブル」
セクション21.29.5「INFORMATION_SCHEMA INNODB_LOCKS テーブル」
セクション21.29.19「INFORMATION_SCHEMA INNODB_METRICS テーブル」
セクション21.29.9「INFORMATION_SCHEMA INNODB_SYS_COLUMNS テーブル」
セクション21.29.14「INFORMATION_SCHEMA INNODB_SYS_DATAFILES テーブル」
セクション21.29.10「INFORMATION_SCHEMA INNODB_SYS_FIELDS テーブル」
セクション21.29.11「INFORMATION_SCHEMA INNODB_SYS_FOREIGN テーブル」
セクション21.29.12「INFORMATION_SCHEMA INNODB_SYS_FOREIGN_COLS テーブル」
セクション21.29.8「INFORMATION_SCHEMA INNODB_SYS_INDEXES テーブル」
セクション21.29.7「INFORMATION_SCHEMA INNODB_SYS_TABLES テーブル」
セクション21.29.15「INFORMATION_SCHEMA INNODB_SYS_TABLESPACES テーブル」
セクション21.29.13「INFORMATION_SCHEMA INNODB_SYS_TABLESTATS ビュー」
セクション21.29.4「INFORMATION_SCHEMA INNODB_TRX テーブル」
セクション8.2.1.19「LIMIT クエリーの最適化」
セクション13.7.5.6「SHOW COLUMNS 構文」
セクション10.1.9.3「SHOW ステートメントと INFORMATION_SCHEMA」
セクション21.32「SHOW ステートメントの拡張」

SHOW COLUMNS FROM tbl_name LIKE 'enum_col'

セクション11.4.4「ENUM 型」

SHOW CONTRIBUTORS

セクション13.7.5.7「SHOW CONTRIBUTORS 構文」

SHOW COUNT()

セクション13.7.5.18「SHOW ERRORS 構文」
セクション13.7.5.41「SHOW WARNINGS 構文」

SHOW CREATE DATABASE

セクション13.7.5.8「SHOW CREATE DATABASE 構文」
セクション10.1.9.3「SHOW ステートメントと INFORMATION_SCHEMA」
セクション5.1.4「サーバーシステム変数」

SHOW CREATE EVENT

セクション13.7.5.19「SHOW EVENTS 構文」
セクション20.4.6「イベントスケジューラと MySQL 権限」
セクション20.4.4「イベントメタデータ」

SHOW CREATE FUNCTION

セクション13.1.15「CREATE PROCEDURE および CREATE FUNCTION 構文」
セクションA.4「MySQL 5.6 FAQ: ストアドプロシージャーおよびストアドファンクション」
セクション13.7.5.11「SHOW CREATE PROCEDURE 構文」
セクション20.2.3「ストアドルーチンのメタデータ」
セクション1.6「質問またはバグをレポートする方法」

SHOW CREATE PROCEDURE

セクション13.1.15「CREATE PROCEDURE および CREATE FUNCTION 構文」
セクションA.4「MySQL 5.6 FAQ: ストアドプロシージャーおよびストアドファンクション」
セクション13.7.5.10「SHOW CREATE FUNCTION 構文」
セクション20.2.3「ストアドルーチンのメタデータ」
セクション1.6「質問またはバグをレポートする方法」

SHOW CREATE SCHEMA

セクション13.7.5.8「SHOW CREATE DATABASE 構文」

SHOW CREATE TABLE

セクション13.1.7.1「ALTER TABLE パーティション操作」
セクション13.1.17「CREATE TABLE 構文」
セクション13.8.2「EXPLAIN 構文」
セクション15.8.2「FEDERATED テーブルの作成方法」
セクション19.2.5「KEY パーティショニング」
セクション7.6.3「MyISAM テーブルの修復方法」
セクション2.11.1.3「MySQL 5.5 から 5.6 へのアップグレード」
セクション18.1.6.1「MySQL Cluster での SQL 構文の不適合」
セクション18.3.4.3「MySQL Cluster のシステム変数」
セクション19.3.1「RANGE および LIST パーティションの管理」
セクション13.7.5.6「SHOW COLUMNS 構文」
セクション13.7.5.12「SHOW CREATE TABLE 構文」
セクション10.1.9.3「SHOW ステートメントと INFORMATION_SCHEMA」
セクション5.1.7「サーバー SQL モード」
セクション5.1.4「サーバーシステム変数」
セクション2.11.4「テーブルまたはインデックスの再作成または修復」
セクション3.4「データベースとテーブルに関する情報の取得」
セクション11.6「データ型デフォルト値」
セクション19.3.5「パーティションに関する情報を取得する」
セクション3.6.6「外部キーの使用」
セクション13.1.17.2「外部キー制約の使用」
セクション13.1.17.3「暗黙のカラム指定の変更」

SHOW CREATE TRIGGER

セクション21.26「INFORMATION_SCHEMA TRIGGERS テーブル」
セクション13.7.5.13「SHOW CREATE TRIGGER 構文」
セクション20.3.2「トリガーのメタデータ」

SHOW CREATE VIEW

セクション13.7.1.4「GRANT 構文」
セクション21.28「INFORMATION_SCHEMA VIEWS テーブル」
セクション6.2.1「MySQL で提供される権限」
セクション13.7.5.14「SHOW CREATE VIEW 構文」
セクション20.5.4「ビューのメタデータ」
セクションD.5「ビューの制約」

SHOW DATABASES

セクション13.1.10「CREATE DATABASE 構文」
セクション13.7.1.4「GRANT 構文」
第21章「INFORMATION_SCHEMA テーブル」
セクションA.11「MySQL 5.6 FAQ: MySQL の中国語、日本語、および韓国語の文字セット」
セクション18.5.11.2「MySQL Cluster と MySQL 権限」
セクション18.5.10「ndbinfo MySQL Cluster 情報データベース」
セクション13.7.5.15「SHOW DATABASES 構文」
セクション21.32「SHOW ステートメントの拡張」
セクション5.1.3「サーバーコマンドオプション」
セクション5.1.4「サーバーシステム変数」
セクション3.4「データベースとテーブルに関する情報の取得」
セクション3.3「データベースの作成と使用」
セクション22.2.1「パフォーマンススキーマビルド構成」
セクション6.2.2「権限システム付与テーブル」
セクション9.2.2「識別子の大文字と小文字の区別」

SHOW ENGINE

セクション6.2.1「MySQL で提供される権限」
セクション13.7.5.16「SHOW ENGINE 構文」

SHOW ENGINE INNODB MUTEX

セクション1.4「MySQL 5.6 の新機能」
セクション13.7.5.16「SHOW ENGINE 構文」

SHOW ENGINE INNODB STATUS

セクション21.29.18「INFORMATION_SCHEMA INNODB_BUFFER_POOL_STATS テーブル」
セクション14.14.3「InnoDB INFORMATION_SCHEMA システムテーブル」
セクション14.14.5「InnoDB INFORMATION_SCHEMA バッファープールテーブル」
セクション14.14.6「InnoDB INFORMATION_SCHEMA メトリックテーブル」
セクション14.6.6「InnoDB と FOREIGN KEY 制約」
セクション14.13.14「InnoDB の読み取り専用トランザクションの最適化」
セクション14.12「InnoDB の起動オプションおよびシステム変数」
セクション8.5.7「InnoDB ディスク I/O の最適化」
セクション14.15.2「InnoDB モニターの有効化」
セクション14.15.3「InnoDB 標準モニターおよびロックモニターの出力」
セクション14.6.4「MyISAM から InnoDB へのテーブルの変換」
MySQL 用語集
セクション13.7.5.16「SHOW ENGINE 構文」
セクションB.1「エラー情報のソース」

セクション14.2.11「デッドロックの対処方法」
セクション14.6.2「別のマシンへの InnoDB テーブルの移動またはコピー」
セクション13.1.17.2「外部キー制約の使用」
セクション14.3.1「読み取り専用操作用の InnoDB の構成」
セクション14.2.13.6「適応型ハッシュインデックス」

SHOW ENGINE NDB STATUS

セクション18.5「MySQL Cluster の管理」
セクション18.6.4「MySQL Cluster レプリケーションスキーマとテーブル」
セクション18.2.3.3「Windows での MySQL Cluster の初期起動」
セクション18.5.9「クイックリファレンス: MySQL Cluster の SQL ステートメント」

SHOW ENGINE NDBCLUSTER STATUS

セクション18.3.4.2「MySQL Cluster 用の MySQL Server オプション」
セクション18.5.9「クイックリファレンス: MySQL Cluster の SQL ステートメント」

SHOW ENGINE PERFORMANCE_SCHEMA STATUS

セクション13.7.5.16「SHOW ENGINE 構文」
セクション22.5「パフォーマンススキーマステータスモニタリング」

SHOW ENGINES

セクション15.5「ARCHIVE ストレージエンジン」
セクション15.6「BLACKHOLE ストレージエンジン」
セクション14.1.2「InnoDB の可用性チェック」
セクションA.10「MySQL 5.6 FAQ: MySQL Cluster」
セクション18.5.4「MySQL Cluster での MySQL サーバーの使用法」
セクション18.3.4.3「MySQL Cluster のシステム変数」
セクション2.3.5.3「MySQL サーバータイプの選択」
セクション18.5.10「ndbinfo MySQL Cluster 情報データベース」
セクション13.7.5.17「SHOW ENGINES 構文」
セクション18.5.9「クイックリファレンス: MySQL Cluster の SQL ステートメント」
セクション5.1.4「サーバーシステム変数」
セクション22.1「パフォーマンススキーマクイックスタート」
セクション22.2.1「パフォーマンススキーマビルド構成」
第15章「代替ストレージエンジン」

SHOW ERRORS

セクション13.6.7.3「GET DIAGNOSTICS 構文」
セクション14.6.6「InnoDB と FOREIGN KEY 制約」
セクション13.7.5.18「SHOW ERRORS 構文」
セクション13.7.5.41「SHOW WARNINGS 構文」
セクションB.1「エラー情報のソース」
セクション5.1.4「サーバーシステム変数」
シグナルの条件情報項目
条件値とオプションの新しいシグナル情報を含む RESIGNAL
診断領域が移入される方法

SHOW EVENTS

セクション21.7「INFORMATION_SCHEMA EVENTS テーブル」
セクション13.7.5.19「SHOW EVENTS 構文」
セクション20.4.6「イベントスケジューラと MySQL 権限」
セクション20.4.4「イベントメタデータ」
セクション17.4.1.11「呼び出される機能のレプリケーション」

SHOW FULL COLUMNS

セクション13.1.17「CREATE TABLE 構文」
セクション21.5「INFORMATION_SCHEMA COLUMN_PRIVILEGES テーブル」
セクション10.1.9.3「SHOW ステートメントと INFORMATION_SCHEMA」

SHOW FULL PROCESSLIST

セクション4.5.2「mysqladmin — MySQL サーバーの管理を行うクライアント」
セクション8.12.5「スレッド情報の検査」

SHOW FULL TABLES

セクション4.5.6「mysqlshow — データベース、テーブル、およびカラム情報の表示」
セクション13.7.5.38「SHOW TABLES 構文」

SHOW FUNCTION CODE

セクション13.7.5.28「SHOW PROCEDURE CODE 構文」

SHOW FUNCTION STATUS

セクション13.7.5.29「SHOW PROCEDURE STATUS 構文」
セクション20.2.3「ストアドルーチンのメタデータ」

SHOW GLOBAL STATUS

セクション21.8「INFORMATION_SCHEMA GLOBAL_STATUS および SESSION_STATUS テーブル」
セクション18.3.4.4「MySQL Cluster のステータス変数」
セクション5.1.4「サーバーシステム変数」

SHOW GLOBAL VARIABLES

セクション21.9「INFORMATION_SCHEMA GLOBAL_VARIABLES および SESSION_VARIABLES テーブル」

SHOW GRANTS

セクション13.7.1.4「GRANT 構文」
セクション6.2「MySQL アクセス権限システム」
セクション13.7.1.6「REVOKE 構文」
セクション13.7.5.22「SHOW GRANTS 構文」
セクション13.7.5.27「SHOW PRIVILEGES 構文」
セクション6.2.7「アクセス拒否エラーの原因」
セクション6.1.1「セキュリティーガイドライン」
セクション6.3.2「ユーザーアカウントの追加」
セクション6.2.2「権限システム付与テーブル」

SHOW INDEX

セクション13.7.2.1「ANALYZE TABLE 構文」
セクション8.8.2「EXPLAIN 出力フォーマット」
セクション13.8.2「EXPLAIN 構文」

セクション21.21「INFORMATION_SCHEMA STATISTICS テーブル」
セクション21.24「INFORMATION_SCHEMA TABLE_CONSTRAINTS テーブル」
セクション8.3.7「InnoDB および MyISAM インデックス統計コレクション」
セクション14.12「InnoDB の起動オプションおよびシステム変数」
セクション14.6.7「InnoDB テーブル上の制限」
セクション18.4.14「ndb_index_stat — NDB インデックス統計ユーティリティー」
セクション13.7.5.6「SHOW COLUMNS 構文」
セクション13.7.5.23「SHOW INDEX 構文」
セクション4.6.3.4「その他の myisamchk オプション」
セクション13.2.9.3「インデックスヒントの構文」
セクション14.13.16.2「非永続的オプティマイザ統計のパラメータの構成」

SHOW MASTER LOGS

セクション13.7.5.2「SHOW BINARY LOGS 構文」

SHOW MASTER STATUS

セクション17.1.3.1「GTID の概念」
セクション18.6.9「MySQL Cluster レプリケーションを使用した MySQL Cluster バックアップ」
セクション6.2.1「MySQL で提供される権限」
セクション17.1.1.5「mysqldump を使用したデータスナップショットの作成」
セクション13.7.5.35「SHOW SLAVE STATUS 構文」
セクション17.1.4.5「グローバルトランザクション ID のオプションと変数」
セクション13.4.1「マスターサーバーを制御するための SQL ステートメント」
セクション17.4.4「レプリケーションのトラブルシューティング」
セクション17.4.5「レプリケーションバグまたは問題を報告する方法」
セクション17.1.1.4「レプリケーションマスターバイナリログ座標の取得」

SHOW OPEN TABLES

セクション13.7.5.25「SHOW OPEN TABLES 構文」

SHOW PLUGINS

セクション21.30.2「INFORMATION_SCHEMA ndb_transid_mysql_connection_map テーブル」
セクション21.14「INFORMATION_SCHEMA PLUGINS テーブル」
セクション24.2.4.6「INFORMATION_SCHEMA プラグインの作成」
セクション13.7.3.3「INSTALL PLUGIN 構文」
セクションA.10「MySQL 5.6 FAQ: MySQL Cluster」
セクション1.4「MySQL 5.6 の新機能」
セクション18.3.4.2「MySQL Cluster 用の MySQL Server オプション」
セクション18.5.10「ndbinfo MySQL Cluster 情報データベース」
PAM 認証プラグインのインストール
セクション13.7.5.26「SHOW PLUGINS 構文」
Windows 認証プラグインのインストール
セクション5.1.4「サーバーシステム変数」
サーバープラグインライブラリおよびプラグインディスクリプタ
セクション5.1.8.2「サーバープラグイン情報の取得」
サーバー側認証プラグインの作成
セクション8.11.6.1「スレッドプールコンポーネントとインストール」
セクション24.2.4.5「デーモンプラグインの作成」
パスワード検証プラグインのインストール
セクション24.2.4.10「パスワード検証プラグインの作成」
第19章「パーティション化」
セクション6.3.7「プラガブル認証」
セクション24.2.2「プラグイン API のコンポーネント」
セクション24.2.1「プラグイン API の特徴」
セクション5.1.8.1「プラグインのインストールおよびアンインストール」
セクション24.2.4.4「全文パーサープラグインの作成」
セクション17.3.8.2「準同期レプリケーションのインストールと構成」
セクション24.2.4.8「監査プラグインの作成」
セクション6.3.12.1「監査ログプラグインのインストール」

SHOW PRIVILEGES

セクション13.7.5.27「SHOW PRIVILEGES 構文」

SHOW PROCEDURE CODE

セクション13.7.5.20「SHOW FUNCTION CODE 構文」

SHOW PROCEDURE STATUS

セクション13.7.5.21「SHOW FUNCTION STATUS 構文」
セクション20.2.3「ストアドルーチンのメタデータ」

SHOW PROCESSLIST

セクション13.4.2.1「CHANGE MASTER TO 構文」
セクション13.7.1.4「GRANT 構文」
セクション21.30.2「INFORMATION_SCHEMA ndb_transid_mysql_connection_map テーブル」
セクション21.15「INFORMATION_SCHEMA PROCESSLIST テーブル」
セクション14.19.4「InnoDB のエラー処理」
セクション13.7.6.4「KILL 構文」
セクションA.13「MySQL 5.6 FAQ: レプリケーション」
セクション18.5.4「MySQL Cluster での MySQL サーバーの使用法」
セクション6.2.1「MySQL で提供される権限」
セクション23.7.7.43「mysql_list_processes()」
セクション4.5.2「mysqladmin — MySQL サーバーの管理を行うクライアント」
セクション18.5.10.20「ndbinfo server_operations テーブル」
セクション18.5.10.21「ndbinfo server_transactions テーブル」
セクション13.7.5.30「SHOW PROCESSLIST 構文」
セクション13.7.5.31「SHOW PROFILE 構文」
セクション13.7.5.35「SHOW SLAVE STATUS 構文」
セクション13.4.2.5「START SLAVE 構文」
セクション13.3.1「START TRANSACTION、COMMIT、および ROLLBACK 構文」
セクション20.4.2「イベントスケジューラの構成」
セクション5.4.1.3「クエリープローブ」
セクション5.4.1.6「クエリー実行プローブ」
セクション5.4.1.2「コマンドプローブ」
セクション22.9.10.3「スレッドテーブル」
セクション8.12.5「スレッド情報の検査」
セクション22.4「パフォーマンススキーマインストゥルメント命名規則」

セクション22.9.5「パフォーマンススキーマステージイベントテーブル」
セクション17.3.6「フェイルオーバー中にマスターを切り替える」
セクション17.4.4「レプリケーションのトラブルシューティング」
セクション17.1.5.1「レプリケーションステータスの確認」
セクション17.2.1「レプリケーション実装の詳細」
セクション8.12.5.2「一般的なスレッドの状態」
セクション12.14「情報関数」
セクションB.5.2.7「接続が多すぎます」
セクション5.4.1.1「接続プローブ」
セクション6.1.3「攻撃者に対する MySQL のセキュアな状態の維持」
セクション17.3.9「遅延レプリケーション」

SHOW PROFILE

セクション21.16「INFORMATION_SCHEMA PROFILING テーブル」
セクション2.9.4「MySQL ソース構成オプション」
セクション13.7.5.31「SHOW PROFILE 構文」
セクション13.7.5.32「SHOW PROFILES 構文」
セクション5.1.4「サーバーシステム変数」
セクション8.12.5「スレッド情報の検査」
セクション8.12.5.2「一般的なスレッドの状態」

SHOW PROFILES

セクション21.16「INFORMATION_SCHEMA PROFILING テーブル」
セクション2.9.4「MySQL ソース構成オプション」
セクション13.7.5.31「SHOW PROFILE 構文」
セクション13.7.5.32「SHOW PROFILES 構文」
セクション5.1.4「サーバーシステム変数」

SHOW RELAYLOG EVENTS

セクション13.7.5.3「SHOW BINLOG EVENTS 構文」
セクション13.7.5.33「SHOW RELAYLOG EVENTS 構文」
セクション13.4.2「スレーブサーバーを制御するための SQL ステートメント」

SHOW SCHEMAS

セクション13.7.5.15「SHOW DATABASES 構文」

SHOW SESSION STATUS

セクション21.8「INFORMATION_SCHEMA GLOBAL_STATUS および SESSION_STATUS テーブル」
セクション18.3.4.4「MySQL Cluster のステータス変数」

SHOW SESSION VARIABLES

セクション21.9「INFORMATION_SCHEMA GLOBAL_VARIABLES および SESSION_VARIABLES テーブル」

SHOW SLAVE HOSTS

セクション13.4.1「マスターサーバーを制御するための SQL ステートメント」
セクション17.1.4「レプリケーションおよびバイナリロギングのオプションと変数」
セクション17.1.5.1「レプリケーションステータスの確認」
セクション17.1.4.3「レプリケーションスレーブのオプションと変数」

SHOW SLAVE STATUS

セクション13.4.2.1「CHANGE MASTER TO 構文」
セクション17.1.3.1「GTID の概念」
セクションA.13「MySQL 5.6 FAQ: レプリケーション」
セクション18.6.3「MySQL Cluster レプリケーションの既知の問題」
セクション6.2.1「MySQL で提供される権限」
セクション13.4.1.1「PURGE BINARY LOGS 構文」
セクション13.7.5.24「SHOW MASTER STATUS 構文」
セクション13.7.5.35「SHOW SLAVE STATUS 構文」
セクション17.3.7「SSL を使用してレプリケーションをセットアップする」
セクション13.4.2.5「START SLAVE 構文」
セクションB.1「エラー情報のソース」
セクション17.1.4.5「グローバルトランザクション ID のオプションと変数」
セクション13.4.2「スレーブサーバーを制御するための SQL ステートメント」
セクション17.2.2.2「スレーブステータスログ」
セクション17.1.4「レプリケーションおよびバイナリロギングのオプションと変数」
セクション17.4.4「レプリケーションのトラブルシューティング」
セクション17.1.5.1「レプリケーションステータスの確認」
セクション8.12.5.6「レプリケーションスレーブの I/O スレッド状態」
セクション17.1.4.3「レプリケーションスレーブのオプションと変数」
セクション17.4.5「レプリケーションバグまたは問題を報告する方法」
セクション17.4.1.26「レプリケーション中のスレーブエラー」
セクション17.2.1「レプリケーション実装の詳細」
セクション17.3.9「遅延レプリケーション」

SHOW STATUS

セクション13.2.5.2「INSERT DELAYED 構文」
セクション18.5「MySQL Cluster の管理」
セクション18.6「MySQL Cluster レプリケーション」
セクション18.3.2.7「MySQL Cluster 内の SQL ノードおよびその他の API ノードの定義」
セクション18.3.4.2「MySQL Cluster 用の MySQL Server オプション」
セクション18.5.15「NDB API 統計のカウンタと変数」
セクション13.7.5.36「SHOW STATUS 構文」
セクション8.2.1.7「インデックス拡張の使用」
セクション18.5.9「クイックリファレンス: MySQL Cluster の SQL ステートメント」
セクション8.9.3.4「クエリーキャッシュのステータスと保守」
セクション5.1.4「サーバーシステム変数」
セクション5.1.6「サーバーステータス変数」
サーバープラグインのステータス変数およびシステム変数
サーバープラグインライブラリおよびプラグインディスクリプタ
セクションD.1「ストアドプログラムの制約」
セクション24.2.1「プラグイン API の特徴」
セクション24.2.4「プラグインの作成」
セクション17.4.1.22「レプリケーションと一時テーブル」
セクション17.4.1.29「レプリケーション再試行とタイムアウト」
セクション17.2.1「レプリケーション実装の詳細」
セクション24.2.4.4「全文パーサープラグインの作成」

セクション17.3.8.3「準同期レプリケーションモニタリング」
セクション24.2.4.8「監査プラグインの作成」

SHOW STATUS LIKE 'perf%'

セクション22.5「パフォーマンススキーマステータスモニタリング」

SHOW TABLE STATUS

セクション15.5「ARCHIVE ストレージエンジン」
セクション13.1.17「CREATE TABLE 構文」
セクション13.8.2「EXPLAIN 構文」
セクション14.12「InnoDB の起動オプションおよびシステム変数」
セクション14.6.1「InnoDB テーブルの作成」
セクション14.6.7「InnoDB テーブル上の制限」
セクション13.7.5.6「SHOW COLUMNS 構文」
セクション13.7.5.37「SHOW TABLE STATUS 構文」
セクション19.3.5「パーティションに関する情報を取得する」
セクション14.10.2「ファイル領域管理」
セクション14.6.5.1「従来の InnoDB の自動インクリメントロック」
セクション14.2.13.7「物理的な行構造」
セクション14.13.16.2「非永続的オプティマイザ統計のパラメータの構成」

SHOW TABLES

第21章「INFORMATION_SCHEMA テーブル」
セクション14.14「InnoDB INFORMATION_SCHEMA テーブル」
セクション18.6.10「MySQL Cluster レプリケーション: マルチマスターと循環レプリケーション」
セクション18.4.20「ndb_restore — MySQL Cluster バックアップのリストア」
セクション18.5.10「ndbinfo MySQL Cluster 情報データベース」
セクション13.7.5.37「SHOW TABLE STATUS 構文」
セクション13.7.5.38「SHOW TABLES 構文」
セクション21.32「SHOW ステートメントの拡張」
セクションB.5.7.2「TEMPORARY テーブルに関する問題」
セクション5.1.3「サーバーコマンドオプション」
セクション3.3.2「テーブルの作成」
セクションB.5.2.16「表 'tbl_name' は存在しません」
セクション9.2.3「識別子とファイル名のマッピング」
セクション9.2.2「識別子の大文字と小文字の区別」

SHOW TABLES FROM some_ndb_database

セクション18.5.11.2「MySQL Cluster と MySQL 権限」

SHOW TRIGGERS

セクション21.26「INFORMATION_SCHEMA TRIGGERS テーブル」
セクションA.5「MySQL 5.6 FAQ: トリガー」
セクション13.7.5.39「SHOW TRIGGERS 構文」
セクション20.3.2「トリガーのメタデータ」

SHOW VARIABLES

セクション5.3「1 つのマシン上での複数の MySQL インスタンスの実行」
セクションA.11「MySQL 5.6 FAQ: MySQL の中国語、日本語、および韓国語の文字セット」
セクション18.6.6「MySQL Cluster レプリケーションの起動 (レプリケーションチャネルが 1 つ)」
セクション13.7.4「SET 構文」
セクション13.7.5.40「SHOW VARIABLES 構文」
セクション20.4.2「イベントスケジューラの構成」
セクション5.1.4「サーバーシステム変数」
セクション5.1.5「システム変数の使用」
セクション24.2.4.10「パスワード検証プラグインの作成」
セクション22.2.2「パフォーマンススキーマ起動構成」
セクション24.2.1「プラグイン API の特徴」
セクション24.2.4「プラグインの作成」
セクション24.2.4.3「プラグインライブラリのコンパイルおよびインストール」
セクション17.1.4.3「レプリケーションスレーブのオプションと変数」
セクション17.3.8.3「準同期レプリケーションモニタリング」
セクション6.3.12.4「監査ログプラグインの口ギング制御」

SHOW WARNINGS

セクション13.1.7「ALTER TABLE 構文」
セクション13.1.26「DROP PROCEDURE および DROP FUNCTION 構文」
EXISTS 戦略によるサブクエリーの最適化
セクション8.8.3「EXPLAIN EXTENDED 出力フォーマット」
セクション8.8.2「EXPLAIN 出力フォーマット」
セクション13.6.7.3「GET DIAGNOSTICS 構文」
セクション10.4.4.3「Index.xml の構文解析中の診断」
セクション13.2.6「LOAD DATA INFILE 構文」
セクション2.11.1.3「MySQL 5.5 から 5.6 へのアップグレード」
セクション1.7.3.1「PRIMARY KEY および UNIQUE インデックス制約」
セクション13.7.5.18「SHOW ERRORS 構文」
セクション13.7.5.41「SHOW WARNINGS 構文」
セクション13.6.7.5「SIGNAL 構文」
セクションB.1「エラー情報のソース」
サブクエリー実体化によるサブクエリーの最適化
セクションB.3「サーバーのエラーコードおよびメッセージ」
セクション5.1.4「サーバーシステム変数」
シグナルの条件情報項目
セクション17.1.2.1「ステートメントベースおよび行ベースレプリケーションのメリットとデメリット」
ハンドラ、カーソル、およびステートメントに対するシグナルの影響
セクション5.2.4.3「混合形式のバイナリロギング形式」
準結合変換によるサブクエリーの最適化
診断領域が移入される方法
セクション9.2.4「関数名の構文解析と解決」

SIGNAL

セクション13.6.7.1「DECLARE ... CONDITION 構文」
セクション13.6.7.2「DECLARE ... HANDLER 構文」
セクション13.6.7.4「RESIGNAL 構文」
セクション13.6.7.5「SIGNAL 構文」
シグナルの条件情報項目
セクションD.1「ストアドプログラムの制約」
ハンドラ、カーソル、およびステートメントに対するシグナルの影響
セクション13.6.7.6「ハンドラのスコープに関するルール」

セクション12.14「情報関数」
セクション13.6.7「条件の処理」
セクションD.2「条件処理の制約」
診断領域が移入される方法
診断領域の情報項目

START SLAVE

セクション18.6.7「2 つのレプリケーションチャネルを使用する MySQL Cluster レプリケーション」
セクション13.4.2.1「CHANGE MASTER TO 構文」
セクション17.1.3.1「GTID の概念」
セクション1.4「MySQL 5.6 の新機能」
セクション18.6.10「MySQL Cluster レプリケーション: マルチマスターと循環レプリケーション」
セクション18.6.6「MySQL Cluster レプリケーションの起動 (レプリケーションチャネルが 1 つ)」
セクション18.6.8「MySQL Cluster レプリケーションを使用したフェイルオーバーの実装」
セクション4.5.4「mysqldump — データベースバックアッププログラム」
セクション18.4.20「ndb_restore — MySQL Cluster バックアップのリストア」
セクション13.4.2.3「RESET SLAVE 構文」
セクション13.7.5.35「SHOW SLAVE STATUS 構文」
セクション13.4.2.5「START SLAVE 構文」
セクション13.4.2.6「STOP SLAVE 構文」
セクション17.1.4.5「グローバルトランザクション ID のオプションと変数」
セクション17.1.5.2「スレーブでレプリケーションを一時停止する」
セクション6.1.2.3「パスワードおよびロギング」
セクション17.1.3.3「フェイルオーバーおよびスケールアウトでの GTID の使用」
セクション17.3.6「フェイルオーバー中にマスターを切り替える」
セクション17.1.4「レプリケーションおよびバイナリロギングのオプションと変数」
セクション17.4.4「レプリケーションのトラブルシューティング」
セクション17.1.4.3「レプリケーションスレーブのオプションと変数」
セクション17.4.1.26「レプリケーション中のスレーブエラー」
セクション17.2.1「レプリケーション実装の詳細」
セクション13.3.3「暗黙的なコミットを発生させるステートメント」
セクション17.3.8.2「準同期レプリケーションのインストールと構成」
セクション17.3.4「異なるデータベースを異なるスレーブに複製する」
セクション17.3.9「遅延レプリケーション」

START SLAVE UNTIL

セクション17.1.4.3「レプリケーションスレーブのオプションと変数」

START SLAVE UNTIL SQL_AFTER_MTS_GAPS

セクション17.1.4.3「レプリケーションスレーブのオプションと変数」

START TRANSACTION

セクション13.6.1「BEGIN ... END 複合ステートメント構文」
セクション13.7.6.3「FLUSH 構文」
セクション14.19.4「InnoDB のエラー処理」
セクション14.2.2「InnoDB のトランザクションモデルおよびロック」
セクション14.13.14「InnoDB の読み取り専用トランザクションの最適化」
セクション13.3.5「LOCK TABLES および UNLOCK TABLES 構文」
セクション13.3「MySQL トランザクションおよびロックステートメント」
セクション4.5.4「mysqldump — データベースバックアッププログラム」
セクション13.3.6「SET TRANSACTION 構文」
セクション13.3.1「START TRANSACTION、COMMIT、および ROLLBACK 構文」
セクション13.3.7.2「XA トランザクションの状態」
セクション5.1.4「サーバーシステム変数」
セクションD.1「ストアドプログラムの制約」
セクション13.3.5.1「テーブルロックとトランザクションの通信」
セクション14.2.11「デッドロックの対処方法」
セクション14.6.3「トランザクションを使用した DML 操作のグループ化」
セクション20.3.1「トリガーの構文と例」
セクション14.2.5「ロック読み取り (SELECT ... FOR UPDATE および SELECT ... LOCK IN SHARE MODE)」
セクション13.3.3「暗黙的なコミットを発生させるステートメント」
セクション17.3.8「準同期レプリケーション」

START TRANSACTION READ ONLY

セクション14.13.14「InnoDB の読み取り専用トランザクションの最適化」
セクション1.4「MySQL 5.6 の新機能」
MySQL 用語集

START TRANSACTION WITH CONSISTENT SNAPSHOT

セクション14.2.4「一貫性非ロック読み取り」

STATS_PERSISTENT=0

セクション14.13.16.2「非永続的オプティマイザ統計のパラメータの構成」

STOP SLAVE

セクション13.4.2.1「CHANGE MASTER TO 構文」
セクション1.4「MySQL 5.6 の新機能」
セクション4.5.4「mysqldump — データベースバックアッププログラム」
セクション13.4.1.2「RESET MASTER 構文」
セクション13.4.2.3「RESET SLAVE 構文」
セクション13.4.2.5「START SLAVE 構文」
セクション13.4.2.6「STOP SLAVE 構文」
セクション17.1.4.5「グローバルトランザクション ID のオプションと変数」
セクション17.1.5.2「スレーブでレプリケーションを一時停止する」
セクション17.3.6「フェイルオーバー中にマスターを切り替える」

セクション17.1.4「レプリケーションおよびバイナリロギングのオプションと変数」
セクション17.1.5.1「レプリケーションステータスの確認」
セクション17.1.4.3「レプリケーションスレーブのオプションと変数」
セクション17.1.1.9「既存のレプリケーション環境への追加スレーブの導入」
セクション13.3.3「暗黙的なコミットを発生させるステートメント」
セクション17.3.8.2「準同期レプリケーションのインストールと構成」
セクション17.1.2.2「行ベースロギングおよびレプリケーションの使用」
セクション17.3.9「遅延レプリケーション」

STOP SLAVE SQL_THREAD

セクション17.1.2.2「行ベースロギングおよびレプリケーションの使用」

T

[索引の先頭]

TRUNCATE TABLE

セクション22.9.7.1「accounts テーブル」
セクション13.1.19「CREATE TRIGGER 構文」
セクション13.2.2「DELETE 構文」
セクション22.9.5.1「events_stages_current テーブル」
セクション22.9.5.2「events_stages_history テーブル」
セクション22.9.5.3「events_stages_history_long テーブル」
セクション22.9.6.1「events_statements_current テーブル」
セクション22.9.6.2「events_statements_history テーブル」
セクション22.9.6.3「events_statements_history_long テーブル」
セクション22.9.4.1「events_waits_current テーブル」
セクション22.9.4.2「events_waits_history テーブル」
セクション22.9.4.3「events_waits_history_long テーブル」
セクション15.8.3「FEDERATED ストレージエンジンの注記とヒント」
セクション13.2.4「HANDLER 構文」
セクション22.9.10.1「host_cache テーブル」
セクション22.9.7.2「hosts テーブル」
セクション21.29.8「INFORMATION_SCHEMA INNODB_SYS_INDEXES テーブル」
セクション21.29.7「INFORMATION_SCHEMA INNODB_SYS_TABLES テーブル」
セクション8.5.6「InnoDB DDL 操作の最適化」
セクション14.5.2「InnoDB File-Per-Table モード」
セクション14.18.7「InnoDB memcached プラグインの内部構造」
セクション14.12「InnoDB の起動オプションおよびシステム変数」
セクション13.3.5「LOCK TABLES および UNLOCK TABLES 構文」
セクション15.3「MEMORY ストレージエンジン」
セクション15.7.2「MERGE テーブルの問題点」
セクション14.6.4「MyISAM から InnoDB へのテーブルの変換」
セクション18.1.6.3「MySQL Cluster でのトランザクション処理に関する制限」
セクション18.1.6.2「MySQL Cluster の制限と標準の MySQL の制限との違い」
セクション18.5.13.1「MySQL Cluster データノードのオンライン追加: 一般的な問題」
セクション18.6.9.2「MySQL Cluster レプリケーションを使用したポイントインタイムリカバリ」
セクション18.5.2「MySQL Cluster 管理クライアントのコマンド」
セクション6.2.1「MySQL で提供される権限」
セクション5.2.4.4「mysql データベーステーブルへの変更に対するロギング形式」
MySQL 用語集
セクション4.5.4「mysqldump — データベースバックアッププログラム」
セクション18.4.9「ndb_delete_all — NDB テーブルからのすべての行の削除」
セクション19.3.1「RANGE および LIST パーティションの管理」
セクション22.9.2.5「setup_timers テーブル」
table_io_waits_summary_by_index_usage テーブル
table_io_waits_summary_by_table テーブル
table_lock_waits_summary_by_table テーブル
セクション13.1.33「TRUNCATE TABLE 構文」
セクション22.9.7.3「users テーブル」
セクション22.2.3.3「イベントの事前フィルタリング」
セクション22.9.9.1「イベント待機サマリーテーブル」
セクション22.9.9.4「オブジェクト待機サマリーテーブル」
セクション23.7.15.2「クエリーからどんな結果を得ることができるか」
セクション8.9.3.1「クエリーキャッシュの動作」
セクション5.1.4「サーバーシステム変数」
セクション22.9.9.2「ステージサマリーテーブル」
セクション22.9.9.3「ステートメントサマリーテーブル」
セクション22.9.9.8「ソケットサマリーテーブル」
セクション22.8「パフォーマンススキーマの一般的なテーブル特性」
セクション22.2.3.1「パフォーマンススキーマイベントタイミング」
セクション22.9.9「パフォーマンススキーマサマリーテーブル」
セクション22.9.7「パフォーマンススキーマ接続テーブル」
セクション19.3.4「パーティションの保守」
セクション22.9.9.5「ファイル I/O サマリーテーブル」
セクション14.18.6「レプリケーションでの InnoDB memcached プラグインの使用」
セクション17.4.1.21「レプリケーションと MEMORY テーブル」
セクション17.4.1.33「レプリケーションと TRUNCATE TABLE」
セクション5.2.1「一般クエリーログおよびスロークエリーログの出力先の選択」
セクション15.2.3.3「圧縮テーブルの特徴」
セクション22.9.9.7「接続サマリーテーブル」
セクション13.3.3「暗黙的なコミットを発生させるステートメント」

TRUNCATE TABLE host_cache

セクション22.9.10.1「host_cache テーブル」

U

[索引の先頭]

UNINSTALL PLUGIN

セクション13.7.6.3「FLUSH 構文」
セクション21.14「INFORMATION_SCHEMA PLUGINS テーブル」

セクション24.2.4.6「INFORMATION_SCHEMA プラグインの作成」
セクション13.7.3.3「INSTALL PLUGIN 構文」
セクション8.11.4.1「MySQL のメモリーの使用方法」
セクション4.4.4「mysql_plugin — MySQL サーバープラグインの構成」
セクション13.7.5.26「SHOW PLUGINS 構文」
セクション13.7.3.4「UNINSTALL PLUGIN 構文」
サーバープラグインライブラリおよびプラグインディスクリプタ
サーバー側認証プラグインの作成
セクション24.2.4.5「デーモンプラグインの作成」
セクション24.2.4.10「パスワード検証プラグインの作成」
セクション22.14「パフォーマンススキーマとプラグイン」
セクション15.11.1「プラガブルストレージエンジンのアーキテクチャー」
セクション24.2.2「プラグイン API のコンポーネント」
セクション5.1.8.1「プラグインのインストールおよびアンインストール」
セクション24.2.4.4「全文パーサープラグインの作成」
セクション24.2.4.8「監査プラグインの作成」

UNION

セクション3.6.7「2 つのキーを使用した検索」
セクション23.7.5「C API データ構造」
セクション13.1.17「CREATE TABLE 構文」
セクション13.1.20「CREATE VIEW 構文」
セクション8.8.2「EXPLAIN 出力フォーマット」
セクション14.2.8「InnoDB のさまざまな SQL ステートメントで設定されたロック」
セクション15.7「MERGE ストレージエンジン」
セクション8.4.4「MySQL が内部一時テーブルを使用する仕組み」
セクション13.2.9「SELECT 構文」
セクション13.2.9.4「UNION 構文」
セクション12.11「XML 関数」
セクション13.2.10「サブクエリー構文」
セクション5.1.6「サーバーステータス変数」
シングルパートインデックスの range アクセスメソッド
セクションD.5「ビューの制約」
セクション20.5.1「ビューの構文」
セクション20.5.2「ビュー処理アルゴリズム」
セクション12.14「情報関数」
セクション11.2.5「数値型の属性」
セクション20.5.3「更新可能および挿入可能なビュー」
準結合変換によるサブクエリーの最適化
セクション10.1.9.1「結果文字列」

UNION ALL

セクション8.4.4「MySQL が内部一時テーブルを使用する仕組み」
セクション13.2.9.4「UNION 構文」
セクション20.5.2「ビュー処理アルゴリズム」
セクション12.14「情報関数」
セクション20.5.3「更新可能および挿入可能なビュー」

UNION DISTINCT

セクション13.2.9.4「UNION 構文」

UNLOCK TABLES

セクション13.7.6.3「FLUSH 構文」
セクション14.6.7「InnoDB テーブル上の制限」
セクション13.3.5「LOCK TABLES および UNLOCK TABLES 構文」
セクション8.6.2「MyISAM テーブルの一括データロード」
セクション4.5.4「mysqldump — データベースバックアッププログラム」
セクション13.3.1「START TRANSACTION、COMMIT、および ROLLBACK 構文」
セクション8.11.1「システム要素およびスタートアップパラメータのチューニング」
セクションD.1「ストアドプログラムの制約」
セクション14.5.5「テーブルスペースの別のサーバーへのコピー (トランスポータブルテーブルスペース)」
セクション13.3.5.1「テーブルロックとトランザクションの通信」
セクション13.3.5.3「テーブルロックの制限と条件」
セクション14.2.11「デッドロックの対処方法」
セクション7.2「データベースバックアップ方法」
セクション1.7.2.3「トランザクションおよびアトミック操作の違い」
セクション13.3.3「暗黙的なコミットを発生させるステートメント」

UPDATE

--safe-updates オプションの使用
セクション10.1.7.6「_bin および binary 照合順序」
セクション15.5「ARCHIVE ストレージエンジン」
セクション3.6.9「AUTO_INCREMENT の使用」
セクション15.6「BLACKHOLE ストレージエンジン」
セクション23.7.10「C API プリペアドステートメント関数の概要」
セクション23.7.6「C API 関数の概要」
セクション13.7.2.2「CHECK TABLE 構文」
セクション15.8.2.1「CONNECTION を使用した FEDERATED テーブルの作成」
セクション13.1.19「CREATE TRIGGER 構文」
セクション13.1.20「CREATE VIEW 構文」
セクションB.5.5.2「DATE カラムの使用に関する問題」
セクション8.2.2「DML ステートメントの最適化」
セクション8.8.3「EXPLAIN EXTENDED 出力フォーマット」
セクション8.8.1「EXPLAIN によるクエリーの最適化」
セクション8.8.2「EXPLAIN 出力フォーマット」
セクション13.8.2「EXPLAIN 構文」
セクション15.8.3「FEDERATED ストレージエンジンの注記とヒント」
セクション13.7.1.4「GRANT 構文」
セクション21.28「INFORMATION_SCHEMA VIEWS テーブル」
第21章「INFORMATION_SCHEMA テーブル」
セクション14.2.8「InnoDB のさまざまな SQL ステートメントで設定されたロック」
セクション14.19.2「InnoDB のリカバリの強制的な実行」
セクション14.2.6「InnoDB のレコード、ギャップ、およびネクストキーロック」
セクション14.12「InnoDB の起動オプションおよびシステム変数」
セクション8.5.4「InnoDB テーブルの一括データロード」
セクション13.2.5.3「INSERT ... ON DUPLICATE KEY UPDATE 構文」
セクション13.2.5「INSERT 構文」
セクション13.2.9.2「JOIN 構文」
セクション13.7.6.4「KILL 構文」
セクション13.2.6「LOAD DATA INFILE 構文」
セクション15.7「MERGE ストレージエンジン」

セクション14.6.4「MyISAM から InnoDB へのテーブルの変換」
セクション15.2「MyISAM ストレージエンジン」
セクション8.6.2「MyISAM テーブルの一括データロード」
セクションA.4「MySQL 5.6 FAQ: ストアドプロシージャーおよびストアドファンクション」
セクション1.4「MySQL 5.6 の新機能」
セクション18.5.12.1「MySQL Cluster ディスクデータオブジェクト」
セクション18.6.3「MySQL Cluster レプリケーションの既知の問題」
セクション18.6.11「MySQL Cluster レプリケーションの競合解決」
セクション8.4.4「MySQL が内部一時テーブルを使用する仕組み」
セクション6.2.1「MySQL で提供される権限」
セクション8.2.1.2「MySQL の WHERE 句の最適化の方法」
セクション4.5.1.1「mysql のオプション」
セクション19.1「MySQL のパーティショニングの概要」
セクション1.3.2「MySQL の主な機能」
セクションB.5.8「MySQL の既知の問題」
セクション6.2「MySQL アクセス権限システム」
セクション5.2.4.4「mysql データベーステーブルへの変更に対するロギング形式」
MySQL 用語集
セクション23.7.7.1「mysql_affected_rows()」
セクション23.7.7.35「mysql_info()」
セクション23.7.7.37「mysql_insert_id()」
セクション23.7.7.48「mysql_num_rows()」
セクション23.7.7.49「mysql_options()」
セクション23.7.15.1「mysql_query() が成功を返したあとに mysql_store_result() が NULL を返すことがあるのはなぜか」
セクション23.7.11.10「mysql_stmt_execute()」
セクション23.7.11.16「mysql_stmt_insert_id()」
セクション23.7.11.18「mysql_stmt_num_rows()」
セクション4.6.8.2「mysqlbinlog 行イベントの表示」
セクション14.7.6「OLTP ワークロードの圧縮」
セクション13.7.2.4「OPTIMIZE TABLE 構文」
セクション1.7.3.1「PRIMARY KEY および UNIQUE インデックス制約」
root のパスワードのリセット: UNIX システム
root のパスワードのリセット: Windows システム
セクション13.7.1.7「SET PASSWORD 構文」
セクション13.3.6「SET TRANSACTION 構文」
セクション13.7.5.41「SHOW WARNINGS 構文」
セクション1.7.2.2「UPDATE の違い」
セクション13.2.11「UPDATE 構文」
セクション3.3.4.1「すべてのデータの選択」
セクション12.18「その他の関数」
セクション6.2.5「アクセス制御、ステージ 2: リクエストの確認」
セクション6.2.7「アクセス拒否エラーの原因」
セクション14.11.1「オンライン DDL の概要」
セクション10.1.13「カラム文字セットの変換」
セクション23.7.15.2「クエリーからどんな結果を得ることができるか」
セクション8.9.3.1「クエリーキャッシュの動作」
セクション13.2.10.9「サブクエリーのエラー」
セクション13.2.10.11「サブクエリーの結合としての書き換え」
セクション13.2.10「サブクエリー構文」
セクション5.1.7「サーバー SQL モード」
セクション5.1.3「サーバーコマンドオプション」
セクション5.1.4「サーバーシステム変数」
セクション5.1.6「サーバーステータス変数」
セクション5.1.12「シャットダウンプロセス」
セクション5.4.1.12「ステートメントプローブ」
セクション17.1.2.1「ステートメントベースおよび行ベースレプリケーションのメリットとデメリット」
セクション13.3.5.3「テーブルロックの制限と条件」
セクション8.10.2「テーブルロックの問題」
セクション11.6「データ型デフォルト値」
セクション20.3.1「トリガーの構文と例」
セクション5.2.4「バイナリログ」
セクション17.1.4.4「バイナリログのオプションと変数」
セクション22.2.2「パフォーマンススキーマ起動構成」
セクション19.6.4「パーティショニングとロック」
セクション19.6「パーティショニングの制約と制限」
セクション19.4「パーティションプルーニング」
セクション19.5「パーティション選択」
セクションD.5「ビューの制約」
セクション8.9.4「プリペアドステートメントおよびストアドプログラムのキャッシュ」
セクション6.3.2「ユーザーアカウントの追加」
セクション17.4.1.16「レプリケーションと LIMIT」
セクション17.4.1.24「レプリケーションとクエリーオプティマイザ」
セクション17.1.4.3「レプリケーションスレーブのオプションと変数」
セクション17.4.1.26「レプリケーション中のスレーブエラー」
セクション14.2.5「ロック読み取り (SELECT ... FOR UPDATE および SELECT ... LOCK IN SHARE MODE)」
セクション5.2.1「一般クエリーログおよびスロークエリーログの出力先の選択」
セクション8.12.5.2「一般的なスレッドの状態」
セクション14.2.4「一貫性非ロック読み取り」
セクション12.9.5「全文制限」
セクション8.10.1「内部ロック方法」
セクション12.3.4「割り当て演算子」
セクション13.1.17.2「外部キー制約の使用」
セクション12.14「情報関数」
セクション11.1.2「日付と時間型の概要」
セクション20.5.3「更新可能および挿入可能なビュー」
セクション2.10.2「最初の MySQL アカウントのセキュリティー設定」
セクション14.6.5.2「構成可能な InnoDB の自動インクリメントロック」
セクション1.7.1「標準 SQL に対する MySQL 拡張機能」
セクション6.2.2「権限システム付与テーブル」
セクション6.2.6「権限変更が有効化される時期」
セクション12.3「演算子」
セクション1.7.3.3「無効データの制約」
セクション11.2.6「範囲外およびオーバーフローの処理」
セクション17.1.2.2「行ベースロギングおよびレプリケーションの使用」
セクション23.7.17「複数ステートメント実行の C API サポート」
第12章「関数と演算子」
セクション12.1「関数と演算子のリファレンス」

UPDATE ... ()

セクション14.2.4「一貫性非ロック読み取り」

UPDATE ... WHERE ...

セクション14.2.8「InnoDB のさまざまな SQL ステートメントで設定されたロック」

UPDATE IGNORE
セクション13.2.11「UPDATE 構文」
セクション5.1.7「サーバー SQL モード」

UPDATE t1,t2 ...
セクション5.4.1.12「ステートメントプローブ」

USE
セクション13.1.15「CREATE PROCEDURE および CREATE FUNCTION 構文」
第21章「INFORMATION_SCHEMA テーブル」
セクション4.5.1.1「mysql のオプション」
セクション4.6.8「mysqlbinlog — バイナリログファイルを処理するためのユーティリティー」
セクション4.5.4「mysqldump — データベースバックアッププログラム」
セクション7.4.1「mysqldump による SQL フォーマットでのデータのダンプ」
セクション18.5.10「ndbinfo MySQL Cluster 情報データベース」
セクション7.4.2「SQL フォーマットバックアップのリロード」
セクション13.8.4「USE 構文」
セクション13.2.9.3「インデックスヒントの構文」
セクション7.4.5.2「サーバー間でのデータベースのコピー」
セクション20.2.1「ストアドルーチンの構文」
セクション3.3「データベースの作成と使用」
セクション3.3.1「データベースの作成と選択」
セクション17.2.3.1「データベースレベルレプリケーションオプションおよびバイナリロギングオプションの評価」
セクション17.1.4.4「バイナリログのオプションと変数」
セクション17.1.4.3「レプリケーションスレーブのオプションと変数」
セクション17.2.3.3「レプリケーションルールの適用」

USE db2
セクション4.6.8「mysqlbinlog — バイナリログファイルを処理するためのユーティリティー」

USE db_name
セクション4.5.1.1「mysql のオプション」

USE test
セクション4.6.8「mysqlbinlog — バイナリログファイルを処理するためのユーティリティー」

W

[索引の先頭]

WHILE
セクション13.6.5.3「ITERATE 構文」
セクション13.6.5.4「LEAVE 構文」
セクション13.6.5.8「WHILE 構文」
セクション13.6.2「ステートメントラベルの構文」
セクション13.6.5「フロー制御ステートメント」

X

[索引の先頭]

XA COMMIT
セクション13.3.7.2「XA トランザクションの状態」
セクション5.1.4「サーバーシステム変数」

XA END
セクション13.3.7.1「XA トランザクションの SQL 構文」
セクションD.6「XA トランザクションの制約」
セクション13.3.7.2「XA トランザクションの状態」

XA PREPARE
セクション13.3.7.2「XA トランザクションの状態」

XA RECOVER
セクション13.3.7.1「XA トランザクションの SQL 構文」
セクション13.3.7.2「XA トランザクションの状態」

XA ROLLBACK
セクション13.3.7.2「XA トランザクションの状態」
セクション5.1.4「サーバーシステム変数」

XA START
セクション13.3.7.1「XA トランザクションの SQL 構文」
セクションD.6「XA トランザクションの制約」
セクション13.3.7.2「XA トランザクションの状態」

XA START xid
セクション13.3.7.1「XA トランザクションの SQL 構文」

ステータス変数の索引

A | B | C | D | F | H | I | K | L | M | N | O | P | Q | R | S | T | U

A

[索引の先頭]

Aborted_clients
セクション5.1.6「サーバーステータス変数」
セクションB.5.2.11「通信エラーおよび中止された接続」

Aborted_connects
セクション5.1.6「サーバーステータス変数」
セクションB.5.2.11「通信エラーおよび中止された接続」

Audit_log_current_size
セクション6.3.12.7「監査ログプラグインのステータス変数」

Audit_log_event_max_drop_size
セクション6.3.12.7「監査ログプラグインのステータス変数」

Audit_log_events
セクション6.3.12.7「監査ログプラグインのステータス変数」

Audit_log_events_filtered
セクション6.3.12.7「監査ログプラグインのステータス変数」

Audit_log_events_lost
セクション6.3.12.7「監査ログプラグインのステータス変数」

Audit_log_events_written
セクション6.3.12.7「監査ログプラグインのステータス変数」

Audit_log_total_size
セクション6.3.12.7「監査ログプラグインのステータス変数」

Audit_log_write_waits
セクション6.3.12.7「監査ログプラグインのステータス変数」

B

[索引の先頭]

Binlog_cache_disk_use
セクション5.1.6「サーバーステータス変数」
セクション5.2.4「バイナリログ」
セクション17.1.4.4「バイナリログのオプションと変数」

Binlog_cache_use
セクション5.1.6「サーバーステータス変数」
セクション5.2.4「バイナリログ」
セクション17.1.4.4「バイナリログのオプションと変数」

Binlog_stmt_cache_disk_use
セクション5.1.6「サーバーステータス変数」
セクション17.1.4.4「バイナリログのオプションと変数」

Binlog_stmt_cache_use
セクション5.1.6「サーバーステータス変数」
セクション17.1.4.4「バイナリログのオプションと変数」

Bytes_received
セクション5.1.6「サーバーステータス変数」

Bytes_sent
セクション5.1.6「サーバーステータス変数」

C

[索引の先頭]

Com_flush
セクション5.1.6「サーバーステータス変数」

Com_stmt_reprepare
セクション8.9.4「プリペアドステートメントおよびストアドプログラムのキャッシュ」

Compression
セクション5.1.6「サーバーステータス変数」

Connection_errors_accept
セクション5.1.6「サーバーステータス変数」

Connection_errors_internal
セクション5.1.6「サーバーステータス変数」

Connection_errors_max_connections
セクション5.1.4「サーバーシステム変数」
セクション5.1.6「サーバーステータス変数」

Connection_errors_peer_addr
セクション5.1.6「サーバーステータス変数」

Connection_errors_select
セクション5.1.6「サーバーステータス変数」

Connection_errors_tcpwrap
セクション5.1.6「サーバーステータス変数」

Connection_errors_xxx
セクション8.11.5.2「DNS ルックアップの最適化とホストキャッシュ」
セクション1.4「MySQL 5.6 の新機能」
セクション5.1.6「サーバーステータス変数」

Connections
セクション5.1.4「サーバーシステム変数」
セクション5.1.6「サーバーステータス変数」

Created_tmp_disk_tables
セクション22.9.6.1「events_statements_current テーブル」
セクション8.4.4「MySQL が内部一時テーブルを使用する仕組み」
セクション5.1.4「サーバーシステム変数」
セクション5.1.6「サーバーステータス変数」

Created_tmp_files
セクション5.1.6「サーバーステータス変数」

Created_tmp_tables
セクション22.9.6.1「events_statements_current テーブル」
セクション8.4.4「MySQL が内部一時テーブルを使用する仕組み」
セクション13.7.5.36「SHOW STATUS 構文」
セクション5.1.4「サーバーシステム変数」
セクション5.1.6「サーバーステータス変数」

D

[索引の先頭]

Delayed_errors
セクション5.1.6「サーバーステータス変数」

Delayed_insert_threads
セクション13.2.5.2「INSERT DELAYED 構文」
セクション5.1.6「サーバーステータス変数」

Delayed_writes
セクション13.2.5.2「INSERT DELAYED 構文」
セクション5.1.6「サーバーステータス変数」

F

[索引の先頭]

Flush_commands
セクション5.1.6「サーバーステータス変数」

H

[索引の先頭]

Handler_commit
セクション5.1.6「サーバーステータス変数」

Handler_delete
セクション5.1.6「サーバーステータス変数」

Handler_discover
セクション18.3.4.4「MySQL Cluster のステータス変数」

Handler_external_lock
セクション5.1.6「サーバーステータス変数」

Handler_mrr_init
セクション5.1.6「サーバーステータス変数」

Handler_prepare
セクション5.1.6「サーバーステータス変数」

Handler_read_first
セクション5.1.6「サーバーステータス変数」

Handler_read_key
セクション5.1.6「サーバーステータス変数」

Handler_read_last
セクション5.1.6「サーバーステータス変数」

Handler_read_next
セクション8.2.1.7「インデックス拡張の使用」
セクション5.1.6「サーバーステータス変数」

Handler_read_prev
セクション5.1.6「サーバーステータス変数」

Handler_read_rnd
セクション5.1.6「サーバーステータス変数」

Handler_read_rnd_next
セクション5.1.6「サーバーステータス変数」

Handler_rollback
セクション5.1.6「サーバーステータス変数」

Handler_savepoint
セクション5.1.6「サーバーステータス変数」

Handler_savepoint_rollback
セクション5.1.6「サーバーステータス変数」

Handler_update
セクション5.1.6「サーバーステータス変数」

Handler_write
セクション5.1.6「サーバーステータス変数」

I

[索引の先頭]

Innodb_available_undo_logs
セクション14.12「InnoDB の起動オプションおよびシステム変数」
セクション5.1.6「サーバーステータス変数」

Innodb_buffer_pool_bytes_data
セクション5.1.6「サーバーステータス変数」

Innodb_buffer_pool_bytes_dirty
セクション5.1.6「サーバーステータス変数」

Innodb_buffer_pool_dump_status
セクション5.1.6「サーバーステータス変数」

Innodb_buffer_pool_load_status
セクション5.1.6「サーバーステータス変数」

Innodb_buffer_pool_pages_data
セクション5.1.6「サーバーステータス変数」

Innodb_buffer_pool_pages_dirty
セクション5.1.6「サーバーステータス変数」

Innodb_buffer_pool_pages_flushed
セクション5.1.6「サーバーステータス変数」

Innodb_buffer_pool_pages_free
セクション5.1.6「サーバーステータス変数」

Innodb_buffer_pool_pages_latched
セクション5.1.6「サーバーステータス変数」

Innodb_buffer_pool_pages_misc
セクション5.1.6「サーバーステータス変数」

Innodb_buffer_pool_pages_total
セクション5.1.6「サーバーステータス変数」

Innodb_buffer_pool_read_ahead
セクション14.12「InnoDB の起動オプションおよびシステム変数」
セクション14.13.1.1「InnoDB バッファープールのプリフェッチ (先読み) の構成」
セクション5.1.6「サーバーステータス変数」

Innodb_buffer_pool_read_ahead_evicted
セクション14.12「InnoDB の起動オプションおよびシステム変数」
セクション14.13.1.1「InnoDB バッファープールのプリフェッチ (先読み) の構成」
セクション5.1.6「サーバーステータス変数」

Innodb_buffer_pool_read_requests
セクション5.1.6「サーバーステータス変数」

Innodb_buffer_pool_reads
セクション5.1.6「サーバーステータス変数」

Innodb_buffer_pool_wait_free
セクション5.1.6「サーバーステータス変数」

Innodb_buffer_pool_write_requests
セクション5.1.6「サーバーステータス変数」

Innodb_data_fsyncs
セクション14.12「InnoDB の起動オプションおよびシステム変数」
セクション5.1.6「サーバーステータス変数」

Innodb_data_pending_fsyncs
セクション5.1.6「サーバーステータス変数」

Innodb_data_pending_reads
セクション5.1.6「サーバーステータス変数」

Innodb_data_pending_writes
セクション5.1.6「サーバーステータス変数」

Innodb_data_read
セクション5.1.6「サーバーステータス変数」

Innodb_data_reads
セクション5.1.6「サーバーステータス変数」

Innodb_data_writes
セクション5.1.6「サーバーステータス変数」

Innodb_data_written
セクション5.1.6「サーバーステータス変数」

Innodb_dblwr_pages_written
セクション5.1.6「サーバーステータス変数」

Innodb_dblwr_writes
セクション5.1.6「サーバーステータス変数」

Innodb_have_atomic_builtins
セクション5.1.6「サーバーステータス変数」

Innodb_log_waits
セクション5.1.6「サーバーステータス変数」

Innodb_log_write_requests
セクション5.1.6「サーバーステータス変数」

Innodb_log_writes
セクション5.1.6「サーバーステータス変数」

Innodb_num_open_files
セクション5.1.6「サーバーステータス変数」

Innodb_os_log_fsyncs
セクション5.1.6「サーバーステータス変数」

Innodb_os_log_pending_fsyncs
セクション5.1.6「サーバーステータス変数」

Innodb_os_log_pending_writes
セクション5.1.6「サーバーステータス変数」

Innodb_os_log_written
セクション5.1.6「サーバーステータス変数」

Innodb_page_size
セクション5.1.6「サーバーステータス変数」

Innodb_pages_created
セクション5.1.6「サーバーステータス変数」

Innodb_pages_read
セクション5.1.6「サーバーステータス変数」

Innodb_pages_written
セクション5.1.6「サーバーステータス変数」

Innodb_row_lock_current_waits
セクション5.1.6「サーバーステータス変数」

Innodb_row_lock_time
セクション5.1.6「サーバーステータス変数」

Innodb_row_lock_time_avg
セクション5.1.6「サーバーステータス変数」

Innodb_row_lock_time_max
セクション5.1.6「サーバーステータス変数」

Innodb_row_lock_waits
セクション5.1.6「サーバーステータス変数」

Innodb_rows_deleted
セクション5.1.6「サーバーステータス変数」

Innodb_rows_inserted
セクション5.1.6「サーバーステータス変数」

Innodb_rows_read
セクション5.1.6「サーバーステータス変数」

Innodb_rows_updated
セクション5.1.6「サーバーステータス変数」

Innodb_truncated_status_writes
セクション5.1.6「サーバーステータス変数」

K

[索引の先頭]

Key_blocks_not_flushed
セクション5.1.6「サーバーステータス変数」

Key_blocks_unused
セクション5.1.4「サーバーシステム変数」
セクション5.1.6「サーバーステータス変数」

Key_blocks_used
セクション5.1.6「サーバーステータス変数」

Key_read_requests
セクション5.1.4「サーバーシステム変数」
セクション5.1.6「サーバーステータス変数」

Key_reads
セクション5.1.4「サーバーシステム変数」
セクション5.1.6「サーバーステータス変数」

Key_write_requests
セクション5.1.4「サーバーシステム変数」
セクション5.1.6「サーバーステータス変数」

Key_writes
セクション5.1.4「サーバーシステム変数」
セクション5.1.6「サーバーステータス変数」

L

[索引の先頭]

Last_query_cost
セクション5.1.6「サーバーステータス変数」

Last_query_partial_plans
セクション5.1.6「サーバーステータス変数」

M

[索引の先頭]

Max_used_connections
セクション13.7.6.3「FLUSH 構文」
セクション5.1.6「サーバーステータス変数」

N

[索引の先頭]

Ndb_api_bytes_received_count
セクション18.3.4.4「MySQL Cluster のステータス変数」
セクション18.5.15「NDB API 統計のカウンタと変数」

Ndb_api_bytes_received_count_session
セクション18.3.4.4「MySQL Cluster のステータス変数」
セクション18.5.15「NDB API 統計のカウンタと変数」

Ndb_api_bytes_received_count_slave
セクション18.3.4.4「MySQL Cluster のステータス変数」
セクション18.5.15「NDB API 統計のカウンタと変数」

Ndb_api_bytes_sent_count
セクション18.3.4.4「MySQL Cluster のステータス変数」
セクション18.5.15「NDB API 統計のカウンタと変数」

Ndb_api_bytes_sent_count_session
セクション18.3.4.4「MySQL Cluster のステータス変数」
セクション18.5.15「NDB API 統計のカウンタと変数」

Ndb_api_bytes_sent_count_slave
セクション18.3.4.4「MySQL Cluster のステータス変数」
セクション18.5.15「NDB API 統計のカウンタと変数」

Ndb_api_event_bytes_count
セクション18.3.4.4「MySQL Cluster のステータス変数」
セクション18.5.15「NDB API 統計のカウンタと変数」

Ndb_api_event_bytes_count_injector
セクション18.3.4.4「MySQL Cluster のステータス変数」
セクション18.5.15「NDB API 統計のカウンタと変数」

Ndb_api_event_data_count
セクション18.3.4.4「MySQL Cluster のステータス変数」
セクション18.5.15「NDB API 統計のカウンタと変数」

Ndb_api_event_data_count_injector
セクション18.3.4.4「MySQL Cluster のステータス変数」
セクション18.5.15「NDB API 統計のカウンタと変数」

Ndb_api_event_nondata_count
セクション18.3.4.4「MySQL Cluster のステータス変数」
セクション18.5.15「NDB API 統計のカウンタと変数」

Ndb_api_event_nondata_count_injector
セクション18.3.4.4「MySQL Cluster のステータス変数」
セクション18.5.15「NDB API 統計のカウンタと変数」

Ndb_api_pk_op_count
セクション18.3.4.4「MySQL Cluster のステータス変数」
セクション18.5.15「NDB API 統計のカウンタと変数」

Ndb_api_pk_op_count_session
セクション18.3.4.4「MySQL Cluster のステータス変数」
セクション18.5.15「NDB API 統計のカウンタと変数」

Ndb_api_pk_op_count_slave
セクション18.3.4.4「MySQL Cluster のステータス変数」
セクション18.5.15「NDB API 統計のカウンタと変数」

Ndb_api_pruned_scan_count
セクション18.3.4.4「MySQL Cluster のステータス変数」
セクション18.5.15「NDB API 統計のカウンタと変数」

Ndb_api_pruned_scan_count_session
セクション18.3.4.4「MySQL Cluster のステータス変数」
セクション18.5.15「NDB API 統計のカウンタと変数」

Ndb_api_pruned_scan_count_slave
セクション18.3.4.4「MySQL Cluster のステータス変数」
セクション18.5.15「NDB API 統計のカウンタと変数」

Ndb_api_range_scan_count
セクション18.3.4.4「MySQL Cluster のステータス変数」
セクション18.5.15「NDB API 統計のカウンタと変数」

Ndb_api_range_scan_count_session
セクション18.3.4.4「MySQL Cluster のステータス変数」
セクション18.5.15「NDB API 統計のカウンタと変数」

Ndb_api_range_scan_count_slave
セクション18.3.4.4「MySQL Cluster のステータス変数」
セクション18.5.15「NDB API 統計のカウンタと変数」

Ndb_api_read_row_count
セクション18.3.4.4「MySQL Cluster のステータス変数」
セクション18.5.15「NDB API 統計のカウンタと変数」

Ndb_api_read_row_count_session
セクション18.3.4.4「MySQL Cluster のステータス変数」
セクション18.5.15「NDB API 統計のカウンタと変数」

Ndb_api_read_row_count_slave
セクション18.3.4.4「MySQL Cluster のステータス変数」
セクション18.5.15「NDB API 統計のカウンタと変数」

Ndb_api_scan_batch_count
セクション18.3.4.4「MySQL Cluster のステータス変数」
セクション18.5.15「NDB API 統計のカウンタと変数」

Ndb_api_scan_batch_count_session
セクション18.3.4.4「MySQL Cluster のステータス変数」
セクション18.5.15「NDB API 統計のカウンタと変数」

Ndb_api_scan_batch_count_slave
セクション18.3.4.4「MySQL Cluster のステータス変数」
セクション18.5.15「NDB API 統計のカウンタと変数」

Ndb_api_table_scan_count
セクション18.3.4.4「MySQL Cluster のステータス変数」
セクション18.5.15「NDB API 統計のカウンタと変数」

Ndb_api_table_scan_count_session
セクション18.3.4.4「MySQL Cluster のステータス変数」
セクション18.5.15「NDB API 統計のカウンタと変数」

Ndb_api_table_scan_count_slave
セクション18.3.4.4「MySQL Cluster のステータス変数」
セクション18.5.15「NDB API 統計のカウンタと変数」

Ndb_api_trans_abort_count
セクション18.3.4.4「MySQL Cluster のステータス変数」
セクション18.5.15「NDB API 統計のカウンタと変数」

Ndb_api_trans_abort_count_session
セクション18.3.4.4「MySQL Cluster のステータス変数」
セクション18.5.15「NDB API 統計のカウンタと変数」

Ndb_api_trans_abort_count_slave
セクション18.3.4.4「MySQL Cluster のステータス変数」
セクション18.5.15「NDB API 統計のカウンタと変数」

Ndb_api_trans_close_count
セクション18.3.4.4「MySQL Cluster のステータス変数」
セクション18.5.15「NDB API 統計のカウンタと変数」

Ndb_api_trans_close_count_session
セクション18.3.4.4「MySQL Cluster のステータス変数」
セクション18.5.15「NDB API 統計のカウンタと変数」

Ndb_api_trans_close_count_slave
セクション18.3.4.4「MySQL Cluster のステータス変数」
セクション18.5.15「NDB API 統計のカウンタと変数」

Ndb_api_trans_commit_count
セクション18.3.4.4「MySQL Cluster のステータス変数」
セクション18.5.15「NDB API 統計のカウンタと変数」

Ndb_api_trans_commit_count_session
セクション18.3.4.4「MySQL Cluster のステータス変数」
セクション18.5.15「NDB API 統計のカウンタと変数」

Ndb_api_trans_commit_count_slave
セクション18.3.4.4「MySQL Cluster のステータス変数」
セクション18.5.15「NDB API 統計のカウンタと変数」

Ndb_api_trans_local_read_row_count
セクション18.3.4.4「MySQL Cluster のステータス変数」
セクション18.5.15「NDB API 統計のカウンタと変数」

Ndb_api_trans_local_read_row_count_session
セクション18.3.4.4「MySQL Cluster のステータス変数」
セクション18.5.15「NDB API 統計のカウンタと変数」

Ndb_api_trans_local_read_row_count_slave
セクション18.3.4.4「MySQL Cluster のステータス変数」
セクション18.5.15「NDB API 統計のカウンタと変数」

Ndb_api_trans_start_count
セクション18.3.4.4「MySQL Cluster のステータス変数」
セクション18.5.15「NDB API 統計のカウンタと変数」

Ndb_api_trans_start_count_session
セクション18.3.4.4「MySQL Cluster のステータス変数」
セクション18.5.15「NDB API 統計のカウンタと変数」

Ndb_api_trans_start_count_slave
セクション18.3.4.4「MySQL Cluster のステータス変数」
セクション18.5.15「NDB API 統計のカウンタと変数」

Ndb_api_uk_op_count
セクション18.3.4.4「MySQL Cluster のステータス変数」
セクション18.5.15「NDB API 統計のカウンタと変数」

Ndb_api_uk_op_count_session
セクション18.3.4.4「MySQL Cluster のステータス変数」
セクション18.5.15「NDB API 統計のカウンタと変数」

Ndb_api_uk_op_count_slave
セクション18.3.4.4「MySQL Cluster のステータス変数」
セクション18.5.15「NDB API 統計のカウンタと変数」

Ndb_api_wait_exec_complete_count
セクション18.3.4.4「MySQL Cluster のステータス変数」
セクション18.5.15「NDB API 統計のカウンタと変数」

Ndb_api_wait_exec_complete_count_session
セクション18.3.4.4「MySQL Cluster のステータス変数」
セクション18.5.15「NDB API 統計のカウンタと変数」

Ndb_api_wait_exec_complete_count_slave
セクション18.3.4.4「MySQL Cluster のステータス変数」
セクション18.5.15「NDB API 統計のカウンタと変数」

Ndb_api_wait_meta_request_count
セクション18.3.4.4「MySQL Cluster のステータス変数」
セクション18.5.15「NDB API 統計のカウンタと変数」

Ndb_api_wait_meta_request_count_session
セクション18.3.4.4「MySQL Cluster のステータス変数」
セクション18.5.15「NDB API 統計のカウンタと変数」

Ndb_api_wait_meta_request_count_slave
セクション18.3.4.4「MySQL Cluster のステータス変数」
セクション18.5.15「NDB API 統計のカウンタと変数」

Ndb_api_wait_nanos_count
セクション18.3.4.4「MySQL Cluster のステータス変数」
セクション18.5.15「NDB API 統計のカウンタと変数」

Ndb_api_wait_nanos_count_session
セクション18.3.4.4「MySQL Cluster のステータス変数」
セクション18.5.15「NDB API 統計のカウンタと変数」

Ndb_api_wait_nanos_count_slave
セクション18.3.4.4「MySQL Cluster のステータス変数」
セクション18.5.15「NDB API 統計のカウンタと変数」

Ndb_api_wait_scan_result_count
セクション18.3.4.4「MySQL Cluster のステータス変数」
セクション18.5.15「NDB API 統計のカウンタと変数」

Ndb_api_wait_scan_result_count_session
セクション18.3.4.4「MySQL Cluster のステータス変数」
セクション18.5.15「NDB API 統計のカウンタと変数」

Ndb_api_wait_scan_result_count_slave
セクション18.3.4.4「MySQL Cluster のステータス変数」
セクション18.5.15「NDB API 統計のカウンタと変数」

Ndb_cluster_node_id
セクション18.3.4.4「MySQL Cluster のステータス変数」

Ndb_config_from_host
セクション18.3.4.4「MySQL Cluster のステータス変数」

Ndb_config_from_port
セクション18.3.4.4「MySQL Cluster のステータス変数」

Ndb_conflict_fn_epoch
セクション18.6.11「MySQL Cluster レプリケーションの競合解決」

Ndb_conflict_fn_epoch_trans
セクション18.3.4.4「MySQL Cluster のステータス変数」
セクション18.6.11「MySQL Cluster レプリケーションの競合解決」

Ndb_conflict_fn_max
セクション18.6.11「MySQL Cluster レプリケーションの競合解決」

Ndb_conflict_fn_old
セクション18.6.11「MySQL Cluster レプリケーションの競合解決」

Ndb_conflict_trans_conflict_commit_count
セクション18.3.4.4「MySQL Cluster のステータス変数」

Ndb_conflict_trans_row_conflict_count
セクション18.3.4.4「MySQL Cluster のステータス変数」

Ndb_conflict_trans_row_reject_count
セクション18.6.11「MySQL Cluster レプリケーションの競合解決」

Ndb_execute_count
セクション18.3.4.4「MySQL Cluster のステータス変数」

Ndb_number_of_data_nodes
セクション18.3.4.4「MySQL Cluster のステータス変数」

Ndb_pruned_scan_count
セクション18.3.4.4「MySQL Cluster のステータス変数」

Ndb_pushed_queries_defined
セクション18.3.4.3「MySQL Cluster のシステム変数」
セクション18.3.4.4「MySQL Cluster のステータス変数」

Ndb_pushed_queries_dropped
セクション18.3.4.3「MySQL Cluster のシステム変数」
セクション18.3.4.4「MySQL Cluster のステータス変数」

Ndb_pushed_queries_executed
セクション18.3.4.3「MySQL Cluster のシステム変数」
セクション18.3.4.4「MySQL Cluster のステータス変数」

Ndb_pushed_reads
セクション18.3.4.3「MySQL Cluster のシステム変数」
セクション18.3.4.4「MySQL Cluster のステータス変数」

Ndb_scan_count
セクション18.3.4.4「MySQL Cluster のステータス変数」

Ndb_slave_last_conflict_epoch
セクション18.3.4.4「MySQL Cluster のステータス変数」

Ndb_slave_max_replicated_epoch
セクション18.3.4.4「MySQL Cluster のステータス変数」

Not_flushed_delayed_rows
セクション13.2.5.2「INSERT DELAYED 構文」
セクション5.1.6「サーバーステータス変数」

O

[索引の先頭]

Open_files
セクション5.1.6「サーバーステータス変数」

Open_streams
セクション5.1.6「サーバーステータス変数」

Open_table_definitions
セクション5.1.6「サーバーステータス変数」

Open_tables
セクション5.1.6「サーバーステータス変数」

Opened_files
セクション5.1.6「サーバーステータス変数」

Opened_table_definitions
セクション5.1.6「サーバーステータス変数」

Opened_tables
セクション8.4.3.1「MySQL でのテーブルのオープンとクローズの方法」
セクション5.1.4「サーバーシステム変数」
セクション5.1.6「サーバーステータス変数」

P

[索引の先頭]

Performance_schema_digest_lost
セクション22.12「パフォーマンススキーマシステム変数」

Performance_schema_mutex_classes_lost
セクション22.5「パフォーマンススキーマステータスモニタリング」

Performance_schema_mutex_instances_lost
セクション22.5「パフォーマンススキーマステータスモニタリング」

Performance_schema_session_connect_attrs
セクション22.12「パフォーマンススキーマシステム変数」

Performance_schema_thread_instances_lost
セクション22.12「パフォーマンススキーマシステム変数」

Prepared_stmt_count
セクション5.1.6「サーバーステータス変数」

Q

[索引の先頭]

Qcache_free_blocks
セクション8.9.3.4「クエリーキャッシュのステータスと保守」
セクション8.9.3.3「クエリーキャッシュの構成」
セクション5.1.6「サーバーステータス変数」

Qcache_free_memory
セクション5.1.6「サーバーステータス変数」

Qcache_hits
セクション8.9.3.1「クエリーキャッシュの動作」

セクション5.1.6「サーバーステータス変数」

Qcache_inserts
セクション5.1.6「サーバーステータス変数」

Qcache_lowmem_prunes
セクション8.9.3.4「クエリーキャッシュのステータスと保守」
セクション8.9.3.3「クエリーキャッシュの構成」
セクション5.1.6「サーバーステータス変数」

Qcache_not_cached
セクション5.1.6「サーバーステータス変数」

Qcache_queries_in_cache
セクション5.1.6「サーバーステータス変数」

Qcache_total_blocks
セクション8.9.3.4「クエリーキャッシュのステータスと保守」
セクション8.9.3.3「クエリーキャッシュの構成」
セクション5.1.6「サーバーステータス変数」

Queries
セクション5.1.6「サーバーステータス変数」

query_cache_min_res_unit
セクション8.9.3.3「クエリーキャッシュの構成」

Questions
セクション4.5.2「mysqladmin — MySQL サーバーの管理を行うクライアント」
セクション5.1.6「サーバーステータス変数」

R

[索引の先頭]

Rpl_semi_sync_master_clients
セクション5.1.6「サーバーステータス変数」
セクション17.3.8.3「準同期レプリケーションモニタリング」
セクション17.3.8.1「準同期レプリケーション管理インタフェース」

Rpl_semi_sync_master_net_avg_wait_time
セクション5.1.6「サーバーステータス変数」

Rpl_semi_sync_master_net_wait_time
セクション5.1.6「サーバーステータス変数」

Rpl_semi_sync_master_net_waits
セクション5.1.6「サーバーステータス変数」

Rpl_semi_sync_master_no_times
セクション5.1.6「サーバーステータス変数」

Rpl_semi_sync_master_no_tx
セクション5.1.6「サーバーステータス変数」

セクション17.3.8.3「準同期レプリケーションモニタリング」
セクション17.3.8.1「準同期レプリケーション管理インタフェース」

Rpl_semi_sync_master_status
セクション5.1.6「サーバーステータス変数」
セクション17.3.8.3「準同期レプリケーションモニタリング」
セクション17.3.8.1「準同期レプリケーション管理インタフェース」

Rpl_semi_sync_master_timefunc_failures
セクション5.1.6「サーバーステータス変数」

Rpl_semi_sync_master_tx_avg_wait_time
セクション5.1.6「サーバーステータス変数」

Rpl_semi_sync_master_tx_wait_time
セクション5.1.6「サーバーステータス変数」

Rpl_semi_sync_master_tx_waits
セクション5.1.6「サーバーステータス変数」

Rpl_semi_sync_master_wait_pos_backtraverse
セクション5.1.6「サーバーステータス変数」

Rpl_semi_sync_master_wait_sessions
セクション5.1.6「サーバーステータス変数」

Rpl_semi_sync_master_yes_tx
セクション5.1.6「サーバーステータス変数」
セクション17.3.8.3「準同期レプリケーションモニタリング」
セクション17.3.8.1「準同期レプリケーション管理インタフェース」

Rpl_semi_sync_slave_status
セクション5.1.6「サーバーステータス変数」
セクション17.3.8.3「準同期レプリケーションモニタリング」
セクション17.3.8.1「準同期レプリケーション管理インタフェース」

Rsa_public_key
セクション6.3.8.4「SHA-256 認証プラグイン」
セクション5.1.6「サーバーステータス変数」

S

[索引の先頭]

Select_full_join
セクション22.9.6.1「events_statements_current テーブル」
セクション5.1.6「サーバーステータス変数」

Select_full_range_join
セクション22.9.6.1「events_statements_current テーブル」

セクション5.1.6「サーバーステータス変数」

Select_range
セクション22.9.6.1「events_statements_current テーブル」
セクション5.1.6「サーバーステータス変数」

Select_range_check
セクション22.9.6.1「events_statements_current テーブル」
セクション5.1.6「サーバーステータス変数」

Select_scan
セクション22.9.6.1「events_statements_current テーブル」
セクション5.1.6「サーバーステータス変数」

Slave_heartbeat_period
セクション5.1.6「サーバーステータス変数」

Slave_last_heartbeat
セクション5.1.6「サーバーステータス変数」

Slave_open_temp_tables
セクション5.1.6「サーバーステータス変数」
セクション17.4.1.22「レプリケーションと一時テーブル」

Slave_received_heartbeats
セクション5.1.6「サーバーステータス変数」

Slave_retried_transactions
セクション5.1.6「サーバーステータス変数」

Slave_running
セクション13.7.5.35「SHOW SLAVE STATUS 構文」
セクション5.1.6「サーバーステータス変数」
セクション17.2.1「レプリケーション実装の詳細」

Slow_launch_threads
セクション5.1.4「サーバーシステム変数」
セクション5.1.6「サーバーステータス変数」

Slow_queries
セクション5.1.4「サーバーシステム変数」
セクション5.1.6「サーバーステータス変数」

Sort_merge_passes
セクション22.9.6.1「events_statements_current テーブル」
セクション5.1.4「サーバーシステム変数」
セクション5.1.6「サーバーステータス変数」

Sort_range
セクション22.9.6.1「events_statements_current テーブル」
セクション5.1.6「サーバーステータス変数」

Sort_rows
セクション22.9.6.1「events_statements_current テーブル」
セクション5.1.6「サーバーステータス変数」

Sort_scan
セクション22.9.6.1「events_statements_current テーブル」
セクション5.1.6「サーバーステータス変数」

Ssl_accept_renegotiates
セクション5.1.6「サーバーステータス変数」

Ssl_accepts
セクション5.1.6「サーバーステータス変数」

Ssl_callback_cache_hits
セクション5.1.6「サーバーステータス変数」

Ssl_cipher
セクション6.3.10.3「SSL 接続の使用」
セクション5.1.6「サーバーステータス変数」

Ssl_cipher_list
セクション6.3.10.4「SSL コマンドのオプション」
セクション5.1.6「サーバーステータス変数」

Ssl_client_connects
セクション5.1.6「サーバーステータス変数」

Ssl_connect_renegotiates
セクション5.1.6「サーバーステータス変数」

Ssl_ctx_verify_depth
セクション5.1.6「サーバーステータス変数」

Ssl_ctx_verify_mode
セクション5.1.6「サーバーステータス変数」

Ssl_default_timeout
セクション5.1.6「サーバーステータス変数」

Ssl_finished_accepts
セクション5.1.6「サーバーステータス変数」

Ssl_finished_connects
セクション5.1.6「サーバーステータス変数」

Ssl_server_not_after
セクション5.1.6「サーバーステータス変数」

Ssl_server_not_before
セクション5.1.6「サーバーステータス変数」

Ssl_session_cache_hits
セクション5.1.6「サーバーステータス変数」

Ssl_session_cache_misses
セクション5.1.6「サーバーステータス変数」

Ssl_session_cache_mode
セクション5.1.6「サーバーステータス変数」

Ssl_session_cache_overflows
セクション5.1.6「サーバーステータス変数」

Ssl_session_cache_size
セクション5.1.6「サーバーステータス変数」

Ssl_session_cache_timeouts
セクション5.1.6「サーバーステータス変数」

Ssl_sessions_reused
セクション5.1.6「サーバーステータス変数」

Ssl_used_session_cache_entries
セクション5.1.6「サーバーステータス変数」

Ssl_verify_depth
セクション5.1.6「サーバーステータス変数」

Ssl_verify_mode
セクション5.1.6「サーバーステータス変数」

Ssl_version
セクション5.1.6「サーバーステータス変数」

T

[索引の先頭]

Table_locks_immediate
セクション5.1.6「サーバーステータス変数」
セクション8.10.1「内部ロック方法」

Table_locks_waited
セクション5.1.6「サーバーステータス変数」
セクション8.10.1「内部ロック方法」

Table_open_cache_hits
セクション5.1.6「サーバーステータス変数」

Table_open_cache_misses
セクション5.1.6「サーバーステータス変数」

Table_open_cache_overflows
セクション5.1.6「サーバーステータス変数」

Tc_log_max_pages_used
セクション5.1.6「サーバーステータス変数」

Tc_log_page_size
セクション5.1.6「サーバーステータス変数」

Tc_log_page_waits
セクション5.1.6「サーバーステータス変数」

Threads_cached
セクション8.11.5.1「MySQL のクライアント接続のためのスレッドの使用方法」
セクション5.1.6「サーバーステータス変数」

Threads_connected
セクション5.1.6「サーバーステータス変数」

Threads_created
セクション8.11.5.1「MySQL のクライアント接続のためのスレッドの使用方法」
セクション5.1.4「サーバーシステム変数」
セクション5.1.6「サーバーステータス変数」

Threads_running
セクションA.14「MySQL 5.6 FAQ: MySQL エンタープライズスケーラビリティースレッドプール」
セクション5.1.6「サーバーステータス変数」

U

[索引の先頭]

Uptime
セクション4.5.2「mysqladmin — MySQL サーバーの管理を行うクライアント」
セクション5.1.6「サーバーステータス変数」

Uptime_since_flush_status
セクション5.1.6「サーバーステータス変数」

システム変数の索引

A | B | C | D | E | F | G | H | I | J | K | L | M | N | O | P | Q | R | S | T | U | V | W

A

[索引の先頭]

audit_log_buffer_size
セクション6.3.12.6「監査ログプラグインのオプションおよびシステム変数」
セクション6.3.12.7「監査ログプラグインのステータス変数」
セクション6.3.12.4「監査ログプラグインのロギング制御」

audit_log_connection_policy
セクション6.3.12.6「監査ログプラグインのオプションおよびシステム変数」
セクション6.3.12.4「監査ログプラグインのロギング制御」

audit_log_current_session
セクション6.3.12.6「監査ログプラグインのオプションおよびシステム変数」

audit_log_exclude_accounts
セクション6.3.12.6「監査ログプラグインのオプションおよびシステム変数」
セクション6.3.12.4「監査ログプラグインのロギング制御」

audit_log_file
セクション6.3.12「MySQL Enterprise Audit ログプラグイン」
セクション6.3.12.6「監査ログプラグインのオプションおよびシステム変数」
セクション6.3.12.2「監査ログプラグインのセキュリティーに関する考慮事項」
セクション6.3.12.4「監査ログプラグインのロギング制御」

audit_log_flush
セクション6.3.12.6「監査ログプラグインのオプションおよびシステム変数」
セクション6.3.12.4「監査ログプラグインのロギング制御」

audit_log_format
セクション6.3.12「MySQL Enterprise Audit ログプラグイン」
セクション6.3.12.3「監査ログファイル」
セクション6.3.12.6「監査ログプラグインのオプションおよびシステム変数」

audit_log_include_accounts
セクション6.3.12.6「監査ログプラグインのオプションおよびシステム変数」
セクション6.3.12.4「監査ログプラグインのロギング制御」

audit_log_policy
セクション6.3.12「MySQL Enterprise Audit ログプラグイン」
セクション6.3.12.6「監査ログプラグインのオプションおよびシステム変数」
セクション6.3.12.4「監査ログプラグインのロギング制御」

audit_log_rotate_on_size
セクション6.3.12.6「監査ログプラグインのオプションおよびシステム変数」
セクション6.3.12.4「監査ログプラグインのロギング制御」

audit_log_statement_policy
セクション6.3.12.6「監査ログプラグインのオプションおよびシステム変数」
セクション6.3.12.4「監査ログプラグインのロギング制御」

audit_log_strategy
セクション6.3.12.6「監査ログプラグインのオプションおよびシステム変数」
セクション6.3.12.4「監査ログプラグインのロギング制御」

authentication_windows_log_level
Windows 認証プラグインの使用
セクション5.1.4「サーバーシステム変数」

authentication_windows_use_principal_name
Windows 認証プラグインの使用
セクション5.1.4「サーバーシステム変数」

auto_increment_increment
セクション3.6.9「AUTO_INCREMENT の使用」
セクションA.1「MySQL 5.6 FAQ: 全般」
セクション17.4.1.34「レプリケーションと変数」
セクション17.1.4.2「レプリケーションマスターのオプションと変数」
セクション14.6.5.1「従来の InnoDB の自動インクリメントロック」
セクション5.2.4.3「混合形式のバイナリロギング形式」

auto_increment_offset
セクション3.6.9「AUTO_INCREMENT の使用」
セクションA.1「MySQL 5.6 FAQ: 全般」
セクション17.4.1.34「レプリケーションと変数」
セクション17.1.4.2「レプリケーションマスターのオプションと変数」
セクション14.6.5.1「従来の InnoDB の自動インクリメントロック」
セクション5.2.4.3「混合形式のバイナリロギング形式」

AUTOCOMMIT
セクション17.4.1.31「レプリケーションとトランザクション」

autocommit
セクション13.2.2「DELETE 構文」
セクション17.1.3.4「GTID ベースレプリケーションの制約」
セクション14.2.8「InnoDB のさまざまな SQL ステートメントで設定されたロック」
セクション14.13.14「InnoDB の読み取り専用トランザクションの最適化」
セクション14.12「InnoDB の起動オプションおよびシステム変数」
セクション14.6.7「InnoDB テーブル上の制限」

セクション14.6.4「MyISAM から InnoDB へのテーブルの変換」
セクション18.3.4.3「MySQL Cluster のシステム変数」
セクション13.3.1「START TRANSACTION、COMMIT、および ROLLBACK 構文」
セクション14.11.5「オンライン DDL の例」
セクション5.1.4「サーバーシステム変数」
セクション8.11.6.2「スレッドプール操作」
セクション13.3.5.1「テーブルロックとトランザクションの通信」
セクション14.2.10「デッドロックの検出とロールバック」
セクション1.7.2.3「トランザクションおよびアトミック操作の違い」
セクション17.4.1.31「レプリケーションとトランザクション」
セクション14.2.5「ロック読み取り (SELECT ... FOR UPDATE および SELECT ... LOCK IN SHARE MODE)」

automatic_sp_privileges

セクション13.1.5「ALTER PROCEDURE 構文」
セクション13.1.15「CREATE PROCEDURE および CREATE FUNCTION 構文」
セクション5.1.4「サーバーシステム変数」
セクション20.2.2「ストアドルーチンと MySQL 権限」

B

[索引の先頭]

back_log

セクション2.11.1.3「MySQL 5.5 から 5.6 へのアップグレード」
セクション5.1.2.1「サーバーのデフォルト値への変更」
セクション5.1.4「サーバーシステム変数」

basedir

セクション13.7.3.3「INSTALL PLUGIN 構文」
セクション5.1.4「サーバーシステム変数」

big_tables

セクション1.4「MySQL 5.6 の新機能」
セクション5.1.4「サーバーシステム変数」

bind_address

セクション5.1.4「サーバーシステム変数」

binlog_cache_size

セクション5.1.6「サーバーステータス変数」
セクション5.2.4「バイナリログ」
セクション17.1.4.4「バイナリログのオプションと変数」

binlog_checksum

セクション2.11.1.3「MySQL 5.5 から 5.6 へのアップグレード」
セクション1.4「MySQL 5.6 の新機能」
セクション17.4.2「MySQL バージョン間のレプリケーション互換性」
MySQL 用語集
セクション5.1.2.1「サーバーのデフォルト値への変更」
セクション5.2.4「バイナリログ」
セクション17.1.4.4「バイナリログのオプションと変数」

binlog_direct_non_transactional_updates

セクション17.1.4.4「バイナリログのオプションと変数」
セクション17.4.1.31「レプリケーションとトランザクション」

binlog_error_action

セクション17.1.4.4「バイナリログのオプションと変数」

binlog_format

セクション15.6「BLACKHOLE ストレージエンジン」
セクション13.2.5.2「INSERT DELAYED 構文」
セクション2.11.1.3「MySQL 5.5 から 5.6 へのアップグレード」
セクションA.13「MySQL 5.6 FAQ: レプリケーション」
セクション17.4.1.23「mysql システムデータベースのレプリケーション」
セクション5.2.4.4「mysql データベーステーブルへの変更に対するロギング形式」
セクション4.6.8「mysqlbinlog — バイナリログファイルを処理するためのユーティリティー」
セクション12.18「その他の関数」
セクション5.1.3「サーバーコマンドオプション」
セクション17.1.2.3「バイナリロギングでの安全および安全でないステートメントの判断」
セクション17.1.4.4「バイナリログのオプションと変数」
セクション5.2.4.2「バイナリログ形式の設定」
セクション14.18.6「レプリケーションでの InnoDB memcached プラグインの使用」
セクション17.4.1.2「レプリケーションと BLACKHOLE テーブル」
セクション17.4.1.21「レプリケーションと MEMORY テーブル」
セクション17.4.1.31「レプリケーションとトランザクション」
セクション17.4.1.22「レプリケーションと一時テーブル」
セクション17.4.3「レプリケーションセットアップをアップグレードする」
セクション17.1.2「レプリケーション形式」
セクション5.2.3「一般クエリーログ」
セクション12.14「情報関数」
セクション12.6.2「数学関数」
セクション12.7「日付および時間関数」
セクション5.2.4.3「混合形式のバイナリロギング形式」
セクション17.1.2.2「行ベースロギングおよびレプリケーションの使用」

binlog_gtid_recovery_simplified

セクション17.1.4.5「グローバルトランザクション ID のオプションと変数」
セクション17.1.4.4「バイナリログのオプションと変数」

binlog_max_flush_queue_time

セクション17.1.4.4「バイナリログのオプションと変数」

binlog_order_commits

セクション17.1.4.4「バイナリログのオプションと変数」

binlog_row_image

セクション1.4「MySQL 5.6 の新機能」
セクション17.4.2「MySQL バージョン間のレプリケーション互換性」
セクション17.1.4.4「バイナリログのオプションと変数」

binlog_rows_query_log_events
セクション17.4.2「MySQL バージョン間のレプリケーション互換性」
セクション17.1.4.4「バイナリログのオプションと変数」

binlog_stmt_cache_size
セクション5.1.6「サーバーステータス変数」
セクション17.1.4.4「バイナリログのオプションと変数」

binlogging_impossible_mode
セクション17.1.4.4「バイナリログのオプションと変数」

block_encryption_mode
セクション5.1.4「サーバーシステム変数」
セクション12.13「暗号化関数と圧縮関数」

bulk_insert_buffer_size
セクション8.2.2.1「INSERT ステートメントの速度」
セクション15.2.1「MyISAM 起動オプション」
セクション5.1.4「サーバーシステム変数」

C

[索引の先頭]

character_set_client
セクション23.7.9.1「C API プリペアドステートメントタイプコード」
セクション21.7「INFORMATION_SCHEMA EVENTS テーブル」
セクション21.18「INFORMATION_SCHEMA ROUTINES テーブル」
セクション21.26「INFORMATION_SCHEMA TRIGGERS テーブル」
セクション21.28「INFORMATION_SCHEMA VIEWS テーブル」
セクション13.2.6「LOAD DATA INFILE 構文」
セクションA.11「MySQL 5.6 FAQ: MySQL の中国語、日本語、および韓国語の文字セット」
セクション13.7.4「SET 構文」
セクション13.7.5.9「SHOW CREATE EVENT 構文」
セクション13.7.5.11「SHOW CREATE PROCEDURE 構文」
セクション13.7.5.13「SHOW CREATE TRIGGER 構文」
セクション13.7.5.14「SHOW CREATE VIEW 構文」
セクション13.7.5.19「SHOW EVENTS 構文」
セクション13.7.5.29「SHOW PROCEDURE STATUS 構文」
セクション13.7.5.39「SHOW TRIGGERS 構文」
セクション5.1.4「サーバーシステム変数」
セクション5.2.4「バイナリログ」
セクション17.4.1.34「レプリケーションと変数」
セクション10.1.4「接続文字セットおよび照合順序」
セクション10.5「文字セットの構成」
セクション5.2.4.3「混合形式のバイナリロギング形式」

character_set_connection
セクション10.1.9.2「CONVERT() と CAST()」
セクションA.11「MySQL 5.6 FAQ: MySQL の中国語、日本語、および韓国語の文字セット」
セクション10.7「MySQL Server のロケールサポート」
セクション13.7.4「SET 構文」
セクション5.1.4「サーバーシステム変数」
セクション17.4.1.34「レプリケーションと変数」
セクション10.1.7.5「式の照合順序」
セクション12.2「式評価での型変換」
セクション10.1.4「接続文字セットおよび照合順序」
セクション10.1.8「文字列のレパートリー」
セクション9.1.1「文字列リテラル」
セクション10.1.3.5「文字列リテラルの文字セットおよび照合順序」
セクション12.7「日付および時間関数」
セクション12.13「暗号化関数と圧縮関数」
セクション5.2.4.3「混合形式のバイナリロギング形式」
セクション10.1.9.1「結果文字列」

character_set_database
セクション13.1.15「CREATE PROCEDURE および CREATE FUNCTION 構文」
セクション13.2.6「LOAD DATA INFILE 構文」
セクション13.7.4「SET 構文」
セクション5.1.4「サーバーシステム変数」
セクション10.1.3.2「データベース文字セットおよび照合順序」
セクション17.4.1.34「レプリケーションと変数」
セクション10.1.4「接続文字セットおよび照合順序」
セクション5.2.4.3「混合形式のバイナリロギング形式」

character_set_filesystem
セクション13.2.6「LOAD DATA INFILE 構文」
セクション13.2.9.1「SELECT ... INTO 構文」
セクション5.1.3「サーバーコマンドオプション」
セクション5.1.4「サーバーシステム変数」
セクション12.5「文字列関数」

character_set_results
セクション23.7.5「C API データ構造」
セクションA.11「MySQL 5.6 FAQ: MySQL の中国語、日本語、および韓国語の文字セット」
セクション13.7.4「SET 構文」
セクション10.1.6「エラーメッセージの文字セット」
セクション5.1.4「サーバーシステム変数」
セクション10.1.12「メタデータ用の UTF-8」
セクション10.1.4「接続文字セットおよび照合順序」

character_set_server
セクション5.1.4「サーバーシステム変数」
セクション10.1.3.1「サーバー文字セットおよび照合順序」
セクション10.1.3.2「データベース文字セットおよび照合順序」
セクション17.4.1.34「レプリケーションと変数」
セクション17.4.1.3「レプリケーションと文字セット」
セクション12.9.4「全文ストップワード」
セクション10.1.4「接続文字セットおよび照合順序」
セクション10.5「文字セットの構成」
セクション5.2.4.3「混合形式のバイナリロギング形式」

character_set_system
セクション5.1.4「サーバーシステム変数」
セクション10.1.12「メタデータ用の UTF-8」
セクション10.5「文字セットの構成」

character_sets_dir
セクション10.4.3「8 ビットの文字セットへの単純な照合順序の追加」

セクション10.4.4.1「LDML 構文を使用した UCA 照合順序の定義」
セクション5.1.4「サーバーシステム変数」

collation_connection

セクション10.1.9.2「CONVERT() と CAST()」
セクション21.7「INFORMATION_SCHEMA EVENTS テーブル」
セクション21.18「INFORMATION_SCHEMA ROUTINES テーブル」
セクション21.26「INFORMATION_SCHEMA TRIGGERS テーブル」
セクション21.28「INFORMATION_SCHEMA VIEWS テーブル」
セクション13.7.4「SET 構文」
セクション13.7.5.9「SHOW CREATE EVENT 構文」
セクション13.7.5.11「SHOW CREATE PROCEDURE 構文」
セクション13.7.5.13「SHOW CREATE TRIGGER 構文」
セクション13.7.5.14「SHOW CREATE VIEW 構文」
セクション13.7.5.19「SHOW EVENTS 構文」
セクション13.7.5.29「SHOW PROCEDURE STATUS 構文」
セクション13.7.5.39「SHOW TRIGGERS 構文」
セクション5.1.4「サーバーシステム変数」
セクション5.2.4「バイナリログ」
セクション17.4.1.34「レプリケーションと変数」
セクション10.1.7.5「式の照合順序」
セクション12.2「式評価での型変換」
セクション10.1.4「接続文字セットおよび照合順序」
セクション10.1.3.5「文字列リテラルの文字セットおよび照合順序」
セクション12.7「日付および時間関数」
セクション12.13「暗号化関数と圧縮関数」
セクション5.2.4.3「混合形式のバイナリロギング形式」
セクション10.1.9.1「結果文字列」

collation_database

セクション13.1.15「CREATE PROCEDURE および CREATE FUNCTION 構文」
セクション5.1.4「サーバーシステム変数」
セクション10.1.3.2「データベース文字セットおよび照合順序」
セクション5.2.4「バイナリログ」
セクション17.4.1.34「レプリケーションと変数」
セクション10.1.4「接続文字セットおよび照合順序」
セクション5.2.4.3「混合形式のバイナリロギング形式」

collation_server

セクション5.1.4「サーバーシステム変数」
セクション10.1.3.1「サーバー文字セットおよび照合順序」
セクション10.1.3.2「データベース文字セットおよび照合順序」
セクション5.2.4「バイナリログ」
セクション17.4.1.34「レプリケーションと変数」
セクション17.4.1.3「レプリケーションと文字セット」
セクション12.9.4「全文ストップワード」
セクション10.1.4「接続文字セットおよび照合順序」
セクション5.2.4.3「混合形式のバイナリロギング形式」

completion_type

セクション23.7.7.6「mysql_commit()」
セクション23.7.7.58「mysql_rollback()」
セクション13.3.1「START TRANSACTION、COMMIT、および ROLLBACK 構文」
セクション5.1.4「サーバーシステム変数」

concurrent_insert

セクション8.6.1「MyISAM クエリーの最適化」
セクション8.2.5「その他の最適化のヒント」
セクション5.1.4「サーバーシステム変数」
セクション8.10.1「内部ロック方法」
セクション8.10.3「同時挿入」

connect_timeout

セクションB.5.2.3「MySQL サーバーへの接続が失われました」
セクション23.7.7.53「mysql_real_connect()」
セクション5.1.4「サーバーシステム変数」
セクションB.5.2.11「通信エラーおよび中止された接続」

core_file

セクション5.1.4「サーバーシステム変数」

D

[索引の先頭]

daemon_memcached_engine_lib_name

セクション14.18.3.2「InnoDB memcached プラグインのインストールおよび構成」

daemon_memcached_engine_lib_path

セクション14.18.3.2「InnoDB memcached プラグインのインストールおよび構成」

daemon_memcached_option

セクション14.18.3.2「InnoDB memcached プラグインのインストールおよび構成」
セクション14.18.8「InnoDB memcached プラグインのトラブルシューティング」
セクション14.18.2「InnoDB および memcached の統合のアーキテクチャー」
セクション14.18.4.1「SASL による memcached インタフェースのパスワード保護」

daemon_memcached_r_batch_size

セクション14.18.3.2「InnoDB memcached プラグインのインストールおよび構成」
セクション14.18.5.3「InnoDB memcached プラグインのパフォーマンスのチューニング」
セクション14.18.2「InnoDB および memcached の統合のアーキテクチャー」
セクション14.12「InnoDB の起動オプションおよびシステム変数」
セクション14.18.5.6「ベースとなる InnoDB テーブルでの DML および DDL ステートメントの実行」
セクション14.18.6「レプリケーションでの InnoDB memcached プラグインの使用」

daemon_memcached_w_batch_size

セクション14.18.3.2「InnoDB memcached プラグインのインストールおよび構成」

セクション14.18.5.4「InnoDB memcached プラグインのトランザクション動作の制御」
セクション14.18.5.3「InnoDB memcached プラグインのパフォーマンスのチューニング」
セクション14.18.2「InnoDB および memcached の統合のアーキテクチャー」
セクション14.12「InnoDB の起動オプションおよびシステム変数」
セクション14.18.5.6「ベースとなる InnoDB テーブルでの DML および DDL ステートメントの実行」
セクション14.18.6「レプリケーションでの InnoDB memcached プラグインの使用」

DATADIR
セクション14.5.4「テーブルスペースの位置の指定」

datadir
セクション21.29.7「INFORMATION_SCHEMA INNODB_SYS_TABLES テーブル」
セクション21.29.15「INFORMATION_SCHEMA INNODB_SYS_TABLESPACES テーブル」
セクション2.3「Microsoft Windows に MySQL をインストールする」
MySQL 用語集
セクション5.1.4「サーバーシステム変数」

date_format
セクション1.4「MySQL 5.6 の新機能」
セクション5.1.4「サーバーシステム変数」

datetime_format
セクション1.4「MySQL 5.6 の新機能」
セクション5.1.4「サーバーシステム変数」

debug
セクション24.4.3「DBUG パッケージ」
セクション5.1.4「サーバーシステム変数」

debug_sync
セクション5.1.4「サーバーシステム変数」

default_storage_engine
セクション13.1.14「CREATE LOGFILE GROUP 構文」
セクション5.1.4「サーバーシステム変数」
セクション15.1「ストレージエンジンの設定」
セクション5.1.8.1「プラグインのインストールおよびアンインストール」
セクション17.4.1.34「レプリケーションと変数」
セクション17.3.2「異なるマスターおよびスレーブストレージエンジンでレプリケーションを使用する」

default_tmp_storage_engine
セクション5.1.3「サーバーコマンドオプション」
セクション5.1.4「サーバーシステム変数」
セクション15.1「ストレージエンジンの設定」
セクション5.1.8.1「プラグインのインストールおよびアンインストール」

default_week_format
セクション5.1.4「サーバーシステム変数」
セクション12.7「日付および時間関数」
セクション19.6.3「関数に関連するパーティショニング制限」

delay_key_write
セクション13.1.17「CREATE TABLE 構文」
セクション5.1.4「サーバーシステム変数」

delayed_insert_limit
セクション13.2.5.2「INSERT DELAYED 構文」
セクション5.1.4「サーバーシステム変数」

delayed_insert_timeout
セクション13.2.5.2「INSERT DELAYED 構文」
セクション5.1.4「サーバーシステム変数」

delayed_queue_size
セクション13.2.5.2「INSERT DELAYED 構文」
セクション5.1.4「サーバーシステム変数」

disable_gtid_unsafe_statements
セクション17.1.4.5「グローバルトランザクション ID のオプションと変数」

disconnect_on_expired_password
セクション5.1.4「サーバーシステム変数」
セクション6.3.6「パスワードの期限切れとサンドボックスモード」

div_precision_increment
セクション5.1.4「サーバーシステム変数」
セクション12.6.1「算術演算子」

E
[索引の先頭]

end_markers_in_json
セクション5.1.4「サーバーシステム変数」

enforce_gtid_consistency
セクション17.1.4.5「グローバルトランザクション ID のオプションと変数」

engine_condition_pushdown
セクション5.1.3「サーバーコマンドオプション」
セクション5.1.4「サーバーシステム変数」

eq_range_index_dive_limit
セクション5.1.4「サーバーシステム変数」
複数値比較の等価範囲の最適化

error_count
セクション13.7.5.18「SHOW ERRORS 構文」
セクションB.1「エラー情報のソース」
セクション5.1.4「サーバーシステム変数」
診断領域関連のシステム変数

event_scheduler
セクション20.4.6「イベントスケジューラと MySQL 権限」

セクション20.4.2「イベントスケジューラの構成」
セクション5.1.4「サーバーシステム変数」
セクション23.6.2「組み込み MySQL サーバーを使用する場合の制限」

expire_logs_days

セクション13.4.1.1「PURGE BINARY LOGS 構文」
セクション5.1.4「サーバーシステム変数」
セクション5.2.7「サーバーログの保守」

explicit_defaults_for_timestamp

セクション2.11.1.3「MySQL 5.5 から 5.6 へのアップグレード」
セクション1.4「MySQL 5.6 の新機能」
セクション11.3.5「TIMESTAMP および DATETIME の自動初期化および更新機能」
セクション5.1.4「サーバーシステム変数」
セクション11.6「データ型デフォルト値」
セクション11.1.2「日付と時間型の概要」

external_user

セクション5.1.4「サーバーシステム変数」
サーバー側認証プラグインの作成
セクション6.3.9「プロキシユーザー」
認証プラグインでのプロキシユーザーサポートの実装

F

[索引の先頭]

flush

セクション5.1.4「サーバーシステム変数」

flush_time

セクション2.11.1.3「MySQL 5.5 から 5.6 へのアップグレード」
セクション5.1.2.1「サーバーのデフォルト値への変更」
セクション5.1.4「サーバーシステム変数」

foreign_key_checks

セクション18.3.4.3「MySQL Cluster のシステム変数」
セクション14.11.6「オンライン DDL の実装の詳細」
セクション14.11.1「オンライン DDL の概要」
セクション5.1.7「サーバー SQL モード」
セクション5.1.4「サーバーシステム変数」
セクション14.5.5「テーブルスペースの別のサーバーへのコピー (トランスポータブルテーブルスペース)」
セクション5.2.4「バイナリログ」
セクション17.4.1.34「レプリケーションと変数」
セクション13.1.17.2「外部キー制約の使用」
セクション5.2.4.3「混合形式のバイナリロギング形式」

ft_boolean_syntax

セクション12.9.6「MySQL の全文検索の微調整」
セクション5.1.4「サーバーシステム変数」
セクション12.9.2「ブール全文検索」

ft_max_word_len

セクション12.9.6「MySQL の全文検索の微調整」
セクション5.1.4「サーバーシステム変数」
セクション12.9.2「ブール全文検索」
セクション17.1.1.6「ローデータファイルを使用したデータスナップショットの作成」
セクション24.2.3.2「全文パーサープラグイン」

ft_min_word_len

セクション12.9.6「MySQL の全文検索の微調整」
セクション5.1.4「サーバーシステム変数」
セクション12.9.2「ブール全文検索」
セクション17.1.1.6「ローデータファイルを使用したデータスナップショットの作成」
セクション24.2.3.2「全文パーサープラグイン」
セクション12.9.1「自然言語全文検索」

ft_query_expansion_limit

セクション5.1.4「サーバーシステム変数」

ft_stopword_file

セクション12.9.6「MySQL の全文検索の微調整」
セクション5.1.4「サーバーシステム変数」
セクション12.9.2「ブール全文検索」
セクション17.1.1.6「ローデータファイルを使用したデータスナップショットの作成」
セクション12.9.4「全文ストップワード」
セクション12.9「全文検索関数」
セクション12.9.1「自然言語全文検索」

G

[索引の先頭]

general_log

MySQL 用語集
セクション5.1.3「サーバーコマンドオプション」
セクション5.1.4「サーバーシステム変数」
セクション5.2.3「一般クエリーログ」
セクション5.2.1「一般クエリーログおよびスロークエリーログの出力先の選択」

general_log_file

セクション5.1.4「サーバーシステム変数」
セクション5.2.3「一般クエリーログ」
セクション5.2.1「一般クエリーログおよびスロークエリーログの出力先の選択」

group_concat_max_len

セクション12.19.1「GROUP BY (集約) 関数」
セクション5.1.4「サーバーシステム変数」

gtid_done

セクション13.4.1.2「RESET MASTER 構文」
セクション13.7.5.24「SHOW MASTER STATUS 構文」
セクション13.7.5.35「SHOW SLAVE STATUS 構文」
セクション17.1.4.5「グローバルトランザクション ID のオプションと変数」

gtid_executed

セクション13.4.2.1「CHANGE MASTER TO 構文」
セクション17.1.3.1「GTID の概念」
セクション17.1.3.4「GTID ベースレプリケーションの制約」
セクション13.4.1.2「RESET MASTER 構文」
セクション13.7.5.24「SHOW MASTER STATUS 構文」
セクション13.7.5.35「SHOW SLAVE STATUS 構文」

セクション17.1.4.5「グローバルトランザクション ID のオプションと変数」
セクション17.1.4.4「バイナリログのオプションと変数」
セクション17.1.3.3「フェイルオーバーおよびスケールアウトでの GTID の使用」
セクション17.1.4.3「レプリケーションスレーブのオプションと変数」

gtid_lost

セクション13.4.1.2「RESET MASTER 構文」
セクション17.1.4.5「グローバルトランザクション ID のオプションと変数」

gtid_mode

セクション13.4.2.1「CHANGE MASTER TO 構文」
セクション17.4.2「MySQL バージョン間のレプリケーション互換性」
セクション12.16「グローバルトランザクション ID とともに使用される関数」
セクション17.1.4.5「グローバルトランザクション ID のオプションと変数」
セクション17.1.3.3「フェイルオーバーおよびスケールアウトでの GTID の使用」

gtid_next

セクション13.1.6「ALTER SERVER 構文」
セクション13.7.2.1「ANALYZE TABLE 構文」
セクション13.7.6.2「CACHE INDEX 構文」
セクション13.4.2.1「CHANGE MASTER TO 構文」
セクション13.7.2.2「CHECK TABLE 構文」
セクション13.1.16「CREATE SERVER 構文」
セクション13.1.27「DROP SERVER 構文」
セクション13.7.6.3「FLUSH 構文」
セクション17.1.3.1「GTID の概念」
セクション13.7.6.5「LOAD INDEX INTO CACHE 構文」
セクション13.7.2.4「OPTIMIZE TABLE 構文」
セクション13.7.2.5「REPAIR TABLE 構文」
セクション13.7.6.6「RESET 構文」
セクション13.4.2.5「START SLAVE 構文」
セクション13.4.2.6「STOP SLAVE 構文」
セクション17.1.4.5「グローバルトランザクション ID のオプションと変数」

gtid_owned

セクション17.1.4.5「グローバルトランザクション ID のオプションと変数」

gtid_purged

セクション17.1.3.1「GTID の概念」
セクション13.4.1.2「RESET MASTER 構文」
セクション17.1.4.5「グローバルトランザクション ID のオプションと変数」
セクション17.1.4.4「バイナリログのオプションと変数」
セクション17.1.3.3「フェイルオーバーおよびスケールアウトでの GTID の使用」

H

[索引の先頭]

have_compress

セクション5.1.4「サーバーシステム変数」

have_crypt

セクション5.1.4「サーバーシステム変数」

have_csv

セクション5.1.4「サーバーシステム変数」

have_dynamic_loading

セクション5.1.4「サーバーシステム変数」
セクション17.3.8.2「準同期レプリケーションのインストールと構成」

have_geometry

セクション5.1.4「サーバーシステム変数」

have_innodb

セクション5.1.4「サーバーシステム変数」

have_ndbcluster

セクション18.3.4.3「MySQL Cluster のシステム変数」

have_openssl

セクション5.1.4「サーバーシステム変数」

have_partitioning

セクション5.1.4「サーバーシステム変数」
第19章「パーティション化」

have_profiling

セクション1.4「MySQL 5.6 の新機能」
セクション5.1.4「サーバーシステム変数」

have_query_cache

セクション8.9.3.3「クエリーキャッシュの構成」
セクション5.1.4「サーバーシステム変数」

have_rtree_keys

セクション5.1.4「サーバーシステム変数」

have_ssl

セクション6.3.10.2「SSL を使用するための MySQL の構成」
セクション5.1.4「サーバーシステム変数」

have_symlink

Unix 上の MyISAM へのシンボリックリンクの使用
Windows 上のデータベースへのシンボリックリンクの使用
セクション5.1.4「サーバーシステム変数」

host_cache_size

セクション8.11.5.2「DNS ルックアップの最適化とホストキャッシュ」
セクション1.4「MySQL 5.6 の新機能」
セクション5.1.2.1「サーバーのデフォルト値への変更」
セクション5.1.3「サーバーコマンドオプション」
セクション5.1.4「サーバーシステム変数」

hostname

セクション5.1.4「サーバーシステム変数」

I

[索引の先頭]

identity
セクション5.1.4「サーバーシステム変数」
セクション17.4.1.34「レプリケーションと変数」
セクション5.2.4.3「混合形式のバイナリロギング形式」

ignore_db_dirs
セクション5.1.3「サーバーコマンドオプション」
セクション5.1.4「サーバーシステム変数」

init_connect
セクション22.9.10.1「host_cache テーブル」
セクション10.1.5「アプリケーションの文字セットおよび照合順序の構成」
セクション5.1.4「サーバーシステム変数」
セクション17.1.4.3「レプリケーションスレーブのオプションと変数」

init_file
セクション5.1.4「サーバーシステム変数」

init_slave
セクション17.1.4.3「レプリケーションスレーブのオプションと変数」

innodb
セクション1.4「MySQL 5.6 の新機能」
セクション14.3.1「読み取り専用操作用の InnoDB の構成」

innodb_adaptive_flushing
セクション8.5.7「InnoDB ディスク I/O の最適化」
セクション14.13.1.6「InnoDB バッファープールのフラッシュのチューニング」
セクション14.13.1.2「InnoDB バッファープールのフラッシュの頻度の構成」

innodb_adaptive_flushing_lwm
セクション14.13.1.6「InnoDB バッファープールのフラッシュのチューニング」

innodb_adaptive_hash_index
セクション14.13.5「InnoDB のスレッド並列性の構成」
セクション14.12「InnoDB の起動オプションおよびシステム変数」
セクション8.5.8「InnoDB 構成変数の最適化」
MySQL 用語集
セクション14.2.13.6「適応型ハッシュインデックス」

innodb_adaptive_max_sleep_delay
セクション14.13.5「InnoDB のスレッド並列性の構成」
セクション14.12「InnoDB の起動オプションおよびシステム変数」

innodb_additional_mem_pool_size
セクション14.13.3「InnoDB のためのメモリーアロケータの構成」
セクション1.4「MySQL 5.6 の新機能」

innodb_api_bk_commit_interval
セクション14.18.5.4「InnoDB memcached プラグインのトランザクション動作の制御」
セクション14.18.2「InnoDB および memcached の統合のアーキテクチャー」

innodb_api_disable_rowlock
セクション14.18.5.4「InnoDB memcached プラグインのトランザクション動作の制御」

innodb_api_enable_binlog
セクション14.18.6「レプリケーションでの InnoDB memcached プラグインの使用」

innodb_api_enable_mdl
セクション14.18.5.4「InnoDB memcached プラグインのトランザクション動作の制御」
セクション14.18.2「InnoDB および memcached の統合のアーキテクチャー」

innodb_api_trx_level
セクション14.18.5.4「InnoDB memcached プラグインのトランザクション動作の制御」
セクション14.18.2「InnoDB および memcached の統合のアーキテクチャー」

innodb_autoextend_increment
セクション14.5.2「InnoDB File-Per-Table モード」
セクション14.3「InnoDB の構成」
セクション14.12「InnoDB の起動オプションおよびシステム変数」
セクション14.5.7「InnoDB ログファイルの数またはサイズの変更、および InnoDB テーブルスペースのサイズの変更」
セクション2.11.1.3「MySQL 5.5 から 5.6 へのアップグレード」
MySQL 用語集
セクション5.1.2.1「サーバーのデフォルト値への変更」

innodb_autoinc_lock_mode
セクション8.5.4「InnoDB テーブルの一括データロード」
MySQL 用語集
セクション12.14「情報関数」
セクション14.6.5.2「構成可能な InnoDB の自動インクリメントロック」

innodb_buffer_pool_dump_at_shutdown
セクション14.12「InnoDB の起動オプションおよびシステム変数」
MySQL 用語集

innodb_buffer_pool_dump_now
セクション14.12「InnoDB の起動オプションおよびシステム変数」

innodb_buffer_pool_filename
セクション14.13.1.5「再起動を高速化するための InnoDB バッファープールのプリロード」

innodb_buffer_pool_instances
セクション14.18.5.3「InnoDB memcached プラグインのパフォーマンスのチューニング」

セクション14.12「InnoDB の起動オプションおよびシステム変数」
セクション8.9.1「InnoDB バッファープール」
セクション14.6.4「MyISAM から InnoDB へのテーブルの変換」
セクション2.11.1.3「MySQL 5.5 から 5.6 へのアップグレード」
MySQL 用語集
セクション5.1.2.1「サーバーのデフォルト値への変更」
セクション14.13.1.4「複数のバッファープールインスタンスの使用」

innodb_buffer_pool_load_abort

セクション5.1.6「サーバーステータス変数」

innodb_buffer_pool_load_at_startup

セクション14.12「InnoDB の起動オプションおよびシステム変数」
MySQL 用語集
セクション5.1.6「サーバーステータス変数」

innodb_buffer_pool_load_now

セクション14.12「InnoDB の起動オプションおよびシステム変数」
セクション5.1.6「サーバーステータス変数」

innodb_buffer_pool_size

セクション14.18.5.3「InnoDB memcached プラグインのパフォーマンスのチューニング」
セクション14.18.2「InnoDB および memcached の統合のアーキテクチャー」
セクション14.19.5「InnoDB のエラーコード」
セクション14.12「InnoDB の起動オプションおよびシステム変数」
セクション8.5.7「InnoDB ディスク I/O の最適化」
セクション8.9.1「InnoDB バッファープール」
セクション14.6.4「MyISAM から InnoDB へのテーブルの変換」
MySQL 用語集
セクション14.7.6「OLTP ワークロードの圧縮」
セクション5.1.6「サーバーステータス変数」
セクション14.13.1.4「複数のバッファープールインスタンスの使用」

innodb_change_buffer_max_size

セクション8.5.7「InnoDB ディスク I/O の最適化」
MySQL 用語集

innodb_change_buffering

セクション8.5.7「InnoDB ディスク I/O の最適化」
セクション8.5.2「InnoDB トランザクション管理の最適化」
セクション14.13.4「InnoDB 変更バッファリングの構成」
MySQL 用語集
セクション14.3.1「読み取り専用操作用の InnoDB の構成」

innodb_checksum_algorithm

セクション14.12「InnoDB の起動オプションおよびシステム変数」
セクション2.11.1.3「MySQL 5.5 から 5.6 へのアップグレード」
セクション1.4「MySQL 5.6 の新機能」
セクション5.1.2.1「サーバーのデフォルト値への変更」
セクション14.13.15「チェックサムの高速化のための CRC32 チェックサムアルゴリズムの使用」

innodb_checksums

セクション14.12「InnoDB の起動オプションおよびシステム変数」
MySQL 用語集
セクション14.13.15「チェックサムの高速化のための CRC32 チェックサムアルゴリズムの使用」

innodb_cmp_per_index_enabled

セクション21.29.2「INFORMATION_SCHEMA INNODB_CMP_PER_INDEX および INNODB_CMP_PER_INDEX_RESET テーブル」
セクション14.7.3「InnoDB テーブルの圧縮の調整」
セクション14.7.4「実行時の圧縮のモニタリング」

innodb_compression_failure_threshold_pct

セクション14.12「InnoDB の起動オプションおよびシステム変数」
セクション14.7.5「InnoDB テーブルでの圧縮の動作」
セクション14.7.3「InnoDB テーブルの圧縮の調整」
セクション1.4「MySQL 5.6 の新機能」
MySQL 用語集
セクション14.7.6「OLTP ワークロードの圧縮」

innodb_compression_level

セクション14.7.5「InnoDB テーブルでの圧縮の動作」
セクション14.7.3「InnoDB テーブルの圧縮の調整」
セクション1.4「MySQL 5.6 の新機能」
MySQL 用語集
セクション14.7.6「OLTP ワークロードの圧縮」

innodb_compression_pad_pct_max

セクション14.7.5「InnoDB テーブルでの圧縮の動作」
セクション14.7.3「InnoDB テーブルの圧縮の調整」
セクション1.4「MySQL 5.6 の新機能」
MySQL 用語集
セクション14.7.6「OLTP ワークロードの圧縮」

innodb_concurrency_tickets

セクション14.13.5「InnoDB のスレッド並列性の構成」
セクション14.12「InnoDB の起動オプションおよびシステム変数」
セクション8.5.8「InnoDB 構成変数の最適化」
セクション2.11.1.3「MySQL 5.5 から 5.6 へのアップグレード」
セクション5.1.2.1「サーバーのデフォルト値への変更」

innodb_data_file_path

セクション14.19.1「InnoDB の I/O に関する問題のトラブルシューティング」
セクション14.3「InnoDB の構成」
セクション14.12「InnoDB の起動オプションおよびシステム変数」
セクション14.5.7「InnoDB ログファイルの数またはサイズの変更、および InnoDB テーブルスペースのサイズの変更」
セクション4.4.3「mysql_install_db — MySQL データディレクトリの初期化」
セクション5.1.2.1「サーバーのデフォルト値への変更」
セクション14.5.8「共有テーブルスペースでの RAW ディスクパーティションの使用」

innodb_data_home_dir

セクション14.19.1「InnoDB の I/O に関する問題のトラブルシューティング」
セクション14.3「InnoDB の構成」
セクション14.12「InnoDB の起動オプションおよびシステム変数」
セクション4.4.3「mysql_install_db — MySQL データディレクトリの初期化」

innodb_doublewrite

セクション14.18.5.3「InnoDB memcached プラグインのパフォーマンスのチューニング」
セクション14.3「InnoDB の構成」
セクション14.10.1「InnoDB ディスク I/O」
セクション14.2.1「MySQL および ACID モデル」
MySQL 用語集

innodb_fast_shutdown

セクション14.8.2.1「InnoDB が起動されたときの互換性チェック」
セクション14.16.1「InnoDB のリカバリプロセス」
セクション14.5.7「InnoDB ログファイルの数またはサイズの変更、および InnoDB テーブルスペースのサイズの変更」
セクション2.11.1「MySQL のアップグレード」
MySQL 用語集
セクション5.1.12「シャットダウンプロセス」

innodb_file_format

セクション13.1.17「CREATE TABLE 構文」
セクション14.9.3「DYNAMIC および COMPRESSED 行フォーマット」
セクション14.8.2.1「InnoDB が起動されたときの互換性チェック」
セクション14.12「InnoDB の起動オプションおよびシステム変数」
セクション14.9.1「InnoDB 行ストレージの概要」
セクション14.6.4「MyISAM から InnoDB へのテーブルの変換」
MySQL 用語集
セクション14.7.7「SQL 圧縮構文の警告とエラー」
セクション14.8.2.2「テーブルが開かれたときの互換性チェック」
セクション14.7.2「テーブル圧縮の有効化」
セクション14.8.1「ファイル形式の有効化」
セクション10.1.11「以前の Unicode サポートから現在の Unicode サポートへのアップグレード」
セクション14.8.3「使用されているファイル形式の識別」

innodb_file_format_check

セクション14.8.2.1「InnoDB が起動されたときの互換性チェック」
セクション14.8.2.2「テーブルが開かれたときの互換性チェック」

innodb_file_format_max

セクション14.12「InnoDB の起動オプションおよびシステム変数」

innodb_file_per_table

セクション13.1.17「CREATE TABLE 構文」
セクション14.5.3「File-Per-Table モードの有効化および無効化」
セクション13.7.6.3「FLUSH 構文」
セクション14.2.13.3「FULLTEXT インデックス」
セクション21.29.15「INFORMATION_SCHEMA INNODB_SYS_TABLESPACES テーブル」
セクション14.5.2「InnoDB File-Per-Table モード」
セクション14.17「InnoDB と MySQL レプリケーション」
セクション14.12「InnoDB の起動オプションおよびシステム変数」
セクション14.6.1「InnoDB テーブルの作成」
セクション14.5.1「InnoDB テーブルスペースの作成」
セクション14.15.4「InnoDB テーブルスペースモニターの出力」
セクション14.19.3「InnoDB データディクショナリの操作のトラブルシューティング」
セクション14.1「InnoDB 入門」
セクション14.9.1「InnoDB 行ストレージの概要」
セクション14.6.4「MyISAM から InnoDB へのテーブルの変換」
セクション2.11.1.3「MySQL 5.5 から 5.6 へのアップグレード」
セクション14.2.1「MySQL および ACID モデル」
MySQL 用語集
セクション13.7.2.4「OPTIMIZE TABLE 構文」
セクション14.7.7「SQL 圧縮構文の警告とエラー」
セクション14.10.5「TRUNCATE TABLE によるディスク領域の再利用」
セクション14.11.6「オンライン DDL の実装の詳細」
セクション5.1.2.1「サーバーのデフォルト値への変更」
セクション14.5.5「テーブルスペースの別のサーバーへのコピー (トランスポータブルテーブルスペース)」
セクション14.7.2「テーブル圧縮の有効化」
セクション19.6「パーティショニングの制約と制限」
セクション14.8.1「ファイル形式の有効化」
セクション14.10.2「ファイル領域管理」
セクション10.1.11「以前の Unicode サポートから現在の Unicode サポートへのアップグレード」
セクション14.6.2「別のマシンへの InnoDB テーブルの移動またはコピー」
セクション14.2.13.7「物理的な行構造」
セクション17.3.4「異なるデータベースを異なるスレーブに複製する」

innodb_flush_log_at_timeout

セクション14.12「InnoDB の起動オプションおよびシステム変数」

innodb_flush_log_at_trx_commit

セクション14.18.5.3「InnoDB memcached プラグインのパフォーマンスのチューニング」
セクション8.5.2「InnoDB トランザクション管理の最適化」
セクション14.2.1「MySQL および ACID モデル」

innodb_flush_method

セクション14.5.2「InnoDB File-Per-Table モード」
セクション14.18.5.3「InnoDB memcached プラグインのパフォーマンスのチューニング」
セクション8.5.7「InnoDB ディスク I/O の最適化」
セクション5.1.6「サーバーステータス変数」

innodb_flush_neighbors

セクション8.5.7「InnoDB ディスク I/O の最適化」
セクション14.13.1.6「InnoDB バッファープールのフラッシュのチューニング」

innodb_flushing_avg_loops
セクション14.13.1.6「InnoDB バッファープールのフラッシュのチューニング」

innodb_force_load_corrupted
セクション14.12「InnoDB の起動オプションおよびシステム変数」

innodb_force_recovery
セクション14.19.2「InnoDB のリカバリの強制的な実行」
セクション14.16.1「InnoDB のリカバリプロセス」
セクション8.5.2「InnoDB トランザクション管理の最適化」
セクション2.11.4「テーブルまたはインデックスの再作成または修復」
セクション1.6「質問またはバグをレポートする方法」

innodb_ft_aux_table
セクション21.29.25「INFORMATION_SCHEMA INNODB_FT_BEING_DELETED テーブル」
セクション21.29.20「INFORMATION_SCHEMA INNODB_FT_CONFIG テーブル」
セクション21.29.24「INFORMATION_SCHEMA INNODB_FT_DELETED テーブル」
セクション21.29.23「INFORMATION_SCHEMA INNODB_FT_INDEX_CACHE テーブル」
セクション21.29.22「INFORMATION_SCHEMA INNODB_FT_INDEX_TABLE テーブル」
セクション14.14.4「InnoDB INFORMATION_SCHEMA FULLTEXT インデックステーブル」
セクション14.12「InnoDB の起動オプションおよびシステム変数」

innodb_ft_cache_size
セクション14.2.13.3「FULLTEXT インデックス」
セクション21.29.23「INFORMATION_SCHEMA INNODB_FT_INDEX_CACHE テーブル」
セクション14.14.4「InnoDB INFORMATION_SCHEMA FULLTEXT インデックステーブル」
セクション14.12「InnoDB の起動オプションおよびシステム変数」
MySQL 用語集

innodb_ft_enable_diag_print
セクション14.12「InnoDB の起動オプションおよびシステム変数」

innodb_ft_enable_stopword
セクション14.12「InnoDB の起動オプションおよびシステム変数」
セクション12.9.6「MySQL の全文検索の微調整」
セクション12.9.2「ブール全文検索」
セクション12.9「全文検索関数」
セクション12.9.1「自然言語全文検索」

innodb_ft_max_token_size
セクション14.12「InnoDB の起動オプションおよびシステム変数」
セクション12.9.6「MySQL の全文検索の微調整」
セクション12.9.2「ブール全文検索」
セクション12.9.4「全文ストップワード」

innodb_ft_min_token_size
セクション14.12「InnoDB の起動オプションおよびシステム変数」
セクション12.9.6「MySQL の全文検索の微調整」
セクション12.9.2「ブール全文検索」
セクション12.9.4「全文ストップワード」
セクション12.9.1「自然言語全文検索」

innodb_ft_num_word_optimize
セクション14.12「InnoDB の起動オプションおよびシステム変数」
セクション12.9.6「MySQL の全文検索の微調整」
セクション13.7.2.4「OPTIMIZE TABLE 構文」

innodb_ft_result_cache_limit
セクション14.12「InnoDB の起動オプションおよびシステム変数」

innodb_ft_server_stopword_table
セクション21.29.21「INFORMATION_SCHEMA INNODB_FT_DEFAULT_STOPWORD テーブル」
セクション14.12「InnoDB の起動オプションおよびシステム変数」
セクション12.9.6「MySQL の全文検索の微調整」
セクション12.9.2「ブール全文検索」
セクション12.9.4「全文ストップワード」
セクション12.9「全文検索関数」
セクション12.9.1「自然言語全文検索」

innodb_ft_sort_pll_degree
セクション14.2.13.3「FULLTEXT インデックス」
セクション14.12「InnoDB の起動オプションおよびシステム変数」

innodb_ft_total_cache_size
セクション14.2.13.3「FULLTEXT インデックス」
セクション21.29.23「INFORMATION_SCHEMA INNODB_FT_INDEX_CACHE テーブル」
セクション14.14.4「InnoDB INFORMATION_SCHEMA FULLTEXT インデックステーブル」
セクション14.12「InnoDB の起動オプションおよびシステム変数」

innodb_ft_user_stopword_table
セクション21.29.21「INFORMATION_SCHEMA INNODB_FT_DEFAULT_STOPWORD テーブル」
セクション14.12「InnoDB の起動オプションおよびシステム変数」
セクション12.9.6「MySQL の全文検索の微調整」
セクション12.9.2「ブール全文検索」
セクション12.9.4「全文ストップワード」
セクション12.9「全文検索関数」
セクション12.9.1「自然言語全文検索」

innodb_io_capacity
セクション14.12「InnoDB の起動オプションおよびシステム変数」
セクション8.5.7「InnoDB ディスク I/O の最適化」
セクション14.13.1.6「InnoDB バッファープールのフラッシュのチューニング」

セクション14.13.8「InnoDB マスタースレッドの I/O レートの構成」

innodb_io_capacity_max

セクション14.12「InnoDB の起動オプションおよびシステム変数」
セクション14.13.1.6「InnoDB バッファープールのフラッシュのチューニング」

innodb_large_prefix

セクション13.1.13「CREATE INDEX 構文」
セクション13.1.17「CREATE TABLE 構文」
セクション14.6.7「InnoDB テーブル上の制限」
セクション8.3.4「カラムインデックス」
セクション2.11.3「テーブルまたはインデックスの再構築が必要かどうかのチェック」
セクション10.1.11「以前の Unicode サポートから現在の Unicode サポートへのアップグレード」

innodb_lock_wait_timeout

セクション14.19.5「InnoDB のエラーコード」
セクション14.12「InnoDB の起動オプションおよびシステム変数」
MySQL 用語集
セクション14.2.10「デッドロックの検出とロールバック」
セクション17.1.4.3「レプリケーションスレーブのオプションと変数」
セクション17.4.1.29「レプリケーション再試行とタイムアウト」

innodb_locks_unsafe_for_binlog

セクション14.2.8「InnoDB のさまざまな SQL ステートメントで設定された口ック」
セクション14.2.6「InnoDB のレコード、ギャップ、およびネクストキーロック」
セクション14.12「InnoDB の起動オプションおよびシステム変数」
MySQL 用語集
セクション13.3.6「SET TRANSACTION 構文」
セクション14.2.4「一貫性非口ック読み取り」

innodb_log_buffer_size

セクション8.5.7「InnoDB ディスク I/O の最適化」
MySQL 用語集

innodb_log_compressed_pages

セクション1.4「MySQL 5.6 の新機能」

innodb_log_file_size

セクション14.3「InnoDB の構成」
セクション14.12「InnoDB の起動オプションおよびシステム変数」
セクション8.5.7「InnoDB ディスク I/O の最適化」
セクション14.13.1.6「InnoDB バッファープールのフラッシュのチューニング」
セクション14.5.7「InnoDB ログファイルの数またはサイズの変更、および InnoDB テーブルスペースのサイズの変更」
セクション2.11.1.3「MySQL 5.5 から 5.6 へのアップグレード」
セクション1.4「MySQL 5.6 の新機能」
MySQL 用語集
セクション4.4.3「mysql_install_db — MySQL データディレクトリの初期化」
セクション5.1.2.1「サーバーのデフォルト値への変更」
セクション14.3.1「読み取り専用操作用の InnoDB の構成」

innodb_log_files_in_group

セクション14.12「InnoDB の起動オプションおよびシステム変数」
セクション14.5.7「InnoDB ログファイルの数またはサイズの変更、および InnoDB テーブルスペースのサイズの変更」
セクション2.11.1.3「MySQL 5.5 から 5.6 へのアップグレード」
セクション1.4「MySQL 5.6 の新機能」
MySQL 用語集

innodb_log_group_home_dir

セクション14.12「InnoDB の起動オプションおよびシステム変数」
MySQL 用語集
セクション4.4.3「mysql_install_db — MySQL データディレクトリの初期化」

innodb_lru_scan_depth

セクション8.5.7「InnoDB ディスク I/O の最適化」
セクション14.13.1.6「InnoDB バッファープールのフラッシュのチューニング」
MySQL 用語集

innodb_max_dirty_pages_pct

セクション14.12「InnoDB の起動オプションおよびシステム変数」
セクション8.5.7「InnoDB ディスク I/O の最適化」
セクション14.13.1.6「InnoDB バッファープールのフラッシュのチューニング」
セクション14.13.1.2「InnoDB バッファープールのフラッシュの頻度の構成」

innodb_max_dirty_pages_pct_lwm

セクション14.13.1.6「InnoDB バッファープールのフラッシュのチューニング」

innodb_max_purge_lag

セクション14.12「InnoDB の起動オプションおよびシステム変数」
セクション8.5.7「InnoDB ディスク I/O の最適化」
セクション14.2.12「InnoDB マルチバージョン」
MySQL 用語集

innodb_max_purge_lag_delay

セクション14.12「InnoDB の起動オプションおよびシステム変数」

innodb_monitor_disable

セクション21.29.19「INFORMATION_SCHEMA INNODB_METRICS テーブル」
セクション14.14.6「InnoDB INFORMATION_SCHEMA メトリックテーブル」

innodb_monitor_enable

セクション21.29.19「INFORMATION_SCHEMA INNODB_METRICS テーブル」

セクション14.14.6「InnoDB INFORMATION_SCHEMA メトリックテーブル」

innodb_monitor_reset

セクション21.29.19「INFORMATION_SCHEMA INNODB_METRICS テーブル」
セクション14.14.6「InnoDB INFORMATION_SCHEMA メトリックテーブル」

innodb_monitor_reset_all

セクション21.29.19「INFORMATION_SCHEMA INNODB_METRICS テーブル」
セクション14.14.6「InnoDB INFORMATION_SCHEMA メトリックテーブル」

innodb_old_blocks_pct

セクション14.12「InnoDB の起動オプションおよびシステム変数」
セクション8.9.1「InnoDB バッファープール」
MySQL 用語集
セクション14.13.1.3「バッファープールをスキャンに耐えられるようにする」

innodb_old_blocks_time

セクション14.12「InnoDB の起動オプションおよびシステム変数」
セクション8.9.1「InnoDB バッファープール」
セクション2.11.1.3「MySQL 5.5 から 5.6 へのアップグレード」
MySQL 用語集
セクション5.1.2.1「サーバーのデフォルト値への変更」
セクション14.13.1.3「バッファープールをスキャンに耐えられるようにする」

innodb_online_alter_log_max_size

MySQL 用語集
セクション14.11.6「オンライン DDL の実装の詳細」

innodb_open_files

セクション8.5.7「InnoDB ディスク I/O の最適化」
セクション2.11.1.3「MySQL 5.5 から 5.6 へのアップグレード」
セクション5.1.2.1「サーバーのデフォルト値への変更」
セクション5.1.4「サーバーシステム変数」

innodb_optimize_fulltext_only

セクション14.2.13.3「FULLTEXT インデックス」
セクション21.29.22「INFORMATION_SCHEMA INNODB_FT_INDEX_TABLE テーブル」
セクション14.14.4「InnoDB INFORMATION_SCHEMA FULLTEXT インデックステーブル」
セクション12.9.6「MySQL の全文検索の微調整」
セクション13.7.2.4「OPTIMIZE TABLE 構文」

innodb_page_size

セクション13.1.17「CREATE TABLE 構文」
セクション21.29.15「INFORMATION_SCHEMA INNODB_SYS_TABLESPACES テーブル」
セクション14.18.8「InnoDB memcached プラグインのトラブルシューティング」
セクション14.2.13.4「InnoDB インデックスの物理構造」
セクション14.7.5「InnoDB テーブルでの圧縮の動作」
セクション14.13.17「InnoDB テーブルに対する ANALYZE TABLE の複雑さの推定」
セクション14.6.7「InnoDB テーブル上の制限」
セクション8.5.7「InnoDB ディスク I/O の最適化」
セクション14.6.4「MyISAM から InnoDB へのテーブルの変換」
セクション2.11.2.1「MySQL 5.5 へのダウングレード」
セクション1.4「MySQL 5.6 の新機能」
MySQL 用語集
セクション14.7.2「テーブル圧縮の有効化」
セクション14.10.2「ファイル領域管理」

innodb_print_all_deadlocks

セクション14.19「InnoDB のトラブルシューティング」
セクション14.6.4「MyISAM から InnoDB へのテーブルの変換」
MySQL 用語集
セクション14.2.11「デッドロックの対処方法」

innodb_purge_batch_size

セクション14.13.13「InnoDB のパージスケジューリングの構成」

innodb_purge_threads

セクション14.13.13「InnoDB のパージスケジューリングの構成」
セクション14.13.8「InnoDB マスタースレッドの I/O レートの構成」
MySQL 用語集

innodb_random_read_ahead

セクション8.5.7「InnoDB ディスク I/O の最適化」
セクション14.13.1.1「InnoDB バッファープールのプリフェッチ (先読み) の構成」

innodb_read_ahead_threshold

セクション14.12「InnoDB の起動オプションおよびシステム変数」
セクション8.5.7「InnoDB ディスク I/O の最適化」
セクション14.13.1.1「InnoDB バッファープールのプリフェッチ (先読み) の構成」

innodb_read_io_threads

セクション14.12「InnoDB の起動オプションおよびシステム変数」
セクション8.5.7「InnoDB ディスク I/O の最適化」
セクション14.13.6「InnoDB バックグラウンド I/O スレッドの数の構成」
セクション14.15.3「InnoDB 標準モニターおよびロックモニターの出力」
セクション24.1.1「MySQL のスレッド」

innodb_rollback_segments

セクション14.12「InnoDB の起動オプションおよびシステム変数」
セクション8.5.7「InnoDB ディスク I/O の最適化」
セクション14.5.6「個別のテーブルスペースへの InnoDB Undo ログの格納」
セクション14.13.12「複数のロールバックセグメントによるスケーラビリティーの向上」

innodb_sort_buffer_size

セクション14.12「InnoDB の起動オプションおよびシステム変数」
MySQL 用語集

innodb_spin_wait_delay

セクション14.13.10「スピンロックのポーリングの構成」

innodb_stats_auto_recalc

セクション13.1.17「CREATE TABLE 構文」
InnoDB 永続的統計テーブルの例
セクション14.13.16.1「永続的オプティマイザ統計のパラメータの構成」

innodb_stats_method

セクション8.3.7「InnoDB および MyISAM インデックス統計コレクション」
MySQL 用語集

innodb_stats_on_metadata

セクション2.11.1.3「MySQL 5.5 から 5.6 へのアップグレード」
MySQL 用語集
セクション5.1.2.1「サーバーのデフォルト値への変更」
セクション14.13.16.2「非永続的オプティマイザ統計のパラメータの構成」

innodb_stats_persistent

セクション13.7.2.1「ANALYZE TABLE 構文」
セクション13.1.13「CREATE INDEX 構文」
セクション13.1.17「CREATE TABLE 構文」
セクション14.12「InnoDB の起動オプションおよびシステム変数」
セクション14.13.17「InnoDB テーブルに対する ANALYZE TABLE の複雑さの推定」
セクション14.6.7「InnoDB テーブル上の制限」
セクション14.13.16.1「永続的オプティマイザ統計のパラメータの構成」
セクション14.13.16.2「非永続的オプティマイザ統計のパラメータの構成」

innodb_stats_persistent_sample_pages

セクション14.12「InnoDB の起動オプションおよびシステム変数」
InnoDB オプティマイザ統計でサンプリングされるページの数の構成
セクション14.13.17「InnoDB テーブルに対する ANALYZE TABLE の複雑さの推定」
セクション14.6.7「InnoDB テーブル上の制限」
セクション14.13.16.1「永続的オプティマイザ統計のパラメータの構成」

innodb_stats_sample_pages

MySQL 用語集

innodb_stats_transient_sample_pages

セクション14.12「InnoDB の起動オプションおよびシステム変数」
セクション14.13.17「InnoDB テーブルに対する ANALYZE TABLE の複雑さの推定」
セクション14.6.7「InnoDB テーブル上の制限」
セクション14.13.16.2「非永続的オプティマイザ統計のパラメータの構成」

innodb_status_output

セクション14.12「InnoDB の起動オプションおよびシステム変数」
セクション14.15.2「InnoDB モニターの有効化」
セクション1.4「MySQL 5.6 の新機能」

innodb_status_output_locks

セクション14.12「InnoDB の起動オプションおよびシステム変数」
セクション14.15.2「InnoDB モニターの有効化」
セクション1.4「MySQL 5.6 の新機能」

innodb_strict_mode

セクション13.1.17「CREATE TABLE 構文」
セクション14.7.5「InnoDB テーブルでの圧縮の動作」
MySQL 用語集
セクション14.7.7「SQL 圧縮構文の警告とエラー」
セクション5.1.7「サーバー SQL モード」
セクション14.7.2「テーブル圧縮の有効化」

innodb_support_xa

セクション14.18.5.3「InnoDB memcached プラグインのパフォーマンスのチューニング」
セクション8.5.2「InnoDB トランザクション管理の最適化」
MySQL 用語集

innodb_table_locks

セクション14.12「InnoDB の起動オプションおよびシステム変数」
セクション14.6.7「InnoDB テーブル上の制限」

innodb_temp_data_file_path

MySQL 用語集

innodb_thread_concurrency

セクション14.13.5「InnoDB のスレッド並列性の構成」
セクション14.12「InnoDB の起動オプションおよびシステム変数」
セクション8.5.8「InnoDB 構成変数の最適化」
セクション14.15.3「InnoDB 標準モニターおよびロックモニターの出力」
セクションA.14「MySQL 5.6 FAQ: MySQL エンタープライズスケーラビリティースレッドプール」

innodb_thread_sleep_delay

セクション14.13.5「InnoDB のスレッド並列性の構成」
セクション14.12「InnoDB の起動オプションおよびシステム変数」

innodb_undo_directory

セクション14.12「InnoDB の起動オプションおよびシステム変数」
MySQL 用語集
セクション14.5.6「個別のテーブルスペースへの InnoDB Undo ログの格納」
セクション14.3.1「読み取り専用操作用の InnoDB の構成」

innodb_undo_logs
セクション14.12「InnoDB の起動オプションおよびシステム変数」
セクション5.1.6「サーバーステータス変数」
セクション14.5.6「個別のテーブルスペースへの InnoDB Undo ログの格納」

innodb_undo_tablespaces
セクション14.12「InnoDB の起動オプションおよびシステム変数」
MySQL 用語集
セクション14.5.6「個別のテーブルスペースへの InnoDB Undo ログの格納」
セクション14.3.1「読み取り専用操作用の InnoDB の構成」

innodb_use_native_aio
セクション14.12「InnoDB の起動オプションおよびシステム変数」
MySQL 用語集

innodb_use_sys_malloc
セクション14.13.3「InnoDB のためのメモリーアロケータの構成」
セクション14.12「InnoDB の起動オプションおよびシステム変数」
セクション1.4「MySQL 5.6 の新機能」

innodb_write_io_threads
セクション14.12「InnoDB の起動オプションおよびシステム変数」
セクション8.5.7「InnoDB ディスク I/O の最適化」
セクション14.13.6「InnoDB バックグラウンド I/O スレッドの数の構成」
セクション14.15.3「InnoDB 標準モニターおよびロックモニターの出力」
セクション24.1.1「MySQL のスレッド」

insert_id
セクション15.8.3「FEDERATED ストレージエンジンの注記とヒント」
セクション5.1.4「サーバーシステム変数」

interactive_timeout
セクション23.7.7.53「mysql_real_connect()」
セクション5.1.4「サーバーシステム変数」
セクションB.5.2.11「通信エラーおよび中止された接続」

J

[索引の先頭]

join_buffer_size
Batched Key Access 結合
セクション2.11.1.3「MySQL 5.5 から 5.6 へのアップグレード」
セクション8.2.1.10「Nested Loop 結合アルゴリズム」
セクション5.1.2.1「サーバーのデフォルト値への変更」
セクション5.1.4「サーバーシステム変数」

K

[索引の先頭]

keep_files_on_create
セクション5.1.4「サーバーシステム変数」

key_buffer_size
セクション8.2.2.3「DELETE ステートメントの速度」
セクション14.6.4「MyISAM から InnoDB へのテーブルの変換」
セクション8.9.2「MyISAM キーキャッシュ」
セクション8.6.2「MyISAM テーブルの一括データロード」
セクション7.6.3「MyISAM テーブルの修復方法」
セクション8.11.4.1「MySQL のメモリーの使用方法」
セクションB.5.8「MySQL の既知の問題」
セクション8.6.3「REPAIR TABLE ステートメントの速度」
セクション4.2.6「オプションファイルの使用」
セクション8.9.2.6「キーキャッシュの再構築」
セクション8.8.4「クエリーパフォーマンスの推定」
セクション5.1.3「サーバーコマンドオプション」
セクション5.1.4「サーバーシステム変数」
セクション5.1.6「サーバーステータス変数」
セクション8.11.2「サーバーパラメータのチューニング」
セクション5.1.5.1「構造化システム変数」
セクション8.9.2.2「複合キーキャッシュ」

key_cache_age_threshold
セクション5.1.4「サーバーシステム変数」
セクション8.9.2.3「ミッドポイント挿入戦略」
セクション5.1.5.1「構造化システム変数」

key_cache_block_size
セクション8.9.2.6「キーキャッシュの再構築」
セクション8.9.2.5「キーキャッシュブロックサイズ」
セクション5.1.4「サーバーシステム変数」
セクション5.1.5.1「構造化システム変数」

key_cache_division_limit
セクション5.1.4「サーバーシステム変数」
セクション8.9.2.3「ミッドポイント挿入戦略」
セクション5.1.5.1「構造化システム変数」

L

[索引の先頭]

large_files_support
セクション5.1.4「サーバーシステム変数」
セクション19.6「パーティショニングの制約と制限」

large_page_size
セクション5.1.4「サーバーシステム変数」

large_pages
セクション5.1.4「サーバーシステム変数」

last_insert_id
セクション5.1.4「サーバーシステム変数」
セクション17.4.1.34「レプリケーションと変数」
セクション5.2.4.3「混合形式のバイナリロギング形式」

lc_messages
セクション10.2「エラーメッセージ言語の設定」
セクション5.1.4「サーバーシステム変数」

lc_messages_dir
セクション10.2「エラーメッセージ言語の設定」
セクション5.1.4「サーバーシステム変数」

lc_time_names
セクション10.7「MySQL Server のロケールサポート」
セクション5.1.4「サーバーシステム変数」
セクション17.4.1.34「レプリケーションと変数」
セクション12.5「文字列関数」
セクション12.7「日付および時間関数」
セクション5.2.4.3「混合形式のバイナリロギング形式」

license
セクション5.1.4「サーバーシステム変数」

local
セクション13.2.6「LOAD DATA INFILE 構文」
セクション6.1.6「LOAD DATA LOCAL のセキュリティーの問題」
セクション13.2.7「LOAD XML 構文」
セクション2.9.4「MySQL ソース構成オプション」

local_infile
セクション5.1.4「サーバーシステム変数」

lock_wait_timeout
セクション5.1.4「サーバーシステム変数」

locked_in_memory
セクション5.1.4「サーバーシステム変数」

log
セクション5.1.3「サーバーコマンドオプション」
セクション5.1.4「サーバーシステム変数」
セクション14.18.6「レプリケーションでの InnoDB memcached プラグインの使用」

log_bin
セクション1.4「MySQL 5.6 の新機能」
セクション18.3.4.3「MySQL Cluster のシステム変数」
セクション17.1.4.4「バイナリログのオプションと変数」

log_bin_basename
セクション1.4「MySQL 5.6 の新機能」
セクション17.1.4.4「バイナリログのオプションと変数」

log_bin_index
セクション17.1.4.4「バイナリログのオプションと変数」

log_bin_trust_function_creators
セクションA.4「MySQL 5.6 FAQ: ストアドプロシージャーおよびストアドファンクション」
セクション5.1.4「サーバーシステム変数」
セクション20.7「ストアドプログラムのバイナリロギング」
セクション17.1.4.4「バイナリログのオプションと変数」

log_bin_use_v
セクション17.4.2「MySQL バージョン間のレプリケーション互換性」
セクション17.1.4.4「バイナリログのオプションと変数」

log_error
セクション5.2.2「エラーログ」
セクション5.1.4「サーバーシステム変数」

log_output
セクション5.1.4「サーバーシステム変数」
セクション5.2.5「スロークエリーログ」
セクション5.2.3「一般クエリーログ」
セクション5.2.1「一般クエリーログおよびスロークエリーログの出力先の選択」

log_queries_not_using_indexes
セクション5.1.4「サーバーシステム変数」
セクション5.2.5「スロークエリーログ」

log_slave_updates
セクション17.1.4.4「バイナリログのオプションと変数」

log_slow_admin_statements
セクション5.1.3「サーバーコマンドオプション」
セクション5.1.4「サーバーシステム変数」
セクション5.2.5「スロークエリーログ」

log_slow_queries
セクション5.1.3「サーバーコマンドオプション」
セクション5.1.4「サーバーシステム変数」

log_slow_slave_statements
セクション5.2.5「スロークエリーログ」
セクション17.1.4.3「レプリケーションスレーブのオプションと変数」

log_throttle_queries_not_using_indexes
セクション5.1.4「サーバーシステム変数」
セクション5.2.5「スロークエリーログ」

log_warnings
セクション5.2.2「エラーログ」
セクション5.1.4「サーバーシステム変数」

long_query_time
セクション5.2「MySQL Server ログ」
セクション4.5.2「mysqladmin — MySQL サーバーの管理を行うクライアント」
セクション5.1.4「サーバーシステム変数」
セクション5.1.6「サーバーステータス変数」
セクション5.2.5「スロークエリーログ」
セクション17.1.4.3「レプリケーションスレーブのオプションと変数」

low_priority_updates
セクション1.4「MySQL 5.6 の新機能」
セクション5.1.4「サーバーシステム変数」
セクション8.10.2「テーブルロックの問題」

lower_case_file_system
セクション5.1.4「サーバーシステム変数」

lower_case_table_names
セクション13.7.1.4「GRANT 構文」

セクション21.29.9「INFORMATION_SCHEMA INNODB_SYS_COLUMNS テーブル」
セクション21.29.7「INFORMATION_SCHEMA INNODB_SYS_TABLES テーブル」
セクション13.7.1.6「REVOKE 構文」
セクション13.7.5.38「SHOW TABLES 構文」
セクション17.2.3「サーバーがレプリケーションフィルタリングルールをどのように評価するか」
セクション5.1.4「サーバーシステム変数」
セクション17.4.1.34「レプリケーションと変数」
セクション13.1.17.2「外部キー制約の使用」
セクション10.1.7.9「照合順序と INFORMATION_SCHEMA 検索」
セクション9.2.2「識別子の大文字と小文字の区別」
セクション1.6「質問またはバグをレポートする方法」

M

[索引の先頭]

master_info_repository

セクション17.1.4.3「レプリケーションスレーブのオプションと変数」

master_verify_checksum

セクション1.4「MySQL 5.6 の新機能」
MySQL 用語集
セクション5.2.4「バイナリログ」
セクション17.1.4.4「バイナリログのオプションと変数」

max_allowed_packet

セクション11.4.3「BLOB 型と TEXT 型」
セクション12.19.1「GROUP BY (集約) 関数」
セクション2.11.1.3「MySQL 5.5 から 5.6 へのアップグレード」
セクション1.4「MySQL 5.6 の新機能」
セクション23.7「MySQL C API」
セクション18.6.11「MySQL Cluster レプリケーションの競合解決」
セクション8.11.4.1「MySQL のメモリーの使用方法」
セクションB.5.2.9「MySQL サーバーが存在しなくなりました」
セクションB.5.2.3「MySQL サーバーへの接続が失われました」
セクション23.7.11.26「mysql_stmt_send_long_data()」
セクション23.7.7.72「mysql_use_result()」
セクション4.2.6「オプションファイルの使用」
セクション5.1.2.1「サーバーのデフォルト値への変更」
セクション5.1.4「サーバーシステム変数」
セクションB.5.2.10「パケットが大きすぎます」
セクション17.4.1.20「レプリケーションと max_allowed_packet」
セクション17.1.4.3「レプリケーションスレーブのオプションと変数」
セクション12.5「文字列関数」
セクション12.3.2「比較関数と演算子」
セクションB.5.2.11「通信エラーおよび中止された接続」
セクションB.5.5.6「関連するテーブルからの行の削除」

max_binlog_cache_size

セクション5.2.4「バイナリログ」
セクション17.1.4.4「バイナリログのオプションと変数」

max_binlog_size

セクション5.2「MySQL Server ログ」
セクション5.1.4「サーバーシステム変数」
セクション5.2.7「サーバーログの保守」
セクション17.2.2.1「スレーブリレーログ」
セクション5.2.4「バイナリログ」
セクション17.1.4.4「バイナリログのオプションと変数」
セクション17.1.4.3「レプリケーションスレーブのオプションと変数」

max_binlog_stmt_cache_size

セクション17.1.4.4「バイナリログのオプションと変数」

max_connect_errors

セクション8.11.5.2「DNS ルックアップの最適化とホストキャッシュ」
セクション13.7.6.3「FLUSH 構文」
セクション22.9.10.1「host_cache テーブル」
セクション2.11.1.3「MySQL 5.5 から 5.6 へのアップグレード」
セクション1.4「MySQL 5.6 の新機能」
セクション5.1.2.1「サーバーのデフォルト値への変更」
セクション5.1.4「サーバーシステム変数」
セクションB.5.2.6「ホスト 'host_name' は拒否されました」

max_connections

セクションB.5.2.18「'File' が見つかりません、および同様のエラー」
セクション24.4.1.4「gdb での mysqld のデバッグ」
セクション2.11.1.3「MySQL 5.5 から 5.6 へのアップグレード」
セクション8.4.3.1「MySQL でのテーブルのオープンとクローズの方法」
セクション6.2.1「MySQL で提供される権限」
セクション8.11.5.1「MySQL のクライアント接続のためのスレッドの使用方法」
セクション5.1.2.1「サーバーのデフォルト値への変更」
セクション5.1.3「サーバーコマンドオプション」
セクション5.1.4「サーバーシステム変数」
セクション5.1.6「サーバーステータス変数」
セクション8.11.6.2「スレッドプール操作」
セクション22.12「パフォーマンススキーマシステム変数」
セクションB.5.2.7「接続が多すぎます」

max_delayed_threads

セクション5.1.4「サーバーシステム変数」

max_error_count

セクション13.2.6「LOAD DATA INFILE 構文」
セクション13.6.7.4「RESIGNAL 構文」
セクション13.7.5.18「SHOW ERRORS 構文」
セクション13.7.5.41「SHOW WARNINGS 構文」
セクション5.1.4「サーバーシステム変数」
診断領域関連のシステム変数

max_heap_table_size

セクション15.3「MEMORY ストレージエンジン」
セクション8.4.4「MySQL が内部一時テーブルを使用する仕組み」
セクション5.1.4「サーバーシステム変数」
セクション5.1.6「サーバーステータス変数」
セクションD.3「サーバー側のカーソルの制約」

セクションD.10.3「テーブルサイズの制限」
セクション17.4.1.21「レプリケーションと MEMORY テーブル」
セクション17.4.1.34「レプリケーションと変数」

max_insert_delayed_threads

セクション5.1.4「サーバーシステム変数」

max_join_size

セクション8.8.2「EXPLAIN 出力フォーマット」
セクション1.4「MySQL 5.6 の新機能」
セクション13.7.4「SET 構文」
セクション5.1.4「サーバーシステム変数」
セクション5.1.5「システム変数の使用」

max_length_for_sort_data

セクション8.2.1.15「ORDER BY の最適化」
セクション5.1.4「サーバーシステム変数」

max_prepared_stmt_count

セクション13.5.3「DEALLOCATE PREPARE 構文」
セクション5.1.4「サーバーシステム変数」
セクション5.1.6「サーバーステータス変数」
セクション8.9.4「プリペアドステートメントおよびストアドプログラムのキャッシュ」
セクション13.5「準備済みステートメントのための SQL 構文」

max_relay_log_size

セクション5.1.4「サーバーシステム変数」
セクション17.2.2.1「スレーブリレーログ」
セクション17.1.4.4「バイナリログのオプションと変数」
セクション17.1.4.3「レプリケーションスレーブのオプションと変数」

max_seeks_for_key

セクション14.6.7「InnoDB テーブル上の制限」
セクション5.1.4「サーバーシステム変数」

max_sort_length

セクション11.4.3「BLOB 型と TEXT 型」
セクション13.1.17「CREATE TABLE 構文」
セクションB.5.8「MySQL の既知の問題」
セクション5.1.4「サーバーシステム変数」

max_sp_recursion_depth

セクション5.1.4「サーバーシステム変数」
セクション20.2.1「ストアドルーチンの構文」

max_tmp_tables

セクション5.1.4「サーバーシステム変数」

max_user_connections

セクション13.7.1.4「GRANT 構文」
セクション6.3.4「アカウントリソース制限の設定」
セクション5.1.4「サーバーシステム変数」
セクション6.1.3「攻撃者に対する MySQL のセキュアな状態の維持」

max_write_lock_count

セクション5.1.4「サーバーシステム変数」
セクション8.10.2「テーブルロックの問題」

metadata_locks_cache_size

セクション5.1.4「サーバーシステム変数」

metadata_locks_hash_instances

セクション5.1.4「サーバーシステム変数」

min_examined_row_limit

セクション5.1.4「サーバーシステム変数」
セクション5.2.5「スロークエリーログ」

myisam_data_pointer_size

セクション13.1.17「CREATE TABLE 構文」
セクション5.1.4「サーバーシステム変数」
セクションD.10.3「テーブルサイズの制限」

myisam_max_sort_file_size

セクション15.2.1「MyISAM 起動オプション」
セクション8.6.3「REPAIR TABLE ステートメントの速度」
セクション5.1.4「サーバーシステム変数」
セクション19.6「パーティショニングの制約と制限」

myisam_mmap_size

セクション5.1.4「サーバーシステム変数」

myisam_recover_options

セクション5.1.4「サーバーシステム変数」

myisam_repair_threads

セクション5.1.4「サーバーシステム変数」

myisam_sort_buffer_size

セクション13.1.7「ALTER TABLE 構文」
セクション15.2.1「MyISAM 起動オプション」
セクション4.6.3.1「myisamchk の一般オプション」
セクション8.6.3「REPAIR TABLE ステートメントの速度」
セクション5.1.4「サーバーシステム変数」

myisam_stats_method

セクション8.3.7「InnoDB および MyISAM インデックス統計コレクション」
セクション5.1.4「サーバーシステム変数」

myisam_use_mmap

セクション8.11.4.1「MySQL のメモリーの使用方法」
セクション5.1.4「サーバーシステム変数」

N

[索引の先頭]

named_pipe

セクション5.1.4「サーバーシステム変数」

ndb_autoincrement_prefetch_sz

セクション18.3.4.3「MySQL Cluster のシステム変数」

ndb_cache_check_time

セクション18.3.4.3「MySQL Cluster のシステム変数」

ndb_deferred_constraints
セクション18.3.4.3「MySQL Cluster のシステム変数」

ndb_distribution
セクション18.3.4.3「MySQL Cluster のシステム変数」

ndb_eventbuffer_max_alloc
セクション18.3.4.3「MySQL Cluster のシステム変数」

ndb_extra_logging
セクション18.3.4.3「MySQL Cluster のシステム変数」

ndb_force_send
セクション18.3.4.3「MySQL Cluster のシステム変数」

ndb_index_stat_cache_entries
セクション18.3.4.3「MySQL Cluster のシステム変数」

ndb_index_stat_enable
セクション18.3.4.3「MySQL Cluster のシステム変数」

ndb_index_stat_option
セクション18.3.4.3「MySQL Cluster のシステム変数」

ndb_index_stat_update_freq
セクション18.3.4.3「MySQL Cluster のシステム変数」

ndb_join_pushdown
セクション8.8.2「EXPLAIN 出力フォーマット」
セクション18.3.4.3「MySQL Cluster のシステム変数」

ndb_log_apply_status
セクション18.3.4.3「MySQL Cluster のシステム変数」

ndb_log_bin
セクション18.3.4.3「MySQL Cluster のシステム変数」

ndb_log_binlog_index
セクション18.3.4.3「MySQL Cluster のシステム変数」

ndb_log_empty_epochs
セクション18.3.4.3「MySQL Cluster のシステム変数」

ndb_log_exclusive_reads
セクション18.1.4.2「MySQL Cluster NDB 7.4 での MySQL Cluster の開発」
セクション18.3.4.3「MySQL Cluster のシステム変数」
セクション18.6.11「MySQL Cluster レプリケーションの競合解決」
セクション18.3.4.2「MySQL Cluster 用の MySQL Server オプション」

ndb_log_orig
セクション18.3.4.3「MySQL Cluster のシステム変数」

ndb_log_transaction_id
セクション18.3.4.3「MySQL Cluster のシステム変数」

ndb_optimized_node_selection
セクション18.3.4.3「MySQL Cluster のシステム変数」
セクション18.5.6.3「MySQL Cluster 管理クライアントでの CLUSTERLOG STATISTICS の使用」

ndb_recv_thread_activation_threshold
セクション18.3.4.3「MySQL Cluster のシステム変数」

ndb_recv_thread_cpu_mask
セクション18.3.4.3「MySQL Cluster のシステム変数」

ndb_report_thresh_binlog_epoch_slip
セクション18.3.4.3「MySQL Cluster のシステム変数」

ndb_report_thresh_binlog_mem_usage
セクション18.3.4.3「MySQL Cluster のシステム変数」

ndb_show_foreign_key_mock_tables
セクション18.3.4.3「MySQL Cluster のシステム変数」

ndb_slave_conflict_role
セクション18.1.4.2「MySQL Cluster NDB 7.4 での MySQL Cluster の開発」
セクション18.3.4.3「MySQL Cluster のシステム変数」
セクション18.1.4「MySQL Cluster の開発履歴」

ndb_table_no_logging
セクション18.3.4.3「MySQL Cluster のシステム変数」

ndb_table_temporary
セクション18.3.4.3「MySQL Cluster のシステム変数」

ndb_use_copying_alter_table
セクション18.3.4.3「MySQL Cluster のシステム変数」

ndb_use_exact_count
セクション18.3.4.3「MySQL Cluster のシステム変数」

ndb_use_transactions
セクション18.3.4.3「MySQL Cluster のシステム変数」

ndb_version
セクション18.3.4.3「MySQL Cluster のシステム変数」

ndb_version_string
セクション18.3.4.3「MySQL Cluster のシステム変数」

ndbinfo_database
セクション18.3.4.3「MySQL Cluster のシステム変数」

ndbinfo_max_bytes
セクション18.3.4.3「MySQL Cluster のシステム変数」

ndbinfo_max_rows
セクション18.3.4.3「MySQL Cluster のシステム変数」

ndbinfo_offline
セクション18.3.4.3「MySQL Cluster のシステム変数」

ndbinfo_show_hidden
セクション18.3.4.3「MySQL Cluster のシステム変数」
セクション18.5.10.4「ndbinfo cluster_operations テーブル」
セクション18.5.10.5「ndbinfo cluster_transactions テーブル」
セクション18.5.10.20「ndbinfo server_operations テーブル」
セクション18.5.10.21「ndbinfo server_transactions テーブル」

ndbinfo_table_prefix
セクション18.3.4.3「MySQL Cluster のシステム変数」

ndbinfo_version
セクション18.3.4.3「MySQL Cluster のシステム変数」

net_buffer_length
セクション23.7「MySQL C API」
セクション8.11.4.1「MySQL のメモリーの使用方法」
セクション4.5.4「mysqldump — データベースバックアッププログラム」
セクション5.1.4「サーバーシステム変数」

net_read_timeout
セクションB.5.2.3「MySQL サーバーへの接続が失われました」
セクション5.1.4「サーバーシステム変数」

net_retry_count
セクション5.1.4「サーバーシステム変数」

net_write_timeout
セクション5.1.4「サーバーシステム変数」

new
セクション5.1.4「サーバーシステム変数」

O

[索引の先頭]

old
セクション5.1.4「サーバーシステム変数」

old_alter_table
セクション13.1.7「ALTER TABLE 構文」
セクション13.7.2.4「OPTIMIZE TABLE 構文」
セクション14.11.2「オンライン DDL でのパフォーマンスと並列性に関する考慮事項」
セクション14.11.3「オンライン DDL の SQL 構文」
セクション5.1.4「サーバーシステム変数」

old_passwords
セクション13.7.1.1「ALTER USER 構文」
セクション13.7.1.2「CREATE USER 構文」
セクション6.1.2.5「MySQL 4.1 でのパスワードハッシュ変更がアプリケーションプログラムに及ぼす影響」
セクション2.11.1.3「MySQL 5.5 から 5.6 へのアップグレード」
セクション1.4「MySQL 5.6 の新機能」
セクション6.1.2.4「MySQL でのパスワードハッシュ」
セクション13.7.1.7「SET PASSWORD 構文」
セクション6.3.8.4「SHA-256 認証プラグイン」
セクション6.3.5「アカウントパスワードの割り当て」
セクションB.5.2.4「クライアントは認証プロトコルに対応できません」
セクション5.1.3「サーバーコマンドオプション」
セクション5.1.4「サーバーシステム変数」
セクション6.3.6「パスワードの期限切れとサンドボックスモード」
セクション12.13「暗号化関数と圧縮関数」

open_files_limit
セクションB.5.2.18「'File' が見つかりません、および同様のエラー」
セクション5.1.2.1「サーバーのデフォルト値への変更」
セクション5.1.4「サーバーシステム変数」
セクション22.12「パフォーマンススキーマシステム変数」
セクション19.6「パーティショニングの制約と制限」

optimizer_join_cache_level
Block Nested Loop および Batched Key Access アルゴリズムの結合バッファー管理
セクション5.1.4「サーバーシステム変数」
外部結合と準結合の Block Nested Loop アルゴリズム

optimizer_prune_level
セクション8.8.5.1「クエリー計画評価の制御」
セクション5.1.4「サーバーシステム変数」

optimizer_search_depth
セクション8.8.5.1「クエリー計画評価の制御」
セクション5.1.4「サーバーシステム変数」

optimizer_switch
Batched Key Access 結合
Block Nested Loop および Batched Key Access アルゴリズムの結合バッファー管理
セクション8.2.1.13「Multi-Range Read の最適化」
セクション1.4「MySQL 5.6 の新機能」
セクション8.2.1.6「インデックスコンディションプッシュダウンの最適化」
セクション8.2.1.7「インデックス拡張の使用」
セクション8.2.1.5「エンジンコンディションプッシュダウンの最適化」
サブクエリー実体化によるサブクエリーの最適化
セクション5.1.4「サーバーシステム変数」
セクション8.8.5.2「切り替え可能な最適化の制御」
外部結合と準結合の Block Nested Loop アルゴリズム
準結合変換によるサブクエリーの最適化

optimizer_trace
セクション21.11「INFORMATION_SCHEMA OPTIMIZER_TRACE テーブル」
セクション5.1.4「サーバーシステム変数」

optimizer_trace_features
セクション5.1.4「サーバーシステム変数」

optimizer_trace_limit
セクション5.1.4「サーバーシステム変数」

optimizer_trace_max_mem_size
セクション5.1.4「サーバーシステム変数」

optimizer_trace_offset
セクション5.1.4「サーバーシステム変数」

P

[索引の先頭]

performance_schema
セクション5.1.2.1「サーバーのデフォルト値への変更」
セクション22.1「パフォーマンススキーマクイックスタート」
セクション22.12「パフォーマンススキーマシステム変数」
セクション22.2.2「パフォーマンススキーマ起動構成」

performance_schema_accounts_size
セクション22.9.7.1「accounts テーブル」
セクション22.12「パフォーマンススキーマシステム変数」

performance_schema_digests_size
セクション22.9.10.2「performance_timers テーブル」
セクション22.9.9.3「ステートメントサマリーテーブル」
セクション22.12「パフォーマンススキーマシステム変数」
セクション22.13「パフォーマンススキーマステータス変数」
セクション22.7「パフォーマンススキーマステートメントダイジェスト」

performance_schema_events_stages_history_long_size
セクション22.9.5.3「events_stages_history_long テーブル」
セクション22.12「パフォーマンススキーマシステム変数」

performance_schema_events_stages_history_size
セクション22.9.5.2「events_stages_history テーブル」
セクション22.12「パフォーマンススキーマシステム変数」

performance_schema_events_statements_history_long_size
セクション22.9.6.3「events_statements_history_long テーブル」
セクション22.12「パフォーマンススキーマシステム変数」

performance_schema_events_statements_history_size
セクション22.9.6.2「events_statements_history テーブル」
セクション22.12「パフォーマンススキーマシステム変数」

performance_schema_events_waits_history_long_size
セクション22.9.4.3「events_waits_history_long テーブル」
セクション13.7.5.16「SHOW ENGINE 構文」
セクション5.1.2.1「サーバーのデフォルト値への変更」
セクション22.12「パフォーマンススキーマシステム変数」
セクション22.9「パフォーマンススキーマテーブルの説明」

performance_schema_events_waits_history_size
セクション22.9.4.2「events_waits_history テーブル」
セクション13.7.5.16「SHOW ENGINE 構文」
セクション5.1.2.1「サーバーのデフォルト値への変更」
セクション22.12「パフォーマンススキーマシステム変数」
セクション22.9「パフォーマンススキーマテーブルの説明」

performance_schema_hosts_size
セクション22.9.7.2「hosts テーブル」
セクション22.12「パフォーマンススキーマシステム変数」

performance_schema_max_cond_classes
セクション22.12「パフォーマンススキーマシステム変数」

performance_schema_max_cond_instances
セクション5.1.2.1「サーバーのデフォルト値への変更」
セクション22.12「パフォーマンススキーマシステム変数」

performance_schema_max_file_classes
セクション22.12「パフォーマンススキーマシステム変数」

performance_schema_max_file_handles
セクション22.12「パフォーマンススキーマシステム変数」

performance_schema_max_file_instances
セクション5.1.2.1「サーバーのデフォルト値への変更」
セクション22.12「パフォーマンススキーマシステム変数」

performance_schema_max_mutex_classes
セクション22.12「パフォーマンススキーマシステム変数」
セクション22.5「パフォーマンススキーマステータスモニタリング」

performance_schema_max_mutex_instances
セクション5.1.2.1「サーバーのデフォルト値への変更」
セクション22.12「パフォーマンススキーマシステム変数」

performance_schema_max_rwlock_classes
セクション22.12「パフォーマンススキーマシステム変数」

performance_schema_max_rwlock_instances
セクション5.1.2.1「サーバーのデフォルト値への変更」
セクション22.12「パフォーマンススキーマシステム変数」

performance_schema_max_socket_classes
セクション22.12「パフォーマンススキーマシステム変数」

performance_schema_max_socket_instances
セクション22.12「パフォーマンススキーマシステム変数」

performance_schema_max_stage_classes
セクション22.12「パフォーマンススキーマシステム変数」

performance_schema_max_statement_classes
セクション22.12「パフォーマンススキーマシステム変数」

performance_schema_max_table_handles
セクション5.1.2.1「サーバーのデフォルト値への変更」
セクション22.12「パフォーマンススキーマシステム変数」

performance_schema_max_table_instances
セクション5.1.2.1「サーバーのデフォルト値への変更」

セクション22.12「パフォーマンススキーマシステム変数」

performance_schema_max_thread_classes
セクション22.12「パフォーマンススキーマシステム変数」

performance_schema_max_thread_instances
セクション13.7.5.16「SHOW ENGINE 構文」
セクション5.1.2.1「サーバーのデフォルト値への変更」
セクション22.12「パフォーマンススキーマシステム変数」
セクション22.13「パフォーマンススキーマステータス変数」

performance_schema_session_connect_attrs_size
セクション22.12「パフォーマンススキーマシステム変数」

performance_schema_setup_actors_size
セクション22.9.2.1「setup_actors テーブル」
セクション22.12「パフォーマンススキーマシステム変数」

performance_schema_setup_objects_size
セクション22.9.2.4「setup_objects テーブル」
セクション22.12「パフォーマンススキーマシステム変数」

performance_schema_users_size
セクション22.9.7.3「users テーブル」
セクション22.12「パフォーマンススキーマシステム変数」

pid_file
セクション5.1.4「サーバーシステム変数」

plugin_dir
セクション12.17.1「Enterprise Encryption のインストール」
セクション21.14「INFORMATION_SCHEMA PLUGINS テーブル」
セクション24.2.4.6「INFORMATION_SCHEMA プラグインの作成」
セクション14.18.3.1「InnoDB memcached プラグインの前提条件」
セクション13.7.3.3「INSTALL PLUGIN 構文」
セクション6.3.8.5「PAM 認証プラグイン」
PAM 認証プラグインのインストール
セクション13.7.5.26「SHOW PLUGINS 構文」
セクション2.10.1「Unix 類似システムでのインストール後の手順」
セクション6.3.8.6「Windows ネイティブ認証プラグイン」
Windows 認証プラグインのインストール
セクション5.1.3「サーバーコマンドオプション」
セクション5.1.4「サーバーシステム変数」
セクション8.11.6.1「スレッドプールコンポーネントとインストール」
セクション6.3.8.8「ソケットピア証明書認証プラグイン」
セクション6.3.8.9「テスト認証プラグイン」
セクション24.2.4.5「デーモンプラグインの作成」
セクション6.1.2.2「パスワードセキュリティーについての管理者ガイドライン」
パスワード検証プラグインのインストール
セクション24.2.4.10「パスワード検証プラグインの作成」
セクションD.9「プラガブルな認証の制約」
セクション15.11.1「プラガブルストレージエンジンのアーキテクチャー」
セクション6.3.7「プラガブル認証」
セクション24.2.2「プラグイン API のコンポーネント」
セクション5.1.8.1「プラグインのインストールおよびアンインストール」
セクション24.2.4.3「プラグインライブラリのコンパイルおよびインストール」
セクション13.7.3.1「ユーザー定義関数のための CREATE FUNCTION 構文」
セクション24.3.2.5「ユーザー定義関数のコンパイルおよびインストール」
セクション24.3.2.6「ユーザー定義関数のセキュリティー上の予防措置」
セクション24.2.4.4「全文パーサープラグインの作成」
セクション6.1.3「攻撃者に対する MySQL のセキュアな状態の維持」
セクション17.3.8.2「準同期レプリケーションのインストールと構成」
セクション24.2.4.7「準同期レプリケーションプラグインの作成」
セクション24.2.4.8「監査プラグインの作成」
セクション6.3.12.1「監査ログプラグインのインストール」
認証プラグインの使用

port
セクションB.5.2.2「[ローカルの] MySQL サーバーに接続できません」
セクション5.1.4「サーバーシステム変数」

preload_buffer_size
セクション5.1.4「サーバーシステム変数」

profiling
セクション21.16「INFORMATION_SCHEMA PROFILING テーブル」
セクション1.4「MySQL 5.6 の新機能」
セクション13.7.5.31「SHOW PROFILE 構文」
セクション5.1.4「サーバーシステム変数」

profiling_history_size
セクション1.4「MySQL 5.6 の新機能」
セクション13.7.5.31「SHOW PROFILE 構文」
セクション5.1.4「サーバーシステム変数」

protocol_version
セクション5.1.4「サーバーシステム変数」

proxy_user
セクション5.1.4「サーバーシステム変数」
セクション6.3.9「プロキシユーザー」

pseudo_slave_mode
セクション5.1.4「サーバーシステム変数」

pseudo_thread_id
セクション5.1.4「サーバーシステム変数」
セクション17.4.1.34「レプリケーションと変数」
セクション5.2.4.3「混合形式のバイナリロギング形式」

Q

[索引の先頭]

Qcache_queries_in_cache
セクション8.9.3.3「クエリーキャッシュの構成」

query_alloc_block_size
セクション5.1.4「サーバーシステム変数」

query_cache_limit
セクション8.9.3.3「クエリーキャッシュの構成」
セクション5.1.4「サーバーシステム変数」

query_cache_min_res_unit
セクション8.9.3.3「クエリーキャッシュの構成」
セクション5.1.4「サーバーシステム変数」

query_cache_size
セクション8.9.3「MySQL クエリーキャッシュ」
セクション8.9.3.3「クエリーキャッシュの構成」
セクション5.1.2.1「サーバーのデフォルト値への変更」
セクション5.1.4「サーバーシステム変数」
セクション5.1.5「システム変数の使用」

query_cache_type
セクション13.2.9「SELECT 構文」
セクション8.9.3.2「クエリーキャッシュ SELECT オプション」
セクション8.9.3.3「クエリーキャッシュの構成」
セクション5.1.2.1「サーバーのデフォルト値への変更」
セクション5.1.4「サーバーシステム変数」
セクション5.1.6「サーバーステータス変数」

query_cache_wlock_invalidate
セクション5.1.4「サーバーシステム変数」

query_prealloc_size
セクション5.1.4「サーバーシステム変数」

R

[索引の先頭]

rand_seed
セクション5.1.4「サーバーシステム変数」

range_alloc_block_size
セクション5.1.4「サーバーシステム変数」

read_buffer_size
セクション8.11.4.1「MySQL のメモリーの使用方法」
セクション8.6.3「REPAIR TABLE ステートメントの速度」
セクション5.1.4「サーバーシステム変数」

read_only
セクション17.1.3.2「GTID を使用したレプリケーションのセットアップ」
セクション6.2.1「MySQL で提供される権限」
セクション13.7.1.7「SET PASSWORD 構文」
セクション13.3.1「START TRANSACTION、COMMIT、および ROLLBACK 構文」
セクション6.3.5「アカウントパスワードの割り当て」
セクション5.1.4「サーバーシステム変数」
セクション17.3.1.3「マスターまたはスレーブを読み取り専用にしてバックアップする」
セクション17.4.1.34「レプリケーションと変数」
セクション5.2.1「一般クエリーログおよびスロークエリーログの出力先の選択」
セクション8.12.5.2「一般的なスレッドの状態」

read_rnd_buffer_size
セクション8.2.1.13「Multi-Range Read の最適化」
セクション8.11.4.1「MySQL のメモリーの使用方法」
セクション8.2.1.15「ORDER BY の最適化」
セクション5.1.4「サーバーシステム変数」
セクション8.11.2「サーバーパラメータのチューニング」

relay_log
セクション17.1.4.3「レプリケーションスレーブのオプションと変数」

relay_log_basename
セクション1.4「MySQL 5.6 の新機能」
セクション17.1.4.3「レプリケーションスレーブのオプションと変数」

relay_log_index
セクション17.1.4.3「レプリケーションスレーブのオプションと変数」

relay_log_info_file
セクション17.1.4.3「レプリケーションスレーブのオプションと変数」

relay_log_info_repository
セクション17.1.4.3「レプリケーションスレーブのオプションと変数」

relay_log_purge
セクション13.4.2.1「CHANGE MASTER TO 構文」
セクション13.7.5.35「SHOW SLAVE STATUS 構文」
セクション5.1.4「サーバーシステム変数」

relay_log_recovery
セクション13.4.2.5「START SLAVE 構文」
セクション17.1.4.3「レプリケーションスレーブのオプションと変数」

relay_log_space_limit
セクション5.1.4「サーバーシステム変数」
セクション8.12.5.6「レプリケーションスレーブの I/O スレッド状態」

report_host
セクション5.1.4「サーバーシステム変数」

report_password
セクション5.1.4「サーバーシステム変数」

report_port
セクション5.1.4「サーバーシステム変数」

report_user
セクション5.1.4「サーバーシステム変数」

rpl_semi_sync_master_enabled
セクション5.1.4「サーバーシステム変数」
セクション17.3.8.2「準同期レプリケーションのインストールと構成」
セクション17.3.8.3「準同期レプリケーションモニタリング」
セクション17.3.8.1「準同期レプリケーション管理インタフェース」

rpl_semi_sync_master_timeout
セクション5.1.4「サーバーシステム変数」
セクション17.3.8.2「準同期レプリケーションのインストールと構成」
セクション17.3.8.1「準同期レプリケーション管理インタフェース」

rpl_semi_sync_master_trace_level
セクション5.1.4「サーバーシステム変数」

rpl_semi_sync_master_wait_no_slave
セクション5.1.4「サーバーシステム変数」

rpl_semi_sync_slave_enabled
セクション5.1.4「サーバーシステム変数」
セクション17.3.8.2「準同期レプリケーションのインストールと構成」
セクション17.3.8.1「準同期レプリケーション管理インタフェース」

rpl_semi_sync_slave_trace_level
セクション5.1.4「サーバーシステム変数」

rpl_stop_slave_timeout
セクション13.4.2.6「STOP SLAVE 構文」
セクション17.1.4.3「レプリケーションスレーブのオプションと変数」

S

[索引の先頭]

secure_auth
セクション6.3.8.3「4.1 よりも前のパスワードハッシュ方式と mysql_old_password プラグインからの移行」
セクション2.11.1.3「MySQL 5.5 から 5.6 へのアップグレード」
セクション1.4「MySQL 5.6 の新機能」
セクション6.1.2.4「MySQL でのパスワードハッシュ」
セクション5.1.2.1「サーバーのデフォルト値への変更」
セクション5.1.4「サーバーシステム変数」

secure_file_priv
セクション13.2.6「LOAD DATA INFILE 構文」
セクション6.2.1「MySQL で提供される権限」
セクション13.2.9.1「SELECT ... INTO 構文」
セクション5.1.4「サーバーシステム変数」
セクション6.1.3「攻撃者に対する MySQL のセキュアな状態の維持」
セクション12.5「文字列関数」

server_id
セクション18.3.4.3「MySQL Cluster のシステム変数」
セクション18.6.11「MySQL Cluster レプリケーションの競合解決」
セクション18.3.4.2「MySQL Cluster 用の MySQL Server オプション」
セクション4.6.8「mysqlbinlog — バイナリログファイルを処理するためのユーティリティー」
セクション13.7.5.35「SHOW SLAVE STATUS 構文」
セクション12.18「その他の関数」
セクション5.1.4「サーバーシステム変数」
セクション6.3.12.3「監査ログファイル」
セクション17.1.2.2「行ベースロギングおよびレプリケーションの使用」

server_id_bits
セクション18.3.4.3「MySQL Cluster のシステム変数」
セクション18.3.4.2「MySQL Cluster 用の MySQL Server オプション」

server_uuid
セクション17.1.3.1「GTID の概念」
セクション13.7.5.35「SHOW SLAVE STATUS 構文」
セクション13.4.2.5「START SLAVE 構文」
セクション17.1.4「レプリケーションおよびバイナリロギングのオプションと変数」

sha
セクション6.3.8.4「SHA-256 認証プラグイン」
セクション5.1.4「サーバーシステム変数」
セクション5.1.6「サーバーステータス変数」

shared_memory
セクション5.1.4「サーバーシステム変数」

shared_memory_base_name
セクション5.1.4「サーバーシステム変数」

simplified_binlog_gtid_recovery
セクション17.1.4.5「グローバルトランザクション ID のオプションと変数」

skip_external_locking
セクション5.1.4「サーバーシステム変数」
セクション8.10.5「外部ロック」

skip_name_resolve
セクション5.1.4「サーバーシステム変数」

skip_networking
セクション5.1.4「サーバーシステム変数」

skip_show_database
セクション5.1.3「サーバーコマンドオプション」
セクション5.1.4「サーバーシステム変数」

slave_allow_batching
セクション18.3.4.3「MySQL Cluster のシステム変数」
セクション18.6.6「MySQL Cluster レプリケーションの起動 (レプリケーションチャネルが 1 つ)」
セクション17.1.4.3「レプリケーションスレーブのオプションと変数」

slave_checkpoint_group
セクション17.1.4.3「レプリケーションスレーブのオプションと変数」

slave_checkpoint_period
セクション17.1.4.3「レプリケーションスレーブのオプションと変数」

slave_compressed_protocol
セクション17.1.4.3「レプリケーションスレーブのオプションと変数」

slave_exec_mode
セクション18.6.3「MySQL Cluster レプリケーションの既知の問題」
セクション17.4.1.21「レプリケーションと MEMORY テーブル」
セクション17.1.4.3「レプリケーションスレーブのオプションと変数」
セクション17.1.2.2「行ベースロギングおよびレプリケーションの使用」

slave_load_tmpdir
セクション13.2.6「LOAD DATA INFILE 構文」
セクション5.1.4「サーバーシステム変数」
セクション17.1.4.3「レプリケーションスレーブのオプションと変数」

slave_max_allowed_packet
セクション17.1.4.3「レプリケーションスレーブのオプションと変数」

slave_net_timeout
セクション13.4.2.1「CHANGE MASTER TO 構文」
セクション5.1.4「サーバーシステム変数」
セクション17.4.1.19「レプリケーションとマスターまたはスレーブシャットダウン」
セクション17.1.5.1「レプリケーションステータスの確認」
セクション8.12.5.6「レプリケーションスレーブの I/O スレッド状態」
セクション17.1.4.3「レプリケーションスレーブのオプションと変数」

slave_parallel_workers
セクション1.4「MySQL 5.6 の新機能」
セクション13.7.5.35「SHOW SLAVE STATUS 構文」
セクション13.4.2.5「START SLAVE 構文」
セクション17.1.4.3「レプリケーションスレーブのオプションと変数」

slave_pending_jobs_size_max
セクション17.1.4.3「レプリケーションスレーブのオプションと変数」

slave_rows_search_algorithms
セクション17.1.4.3「レプリケーションスレーブのオプションと変数」

slave_skip_errors
セクション17.1.4.3「レプリケーションスレーブのオプションと変数」

slave_sql_verify_checksum
MySQL 用語集
セクション5.2.4「バイナリログ」
セクション17.1.4.3「レプリケーションスレーブのオプションと変数」

slave_transaction_retries
セクション17.1.4.3「レプリケーションスレーブのオプションと変数」
セクション17.4.1.29「レプリケーション再試行とタイムアウト」

slave_type_conversions
セクション18.6.3「MySQL Cluster レプリケーションの既知の問題」
セクション17.1.4.3「レプリケーションスレーブのオプションと変数」

slow_launch_time
セクション5.1.4「サーバーシステム変数」
セクション5.1.6「サーバーステータス変数」

slow_query_log
セクション5.1.4「サーバーシステム変数」
セクション5.2.5「スロークエリーログ」
セクション5.2.1「一般クエリーログおよびスロークエリーログの出力先の選択」

slow_query_log_file
セクション5.1.4「サーバーシステム変数」
セクション5.2.5「スロークエリーログ」
セクション5.2.1「一般クエリーログおよびスロークエリーログの出力先の選択」

socket
セクション5.1.4「サーバーシステム変数」

sort_buffer_size
セクション8.2.1.19「LIMIT クエリーの最適化」
セクション7.6.3「MyISAM テーブルの修復方法」
セクション1.4「MySQL 5.6 の新機能」
セクション8.2.1.15「ORDER BY の最適化」
セクション5.1.4「サーバーシステム変数」
セクション5.1.6「サーバーステータス変数」

sql_auto_is_null
セクション5.1.4「サーバーシステム変数」
セクション5.2.4「バイナリログ」
セクション17.4.1.34「レプリケーションと変数」
セクション12.3.2「比較関数と演算子」
セクション5.2.4.3「混合形式のバイナリロギング形式」

sql_big_selects
セクション5.1.4「サーバーシステム変数」

sql_buffer_result
セクション5.1.4「サーバーシステム変数」

sql_log_bin
セクション18.1.6.1「MySQL Cluster での SQL 構文の不適合」
セクション18.1.6.8「MySQL Cluster に限定された問題」
セクション18.6.3「MySQL Cluster レプリケーションの既知の問題」
セクション13.4.1.3「SET sql_log_bin 構文」
セクション13.7.4「SET 構文」
セクション5.1.4「サーバーシステム変数」
セクション17.1.4.4「バイナリログのオプションと変数」
セクション17.1.4.1「レプリケーション、バイナリロギングオプション、および変数のリファレンス」
セクション17.4.3「レプリケーションセットアップをアップグレードする」

sql_log_off
MySQL 用語集
セクション5.1.4「サーバーシステム変数」
セクション17.1.4.4「バイナリログのオプションと変数」
セクション17.1.4.1「レプリケーション、バイナリロギングオプション、および変数のリファレンス」
セクション5.2.3「一般クエリーログ」
セクション5.2.1「一般クエリーログおよびスロークエリーログの出力先の選択」

SQL_MODE
セクション14.11.5「オンライン DDL の例」
セクション14.11.1「オンライン DDL の概要」

sql_mode
セクション13.1.11「CREATE EVENT 構文」
セクション13.1.15「CREATE PROCEDURE および CREATE FUNCTION 構文」
セクション13.1.19「CREATE TRIGGER 構文」
セクションB.5.5.2「DATE カラムの使用に関する問題」
セクション21.28「INFORMATION_SCHEMA VIEWS テーブル」
セクション13.2.6「LOAD DATA INFILE 構文」
セクション2.11.1.3「MySQL 5.5 から 5.6 へのアップグレード」
セクションA.11「MySQL 5.6 FAQ: MySQL の中国語、日本語、および韓国語の文字セット」
セクション1.7「MySQL の標準への準拠」
セクション4.4.3「mysql_install_db — MySQL データディレクトリの初期化」
セクション13.7.5.14「SHOW CREATE VIEW 構文」
セクション5.1.2.2「サンプルのデフォルトサーバー構成ファイルの使用」
セクション5.1.7「サーバー SQL モード」
セクション5.1.2.1「サーバーのデフォルト値への変更」
セクション5.1.4「サーバーシステム変数」
セクション5.1.5「システム変数の使用」
ハンドラ、カーソル、およびステートメントに対するシグナルの影響
セクション5.2.4「バイナリログ」
セクション17.4.1.34「レプリケーションと変数」
セクション12.20.3「式の処理」
セクション11.3.6「時間値での小数秒」
セクション5.2.4.3「混合形式のバイナリロギング形式」
セクション1.6「質問またはバグをレポートする方法」

sql_notes
セクション13.7.5.41「SHOW WARNINGS 構文」
セクション5.1.4「サーバーシステム変数」
診断領域関連のシステム変数

sql_quote_show_create
セクション13.7.5.8「SHOW CREATE DATABASE 構文」
セクション13.7.5.12「SHOW CREATE TABLE 構文」
セクション5.1.4「サーバーシステム変数」

sql_safe_updates
セクション5.1.4「サーバーシステム変数」

sql_select_limit
セクション5.1.4「サーバーシステム変数」

sql_slave_skip_counter
セクション17.1.3.4「GTID ベースレプリケーションの制約」
セクション13.4.2.4「SET GLOBAL sql_slave_skip_counter 構文」
セクション13.7.5.35「SHOW SLAVE STATUS 構文」
セクション17.1.4.5「グローバルトランザクション ID のオプションと変数」
セクション17.1.4.3「レプリケーションスレーブのオプションと変数」

sql_warnings
セクション5.1.4「サーバーシステム変数」

ssl_ca
セクション5.1.4「サーバーシステム変数」

ssl_capath
セクション5.1.4「サーバーシステム変数」

ssl_cert
セクション5.1.4「サーバーシステム変数」

ssl_cipher
セクション5.1.4「サーバーシステム変数」

ssl_crl
セクション5.1.4「サーバーシステム変数」

ssl_crlpath
セクション5.1.4「サーバーシステム変数」

ssl_key
セクション5.1.4「サーバーシステム変数」

storage_engine
セクション13.1.14「CREATE LOGFILE GROUP 構文」

セクション5.1.4「サーバーシステム変数」
セクション17.4.1.34「レプリケーションと変数」
セクション17.3.2「異なるマスターおよびスレーブストレージエンジンでレプリケーションを使用する」

stored_program_cache

セクション5.1.4「サーバーシステム変数」
セクション8.9.4「プリペアドステートメントおよびストアドプログラムのキャッシュ」

sync_binlog

セクション14.12「InnoDB の起動オプションおよびシステム変数」
セクション8.5.7「InnoDB ディスク I/O の最適化」
セクション14.2.1「MySQL および ACID モデル」
セクション5.2.4「バイナリログ」
セクション17.1.4.4「バイナリログのオプションと変数」
セクション17.4.1.19「レプリケーションとマスターまたはスレーブシャットダウン」

sync_frm

セクション5.1.4「サーバーシステム変数」

sync_master_info

セクション2.11.1.3「MySQL 5.5 から 5.6 へのアップグレード」
セクション5.1.2.1「サーバーのデフォルト値への変更」
セクション17.1.4.3「レプリケーションスレーブのオプションと変数」

sync_relay_log

セクション2.11.1.3「MySQL 5.5 から 5.6 へのアップグレード」
セクション5.1.2.1「サーバーのデフォルト値への変更」
セクション17.1.4.3「レプリケーションスレーブのオプションと変数」

sync_relay_log_info

セクション2.11.1.3「MySQL 5.5 から 5.6 へのアップグレード」
セクション5.1.2.1「サーバーのデフォルト値への変更」
セクション17.4.1.19「レプリケーションとマスターまたはスレーブシャットダウン」
セクション17.1.4.3「レプリケーションスレーブのオプションと変数」

system_time_zone

セクション10.6「MySQL Server でのタイムゾーンのサポート」
セクション5.1.3「サーバーコマンドオプション」
セクション5.1.4「サーバーシステム変数」

T

[索引の先頭]

table_definition_cache

セクション1.4「MySQL 5.6 の新機能」
セクション5.1.2.1「サーバーのデフォルト値への変更」
セクション5.1.4「サーバーシステム変数」

table_open_cache

セクションB.5.2.18「'File' が見つかりません、および同様のエラー」
セクション14.12「InnoDB の起動オプションおよびシステム変数」
セクション2.11.1.3「MySQL 5.5 から 5.6 へのアップグレード」
セクション1.4「MySQL 5.6 の新機能」
セクション8.4.3.1「MySQL でのテーブルのオープンとクローズの方法」
セクション8.11.4.1「MySQL のメモリーの使用方法」
セクション5.1.2.1「サーバーのデフォルト値への変更」
セクション5.1.3「サーバーコマンドオプション」
セクション5.1.4「サーバーシステム変数」
セクション5.1.6「サーバーステータス変数」
セクション8.11.2「サーバーパラメータのチューニング」
セクション8.12.5.2「一般的なスレッドの状態」

table_open_cache_instances

セクション5.1.4「サーバーシステム変数」
セクション5.1.6「サーバーステータス変数」

thread_cache_size

セクション24.4.1.4「gdb での mysqld のデバッグ」
セクション8.11.5.1「MySQL のクライアント接続のためのスレッドの使用方法」
セクション8.2.5「その他の最適化のヒント」
セクション5.1.2.1「サーバーのデフォルト値への変更」
セクション5.1.4「サーバーシステム変数」
セクション5.1.6「サーバーステータス変数」

thread_concurrency

セクション5.1.4「サーバーシステム変数」

thread_handling

セクション5.1.4「サーバーシステム変数」
セクション8.11.6.1「スレッドプールコンポーネントとインストール」

thread_pool_algorithm

セクション21.31.2「INFORMATION_SCHEMA TP_THREAD_GROUP_STATE テーブル」
セクション5.1.4「サーバーシステム変数」
セクション8.11.6.1「スレッドプールコンポーネントとインストール」

thread_pool_high_priority_connection

セクション5.1.4「サーバーシステム変数」
セクション8.11.6.1「スレッドプールコンポーネントとインストール」
セクション8.11.6.2「スレッドプール操作」

thread_pool_max_unused_threads

セクション5.1.4「サーバーシステム変数」
セクション8.11.6.1「スレッドプールコンポーネントとインストール」

thread_pool_prio_kickup_timer

セクション21.31.2「INFORMATION_SCHEMA TP_THREAD_GROUP_STATE テーブル」

セクション21.31.3「INFORMATION_SCHEMA TP_THREAD_GROUP_STATS テーブル」
セクション5.1.4「サーバーシステム変数」
セクション8.11.6.3「スレッドプールのチューニング」
セクション8.11.6.1「スレッドプールコンポーネントとインストール」
セクション8.11.6.2「スレッドプール操作」

thread_pool_size

セクション5.1.4「サーバーシステム変数」
セクション8.11.6.3「スレッドプールのチューニング」
セクション8.11.6.1「スレッドプールコンポーネントとインストール」
セクション8.11.6.2「スレッドプール操作」

thread_pool_stall_limit

セクション21.31.2「INFORMATION_SCHEMA TP_THREAD_GROUP_STATE テーブル」
セクション21.31.3「INFORMATION_SCHEMA TP_THREAD_GROUP_STATS テーブル」
セクション5.1.4「サーバーシステム変数」
セクション8.11.6.3「スレッドプールのチューニング」
セクション8.11.6.1「スレッドプールコンポーネントとインストール」
セクション8.11.6.2「スレッドプール操作」

thread_stack

セクション8.11.4.1「MySQL のメモリーの使用方法」
セクション5.1.4「サーバーシステム変数」
セクション20.2.1「ストアドルーチンの構文」

time_format

セクション1.4「MySQL 5.6 の新機能」
セクション5.1.4「サーバーシステム変数」

time_zone

セクション13.1.11「CREATE EVENT 構文」
セクション11.3.1「DATE、DATETIME、および TIMESTAMP 型」
セクション10.6「MySQL Server でのタイムゾーンのサポート」
セクション20.4.4「イベントメタデータ」
セクション5.1.3「サーバーコマンドオプション」
セクション5.1.4「サーバーシステム変数」
セクション17.4.1.34「レプリケーションと変数」
セクション12.7「日付および時間関数」
セクション5.2.4.3「混合形式のバイナリロギング形式」

timed_mutexes

セクション1.4「MySQL 5.6 の新機能」
セクション5.1.4「サーバーシステム変数」

timestamp

セクション15.8.3「FEDERATED ストレージエンジンの注記とヒント」
セクション5.1.4「サーバーシステム変数」
セクション17.4.1.34「レプリケーションと変数」
セクション5.2.4.3「混合形式のバイナリロギング形式」

tmp_table_size

セクション8.4.4「MySQL が内部一時テーブルを使用する仕組み」
セクション5.1.4「サーバーシステム変数」
セクション5.1.6「サーバーステータス変数」
セクションD.3「サーバー側のカーソルの制約」

tmpdir

セクション13.2.6「LOAD DATA INFILE 構文」
セクションB.5.4.4「MySQL が一時ファイルを格納する場所」
セクション2.9.4「MySQL ソース構成オプション」
セクション8.2.1.15「ORDER BY の最適化」
セクション14.11.2「オンライン DDL でのパフォーマンスと並列性に関する考慮事項」
セクション14.11.9「オンライン DDL の制限」
セクション14.11.6「オンライン DDL の実装の詳細」
セクション5.1.4「サーバーシステム変数」
セクション17.3.1.2「スレーブからローデータをバックアップする」
セクション7.2「データベースバックアップ方法」
セクションB.5.2.13「ファイルを作成/書き込みできない」
セクション17.1.4.3「レプリケーションスレーブのオプションと変数」

transaction_alloc_block_size

セクション5.1.4「サーバーシステム変数」

transaction_allow_batching

セクション18.3.4.3「MySQL Cluster のシステム変数」

transaction_prealloc_size

セクション5.1.4「サーバーシステム変数」

tx_isolation

セクション13.3.6「SET TRANSACTION 構文」
セクション5.1.3「サーバーコマンドオプション」
セクション5.1.4「サーバーシステム変数」

tx_read_only

セクション13.3.6「SET TRANSACTION 構文」
セクション5.1.3「サーバーコマンドオプション」
セクション5.1.4「サーバーシステム変数」

U

[索引の先頭]

unique_checks

セクション14.6.4「MyISAM から InnoDB へのテーブルの変換」
セクション5.1.4「サーバーシステム変数」
セクション5.2.4「バイナリログ」
セクション17.4.1.34「レプリケーションと変数」
セクション5.2.4.3「混合形式のバイナリロギング形式」

updatable_views_with_limit

セクション5.1.4「サーバーシステム変数」
セクション20.5.3「更新可能および挿入可能なビュー」

V

[索引の先頭]

診断領域関連のシステム変数

validate_password_dictionary_file
パスワード検証プラグインのオプションおよび変数

validate_password_length
パスワード検証プラグインのオプションおよび変数
セクション12.13「暗号化関数と圧縮関数」

validate_password_mixed_case_count
パスワード検証プラグインのオプションおよび変数

validate_password_number_count
パスワード検証プラグインのオプションおよび変数

validate_password_policy
セクション6.1.2.6「パスワード検証プラグイン」
パスワード検証プラグインのオプションおよび変数

validate_password_special_char_count
パスワード検証プラグインのオプションおよび変数

validate_user_plugins
セクション5.1.4「サーバーシステム変数」

version
セクション14.12「InnoDB の起動オプションおよびシステム変数」
セクション5.1.4「サーバーシステム変数」
セクション12.14「情報関数」
セクション6.3.12.3「監査ログファイル」

version_comment
セクション5.1.4「サーバーシステム変数」

version_compile_machine
セクション5.1.4「サーバーシステム変数」

version_compile_os
セクション5.1.4「サーバーシステム変数」

W

[索引の先頭]

wait_timeout
セクションB.5.2.9「MySQL サーバーが存在しなくなりました」
セクション23.7.7.53「mysql_real_connect()」
セクション5.1.4「サーバーシステム変数」
セクションB.5.2.11「通信エラーおよび中止された接続」

warning_count
セクション13.7.5.18「SHOW ERRORS 構文」
セクション13.7.5.41「SHOW WARNINGS 構文」
セクションB.1「エラー情報のソース」
セクション5.1.4「サーバーシステム変数」
ハンドラ、カーソル、およびステートメントに対するシグナルの影響

トランザクション分離レベルのインデックス

R | S

R

[索引の先頭]

READ COMMITTED

セクション14.2.8「InnoDB のさまざまな SQL ステートメントで設定されたロック」
セクション14.2.2「InnoDB のトランザクションモデルおよびロック」
セクション14.2.6「InnoDB のレコード、ギャップ、およびネクストキーロック」
セクション14.12「InnoDB の起動オプションおよびシステム変数」
セクション8.5.2「InnoDB トランザクション管理の最適化」
セクションA.10「MySQL 5.6 FAQ: MySQL Cluster」
セクションA.1「MySQL 5.6 FAQ: 全般」
セクション18.1.6.3「MySQL Cluster でのトランザクション処理に関する制限」
セクション18.1.5.3「NDB および InnoDB の機能使用のサマリー」
セクション18.1.5.1「NDB および InnoDB ストレージエンジンの違い」
セクション13.3.6「SET TRANSACTION 構文」
セクション14.2.11「デッドロックの対処方法」
セクション5.2.4.2「バイナリログ形式の設定」
セクション14.2.4「一貫性非ロック読み取り」

READ UNCOMMITTED

セクション14.18.2「InnoDB および memcached の統合のアーキテクチャー」
セクション14.2.2「InnoDB のトランザクションモデルおよびロック」
セクション14.12「InnoDB の起動オプションおよびシステム変数」
セクション18.1.6.3「MySQL Cluster でのトランザクション処理に関する制限」
セクション13.3.6「SET TRANSACTION 構文」
セクション5.2.4.2「バイナリログ形式の設定」
セクション14.18.5.6「ベースとなる InnoDB テーブルでの DML および DDL ステートメントの実行」
セクション14.2.4「一貫性非ロック読み取り」

READ-COMMITTED

セクション13.3.6「SET TRANSACTION 構文」
セクション5.1.3「サーバーコマンドオプション」

READ-UNCOMMITTED

セクション13.3.6「SET TRANSACTION 構文」
セクション5.1.3「サーバーコマンドオプション」

REPEATABLE READ

セクション14.18.5.4「InnoDB memcached プラグインのトランザクション動作の制御」
セクション14.2.2「InnoDB のトランザクションモデルおよびロック」
セクション14.2.6「InnoDB のレコード、ギャップ、およびネクストキーロック」
セクション14.12「InnoDB の起動オプションおよびシステム変数」
セクション8.5.2「InnoDB トランザクション管理の最適化」
セクション18.1.6.3「MySQL Cluster でのトランザクション処理に関する制限」
セクション13.3.6「SET TRANSACTION 構文」
セクション13.3.1「START TRANSACTION、COMMIT、および ROLLBACK 構文」
セクション13.3.7「XA トランザクション」
セクション14.2.4「一貫性非ロック読み取り」
セクション5.2.4.3「混合形式のバイナリロギング形式」

REPEATABLE-READ

セクション13.3.6「SET TRANSACTION 構文」
セクション5.1.3「サーバーコマンドオプション」
セクション5.1.4「サーバーシステム変数」

S

[索引の先頭]

SERIALIZABLE

セクション14.2.8「InnoDB のさまざまな SQL ステートメントで設定されたロック」
セクション14.2.2「InnoDB のトランザクションモデルおよびロック」
セクション14.12「InnoDB の起動オプションおよびシステム変数」
セクション18.1.6.3「MySQL Cluster でのトランザクション処理に関する制限」
セクション13.3.6「SET TRANSACTION 構文」
セクション13.3.1「START TRANSACTION、COMMIT、および ROLLBACK 構文」
セクション13.3.7「XA トランザクション」
セクション8.9.3.1「クエリーキャッシュの動作」
セクション5.1.3「サーバーコマンドオプション」
セクション14.2.4「一貫性非ロック読み取り」
セクション5.2.4.3「混合形式のバイナリロギング形式」